PROYECTO FINAL DE INGENIERÍA

SISTEMA INTELIGENTE DE ARMADO DE LISTAS MUSICALES DE REPRODUCCIÓN

Fernandez, Hernán Federico - LU 1016503

Ingeniería Informática

Lopez Melnyk, Maximiliano Leonel - LU 1023050

Ingeniería Informática

Tutor

Castro, Marcelo – marcecastro@uade.edu.ar

Cotutor

Cuadrado Estrebou, María Fernanda – mcuadrado@uade.edu.ar

Marzo 14, 2016

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

RESUMEN

El armado de una lista de reproducción resulta una tarea tediosa y molesta. Son incontables las veces que uno quiere sentarse a escuchar música durante su tiempo libre y se ve forzado a utilizar parte de ese tiempo en escoger qué canciones formarán parte de la lista musical.

Este trabajo de investigación evalúa la posibilidad de crear un modelo de algoritmo genético con el objetivo de resolver el problema planteado: crear una lista de reproducción musical. Durante el transcurso de este trabajo se investigaron formas de llevar a cabo la implementación del algoritmo con el fin de construir un prototipo que muestre los resultados obtenidos.

Para verificar la calidad de los resultados se realizó el entrenamiento del algoritmo para evaluar los operadores genéticos involucrados durante la ejecución, el tamaño de la población y la parametrización de aquellos factores que afectaban el rendimiento y resultado del procesamiento del algoritmo genético.

El trabajo realizado en este proyecto muestra que es posible construir un modelo de algoritmo genético que mejore la experiencia del usuario para la construcción de una lista de reproducción musical.

ABSTRACT

Creating a playlist can be a tedious and cumbersome task. There are countless instances where people wants to relax and listen to music in their spare time, and they are forced to use up some of that time picking the songs that will comprise said playlist.

This research project evaluates the possibility of using a genetic algorithm model aiming to solve the aforementioned situation in order to create a desired music playlist in short time.

The objective of this research was to find the way to implement the algorithm with the purpose of using it to construct a working prototype that can show the obtained results.

With the intention of verifying the quality of the results, the algorithm is trained to evaluate genetic operators involved in its execution, the population size and factors parametrization that affect the performance and result of the genetic algorithm implementation.

The result of this work proves the possibility of building a genetic algorithm model that improves the user experience while creating a music playlist.

CONTENIDOS

1 Introducción 7

1.1 Objetivos 8

2 Estado del arte 10

2.1 Introducción 10

2.2 Descripción 12

2.2.1 Aplicaciones basadas en estado de ánimo. 12

2.2.1.1 MoodAgent 12

2.2.1.2 Apple Watch 14

2.2.1.3 Mico 15

2.2.1.4 JukeFox 16

2.2.1.4.1 Smart Shuffle (Reproducción Aleatoria Inteligente) 17

2.2.1.4.2 Play Similar Songs 17

2.2.1.4.3 Tag Cloud 18

2.2.2 Aplicaciones de reproducción sin considerar estado de ánimo 18

2.2.2.1 Groove 18

2.2.2.2 Spotify 20

2.2.2.3 PlayerPro Music Player 20

2.2.3 Implementaciones y herramientas de desarrollo 21

2.2.3.1 Spotify Web API 21

2.2.3.1.1 Autenticación 22

2.2.3.1.2 Búsquedas 22

2.2.3.1.3 Reproducción de música 23

2.2.3.2 JAudioTagger 23

2.2.3.3 JID3 24

2.2.3.4 EchoPrint 24

2.2.4 Bancos de datos de música 26

2.2.4.1 MusicBrainz 26

2.2.4.2 EchoNest 28

2.3 Conclusión 30

3 Marco teórico 32

3.1 Introducción 32

3.2 Estudio de Metodologías A Utilizar. 32

3.2.1 Metodologías Ágiles 33

3.2.1.1 Manifiesto por el Desarrollo Ágil de Software 33

3.2.1.2 Principios de las Metodologías Ágiles 34

3.2.1.3 Técnicas 35

3.2.2 Conclusión 35

3.3 Estudio de Algoritmos. 36
3.3.1 Algoritmos Genéticos 36
3.3.1.1 Conceptos base de los Algoritmos Genéticos 38
3.3.1.2 Operadores Genéticos 40
3.3.1.2.1 Selección 41
3.3.1.2.1.1 Ruleta o Selección Proporcional 41
3.3.1.2.1.2 Selección por Ranking 42
3.3.1.2.1.3 Selección por Torneo 42
3.3.1.2.2 Cruce 42
3.3.1.2.3 Mutación 45
3.3.2 ¿Por qué Algoritmos Genéticos? - Ventajas y Desventajas - Aplicación al proyecto propuesto. 46
3.3.2.1 Ventajas 46
3.3.2.2 Limitaciones 47
3.3.2.3 Aplicación al proyecto propuesto 48

3.4 Modelo De Aplicación 49
3.4.1 Arquitectura de la Aplicación 49
3.4.1.1 Servidor de Aplicaciones 50
3.4.1.2 Cliente 51
3.4.2 Librerías / Aplicaciones A Utilizar. 52
3.4.2.1 Servidor 52
3.4.2.2 Cliente 53
3.4.2.3 Librerías para Algoritmos Genéticos 57

3.5 Conclusión 60

4 Desarrollo del Prototipo 61

4.1 Introducción 61

4.2 Desarrollo 61

4.2.1 Funcionamiento ¡Error! Marcador no definido.

4.2.2 Diagramas del Prototipo 61

4.2.3 Integración con Servicios 62
4.2.3.1 Spotify API 64
4.2.3.2 EchoNest 64

4.2.4 Parámetros de entrada 65
Artistas 65
Géneros 65
Estado de ánimo 65
Tiempo de reproducción 66
Tiempo de duración de canción 66
Recopilación de información 66

4.2.5 Algoritmo Genético 68
4.2.5.1 Introducción 68

4.2.5.2 Elementos 69
4.2.5.3 Configuración del AG 70
4.2.5.4 Función de ajuste 71
4.2.5.5 Método de cruzamiento 76
4.2.5.6 Método de selección 78
4.2.5.7 Evolución 78
4.2.5.8 Criterio de corte 79
4.2.5.9 Entrenamiento 81

4.2.6 User Stories 83

4.2.7 Sprints 84

4.2.8 Limitaciones Encontradas 86
4.2.8.1 JBoss 86
4.2.8.2 Spotify API 86
4.2.8.3 EchoNest 86

4.3 Conclusión 87

5 Conclusiones 89

5.1 Futuras Líneas de Investigación 90

6 Referencias Bibliográficas 92

7 Anexos 95

7.1 Anexo 1 – Técnidas de las Metodologías Ágiles 95

7.2 Anexo 2 – User Stories Identificadas 100

8 Tabla de Ilustraciones 107

9 Glosario 108

10 Abreviaturas 109

1 INTRODUCCIÓN

Cuando analizamos la forma en la que han evolucionado los reproductores de música desde la invención del casete hasta los reproductores inteligentes que están disponibles en los stores[1] de los diferentes dispositivos móviles podemos ver un crecimiento orientado a la comodidad del usuario. Ya sea a través del hardware, o por software, el principal objetivo de éstos siempre fue orientado a capacidades funcionales de una aplicación tales como armar listas, pasar canciones o comodidad para subir y bajar el volumen.

Hace ya varios años, desde la invención del iPod[2], que la tendencia tecnológica dejó de estar enfocada en crear dispositivos con nuevas funcionalidades a nivel hardware. Esta situación se debe gran parte a la invención de los Smartphones[3]. Estos dispositivos ya proveen esas características típicas de un buen reproductor de música, entre ellas tenemos el tamaño del dispositivo, la capacidad de almacenamiento y por supuesto la calidad de reproducción.

Es así como los esfuerzos por mejorar la experiencia del usuario al reproducir música pasaron de, hacer dispositivos más pequeños, más transportables y con mejor calidad de reproducción a crear softwares que permiten al usuario interactuar de forma más dinámica con sus dispositivos. Mientras que antes solo bastaba con iniciar la reproducción de un disco compacto pulsando el botón Play, donde solo podíamos reproducir doce (12) canciones aproximadamente, hoy en día podemos elegir nuestras listas de reproducciones basadas en información, moldearlas, y elegir sobre una cantidad ilimitada de canciones.

[1] Un store es un mercado de aplicaciones en el mundo de los teléfonos inteligentes. Allí se pueden descargar distintas herramientas para el teléfono.
[2] Un iPod es un reproductor de música digital, que con el tiempo fue evolucionando y hoy en día tiene las mismas funciones que un smartphone menos la de telefonía.
[3] Un Smartphone es un teléfono celular con todas las funcionalidades de un teléfono convencional pero que además tiene funciones similares a la de una minicomputadora con conexión a internet. Se lo llama teléfono inteligente.

A su vez, en los últimos 5 años, las tecnologías fueron creciendo exponencialmente, y una enorme cantidad de aplicaciones y juegos fueron surgiendo para los teléfonos. Con este mercado creciente de aplicaciones, es muy frecuente que los usuarios dispongan de poco espacio en el celular, ya que generalmente, lo ocupan la cantidad de aplicaciones instaladas en el mismo.

Este poco espacio que disponen los usuarios en el teléfono, provoca que la música no esté almacenada en el celular. Por esta razón, existe una tendencia actual, introducida por YouTube[4] con sus videos por streaming[5] que permite al usuario poder consumir archivos multimedia sin necesidad de tenerlos instalados en el dispositivo.

Así es como en junio del 2015, Spotify[6], una aplicación de reproducción de música por streaming, llegó a los 75 millones de usuarios, pasando a ser la número uno en cuestiones musicales.

A continuación se adjunta un vínculo a un video en la plataforma YouTube en el que muestra una breve descripción sobre el prototipo funcionando. Se muestran resultados a distintos parámetros de entrada.

https://www.youtube.com/watch?v=ZVRwGqHPY2A

1.1 OBJETIVOS

El objetivo general del proyecto final de ingeniería es diseñar un algoritmo que pueda generar una lista de reproducción dinámica de temas musicales. El mismo deberá recibir parámetros ingresados por un usuario externo (artistas, géneros, estados de

[4] Sitio web donde los usuarios pueden ver, subir y compartir videos. Se estima que la empresa recibe 300 horas de videos por minuto.
[5] El streaming es la distribución digital de multimedia (videos, canciones) a través de una red de computadoras, de manera que un usuario consume el producto de manera paralela mientras se descarga.
[6] Spotify es una aplicación empleada para la reproducción de música en internet por streaming. Existen dos versiones de la misma, una gratis o otra de pago. En junio de 2015, 20 millones de usuarios eran premium.

ánimo, duración de la lista de reproducción, duración promedio de la canción) y mediante la aplicación del algoritmo, deberá retornar una lista de reproducción musical.

El objetivo principal será optimizar la selección de la mejor lista de canciones posible, minimizando la necesidad de que el usuario reciba canciones que no desea escuchar.

Como objetivos específicos del trabajo nos proponemos realizar la investigación que permita, utilizando un banco de datos de música, generar listas de reproducción dinámicas. Este algoritmo deberá tener la inteligencia necesaria para que, a medida que es utilizado, aprenda más acerca de las preferencias del usuario y logre aumentar la aceptación del mismo con cada generación de las listas.

Finalmente nos proponemos realizar un prototipo que permita generar las listas de reproducción dinámicas, sin tener en cuenta la retroalimentación de datos a partir de la reproducción de música.

2 ESTADO DEL ARTE

2.1 INTRODUCCIÓN

Como parte de la investigación nos hemos propuesto identificar aquellas herramientas en el mercado orientadas a satisfacer las necesidades del usuario de forma similar a la que nos hemos propuesto hacer en nuestro proyecto final de ingeniería.

Para esto hemos realizado un estudio de mercado en el cual hemos encontrado un grupo de distintas tecnologías y estudios aplicados en diversos sectores e implementadas de distintas formas.

Para esta selección utilizamos como criterio buscar aquellas aplicaciones o estudios que hagan referencia a la utilización de modelos conceptuales y matemáticos que se apliquen al ámbito de la música y su categorización.

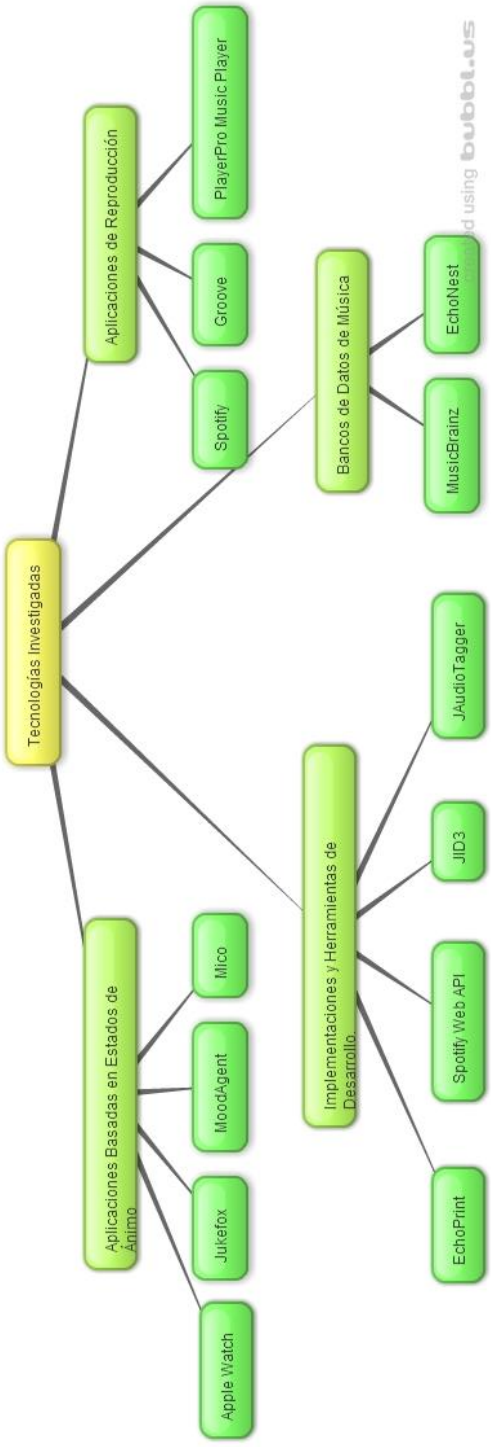

Figura 1: Estado del arte investigado.

2.2 DESCRIPCIÓN

En esta sección presentaremos un informe detallado de las aplicaciones y estudios que hemos encontrado pertinentes para el objetivo de este PFI.

2.2.1 Aplicaciones basadas en estado de ánimo.

Las aplicaciones basadas en estado de ánimo intentan de una forma u otra identificar el humor de la persona que escucha música y así sugerir otras canciones similares. Entre ellas analizamos las aplicaciones Moodagent (que utiliza un banco de datos con metadata[7]), HeartbeatsMUSIC (una idea a futuro de la empresa Apple con su SmartWatch), Jukefox (un reproductor musical para Android) y Mico (una aplicación creada por una empresa China llamada Neurowear).

2.2.1.1 MoodAgent

Moodagent era un banco de música online desarrollado por una empresa Danesa llamada Syntonetic[8] que clasificaba canciones en cuatro estados de ánimo: sensual, tierna, feliz y enojado (Strikingly, 2014). Una vez que tuvo suficientes canciones clasificadas, desarrolló una aplicación móvil para smartphones y tablets (ya discontinuada), en la que generaba listas de reproducción en donde un usuario seleccionaba una canción para escuchar y luego daba recomendaciones en base a la clasificación del estado de ánimo de la canción reproducida inicialmente. Otra opción que brindaba la aplicación era seleccionar un estado de ánimo en particular, y

[7] La metadata se refiere a datos que describen otros datos. En general son datos que describen el contenido informativo de un objeto.

[8] Syntonetic Media Solutions A/S es una pequeña empresa ubicada en Dinamarca. Creó la aplicación MoodAgent para generación de listas de reproducción en base a los estados de ánimo de las canciones.

ésta generaba una lista de reproducción con canciones dentro de la clasificación del mismo. (MoodAgent, 2015)

Más tarde, una vez que la aplicación fue discontinuada, Moodagent apareció como una sub-aplicación de Spotify donde realizaba las mismas acciones que su aplicación anterior.

Básicamente lo que hacían era proveer el banco de música con cinco billones de canciones clasificadas con metadata que básicamente contenía el nombre de la pista, artista, álbum, instrumentos, vocalización, Beats Per Minute (BPM)[9], estado de ánimo estandarizado para escuchar la canción, género de la canción y artistas similares. Éste banco de música era un servicio online que se podía consumir mediante licencia y se podía incluir en cualquier aplicación a desarrollar. (MoodAgent, 2015)

Son varias las empresas que utilizaron los servicios de Moodagent para poder realizar distintas actividades.

Como primer ejemplo, el 7 de Julio de 2014 el grupo de publicación de música de la empresa Universal Music contrató el servicio para realizar el etiquetado automático de las canciones para indexar su catálogo musical.

El objetivo era mejorar el motor de búsqueda para su staff de empleados y sus clientes para poder encontrar cualquier pista mediante pocas palabras claves.

Un segundo ejemplo es el de la empresa MixBerry Media[10] que obtenía la canción reproducida por un usuario (en cualquier reproductor) luego se conectaba con Moodagent, y generaba un anuncio de 30 segundos

[9] BPM son pulsaciones por minuto. Es la unidad de medida utilizada para medir el tempo de una canción.
[10] MixBerry Media es una empresa publicitaria que realiza anuncios de 30 segundos durante la carga de videos en transmisión, juegos o radio en internet.

de sus distintos anunciantes con una pista musical. Esta pista musical era similar (gracias a la conexión con Moodagent) a la canción reproducida por el usuario en primera instancia, y se colocaba como fondo del anuncio.

Por último, Spotify contrató sus servicios para realizar listas de reproducciones sugeridas o sugerir música similar en base a la escuchada anteriormente. Hoy en día éste servicio ya fue retirado de Spotify por la empresa Moodagent para relanzar su negocio con una nueva aplicación propia aún en desarrollo.

2.2.1.2 Apple Watch

Apple está constantemente buscando formas de motivar al usuario a realizar actividad física, una de las razones por la que lanza el Apple Watch, (más allá de ser, un producto de venta, ponerse al día en el mercado y generar ganancias exponenciales) es promover sus Aplicaciones de Salud para medir latidos del corazón y pulso.

Ahora, para que los usuarios utilicen sus aplicaciones de salud, los mismos deben realizar gimnasia, sino jamás conocerán la existencia de las nuevas aplicaciones nativas. Para ello, lo que estarían pensando es hacer una aplicación para sugerir canciones que motiven al usuario a realizar actividad física mediante la música. La aplicación obtendría el estado de ánimo del usuario para sugerir canciones nuevas.

En primer lugar, con la reciente compra de Beats[11], Apple se asegura tener una aplicación nativa para que los usuarios puedan escuchar música por streaming.

[11] Beats es una empresa musical dedicada a la venta de auriculares que cuenta con una aplicación de reproducción de música por streaming.

En segundo lugar, una vez que obtienen la aplicación, debían captar el estado de ánimo del usuario. Entonces aquí es cuando entra en juego el SmartWatch[12]. La empresa, utilizaría el sensor del reloj para captar los latidos del corazón y su pulso. Procesando esos datos, obtendrán el estado de ánimo de cualquier persona.

Así es como nace HeartBeatsMUSIC. Una nueva aplicación que Apple utilizaría para sugerir música al momento de realizar gimnasia e impulsar al usuario a salir a correr. Medirá los latidos del corazón y su pulso para obtener el estado de ánimo del usuario y sugerirá canciones en base a los parámetros obtenidos. Además, con el micrófono del reloj escuchará música que se reproduce en las cercanías del reloj y sugerirá agregar la canción reconocida a la lista de reproducción. Actúa similar a Shazam[13] que identifica el tempo de la canción para poder reconocer la pista en reproducción. (Spilka, 2015)

2.2.1.3 Mico

Mico es un proyecto de la empresa japonesa Neurowear[14], focalizado en realizar un dispositivo que puedan interactuar con el cerebro. Ésta iniciativa consiste en un dispositivo similar a un auricular que lee las ondas cerebrales del usuario y una aplicación para iPhone que interpreta las ondas del dispositivo para sugerir una canción de acuerdo a la lectura

[12] SmartWatch: reloj de pulsera con muchas más funcionalidades que uno convencional (inteligente).
[13] Shazam es una aplicación que escucha 30 segundos de cualquier canción, identifica el tempo y otros factores, y le indica al usuario qué canción es la que se reprodujo en esos 30 segundos.
[14] Neurowear es una empresa Japonesa situada en Tokio, dedicada a la creación de artefactos y productos basados en señales biológicas tal como ondas cerebrales, latidos del corazón, etc.

recibida y a una base de datos de música contenida en la aplicación. (Neurowear, 2012)

El usuario nunca selecciona una canción físicamente sino que la aplicación lo hace por él mediante la lectura de las ondas cerebrales efectuada por el dispositivo. En la interpretación de las mismas, se analiza el estado de ánimo del usuario y en base a él, se sugiere la canción. No se interpretan canciones en particular que el usuario esté pensando, sino que analiza la onda cerebral y verifica el estado de ánimo.

Figura 2: Usuario utilizando el dispositivo lector de las ondas cerebrales.

2.2.1.4 JukeFox

JukeFox surge de un proyecto nacido de una tesis de investigación del Swiss Federal Institute of Technology (ETH Zurich) que actualmente continúa en proceso. Es una aplicación construida para los dispositivos Android cuya función es proveer al usuario una nueva forma de reproducir su música sin tener que construir listas de reproducción genéricas.

La concepción de este software se basa en intentar que el usuario evite realizar consultas sobre su base de datos de música e intentar

descubrir qué música desea escuchar. Para esto JukeFox construye un mapa virtual en donde ordena las canciones según estilo, genero, artistas y las ubica en un espacio en el mapa. Cada nodo tiene otro nodo con una distancia equivalente a la similaridad que comparte con el mismo (Jukefox, 2011). Por lo que dos canciones de Bob Dylan que estén en el mismo álbum estarán separadas por una distancia menor que una canción de Jefferson Airplane. Así mismo una canción de David Bisbal estará muy lejos de estar próxima a estas dos mencionadas anteriormente.

Esta aplicación cuenta con las siguientes funcionalidades:

2.2.1.4.1 Smart Shuffle (Reproducción Aleatoria Inteligente)

JukeFox arma una lista de reproducción aleatoria en base al estado de humor del usuario. Esta aplicación identifica cuando un usuario pasa por alto canciones de un determinado género y trata de alejarse de la sección del mapa virtual previamente mencionado en donde se encuentran. De esta forma JukeFox minimiza la probabilidad que el usuario omita muchas canciones dentro de una lista de reproducción. (Jukefox, 2011)

2.2.1.4.2 Play Similar Songs

Play Similar Songs es un mecanismo para, a partir de un nodo del mapa (canción), buscar a través de sus nodos adyacentes canciones que sean similares a esta misma. Básicamente utiliza una semilla para armar la lista de reproducción. Lo innovador de este modelo es que permite realizar esta función de modo off line, en otras palabras, no necesita conexión a internet. (Jukefox, 2011)

2.2.1.4.3 Tag Cloud

Uno de los puntos más importantes de la investigación realizada por el instituto suizo es la customización de etiquetas sociales actualizadas constantemente. Este mismo es una base de datos ubicado en la nube que actualiza siempre que haya conexión las etiquetas sociales de las canciones que permiten la identificación de canciones en base a humor y parámetros subjetivos al usuario.

También permite al usuario crear sus propias listas de reproducción creando sus propias consultas a la base de datos de música filtrado por estas etiquetas sociales. (Jukefox, 2011)

2.2.2 Aplicaciones de reproducción sin considerar estado de ánimo

Así como hay aplicaciones que intentan sugerir canciones según parámetros subjetivos al usuario también están aquellas que utilizan algoritmos basados en los datos inherentes a la música tales como artista, título, álbum, tempo, género, etc. En esta sección analizaremos tres aplicaciones: Groove, Spotify y PlayerPro Music Player.

2.2.2.1 Groove

Groove es una aplicación de la empresa Zikera[15] y tiene como objetivo ayudar al usuario a re-descubrir canciones que pensaba perdidas en su vasta lista de música archivada en su teléfono. Groove no posee un banco de música propia sino que se encarga de categorizar a través de etiquetas la música que se encuentra en el dispositivo del usuario. La

[15] Fundada en el año 2009, su misión es proveer la mejor experiencia de reproducción musical a la gente. (Groove, 2014 a)

aplicación diseñada originalmente para dispositivos Apple. (Groove, 2014 a). El día 9 de febrero de 2016, Microsoft compró toda la empresa y se quedó con los derechos de la aplicación por lo que Groove ya fue retirado de los mercados de aplicaciones de Apple. (Sanchez, 2016).

El objetivo de Groove es similar a lo que nosotros deseamos conseguir en nuestro PFI, evitar que el usuario genere sus propias listas de reproducción. Groove se encarga de analizar los hábitos del usuario para crear listas instantáneas, a su vez intenta redescubrir música que el usuario no escucha hace un largo tiempo.

Groove parece estar diseñado para usuarios que poseen infinita cantidad de música en su dispositivo, de los cuales el usuario sólo utiliza muy poco.

Estas son las características principales de la aplicación:

- Ordena automáticamente la música del usuario y sugiere una variedad de mix basado en los hábitos de reproducción.
- Ayuda a redescubrir al usuario aquella música olvidada en su librería personal.
- Descarga las fotos de los álbumes.
- Organiza en etiquetas la música, similar a JukeFox. (Groove, 2014 a)

Actualmente, con la compra de Microsoft, Groove implementó un servicio en streaming pago y la probabilidad de subir música a la plataforma OneDrive[16] y utilizar Groove, de manera gratuita, para reproducir los archivos mediante la creación de las listas de reproducción. (Groove, 2016 b)

[16] Servicio gratuito de Microsoft utilizado para almacenar archivos en la nube.

2.2.2.2 Spotify

Spotify es, hoy en día, la aplicación más robusta hablando de funcionalidades para el usuario. Además de proveer su propio banco de música, uno de los más grandes del mundo, también provee de funcionalidades muy útiles. (Spotify, 2009 a)

Estos dos puntos sumado a que posee una interfaz disponible para toda clase de dispositivos (celulares, tablets, ordenadores, playstation, Xbox, etc) hacen que sea la aplicación número uno en el mercado para escuchar música.

Spotify posee diversas funcionalidades pero enumeramos solamente las que creemos más importantes y que se relacionan con el objetivo de nuestro trabajo:

1) Explorar: Esta herramienta provee miles y miles de listas de reproducción destinadas a un momento en particular, sea fiesta, aburrimiento o hasta estudio. (Spotify, 2009 a)
2) Running: Toma en cuenta el pulso del usuario y el tempo de las canciones para ayudar a maximizar el rendimiento a la hora de hacer ejercicio. (Spotify, 2009 a)

2.2.2.3 PlayerPro Music Player

Este reproductor se encuentra solamente disponible para los dispositivos con Android[17]. Es uno de los más descargados del store por poseer estabilidad en cuanto a las funcionalidades más comunes de un reproductor y su bajo costo. A los efectos del análisis que pueden ser relevantes para nuestro trabajo distinguimos a esta aplicación por dos funcionalidades:

[17] Sistema operativo desarrollado por google para dispositivos móviles.

1) El reproductor permite que el usuario realice sus propios etiquetados y categorice su música a gusto.
2) Permite armar listas de reproducción inteligentes basadas en características como álbum, artista, género, título, duración, calificación y otras no tan relevantes. (PlayerPro, 2009)

2.2.3 Implementaciones y herramientas de desarrollo

Hoy en día existen diversos formatos y tipos de archivos usados para almacenar música. Cada uno tiene su diferente forma de acceso y obtención de datos de los archivos. Una parte de este trabajo será analizar cómo obtener la mayor cantidad de información de las canciones musicales que el usuario tenga en su banco de datos de música para poder generar una lista de reproducción.

Para este objetivo analizaremos las distintas herramientas en el mercado que nos permiten explotar y explorar toda la información inherente a las canciones y al usuario. Las mismas son Spotify Web API, JAudioTagger, JID3 y EchoPrint.

2.2.3.1 Spotify Web API

Spotify provee una interfaz de desarrollo disponible para los desarrolladores que quieren armar sus propias aplicaciones consumiendo datos de su banco de música.

Esta API[18] es muy completa y fácil de utilizar, además provee muchas funcionalidades que pueden ser extremadamente útiles para nuestro propósito.

[18] Una API es la Interfaz de programación de la aplicación. Es el conjunto de funciones o métodos que ofrece una aplicación para que sean implementados en otro software.

Todas las funcionalidades que nos provee Spotify están disponibles para diversas plataformas. Entre ellas están Android SDK[19], iOS SDK, AppleScript y Web API. (Spotify, 2009 b)

La API está expuesta a través de servicios REST que proveen las siguientes funcionalidades:

2.2.3.1.1 Autenticación

Para utilizar la API de Spotify y los servicios es necesario utilizar su servicio de autenticación primero. Una vez que estamos autenticados podremos acceder a todos los servicios que la aplicación nos provee. La documentación para realizar la autenticación es muy completa y provee ejemplos muy fáciles de visualizar en https://developer.spotify.com/web-api/authorization-guide/ (Spotify, 2009 b).

2.2.3.1.2 Búsquedas

Con la API de Spotify podemos recolectar información acerca de canciones, artistas álbumes, etc. Podemos buscar listas de reproducción populares u obtener artistas relacionados por ejemplo.

En sí nos provee un enorme banco de datos de metadata muy fácil de utilizar y muy útil. Los servicios que provee la aplicación para realizar consultas están especificados en los siguientes enlaces:

https://developer.spotify.com/web-api/endpoint-reference/

https://developer.spotify.com/web-api/console/ (Spotify, 2009 b).

[19] SDK significa Software Development Kit o Conjunto de herramientas para el desarrollo de software, es un paquete de desarrollo que incluye funcionalidades ya empaquetadas.

2.2.3.1.3 Reproducción de música

Además de proveer las interfaces para acceder a su banco de datos, Spotify nos permite utilizar sus widgets[20] que nos proveen de la interfaz de usuario para reproducir música, visualizar listas, agregar canciones, etc. La documentación correspondiente se encuentra en: https://developer.spotify.com/technologies/widgets/ (Spotify, 2009 b).

2.2.3.2 JAudioTagger

Jaudiotagger nos permite acceder a los distintos tipos de archivo de música, escribir y leer su metadata. Es una librería JAVA que provee una interfaz genérica de más o menos 30 atributos, para la mayoría de los formatos de música los cuales soporta (mp4, m4a, mp4p, mp3, id3v1, id3v11, id3v2.2, id3v2.3, id3v2.4). Además soporta comentarios Vorbis (metadata del estilo Campo=Datos, case insensitive) y Flac (audio codec del estilo del mp3, más liviano).

Lo interesante de la librería es la integración que posee, entre sus atributos de metadata para escribir o leer, con MusicBrainz (tecnología próxima a analizar en el siguiente apartado del capítulo: Bancos de datos de música).

La librería permite escribir y leer una gran cantidad de atributos por cada archivo cargado, la gran mayoría de ellos son álbum, artista, título, año, sitio de Lyric, código de barras, BPM (tempo), compositor del tema, copyright, país, cover, número de disco / álbum, fecha, género, ISRC[21],

[20] Pequeño programa que permite ser incluido en otra página web.
[21] Código Internacional de Identificación de Grabaciones Sonoras y Audiovisuales.

idioma, estado de ánimo, ocasión, artista original, calidad, rating, país de release y track ID de Musicbrainz, por ejemplo. (JAudioTagger, 2004)

2.2.3.3 JID3

Esta es otra librería JAVA[22] que permite leer y escribir los metadatos de los archivos de música. No se encuentran diferencias con la librería anterior, y ésta es más acotada, en el sentido de que no maneja tantas etiquetas o clasificaciones en los archivos de música. Únicamente maneja los atributos de álbum, artista, comentarios, género, título y año.

Dentro de sus métodos, cuenta con la posibilidad de encriptar y desencriptar datos, lo que lo diferencia del resto. Pero comparándola con JAudioTagger, JID3, es bastante precaria. (Grebenc, 2003)

2.2.3.4 EchoPrint

EchoPrint es un servicio de identificación de música open source[23] (desarrollado por la empresa EchoNest), escucha las señales de la música e identifica qué canción es. Es una tecnología muy similar a Shazam, con la diferencia de que es de código abierto y gratis. Se puede utilizar el generador de código que proveen o su servidor. Además cuentan con un banco de música enorme que se puede consumir o descargar (varios archivos de formato json de 550MB cada uno). La única condición para utilizar con el banco de datos es que si es utilizado, y aparecen nuevas identificaciones de música, que esas nuevas sean compartidas con la comunidad. (Echonest, 2012)

[22] Tecnología que se usa para el desarrollo de aplicaciones
[23] Software distribuido libre y gratuito.

Nos provee la posibilidad de escanear un archivo (con al menos 20 segundos de señal de audio) y acceder a la metadata del mismo. El sistema cuenta con 3 etapas:

1) El generador de código: convierte audio en códigos. El mismo computa pares de {tiempo, hash} de una señal de audio usando procesamiento avanzado de señales que tiene en cuenta ruidos y modificaciones. A partir de una señal mono de 11Khz, se le pasa un filtro de "blanqueamiento" (o corrección), para luego una descomposición de subbandas (se transforma la señal en un número, con la frecuencia de 8 bandas). Esa descomposición es una búsqueda de puntos de partida que será "hasheado"[24] en un espacio de 20 bits y almacenado. (Echonest, 2012)

2) El servidor: almacena e indexa códigos. El servidor indexa cada punto de partida como un índice invertido, almacenando cada ocurrencia de la pista de ese comienzo, en una enorme lista, para su rápida búsqueda. El resultado devuelto es la cantidad de "overlaps[25]" de puntos de partida encontrados entre el query y cada pista en particular. (Echonest, 2012)

3) Los datos: alimentación a través de partners o usuarios.

[24] Técnica de encriptación computacional de datos.
[25] "overlaps" hace referencia al solapamiento entre pistas musicales.

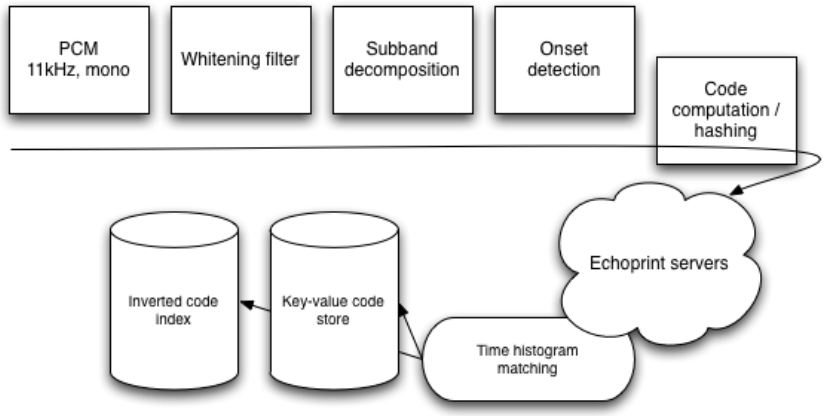

Figura 3: Explicación gráfica de la herramienta.

La empresa ya entrega el generador de código, todo el contenido para el servidor (y como instalarlo) y también provee de un cliente para pc y un cliente para dispositivos iOS. (Echonest, 2012)

2.2.4 Bancos de datos de música

Hay ciertas herramientas y APIs que simplifican la obtención de los datos que utilizaremos para construir nuestro algoritmo. Estas herramientas son bases de datos de canciones musicales ya categorizadas y con sus respectivas clasificaciones y etiquetas. Algunas proveen mejores servicios que otras y eso es lo que analizaremos, en únicamente, dos aplicaciones que resultaron ser las más interesantes: EchoNest y MusicBrainz.

2.2.4.1 MusicBrainz

La empresa MusicBrainz tiene varias herramientas relacionadas con la música y su tratamiento. En este apartado, trataremos únicamente a la base de datos.

La misma es una base de datos relacional, creada en PostgreSQL que contiene toda la metadata musical. La misma está creada con la misma

idea que wikipedia, para que cualquiera pueda contribuir con la misma y crecer. Los datos almacenados incluyen información dividida en varios grupos como artistas y grupos musicales, trabajos, releases, marcas, canciones, y todas las relaciones entre toda la información. Además guarda el historial de cambios que los usuarios introducen.

Aquí detallamos la información por cada grupo de información:

- Artistas: nombre, nombre de orden (una variante del nombre del artista a la hora de ordenar artistas), tipo, persona, grupo, orquesta, coro, personaje, otro, género, área (de donde es oriundo el artista), comienzos y fines, código IPI (es un ID de la base de datos CISAC para los derechos musicales), código ISNI (el identificador internacional del nombre estandarizado del artista), alias, MBID (MusicBrainz ID), comentarios de desambigüedad (Comentarios para diferenciar artistas con el mismo nombre) y anotaciones.
- Marcas: (trademarks): nombre, código de marca, tipo, discográfica, distribuidora, holding, sociedad de derecho, código IPI, código INSI, alias, comienzos y finales, país, MBID, comentarios de desambigüedad y anotaciones.
- Release (Este es un single o un álbum): título, artista, fecha, país, marca, número de catálogo, código de barras, estado (cuan oficial es el release), oficial, promocional, release pirata, pseudo-release, packaging, idioma, MBID, comentarios de desambigüedad, anotaciones y calidad de los datos (alto, default o bajo).
- Grupo de Releases (agrupa distintos releases en una única entrada).

- Trabajos (pueden ser poemas, novelas o ensayos que luego son grabados en un audio-libro o una oratoria).
- Relaciones (Son todas las relaciones posibles entre todos los grupos anteriores descritos).

La base de datos se puede descargar e instalar gratis, y por lo que se detalló anteriormente, son bastantes los datos con los que se cuenta y los que se pueden consumir. (MusicBrainz, 2000)

2.2.4.2 EchoNest

La empresa EchoNest es una de las líderes en la industria de inteligencia musical, dando a los desarrolladores una amplia variedad de conocimiento musical. La empresa ofrece un enorme paquete de datos musicales y servicios para crear excelentes aplicaciones (Echonest, 2012). Recientemente la empresa llegó a un acuerdo con Spotify por lo que su API está relacionada y cada pista buscada contiene el ID en Spotify, lo que colabora con solventar algunas limitaciones que puede llegar a tener esta API y se solucionan obteniendo los datos desde la API de Spotify.

El primer paso es registrar una API key, para obtener acceso a sus servicios. Luego ya se puede comenzar a realizar pedidos y obtener respuestas. Este es un servicio REST[26], el cual se accede mediante una URL[27] con la clave de la aplicación y el pedido deseado. Se debe

[26] Transferencia de Estado Representacional, es un estilo de arquitectura de software para sistemas distribuidos.
[27] Localizador de recursos uniformes, son cadenas de caracteres que designan recursos en una red. Comúnmente se los conoce como las direcciones web con las que los usuarios acceden a distintos sitios (ei. www.google.com)

especificar además el formato de respuesta, EchoNest puede devolver 3 distintos tipos de archivos: JSON, XML o JSONP[28]. (Echonest, 2012)

Los pedidos que se pueden realizar a la API son, obtener las canciones de un artista, obtener las canciones de un género en particular, una canción y un álbum en particular, además se puede filtrar la búsqueda por un estado de ánimo particular. (Echonest, 2012)

Los datos que se pueden consultar dentro de cada grupo de datos son:

- Artistas
    - Biografía.
    - Blogs.
    - Familiaridad.
    - Ranking de popularidad (Hotness).
    - Imágenes.
    - Lista de Géneros.
    - Lista de Términos (Estilo del artista o estado de ánimo recomendado para escuchar al artista).
    - Novedades.
    - Perfil.
    - Valoración del usuario.
    - Canciones.
    - Similares.
    - Top de popularidad.
    - Top de términos.
    - Twitter.

[28] JSON, XML, JSONP son estructuras de archivos utilizadas para almacenar datos de forma legible.

- URLs.
- Videos.

- Géneros.
  - Artistas.
  - Lista de todos los géneros posibles.
  - Perfil (descripción del estilo musical).
  - Similares (con un porcentaje de similaridad).
- Canción.
  - Resumen de la canción.
  - Artista.
  - Popularidad (Hotness).
  - Ranking de la canción.
  - Tipo de canción.
  - Estilo.
  - Estado de ánimo.
  - Tempo.
  - Duración.

Las respuestas que se desean se especifican en la URL del pedido y la API devuelve, en el formato especificado, un archivo con los datos buscados.

2.3 CONCLUSIÓN

Como análisis del estado del arte, hemos abarcado un espectro bastante amplio sobre las herramientas que existen hoy en día para tener un mayor conocimiento sobre el tema en cuestión. Con lo estudiado hasta el momento, tenemos varias tecnologías para utilizar (o para basarnos en ellas) a la hora de realizar las listas de reproducción musical.

El proyecto puede ser encarado desde distintos puntos de vista y con distintas herramientas. Para nuestro proyecto hemos decidido utilizar, en base a la investigación realizada, las distintas tecnologías:

- Spotify API: Seleccionamos esta tecnología, por la inmensa cantidad de funcionalidades que provee con respecto a las otras tecnologías analizadas, por la documentación de la API en el sitio oficial y por la cantidad de información con la que se cuenta en la web si surge algún inconveniente. Además nos provee de un Widget HTML[29] para el armado del reproductor de la lista.
- EchoNest: Seleccionamos esta tecnología para consumir como banco de datos, ya que es la única oficialmente integrada con Spotify y provee todos los datos necesarios para el armado de nuestro algoritmo.

[29] HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de paginas web.

3 MARCO TEÓRICO

3.1 INTRODUCCIÓN

En esta sección abordaremos todo el estudio del desarrollo del prototipo, desde la metodología a utilizar, los algoritmos base, la estructura de la aplicación y finalmente las conexiones a los servicios web para obtener las canciones necesarias para el uso del prototipo.

3.2 ESTUDIO DE METODOLOGÍAS A UTILIZAR.

Durante la carrera de Ingeniería Informática, hemos conocido diversos paradigmas de programación o metodologías de desarrollo y sus técnicas. Algunos ejemplos son:

Tabla I: Metodologías de Desarrollo

Paradigma	*Autores*	*Metodología*	*Técnicas*
Estructurado	• Gane y Sarson • Yourdon • Jaime Cabrera	Estructurada	• Diagrama de Flujo de Datos • Diagrama de Entidad Relación • Diagrama de Estados y Transiciones • Carta de estructura
Objetos	• Jacobson • Rumbaugh • Booch • Gustavo Rossi	Proceso Unificado	• UML • Casos de Uso • Diagrama de Clases • Diagramas de Secuencia • Diagrama de Colaboración

Ágiles	• Cockburn • KentBeck • KentSchwaber • Poppendieck • J. Gabardini • Salías • T. Wallet	Sin Unificar Manifiesto ágil (valores y principios)	• Scrum • Lean - Kanban • XP: Extreme Programming • TDD: Test Driven Development • Integración Continua • User Stories • Crystal Methodologies

Fuente: Material de cursada materia Seminario de Integración Profesional II, clase Ciclo de Vida Metodologías, cátedra: Bibiana Rossi.

Dada la experiencia con la que contamos gracias al ámbito laboral y al marco teórico brindado durante la carrera, decidimos trabajar con las Metodologías Ágiles.

3.2.1 Metodologías Ágiles

Durante los años 90, un grupo de personas comenzaron a aplicar métodos y técnicas de desarrollo de software que se adaptasen más fácilmente al nuevo mundo de los negocios que estaba surgiendo (un mercado cambiante, con clientes más fluctuantes y ciclos de productos más cortos, donde se tiene un foco en la calidad y en la satisfacción del cliente).

Los métodos y técnicas propuestas difieren, aunque comparten principios y valores que se agruparon bajo el nombre de "Metodologías Ágiles".

3.2.1.1 Manifiesto por el Desarrollo Ágil de Software

"Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

- *Individuos e interacciones sobre procesos y herramientas.*

- *Software funcionando sobre documentación extensiva.*
- *Colaboración con el cliente sobre negociación contractual.*
- *Respuesta ante el cambio sobre seguir un plan.*

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda" (Beck *et al*, 2001).

3.2.1.2 Principios de las Metodologías Ágiles

1) *"Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.*
2) *Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.*
3) *Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
4) *Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.*
5) *Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.*
6) *El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
7) *El software funcionando es la medida principal de progreso.*

8) Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9) La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10) La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11) Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12) A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia."

(Beck *et al*, 2001)

3.2.1.3 Técnicas

En el Anexo Nº 1 se estudian las distintas técnicas evaluadas para la gestión y desarrollo del prototipo. Las evaluadas fueron SCRUM, Extreme Programming (XP) y Test Driven Development (TDD).

3.2.2 Conclusión

Dada la experiencia laboral, y las limitaciones encontradas en XP y en TDD, decidimos trabajar con la técnica de SCRUM. Como herramienta de documentación de la metodología, utilizaremos Vivify Scrum web app, en su versión free. Una herramienta de tracking que permite documentar cada tarea, añadirla al backlog y a los respectivos sprints.

Otra razón que nos impulsó a seleccionar la técnica de SCRUM es que las metodologías ágiles para el desarrollo no están muy relacionadas con las

técnicas de Inteligencia Artificial pero como la técnica seleccionada, según la teoría, aplica a cualquier industria y a cualquier actividad decidimos emplearla. Además no utilizaremos la técnica para el desarrollo en sí del prototipo, ya que la misma es utilizada para la gestión de nuestras actividades en el proyecto.

3.3 ESTUDIO DE ALGORITMOS.

Dentro del mundo los algoritmos convencionales y la inteligencia artificial estudiados durante la carrera, enfocaremos el desarrollo del prototipo desde el área de la Inteligencia Artificial. Más precisamente dentro del área encontramos, de gran utilidad, la rama de los Algoritmos Genéticos para la búsqueda y optimización de la selección de las canciones dentro de una lista de reproducción.

3.3.1 Algoritmos Genéticos

Los algoritmos genéticos (AGs) fueron concebidos como métodos de optimización basados en las teorías evolutivas biológicas de Darwin para la resolución de problemas de la vida real (Goldberg, 1989). Así como los organismos vivientes sufren un proceso de cambio a través de su vida con el objetivo de evolucionar hacia una mejor versión de sí mismos también lo hacen los algoritmos genéticos utilizando los mecanismos de reproducción, mutación, competición y selección natural. (Holland, 1992)

En la naturaleza, los individuos de una población deben compartir los recursos disponibles, tales como comida, tierra, bebida, incluso éstos mismos deben competir por la búsqueda de un compañero con el cual puedan generar descendencia. (Arroyo Apaza, 2013)

Este comportamiento lleva al problema en el que no todos los individuos de una población logran reproducirse, sólo los más aptos y aquellos con mayor probabilidad de sobrevivir serán los que produzcan una mayor cantidad de descendientes. Mientras que los que no estén en condiciones de generar una

mejor versión de sí mismos serán descartados o tendrán menor posibilidad de reproducción. (Arroyo Apaza, 2013)

Esto significa que sólo los individuos cuyos genes están mejor adaptados propagarán sus características provenientes de distintas generaciones a través de las subsiguientes.

De esta manera es como una especie logra evolucionar, con el objetivo de lograr una mejor adaptación al ambiente en el que viven.

Más formalmente, y siguiendo la definición dada por Goldberg, *"Los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas"* (Goldberg, 1989)

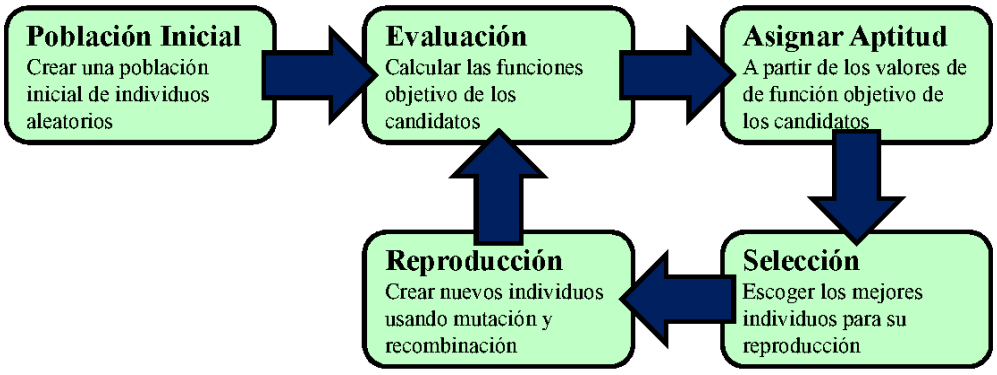

Figura 4: Ciclo básico de un algoritmo genético. (ProyectoLatin, 2013)

El poder de los AGs proviene del hecho de que se trata de una técnica robusta y pueden tratar una gran variedad de problemas de diferentes áreas. Si bien no se garantiza que los AGs encuentren la solución óptima del problema en cuestión, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de

optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al AG, tanto en rapidez como en eficacia. (Rudoph, 1994)

Los AGs están compuestos básicamente por tres fases. La primera consiste en la creación de una población inicial que sea representativa del objeto de estudio. La misma debe poseer un tamaño adecuado para el éxito del algoritmo, esto quiere decir, que si la población es muy pequeña puede que nunca se encuentre una solución óptima y que si esta es muy grande, el rendimiento del algoritmo sea pobre o que las soluciones encontradas converjan a un óptimo local, desestimando el óptimo global. (Arroyo Apaza, 2013)

La segunda fase consiste en la selección de los individuos que formarán parte del proceso de reproducción. En esta fase, se evalúa la adaptación de cada individuo en relación a sus pares, descartando aquellos que no son adecuados para generar descendencia. A este concepto se lo conoce como elitismo, los procesos de selección deben controlarlo para que aquellos individuos que no sean considerados óptimos, tengan la oportunidad de generar descendencia. Así lograr una mayor diversidad en generaciones posteriores. (Arroyo Apaza, 2013)

La tercera fase se corresponde a los procesos que involucran la manipulación de los individuos seleccionados utilizando los operadores genéticos de cruce y mutación.

3.3.1.1 Conceptos base de los Algoritmos Genéticos

A raíz de lo explicado anteriormente, se procederá a dar una breve explicación de los componentes involucrados en los AGs.

Individuo: Un individuo representa una posible solución, dentro de la población, al problema que se intenta resolver mediante el algoritmo genético. (ProyectoLatin, 2013)

Cromosoma: Se refiere a una estructura de datos que contiene una serie de parámetros de diseño también llamados genes. Intenta codificar la información de cada solución en una estructura de datos, tales como cadenas de bits, números reales, etc. (ProyectoLatin, 2013)

Gen: Es una subsección de un cromosoma, que codifica el valor de uno de los parámetros del problema. (ProyectoLatin, 2013)

Población: Es el conjunto de cromosomas que serán tratados en el proceso evolutivo. (ProyectoLatin, 2013)

Generación: Es la representación de la evolución de los cromosomas en una iteración. (ProyectoLatin, 2013)

Genotipo: Es la codificación utilizada en el cromosoma, para los parámetros del problema a solucionar. El conjunto de posibles valores del genotipo conforma el espacio de búsqueda del algoritmo. (ProyectoLatin, 2013)

Fenotipo: Es el resultado de decodificar el genotipo. (ProyectoLatin, 2013)

Función de evaluación: Es una medida que indica la calidad del individuo en el ambiente. (ProyectoLatin, 2013)

Alelo: Es la representación de cada valor posible que un gen pueda adquirir. (ProyectoLatin, 2013)

Todos los organismos vivos están constituidos por células, y cada célula contiene uno o más cromosomas (cadenas de ADN), que le sirven como una especie de "plano" al organismo. Un cromosoma puede ser conceptualmente dividido en genes, entonces se puede decir que, un gen se codifica como si fuera un rasgo, como puede ser el color de ojos (fenotipo). Cada gen se encuentra en una posición particular del cromosoma, y está formado por alelos.

Un individuo es una solución potencial al problema que se trata. Cada individuo contiene un cromosoma. A un conjunto de individuos se le denomina población. El fitness de un individuo es el resultado de la función de evaluación e indica qué tan bueno es el individuo (es decir, la solución al problema) con respecto a los demás.

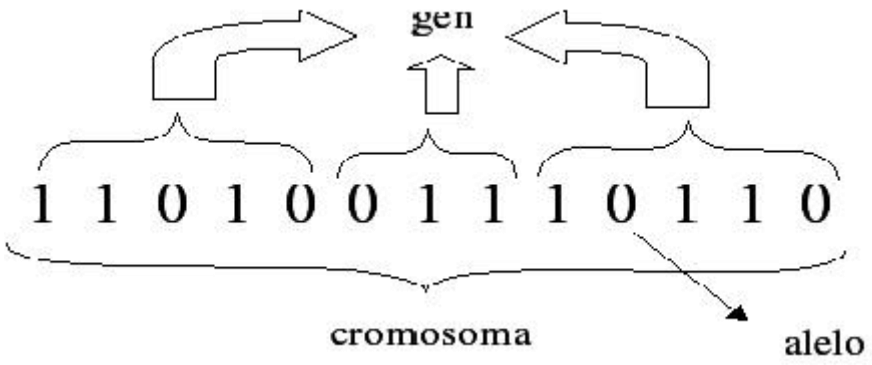

Figura 5: Individuo Genético Binario (Gestal Pose, 2007)

La función de evaluación o de fitness de un problema es realmente la función que se desea optimizar. Su diseño es, junto con el del genotipo, una de las características más importantes a la hora de encontrar la mejor solución a un problema. (Arroyo Apaza, 2013)

La función de evaluación (fitness) es la que permite valorar la aptitud de los individuos y debe tomar siempre valores positivos.

Una vez definido el sistema de codificación a emplear se verá cómo actúan los operadores básicos de selección, cruce y mutación sobre este código.

3.3.1.2 Operadores Genéticos

Para el paso de una generación a la siguiente se aplican una serie de operadores genéticos: selección, cruce y mutación.

3.3.1.2.1 Selección

El proceso de selección sirve para escoger a los individuos de la población mejor adaptados, para que actúen de progenitores de la siguiente generación. Este operador es el encargado de transmitir y conservar aquellas características de las soluciones que se consideran valiosas a lo largo de las generaciones.

Es necesario que en todo algoritmo de selección, se le permita, con un bajo margen probabilístico, al individuo menos apto, tener la posibilidad de poder ser padre de una futura generación.

En los algoritmos genéticos, la selección es un conjunto de reglas que sirven para elegir a los progenitores de la siguiente generación. Estos progenitores se reproducirán y generarán descendencia.

Algunas técnicas de las cuales se dispone se detallan a continuación.

3.3.1.2.1.1 Ruleta o Selección Proporcional

Propuesto por DeJong, con este método la probabilidad que tiene un individuo de reproducirse es proporcional a su valor de función de evaluación, de tal forma que la suma de todos los porcentajes, sea la unidad. Los mejores individuos, obtendrán una porción mayor en la ruleta que los peores. El comportamiento es similar al de una ruleta, donde se sitúan, generalmente, en el inicio de la ruleta, los de mayor ajuste.

Luego, basta con seleccionar un número aleatorio entre [0...1] y devolver el individuo situado en esa posición. Tiene la ventaja de que no es posible seleccionar dos veces el mismo elemento. (Arroyo Apaza, 2013)

3.3.1.2.1.2 Selección por Ranking

Esta selección se efectúa mediante la ordenación de la población de acuerdo a su fitness y entonces las probabilidades de selección se asignan de acuerdo a su ordenamiento.

Puede ser computacionalmente costosa por la necesidad de ordenar la población. (Arroyo Apaza, 2013)

3.3.1.2.1.3 Selección por Torneo

Consiste en seleccionar un grupo de "n" individuos al azar y se genera un número aleatorio entre 0 y 1. Si este número es mayor a un cierto umbral "k" (generalmente mayor a 0.7), se selecciona al individuo con peor adaptación. Las ventajas de esta técnica son que permite un cierto grado de "elitismo" ya que el mejor nunca va a morir, y los mejores tienen más probabilidad de reproducirse; La velocidad de aplicación; Y la capacidad de prevenir la convergencia prematura.

Cuando participan muchos individuos en cada torneo, la presión de selección es elevada y los peores individuos tienen muy pocas oportunidades de reproducción, pero cuando el tamaño es reducido (n=2), los peores individuos, tienen más oportunidades de ser seleccionados. (Arroyo Apaza, 2013)

3.3.1.2.2 Cruce

El operador de cruce permite realizar una exploración de toda la información almacenada hasta el momento en la población y combinarla para crear mejores individuos (Gestal Pose, 2007). Existen diversos métodos de cruce, pero los más utilizados son los siguientes:

- Cruce de un punto: los dos individuos seleccionados para jugar el papel de padres, son recombinados por medio de la selección de un punto de corte, para posteriormente intercambiar las secciones que se encuentran a la izquierda de dicho punto. (Gestal Pose, 2007)

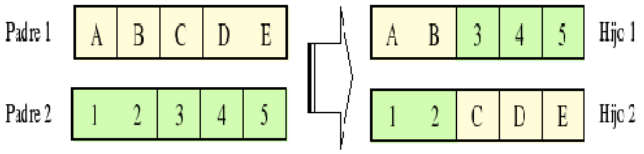

Figura 6: Cruce de un punto (Gestal Pose, 2007)

- Cruce de n puntos: este tipo de cruce es similar al anterior pero se seleccionan varias posiciones (n) y se intercambian los genes a ambos lados de estas posiciones. (Gestal Pose, 2007)

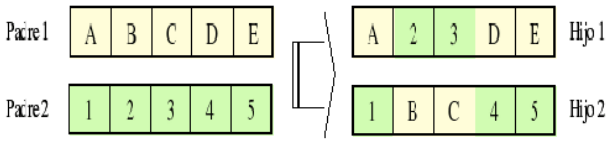

Figura 7: Cruce de n puntos (Gestal Pose, 2007)

- Cruce uniforme: en este tipo de cruce se selecciona aleatoriamente la cantidad de puntos que se van a utilizar para el cruce. De esta forma, y de manera análoga al anterior cruce, se irán intercambiando los genes para formar los dos nuevos

hijos. (Gestal Pose, 2007)

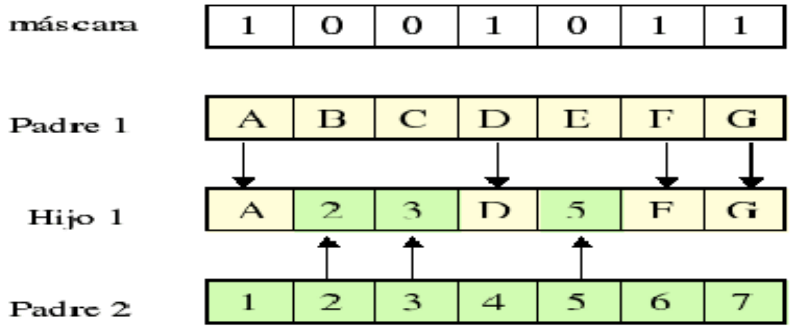

Figura 8: Cruce uniforme(Gestal Pose, 2007)

- Cruces específicos de codificaciones no binarias: Para este tipo de codificación se pueden definir, además de los anteriores, otros tipos de operadores de cruce:
    - Media: el gen de la descendencia toma el valor medio de los genes de los padres. Tiene la desventaja de que únicamente se genera un descendiente en el cruce de dos padres. (Gestal Pose, 2007)
    - Media geométrica: cada gen de la descendencia toma como valor la raíz cuadrada del producto de los genes de los padres. Presenta el problema añadido de qué signo dar al resultado si los padres tienen signos diferentes. (Gestal Pose, 2007)
    - Extensión: se toma la diferencia existente entre los genes situados en las mismas posiciones de los padres y se suma el valor más alto o se resta del valor más bajo. Solventa el problema de generar un único descendiente. (Gestal Pose, 2007)

3.3.1.2.3 Mutación

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en los individuos de la población. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, el operador de mutación es el responsable del aumento o reducción del espacio de búsqueda dentro del algoritmo genético y del fomento de la variabilidad genética de los individuos de la población. (Gestal Pose, 2007)

El objetivo del operador de mutación es producir nuevas soluciones a partir de la modificación de un cierto número de genes de una solución existente, con la intención de fomentar la variabilidad dentro de la población.

Existen varios métodos para aplicar la mutación a los individuos de una población, pero el más comúnmente utilizado es el de mutar un porcentaje de los genes totales de la población. (Gestal Pose, 2007)

Este porcentaje de genes a mutar se puede seleccionar de dos maneras, de forma fija, especificando el mismo porcentaje de mutación a todas las generaciones del algoritmo genético y de forma variable, es decir, modificando el porcentaje de mutación de una generación a otra, por ejemplo reduciéndolo. De esta manera, se consigue hacer una búsqueda más amplia y global al principio e ir reduciéndola en las siguientes generaciones. (Gestal Pose, 2007)

3.3.2 ¿Por qué Algoritmos Genéticos? - Ventajas y Desventajas - Aplicación al proyecto propuesto.

La razón de la utilización de los Algoritmos Genéticos es que estos son un método global y robusto de búsqueda de las soluciones de problemas. Se pueden encontrar soluciones aproximadas a problemas complejos.

Los algoritmos genéticos trabajan con un conjunto de puntos, no con uno y su entorno. Utilizan un subconjunto del espacio total para obtener información sobre el universo de búsqueda, a través de las distintas evaluaciones y resultados de la función a optimizar.

Además, utilizan operadores probabilísticos, en vez de los típicos operadores determinísticos de las técnicas convencionales. Lo que nos permite llegar a una solución óptima y siempre distinta.

Finalmente, cuando se usan para problemas de optimización, resultan menos afectados por los máximos locales que las técnicas tradicionales.

3.3.2.1 Ventajas

- La mayoría de los otros algoritmos son en serie y sólo pueden explorar el espectro de soluciones hacia una única dirección a la vez, y si la solución no es la deseada, se debe abandonar el trabajo y comenzar nuevamente. En cambio, los algoritmos genéticos, son intrínsecamente paralelos, tienen descendencia múltiple, es decir, pueden explorar el espacio de soluciones en múltiples direcciones a la vez. Si un camino resulta ser un callejón sin salida, lo desechan y continúan. (Arranz de la Peña y Parra Truyol, 2005) Este paralelismo les permite evaluar varios esquemas a la vez, por lo que funciona bastante bien con problemas que tienen un espectro de soluciones muy grande.

- El algoritmo genético puede dirigirse hacia el espacio con los individuos más aptos y encontrar el mejor de ese grupo.
- Capacidad de escapar a los máximos locales y converger en una solución frecuente. (Arranz de la Peña y Parra Truyol, 2005)
- Muchos de los problemas de la vida real no pueden definirse en términos de un único valor a optimizar, sino que deben expresarse en términos de varios parámetros simultáneamente. Los algoritmos genéticos tienen esta habilidad de manipular múltiples objetivos a la vez. (Arranz de la Peña y Parra Truyol, 2005)
- Los algoritmos genéticos, no saben absolutamente nada del problema a resolver, es decir, pueden aplicarse a cualquier contexto, ya que sus decisiones están basadas en la aleatoriedad. (Arranz de la Peña y Parra Truyol, 2005)

3.3.2.2 Limitaciones

- Si bien no se garantiza que el algoritmo genético encuentre la solución óptima del problema, existe evidencia empírica que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de los algoritmos de optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable y posible es que la solución supere a la solución obtenida con el genético, tanto en rapidez como en eficacia. (Rudoph, 1994)
- El problema de cómo considerar la función objetivo debe considerarse cuidadosamente para que se pueda obtener una mayor aptitud y que verdaderamente signifique una solución al problema. (Marczyk, 2004)

- En poblaciones pequeñas se puede dar que si un individuo, que es más apto que la mayoría de sus competidores, emerge muy pronto en el curso de la ejecución, se puede reproducir tan abundantemente que merme la diversidad de la población demasiado pronto, provocando que el algoritmo converja hacia el óptimo local y no rastree el paisaje adaptativo a fondo para obtener el máximo global. (Arranz de la Peña y Parra Truyol, 2005)

3.3.2.3 Aplicación al proyecto propuesto

Como se estuvo estudiando a lo largo de la sección de los algoritmos genéticos la aplicación a nuestro reproductor de música se debe adaptar y abstraer para poder utilizar el concepto de los algoritmos genéticos en nuestra aplicación.

Se definió que se creará un set inicial de individuos (población) que serán codificados como listas de reproducción. Cada lista contendrá un número fijo de genes cuyos alelos serán las canciones pertenecientes a dichas listas.

La función de evaluación tendrá en cuenta los atributos de los alelos de los diversos genes y evaluará la bondad del cromosoma en base a un conjunto de operaciones sobre dichos atributos.

Para poder mejorar la convergencia del AG se propuso mantener aquellos individuos que sobresalgan en una generación como súper individuos que serán utilizados como padres en posteriores generaciones.

En principio decidimos utilizar el modelo de AG básico con los operadores genéticos de selección y cruce. Para el proceso de selección nos decidimos por utilizar el selector natural por Torneo por sus ventajas frente a las otras alternativas. En principio la decisión fue tomada por su grado de adaptación al problema del elitismo.

Como operador de cruce elegimos el método de cruce uniforme explicado anteriormente. Se ha observado en pruebas que muestra mejor convergencia hacia un óptimo que los otros métodos.

3.4 MODELO DE APLICACIÓN

3.4.1 Arquitectura de la Aplicación

Para la arquitectura de la aplicación se decidió utilizar el concepto del modelo Cliente / Servidor, con un cliente web para que el usuario realice el pedido y un servidor de aplicaciones para que procese los datos ingresados, se conecte a los servicios web a utilizar, aplique los algoritmos escogidos y finalmente retorne al usuario la lista de reproducción deseada.

Dentro de las ventajas de esta estructura de n capas, se pueden identificar:

- Abstracción entre capas.
- Minimizar las dependencias.
- Buenas para estandarizar.
- Se pueden reemplazar “fácilmente”.
- Cada capa usa los servicios brindados por la capa inferior y expone servicios para la capa superior.
- Divide un sistema complejo.

(Rousselot, 2014 a)

Dentro de las desventajas podemos observar que capas extras e innecesarias pueden impactar en la performance, y que cambios grandes a la API deben ser en forma de cascada.

Es muy importante decidir qué capas tener y la responsabilidad de cada una. En nuestro caso tendremos para el prototipo únicamente 2 capas, una que

contendrá la comunicación con los servicios web y la lógica de negocios y otra capa que será la del cliente. (Rousselot, 2014 a)

En un futuro, se deberá agregar una nueva capa de base de datos, para poder mantener estadísticas del usuario y poder obtener mayores datos sobre los gustos del mismo para sugerir mejores canciones, pero esto esté fuera del prototipo.

3.4.1.1 Servidor de Aplicaciones

Es la capa central donde se ejecuta la aplicación, mantenida independientemente de los clientes y de la base de datos en caso de que exista. Es un conjunto estandarizado de frameworks y servidores sobre los cuales se construye la aplicación. Es una plataforma de Middleware[30] para el desarrollo y despliegue de software basado en componentes. (Rousselot, 2014 b)

Un componente es una pieza de software que es accedida solo vía interfaces, construida para su customización, composición y colaboración con otros componentes. Tiene un tamaño adecuado para su reúso, reemplazo y mantenimiento. (Rousselot, 2014 b)

Puede verse como una forma de agrupar implementaciones de objetos en una sola entidad, que puede ser ensamblada dentro de un sistema más grande. El componente debe seguir las reglas del modelo de componentes y usar el conjunto de servicios establecidos por el mismo. Este desarrollo basado en componentes, mejora el manejo de la complejidad, reduce el tiempo de entrega, incrementa la productividad,

[30] El Middleware es un software que asiste a una aplicación a comunicarse con otras aplicaciones, redes, hardware y/o sistemas operativos. Funciona como una capa de abstracción de software, situada entre las aplicaciones y el sistema operativo.

permite el desarrollo paralelo y distribuido, reduce costos de mantenimiento y permite usar el mejor de su clase. (Rousselot, 2014 b)

El servidor de aplicaciones, provee servicios configurables disponibles y define las interfaces para accederlos. Estos servicios son: concurrencia, coordinación de transacciones, seguridad, administración de recursos, distribución entre distintos servidores, comunicación entre distintas partes del sistema distribuido y naming. (Rousselot, 2014 b)

La lógica de negocios de una aplicación, debe ser escrita y "deployada" dentro del servidor de aplicaciones, el cual a su vez sugiere un modelo de programación para instalar el software.

3.4.1.2 Cliente

Es la capa de presentación, la interacción entre el usuario y el software desarrollado. Esta es la que el usuario ve, presenta el sistema al usuario, le comunica la información y captura la misma en un mínimo de procesos. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" para el usuario. Se comunica únicamente con la capa de negocio. (Rousselot, 2014 b)

Aquí se inician las solicitudes o peticiones, por lo tanto tienen un papel activo en la comunicación. Por lo general, puede conectarse con varios servidores a la vez.

Existen diversas formas de presentar esta capa, algunas son:

- Cliente local - offline, con servidor en sitio.
- Página WEB.
- Dispositivo Mobile.
- SmartWatch.

3.4.2 Librerías / Aplicaciones A Utilizar.

3.4.2.1 Servidor

Como servidor de aplicaciones se utilizará JBoss, un servidor extensible y modular. Es un proyecto de código abierto, basado en J2EE e implementado al 100% en Java. Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones, tareas relacionadas con el mantenimiento de la seguridad y del estado, acceso a datos y persistencia entre otros.

Sirve para desarrollar aplicaciones web que involucran bases de datos, aplicaciones web con altos grados de complejidad (o sencillas), y aplicaciones middleware cruzadas.

El JEE es un conjunto de tecnologías para el desarrollo de aplicaciones multicapas. Esta especificación se implementa en los servidores de aplicación y sus aplicaciones pueden ejecutarse en distintos servidores sin modificar el código. (Rousselot, 2014 b)

Define la interacción y API entre diferentes servicios de base como directorios, transacciones, base de datos, mensajería, seguridad, concurrencia y ciclo de vida. Elimina la necesidad de codificar los servidos de base dentro de la aplicación, lo cual facilita el desarrollo, reduce su tiempo y evita la introducción de errores por estos servicios. (Rousselot, 2014 b)

Estandariza la forma de empaquetar aplicaciones e instalarlas como JAR, WAR o EAR.

Se decidió utilizar esta estructura para eliminar la dependencia del cliente con el servidor, y poder tener varios clientes, como aplicaciones web, mobile, para smartwatches y para toda nueva tecnología que surja.

3.4.2.2 Cliente

Para el prototipo, del lado del cliente se utilizará en principio Yeoman para la construcción del esqueleto de la aplicación web. Esta herramienta sirve para la agilización del proceso de inicio del desarrollo mediante la construcción de la estructura de la aplicación web. Este procedimiento se conoce como scaffolding. (Yeoman, 2012)

Yeoman es la combinación de 3 herramientas que en conjunto logran este cometido:

- Bower para el manejo de dependencias. Es un manejador de paquetes para la web, está enfocado y optimizado para el front-end por lo que el manejo del árbol de dependencias es plano, lo que evita problemas de versionamiento de las librerías además de una cantidad innecesaria de dependencias por versión. (Yeoman, 2012)
- Grunt para la previsualización, ejecución de pruebas y construcción de la aplicación. Esta herramienta puede ejecutar casi cualquier tarea que pueda ser definida de manera programada. Es una herramienta muy utilizada en el desarrollo de aplicaciones web, ya que existen muchas tareas predefinidas y repetitivas que se deben llevar a cabo para lograr optimizar el software y convertirlo en una pieza de calidad que siga los estándares de la web con el menor esfuerzo posible. (Yeoman, 2012)
- Yo[31] para la construcción del esqueleto de la aplicación dependiendo su tipo. (Yeoman, 2012)

[31] Abreviación de Yeoman.

Yeoman utiliza lo que se llama generadores para saber qué tipo de aplicación debe construir. Un generador es una rutina escrita para construir un esqueleto en particular. (Yeoman, 2012)

En nuestro prototipo utilizaremos Yeoman para la construcción del esqueleto de AngularJS para la aplicación web. AngularJS es un framework de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. (AngularJS, 2010)

AngularJS corre del lado del cliente y se centra en intentar dinamizar documentos HTML. Normalmente esto se consigue haciendo uso de CSS y javascript de forma que en función de los eventos que se produzcan en nuestra página, estos se actualicen, creen o eliminen determinados componentes de nuestro DOM. Con AngularJS se cambia un poco el enfoque de "dinamización" de documentos HTML estáticos mediante la vinculación de elementos de nuestros documentos HTML con el modelo de datos. De este modo, se define un modelo de datos (javascript) que se corresponderá con determinadas partes del HTML y siempre que haya cambios en una parte, automáticamente se verán reflejados en la otra. (AngularJS, 2010)

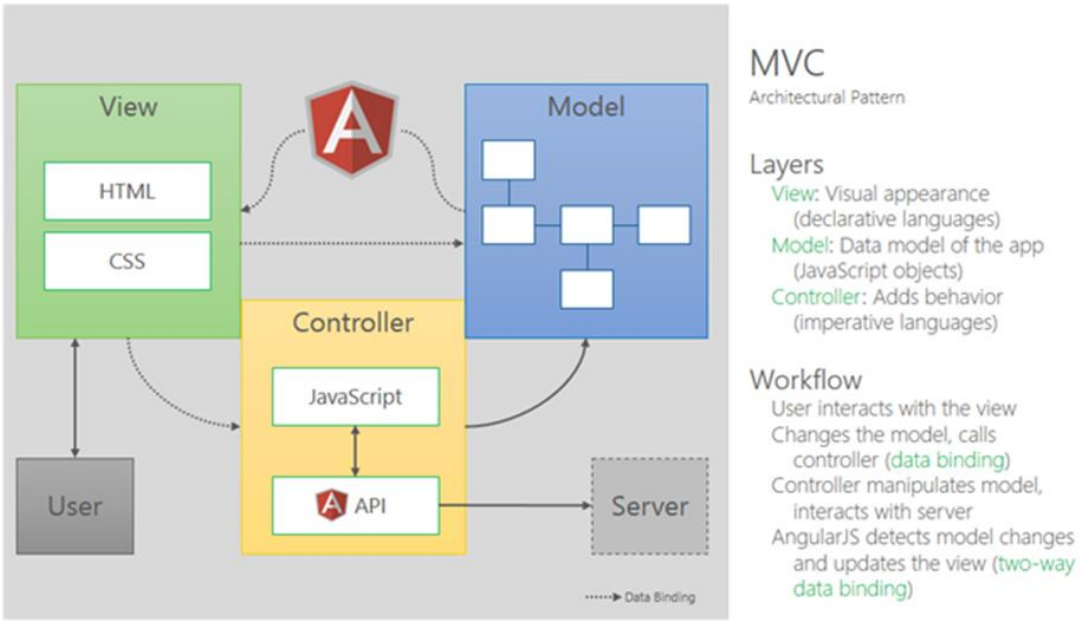

Figura 9: Arquitectura MVC de AngularJS (AngularJS, 2010)

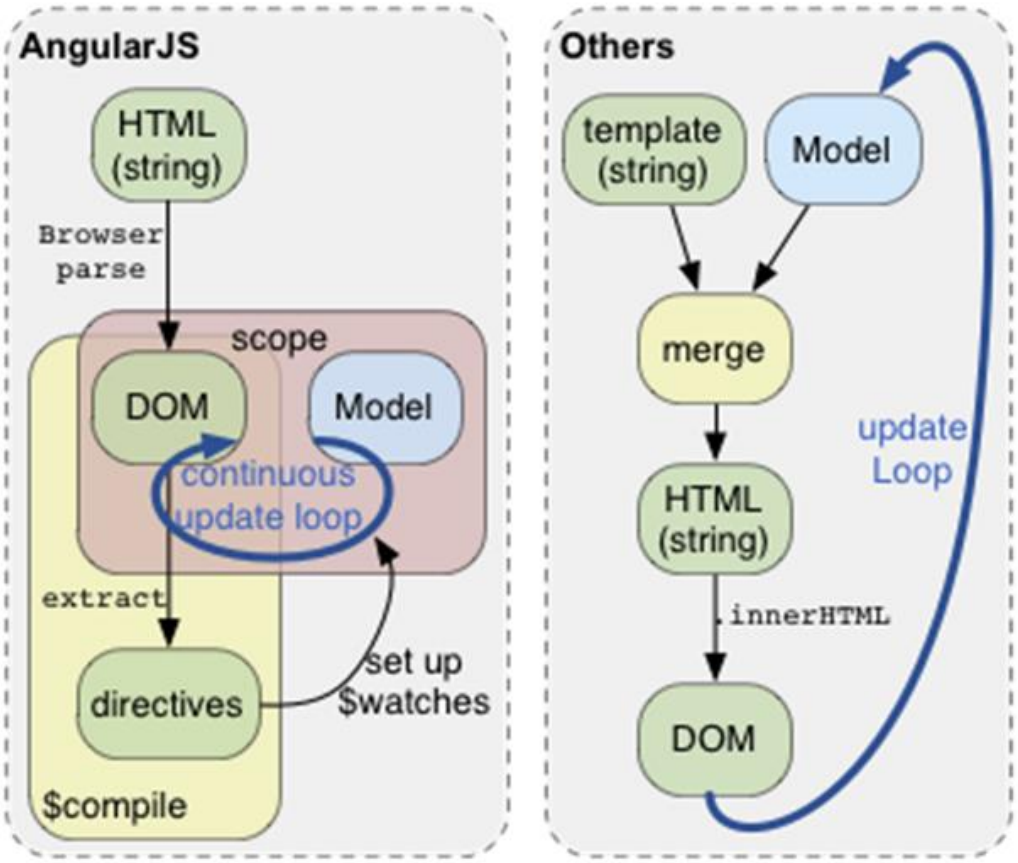

Figura 10: Comparación de funcionamiento en AngularJS vs Otros (AngularJS, 2010)

Dentro de los objetivos de AngularJS podemos encontrar:

- Disociar la manipulación del DOM desde la lógica de la aplicación. Esto mejora la capacidad de prueba del código.
- Valorar el testing de aplicaciones, dándole igual importancia que a la escritura de la aplicación.
- Disociar el lado del cliente de una aplicación del lado del servidor.

Para realizar el front-end web responsive se utilizará BootStrap. Este es un framework creado por la empresa Twitter para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía,

formularios, botones, cuadros, menús de navegación, y otros elementos de diseño basados en HTML y CSS.

3.4.2.3 Librerías para Algoritmos Genéticos

Una vez definidos todos los aspectos del servidor, del cliente y de los servicios web a consumir para obtener los datos, es necesario contar con alguna librería que nos facilite la implementación para desarrollar el Algoritmo Genético de optimización de las listas de reproducción. Lo que buscamos con la librería es poder configurar el algoritmo de la forma deseada, tener la posibilidad de definir la función de fitness adecuada para optimizar nuestro proceso y poder evolucionar las listas de reproducción hasta encontrar la más adecuada para el usuario.

Allí encontramos JGap, un framework que proporciona los métodos y mecanismos básicos para implementar los algoritmos genéticos. Fue diseñado para que fuera muy fácil de usar, siendo altamente modular y configurable, llegando a ser realmente fácil crear incluso nuevos y personalizados operadores genéticos. (López, 2010)

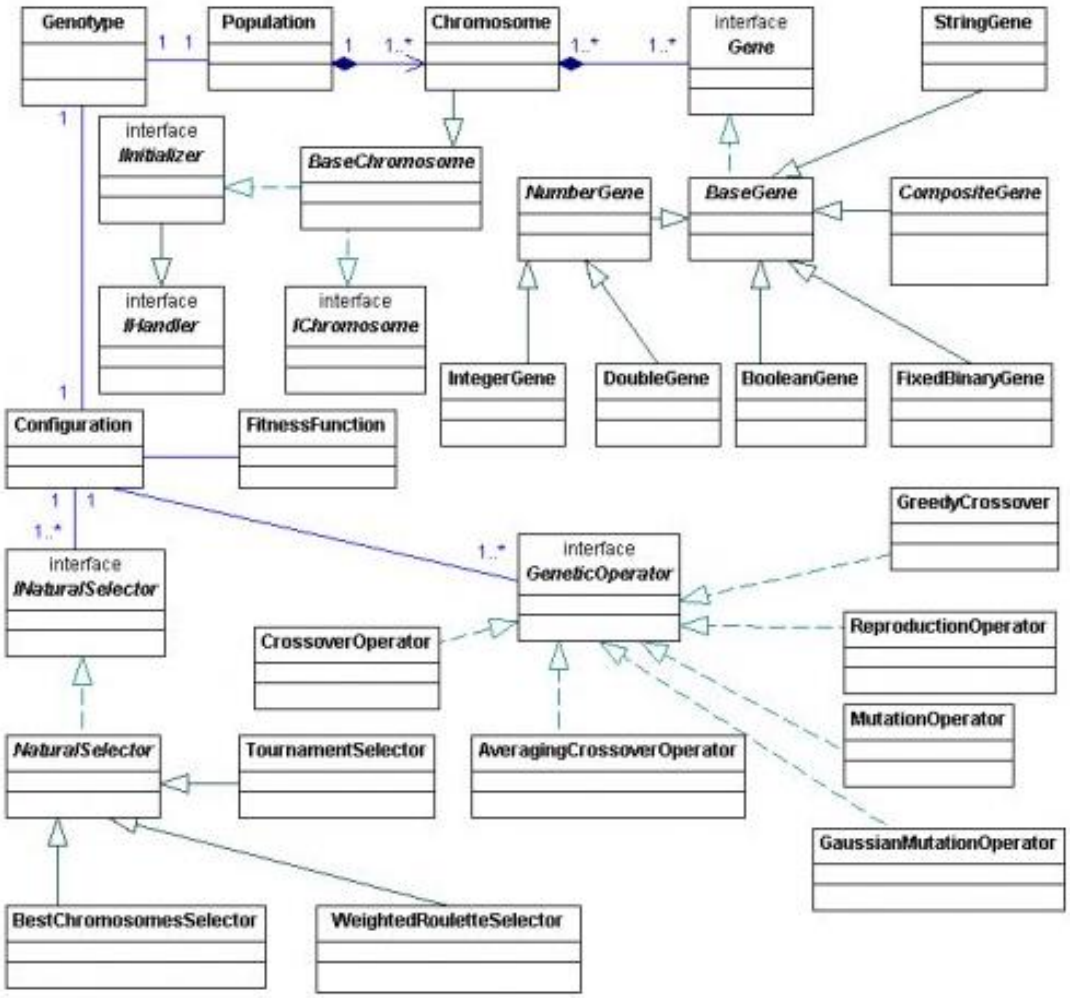

Figura 11: Diagrama de clases en JGap (López, 2010).

El framework se encarga de ejecutar los operadores genéticos en cada iteración, retornando el individuo con mayor fitness (permitiendo incluso obtener cualquier individuo de la población, en cualquier iteración).

Para utilizar JGap en una aplicación se deben seguir 5 pasos:

1) Planear el Cromosoma. En el corazón del algoritmo genético se encuentra el Cromosoma. Este representa una solución potencial y es definido en múltiples genes. Los genes en JGap representan

distintos aspectos de la solución como un conjunto, así como en la vida real los genes humanos representan distintos aspectos individuales de las personas, como su sexo o color de pelo. Durante el proceso de evolución en JGap, los cromosomas quedan expuestos a múltiples operadores genéticos para luego ser escogidos para la nueva generación durante una fase de selección natural basada en el fitness de cada cromosoma, que mide cuán óptima es una solución en relación con otras potenciales soluciones. El objetivo central es imitar el proceso natural de la evolución para producir soluciones superiores. (Meffert y Rotstan, 2002)

2) Implementar una función fitness. JGap no tiene el conocimiento del problema específico que se desea resolver, por consiguiente, no tiene manera de decidir si una solución potencial es mejor o peor a otra. Allí es donde la función de fitness entra en juego, este es un simple método a implementar que recibe una potencial solución al problema y devuelve un número que indica que tan buena es esa solución en relación al resto. (Meffert y Rotstan, 2002)
3) Preparar un objeto de configuración. Aquí se define la función de fitness a utilizar, la población inicial que se utilizará y cómo se realizará el setup del cromosoma. También se pueden crear operadores genéticos nuevos, selectores naturales o generadores aleatorios, y es aquí donde deben definir cómo estos métodos creados se utilizarán.(Meffert y Rotstan, 2002)

4) Generar una población inicial. Una población de cromosomas se denomina Genotipo y esta es la clase a construir para crear la población. (Meffert y Rotstan, 2002)
5) Evolucionar la población. Una vez definidos todos los parámetros de configuración, es hora de evolucionar la población hasta encontrar soluciones potenciales óptimas. En JGap, este proceso demanda muy poco esfuerzo, ya que toma toda la configuración antes definida. Finalmente, en cada ciclo de evolución, se puede observar el cromosoma con mejor fitness y ver si el criterio de corte cumple con el fitness obtenido. Otro criterio de corte con el que contamos es llegar a un número máximo de evoluciones.(Meffert y Rotstan, 2002)

3.5 CONCLUSIÓN

Una vez realizada la investigación del estado del arte, y concluida la fase del marco teórico, podemos afirmar que contamos con todas las herramientas ya definidas para comenzar con el desarrollo del prototipo.

Como ya especificamos durante el capítulo en resumidas cuentas utilizaremos:

- Metodología ágil: SCRUM.
- Servidor de Aplicación: JBoss.
- Cliente: Yeoman + AngularJS + Bootstrap.
- Obtención datos: Spotify + EchoNest.
- Librería para Algoritmo Genético: jGap.

Si bien el desarrollo de algoritmos genéticos, no está relacionado explícitamente con las metodologías ágiles, la técnica de SCRUM aplica a cualquier industria y a cualquier tecnología. La estaremos usando para nuestra organización en el desarrollo del prototipo y para la documentación del mismo.

4 DESARROLLO DEL PROTOTIPO

4.1 INTRODUCCIÓN

Hemos decidido desarrollar un prototipo que nos permita mostrar nuestro trabajo de investigación para la resolución y optimización de generación de listas musicales.

Para la realización del prototipo utilizamos una arquitectura Cliente-Servidor ya previamente explicada, con una aplicación web desarrollada utilizando AngularJS que nos sirvió para que el usuario pueda interactuar con el algoritmo permitiéndole el ingreso de ciertos parámetros iniciales.

Mientras que para el servidor de aplicaciones utilizamos JBoss en el que expondremos ciertos servicios, utilizando la tecnología REST, necesarios para la integración entre la aplicación web y el algoritmo.

4.2 DESARROLLO

4.2.1 FUNCIONAMIENTO

El funcionamiento del prototipo se basa en la selección de ciertos parámetros de entrada por parte del usuario utilizando la pantalla principal de búsqueda desde la aplicación web.

Desde esta pantalla el usuario puede buscar artistas, géneros y estados de ánimo, los cuales serán utilizados para generar la población inicial del algoritmo y que este se encargue de la generación de la una lista óptima.

Adicionalmente el usuario tiene la posibilidad de ingresar una cantidad de tiempo X el cual indica el tiempo total de duración de la lista y un tiempo Y utilizado para restringir el tiempo particular de duración de cada canción.

Luego de completar estos datos se envía a través de un servicio REST la información ingresada por el usuario hacia el servidor el cual procesa la información y genera una lista de canciones óptima, gracias a la utilización del algoritmo genético.

Esta lista de canciones es enviada al cliente y se reproduce a través del Widget HTML, servicio provisto por Spotify.

4.2.2 DIAGRAMAS DEL PROTOTIPO

A continuación incluimos dos diagramas del prototipo. El primero detalla la comunicación entre el usuario y el servidor a través de la capa de comunicación. El segundo diagrama muestra el diagrama de clases de la implementación del algoritmo genético en el prototipo.

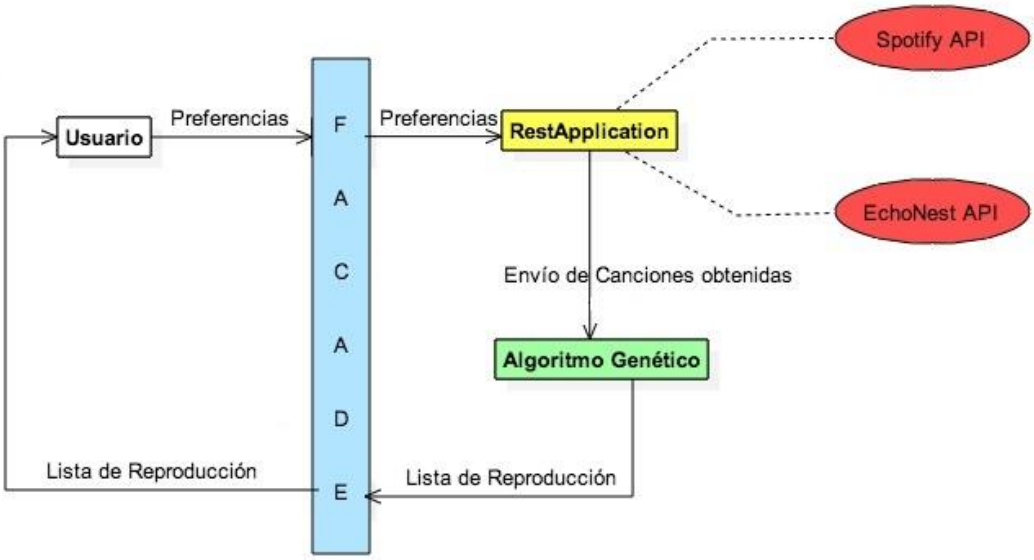

Figura 12: Diagrama interacción cliente-servidor.

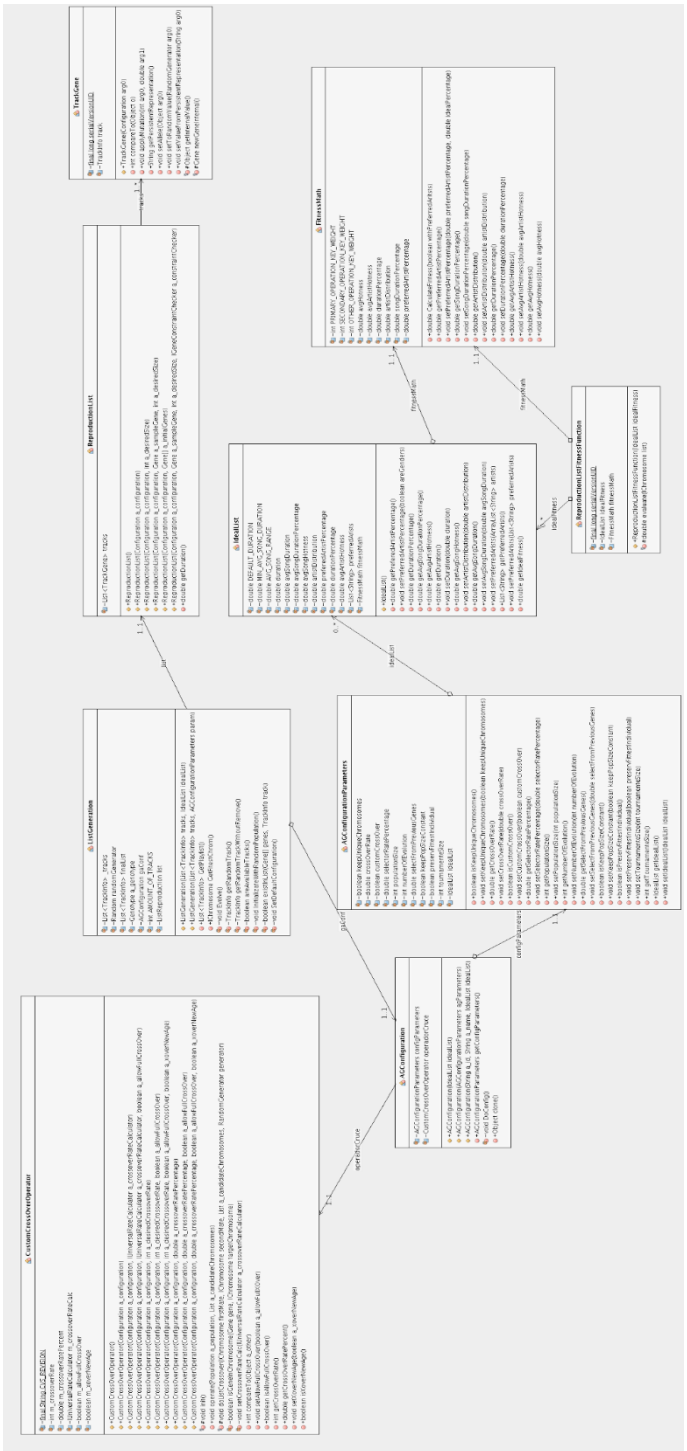

Figura 13: Diagrama de clases de la implementación del Algoritmo Genético.

4.2.3 INTEGRACIÓN CON SERVICIOS

El prototipo consume información provista por Spotify y EchoNest. Para poder obtener esta información se buscaron frameworks que faciliten la conexión con estos, utilizando la tecnología Java para poder integrarlo en nuestro servidor de aplicaciones.

4.2.3.1 Spotify API

Se ha utilizado una de las librerías desarrolladas en Java promovidas por Spotify, la cual fue creada por Michael Thelin[32].

Esta librería encapsula los servicios básicos de la API Web de Spotify tales como búsqueda, autenticación y creación de listas.

Utilizamos esta librería para:

- Autenticarnos con Spotify
- Obtener los artistas y su información para la pantalla de búsqueda.

Toda la documentación sobre esta librería se encuentra en: https://github.com/thelinmichael/spotify-web-api-java

4.2.3.2 EchoNest

Para la integración con los servicios provistos por EchoNest utilizamos la librería jEN[33].

Se ha empleado esta librería para:

- Obtener los géneros disponibles para la página de búsqueda.
- Obtener los distintos estados de ánimo disponibles para la página de búsqueda.
- Realizar la búsqueda de canciones y su respectiva metadata.

[32] Empleado de Spotify en el área de Desarrollo.
[33] Librería desarrollada en Java que encapsula los servicios básicos de la API de EchoNest.

Se puede encontrar toda la documentación en: http://static.echonest.com.s3.amazonaws.com/jEN/javadoc/index.html

4.2.4 PARÁMETROS DE ENTRADA

El prototipo cuenta con la posibilidad de dejar al usuario establecer cierta parametrización que impactará en el resultado del algoritmo. Aquellos identificados fueron los que se describen a continuación.

Artistas

El usuario podrá ingresar una cierta cantidad de artistas, estos mismos serán seleccionados del catálogo de Spotify. De esta forma nos aseguramos que los artistas elegidos estén disponibles para la búsqueda y recopilación de datos de su música.

Géneros

Decidimos darle la posibilidad al usuario de buscar canciones por géneros musicales. Al igual que los artistas se pueden seleccionar múltiples géneros. Los mismos son obtenidos utilizando la API de EchoNest la cual provee más de cien géneros.

Al momento de realizar la búsqueda de canciones utilizamos un filtro especial que nos permite obtener sólo aquellas disponibles en el mercado de Spotify Argentina.

Estado de ánimo

Dentro de los parámetros de entrada que incluimos, se encuentra el filtro de estados de ánimo. Los mismos son obtenidos a través de EchoNest, quién cuenta con varios estados de ánimo disponibles para seleccionar. Su selección puede ser tanto individual como múltiple.

Tiempo de reproducción

Permitiremos al usuario elegir el tiempo de reproducción de su lista, el mismo puede variar entre 40 - 240 minutos. Si el usuario no ingresa un tiempo se tomará 60 minutos de duración por defecto

Tiempo de duración de canción

Se permitirá al usuario ingresar el tiempo de duración promedio de las canciones a reproducir. El mismo puede ir de 2 minutos a 20 minutos. Utilizando este filtro el algoritmo prioriza aquellas canciones cuyo tiempo de reproducción esté entre el valor deseado y un rango de un minuto. (ej: Con un valor de 4 minutos el algoritmo valoriza aquellas listas que posean más canciones cuyos rangos están entre 3 minutos y 5 minutos, pero no descarta canciones con menor o mayor tiempo).

Recopilación de información

Luego de que el usuario ingresa sus preferencias de búsqueda, utilizamos EchoNest para buscar todas las canciones que cumplan con los requisitos ingresados por el usuario, para luego nutrir la población inicial del algoritmo genético.

En primera instancia se buscan todas las canciones correspondientes a él/los artista/s ingresado/s que pertenezcan al mercado de Spotify Argentina. El motivo de esta configuración es que Spotify divide su mercado según regiones. Por este motivo, debemos asegurarnos que las canciones obtenidas por EchoNest estén disponibles en el mercado local.

Luego procedemos a buscar todas las canciones de los distintos géneros y estados de ánimo ingresados. Para realizar esta operación también utilizamos la API de EchoNest con las mismas restricciones.

En ambos casos utilizamos filtros especiales que nos permiten focalizar la búsqueda en los resultados que nos son útiles. Para mayor detalle, los filtros

están disponibles en la documentación de EchoNest, de cualquier forma a continuación explicamos su uso:

Tabla II: Filtros de Entrada a Pedido EchoNest

Filtro	Valor	Descripción
sort	song_hotttnesss-desc	Ordena las canciones de forma descendente según su popularidad
results	100	Retorna los primeros 100 resultados disponibles.
song_type	studio	Filtra las canciones que no fueron grabadas en estudio, excluye las canciones en vivo.
max_speechiness	0.6	En la búsqueda de pistas, EchoNest no puede diferenciar entre canciones o entrevistas. Utilizando este parámetro filtramos aquellas pistas que en realidad no son canciones.
limit	true	Limita la búsqueda a los resultados que contengan los parámetros ingresados.

Una vez que EchoNest nos retorna todas las canciones disponibles seleccionamos la metadata que nos es útil para el algoritmo. De la información de las pistas utilizamos:

Tabla III: Información de las pistas utilizadas para el algoritmo.

Metadata	Descripción
spotifyId	El id correspondiente a la canción en el banco de datos de spotify.
artistHotttnesss	El valor asignado por EchoNest que indica que tan popular es el artista.
songHotttnesss	El valor asignado por EchoNest que indica que tan popular es la canción.
artistId	El id del artista correspondiente al banco de datos de spotify.
duration	Duración de la canción.

Una vez obtenidas todas las pistas musicales, el tiempo de reproducción de la lista y el valor de duración de cada canción damos por finalizado el trabajo de la API de EchoNest e inicializamos el proceso de trabajo de nuestro AG.

4.2.5 ALGORITMO GENÉTICO

4.2.5.1 Introducción

En esta sección utilizamos lo analizado anteriormente sobre la teoría de algoritmos genéticos para poder desarrollar un algoritmo que, utilizando los parámetros de entrada ingresados por el usuario, nos devuelva una lista óptima de canciones.

En esta breve introducción queremos volver a poner en consideración el problema, expuesto en nuestra introducción del PFI, que se desea resolver con nuestro AG.

Dada una serie de parámetros ingresados por el usuario se desea obtener una lista de reproducción generada aleatoriamente que cumpla, tanto con los parámetros del usuario, como con los índices de selección elegidos por nosotros, con la mejor aproximación a una solución aceptable.

Decidimos utilizar la librería jGAP para el desarrollo del AG con el objetivo de reducir el costo de desarrollo y minimizar posibles errores de implementación.

4.2.5.2 Elementos

Habiendo realizado una introducción a los elementos de un algoritmo genético, explicaremos de qué forma representamos cada uno de ellos en nuestro problema a resolver.

Individuo: Lista de reproducción. Representa una posible solución a nuestro problema.

Cromosoma: Lista de reproducción. Contiene una lista de pistas de música con su correspondiente metadata. El cromosoma representa al individuo.

Gen: En nuestro caso, el gen, es una estructura de datos con la información y funcionalidad que representan a una canción contenida en un cromosoma.

Población: Conjunto de listas de reproducción.

Generación: Es la representación de la evolución de las listas de reproducción en una iteración.

Genotipo: Conjunto de listas correspondientes al espacio de búsqueda del algoritmo.

Fenotipo: Dado que no se realiza una codificación del algoritmo, no aplica en nuestro problema.

Función de evaluación: La función de evaluación es la responsable de evaluar cada lista de reproducción y asignarle un valor de ajuste según su calidad.

Alelo: Representa el valor de gen, en nuestro caso el alelo es la pista musical.

4.2.5.3 Configuración del AG

La configuración del algoritmo es una de las etapas claves en el proceso. Contamos con el framework provisto por jGAP que nos indica cuales son los parámetros de configuración necesarios para que el AG trabaje. A continuación detallamos la configuración más óptima que hemos encontrado. En la sección de Entrenamiento (4.2.6.9) explicamos cómo hemos llegado a esta configuración.

Tabla IV: Configuración óptima del Algoritmo Genético

Parámetro	Valor
Operador de selector natural	Selección por torneo
Evaluador de valor de ajuste	Por defecto. Evalúa si un valor de ajuste es mejor que otro si su valor es mayor.
Función de ajuste	Implementación que pondera un individuo dentro de la población asignando un valor de ajuste.
Operadores genéticos	Incluimos un operador de cruzamiento uniforme, dado que el framework no proveía uno, el mismo

	fue desarrollado por nosotros. No se han incluido operadores de mutación.
Tamaño de la población	450 individuos por generación.
Prevalecer el mejor individuo	Verdadero. El mejor individuo de cada generación pasará a ser padre en la siguiente.
Mantener población constante	Verdadero. Cada generación contará con 450 individuos. El método de selección por torneo será el responsable de limitar la cantidad de individuos.
Número de evoluciones	350. El proceso de selección y cruzamiento se realizará hasta 350 veces.

4.2.5.4 Función de ajuste

La función de ajuste es tal vez, la tarea más crítica del AG y la más difícil de definir. Todas las operaciones que realiza el AG son de alguna forma mecánicas, en cambio, la función de ajuste es la que nos define que tan buena es una posible solución frente a las otras, y este análisis lo debemos realizar nosotros mismos. Es por esto que hay dos temas claves a analizar en este instante.

En primer lugar se debe tener muy en claro cuál es el problema. Si no se entiende el mismo, muy posiblemente no tengamos una buena solución. Y en segundo lugar, se debe definir que es una buena solución para nuestro problema.

En nuestro caso ya está claro que el problema que debemos resolver es optimizar el armado de una lista de reproducción. A lo que inmediatamente viene la pregunta ¿Qué es y qué cualidades debe tener una buena lista de reproducción?

Justamente esto es lo que debe resolver la función de ajuste. Debe tomar una lista de reproducción y ponderarla asignándole un valor. Para esto hemos definido una serie de índices que, nosotros creemos, son los indicados para definir si una lista de reproducción es buena, o no.

A. Porcentaje de cumplimiento de duración de la lista.

El objetivo de este índice es medir el grado de cumplimiento de duración de la lista frente al tiempo ingresado por el usuario. Para esto se suma el tiempo de cada canción perteneciente a la lista evaluada y se compara frente al tiempo ingresado por el usuario (o el tiempo por defecto). En caso de que el tiempo total de la lista supere el tiempo ingresado por el usuario se restará dos veces la diferencia entre el tiempo de la lista y el tiempo del usuario, de esta forma nos aseguramos que el valor total representa un número entre [0...1].

B. Distribución de artistas

La razón de este índice es medir la relación que existe entre la cantidad de aristas en la lista evaluada y la cantidad de canción pertenecientes a este artista. El índice se mide sumando la cantidad de artistas. Una vez obtenido el total de

artistas en la lista se procede a sacar la proporción ideal que debería existir realizando la siguiente operación [1/cantidadArtistas]. Una vez obtenida esta proporción se evalúa cada artista en particular y su cantidad de canciones. Si este artista cumple con dicha proporción, agregando un margen 0.05 se suma una unidad a el índice de distribución de artistas. Una vez analizados todos los artistas, se evalúa la cantidad que cumplieron con la distribución y se divide por el total de artistas.

C. Porcentaje de cumplimiento de duración de canciones

El objetivo de este índice es medir en qué grado la lista evaluada cumple con la restricción propuesta por el usuario. De esta forma se evalúan todas las canciones de la lista que cumplen con el tiempo propuesto por el usuario con un margen de 60 segundos. Luego se suma el total de canciones que cumplen y se divide por el total de canciones en la lista.

D. Porcentaje de asignación a los artistas preferidos

El motivo por el cual se agregó este índice es para marcar una superioridad en las canciones que pertenecen a los artistas seleccionados por el usuario. Recordamos que el usuario puede elegir tantos artistas, géneros y estados de ánimo como le plazca. Esto nos lleva a pensar que el usuario prefiere escuchar más canciones de sus artistas elegidos, que tal vez canciones al azar de un género particular.

Con esta idea en mente nos propusimos establecer una proporción de 70/30, 70% de las canciones en la lista deben pertenecer a los artistas seleccionados, mientras que el 30%

restante son de artistas provenientes de los géneros y estados de ánimo.

Para realizar el cálculo de este índice se evalúa la proporción de canciones pertenecientes a los artistas preferidos. De esta forma se suman todas las canciones que estos tienen en la lista de reproducción y se divide por el total de canciones. Aquí se aplica la misma regla que para la duración de la lista, si el porcentaje de canciones pertenecientes a los artistas supera el 70% se procederá a restar la diferencia (porcentajeObtenido - 70%) multiplicado por dos. De esta forma si el porcentaje obtenido es del 80% la cuenta sería la siguiente [80 - 2 * (80 - 70) = 60]. El motivo de este tipo de cálculo se entenderá mejor una vez explicados los índices.

E. Promedio de popularidad de canciones

Este índice refleja que tan buenas son las canciones que forman parte de una lista. Esta medida se calcula sumando la popularidad de cada canción y se divide el resultado por el total de canciones.

F. Promedio de popularidad de artistas

Al igual que el índice explicado anteriormente, se suma la popularidad de cada artista y se divide por el total de artistas en la lista.

Ahora bien, si tenemos todos estos indicadores que aportan una visión de lo que es importante para que la lista sea medible, debemos asignar un peso único, el llamado valor de ajuste, a la lista.

En nuestra operación decidimos crear tres grupos de operadores:

A. Primarios

Los más importantes. Son aquellos índices en los que el usuario tuvo inferencia sobre su valor. Estos son:

1. Tiempo total de la lista.
2. Tiempo de duración de canción.

B. Secundarios

Definimos como secundarios aquellos operadores que aportan armonía a la lista y de alguna forma fueron sugeridos por el usuario indirectamente. En nuestro caso contamos con un solo operador de este tipo, la distribución de artistas preferidos.

C. Otros

Por último tenemos aquellos que pensamos que no deberían influir demasiado en el resultado dado que son aquellos índices en los que el usuario no tuvo poder de decisión.

1. Promedio de popularidad de canciones.
2. Promedio de popularidad de artistas.
3. Distribución de artistas.

Decidimos dejar la distribución de artistas en este grupo dado que, por un lado el usuario no elige una distribución proporcional, lo hacemos nosotros, y segundo este índice puede verse afectado por la distribución de artistas preferidos, el cual consideramos que debe ponderarse por encima de la distribución de artistas.

Ya explicados estos operadores definimos que la fórmula para evaluar el ajuste de la lista será la siguiente

$$valorAjuste = 45 * SUMA(A) + 30 * SUMA(B) + 10 * SUMA(C) \quad (1)$$

Siendo A los Operadores primarios, B los operadores secundarios y C los otros. De esta forma nos aseguramos que la lista que mejor se ajuste, tendrá mejor acoplamiento a las preferencias del usuario. Si el usuario no ha ingresado artistas preferidos no se tienen en cuenta los operadores del grupo B para el cálculo de ajuste.

4.2.5.5 Método de cruzamiento

Durante el desarrollo del AG nos hemos enfrentado a la tarea de seleccionar un método de cruzamiento. Primero elegimos el método punto a punto. Al ver que los resultados no eran los esperados decidimos inclinarnos más hacia el método uniforme. En este cambio hay una buena razón explicada posteriormente.

Cuando uno define un cromosoma está obligado a que tenga una capacidad fija de genes, esto se debe a que sin esta restricción no sería posible aplicar los operadores genéticos por la naturaleza de estos mismos. Por lo tanto se definió que cada cromosoma tenga treinta (30) genes. Esto significa que como máximo cada lista puede tener treinta (30) canciones.

Ahora bien, una lista de reproducción podría durar sesenta (60) minutos, por lo que treinta canciones no solo son más que suficientes, sino que sobran lugares disponibles en la lista de genes que, de alguna forma, deben ser completados. Para esto definimos que estos espacios vacíos deben representar una canción nula, sin tiempo, ni artista, ni ninguna clase de metadata, pero identificando para que no sea tenida en cuenta para la lista final.

Si quisiéramos representar un cromosoma, no hay mejor estructura de datos para hacerlo que un vector. En nuestro ejemplo usamos dos

cromosomas de tamaño cinco (5) en donde los dos primeros genes son canciones reales y los subsiguientes tres son canciones vacias.

Canción1	Canción2	CanciónNula	CanciónNula	CanciónNula

Figura 14: Cromosoma uno

Canción1	Canción2	CanciónNula	CanciónNula	CanciónNula

Figura 15: Cromosoma dos

Ahora bien, en el método punto a punto se intercambian los genes de un punto seleccionado al azar entre cromosomas. Si nuestro número fuese la posición número tres, los cromosomas intercambiarán sólo posiciones con canciones nulas. Esta limitación hace que el cruzamiento sea poco efectivo.

Con este motivo decidimos que, utilizar el cruce uniforme, es más apropiado, dado que se seleccionan una cantidad X de genes obtenidos al azar y se intercambian, por lo que se puede notar un intercambio de material genético útil entre los cromosomas, situación que no se da muy seguido con el método punto a punto.

Al momento de definir el operador de cruce se debe definir también el porcentaje de cruzamiento que se hará entre la población. Este porcentaje decidirá cuantos cruces se harán, y cuantos cromosomas pasarán a ser evaluados por el método de selección.

Luego de nuestro entrenamiento realizado obtuvimos que “0.85” es el porcentaje que mejores resultados obtuvo. Esto quiere decir que habrá tantos cruces como:

$$cantidadDeCruces = 0.85 * tamañoPoblación \quad (2)$$

Por lo que al final del proceso de cruce la población crece de la siguiente forma:

$$poblaciónFinal = poblaciónActual + cantidadDeCruces * 2 \quad (3)$$

Debemos tener en cuenta que los padres se mantendrán y cada cruce genera dos hijos. Posteriormente en el proceso de selección la población vuelve a converger a su número original, seleccionando aquellos individuos más aptos y dejando atrás aquellos menos aptos.

4.2.5.6 Método de selección

El método de selección utilizado es el método por torneo. Nuestro entrenamiento del algoritmo arrojó que la probabilidad de que un individuo de menor ajuste pueda ganarle a uno mayor debe ser 0.75. Con este porcentaje logramos darle la diversidad necesaria a la población, eliminar el elitismo y evitar la convergencia prematura hacia un óptimo local.

El número de participantes en el torneo será de dos cromosomas por competición. De esta forma aquellos con mejor valor de ajuste tienen más probabilidades de pasar a la siguiente generación.

4.2.5.7 Evolución

El proceso de evolución es sencillo, una buena parte de esto se debe al framework utilizado. Primero se inicializa una población inicial. Para esto se crea un genotipo, el cual define un modelo de cromosoma y se especifica el tamaño de la población. Automáticamente el genotipo genera una población inicial con la cantidad de cromosomas especificados. Luego, manualmente, se completan los genes de estos cromosomas con canciones elegidas al azar. Cada cromosoma se completa hasta cumplir con la cantidad de tiempo especificado por el usuario.

Una vez que los cromosomas están completos se procede a ejecutar el proceso de evolución. El procedimiento es sencillo, primero se ejecutan los operadores de cruce, esto produce un aumento en la población. Luego

se re evalúa el ajuste de cada posible solución. Por último se ejecuta el proceso de selección que filtra aquellas listas que no serán consideradas aptas para pasar a la siguiente generación. Dado que para nosotros se debe mantener el tamaño de la población fijo, el proceso de selección agrega a la nueva población tantas listas como tamaño de la población esté especificado.

Para poner en números concretos este procedimiento, tomaremos un ejemplo de una población inicial de 100 individuos.

Primero se ejecuta el proceso de cruzamiento, esto nos da una descendencia de 270 individuos (obtenidos según fórmula 4). Estos individuos son filtrados por el operador de selección, el cual se ejecuta hasta obtener una nueva población de 100 individuos.

$$100 + (0.85 * 100) * 2 = 270 \quad (4)$$

Este proceso se ejecuta unas 350 veces hasta que se cumpla el criterio de corte.

4.2.5.8 Criterio de corte

El proceso de un AG puede evolucionar tantas veces como sea necesario. Muchas veces este proceso es largo y computacionalmente costoso. En general, los programas que utilizan un AG poseen un criterio de corte que permite acotar el tiempo de procesamiento cuando se haya encontrado una solución óptima.

Es aquí donde tenemos la necesidad de definir cuándo es que mi solución es lo suficientemente buena como para detener el proceso, debido a que no basta solamente con que el ajuste sea un número y que el resultado de procesar el algoritmo genético sea obtener el número más alto.

Por lo tanto definimos que uno de nuestros criterios de corte sea llegar a una lista óptima.

Para que una lista sea óptima definimos que esta misma debe cumplir con ciertos requisitos mínimos. Para esto, por supuesto, utilizamos nuestros índices ya mencionados en la función de ajuste.

A. Porcentaje de cumplimiento de duración de la lista.

   Este es considerado por nosotros el índice más importante. Es por eso que definimos que se debe cumplir en lo posible, en un 99%.

B. Distribución de artistas

   Si bien este índice es parte de la categoría con menor peso, creemos que es uno de los que mayor grado de cumplimiento debe tener por la calidad que le aporta a la lista de reproducción. Está claro que mientras más equitativa sea la lista con relación a los artistas, más aceptación tendrá por parte del usuario. Por esto mismo decidimos que el grado de cumplimiento debe ser de 95% en lo posible.

C. Porcentaje de cumplimiento de duración de canciones

   El grado de cumplimiento que hemos adoptado para este índice es de 97.5% por los mismos motivos ya mencionados en el índice A.

D. Porcentaje de asignación a los artistas preferidos

   Este índice ya fue explicado previamente, la lista óptima definida debería cumplir con un 70% de cumplimiento en lo posible.

E. Promedio de popularidad de canciones

   Definimos que nuestra lista debe tener al menos un porcentaje de 2% (ver sección Limitaciones para entender el porcentaje definido) en promedio de popularidad de canciones.

F. Promedio de popularidad de artistas

Definimos que nuestra lista debe tener al menos un porcentaje de 75% en promedio de popularidad de artistas.

Utilizando todos estos parámetros se realiza el cálculo del valor de ajuste para lo que es nuestra lista óptima de la misma forma en la que se realiza para cualquier cromosoma.

Nuestro otro criterio de corte es que se llegue a una cantidad de evoluciones máxima sin haber encontrado una buena solución. En este caso se devuelve la mejor posible encontrada en el último número de evolución.

En términos simples el AG continuará evolucionando hasta:

A. Llegar a la última iteración de evolución
B. El mejor valor de ajuste de la iteración en evaluación alcance o supere el valor de ajuste de la lista óptima.

4.2.5.9 Entrenamiento

En función de optimizar nuestro algoritmo genético realizamos pruebas de entrenamiento que nos permitieron encontrar la mejor configuración, ajustar porcentajes y mejorar la convergencia hacia la mejor solución posible. Para esto se crearon las siguientes pruebas utilizando datos de pruebas reales (Ver Anexo con archivos de prueba para cada caso).

A. Evaluar ajuste en base al tamaño de la población

Para esta prueba se ejecutó el AG con un tamaño inicial de población de 50 individuos, en cada iteración se ejecutó el algoritmo 30 veces, se obtuvo el promedio de ajuste y el mejor ajuste encontrado. Luego se incrementó la población en 50

individuos más y se volvió a correr el mismo proceso. Esto se realizó hasta llegar a un tamaño de población de 500 individuos. Se observaron mejores resultados con una población de 450 individuos.

B. Evaluar proceso de selección natural

Esta prueba tenía como objetivo comparar el método de selección por torneo clásico contra una adaptación propia del método en donde no se permitía que un individuo gane dos veces un torneo. Los resultados arrojaron que en cuestión de tiempo de procesamiento el segundo método es inviable.

C. Evaluar método de cruce

Se evaluó el método de cruce punto a punto frente al uniforme. Se encontraron mejores resultados en el método de cruce uniforme, donde la convergencia es mejor y se encuentran soluciones con mejor ajuste. En el método punto a punto muy difícilmente la solución llegaba a alcanzar el ajuste propuesto como ideal.

D. Evaluar porcentaje de cruce

Al igual que en las pruebas de población se empezó la ejecución de pruebas con un porcentaje de cruce 0.5 y se fue incrementando en cada iteración en 0.05. Los mejores resultados se obtuvieron entre el rango de 0.8 y 0.95. Por esto mismo se decidió que el porcentaje a utilizar será de 0.85.

E. Evaluar porcentaje de selección

Al igual que en la prueba de porcentaje de cruce se itero el porcentaje de selección desde 0.05 hasta llegar a 1.0. Al igual que la información obtenida en nuestra investigación, donde el

porcentaje ideal para este índice rondaba el 0.7, aunque con el entrenamiento se obtuvieron los mejores resultados utilizando un porcentaje de 0.75.

F. Evaluar si preservar o no el mejor individuo para la siguiente población

La razón de esta prueba era evaluar la convergencia hacia una mejor solución, si se mantenía o no el mejor individuo encontrado en una iteración y se pasaba a la siguiente. Los resultados se inclinaron a mantener el mejor individuo.

G. Evaluar si la población debe mantenerse constante o no

En cuestión de números (promedio de ajuste, mejor ajuste) no se observaron muchas modificaciones, pero en términos de tiempo de procesamiento se notó una gran diferencia al mantener la población constante.

H. Evaluar el número óptimo de evoluciones

Para esta prueba se realizaron varias ejecuciones variando el número de evoluciones. A medida que el AG va evolucionando la población converge hacia un valor de ajuste. Se encontró que entre 300 y 400 evoluciones son suficientes para que el AG pueda converger hacia su óptimo. Por esto mismo utilizamos 350 evoluciones para nuestra configuración.

4.2.6 USER STORIES

Para el desarrollo del prototipo se han identificado 25 historias de usuario que engloban tareas de Investigación, Análisis, Desarrollo y Entrenamiento. Las mismas están detalladas con sus criterios de aceptación en el Anexo Nº 2.

4.2.7 SPRINTS

La duración de los Sprints fue definida en 2 semanas. En la imágen a continuación mostraremos los sprints exportados de la herramienta utilizada para el seguimiento de las tareas: Vivify Scrum.

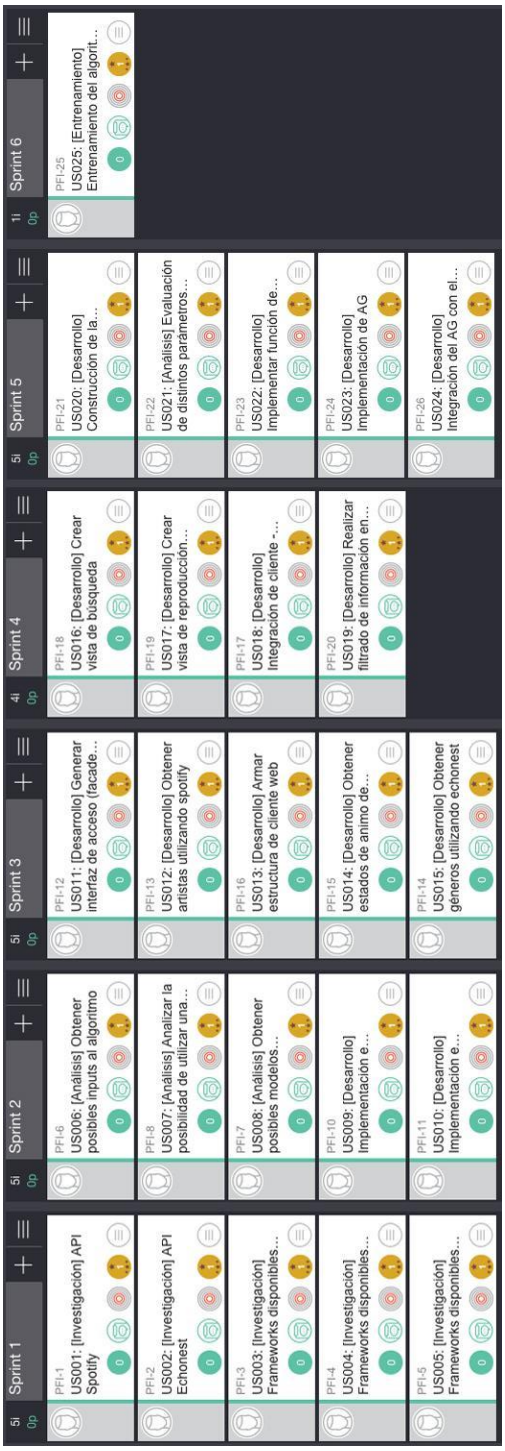

Figura 16: Sprints definidos en Vivify Scrum

4.2.8 LIMITACIONES ENCONTRADAS

4.2.8.1 JBoss

Si bien el servidor de aplicaciones utilizado fue extremadamente útil, nos topamos con varias dificultades muy difíciles de resolver que hicieron que nuestro trabajo se complique en vez de que se aliviane. Nos dimos cuenta que la versión de RestEasy que provee el servidor de aplicaciones en sus librerías nativas, era antigua y que no permitía configurar los servicios rest para que sean consumibles desde otras aplicaciones (Spotify).

Cuando deseamos actualizar la versión de la librería de RestEasy, el proceso rompió toda la configuración del ambiente y debimos reinstalar todo el servidor de aplicaciones. Se consumió mucho tiempo leyendo documentación sobre el upgrade de versión, hasta que de manera poco convencional pudimos dejar el ambiente en condiciones.

4.2.8.2 Spotify API

Si bien la API de Spotify provee varios servicios, encontramos que no expone metadata con respecto a los géneros musicales y estados de ánimo. Por este motivo tuvimos la necesidad de obtener los mismos de EchoNest de una forma más costosa.

4.2.8.3 EchoNest

Utilizar EchoNest fue un factor clave para nuestro trabajo. Fue sin duda la herramienta con más utilidad. Aun así encontramos ciertas limitaciones que nos costaron varias semanas de investigación y trabajo.

- EchoNest permite solo 20 llamadas por minuto a sus servidores, con un máximo de 100 resultados por llamada. Esto hace imposible obtener todas las canciones provenientes

de un género musical.
Si se quisiera resolver el inconveniente de las llamadas por minuto, EchoNest provee una base de datos local en formato JSON. Pero la información en estas listas está desactualizada e incompleta, sin mencionar que son más de 450 GB de información.
Entramos en la dicotomía de instalar la base de datos local o realizar llamadas online.

- Encontramos que la API no permite combinar estados de ánimo con la búsqueda de artistas, por eso mismo debímos permitir combinar géneros con los estados de ánimo.
- EchoNest no expone su método de cálculo de popularidad, nos hemos topado con que canciones que, o bien poseen índice de popularidad ridículo, o bien ni siquiera tienen asignado un valor de popularidad.
- No hay suficiente información para artistas locales, al parecer EchoNest posee su mayor cantidad de información relacionada a artistas internacionales.

4.3 CONCLUSIÓN

Una vez finalizado el desarrollo del prototipo nos dimos cuenta que la implementación de un algoritmo genético para el armado de listas musicales tiene un gran potencial y fue de gran utilidad para nuestro sistema.

Dada la cantidad de información disponible hoy en día sobre artistas, canciones y géneros musicales pueden crearse numerosos índices e indicadores que ayuden a crear un set de música con mayor precisión.

Implementar un AG propio puede ser muy complicado, pero utilizando el framework provisto por jGAP, realmente hizo que implementar nuestro prototipo sea más fácil y legible.

Tal vez la parte más complicada de implementar un AG es hallar la configuración indicada, se necesita mucho entrenamiento y tiempo para evaluar todas las posibilidades que forman parte del proceso. Si bien realizamos varias pruebas para encontrar la mejor configuración, no encontramos forma de probar que es la configuración ideal.

5 CONCLUSIONES

A lo largo de este trabajo se modeló e implementó el uso de un algoritmo genético el cual tiene como objetivo la construcción del armado de una lista de reproducción musical.

Adicionalmente se realizó la construcción de un prototipo que consume el algoritmo desarrollado utilizando distintas tecnologías y frameworks. Durante el transcurso de este desarrollo y en su finalización, se obtuvieron los siguientes resultados:

- Si bien las técnicas y la teoría de los algoritmos genéticos no fueron pensadas para entidades complejas como una lista de reproducción musical, pudimos abstraernos de la teoría y utilizar el concepto exitosamente en lo que deseábamos modelar.
- Aunque las metodologías ágiles no estén relacionadas con las técnicas de inteligencia artificial, encontramos de gran utilidad el uso de la técnica SCRUM para la gestión del desarrollo de nuestro prototipo; comprobando que la misma aplica a cualquier industria y cualquier actividad.
- Configurar el algoritmo no es una tarea muy simple como parece en la teoría. Si bien nuestro entrenamiento arrojó resultados positivos, se observó que es mejor utilizar técnicas más avanzadas para realizar dicha tarea. Hay desarrolladores que utilizan otro algoritmo genético para la búsqueda de una buena configuración.
- Encontramos que las librerías y los servicios utilizados para la obtención de datos fueron de gran utilidad, pero se hubiesen obtenido mejores resultados con un banco de datos más personalizado a nuestra necesidad. Probablemente una base de datos local propia con metadata, solventaría dicho inconveniente, aunque hubiese sido muy costosa su implementación.

- Al ser esta nuestra primera experiencia con algoritmos genéticos y dada la gran cantidad de información a procesar, quedamos sorprendidos con el corto lapso de tiempo consumido en realizar las 350 evoluciones definidas. Creemos que los tiempos de respuesta del prototipo son adecuados para la utilización del sistema por parte de cualquier usuario.
- Comprobamos que la implementación de un algoritmo genético es relativamente sencilla, pero definir el problema a resolver y una función de ajuste que evalúe las posibles soluciones son el verdadero desafío a resolver.
- Como resultado de la investigación, encontramos que ya existían aplicaciones que intentaban resolver una problemática similar a la nuestra. Cuando planteamos la necesidad de desarrollar esta aplicación, no conocíamos la existencia de las mismas. Más allá de este hallazgo, pudimos modelar una aplicación novedosa para el ámbito musical.
- Al intentar definir las variables de ajuste, encontramos que aquellas que son definidos por el usuario son las que más aportan a la construcción la lista óptima. El resto de los índices propuestos son útiles en su medida, ya que tienen una dependencia con lo que nosotros definimos como ideal que puede no ser lo mejor.

5.1 FUTURAS LÍNEAS DE INVESTIGACIÓN

Como quedó claro durante el desarrollo de este proyecto final de ingeniería, el prototipo se realizó hasta la primera etapa definida que era el armado del algoritmo genético en base a la obtención de las canciones. Luego, se pueden definir diversos puntos, como futuras líneas de investigación, con los cuales la obtención del producto final se mejoraría notablemente.

Algunos de los puntos se detallan a continuación.

- Consideración de nuevas variables con datos ya existentes para incluir en la función de ajuste. Algunas de ellas, por ejemplo, podrían ser

distribución de géneros, géneros preferidos y distribución de estados de ánimo.

- Retroalimentación de datos con estadísticas de uso por parte del usuario. Se podrían obtener los inputs del usuario como para conocer preferencias, o guardar listas de reproducción favoritas, o guardar inputs favoritos. Además se podría obtener eventos de uso, como por ejemplo si el usuario omite una canción de la lista de reproducción, guardarlo para no volver a incluir la misma en futuras listas musicales.
- Consumir una base de datos local con toda la metadata necesaria y que la misma se sincronice con algún servicio que provea datos actualizados.
- Evaluar la posibilidad de utilizar un método más efectivo para obtener la mejor configuración posible del algoritmo genético. Existen técnicas que se pueden aplicar para encontrar una configuración óptima, también se puede implementar un algoritmo genético que se encargue de buscar la misma.

6 REFERENCIAS BIBLIOGRÁFICAS

Alliance, S. (2001). "Web oficial de la organización Scrum Alliance." Retrieved 22/09/2015, from https://www.scrumalliance.org/.

AngularJS. (2010). "Web oficial de la librería." Retrieved 12/09/2015, from https://angularjs.org/.

Apaza, V. M. A. (2013). Modelo de un Algoritmo Genético con selección discriminatoria de individuos bajo un esquema de ponderaciín de probabilidades de mutación, Universidad Católica San Pablo.

AppleInsider. (2015). "Apple's Beats Music relaunch might hurt Spotify more than others, data suggests." Retrieved 29/05/2015, from http://appleinsider.com/articles/15/05/05/apples-beats-music-relaunch-might-hurt-spotify-more-than-others-data-suggests.

Bertrand Meyer, M. O. y. A. L. (2006). "Extreme Programming and Test driven developement." Retrieved 29/10/2015, from http://se.inf.ethz.ch/old/teaching/ws2006/0239/slides/TC2006_XP_TDD.pdf.

Craane, J. (2009). "Introduction to Genetic Algorithms with JGAP." Retrieved 04/01/2016, from http://jcraane.blogspot.com.ar/2009/02/introduction-to-genetic-algorithms-with.html.

EchoNest. (2012). "Web Oficial de EchoNest & EchoPrint." Retrieved 25/06/2015, from http://echoprint.me/, from http://echoprint.me/how, from http://echoprint.me/data, from http://the.echonest.com/, from http://the.echonest.com/company/.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Reading, Mass. ; Wokingham, Addison-Wesley.

Grebenc, P. (2003). "JID3 - A JavaTM ID3 Class Library Implementation." Retrieved 25/06/2015, from https://blinkenlights.org/jid3/, from https://blinkenlights.org/jid3/samples.html, from https://blinkenlights.org/jid3/javadoc/index.html.

Groove. (2016). "Web oficial de la tecnología luego de la compra de Microsoft." Retrieved 05/03/2016, from https://www.microsoft.com/en-us/groove.

Groove Music App, Z. (2014). "Web Oficial de la aplicación." Retrieved 20/04/2015, from http://www.groovemusicapp.com/.

Holland, J. H. (1992). Adaptation in Natural and Artificial Systems.

JAudioTagger. (2004). "Web Oficial de la librería." Retrieved 25/06/2015, from http://www.jthink.net/jaudiotagger/index.jsp, from http://www.jthink.net/jaudiotagger/tagmapping.html, from http://www.jthink.net/jaudiotagger/examples_id3.jsp,

fromhttp://www.jthink.net/jaudiotagger/examples_oggvorbis.jsp, from http://www.jthink.net/jaudiotagger/maven/apidocs/index.html.

JukeFox. (2011). "Jukefox - a smart music player for Android. Web Oficial de la tecnología." Retrieved 25/05/2015, from http://www.jukefox.org/.

Kent Beck, M. B., Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Shwaber, Jeff Sutherland y Dave Thomas. (2001). "Manifiesto por el Desarrollo Ágil de Software." Retrieved 08/11/2015, from http://www.agilemanifesto.org/iso/es/.

López, J. C. (2010). "Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA." Retrieved 04/01/2016, from http://www.adictosaltrabajo.com/tutoriales/jgap/.

Marczyk, A. (2004). "Algoritmos genéticos y computación evolutiva." Retrieved 28/11/2015, from http://the-geek.org/docs/algen/.

Media, M. (2011). "Web Oficial de Empresa MixBerry Media." Retrieved 02/03/2016, from http://www.mixberrymedia.com/.

MoodAgent. (2015). "Web Oficial de MoodAgent." Retrieved 22/05/2015, from http://www.moodagent.com/, from http://www.moodagent.com/blog/a-farewell-to-apps.

MusicBrainz. (2000). "Web oficial de la tecnología." Retrieved 06/11/2015, from https://musicbrainz.org/.

Neurowear. (2012). "Web oficial de la empresa.", from http://neurowear.com/about/, from http://neurowear.com/faq/.

PlayerPro. (2012). "Sitio Oficial de la aplicación." 11/06/2015, from http://www.aplayerpro.com/.

Pose, M. G. (2006). Introducción a los Algoritmos Genéticos. Universidade da Coruña.

ProyectoLatin. (2013). "Conceptos básicos de algoritmo evolutivo." Retrieved 11/12/2015, from http://escritura.proyectolatin.org/inteligencia-artificial/conceptois-basicos-de-algoritmos-evolutivos/.

Rossi, B. (2013). Seminario de Integración Profesional II - Clase II: Metodologías Ágiles.

Rotstan, D. K. M. y. N. (2002). "Getting Started With JGAP." Retrieved 04/01/2016, from http://jgap.sourceforge.net/doc/tutorial.html.

Rousselot, J. (2014). Materia Arquitectura de Aplicaciones. Clase IV: Arquitectura.

Rousselot, J. (2014). Materia: Integración de Aplicaciones. Clase II: Servidor de Aplicaciones.

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms.

Sanchez, A. (2016). "Microsoft se hace con Groove, la competencia de Apple Music en iOS." Retrieved 05/03/2016, from http://hipertextual.com/2016/02/microsoft-compra-groove.

Spilka, S. (2015). "Apple Watch and Beats Music Unite in Pulse-Activated Playlist App." Retrieved 29/05/2015, from http://www.psfk.com/2015/04/apple-watch-beats-music-heatbeatsmusic-app-exercise.html.

Spotify. (2009). "Spotify Web API." Retrieved 23/04/2015, from https://developer.spotify.com/web-api/.

Spotify. (2009). "Web oficial de la aplicación." Retrieved 23/04/2015, from https://www.spotify.com/ar/.

Strikingly. (2014). "Setting the Mood: Moodagent and the Music App Landscape." from http://www.strikingly.com/blog/moodagent-music-app/.

Team, I. L. (2015). "Apple Watch May Help Make Playlists Based On Your Pulse." Retrieved 30/05/2015, from https://www.ipglab.com/2015/04/15/apple-watch-may-help-make-playlists-based-on-your-pulse/.

Truyol, J. A. d. l. P. y. A. P. (2005). "Algoritmos Genéticos." from http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf.

Yeoman. (2012). "Web oficial de la herramienta." Retrieved 18/09/2015, from http://yeoman.io/.

7 ANEXOS

7.1 ANEXO 1 – TÉCNCAS DE LAS METODOLOGÍAS ÁGILES

Dentro de las técnicas podemos encontrar algunas que detallaremos a continuación como SCRUM, XP y TDD.

Scrum.

Scrum es un framework[34] iterativo e incremental para la gestión ágil de proyectos, para el desarrollo de productos y aplicaciones. La propuesta consiste en organizar el trabajo en ciclos llamados SPRINT. Estos son ciclos sucesivos, iterativos que van de 2 a 4 semanas, la duración es fija (se haya terminado o no el trabajo) y cuenta con equipos autogestionados. (Scrum Alliance, 2001)

Figura 17: Gráfico explicativo sobre ciclo de SCRUM. (Scrum Alliance, 2001)

Sus elementos son:

- Backlog: Conjunto total de las tareas identificadas hasta el momento, a realizarse durante el desarrollo. Deben ser priorizadas en una lista, que cambia continuamente y son re-priorizadas.
- Sprint Planning: Priorización de tareas y elementos del backlog que serán utilizadas en el Sprint Backlog.

[34] Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

- Sprint Backlog: Subconjunto de tareas del backlog que conforman el alcance definido para completar un Sprint.
- Sprint: período (2 a 4 semanas) en el cual se desarrollan las tareas definidas y priorizadas. Durante el sprint no se pueden agregar/eliminar tareas o cambiarles la prioridad.
- Daily Meetings: reuniones diarias de corta duración (15 minutos de pie, cara a cara), para evaluar el progreso del trabajo realizado. Cada integrante informa qué hizo desde la última reunión, qué tiene planificado hacer y qué impedimentos tuvo para completar las tareas.

Roles en Scrum:

- Product Owner: Es quien representa al usuario, él es responsable de:
  - Definir las funcionalidades del producto.
  - Considerar input de los usuarios, stakeholders y otras partes interesadas.
  - Decidir fechas, release y contenidos.
  - Decidir las prioridades a implementar.
  - Priorizar y refinar continuamente el backlog.
  - Aceptar o rechazar los entregables.
  - Interactuar permanentemente con el equipo.
- Scrum Master: Debe ser una persona distinta al Product Owner. Es el responsable de:
  - Dedicarse al proyecto a tiempo completo.
  - Ayudar al grupo a aplicar las prácticas de SCRUM.
  - Proteger al equipo de interferencias externas.
  - Aportar ideas creativas.
  - Colaborar con el equipo a encontrar soluciones.
  - Fomentar la cooperación.

- Mejorar la productividad.
- Resolver los inconvenientes.

- Team Member: Grupo multidisciplinario, multifuncional, con todas las competencias y habilidades necesarias para entregar un producto en cada sprint. Es autoorganizado, autogestionado y con alto grado de autonomía, responsabilidad y compromiso. Está conformado por 5 a 9 integrantes. Es responsable de:
  - Construir el producto que va a usar el cliente.
  - Acordar con el PO el alcance de cada iteración y hacerlo.
  - Identificar el objetivo de cada iteración.
  - Organizar, planificar, estimar su propio trabajo.
  - Mostrar los resultados de lo realizado durante el sprint.
  - Entregar software de alta calidad.

(Scrum Alliance, 2001)

XP: Extreme Programming.

La programación extrema es una técnica de desarrollo ágil basada en una serie de valores y una docena de prácticas que propician un aumento en la productividad a la hora de generar software. Sus objetivos son la satisfacción del cliente y el potenciar al máximo el trabajo en equipo. XP define cuatro variables para cualquier proyecto de software: Costo, Tiempo, Calidad y Alcance y especifica que solo 3 de ellas pueden ser fijadas por fuerzas externas al proyecto, mientras que el valor de la variable libre será establecido por el equipo de desarrollo. (Meyer *et al*, 2006)

Sus principios son:

- Simplicidad (diseño simple, código simple y comentado, documentación simplificada).
- Retroalimentación (ciclos de desarrollo cortos, pruebas unitarias, integración continua).

- Comunicación (programación en parejas, comunicación continua con el cliente).
- Valentía o Coraje (expresar las opiniones sobre el proyecto, reconstruir el código cuando fuera necesario).

(Bibiana Rossi, 2013)

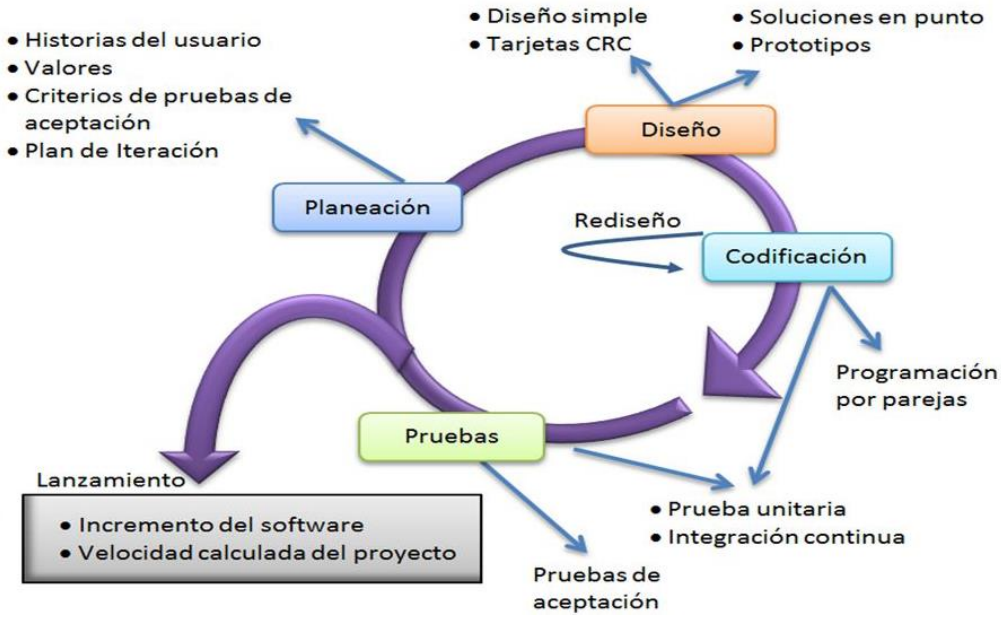

Figura 18: Ciclo de vida de XP

Ventajas:

- Pruebas unitarias en el código.
- La integración continua es aceptada y recomendada para evitar catástrofes.
- La simplicidad y refactorización es reconocido como un factor saludable en la práctica de programación.
- El cliente también se integra con el enfoque humano ya que participa cotidianamente en el desarrollo

Aspectos Controversiales:

- No es aconsejable si la tecnología o entorno no permite realizar integraciones frecuentes.
- No se recomienda XP si la distribución física impide la programación en pares o si no todos los programadores se encuentran en el mismo sitio.
- El exceso de pruebas retrasa el desarrollo.
- El diseño simple solo aplica a proyectos simples.
- La programación en pares consume mayor tiempo y recursos.

(Bibiana Rossi, 2013)

TDD: Test Driven Developement.

El desarrollo guiado por las pruebas fue introducido por Kent Beck, creador de la técnica XP. Es una forma para desarrollar y también de proporcionar la especificación de los requisitos. Por lo tanto es un método de diseño. Cada prueba será suficientemente pequeña como para que permita determinar unívocamente si el código probado pasa o no la verificación que ésta le impone. (Meyer *et al*, 2006)

El ciclo de desarrollo está compuesto por 4 fases:

- Escribir la Prueba: Se elige un requisito y se entienden las especificaciones. Se escribe la prueba correctamente y se verifica que falla.
- Escribir el código: Se debe escribir el código que pase la prueba diseñada. EL desarrollo se debe basar en las interfaces.
- Ejecutar pruebas automatizadas: Se ejecutan las pruebas y se comprueba que pasen. Si pasan, se continúa con la fase 4, caso contrario se vuelve a revisar el código.
- Refactorizar: Limpieza de código y mejora de diseño. Se vuelven a ejecutar los pasos de prueba para comprobar que no se han introducido errores, actualizar la lista de requerimientos.

(Bibiana Rossi, 2013)

Beneficios:

- Se gana significativamente en calidad.
- Los bugs repetidos son eliminados por su rápida detección.
- Las pruebas generadas sirven como documentación del sistema.
- Utilizado correctamente, se asegura que todo el código escrito esté cubierto por una prueba.

Limitaciones:

- Es complicado realizar las pruebas de código cuando se trabaja con la base de datos.
- Interfaces gráficas de usuario.
- Tiempo consumido en pruebas y no en el desarrollo core de la aplicación.

7.2 ANEXO 2 – USER STORIES IDENTIFICADAS

US001: [Investigación] API Spotify.

Como desarrollador del prototipo quiero estudiar la documentación de la API de Spotify para poder implementar el servicio al sistema.

Criterios de aceptación.

- Documentación de Spotify leída
- Conocimientos sobre conexión con API
- Investigar posibles librerías a utilizar

US002: [Investigación] API EchoNest

Como desarrollador del prototipo, quiero estudiar la documentación de la Api de EchoNest para poder implementar el servicio al sistema.

Criterios de aceptación.

- Documentación de EchoNest leída

- Conocimientos sobre conexión con API
- Investigar posibles librerías a utilizar

US003: [Investigación] Frameworks disponibles para implementación de servicio REST

Como desarrollador del prototipo deseo conocer distintos frameworks disponibles en el servidor de aplicaciones para poder conectarme con los servicios REST necesarios.

Criterios de aceptación.

- Investigación de Frameworks
- Implementación/Upgrade en servidor de aplicaciones

US004: [Investigación] Frameworks disponibles para la implementación de un algoritmo genético

Como desarrollador del prototipo quiero investigar distintos frameworks disponibles sobre algoritmos genéticos para aplicar el que mejor se adapte a mis necesidades en el sistema.

Criterios de aceptación.

- Investigación de distintas librerías.
- Verificar funcionalidad de librerías para el prototipo.

US005: Frameworks disponibles para scaffolding y construcción de cliente web

Como desarrollador del cliente web necesito investigar frameworks de scaffolding para facilitar el armado del esqueleto del cliente y manejar las distintas versiones o dependencias de las librerías a utilizar.

Criterios de aceptación.

- Instalación de framework.
- Framework funcionando.
- Creación de esqueleto web.

US006: [Análisis] Obtener posibles inputs al algoritmo

Como desarrollador del prototipo necesito pensar distintos inputs al algoritmo para optimizar el armado de las listas de reproducción.

Criterios de aceptación.

Se desean obtener al menos 3 distintos inputs al proceso para optimizar el armado de las listas de reproducción.

US007: [Análisis] Analizar la posibilidad de utilizar una librería para la implementación del AG

Como desarrollador del prototipo quiero analizar el costo de usar algunas de las librerías ya investigadas contra el costo de implementar un algoritmo genético propio para agilizar el desarrollo.

Criterios de aceptación.

Se debe realizar la prueba conceptual de la librería y analizar si es mejor la utilización de la misma o el desarrollo propio del algoritmo.

US008: [Análisis] Obtener posibles modelos conceptuales de la arquitectura/patrones a utilizar para la implementación

Como desarrollador del prototipo necesito analizar que arquitectura es la más conveniente para el sistema para que el mismo sea eficiente y amigable para el usuario

Criterios de aceptación.

Como salida se deberá decidir qué arquitectura del sistema será utilizada.

US009: [Desarrollo] Implementación e integración con API Spotify

Como desarrollador del prototipo necesito integrar mi sistema con la API de Spotify para poder obtener los datos necesarios para el funcionamiento de mi algoritmo.

Criterios de aceptación.

El sistema debe integrarse y comunicarse correctamente con Spotify.

US010: [Desarrollo] Implementación e integración con API EchoNest

Como desarrollador del prototipo preciso integrar mi sistema con la API de EchoNest para poder obtener los datos necesarios para el funcionamiento de mi algoritmo.

Criterios de aceptación.

El sistema debe integrarse y comunicarse correctamente con EchoNest.

US011: [Desarrollo] Generar interfaz de acceso (facade) con la comunicación entre las APIs

Como desarrollador del prototipo quiero generar una interfaz de acceso para que sea la única capa encargada de comunicarse con el cliente y con el servidor.

Criterios de aceptación.

La interfaz de acceso debe comunicarse correctamente con los servicios del servidor, sin necesidad de comunicarse con el cliente (aún no generado).

US012: [Desarrollo] Obtener artistas utilizando spotify

Como desarrollador del prototipo necesito obtener los artistas desde la API de Spotify para poder sugerir los mismos en la vista de búsqueda del usuario.

Criterios de aceptación.

Requerimiento a servicio de Spotify con obtención de todos los artistas.

US013: [Desarrollo] Armar estructura de cliente web

Como desarrollador del prototipo necesito generar la estructura del cliente web para que el usuario pueda conectarse a la aplicación.

Criterios de aceptación.

Estructura web armada.

US014: [Desarrollo] Obtener estados de ánimo de EchoNest

Como desarrollador del prototipo necesito obtener los estados de ánimo desde la API de EchoNest para poder sugerir los mismos en la vista de búsqueda del usuario.

Criterios de aceptación.

Requerimiento a servicio de EchoNest con obtención de todos los estados de ánimo.

US015: [Desarrollo] Obtener géneros utilizando echonest

Como desarrollador del prototipo necesito obtener todos los géneros desde la API de EchoNest para poder sugerir los mismos en la vista de búsqueda del usuario.

Criterios de aceptación.

Requerimiento a servicio de EchoNest con obtención de todos los estados de ánimo.

US016: [Desarrollo] Crear vista de búsqueda

Como desarrollador del prototipo necesito crear la vista de búsqueda en AngularJS para que el usuario pueda ingresar los parámetros de entrada al sistema.

Criterios de aceptación.

Vista de búsqueda finalizada con artistas, géneros, estados de ánimo, duración de lista de reproducción y duración promedia deseada de cada canción.

US017: [Desarrollo] Crear vista de reproducción utilizando HTML Widget de spotify

Como desarrollador del prototipo necesito generar la vista en AngularJS para poder reproducir la lista de reproducción generada por el algoritmo.

Criterios de aceptación.

Vista de reproducción finalizada con HTML Widget de Spotify.

US018: [Desarrollo] Integración de cliente - servidor

Como desarrollador del prototipo necesito integrar lo desarrollado en el servidor con las vistas del cliente para finalizar el sistema y probar que lo realizado funciona correctamente.

Criterios de aceptación.

Cliente integrado con el servidor a través de la interfaz de acceso (facade).

US019: [Desarrollo] Realizar filtrado de información en base a los parámetros ingresados por el usuario y obtener lista de reproducción sin AG

Como desarrollador del prototipo necesito ingresar información en el cliente y obtener la lista de reproducción en base a los parámetros para poder probar la integración del cliente con el servidor.

Criterios de aceptación.

Lista de reproducción en vista del cliente en base a los parámetros ingresados por el usuario.

US020: [Desarrollo] Construcción de la configuración del AG

Como desarrollador del prototipo necesito generar la configuración del algoritmo genético para que éste funcione según lo deseado.

Criterios de aceptación.

Algoritmo genético configurado correctamente.

US021: [Análisis] Evaluación de distintos parámetros ponderadores para la función de ajuste

Como desarrollador del prototipo necesito evaluar cómo impactan los parámetros ponderadores definidos para poder verificar que la función de ajuste optimice la lista de reproducción.

Criterios de aceptación.

Parámetros ponderadores validados correctamente.

US022: [Desarrollo] Implementar función de ajuste

Como desarrollador del prototipo necesito implementar la función de ajuste para que el algoritmo genético funcione correctamente.

Criterios de aceptación.

Función de ajuste implementada correctamente.

US023: [Desarrollo] Implementación de AG

Como desarrollador del prototipo tengo que implementar el algoritmo genético para devolver la mejor lista de reproducción posible.

Criterios de aceptación.

Algoritmo genético implementado correctamente, teniendo en cuenta su función de ajuste y su configuración definida.

US024: [Desarrollo] Integración del AG con el resto de los componentes

Como desarrollador del prototipo necesito integrar el algoritmo genético con el resto de los componentes para finalizar el prototipo.

Criterios de aceptación.

Algoritmo genético integrado con la interfaz de usuario.

US025: [Entrenamiento] Entrenamiento del algoritmo genético

Como desarrollador del prototipo necesito entrenar mi aplicación para poder obtener la mejor configuración posible en el algoritmo genético.

Criterios de aceptación.

Configuración óptima del algoritmo genético obtenida.

8 TABLA DE ILUSTRACIONES

Figura 1: Estado del arte investigado 11
Figura 2: Usuario utilizando el dispositivo lector de las ondas cerebrales..... 16
Figura 3: Explicación gráfica de la herramienta. 26
Figura 4: Ciclo básico de un algoritmo genético. (ProyectoLatin, 2013) 37
Figura 5: Individuo Genético Binario (Gestal Pose, 2007) 40
Figura 6: Cruce de un punto (Gestal Pose, 2007) 43
Figura 7: Cruce de n puntos (Gestal Pose, 2007) 43
Figura 8: Cruce uniforme(Gestal Pose, 2007) 44
Figura 9: Arquitectura MVC de AngularJS (AngularJS, 2010) 55
Figura 10: Comparación de funcionamiento en AngularJS vs Otros (AngularJS, 2010)..... 56
Figura 11: Diagrama de clases en JGap (López, 2010). 58
Figura 12: Diagrama interacción cliente-servidor. 62
Figura 13: Diagrama de clases de la implementación del Algoritmo Genético. 63
Figura 14: Cromosoma uno 77
Figura 15: Cromosoma dos 77
Figura 16: Sprints definidos en Vivify Scrum..... 85
Figura 17: Gráfico explicativo sobre ciclo de SCRUM. (Scrum Alliance, 2001) 95
Figura 18: Ciclo de vida de XP 98
Tabla I: Metodologías de desarrollo 32
Tabla II: Filtros de entrada a pedido EchoNest 67
Tabla III: Información de las pistas utilizadas para el algoritmo 68
Tabla IV: Configuración óptima del algoritmo genético..... 70

9 GLOSARIO

Framework - Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Metadata - Datos que describen otros datos. En general son datos que describen el contenido informativo de un objeto.

Store - Mercado de aplicaciones en el mundo de los teléfonos inteligentes. Allí se pueden descargar distintas herramientas para el teléfono.

Streaming - Distribución digital de multimedia (videos, canciones) a través de una red de computadoras, de manera que un usuario consume el producto de manera paralela mientras se descarga.

10 ABREVIATURAS

AGs ALGORITMOS GENÉTICOS

AG ALGORITMO GENÉTICO

YO YEOMAN

BPM PULSACIONES POR MINUTO

API INTERFAZ DE PROGRAMACIÓN DE LA APLICACIÓN

SDK CONJUNTO DE HERRAMIENTAS PARA EL DESARROLLO DE SOFTWARE