

# **ESCUELA SUPERIOR DE INGENIERÍA**

## **INGENIERÍA INFORMÁTICA**

**jUCAblox**  
**jukebox Social**

AUTOR: Sergio Ruiz Piulestán

Cádiz, 14 de julio de 2017



# **ESCUELA SUPERIOR DE INGENIERÍA**

## **INGENIERÍA INFORMÁTICA**

**jUCAblox**  
**jukebox Social**

AUTOR: Sergio Ruiz Piulestán  
DIRECTOR: Iván Ruiz Rube  
Cádiz, 14 de julio de 2017



## ***Agradecimientos***

Quisiera agradecer a mis padres, amigos, familia, y sobre todo a Iván Ruiz, mi tutor del pfc, por el tiempo y apoyo dedicado para poder realizar el proyecto.



# Índice general

<b>Parte I – Prolegómeno .....</b>	<b>1</b>
<b>Capítulo 1 – Introducción .....</b>	<b>3</b>
1.1.    Motivación .....	4
1.2.    Alcance.....	4
1.3.    Glosario de Términos.....	4
1.4.    Organización del documento .....	5
<b>Capítulo 2 – Planificación .....</b>	<b>7</b>
2.1.    Metodología de desarrollo.....	7
2.2.    Planificación del proyecto.....	8
2.3.    Organización .....	10
2.4.    Costes.....	11
2.5.    Riesgos.....	13
2.6.    Aseguramiento de calidad.....	14
<b>Parte II – Desarrollo .....</b>	<b>15</b>
<b>Capítulo 3 – Requisitos del Sistema .....</b>	<b>12</b>
3.1.    Situación actual .....	12
3.1.1.        Procesos de Negocio.....	12
3.1.2.        Entorno Tecnológico .....	12
3.2.    Necesidades de Negocio.....	13
3.2.1.        Objetivos de Negocio.....	13
3.3.    Objetivos del Sistema .....	13
3.4.    Catálogo de Requisitos .....	13
3.4.1.        Requisitos funcionales .....	13
3.4.2.        Requisitos no funcionales .....	14
3.4.3.        Reglas de negocio.....	15

3.4.4.	Requisitos de información .....	15
<b>3.5.</b>	<b>Alternativas de Solución .....</b>	<b>16</b>
<b>3.6.</b>	<b>Solución Propuesta .....</b>	<b>16</b>
<b>Capítulo 4 – Análisis del Sistema .....</b>		<b>18</b>
<b>4.1.</b>	<b>Modelo Conceptual.....</b>	<b>18</b>
<b>4.2.</b>	<b>Modelo de Casos de Uso.....</b>	<b>20</b>
4.2.1.	Actores .....	20
4.2.2.	Casos de Uso .....	21
<b>4.3.</b>	<b>Modelo de Comportamiento .....</b>	<b>31</b>
4.3.1.	Acceso y salida del sistema.....	32
4.3.2.	Búsqueda y envío de canciones.....	33
4.3.3.	Gestión de usuarios.....	34
4.3.4.	Gestión de lugares.....	36
4.3.5.	Contrato de operaciones .....	39
<b>4.4.</b>	<b>Modelo de Interfaz de Usuario.....</b>	<b>44</b>
<b>Capítulo 5 – Diseño del Sistema .....</b>		<b>56</b>
<b>5.1.</b>	<b>Arquitectura del Sistema .....</b>	<b>56</b>
5.1.1.	Arquitectura Física.....	56
5.1.2.	Arquitectura Lógica.....	56
<b>5.2.</b>	<b>Diseño Físico de Datos .....</b>	<b>57</b>
<b>5.3.</b>	<b>Diseño detallado de Componentes .....</b>	<b>63</b>
<b>5.4.</b>	<b>Diseño detallado de la Interfaz de Usuario .....</b>	<b>77</b>
<b>Capítulo 6 – Construcción del Sistema .....</b>		<b>78</b>
<b>6.1 .</b>	<b>Entorno de Construcción.....</b>	<b>78</b>
<b>6.2.</b>	<b>Código Fuente.....</b>	<b>79</b>
6.2.1.	Estructura de ficheros .....	79
<b>6.3.</b>	<b>Scripts de Base de datos .....</b>	<b>84</b>
<b>Capítulo 7 .....</b>		<b>86</b>
<b>Pruebas del Sistema .....</b>		<b>86</b>

<b>7.1.</b>	<b>Estrategia .....</b>	<b>86</b>
<b>7.2.</b>	<b>Entorno de Pruebas .....</b>	<b>86</b>
<b>7.3.</b>	<b>Roles .....</b>	<b>86</b>
<b>7.4.</b>	<b>Niveles de Pruebas .....</b>	<b>87</b>
7.4.1.	Pruebas Unitarias.....	87
7.4.2.	Pruebas de Integración .....	87
7.4.3.	Pruebas de Sistema .....	87
7.4.4.	Pruebas de Aceptación.....	93

## **Parte III – Epílogo .....** **94**

### **Capítulo 8 – Manual de implantación y explotación .....** **95**

<b>8.1.</b>	<b>Introducción .....</b>	<b>96</b>
<b>8.2.</b>	<b>Requisitos previos .....</b>	<b>96</b>
<b>8.3.</b>	<b>Inventario de componentes .....</b>	<b>96</b>
<b>8.4.</b>	<b>Procedimientos de instalación .....</b>	<b>97</b>
<b>8.5.</b>	<b>Pruebas de implantación .....</b>	<b>107</b>
<b>8.6.</b>	<b>Procedimientos de operación y nivel de servicio .....</b>	<b>107</b>

### **Capítulo 9 – Manual de usuario .....** **110**

<b>9.1.</b>	<b>Introducción .....</b>	<b>110</b>
<b>9.2.</b>	<b>Características.....</b>	<b>110</b>
<b>9.3.</b>	<b>Requisitos previos .....</b>	<b>110</b>
<b>9.4.</b>	<b>Uso del Sistema .....</b>	<b>111</b>
9.4.1.	Aspecto visual.....	111
9.4.2.	Búsqueda de canciones y envío de peticiones .....	111
9.4.3.	Búsqueda de lugares .....	115
9.4.4.	Registro e inicio de sesión.....	118
9.4.5.	Búsqueda de canciones y envío de peticiones usuarios registrados .....	119
9.4.6.	Búsqueda de lugares de usuarios registrados .....	120
9.4.7.	Área de usuarios.....	121
9.4.8.	Perfil usuario.....	122
9.4.9.	Búsqueda de lugares de usuarios administradores.....	124

### **Capítulo 10 – Conclusiones .....** **130**

<b>Conclusiones .....</b>	<b>130</b>
<b>10.1.    Objetivos alcanzados.....</b>	<b>130</b>
<b>10.2.    Lecciones aprendidas .....</b>	<b>130</b>
<b>10.3.    Trabajo futuro.....</b>	<b>130</b>
<b>Bibliografía .....</b>	<b>134</b>

# Índice de figuras

<b>Figura 2.1 – Fase RUP .....</b>	<b>8</b>
<b>Figura 2.2. Diagramas de Gantt .....</b>	<b>9</b>
<b>Figura 4.1 – Diagrama relaciones UML .....</b>	<b>19</b>
<b>Figura 4.2 – Actores jUCAbox.....</b>	<b>20</b>
<b>Figura 4.3 – Paquetes de casos de uso .....</b>	<b>21</b>
<b>Figura 4.4 – Casos de uso Búsqueda y envío de canciones.....</b>	<b>21</b>
<b>Figura 4.5 – Casos de uso Gestión de usuarios .....</b>	<b>24</b>
<b>Figura 4.6 – Casos de uso Gestionar Lugares .....</b>	<b>27</b>
<b>Figura 4.7 – Modelo comportamiento Login interno .....</b>	<b>32</b>
<b>Figura 4.8 – Modelo comportamiento Login externo .....</b>	<b>32</b>
<b>Figura 4.9 – Modelo comportamiento Logout .....</b>	<b>32</b>
<b>Figura 4.10 – Modelo comportamiento Búsqueda Ítem.....</b>	<b>33</b>
<b>Figura 4.11 – Modelo comportamiento Enviar Canción .....</b>	<b>33</b>
<b>Figura 4.11 – Modelo comportamiento Agregar Artista Favorito.....</b>	<b>34</b>
<b>Figura 4.12 – Modelo comportamiento Registro usuario .....</b>	<b>34</b>
<b>Figura 4.13 – Modelo comportamiento Log de Actividad .....</b>	<b>35</b>
<b>Figura 4.14 – Modelo comportamiento Búsqueda usuarios .....</b>	<b>35</b>
<b>Figura 4.15 – Modelo comportamiento Agregar Amigo .....</b>	<b>35</b>
<b>Figura 4.16 – Modelo comportamiento Chatear .....</b>	<b>36</b>
<b>Figura 4.17 – Modelo comportamiento Editar Información .....</b>	<b>36</b>
<b>Figura 4.18 – Modelo comportamiento Búsqueda de Lugares.....</b>	<b>36</b>
<b>Figura 4.19 – Modelo comportamiento Crear Lugar .....</b>	<b>37</b>
<b>Figura 4.20 – Modelo comportamiento Ver Lugares Favoritos .....</b>	<b>37</b>
<b>Figura 4.21 – Modelo comportamiento Publicar Mensaje .....</b>	<b>38</b>
<b>Figura 4.22 – Modelo comportamiento Crear Playlist.....</b>	<b>38</b>
<b>Figura 4.23 – Modelo comportamiento Validar Canción .....</b>	<b>39</b>
<b>Figura 4.24 – Modelo comportamiento Reproducir Canción .....</b>	<b>39</b>
<b>Figura 4.25 –Prototipado – Home .....</b>	<b>45</b>
<b>Figura 4.26 – Prototipado – Buscar Canción .....</b>	<b>46</b>
<b>Figura 4.27 – Prototipado – Buscar lugar.....</b>	<b>47</b>

<b>Figura 4.28 – Prototipado – Datos lugar .....</b>	<b>48</b>
<b>Figura 4.29 – Prototipado – Playlist .....</b>	<b>48</b>
<b>Figura 4.30 – Prototipado – Top canciones .....</b>	<b>49</b>
<b>Figura 4.31 – Prototipado – Usuarios Lugar.....</b>	<b>49</b>
<b>Figura 4.32 – Prototipado – Mensajes lugar.....</b>	<b>50</b>
<b>Figura 4.33 – Prototipado – Editar información Lugar .....</b>	<b>50</b>
<b>Figura 4.34 – Prototipado – Historial Acciones Usuario .....</b>	<b>51</b>
<b>Figura 4.35 – Prototipado – Amigos de Usuario .....</b>	<b>52</b>
<b>Figura 4.36 – Prototipado – Artistas Favoritos Usuario.....</b>	<b>53</b>
<b>Figura 4.37 – Prototipado – Lugares Favoritos Usuario.....</b>	<b>54</b>
<b>Figura 4.38 – Prototipado – Información Personal Usuario .....</b>	<b>55</b>
<b>Figura 5.1 – Diagrama de componentes.....</b>	<b>57</b>
<b>Figura 5.2 – Tipos de datos MongoDB .....</b>	<b>58</b>
<b>Figura 5.3 – Objeto Usuario.....</b>	<b>58</b>
<b>Figura 5.4 – Objeto de definición de la colección Usuario .....</b>	<b>59</b>
<b>Figura 5.5 – Objeto de definición de la colección Lugar.....</b>	<b>60</b>
<b>Figura 5.6 – Objeto de definición de la colección Playlist .....</b>	<b>61</b>
<b>Figura 5.7 – Objeto de definición de la colección Canción .....</b>	<b>62</b>
<b>Figura 5.8 – Objeto de definición de la colección Log .....</b>	<b>63</b>
<b>Figura 5.9 – Diagrama de secuencia – búsqueda de ítems.....</b>	<b>64</b>
<b>Figura 5.10 – Diagrama de secuencia – enviar canción .....</b>	<b>64</b>
<b>Figura 5.11 – Diagrama de secuencia – añadir artista a favoritos .....</b>	<b>65</b>
<b>Figura 5.12 – Diagrama de secuencia – eliminar artista.....</b>	<b>66</b>
<b>Figura 5.13 – Diagrama de secuencia – registro en el sistema .....</b>	<b>67</b>
<b>Figura 5.14 – Diagrama de secuencia – ver log actividad.....</b>	<b>68</b>
<b>Figura 5.15 – Diagrama de secuencia – buscar usuarios.....</b>	<b>68</b>
<b>Figura 5.16 – Diagrama de secuencia – agregar nuevo amigo .....</b>	<b>69</b>
<b>Figura 5.17 – Diagrama de secuencia – agregar nuevo amigo .....</b>	<b>70</b>
<b>Figura 5.18 – Diagrama de secuencia – editar información.....</b>	<b>71</b>
<b>Figura 5.19 – Diagrama de secuencia – buscar lugares .....</b>	<b>71</b>
<b>Figura 5.20 – Diagrama de secuencia – añadir lugar a favoritos .....</b>	<b>72</b>
<b>Figura 5.21 – Diagrama de secuencia – eliminar lugar favoritos .....</b>	<b>73</b>
<b>Figura 5.22 – Diagrama de secuencia – crear lugar .....</b>	<b>74</b>

<b>Figura 5.23 – Diagrama de secuencia – publicar mensaje .....</b>	<b>74</b>
<b>Figura 5.24 – Diagrama de secuencia – publicar mensaje .....</b>	<b>75</b>
<b>Figura 5.25 – Diagrama de secuencia – reproducir canciones .....</b>	<b>76</b>
<b>Figura 5.26 – Diseño de la pantalla principal jUCAb ox .....</b>	<b>77</b>
<b>Figura 6.1 – Estructura de ficheros de jUCAb ox .....</b>	<b>79</b>
<b>Figura 6.2 – Estructura de ficheros de la API de jUCAb ox .....</b>	<b>80</b>
<b>Figura 6.3 – Petición GET Node.js .....</b>	<b>81</b>
<b>Figura 6.4 – Rutas lugares Node.js .....</b>	<b>82</b>
<b>Figura 6.5 – Estructura de ficheros del Front de jUCAb ox .....</b>	<b>83</b>
<b>Figura 6.6 – Modelo de Usuario .....</b>	<b>84</b>
<b>Figura 7.1 – jUCAb ox Ipad .....</b>	<b>89</b>
<b>Figura 7.2 – jUCAb ox Iphone 6 .....</b>	<b>89</b>
<b>Figura 7.3 – Navegador Web.....</b>	<b>90</b>
<b>Figura 7.4 – Test SonarQube líneas de código .....</b>	<b>91</b>
<b>Figura 7.5 – Test Global SonarQube .....</b>	<b>91</b>
<b>Figura 7.6 – Peticiones y Tiempos Performance-Analyser .....</b>	<b>92</b>
<b>Figura 7.7 – Tiempo de navegación Performance-Analyser.....</b>	<b>93</b>
<b>Figura 7.8 – Clasificación de peticiones Performance-Analyser.....</b>	<b>93</b>
<b>Figura 8.1 – Amazon Web Service Ec2 .....</b>	<b>97</b>
<b>Figura 8.2 – Key Pairs AWS.....</b>	<b>98</b>
<b>Figura 8.3 – Create Key Pairs AWS.....</b>	<b>98</b>
<b>Figura 8.4 – Key Pairs Creada AWS.....</b>	<b>99</b>
<b>Figura 8.5 – Bitnami AWS Marketplace .....</b>	<b>99</b>
<b>Figura 8.6 – Configuración Bitnami AWS Marketplace .....</b>	<b>100</b>
<b>Figura 8.7 – AWS Instancia .....</b>	<b>101</b>
<b>Figura 8.8 – AWS Instancia Running .....</b>	<b>101</b>
<b>Figura 8.9 – AWS Instancia Get System Log.....</b>	<b>102</b>
<b>Figura 8.10 – AWS Log Bitnami.....</b>	<b>102</b>
<b>Figura 8.11 – Putty Key Generator .....</b>	<b>103</b>
<b>Figura 8.12 – AWS Dns .....</b>	<b>104</b>
<b>Figura 8.13 – Putty Session.....</b>	<b>104</b>
<b>Figura 8.14 – Putty Session SSH Auth .....</b>	<b>105</b>
<b>Figura 8.15 – Putty Inicio Sesión .....</b>	<b>105</b>

<b>Figura 8.16 – Editar puertos AWS .....</b>	<b>106</b>
<b>Figura 8.17 – jUCAb ox en AWS .....</b>	<b>107</b>
<b>Figura 9.1 – Aspecto visual jUCAb ox .....</b>	<b>111</b>
<b>Figura 9.2 – Búsqueda de canción jUCAb ox .....</b>	<b>112</b>
<b>Figura 9.3 – Ficha Artista jUCAb ox .....</b>	<b>112</b>
<b>Figura 9.4 – Búsqueda de canción 2 jUCAb ox .....</b>	<b>113</b>
<b>Figura 9.5 – Enviar canción local jUCAb ox.....</b>	<b>114</b>
<b>Figura 9.6 – Enviar canción local jUCAb ox.....</b>	<b>114</b>
<b>Figura 9.7 – Confirmación de petición de canción jUCAb ox .....</b>	<b>115</b>
<b>Figura 9.8 – Búsqueda de lugares jUCAb ox.....</b>	<b>115</b>
<b>Figura 9.9 – Búsqueda avanzada de lugares jUCAb ox .....</b>	<b>116</b>
<b>Figura 9.9 – Detalle de lugar jUCAb ox.....</b>	<b>116</b>
<b>Figura 9.10 – Canciones solicitadas al local jUCAb ox .....</b>	<b>117</b>
<b>Figura 9.11 – Top Canciones local jUCAb ox .....</b>	<b>117</b>
<b>Figura 9.12 – Registro / Inicio de sesión jUCAb ox .....</b>	<b>118</b>
<b>Figura 9.13 – Pantalla principal usuario Logado jUCAb ox .....</b>	<b>119</b>
<b>Figura 9.13 – Artista usuario registrado jUCAb ox .....</b>	<b>119</b>
<b>Figura 9.13 – Envío de canciones de usuario registrados jUCAb ox ..</b>	<b>120</b>
<b>Figura 9.14 – Búsqueda de lugares de usuarios jUCAb ox .....</b>	<b>120</b>
<b>Figura 9.15 – Detalle de lugar usuarios registrados jUCAb ox .....</b>	<b>121</b>
<b>Figura 9.16 – Acceso área de usuarios jUCAb ox .....</b>	<b>121</b>
<b>Figura 9.17 – Perfil usuario jUCAb ox.....</b>	<b>122</b>
<b>Figura 9.18 – Lista de usuarios jUCAb ox .....</b>	<b>123</b>
<b>Figura 9.19 – Lista de artistas favoritos jUCAb ox .....</b>	<b>123</b>
<b>Figura 9.20 – Lista de lugares favoritos jUCAb ox .....</b>	<b>124</b>
<b>Figura 9.21 – Información personal usuario jUCAb ox.....</b>	<b>124</b>
<b>Figura 9.14 – Búsqueda de lugares de usuario jUCAb ox .....</b>	<b>125</b>
<b>Figura 9.15 – Publicar mensaje desde lugar jUCAb ox .....</b>	<b>125</b>
<b>Figura 9.16 – Editar información acerca del lugar jUCAb ox .....</b>	<b>126</b>
<b>Figura 9.16 – Playlist lugar administrador jUCAb ox.....</b>	<b>126</b>
<b>Figura 9.17 – Playlist lugar administrador jUCAb ox .....</b>	<b>127</b>
<b>Figura 9.18 – Playlist lugar administrador Spotify jUCAb ox .....</b>	<b>127</b>
<b>Figura 9.19 – Validación canción playlist lugar Spotify jUCAb ox .....</b>	<b>128</b>

**Figura 9.20 – Reproducción de canción desde jUCAbbox .....128**



# Índice de tablas

<b>Tabla 2.1 – Tiempo de realización del proyecto.....</b>	<b>10</b>
<b>Tabla 2.2 – Coste de desarrollo del proyecto.....</b>	<b>12</b>
<b>Tabla 2.3 – Coste de mantenimiento anual del proyecto .....</b>	<b>12</b>
<b>Tabla 4.1 – Contrato: Login(user,password).....</b>	<b>40</b>
<b>Tabla 4.2 – Contrato: Logout() .....</b>	<b>40</b>
<b>Tabla 4.3 – Contrato: PeticiónItem(termino) .....</b>	<b>40</b>
<b>Tabla 4.4 – Contrato: EnviarCanción(canción,lugar) .....</b>	<b>41</b>
<b>Tabla 4.5 – Contrato: AgregarArtistaFav(artista) .....</b>	<b>41</b>
<b>Tabla 4.6 – Contrato: SolicitarLogActividad() .....</b>	<b>41</b>
<b>Tabla 4.7 – Contrato: SolicitarUsuarios() .....</b>	<b>42</b>
<b>Tabla 4.8 – Contrato: AgregarAmigo(user) .....</b>	<b>42</b>
<b>Tabla 4.9 – Contrato: solicitarLugares(termino) .....</b>	<b>42</b>
<b>Tabla 4.10 – Contrato: crearLugar(datosLugar) .....</b>	<b>43</b>
<b>Tabla 4.11 – Contrato: publicarMensaje(lugarAdmin,mensaje) .....</b>	<b>43</b>
<b>Tabla 4.12 – Contrato: crearPlaylist(lugarAdmin,playlist).....</b>	<b>43</b>
<b>Tabla 4.13 – Contrato: validarCancion(canción,playlist) .....</b>	<b>44</b>
<b>Tabla 5.1 – Campos entidad usuario .....</b>	<b>60</b>
<b>Tabla 5.2 – Campos entidad lugar.....</b>	<b>61</b>
<b>Tabla 5.3 – Campos entidad playlist .....</b>	<b>62</b>
<b>Tabla 5.4 – Campos entidad canción.....</b>	<b>62</b>
<b>Tabla 5.4 – Campos entidad log .....</b>	<b>63</b>



# **Parte I**

## **Prolegómeno**



# Capítulo 1

## Introducción

El presente documento detalla el proceso que se ha seguido para la creación de la herramienta jUCAbbox.

En la actualidad, muchos locales tienen sistemas de reproducción musical. Cuando un cliente quiere solicitar alguna canción especial, tiene que ir al encargado de la música y decirle el nombre de la canción y el artista que desea. Conllevaría la interrupción de la labor desempeñada por el encargado de la música, así como la dificultad por el ruido y el ambiente en el local de poder transmitir que canción desea.

Por otro lado, está creciendo la necesidad de solicitar canciones antes de realizar un evento social, como una boda, una comunión, o simplemente una fiesta entre amigos. Por lo que el organizador, debería pedir, mediante correos, mensajes y/o personalmente, las canciones que le gustaría que sonaran en el evento.

La herramienta **jUCAbbox** es un Jukebox social, una aplicación pensada para pubs, discotecas y eventos sociales.

Con **jUCAbbox**, se pretende paliar estos problemas. El organizador del evento, o el encargado del sistema musical del local, registrarán su evento/local en la herramienta y automáticamente está disponible para que los usuarios puedan solicitar/Registrar las canciones deseadas. Evitando el tráfico de información que pueda ser susceptible de perderse.

Con **jUCAbbox** se podrá interactuar con el DJ en tiempo real para solicitar canciones, intercambiar mensajes con otros usuarios de la herramienta, e incluso, el propio local podrá añadir promociones para favorecer a que el cliente use la herramienta.

A su vez, se podrán enviar canciones a una lista de reproducción futura, no teniendo que ser en el momento del evento, para su posterior validación. Este listado de canciones, son todas las disponibles en el ámbito de Spotify, ya que la herramienta se conecta a través de un API proporcionado por la misma.

**jUCAbbox** se compone de 3 partes:

- **Búsqueda de canciones:** El usuario podrá buscar todas las canciones disponibles en Spotify, para luego poder enviarla a un evento, de entre todos los registrados en la herramienta. Esta funcionalidad no requiere que el usuario esté registrado en jUCAbbox. Si quiere guardar información de que canciones ha enviado, o guardar lugares y artistas favoritos, entonces sí deberá registrarse.
- **Búsqueda de lugares:** El usuario podrá buscar todos los lugares dados de alta en la herramienta, para poder ver una descripción más detallada del mismo y canciones enviadas y top de canciones entre rangos de fechas. El usuario podrá crear lugares y administrarlos, para controlar las canciones que le envíen y postear mensajes a los usuarios que tengan al lugar como favorito.
- **Usuario:** Tendrá un apartado para ver los datos referente a su usuario. Log de

actividad, lugares y artistas favoritos, y así como los amigos que tenga agregados para empezar un chat si están en línea.

## 1.1. Motivación

Las motivaciones para realizar esta aplicación han sido diversas.

Conocer nuevos frameworks de desarrollo web, nuevos modelos de datos no convencionales, como son los modelos de base de datos no relaciones, y poder explotar toda esa información mediante API Rest full.

El sistema utilizado se define como SPA, Single Page Application, por el cual la navegación es muy fluida y la actualización de los datos es en tiempo real. Por lo cual, es muy importante para el contexto en el que se basa la herramienta. Todo sistema con interacción social requiere de tiempos de respuesta muy cortos, uno de los puntos fuertes de los frameworks utilizados.

La principal motivación de este proyecto es poner en práctica los conocimientos adquiridos a lo largo de las asignaturas de la Titulación de Ingeniería en Informática así como conocer nuevos lenguajes de programación que nos permita ampliar nuestros conocimientos dentro del abanico de posibilidades que nos ofrece la informática hoy en día y aprender a desenvolverse en situaciones desconocidas hasta el momento por uno mismo.

## 1.2. Alcance

El objetivo principal de este proyecto es que el usuario pueda solicitar una canción de manera autónoma y que el administrador del local/evento pueda llevar un control de las canciones enviadas y a su vez poder utilizarlo como portal de promociones hacia los usuarios registrados, pudiendo promover el uso de la aplicación.

De modo que se obtendrán estadísticas de las canciones más solicitadas, para poder adaptar las canciones futuras a los gustos de las personas que frecuenten el local.

## 1.3. Glosario de Términos

A continuación mostramos algunas definiciones, acrónimos y abreviaturas que irán apareciendo a lo largo de la memoria, para así facilitar la comprensión de algunos conceptos.

**Angular:** Framework JavaScript para desarrollo de páginas Web basado en el modelo MVC. La primera versión de este Framework, fue llamada AngularJS, fue creada por google y se desarrollaba utilizando JavaScript. El 15 de Septiembre de 2016, aparecería la versión 2.0, no siendo un simple evolutivo de la primera versión, si no que dejaba de lado JavaScript, para utilizar typescript, como lenguaje base para su implementación. Los sistemas desarrollados por Angular se denominan SPA, en el cual se pasa la lógica de negocio al cliente en vez de al servidor. Mejorando tiempos de espera y latencia con la aplicación.

**TypeScript:** Es un lenguaje de programación de código abierto, desarrollado por Microsoft. Se basa en JavaScript, añadiéndole tipado a las variables y clases. Es un

lenguaje, en el cual su salida compilada es lenguaje JavaScript.

**MongoDB:** Es un sistema de gestión de base de datos no relacionales basadas en JSON, orientado a documentos. Es una base de datos de código abierto nacida en 2007.

**Express:** Es una infraestructura para Node.js en el cual proporciona una serie de características para conectar el servidor Node con la base de datos. Se suele utilizar con MongoDB, mediante mongoose, una librería JavaScript para facilitar la integración.

**Node.js:** Es un intérprete de JavaScript en el lado del servidor basado en eventos basado en el motor V8 desarrollado por Google. Con lo cual permite al usuario desarrollar en el servidor con el mismo lenguaje de programación del cliente, siendo su curva de aprendizaje mínima.

**QlikView:** Es una plataforma de Business Intelligence que sirve, para mostrar y analizar datos de manera inmediata. Se utiliza como cuadro de mandos, para poder reportar datos a gran escala, y que cada usuario pueda filtrar y ordenar la información como desee.

**Git:** Repositorio de código utilizado para la gestión de versiones. Fue creado pensando en la eficiencia y confiabilidad del mantenimiento de aplicaciones, cuando estas tienen un número elevado de ficheros fuente.

**Spotify:** Es una herramienta multiplataforma utilizada para la reproducción de música vía Streaming.

**Api Rest:** Es un sistema por el cual los programadores pueden interactuar con una aplicación y obtener y enviar datos de manera ordenada y planificada. Se basa en obtener y enviar datos mediante JSON y peticiones HTTP.

## 1.4. Organización del documento

El documento está dividido en once capítulos, en los cuales se irán describiendo las distintas etapas que se han seguido a la hora de desarrollar el proyecto.

En este primer capítulo, ofrece una visión algo general de lo que este proyecto abarcará. Definiremos los límites que esperamos alcanzar en la aplicación a desarrollar al final del documento.

En el segundo de los capítulos indicaremos la metodología utilizada para el desarrollo de la herramienta, así como la planificación y organización del proyecto. Abarcaremos los costes y riesgos que conllevarían su implantación.

En los cinco capítulos siguientes se abordarán las distintas fases que comprenden cualquier desarrollo de software bajo el modelo de desarrollo lineal secuencial, que son:

Requisitos del sistema, Análisis, Diseño, Construcción del sistema y Pruebas.

**Requisitos del sistema** Se definen los requisitos necesarios para poder realizar el sistema, así como las necesidades de negocio y estudio de la solución propuesta

**Análisis.** Fase en la cual se estudia la información que deberá manejar el sistema así como los procesos a desarrollar sobre estos y la interconexión entre ambos.

**Diseño.** Etapa consistente en la estructuración de los datos y en el diseño de los

algoritmos que tratarán dichas estructuras de información. En esta fase, se obtiene ya una representación del software previa a la implementación que podrá dar una idea de la calidad que tendrá este.

**Construcción del sistema e Implementación.** Paso en el que se transformará la representación del software generada en el paso anterior a un lenguaje entendible por la máquina.

**Pruebas.** Previa a la implantación del software se harán necesarias unas pruebas que garanticen la calidad del sistema. Esta fase se prorrogará incluso estando ya el software en funcionamiento, ya que no hay mejor prueba que un entorno real capaz de simular todas y cada una de las situaciones existentes.

En el octavo capítulo describiremos un completo manual para la implantación del sistema, donde explicamos cómo realizar el proceso de implantación.

En el noveno capítulo describiremos un completo manual para el usuario, mostrando de manera detallada e ilustrada cómo realizar las distintas funcionalidades.

En el capítulo diez analizaremos cómo ha ido el desarrollo completo del sistema, si se han cumplido las expectativas y los distintos problemas que nos hemos encontrado.

Y por último describiremos el software usado para la bibliografía usada y la licencia aplicada a la aplicación.

# Capítulo 2

## Planificación

### 2.1. Metodología de desarrollo

La metodología utilizada para la elaboración del proyecto se basa en el Racional Unified Process (RUP) [Philippe Kruchten, 1999], que traducido es proceso racional unificado. Este proceso de desarrollo software unido al lenguaje unificado de modelado (UML) [Grady Booch, 1999] componen la metodología habitual estándar para el análisis, implementación y documentación de sistemas orientados a objetos.

El proceso unificado racional (RUP) se utiliza como en nuestro caso para desarrollar sistemas orientados a objetos, basado originalmente en el Proceso unificado (UP) y que utiliza Lenguaje unificado de modelado (UML) para su notación.

Consta de cuatro fases:

- **Inicio:** Se plantea y determina las características del proyecto. Se define el alcance del proyecto, se identifican los riesgos y donde se propone una visión general del proyecto y se procede a la toma de requisitos.
- **Elaboración:** Donde se realiza el diseño, análisis y se van tomando requisitos adicionales. Se seleccionan los casos de uso que permitan definir la arquitectura del sistema, en el cual se diseña una solución preliminar.
- **Desarrollo:** Donde se realiza la implementación. Se le llama también fase de Construcción. Su objetivo es completar la funcionalidad del sistema de acuerdo a las evaluaciones hechas por los usuarios.
- **Transición:** El objetivo de esta fase es asegurar que la herramienta esté disponible a los usuarios finales, verificando que el producto cumpla con los requisitos tomados.

Como hemos mencionamos anteriormente, en el desarrollo de nuestro proyecto seguimos un enfoque orientado a objetos. La programación orientada a objetos es un paradigma de la ingeniería del software que basa la arquitectura de los sistemas en los objetos y sus interacciones. Estos objetos representan las entidades físicas y conceptuales del mundo real y están estrechamente relacionados con la aplicación a construir.

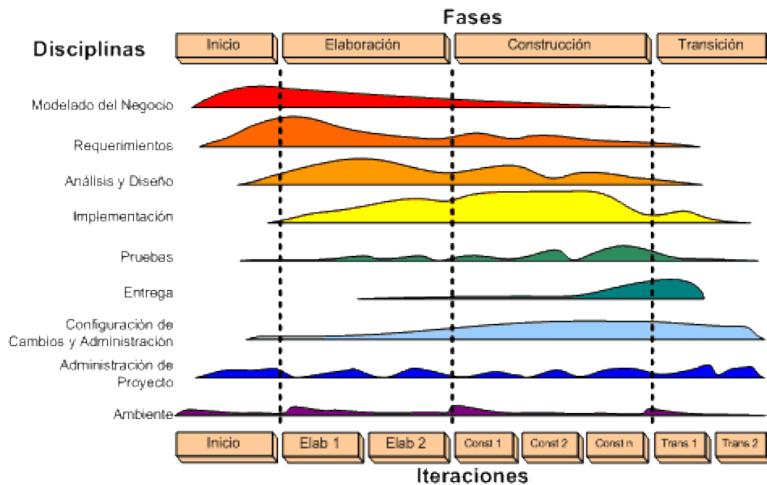


Figura 2.1 – Fase RUP

Basado en el desarrollo incremental, donde todas los roles o actores se involucran más en el proyecto. Haciendo entregas parciales del software, evaluándose como partes individuales, pudiendo reconstruirse partes anteriores conforme se elaboran las nuevas.

De esta forma se determinan las funcionalidades importantes y las comprobaciones son constantes a lo largo del desarrollo, permite una implementación con refinamientos sucesivos y con cada incremento se agrega nueva funcionalidad, se cubren nuevos requisitos o bien se mejora la versión previamente implementada del sistema.

## 2.2. Planificación del proyecto

A continuación se detallan las iteraciones que se ha seguido a lo largo del proyecto. Después de cada iteración se procede a realizar la documentación de cada una de ellas.

### Iteración 0 – Elicitación de requisitos y prototipado

- Se realiza un estudio de los requisitos necesarios para elaborar el sistema. Se realizan varias reuniones con clientes y usuarios potenciales, elaborando una serie de condiciones para que la herramienta sea lo más funcional posible.
- Se realiza un prototipo de la herramienta, organizándose en bloques para su consiguiente desarrollo.

### Iteración 1 – Usuarios/Roles

- Se desarrolla un sistema de autenticación y perfilado de la herramienta. Asignando una función a cada uno de los perfiles dados de alta.

### Iteración 2 – Búsqueda de Canciones

- Se crea la conexión con la API de Spotify. Pudiendo realizar conexiones y extraer y parsear la información para que esté disponible en la herramienta en tiempo real.

### Iteración 3 – Lugares

- Se crea un menú de administración para crear y editar distintos lugares, locales y eventos para que estén disponibles en la herramienta.
- Se crea el módulo de búsqueda de lugares, habilitando la posibilidad de que envíen canciones ya buscadas a un lugar concreto.

- Se crea el módulo de favoritos, añadiendo los usuarios con un local en concreto.

#### Iteración 4 – Envío y Reproducción de canciones

- Se crea la conexión entre la canción y el lugar, habilitando la validación de las mismas.
- Se crea la conexión de Spotify con el local. Añadiendo las canciones en tiempo real a Spotify.
- Se desarrolla un módulo para la reproducción de canciones en Spotify directamente desde la herramienta con actualizaciones en tiempo real.

#### Iteración 5 – Mensajería

- Se crea un módulo para publicar mensajes en el tablón. Dando la opción al administrador del lugar enviar un mensaje a los usuarios que tienen agregado al local como favorito.

#### Iteración 6 – Reporting

- Se desarrollan una serie de informes para que el administrador del local pueda tener constancia de las peticiones que recibe, así como el género de las mismas, para adecuar la música al gusto de los clientes.

Se muestra el diagrama de Gantt, basado en las actividades descritas.

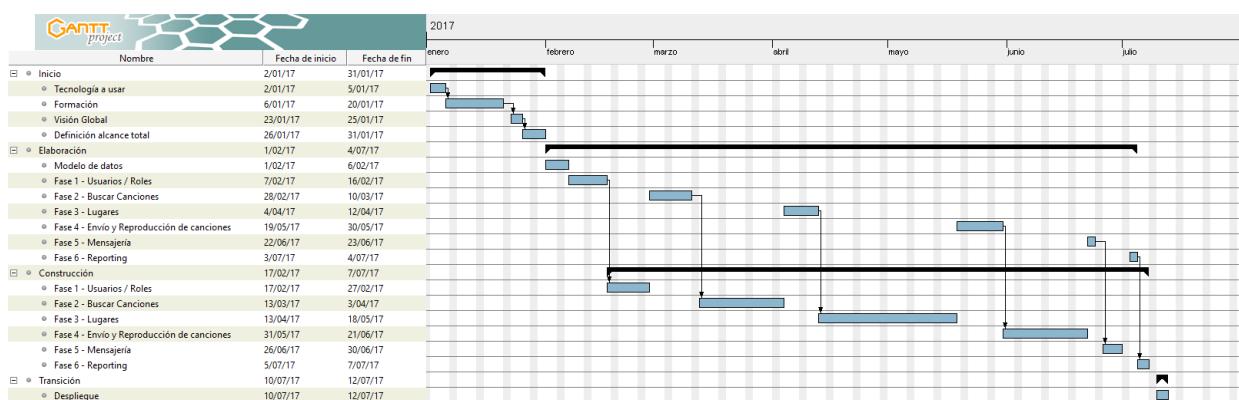


Figura 2.2. Diagramas de Gantt

Se realiza una planificación de los tiempos de realización de cada una de las fases del proyecto. Algunas etapas, por problemas con los sistemas fuente o por mayor complejidad de lo esperado, superan el tiempo estimado, otras, sin embargo, una vez comenzado el desarrollo, se realizaron en un tiempo menor del esperado.

<b>Iteración</b>	<b>Fase</b>	<b>Actividad</b>	<b>Tiempo estimado horas</b>	<b>Tiempo real horas</b>
Iteración 0	Inicio	Tecnología a usar	176	216
		Formación	32	38
		Visión Global	88	106
		Definición alcance	24	27
Iteración 1	Elaboración	Análisis	43	53
		Diseño	28	25
	Construcción	Implementación	48	44
	Construcción	Pruebas	6	6
		Documentación	3	3
Iteración 2	Elaboración	Análisis	56	64
		Diseño	24	30
	Construcción	Implementación	109	109
	Construcción	Pruebas	13	13
		Documentación	6	6
Iteración 3	Elaboración	Análisis	34	43
		Diseño	28	29
	Construcción	Implementación	177	198
	Construcción	Pruebas	21	22
		Documentación	10	10
Iteración 4	Elaboración	Análisis	52	53
		Diseño	17	17
	Construcción	Implementación	109	117
	Construcción	Pruebas	13	15
		Documentación	6	6
Iteración 5	Elaboración	Análisis	12	13
		Diseño	8	9
	Construcción	Implementación	34	37
	Construcción	Pruebas	4	4
		Documentación	2	2
Iteración 6	Elaboración	Análisis	11	11
		Diseño	7	7
	Construcción	Implementación	20	20
	Construcción	Pruebas	2	2
		Documentación	1	1
Iteración 7	Transición	Despliegue	24	25
<b>Total</b>			<b>1.138</b>	<b>1.240</b>

Tabla 2.1 – Tiempo de realización del proyecto

## 2.3. Organización

Para la realización de la aplicación, han estado involucrados los distintos roles:

- Analista:

- Encargado de obtener los requisitos funcionales necesarios para la elaboración de la herramienta
- Desarrollador:
  - Encargado de llevar a cabo los requisitos obtenidos por el analista
- Encargado de pruebas:
  - Encargado de evaluar la herramienta, tanto de fallos técnicos, como requisitos que no se hayan desarrollado de forma correcta.

Puesto que el proyecto no ha sido colaborativo, los roles de analista, desarrollo y encargado de pruebas técnicas, han sido asumidos por el autor del proyecto.

Para poder desarrollar la herramienta jUCAbbox se han utilizado los frameworks propuestos por el MEAN stack [Simon Holmes, 2015].

- **MongoDB:** Sistema de base de datos no relacionales basada en JSON. Utiliza una versión propia llamada BSON.
- **Express:** Framework encargado de conectar MongoDB con el servidor WEB
- **Angular2:** Framework basado en MVC utilizado para la parte Cliente de la aplicación WEB. Se utiliza para el desarrollo de páginas SPA (Single Page Application). Se desarrolla con typescript, una versión libre de JavaScript creada por Microsoft.
- **Node.js:** Framework utilizado como servidor WEB. Se utiliza JavaScript para el desarrollo de las reglas del servidor.

A su vez, se ha empleado otro sistema de gestión de base de datos, para el envío en tiempo real de los mensajes entre usuarios. FireBase, un sistema de base de datos no relacionales basada en JSON. La gran diferencia que tiene con MongoDB es que se trata de un sistema en tiempo real.

A su vez, se han desarrollado una serie de informes, mediante la herramienta QlikView. Una herramienta capaz de transformar la información en tiempo real.

El hardware utilizado para el desarrollo de esta herramienta ha sido un portátil Lenovo ideapad 100 con un procesador intel i5 con 8gb Ram y disco duro de 256 GB.

## 2.4. Costes

El coste final del producto, sería la suma del coste de los recursos humanos, con los recursos materiales. El cálculo se realiza con los tiempos estimados.

Los **recursos materiales** se dividen en dos:

- Recursos software
  - Conjunto de herramientas propuestos por el Mean Stack. Todo lo utilizado es software libre, por lo tanto el coste de los recursos software es

0€.

- Recursos hardware
  - Para el desarrollo de la herramienta se ha utilizado un portátil Lenovo ideapad con un precio de 585€.
  - Para el despliegue de la aplicación, se utilizaría un sistema en la nube basado en AWS de Amazon. El precio para el alojamiento de la aplicación depende del uso de la herramienta, y el impacto social que pueda tener. Todos estos servidores son escalables, es decir, en el momento que se necesiten más recursos se pueden contratar, y automáticamente serían añadido. El modelo utilizado para el despliegue de esta herramienta sería Amazon Ec2 bajo demanda, con un almacenamiento estándar de 50TB/mes de uso de datos, y un servidor Linux t2.medium con unas especificaciones de 2 CPU virtuales y 4 GB de ram, serían 34.41€ al mes.

Los **recursos humanos** necesarios para el desarrollo serían:

- **Analista:** Sueldo medio anual de un analista funcional es de 45.000€. Siendo la hora a 23€
- **Desarrollador:** Sueldo medio anual de un desarrollador web es de 35.000€. Siendo la hora a 18€.
- **Encargado de Pruebas:** Sueldo medio anual de un especialista encargado de realizar pruebas técnicas y funcionales es de 30.000€. Siendo la hora a 15€.

Los costes de cada rol han sido obtenidos a través de la web <http://espana.jobtonic.es>

Coste desarrollo de **jUCAb ox**:

	Cantidad	Precio
<b>Recursos materiales para el desarrollo</b>	3 Lenovo ideapad	1.755 €
<b>Recursos humanos necesarios</b>	1 Analista 474 horas	10.902 €
	1 Desarrollador 544 horas	9.792 €
	1 Encargado de pruebas 120 horas	1.800 €
<b>Total</b>		24.249 €

Tabla 2.2 – Coste de desarrollo del proyecto

Coste anual de mantenimiento de **jUCAb ox**:

	Cantidad	Precio
<b>Recursos materiales</b>	Alojamiento	412,92 €
<b>Total</b>		412,92 €

Tabla 2.3 – Coste de mantenimiento anual del proyecto

## 2.5. Riesgos

La gestión de riesgos es un enfoque estructurado para manejar la incertidumbre de las diferentes amenazas existentes en el desarrollo y puesta en marcha de la aplicación. Para ello es necesario realizar una evaluación de riesgos, estrategias de desarrollo para manejarlos y mitigación del riesgo.

### Principales riesgos en el desarrollo del proyecto:

- **Falta de experiencia en el desarrollo con el framework**
  - Al ser desarrollado con herramientas nuevas en el mercado, el conocimiento de las mismas no es tan elevado, por lo que podría suponer retrasos en el desarrollo. La exposición al riesgo serían de 2 días por cada etapa. La forma de mitigar este riesgo es mediante el estudio y análisis de los frameworks utilizados, buscando suficiente documentación técnica que sirva de apoyo.
- **Cambio en la obtención de Api externa**
  - El recurso de datos principal del que se nutre nuestra aplicación es la API proporcionada por Spotify. Por lo tanto uno de los principales riesgos de la herramienta es el posible cambio en la definición de las funciones proporcionadas por el API. Como medida, Spotify, anuncia los cambios que va a efectuar en su API y cuando serán efectivos. Por lo que se podrían reprogramar esas conexiones en nuestra herramienta, y hacer el cambio cuando sea posible.

### Principales riesgos técnicos:

- **No disponibilidad del sistema por la caída del servidor**
  - Para mitigar este riego y no sufrir dependencia de un único servidor, se utiliza el balanceo de servidores. Su estructura se compone de 3 servidores. El primer servidor es el que se encarga de gestionar la comunicación de la base de datos con el cliente Web. El Segundo hace una réplica en tiempo real de los datos del primero. Y el tercero se mantiene a la espera de posibles daños.
  - Cuando en el primer servidor se produce un fallo, comprometiendo la integridad de la aplicación, el Segundo servidor se pone como maestro y el tercero ocupa el puesto que este deja, siendo el que replica la información del servidor.
  - Cuando el primer servidor vuelve a la normalidad, se convierte en el tercero de ellos, esperando que se produzcan posibles fallos.
- **Pérdida de datos**
  - Para ello, como se ha comentado antes, en el grupo de servidores, el

Segundo servidor no solo estaría esperando a que el primero falle para ocupar su lugar. Serviría también de almacén de copia de seguridad a corto plazo, pudiendo recuperar la información si fuera necesario.

## **2.6. Aseguramiento de calidad**

Para la verificar la calidad de realización de este proyecto, se han comparado los frameworks con más repercusión en la actualidad, llegando a la conclusión que el nuevo framework Angular, teniendo detrás el soporte de Google, es una gran apuesta de futuro, garantizando la continuidad del proyecto, gracias a su gran comunidad de desarrolladores. Además al realizarse una fase de pruebas, se garantiza la calidad funcional de la herramienta.

Para el maquetado y diseño de la web se utiliza HTML5 y CSS3, siguiendo los estándares propuestos por la World Wide Web Consortium. Por lo que se sigue con el estándar marcado para la realización de webs.

Para el Front se utiliza Angular 2 [Ari Lerner et al., 2016], basándose en TypeScript, un framework creado por Microsoft. A su vez es un lenguaje que genera código compilado de JavaScript, siguiendo el estándar ECMA6. Con el cuál se ha desarrollado el proyecto, garantizando así su integridad.

Para el Back-end, pasa exactamente lo mismo, pues se ha desarrollado con Node.js, utilizando JavaScript como lenguaje de programación. Igualmente se ha seguido con el estándar ECMA6 como pilar de desarrollo.

## **Parte II**

## **Desarrollo**





# **Capítulo 3**

## **Requisitos del Sistema**

### **3.1. Situación actual**

Actualmente, cuando se va a celebrar un evento, o acudir a un local, existe la necesidad de introducir un ambiente musical. Cuando un cliente/invitado quiere escuchar alguna canción, tiene que acercarse al encargado de la música, y decirle el nombre de la canción, con lo que conllevaría la interrupción de la labor desempeñada por el encargado de la música, así como la dificultad de entendimiento por el ruido.

Por otro lado, está creciendo la necesidad de solicitar unas canciones antes de realizar un evento social, como una boda, una comunión, o simplemente una fiesta entre amigos. Por lo que el organizador, debería pedir, mediante correos, mensajes y/o personalmente, las canciones que le gustaría que sonaran en el evento. Dado que las fuentes de información son diversas, se corre el riesgo de, que bien, se pierda esa información, o que la información sea la misma y se duplique el trabajo de organización de las canciones.

jUCAbbox nace con la necesidad de crear un sistema autónomo de envío de canciones en lugares con ambiente musical. No se desarrolla para una entidad concreta, sino que tiene como objetivo crear una herramienta, en la que se centralicen todos los locales y/o eventos, con el fin de tener la información unificada.

#### **3.1.1. Procesos de Negocio**

En la actualidad, no existe ningún proceso específico, puesto que con la herramienta se va a dar una utilidad que anteriormente no tenían.

El proceso actual es el intercambio de información de manera verbal. El usuario/cliente pediría una canción a la persona encargada de la música. Esta petición no sería registrada en ningún sistema, por lo que se perdería.

Con la herramienta, se intenta emular esta situación, pero garantizando la recepción de la petición, haciendo un seguimiento, y aportando un valor añadido al proceso.

#### **3.1.2. Entorno Tecnológico**

Puesto que la herramienta no ha sido desarrollada para ninguna empresa en particular, no se han tenido en cuenta el entorno tecnológico de las mismas. Para poder utilizar el software, sería necesario un portátil y/o móvil con conexión a internet. El entorno anterior a la herramienta, será exactamente el mismo que después de la misma. Puesto que para reproducir música en un local sería necesario un ordenador con conexión a internet.

## **3.2. Necesidades de Negocio**

Se busca cumplir con los procesos de negocio descritos anteriormente. Emulando el ciclo de peticiones que tenían, y mejorando el flujo de información, pudiendo explotarla de la manera más sencilla posible.

### **3.2.1. Objetivos de Negocio**

El principal objetivo que se buscan conseguir es el de proporcionar al cliente la posibilidad de solicitar una canción sin tener que moverse por todo el local. Y a su vez, facilitar al responsable de la música los gustos y preferencia de las personas que se encuentren en el local.

Se intenta dar, a su vez, la posibilidad de crear listas de reproducción por un único canal, para que el organizador del evento sepa qué música poner antes de la celebración del mismo.

Un objetivo colateral del resto, es hacer sentir al cliente escuchado, pudiendo tomar parte en la reproducción de canciones en el local.

## **3.3. Objetivos del Sistema**

Los objetivos que se pretenden conseguir con este nuevo sistema es cubrir todos los procesos de negocio descritos anteriormente en los que son necesarios cubrir el registro de lugares y eventos en la herramienta, así como el registro de usuarios para la interacción con la misma.

Teniendo en cuenta otro de los objetivos a cumplir por nuestro sistema, es la conexión en tiempo real con la aplicación Spotify, siendo la comunicación bidireccional.

Se intentará crear un semired social, donde los usuarios podrán interactuar con otros usuarios de la misma, así como los locales podrán dejar mensaje en los tablones de la herramienta, para que los usuarios que los tengan como favorito poder leerlos.

## **3.4. Catálogo de Requisitos**

A continuación se describen los requisitos funcionales y no funcionales del sistema, para conocer las características que debe cumplir la aplicación una vez finalizada.

### **3.4.1. Requisitos funcionales**

En esta sección se van a describir los requisitos funcionales del sistema. Es necesario recalcar, que el perfil de usuario, se comportaría como administrador en el caso de gestionar algún lugar. Se puede acceder a la herramienta sin ser un usuario registrado. Siempre que se hable de usuario, se hablará de usuario registrado. Si el usuario no está registrado se indicará como invitado.

- Se deberá tener un apartado para el registro de usuarios

- El invitado se podrá registrar con Google, Facebook, Twitter o registro interno
  - El usuario podrá buscar a usuarios y agregarlos como amigo
  - El usuario podrá ver los datos de otros usuarios si los agrega como amigos
  - El usuario podrá modificar sus datos
  - El usuario verá un log de actividad
  - El usuario podrá chatear con otros usuarios si son amigos
- Se deberá tener un apartado para la gestión de lugares
  - El usuario podrá crear lugares. Pasará a ser administrador para ese lugar.
  - El usuario podrá ver lugares agregados como favoritos
  - El usuario/invitado podrá buscar lugares
  - El administrador podrá publicar mensajes de sus lugares
  - El administrador podrá validar o rechazar las canciones enviadas
  - El administrador podrá crear distintas playlist asociadas al lugar
  - El administrador podrá reproducir la lista de canciones desde la herramienta
- Se deberá tener un apartado de búsqueda de canciones y artistas
  - El usuario/invitado podrá buscar cualquier canción/artista de Spotify
  - El usuario/invitado podrá enviar una canción a cualquier lugar registrado en la herramienta
  - El usuario podrá agregar como favorito a artistas

### **3.4.2. Requisitos no funcionales**

La aplicación deberá cumplir los requisitos no funcionales listados a continuación:

- **Disponibilidad**
  - La aplicación debe estar disponible las 24 horas del día, los 365 días del año. Para garantizar una mayor experiencia al usuario.
- **Usabilidad**
  - La herramienta debe ser de fácil uso. No restringiendo solo a personas con alto conocimiento tecnológico, sino de sencilla comprensión, y con funcionalidades intuitivas.
- **Portabilidad**
  - La aplicación debe poder usarse en cualquier dispositivo, independientemente del sistema operativo usado y/o el navegador.
- **Seguridad**

- La herramienta ante todo tendrá implementada una capa de seguridad, en la que impediría a cualquier usuario externo acceder a datos internos. Será necesario evitar cualquier ataque a los datos, estableciendo unas medidas extremas en el lado del servidor.
- **Interfaz**
  - La interfaz de usuario debe ser sencilla a la vez que completa. Pudiendo permitir al usuario de forma ágil y veloz acometer su acción de la forma más sencilla posible. Siendo, a su vez, una interfaz atractiva y correctamente consultable desde cualquier dispositivo.
- **Mantenibilidad**
  - La herramienta podrá ser mantenida por cualquier desarrollador previamente formado en los frameworks utilizados. Por lo que es necesario mantener unas buenas prácticas a la hora de desarrollar la aplicación, evitando duplicidad de código o creando funcionalidades demasiado complejas, pudiendo desglosarse en varias.
- **Rendimiento**
  - Al ser una herramienta con conexión de datos con sistemas terceros, los tiempos de respuestas deben de ser cortos, implementando la transmisión asíncrona de datos siempre que sea posible.
- **Fiabilidad**
  - La aplicación debe de ser lo suficientemente robusta para evitar errores en su ejecución.

### **3.4.3. Reglas de negocio**

Se exponen unas reglas de negocio necesarias que debe cumplir la herramienta.

- Solo puede existir una persona que gestione un local.
- Para poder chatear es necesario que los dos usuarios sean amigos.
- Para gestionar un local es necesario estar registrado.

### **3.4.4. Requisitos de información**

En esta sección se describen los requisitos de información que debe cumplir la herramienta.

- Se almacenará la dirección de los lugares a registrar. Para poder realizar búsquedas dependiendo de la ciudad o provincia.
- Se almacenarán los datos referentes a las canciones enviadas al local. Será necesario para la explotación a futuro de los datos, conocer la temática o autores más relevantes.
- Será necesario guardar en un log de actividad toda la actividad de los usuarios. En una siguiente fase se podrá hacer un estudio de mercado con la información

obtenida.

### **3.5. Alternativas de Solución**

En una primera fase se analizó los requisitos que debería cumplir una aplicación para que pudiera enviar y recibir canciones.

En primera instancia se pensó en crear una NAS de alojamiento de ficheros de audio. Un catálogo de canciones para poder relacionar esos ficheros con el artista, nombre de la canción etc...

En segundo lugar, para no tener que litigar con derechos de autor, se pensó en utilizar herramientas de mercado que dieran conexión a terceros. Entre las más utilizadas por la sociedad estaban, Spotify, Napster y SoundCloud.

Las herramientas ponen al servicio de terceros unas API de solicitud de información de datos que manejan.

Para el desarrollo de aplicación se pensó en realizar una herramienta móvil para la utilización del usuario y otra herramienta de escritorio para comunicar las canciones solicitadas.

La otra opción es realizar una Web responsive para el uso tanto de móvil como desde pc. Para el desarrollo de esta herramienta se barajó utilizar frameworks como Symphony2, basados en php, ASP .NET MVC o el conjunto de herramientas propuestos por el Mean Stack.

### **3.6. Solución Propuesta**

La solución propuesta finalmente fue realizar una herramienta con conexión a Spotify, para la obtención de datos a través de su API. Para el desarrollo de la herramienta se realizaría con MEAN stack, compuesto por MongoDB y Express para los datos, Angular 2 para el Front y Node.js para el Backend, puesto que son frameworks que se prevén que van a tener una gran comunidad de desarrolladores detrás, facilitando su mantenimiento futuro.

Además este conjunto de herramientas, ofrece de manera implícita, el desarrollo responsive, por lo que el desarrollo será más sencillo. En un futuro si se quiere implementar una aplicación móvil, existe un componente llamado IONIC 2, que transforma el código de Angular 2 en una aplicación nativa de móvil, tanto IOS como Android.



# **Capítulo 4**

## **Análisis del Sistema**

Esta sección va a contener en análisis del sistema, utilizando modelado UML para ello. El primer modelo mostrará las relaciones entre las entidades del sistema desarrollado.

A continuación se mostraran los casos de uso de cada requisito funcional descrito en el capítulo anterior. En cada uno de ellos, se hará una descripción, conociendo los actores que interactúan en el requisito y las condiciones necesarias para ser cumplido, identificando los escenarios donde se producen.

Más adelante se mostraran los modelos de comportamiento, que describirán el comportamiento del sistema, ofreciendo una visión global de cómo se comportará con las interacciones de los distintos usuarios.

Y al final de esta sección, detallaremos los modelos de interfaz de usuario, pudiendo ver los diseños de las distintas funcionalidades de la herramienta y cómo interactúan entre ellas.

### **4.1. Modelo Conceptual**

A continuación se muestra el diagrama conceptual de clases basado en UML. Cabe recordar, que este sistema está diseñado con bases de datos no relacionales, por lo que la relación entre clases no se comporta como un sistema relacional clásico.

Las entidades externas del diagrama siguiente corresponden a información obtenida a través del api de Spotify.

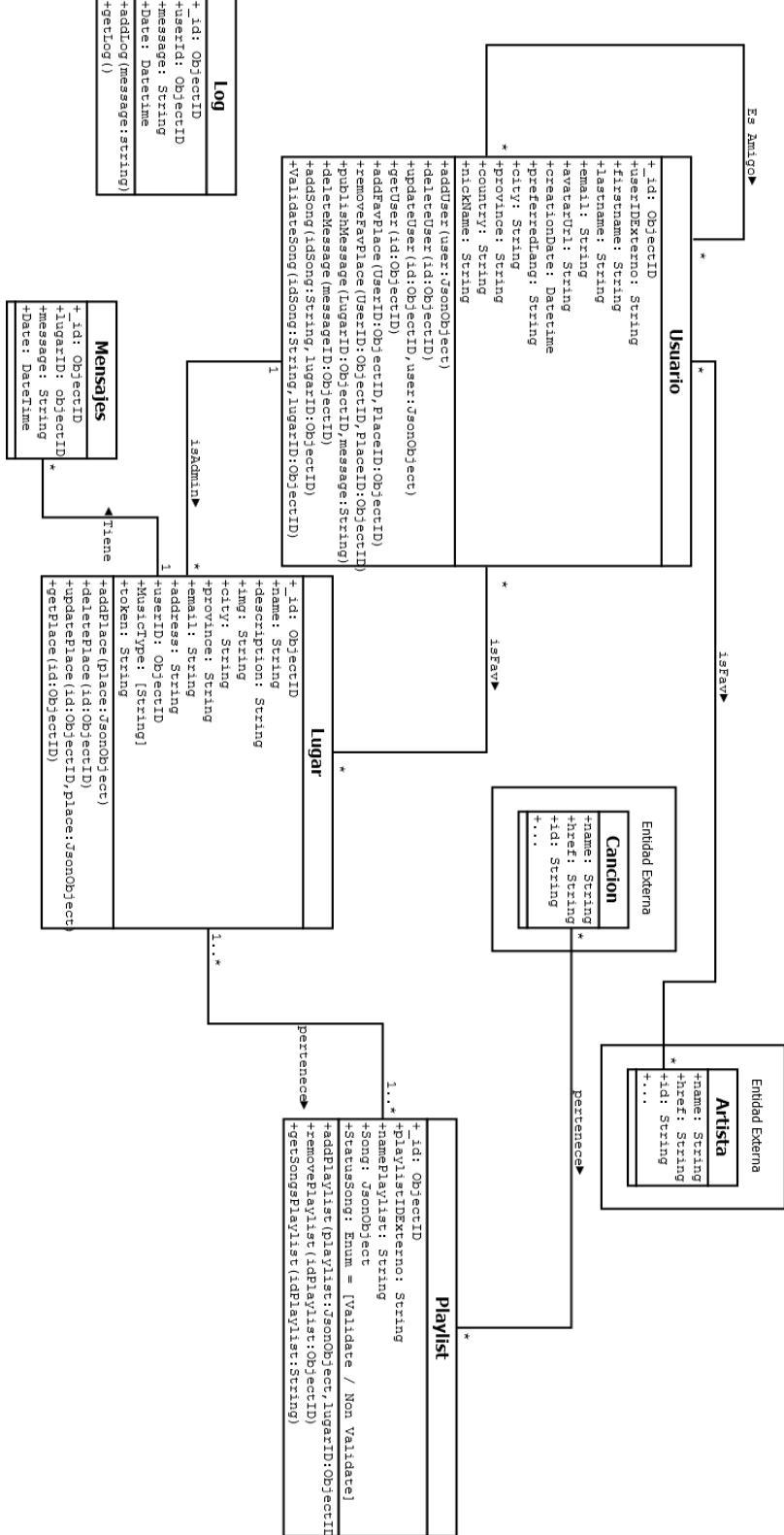


Figura 4.1 – Diagrama relaciones UML

## 4.2. Modelo de Casos de Uso

En el siguiente apartado, se describirán los actores involucrados en los requisitos funcionales, nombrados anteriormente. Posteriormente, se mostrarán cada uno de los casos de uso necesarios para los requisitos funcionales.

### 4.2.1. Actores

En este apartado se describirán los diferentes actores involucrados en el sistema.

- **Invitado:**

- Son las personas que acceden a la herramienta sin registro previo. Podrán utilizar la parte esencial de la aplicación, como es enviar canciones. Pero no podrán añadir como favoritos lugares ni artistas, ni tampoco conocer los mensajes publicados de los lugares. Podrán ver los lugares y canciones enviadas, a su vez podrán filtrar el top de canciones del lugar entre fechas.

- **Usuario:**

- Son las personas que acceden a la herramienta habiéndose registrados. Podrán realizar todas las funciones descritas para el invitado. Además de añadir lugares y artistas como favoritos. Intercambiar mensajes con otros usuarios, ver mensajes publicados por los lugares que tenga añadido como favorito. Así como crear nuevos lugares en la aplicación.

- **Administrador:**

- Son las personas que acceden a la herramienta habiéndose registrado y que gestionan algún lugar. Podrán hacer las mismas gestiones que los usuarios además de validar las canciones enviadas y publicar mensajes en el tablón del lugar.

Los actores Administrador y Usuario no son excluyentes, es decir, para los lugares creados por el Usuario, será considerado como Administrador, para el resto seguirá siendo Usuario. A partir de este momento los perfiles de los usuarios solo se describirán el de menor rango. Siempre que se defina un caso de uso para un invitado, podrá realizarlo Usuarios y Administradores. Si un Caso de uso es realizado por un Usuario, podrá ser realizado por un Administrador. A excepción del registro en la herramienta, solo se podrá registrar un invitado.

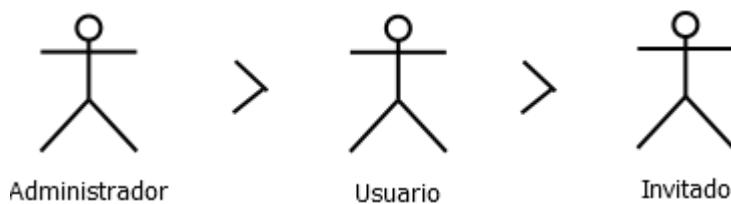


Figura 4.2 – Actores jUCAblox

#### 4.2.2. Casos de Uso

En esta sección se muestra el paquete de casos de uso a alto nivel, en el que se muestran todos los requisitos funcionales descritos anteriormente. Posteriormente se irán desglosando los requisitos en distintos casos de uso.

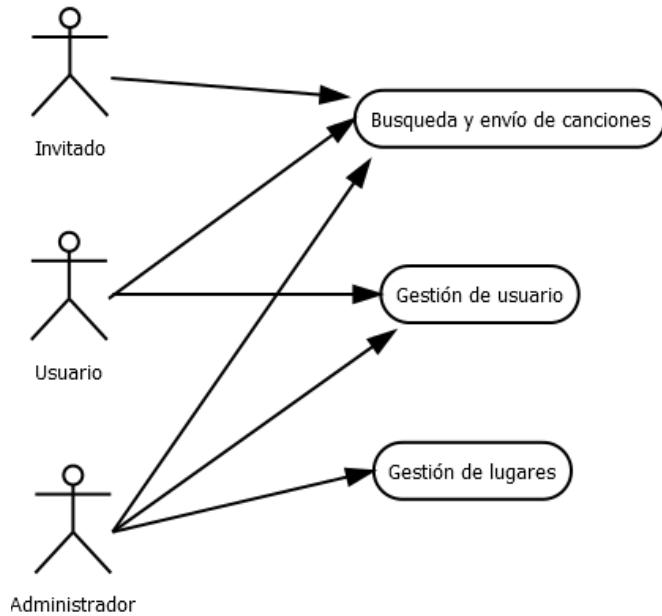


Figura 4.3 – Paquetes de casos de uso

##### 4.2.2.1. Búsqueda y envío de canciones

En el siguiente modelo, se va a mostrar el caso de uso de búsqueda y envío de canciones, de los requisitos funcionales vistos anteriormente.

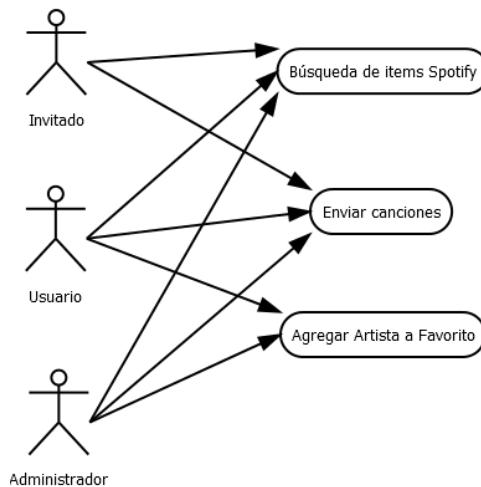


Figura 4.4 – Casos de uso Búsqueda y envío de canciones

- **Caso de uso:** Búsqueda de ítems Spotify
  - **Descripción:** El invitado buscará un ítem en jUCABox perteneciente a Spotify. Este ítem puede ser una canción, un artista, una lista de reproducción y/o un álbum.
  - **Actores:** Invitado
  - **Precondiciones:** Ninguna.
  - **Postcondiciones:** El sistema devolverá el ítem buscado.
- **Identificación de escenarios:**
  - **Escenario principal:**
    1. El caso de uso se inicia cuando el invitado decide buscar un ítem en la herramienta.
    2. El invitado elige el tipo de ítem a buscar.
    3. El invitado pulsa el botón buscar.
    4. El sistema comprueba que los datos introducidos corresponde a algún dato de Spotify.
    5. El sistema devuelve los datos encontrados.
  - **Escenario alternativos:**
    4. a. Los datos no existen en Spotify.
      1. El sistema muestra el error diciendo que no ha encontrado datos.

- 
- **Caso de uso:** Enviar Canciones
    - **Descripción:** El invitado seleccionara la canción que desea enviar, y elegirá a qué lugar quiere hacer la petición.
    - **Actores:** Invitado, Usuario
    - **Precondiciones:** Si se desea elegir de una lista de lugares favoritos, el perfil necesario para él envió debe ser un Usuario registrado.
    - **Postcondiciones:** El sistema enviará la canción solicitada al lugar.
  - **Identificación de escenarios:**
    - **Escenario principal:**
      1. El caso de uso se inicia cuando el invitado decide enviar una canción.
      2. El usuario busca una canción.
      3. El invitado elige la canción deseada.
      4. El invitado pulsa el botón enviar.
      5. El invitado elige del listado de lugares a cual quiere enviar la canción
      6. El invitado introduce el token para comprobar la validez del lugar
      7. El invitado pulsa enviar
      8. El sistema comprueba que los datos introducidos son correcto.
      9. El sistema envía la solicitud de la canción al lugar.
    - **Escenario alternativos:**
      6. a. El token introducido es incorrecto.
        1. El sistema muestra el error diciendo que el token es incorrecto.

- 
- **Caso de uso:** Añadir Artista Favorito
    - **Descripción:** El usuario buscará un artista, y lo seleccionará como favorito.
    - **Actores:** Usuario
    - **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
    - **Postcondiciones:** El sistema almacenará el artista como favorito.
  - **Identificación de escenarios:**
    - **Escenario principal:**
      1. El caso de uso se inicia cuando el usuario decide agregar un artista como favorito.
      2. El usuario busca el artista.
      3. El usuario accede al artista.
      4. El usuario selecciona añadir a favoritos
      5. El sistema añade el artista a favoritos del usuario.
  - **Caso de uso:** Eliminar Artista Favorito
    - **Descripción:** El usuario buscará un artista, y lo eliminará como favorito.
    - **Actores:** Usuario
    - **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.  
El artista debe ser un favorito del usuario.
    - **Postcondiciones:** El sistema eliminará el artista como favorito.
  - **Identificación de escenarios:**
    - **Escenario principal:**
      1. El caso de uso se inicia cuando el usuario decide eliminar un artista como favorito.
      2. El usuario busca el artista desde la búsqueda de ítems.
      3. El usuario accede al artista.
      4. El usuario selecciona eliminar de favoritos
      5. El sistema elimina el artista a favoritos del usuario.
    - **Escenario alternativos:**
      2. a. El usuario busca el artista desde el menú de usuario.
      3. a. El usuario selecciona eliminar de favoritos
      4. a. El sistema elimina el artista a favoritos del usuario.

#### 4.2.2.2. Gestión de Usuario

En el siguiente modelo, se va a mostrar el caso de uso de gestión de usuarios, de los requisitos funcionales vistos anteriormente.

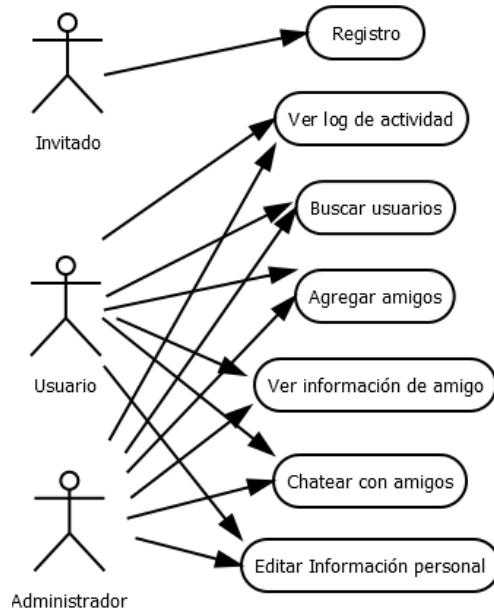


Figura 4.5 – Casos de uso Gestión de usuarios

- **Caso de uso: Registro en el sistema**

- **Descripción:** El invitado se registrará en la herramienta.
- **Actores:** Invitado
- **Precondiciones:** Ninguna.
- **Postcondiciones:** El sistema creará una cuenta de usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el invitado decide registrarse en la herramienta.
  2. El invitado elige el método de registro Google / Facebook / Twitter / Interno.
  3. El invitado introduce su correo y contraseña.
  4. El sistema crea un nuevo usuario.

- **Caso de uso: Ver log de actividad**

- **Descripción:** El usuario verá el log de actividad y peticiones de la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario deberá estar registrado y logeado en la herramienta.
- **Postcondiciones:** El sistema devolverá el log de usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el usuario desea ver su log de actividad.

- 
2. El usuario accede a su perfil.
  3. El usuario solicita su log de actividad.
  4. El sistema devuelve el log de actividad.
- 

- **Caso de uso: Buscar usuarios**

- **Descripción:** El usuario buscará a otros usuarios en la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema devolverá los usuarios buscados.

- **Identificación de escenarios:**

- **Escenario principal:**
    1. El caso de uso se inicia cuando el usuario decide buscar a otro usuario.
    2. El usuario busca el nombre o apellidos del usuario desde su perfil.
    3. El sistema devuelve los usuarios que coincidan con la búsqueda.
  - **Escenario alternativos:**
    3. a. El sistema no encuentra a los usuarios solicitados.
- 

- **Caso de uso: Agregar nuevo amigo**

- **Descripción:** El usuario agregará a otros usuarios como amigos en la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema añadirá un nuevo amigo al usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
    1. El caso de uso se inicia cuando el usuario decide buscar a otro usuario y agregarlos.
    2. El usuario busca el nombre o apellidos del usuario desde su perfil.
    3. El sistema devuelve los usuarios que coincidan con la búsqueda.
    4. El usuario agrega al usuario que desea seguir.
    5. El sistema añade al usuario como amigo
- 

- **Caso de uso: Eliminar amigo**

- **Descripción:** El usuario desea eliminar a otros usuarios como amigos en la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema eliminará el amigo del usuario.

- **Identificación de escenarios:**

- **Escenario principal:**

1. El caso de uso se inicia cuando el usuario decide buscar a otro usuario y eliminarlo como amigo.
  2. El usuario busca el nombre o apellidos del usuario desde su perfil.
  3. El sistema devuelve los usuarios que coincidan con la búsqueda.
  4. El usuario eliminará el usuario al que ya no desea seguir.
  5. El sistema eliminará el amigo del usuario.
- 

- **Caso de uso:** Ver información de amigo

- **Descripción:** El usuario desea ver la información de otros usuarios como amigos en la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta. El usuario debe ser amigo del usuario que desea consultar.
- **Postcondiciones:** El sistema devolverá los datos del usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
    1. El caso de uso se inicia cuando el usuario decide buscar a otro usuario y ver su información.
    2. El usuario busca el nombre o apellidos del usuario desde su perfil.
    3. El sistema devuelve los usuarios que coincidan con la búsqueda.
    4. El usuario seleccionará el usuario que desea ver.
    5. El sistema devolverá los datos del amigo del usuario.
- 

- **Caso de uso:** Chatear con amigo

- **Descripción:** El usuario desea chatear con otros usuarios en la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta. El usuario debe ser amigo del usuario que desea consultar.
- **Postcondiciones:** El sistema devolverá y enviará los mensajes intercambiados con el usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el usuario decide buscar a otro usuario y empezar a chatear.
  2. El usuario busca el nombre o apellidos del usuario desde su perfil.
  3. El sistema devuelve los usuarios que coincidan con la búsqueda.
  4. El usuario seleccionará el usuario con el que desea chatear.
  5. El sistema devolverá y enviará los mensajes intercambiados con el usuario seleccionado.
- **Escenario alternativos:**
  4. a. El usuario no está conectado para poder chatear.

1. El usuario dejará escrito los mensajes y el sistema se lo mostrará al usuario cuando se conecte
- 

- **Caso de uso:** Editar Información personal

- **Descripción:** El usuario desea modificar sus datos en la herramienta.
- **Actores:** Usuario
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema modificará los datos solicitados por el usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el usuario modificar sus datos registrados en la herramienta.
  2. El usuario accede a su perfil.
  3. El usuario modifica los datos en su perfil.
  4. El sistema guarda los datos modificados por el usuario.
- **Escenario alternativos:**
  4. a. El usuario ha introducido incorrectamente los datos.
    1. El sistema devuelve que campos son incorrectos sin actualizar los datos del usuario

#### 4.2.2.3. Gestionar Lugares

En el siguiente modelo, se va a mostrar el caso de uso de gestión de lugares, de los requisitos funcionales vistos anteriormente.

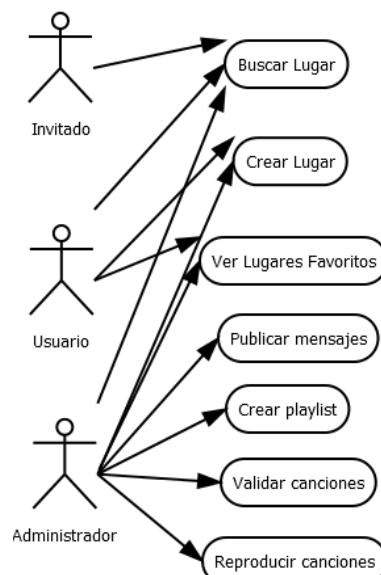


Figura 4.6 – Casos de uso Gestión de Lugares

- **Caso de uso: Buscar lugares**

- **Descripción:** El invitado buscará lugares en la herramienta.
- **Actores:** Invitado
- **Precondiciones:** Ninguno.
- **Postcondiciones:** El sistema devolverá los lugares buscados.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el invitado decide buscar un lugar.
  2. El invitado busca el nombre, ciudad o provincia desde el menú de búsqueda de lugares.
  3. El sistema devuelve los lugares que coincidan con la búsqueda.
- **Escenario alternativos:**
  3. a. El sistema no encuentra los lugares solicitados.

---

- **Caso de uso: Agregar lugar a favoritos**

- **Descripción:** El usuario buscará lugares en la herramienta y los agregará como favoritos.
- **Actores:** Usuario.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema añadirá el lugar a los lugares favoritos del usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el usuario decide agregar un lugar a favoritos.
  2. El usuario busca el nombre, ciudad o provincia desde el menú de búsqueda de lugares.
  3. El usuario selecciona el lugar que quiere añadir a favoritos.
  4. El sistema añade el lugar a los lugares favoritos del usuario.

---

- **Caso de uso: Ver lugares favoritos**

- **Descripción:** El usuario accederá a sus lugares favoritos.
- **Actores:** Usuario.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.  
El usuario deberá tener agregado el lugar como favorito
- **Postcondiciones:** El sistema mostrará los lugares favoritos del usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el usuario decide agregar un lugar a favoritos.

- 
2. El usuario buscará el lugar favorito en su perfil.
  3. El sistema mostrará los lugares favoritos del usuario.
- 

- **Caso de uso:** Eliminar lugar de favoritos

- **Descripción:** El usuario eliminará un lugar de sus favoritos.
- **Actores:** Usuario.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema eliminará el lugar de los lugares favoritos del usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
    1. El caso de uso se inicia cuando el usuario decide agregar un lugar a favoritos.
    2. El usuario buscará el lugar favorito en su perfil.
    3. El usuario selecciona el lugar que quiere eliminar de favoritos.
    4. El sistema eliminará el lugar de los lugares favoritos del usuario.
- 

- **Caso de uso:** Crear Lugar

- **Descripción:** El usuario creará un nuevo lugar en la herramienta.
- **Actores:** Usuario.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta.
- **Postcondiciones:** El sistema creará un lugar administrado por el usuario.

- **Identificación de escenarios:**

- **Escenario principal:**
    1. El caso de uso se inicia cuando el usuario decide crear un nuevo lugar.
    2. El usuario accederá al menú de lugares.
    3. El usuario seleccionará crear un nuevo lugar.
    4. El usuario añadirá todos los campos necesarios para la creación del lugar.
    5. El usuario seleccionará crear nuevo lugar.
    6. El sistema añadirá un nuevo lugar.
- 

- **Caso de uso:** Publicar mensaje

- **Descripción:** El administrador publicará un nuevo mensaje en el tablón del lugar.
- **Actores:** Administrador.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta. El usuario debe ser administrador del lugar.

- **Postcondiciones:** El sistema publicará un nuevo mensaje en el tablón del lugar.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el administrador decide añadir un nuevo mensaje al tablón del lugar.
  2. El administrador accederá al menú de lugares.
  3. El administrador seleccionará el lugar que desea administrar.
  4. El administrador añadirá todos los campos necesarios para la publicación del mensaje.
  5. El sistema añadirá un mensaje al tablón del lugar.

---

- **Caso de uso: Crear Playlist**

- **Descripción:** El administrador creará una nueva playlist para un lugar en la herramienta.
- **Actores:** Administrador.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta. El usuario debe ser administrador del lugar. El administrador debe tener una cuenta Spotify y estar logeado como usuario Spotify.
- **Postcondiciones:** El sistema creará una nueva playlist asociada al lugar.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el administrador decide crear una nueva playlist.
  2. El administrador accederá al menú de lugares.
  3. El administrador buscará el lugar que desea administrar.
  4. El administrador seleccionará administrar lugar.
  5. El administrador accederá al apartado de playlist.
  6. El administrador creará una nueva playlist.
  7. El sistema añadirá una nueva playlist al lugar.

---

- **Caso de uso: Validar/Rechazar canciones**

- **Descripción:** El administrador validará o rechazará las canciones enviadas al lugar.
- **Actores:** Administrador.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta. El usuario debe ser administrador del lugar. El administrador debe tener una cuenta Spotify y estar logeado como usuario Spotify.
- **Postcondiciones:** El sistema validará o rechazará la canción seleccionada.

- **Identificación de escenarios:**

- **Escenario principal:**
    1. El caso de uso se inicia cuando el administrador validar una canción.
    2. El administrador accederá al menú de lugares.
    3. El administrador buscará el lugar que desea administrar.
    4. El administrador seleccionará administrar lugar.
    5. El administrador accederá al apartado de playlist.
    6. El administrador accede a la playlist que quiere utilizar.
    7. El administrador validará la canción seleccionada.
    8. El sistema envía a Spotify la canción validada.
    9. El sistema añadirá una nueva playlist al lugar.
  - **Escenario alternativos:**
    7. a. El administrador rechaza la canción seleccionada.
    8. a. El sistema elimina la canción rechazada.
- 

- **Caso de uso: Reproducir canciones**

- **Descripción:** El administrador reproducirá las canciones que tiene validada.
- **Actores:** Administrador.
- **Precondiciones:** El usuario debe estar registrado y logado en la herramienta. El usuario debe ser administrador del lugar. El administrador debe tener una cuenta Spotify y estar logeado como usuario Spotify.
- **Postcondiciones:** El sistema reproducirá, a través de Spotify, la canción seleccionada.

- **Identificación de escenarios:**

- **Escenario principal:**
  1. El caso de uso se inicia cuando el administrador decide reproducir una canción que tiene validada.
  2. El administrador accederá al menú de lugares.
  3. El administrador buscará el lugar que desea administrar.
  4. El administrador seleccionará administrar lugar.
  5. El administrador accederá al apartado de playlist.
  6. El administrador accede a la playlist que quiere utilizar.
  7. El administrador elige en que dispositivo quiere reproducir la canción.
  8. El administrador elegirá que canción quiere reproducir.
  9. El sistema reproducirá la canción en el dispositivo a través de Spotify.

### **4.3. Modelo de Comportamiento**

A partir de los casos de uso anteriores, se crea el modelo de comportamiento. Para ello, se realizarán los diagramas de secuencia del sistema, donde se identificarán las operaciones o servicios del sistema. Luego, se detallará el contrato de las operaciones identificadas. Se realizarán los modelos de comportamientos más importantes de la herramienta, puesto que muchos son similares.

#### 4.3.1. Acceso y salida del sistema

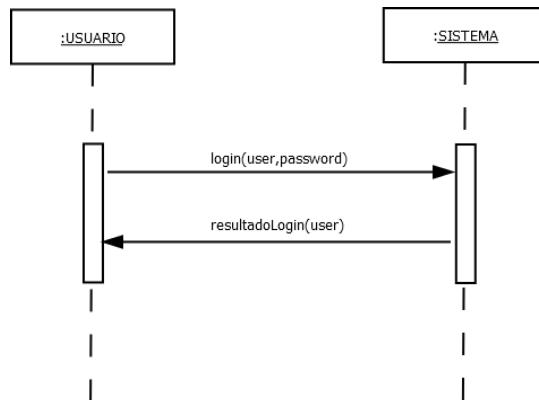


Figura 4.7 – Modelo comportamiento Login interno

A continuación se expone el modelo de comportamiento, cuando el sistema utiliza un proveedor como google para iniciar sesión. El sistema externo devuelve Ok cuando el usuario inicia sesión, desconociendo el sistema la contraseña del usuario para el sistema externo. El sistema externo únicamente devuelve el usuario con el que se ha logado.

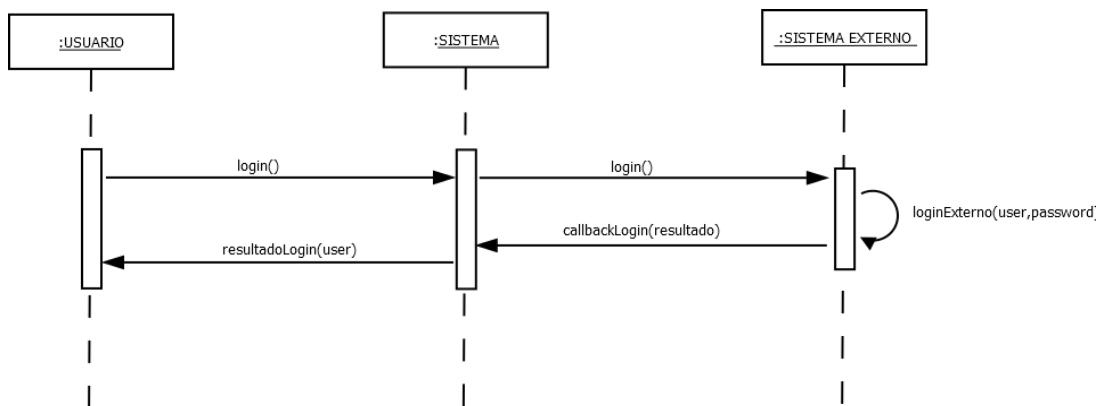


Figura 4.8 – Modelo comportamiento Login externo

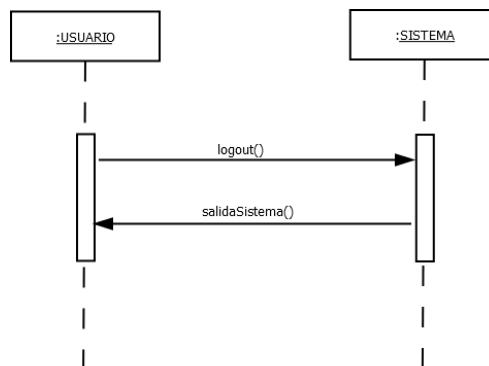


Figura 4.9 – Modelo comportamiento Logout

#### 4.3.2. Búsqueda y envío de canciones

Para la realización de búsquedas es necesario que el sistema le mande a Spotify un token, autenticando el sistema origen que hace la petición.

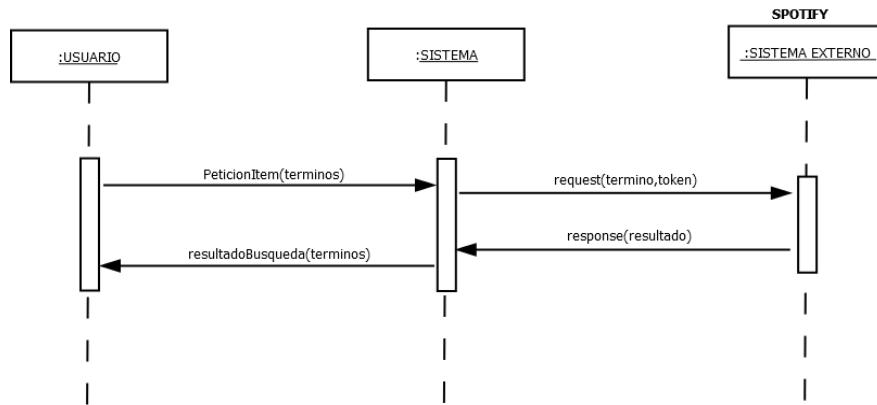


Figura 4.10 – Modelo comportamiento Búsqueda Ítem

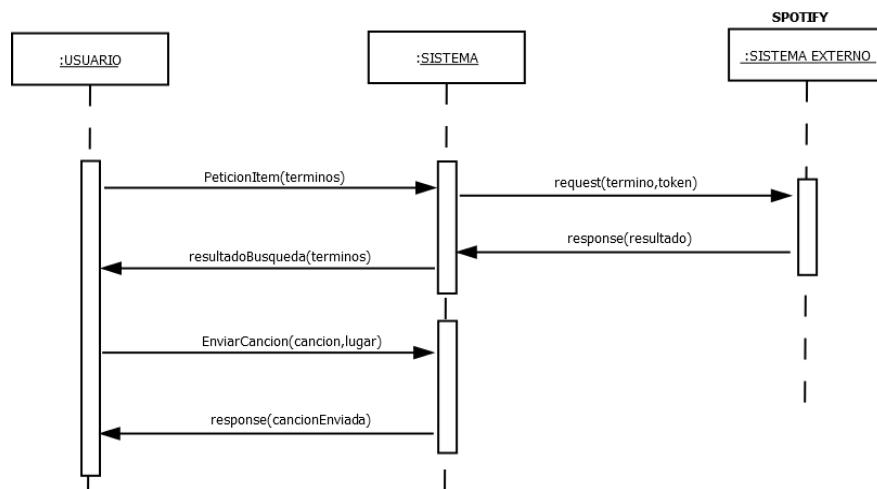


Figura 4.11 – Modelo comportamiento Enviar Canción

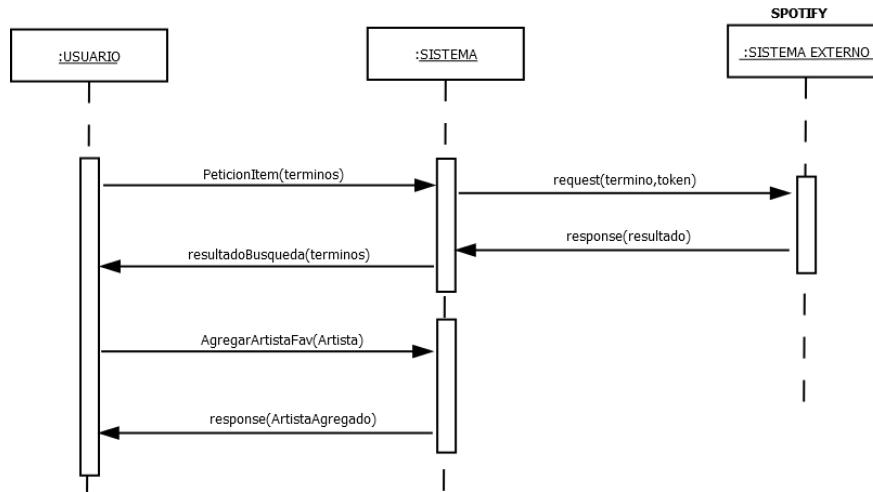


Figura 4.11 – Modelo comportamiento Agregar Artista Favorito

#### 4.3.3. Gestión de usuarios

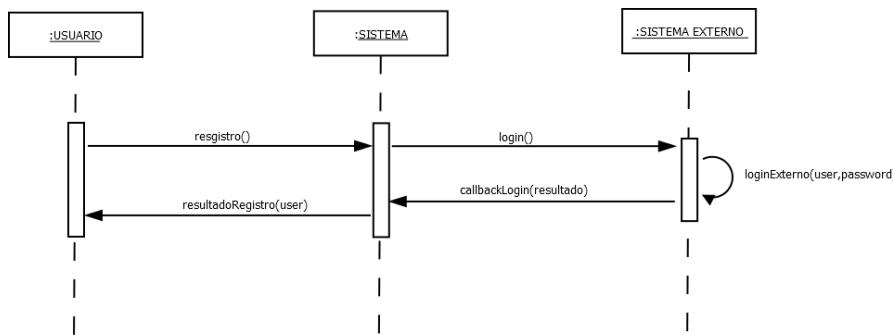


Figura 4.12 – Modelo comportamiento Registro usuario

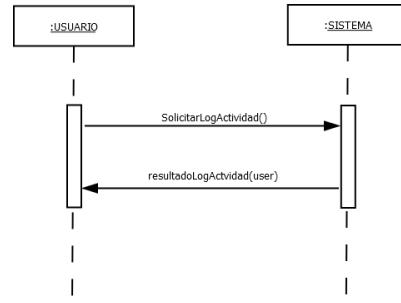


Figura 4.13 – Modelo comportamiento Log de Actividad

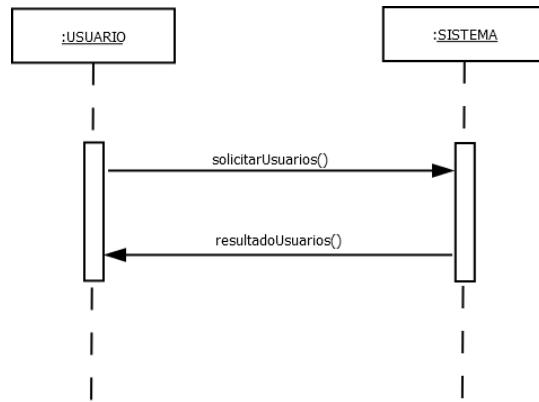


Figura 4.14 – Modelo comportamiento Búsqueda usuarios

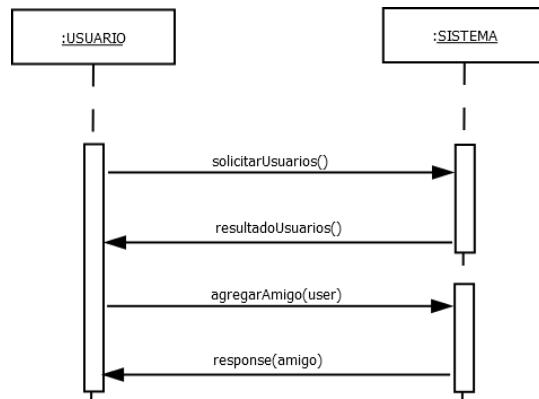


Figura 4.15 – Modelo comportamiento Agregar Amigo

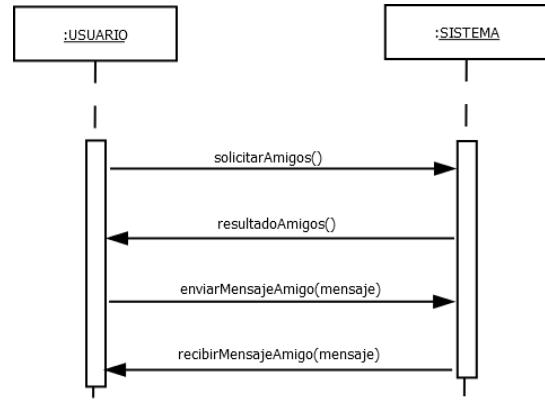


Figura 4.16 – Modelo comportamiento Chatear

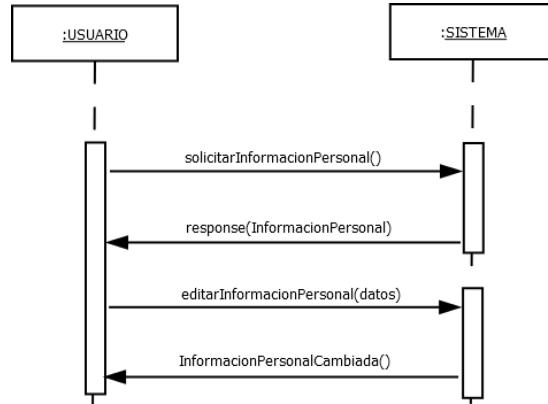


Figura 4.17 – Modelo comportamiento Editar Información Personal

#### 4.3.4. Gestión de lugares

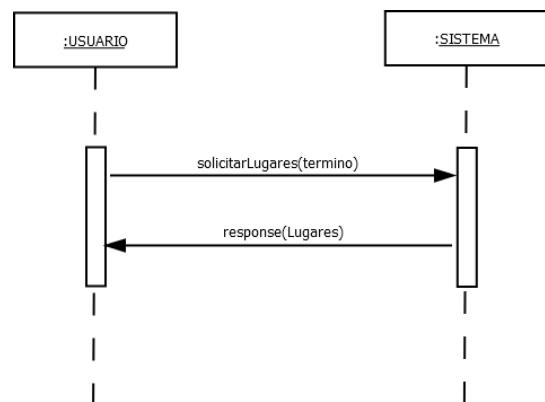


Figura 4.18 – Modelo comportamiento Búsqueda de Lugares

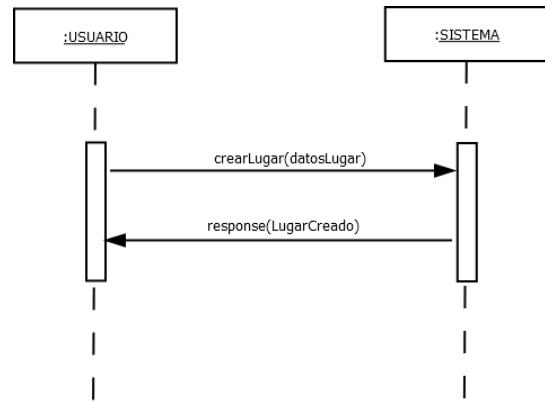


Figura 4.19 – Modelo comportamiento Crear Lugar

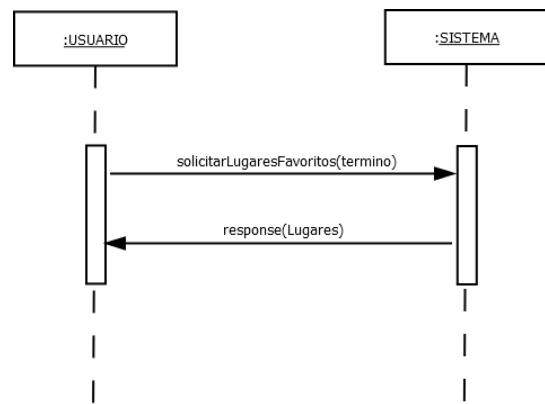


Figura 4.20 – Modelo comportamiento Ver Lugares Favoritos

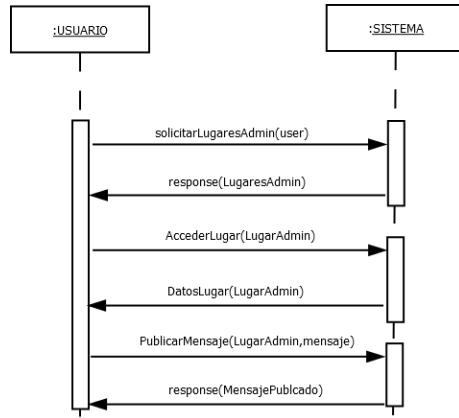


Figura 4.21 – Modelo comportamiento Publicar Mensaje

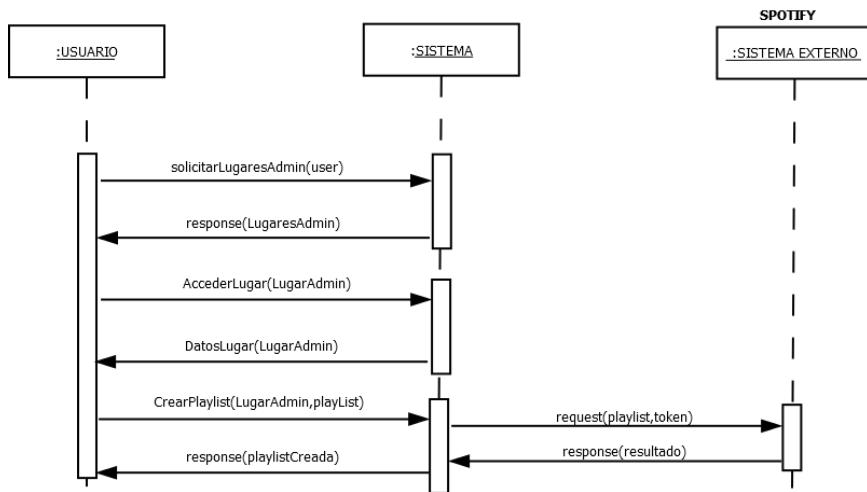


Figura 4.22 – Modelo comportamiento Crear Playlist

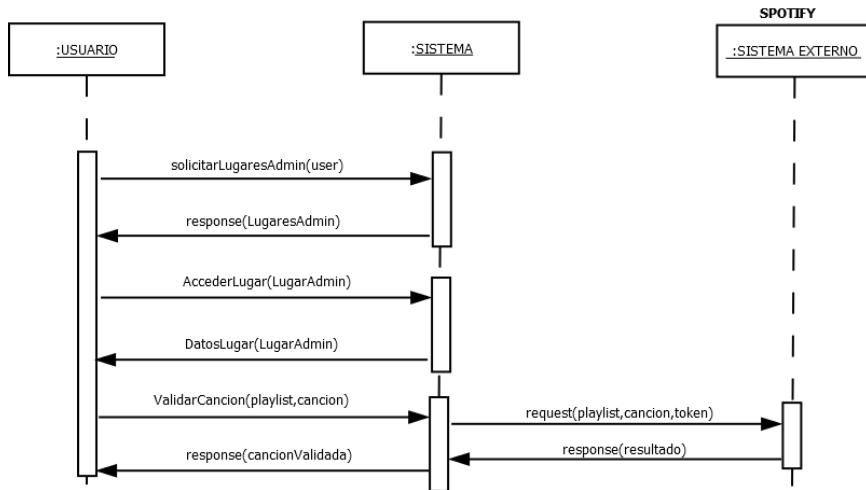


Figura 4.23 – Modelo comportamiento Validar Canción

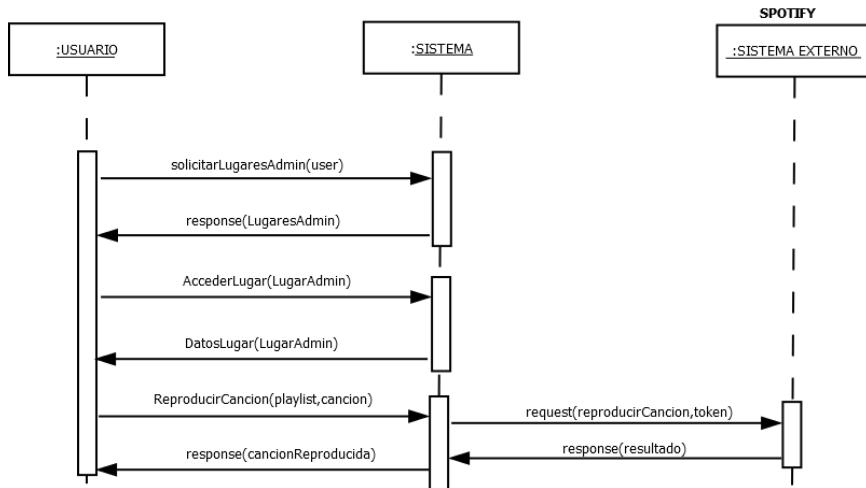


Figura 4.24 – Modelo comportamiento Reproducir Canción

#### 4.3.5. Contrato de operaciones

En esta sección describiremos las operaciones de los modelos de comportamiento. Solo describiremos las más relevantes. Se indicará que hacen, no como lo hacen. Para diferenciar las variables del sistema con las de la función, las connotaremos con una S al final de la variable.

<b>Contrato:</b> Login Interno	
<b>Nombre</b>	Login(user,password)
<b>Responsabilidad</b>	Inicia sesión en la aplicación
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Datos usuario : user
<b>Precondiciones</b>	Debe existir:  userS = user passwordS = password
<b>Poscondiciones</b>	Se actualiza que el usuario está conectado, y el sistema muestra la ventana de inicio de la herramienta

Tabla 4.1 – Contrato: Login(user,password)

<b>Contrato:</b> Logout	
<b>Nombre</b>	Logout()
<b>Responsabilidad</b>	Finaliza sesión en la aplicación
<b>Excepciones</b>	Si el usuario no está logado
<b>Salidas</b>	Ninguna
<b>Precondiciones</b>	El usuario debe estar logado
<b>Poscondiciones</b>	Se actualiza que el usuario está desconectado, y el sistema muestra la ventana de inicio de la herramienta sin usuario

Tabla 4.2 – Contrato: Logout()

<b>Contrato:</b> Búsqueda Ítem	
<b>Nombre</b>	PeticiónItem(termino)
<b>Responsabilidad</b>	El sistema devuelve una petición de un elemento buscado
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con el resultado de la petición
<b>Precondiciones</b>	Ninguna
<b>Poscondiciones</b>	Se muestra por pantalla los datos solicitados. Si no existe ningún elemento, se muestra un mensaje.

Tabla 4.3 – Contrato: PeticiónItem(termino)

<b>Contrato:</b> Enviar Canción	
<b>Nombre</b>	EnviarCanción(canción,lugar)
<b>Responsabilidad</b>	El sistema añade una nueva petición de una canción a un lugar.
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con la canción enviada
<b>Precondiciones</b>	Ninguna
<b>Poscondiciones</b>	Se añade a la lista de canciones pendientes de validar por parte del administrador del lugar.

Tabla 4.4 – Contrato: EnviarCanción(canción,lugar)

<b>Contrato:</b> Agregar artista a favorito	
<b>Nombre</b>	AgregarArtistaFav(artista)
<b>Responsabilidad</b>	El sistema añade un artista favorito al usuario que está logado
<b>Excepciones</b>	Si el artista ya está en la lista de favoritos
<b>Salidas</b>	Objeto Json con la artista añadido
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta
<b>Poscondiciones</b>	Se añade a la lista de favoritos el usuario seleccionado.

Tabla 4.5 – Contrato: AgregarArtistaFav(artista)

<b>Contrato:</b> Solicitar log de actividad	
<b>Nombre</b>	SolicitarLogActividad()
<b>Responsabilidad</b>	El sistema devuelve el log de actividad del usuario conectado
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con el log de actividad
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta
<b>Poscondiciones</b>	El sistema muestra la pantalla del log de actividad con los datos del mismo.

Tabla 4.6 – Contrato: SolicitarLogActividad()

<b>Contrato:</b> Solicitar usuarios	
<b>Nombre</b>	SolicitarUsuarios()
<b>Responsabilidad</b>	El sistema devuelve los usuarios registrados en la aplicación
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con los usuarios de la aplicación
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta
<b>Poscondiciones</b>	El sistema muestra la pantalla de usuarios con los usuarios del sistema

Tabla 4.7 – Contrato: SolicitarUsuarios()

<b>Contrato:</b> Agregar Amigo	
<b>Nombre</b>	AgregarAmigo(user)
<b>Responsabilidad</b>	El sistema inserta un nuevo amigo al conjunto de amigos del usuario
<b>Excepciones</b>	El amigo no debe estar previamente en la lista de amigos
<b>Salidas</b>	Objeto Json con el amigo añadido
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta
<b>Poscondiciones</b>	El sistema añade a user a la colección de amigos del usuario

Tabla 4.8 – Contrato: AgregarAmigo(user)

<b>Contrato:</b> Solicitar lugares	
<b>Nombre</b>	solicitarLugares(termino)
<b>Responsabilidad</b>	El sistema devuelve los lugares que coincidan con el término
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con la lista de lugares
<b>Precondiciones</b>	Ninguna
<b>Poscondiciones</b>	El sistema muestra la pantalla de lugares, filtrada por el término. Si no existe ninguno el sistema mostrará un mensaje.

Tabla 4.9 – Contrato: solicitarLugares(termino)

<b>Contrato:</b> Crear Lugar	
<b>Nombre</b>	crearLugar(datosLugar)
<b>Responsabilidad</b>	El sistema añade un nuevo lugar
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con el lugar creado
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta
<b>Poscondiciones</b>	El sistema crea el lugar con los datos proporcionados por el usuario.

	<p>El sistema muestra la pantalla de detalle del lugar.</p> <p>El sistema asigna el rol de Administrador del lugar al usuario que lo ha creado.</p>
--	---

Tabla 4.10 – Contrato: crearLugar(datosLugar)

<b>Contrato:</b> Publicar mensaje	
<b>Nombre</b>	publicarMensaje(lugarAdmin,mensaje)
<b>Responsabilidad</b>	El sistema inserta un nuevo mensaje en el lugarS = lugarAdmin
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con el mensaje insertado
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta y ser el administrador del lugar
<b>Poscondiciones</b>	El sistema inserta el mensaje, en la lista de mensajes del lugar

Tabla 4.11 – Contrato: publicarMensaje(lugarAdmin,mensaje)

<b>Contrato:</b> Crear Playlist	
<b>Nombre</b>	crearPlaylist(lugarAdmin,playlist)
<b>Responsabilidad</b>	El sistema inserta una nueva playlist en el lugarS = lugarAdmin
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con la playlist creada
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta, ser el administrador del lugar y estar logado con el usuario de Spotify.
<b>Poscondiciones</b>	El sistema inserta una nueva playlist, y la crea en Spotify. El sistema muestra las canciones a validar con la playlist creada.

Tabla 4.12 – Contrato: crearPlaylist(lugarAdmin,playlist)

<b>Contrato:</b> Validar Canción	
<b>Nombre</b>	validarCancion(canción,playlist)
<b>Responsabilidad</b>	El sistema valida una canción previamente solicitada por usuarios.
<b>Excepciones</b>	Ninguna
<b>Salidas</b>	Objeto Json con la canción validada
<b>Precondiciones</b>	El usuario debe estar logado en la herramienta, ser el administrador del lugar y estar logado con el usuario de Spotify.
<b>Poscondiciones</b>	El sistema añade la canción a la playlist seleccionada. El sistema añade al log que ha sido validada la canción.

Tabla 4.13 – Contrato: validarCancion(canción,playlist)

#### 4.4. Modelo de Interfaz de Usuario

Estos prototipos serán utilizado para el desarrollo de la aplicación más adelante, siendo lo más fieles posibles.

A continuación se muestran los prototipos de la aplicación web, mostrando las pantallas más relevantes y la estructura general de la misma.

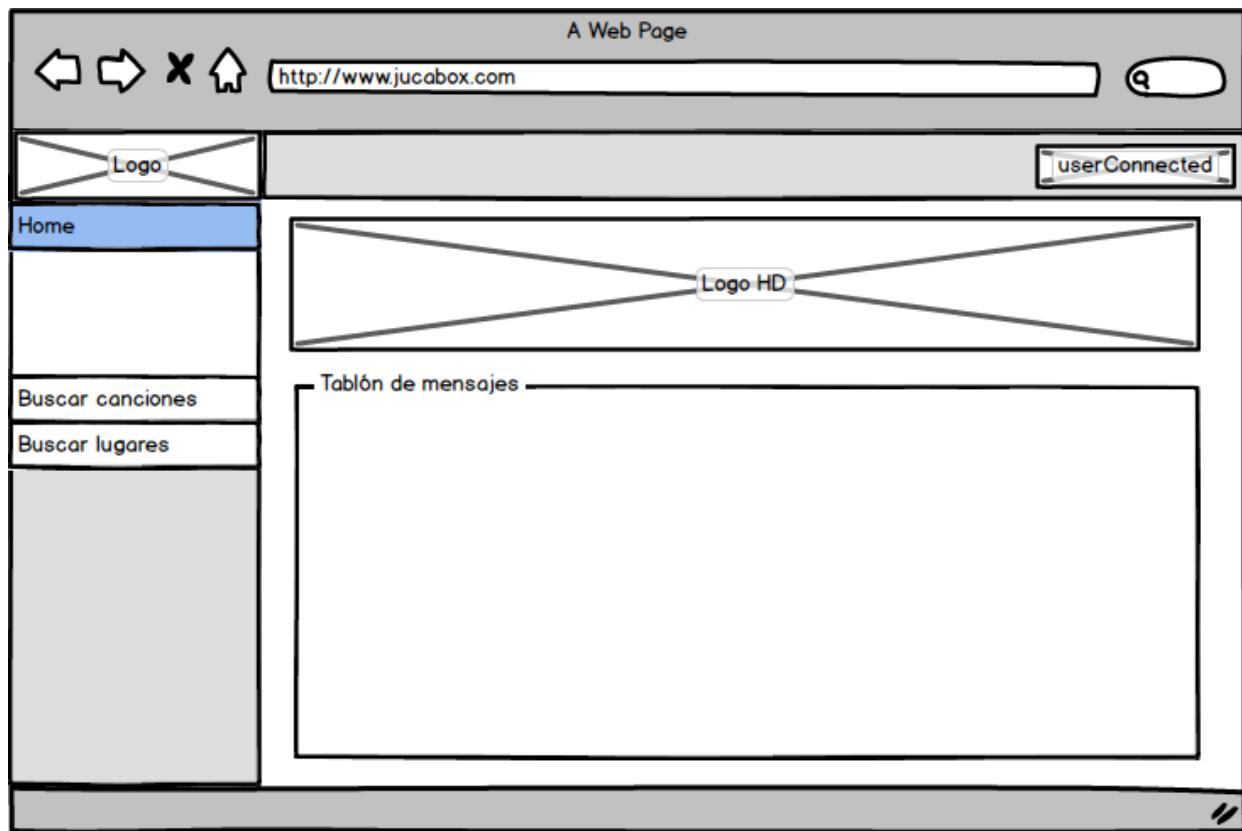


Figura 4.25 –Prototipado – Home

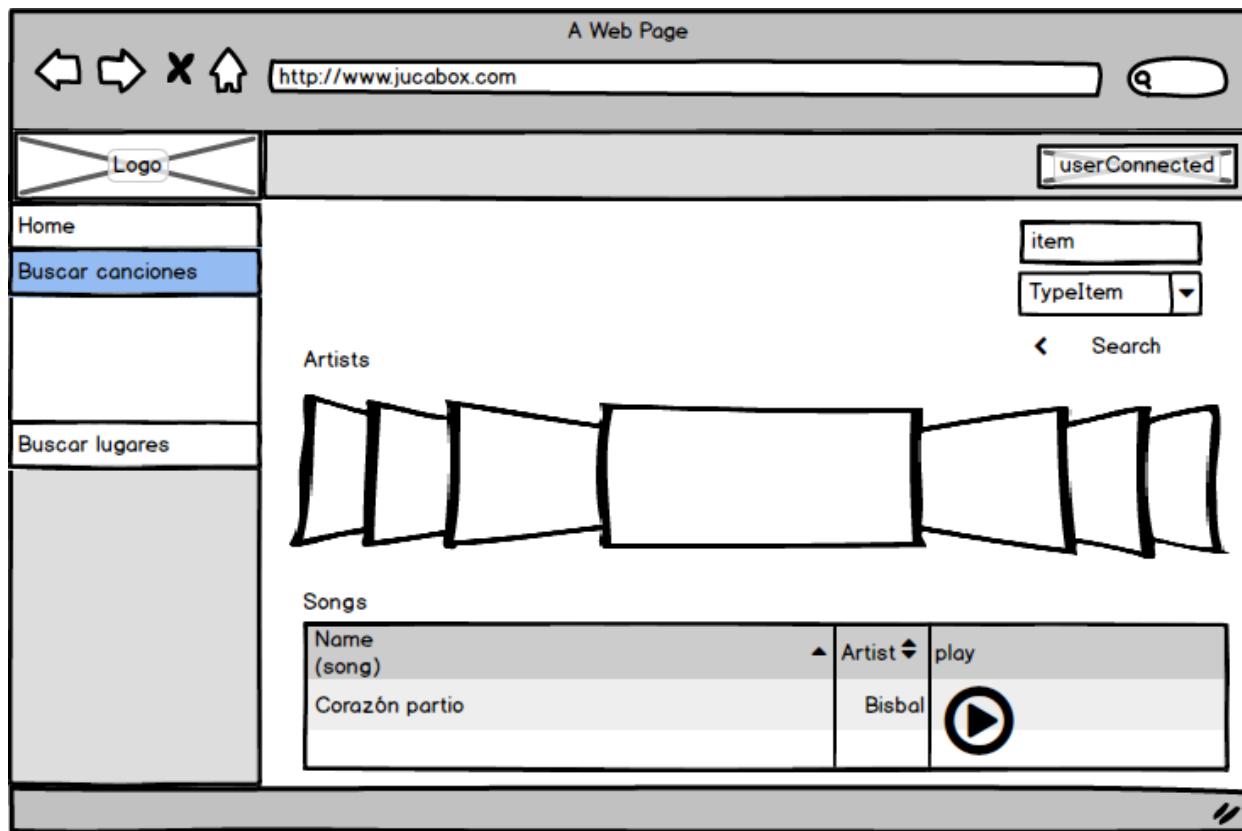


Figura 4.26 – Prototipado – Buscar Canción

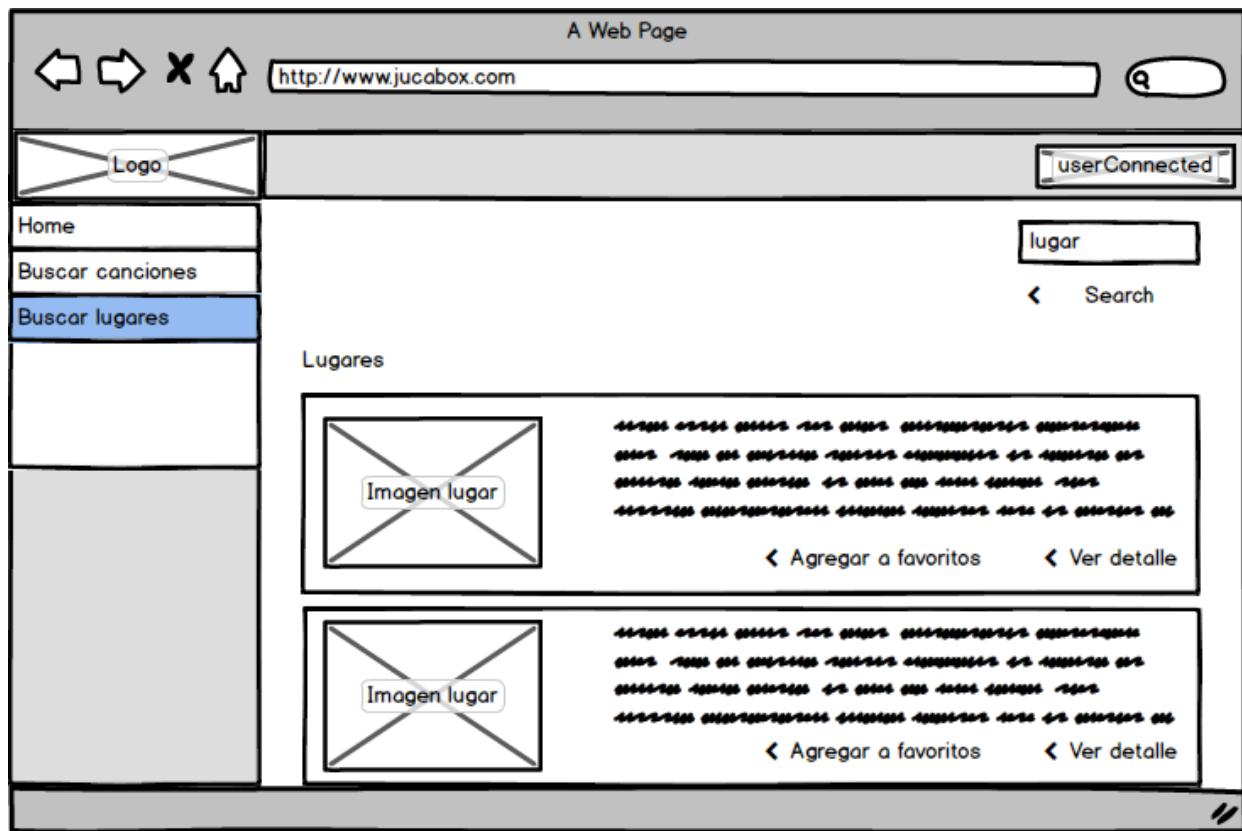
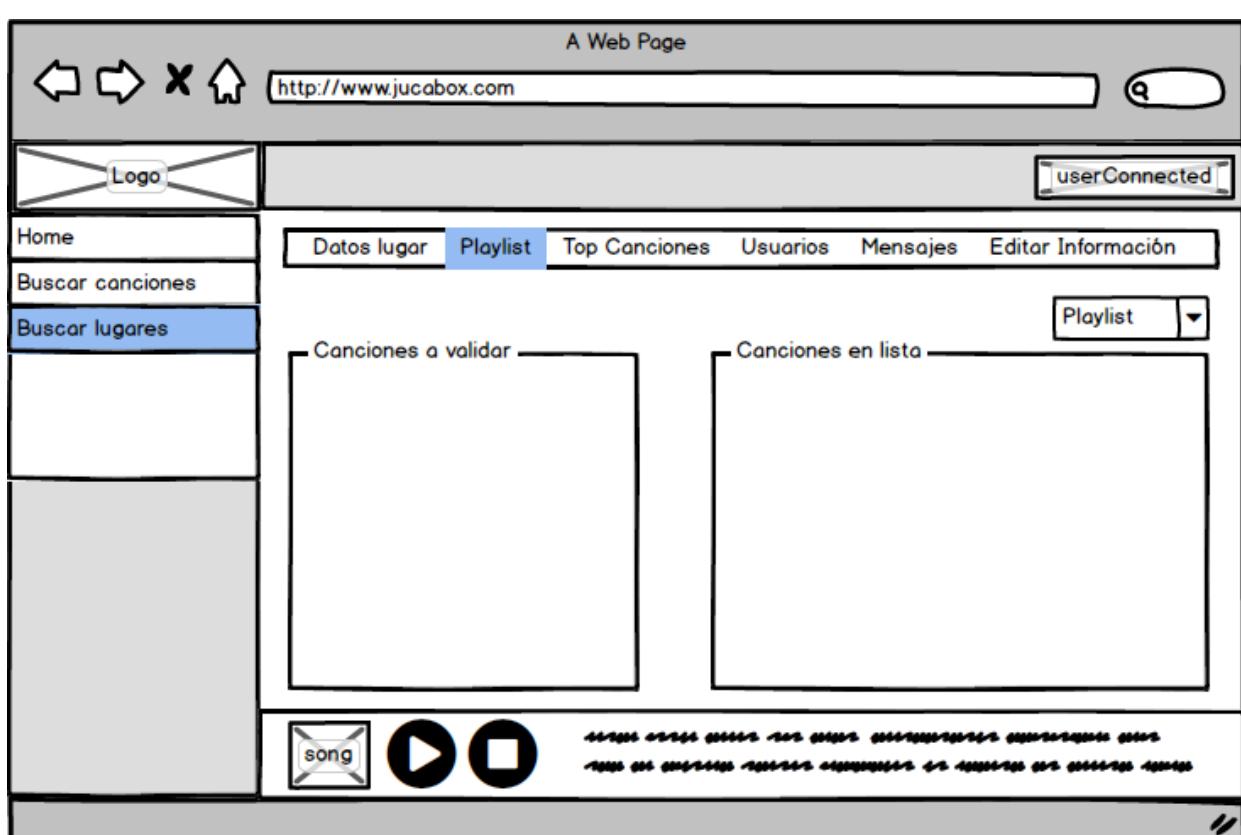
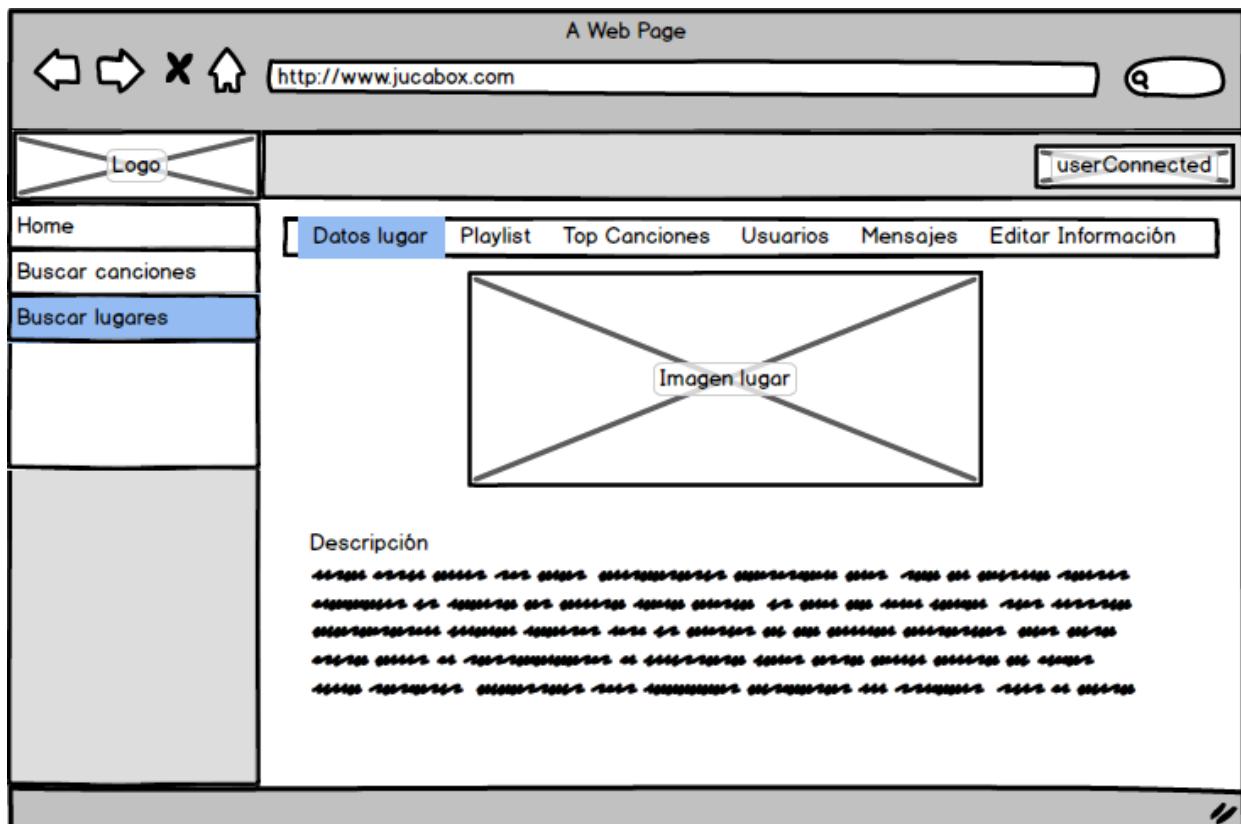


Figura 4.27 – Prototipado – Buscar lugar



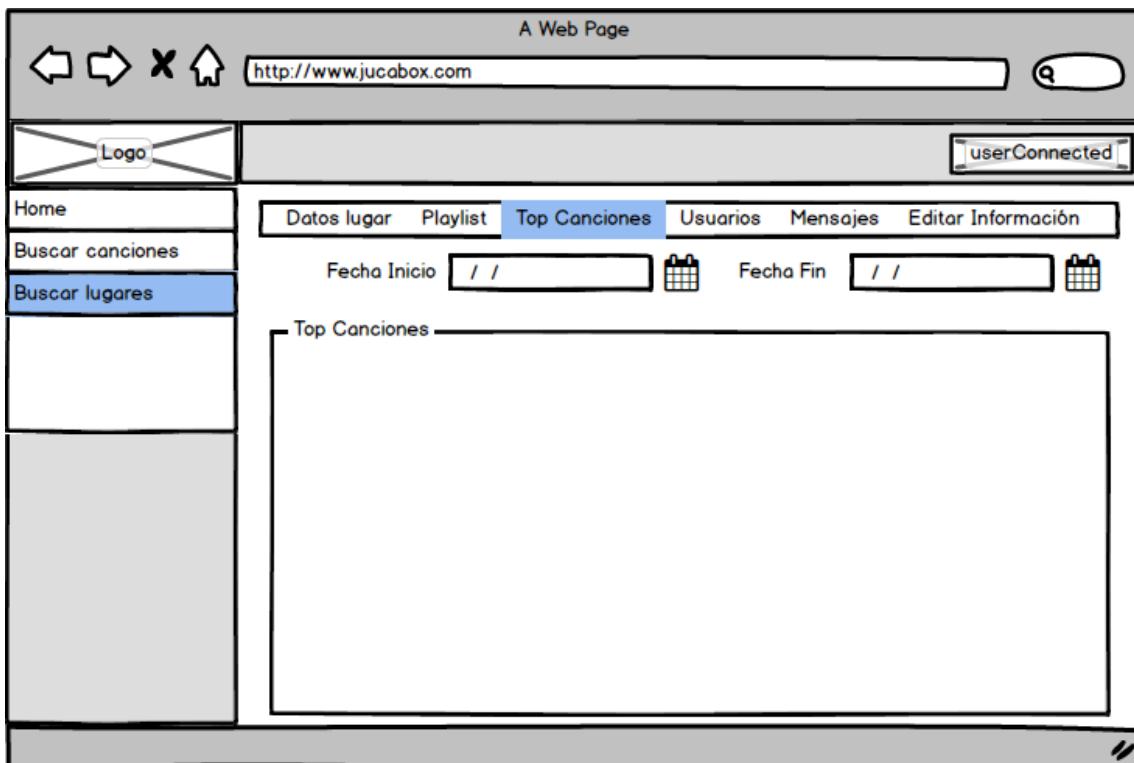


Figura 4.30 – Prototipado – Top canciones

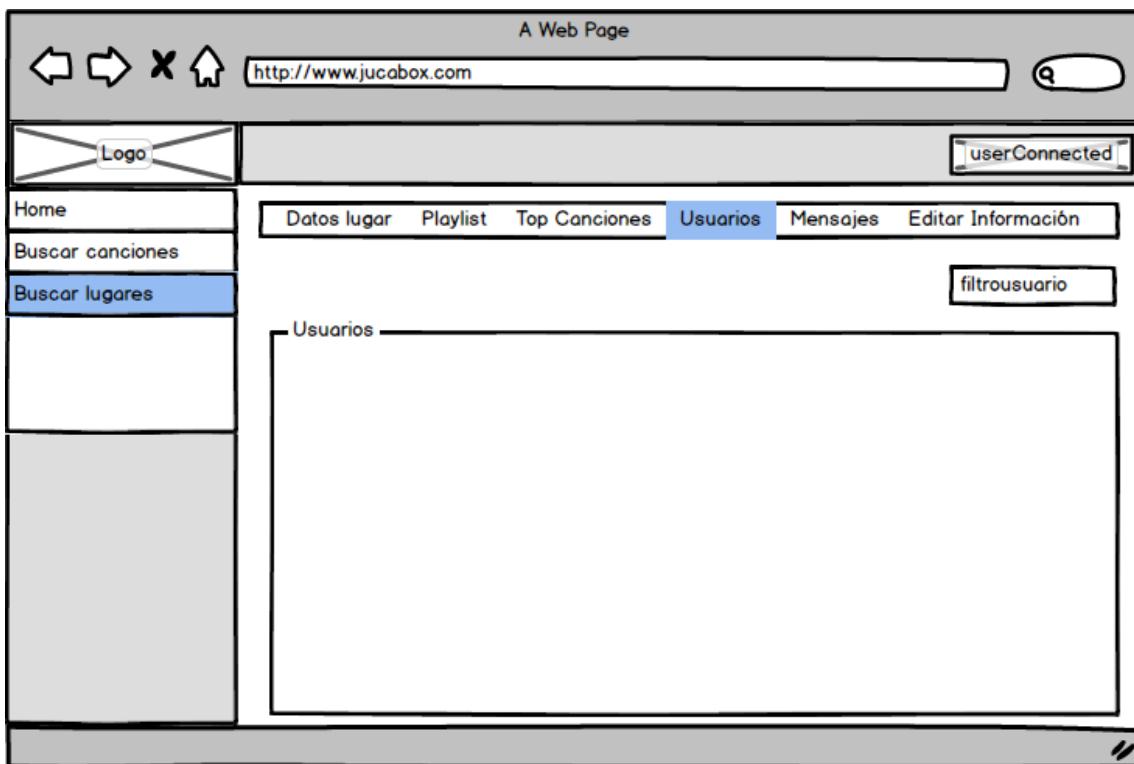


Figura 4.31 – Prototipado – Usuarios Lugar

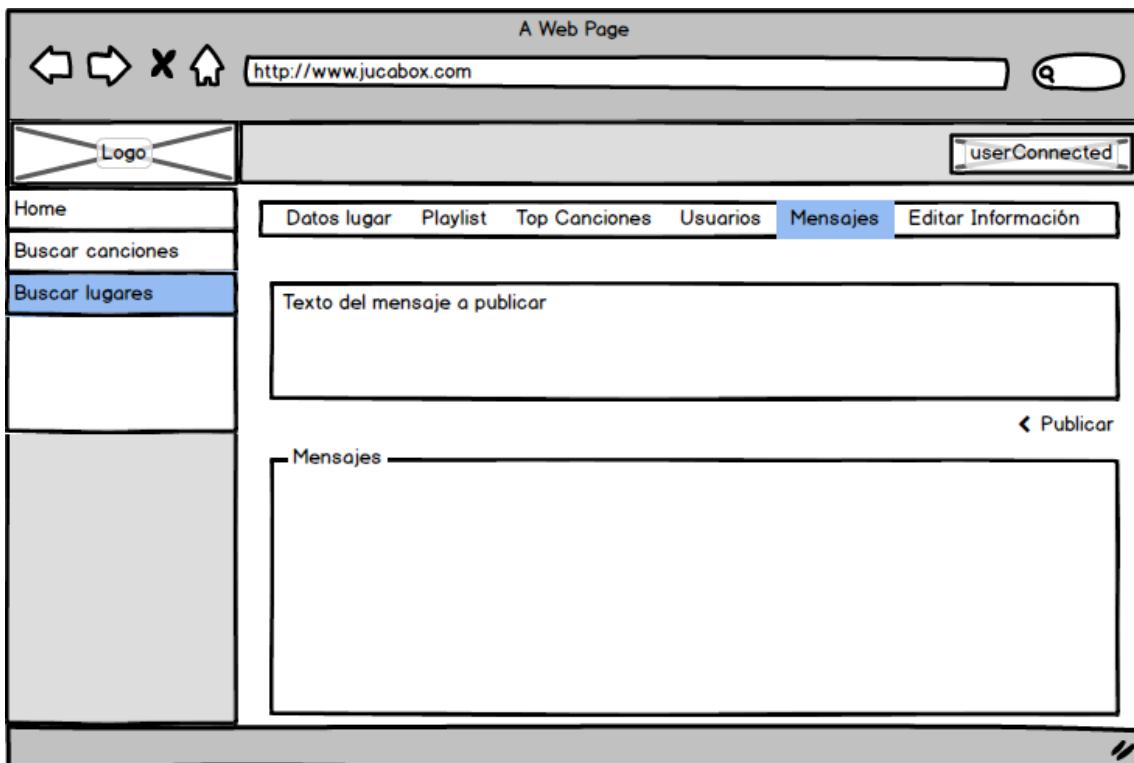


Figura 4.32 – Prototipado – Mensajes lugar

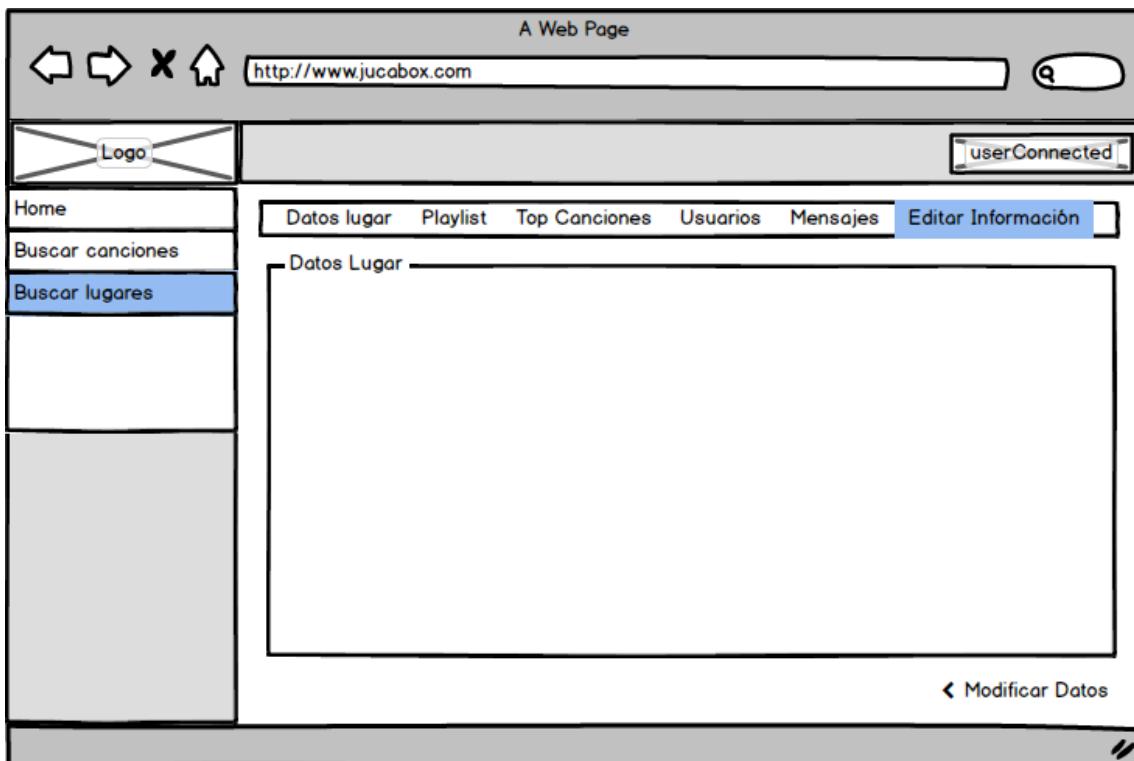


Figura 4.33 – Prototipado – Editar información Lugar

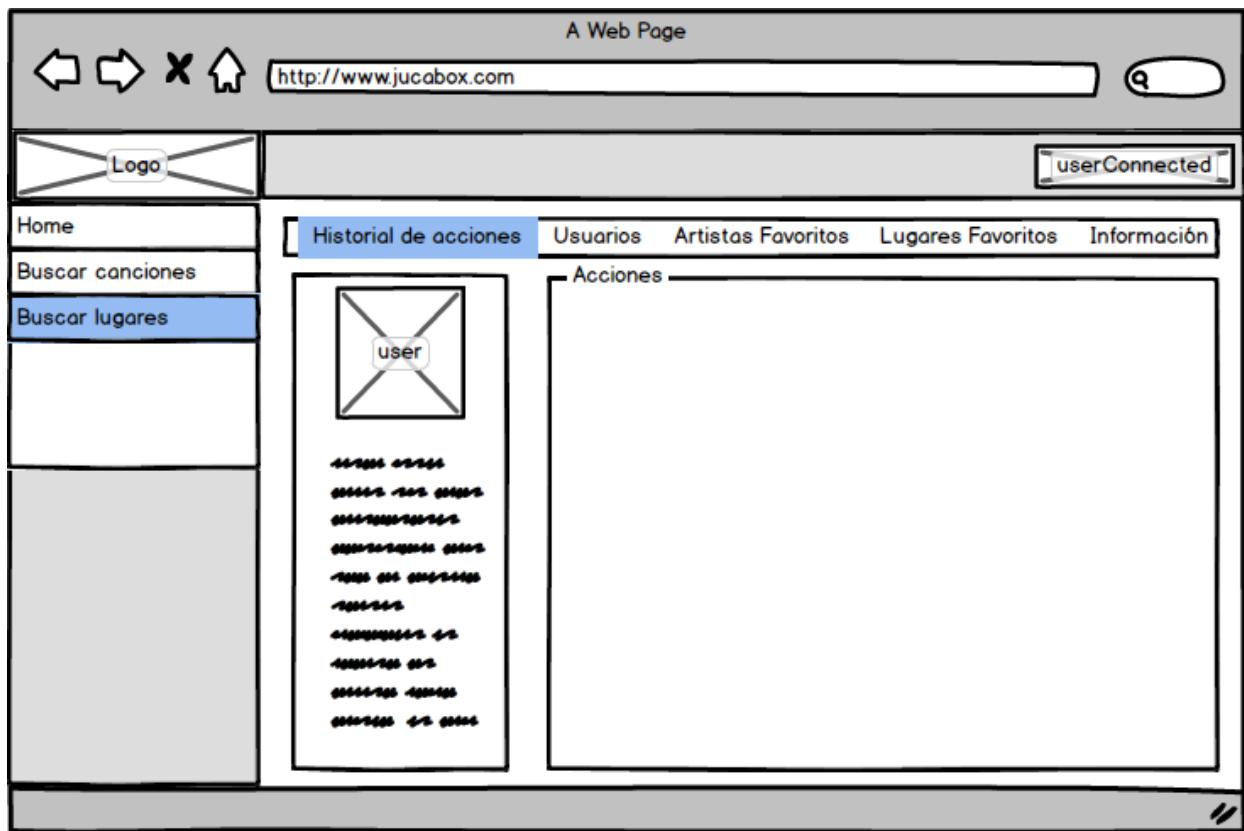


Figura 4.34 – Prototipado – Historial Acciones Usuario

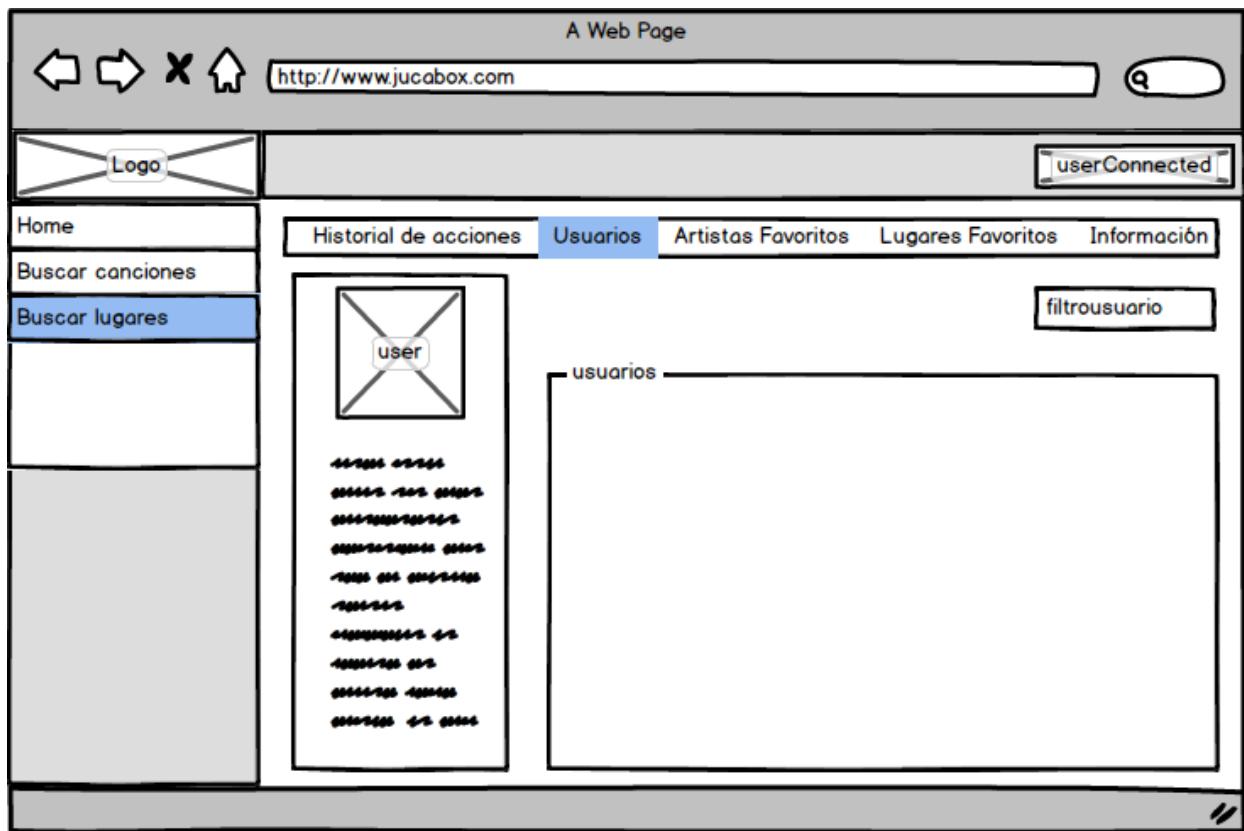


Figura 4.35 – Prototipado – Amigos de Usuario

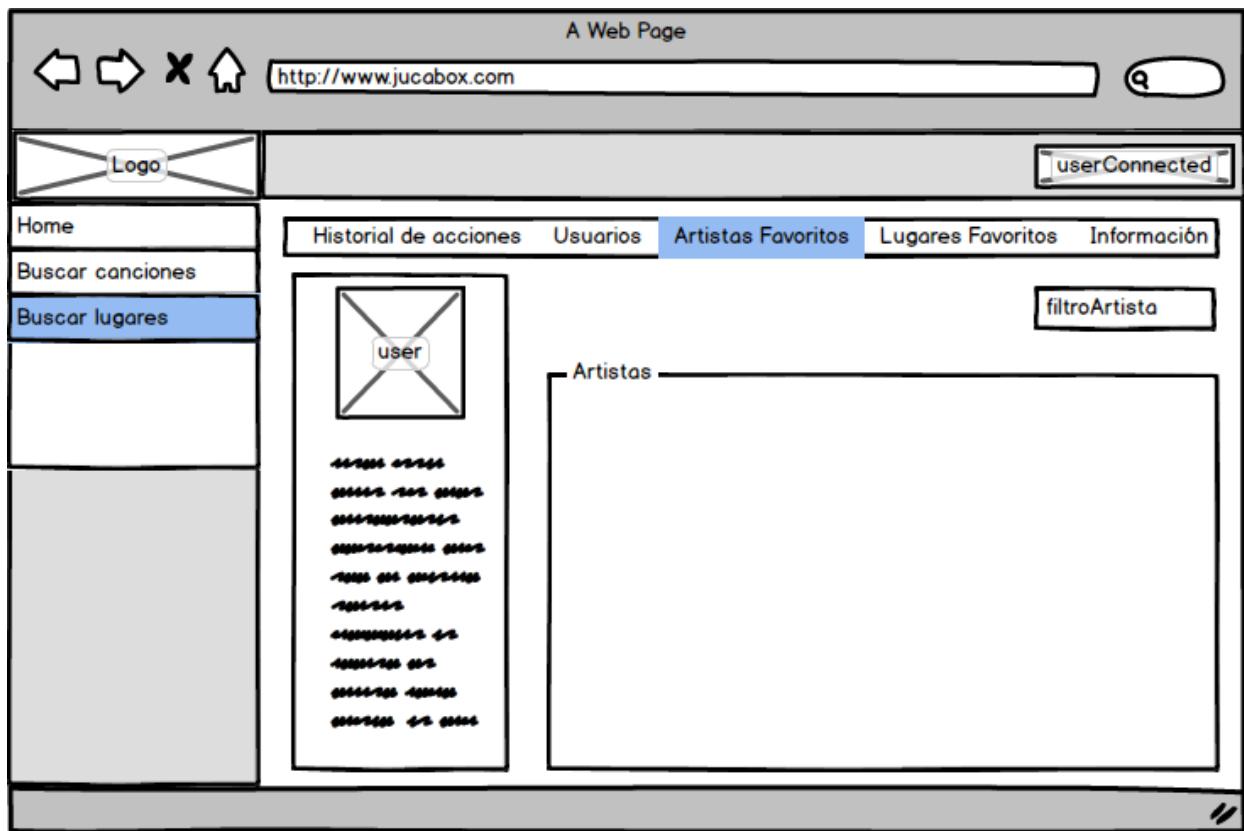


Figura 4.36 – Prototipado – Artistas Favoritos Usuario

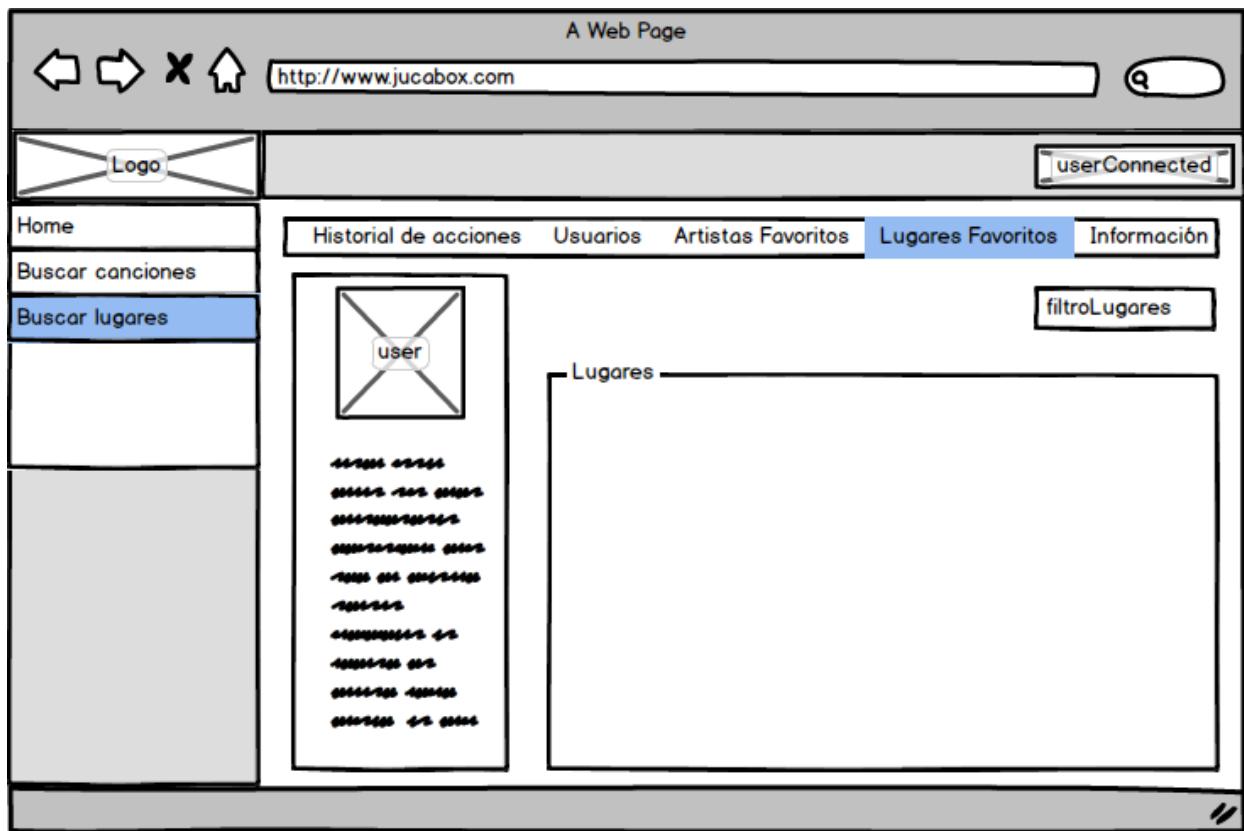


Figura 4.37 – Prototipado – Lugares Favoritos Usuario

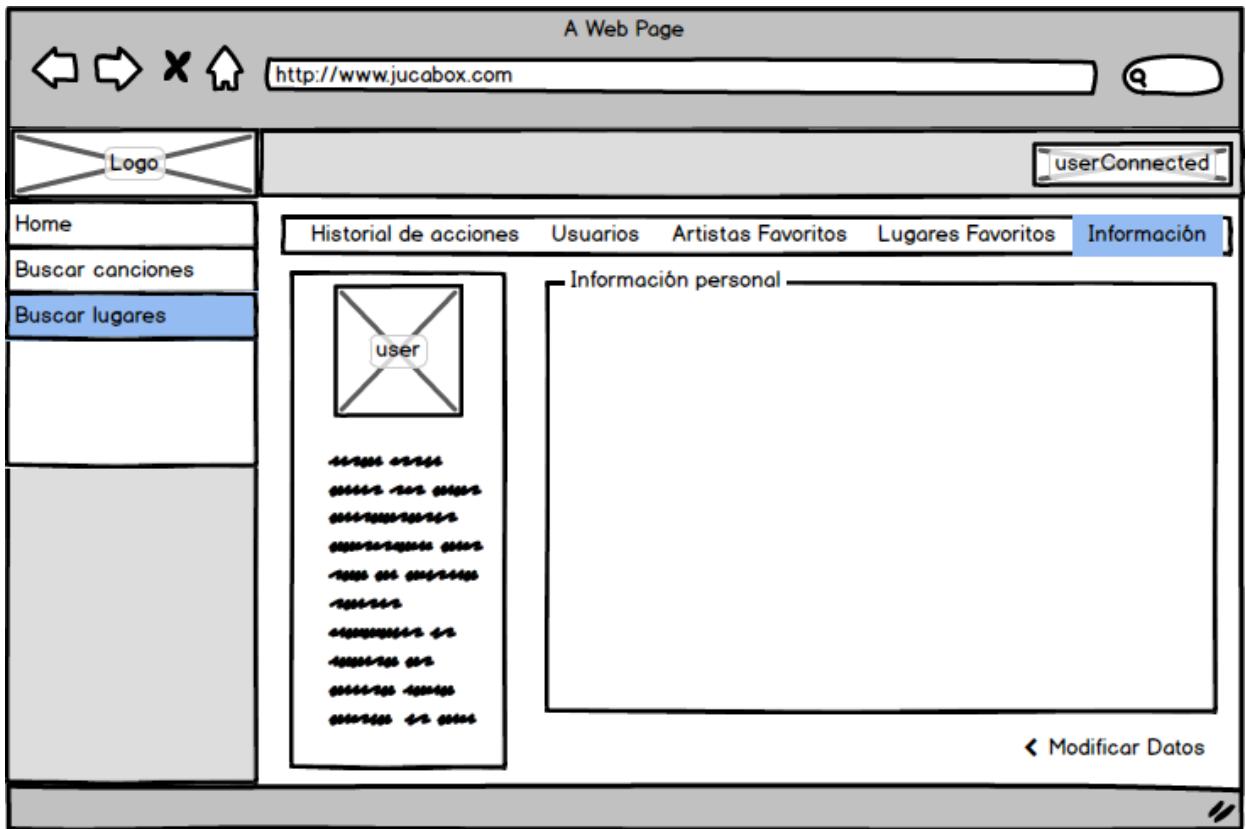


Figura 4.38 – Prototipado – Información Personal Usuario

A continuación se muestran los diagramas de interacción de la aplicación Web. A través de ellos se puede observar la navegación por el sistema.

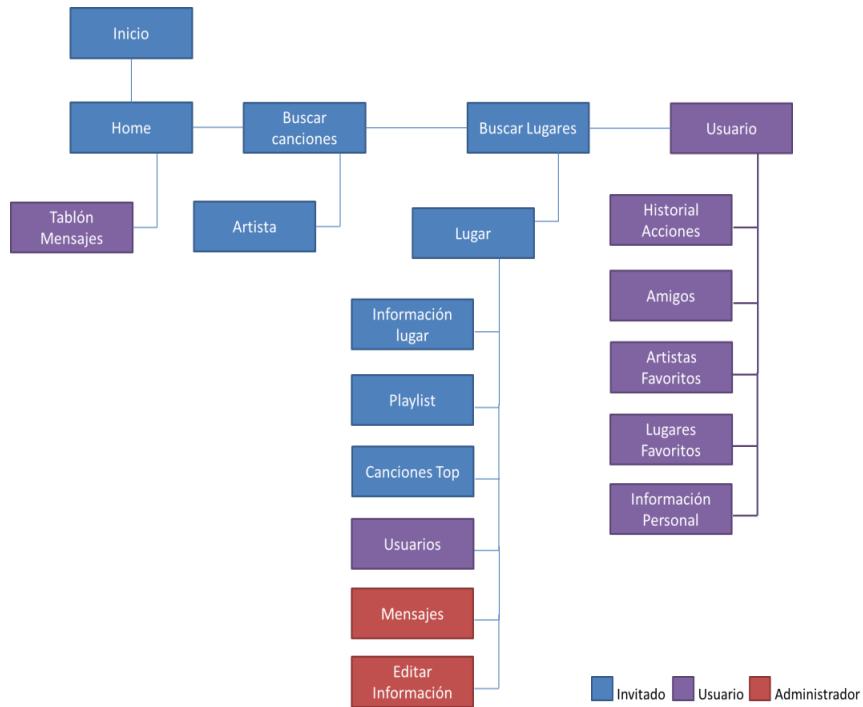


Figura 4.39 – Diagrama de interacción jUCAblox

# Capítulo 5

## Diseño del Sistema

En esta sección se recoge la arquitectura general del sistema de información, el diseño físico de datos, el diseño detallado de componentes software y el diseño detallado de la interfaz de usuario.

### 5.1. Arquitectura del Sistema

En esta sección se define la arquitectura general del sistema de información, donde se especifica la infraestructura tecnológica necesaria para dar soporte al software y la estructura de los componentes que lo conforman.

#### 5.1.1. Arquitectura Física

En este apartado se va a definir la arquitectura utilizada para el desarrollo del sistema, tanto de la parte cliente como de la parte servidor.

En el lado del servidor, vamos a necesitar una infraestructura necesaria para albergar MongoDB 3.2.11, que será nuestro sistema de base de datos. A su vez debe tener instaladas las dependencias de Express 4.15.13 para poder comunicarse con Node.js 6.11.10 que a su vez incorpora el inyector de dependencias npm 3.10.10.

Para la ejecución del software, es necesario únicamente conexión a internet y un navegador web. No existen limitaciones sobre el navegador, ni de dispositivos desde el cual pueda visualizar la web.

#### 5.1.2. Arquitectura Lógica

En este apartado se va a describir la arquitectura lógica de la aplicación.

En la siguiente figura se muestra el diagrama de componentes y las dependencias existentes de la herramienta.

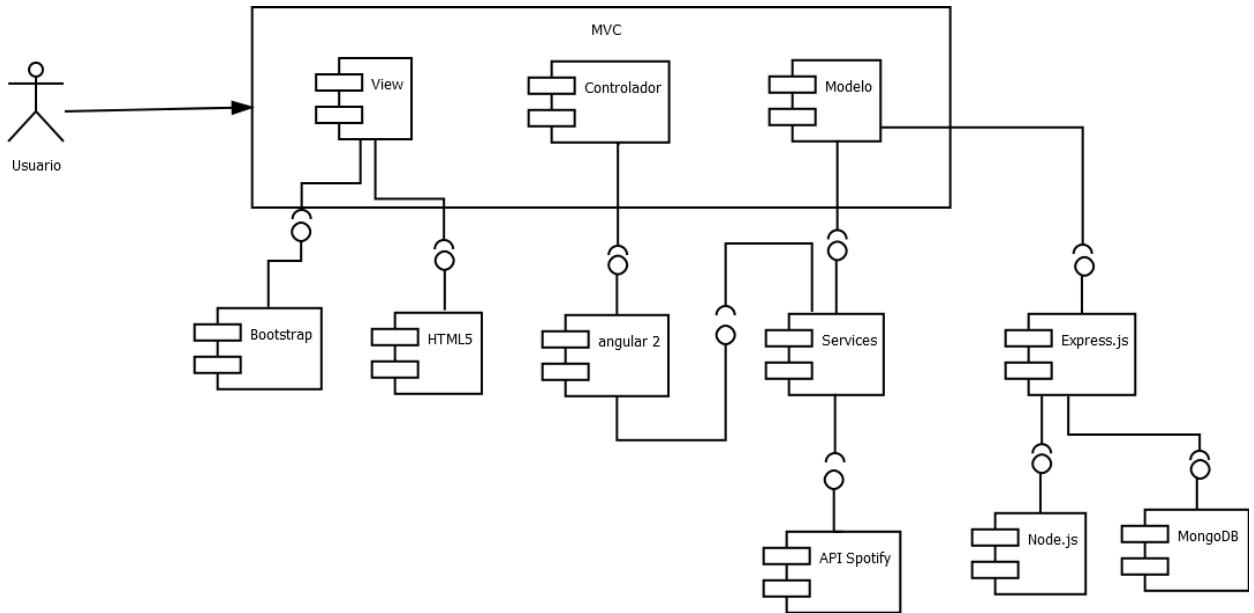


Figura 5.1 – Diagrama de componentes

Para la arquitectura de la aplicación web se utilizó el patrón de diseño MVC (Modelo – Vista – Controlador) pues es en el que se basa el MEAN Stack. En este patrón de diseño, se separa la lógica de negocio, de la lógica visual. El patrón de diseño se divide en tres partes anteriormente mencionadas:

- **Modelo**

- Es el encargado de controlar los datos, así como su acceso y modificación. En el cual se implementan los roles de acceso y permisos para dichas consultas. En nuestra aplicación el encargado es MongoDB, que a través de Express.js y Node.js, hacen que el acceso a los datos sean seguro.

- **Vista**

- Es el encargado de renderizar los datos proporcionado por el controlador, es la capa visual que puede ver y utilizar el usuario. Es el que muestra la interfaz de usuario. En nuestra aplicación es controlado por Angular2, mediante el ViewModel.

- **Controlador**

- Es el encargado de invocar peticiones y responde a eventos cuando se solicita una petición a través de la vista. Esta función la cubre también Angular2, a través de su interacción con los Templates de html5.

## 5.2. Diseño Físico de Datos

En esta sección se define la estructura física de datos que utilizará el sistema, a partir del modelo de conceptual de clases, de manera que teniendo presente los requisitos establecidos para el sistema de información y las particularidades del entorno

tecnológico, se consiga un acceso eficiente de los datos. La estructura física en este caso difiere a la de los sistemas relacionales. MongoDB al ser un sistema NoSQL, no contiene tablas ni registros, sino colecciones y documentos.

Este tipo de bases de datos son utilizadas para el conocido mundo BIG Data, con lo que pueden manejar gran cantidad de datos de manera muy eficiente. El modelo de datos se considera muy flexible, pues dos documentos de una misma colección podrían no contener los mismos campos.

Aun así, se establecen unos modelos básicos, donde se definen que campos puede tener un documento, no se consideran obligatorios, pero si deben de cumplir con el tipo de datos definido.

Estas bases de datos, se basan en documentos JSON, en el caso de MongoDB los llama BSON, son documentos de pares clave-valor.

Los tipos de datos soportados por cada clave-valor son los siguientes:

```

string: String
binary: Buffer
boolean: Boolean
mixed: Schema.Types.Mixed
_someId: Schema.Types.ObjectId,
array: [],
ofString: [String],
ofNumber: [Number],
ofDates: [Date],
ofBuffer: [Buffer],
ofBoolean: [Boolean],
ofMixed: [Schema.Types.Mixed],
ofObjectId: [Schema.Types.ObjectId],
nested: {
  stuff: { type: String, lowercase: true, trim: true }
}

```

Figura 5.2 – Tipos de datos MongoDB

Cabe recordar que en MongoDB, existe un identificador autonumérico, que nunca se define y se crea por defecto que es el `_id`. Siendo único en toda la colección, y de tipo Objectid.

En los sistemas relaciones, si existe una relación M:N se crearía una tabla intermedia. Por ejemplo, en el caso de Lugares Favoritos, se crearía una tabla de relación con Usuario-Lugares. Pero en las no relacionales directamente en el modelo Usuario, se añadiría un nuevo atributo de tipo Lugar, el cual, sería un array de lugares.

```

1  Usuario = {
2    _id: ObjectId("32309cdzcx98cl"),
3    userIDExterno: 'googleauth|90033944',
4    firstname: 'Sergio',
5    lastname: 'Ruiz Piulestan',
6    creationDate: '01/06/2017',
7    city: 'Cádiz',
8    province: 'Cádiz',
9    country: 'Spain',
10   nickname: 'Kodorniz',
11   LugaresFav: [
12     ObjectId("32930cdfds8cl"), ObjectId("32we456zcx98cl"), ObjectId("32309cdz098779")
13   ]
14 }

```

Figura 5.3 – Objeto Usuario

Mediante el método populate de MongoDB, se podría extraer los datos del lugar, puesto que en la definición del modelo Usuario, se ha indicado que `LugaresFav` es un array de

lugares.

Cabe destacar que en los modelos no relacionales, los campos no son obligatorios.

A continuación se van a definir los modelos en los que se va a basar nuestra herramienta:

- Usuario

```
var UserSchema = Schema({
  userID: String,
  firstname: String,
  lastname: String,
  email: String,
  avatarUrl: String,
  creationDate: Date,
  preferredLang: String,
  clientID: String,
  GlobalClientID: String,
  ciudad: String,
  provincia: String,
  pais: String,
  nickName: String,
  online: String,
  ArtistasFav: [String],
  usersFriend: [{ type: Schema.ObjectId, ref: 'User' }],
  lugaresFav: [{ type: Schema.ObjectId, ref: 'Lugar' }]
});
```

Figura 5.4 – Objeto de definición de la colección Usuario

Como se puede observar en UsersFriends, es un array de objetos de identificadores de usuario. En la base de datos solo se guardaría los identificadores que hace referencia, pero luego cuando se obtienen los datos se puede hacer una población (populate) de los mismos para extraer toda la información.

Nombre de campo	Tipo de datos	Descripción
<b>userID</b>	String	Id externo proporcionado por el plugin Autho
<b>firstname</b>	String	Nombre del usuario
<b>lastname</b>	String	Apellidos del usuario
<b>email</b>	String	Correo del usuario
<b>avatarUrl</b>	String	Dirección url de la imagen del usuario
<b>creationDate</b>	Date	Fecha de registro del usuario
<b>preferredLanguage</b>	String	Lenguaje de origen del usuario

<b>clientID</b>	String	Id externo proporcionado por el plugin Autho
<b>globalclientID</b>	String	Id externo proporcionado por el plugin Autho
<b>ciudad</b>	String	Ciudad del usuario
<b>provincia</b>	String	Provincia del usuario
<b>país</b>	String	País del usuario
<b>nickname</b>	String	Alias del usuario
<b>online</b>	String	Estado de conexión del usuario
<b>artistasFav</b>	Array de String	Artistas favoritos del usuario. Al no ser una colección de jUCAbbox, guardamos el objeto completo
<b>userFriends</b>	Array de ObjectId de users	Usuarios agregados como amigos
<b>lugaresFav</b>	Array de ObjectId de lugares	Lugares favoritos del usuario

Tabla 5.1 – Campos entidad usuario

- Lugar

```
var LugaresSchema = Schema({
  nombre: String,
  descripcion: String,
  img: [String],
  provincia: String,
  ciudad: String,
  email: String,
  direccion: String,
  userAdmin: { type: Schema.ObjectId, ref: 'User' },
  tipoMusica: [Schema.Types.Mixed],
  token: String,
  playLists: [{ type: Schema.ObjectId, ref: 'Playlist' }]
});
```

Figura 5.5 – Objeto de definición de la colección Lugar

<b>Nombre de campo</b>	<b>Tipo de datos</b>	<b>Descripción</b>
<b>nombre</b>	String	Nombre del lugar
<b>descripción</b>	String	Descripción del lugar
<b>img</b>	Array de String	Rutas de imágenes a mostrar
<b>provincia</b>	String	Provincia del lugar
<b>ciudad</b>	String	Ciudad del lugar
<b>email</b>	String	Correo del lugar
<b>direccion</b>	String	Dirección física del lugar para situarlo en google maps
<b>userAdmin</b>	ObjectID de usuario	Id del usuario administrador
<b>tipoMusica</b>	Array de objetos	Tipo de música del local
<b>token</b>	String	Clave por la cual se comprueba para que el usuario pueda enviar una canción
<b>playlist</b>	Array de ObjectID de Playlist	Id de las playlists del lugar

Tabla 5.2 – Campos entidad lugar

- Playlist

```
var playList = Schema({
  playlistID: String,
  namePlaylist: String,
  userID: String
  cancionesEnviadas: [{ type:Schema.ObjectId, ref:'Cancion'}]
});
```

Figura 5.6 – Objeto de definición de la colección Playlist

Nombre de campo	Tipo de datos	Descripción
<b>playlistID</b>	String	Id de la playlist en Spotify
<b>namePlaylist</b>	String	Nombre de la playlist
<b>userID</b>	String	Usuario de Spotify owner de la playlist
<b>cancionesEnviadas</b>	Array de ObjectId de Canciones	Canciones enviadas a la playlist

Tabla 5.3 – Campos entidad playlist

- Canción

```
var Cancion = Schema({
  cancion: Schema.Types.Mixed,
  FechaEnvio: Date,
  userID: { type: Schema.ObjectId, ref: 'User' },
  Estado: String
});
```

Figura 5.7 – Objeto de definición de la colección Canción

Nombre de campo	Tipo de datos	Descripción
<b>canción</b>	Objeto JSON	Objeto completo de la canción de Spotify
<b>Fecha Envío</b>	Date	Fecha de Envío de la canción a jUCAbbox
<b>Estado</b>	String	Estado de la canción. Si está pendiente, validada o rechazada
<b>userID</b>	String	Usuario que ha solicitado la canción

Tabla 5.4 – Campos entidad canción

- Log

```
var LogSchema = Schema({
  userID: { type: Schema.ObjectId, ref: 'User' },
  tipoMensaje: String,
  mensaje: String,
  FechaLog: Date,
});
```

Figura 5.8 – Objeto de definición de la colección Log

Log es una colección donde se almacenan los eventos de la aplicación.

Nombre de campo	Tipo de datos	Descripción
<b>userID</b>	ObjectID de usuario	Id del usuario que ha realizado la acción
<b>tipoMensaje</b>	String	La categoría del mensaje, si es envío canción, creación de lugar, validación de canción o agregar amigo
<b>mensaje</b>	String	Cuerpo del log
<b>FechaLog</b>	Date	Fecha cuando se realiza la acción

Tabla 5.4 – Campos entidad log

Al no ser un sistema relacional, el esquema relacional se basará en el modelo conceptual de datos ya descrito en capítulos anteriores.

### 5.3. Diseño detallado de Componentes

A continuación vamos a describir algunos diagramas de secuencia de los módulos funcionales de la herramienta. Solo mostraremos los más importantes, debido a las similitudes con otros.

- **Caso de uso: Búsqueda de ítems Spotify**

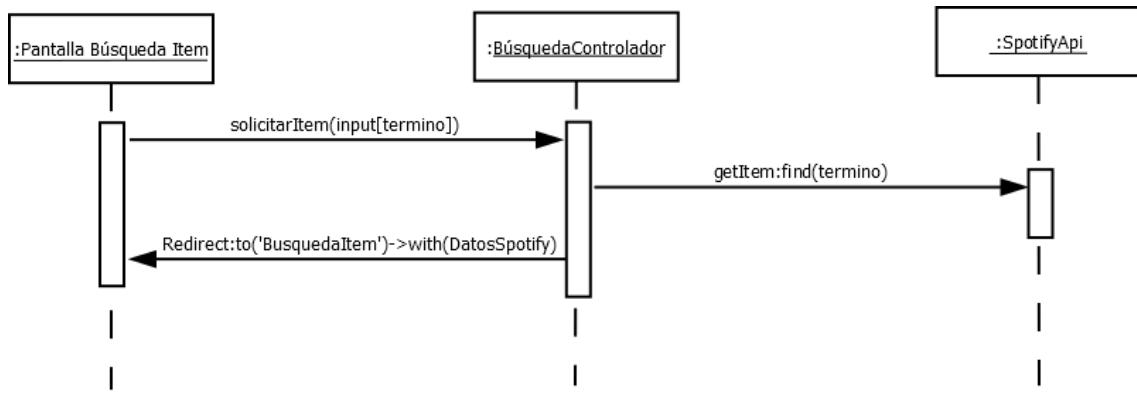


Figura 5.9 – Diagrama de secuencia – Caso de uso búsqueda de ítems

- **Caso de uso: Enviar Canción**

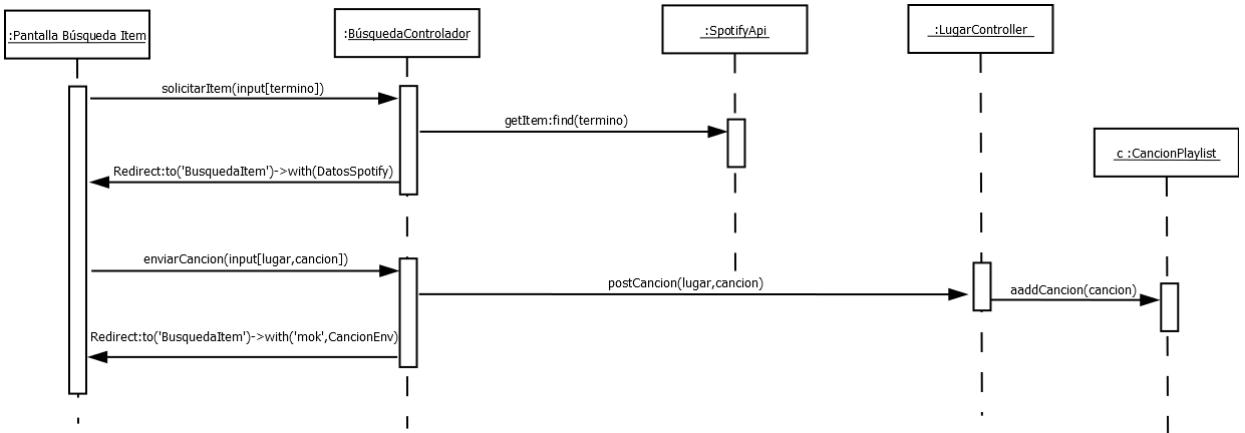


Figura 5.10 – Diagrama de secuencia – Caso de uso enviar canción

- **Caso de uso: Añadir artista a favoritos**

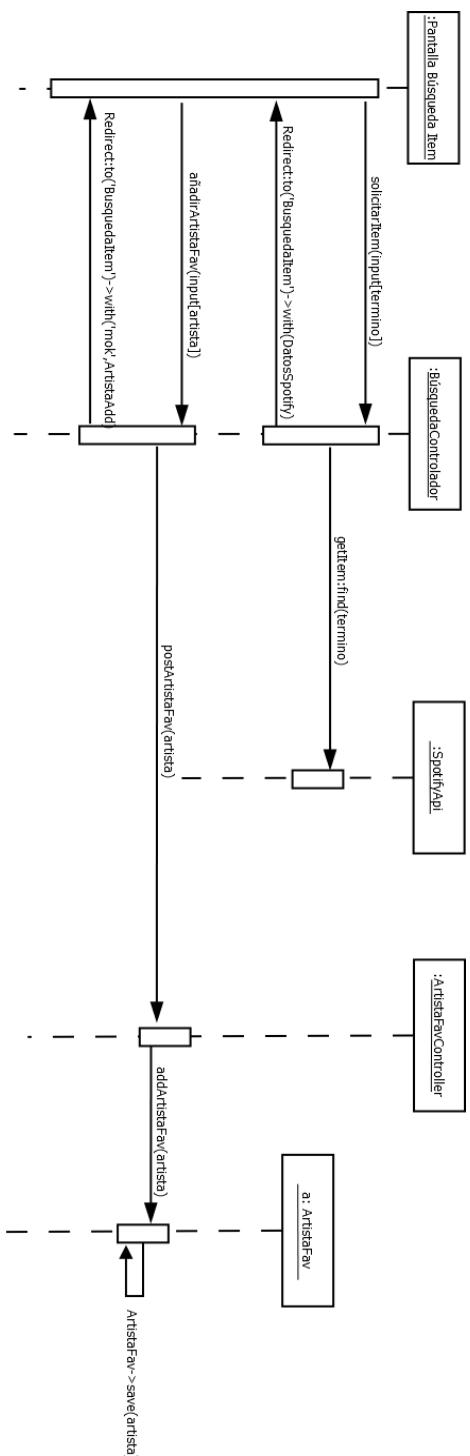


Figura 5.11 – Diagrama de secuencia – Caso de uso añadir artista a favoritos

- **Caso de uso:** Eliminar artista de favoritos

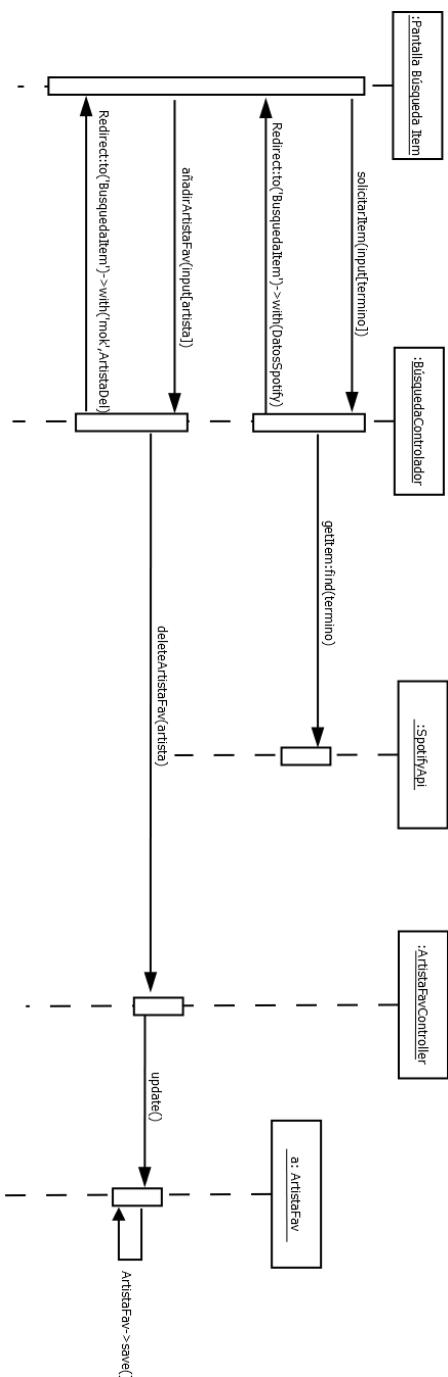


Figura 5.12 – Diagrama de secuencia – Caso de uso eliminar artista de favoritos

- **Caso de uso: Registro en el sistema**

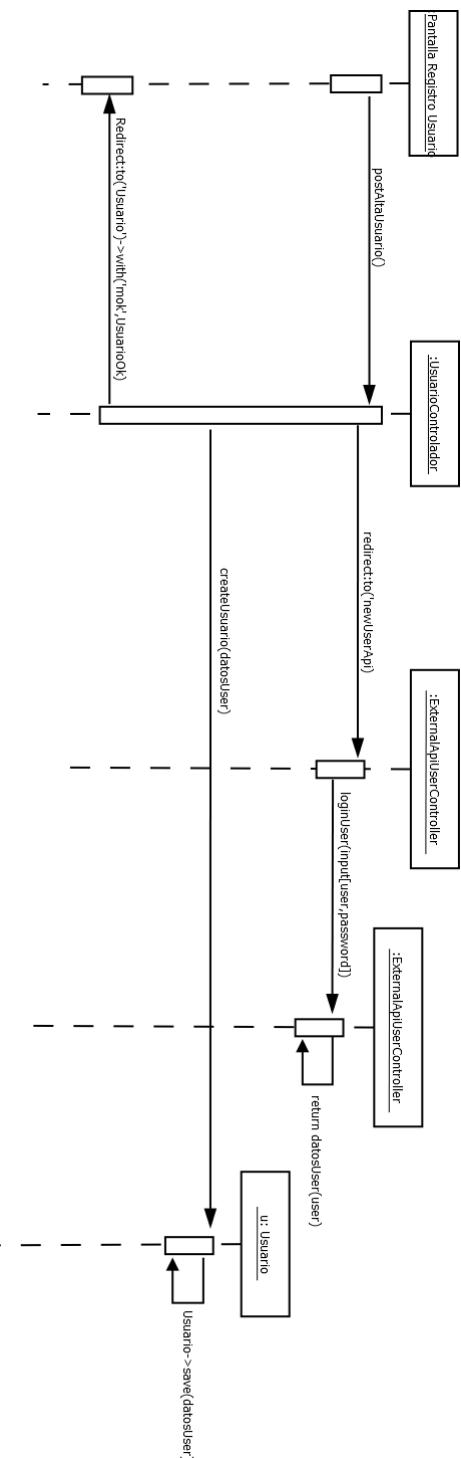


Figura 5.13 – Diagrama de secuencia – Caso de uso registro en el sistema

- **Caso de uso: Ver log actividad**

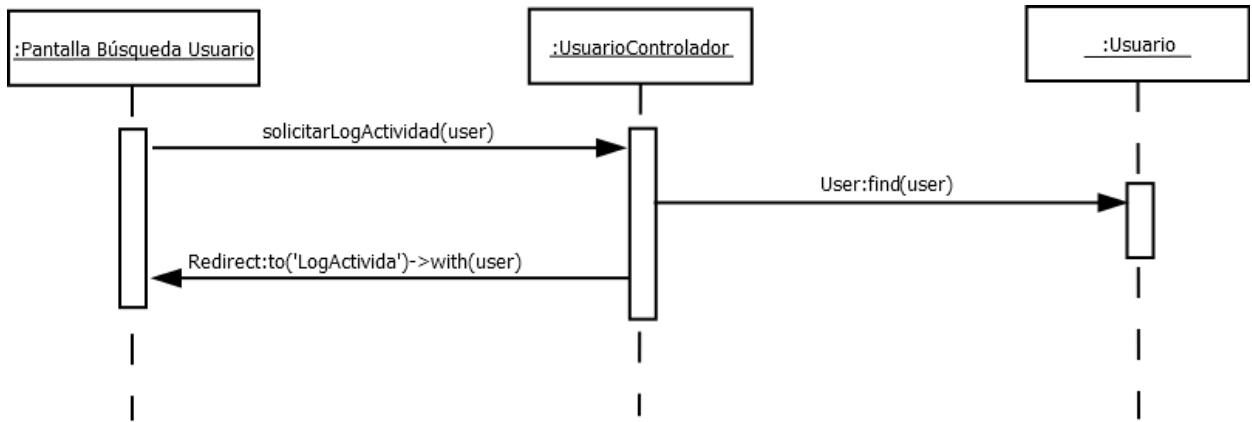


Figura 5.14 – Diagrama de secuencia – Caso de uso ver log actividad

- **Caso de uso: Buscar usuarios**

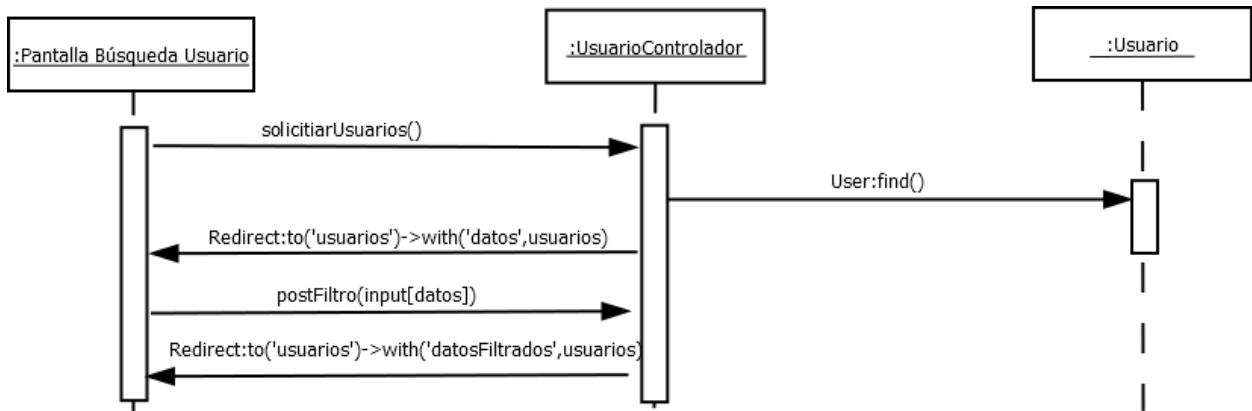


Figura 5.15 – Diagrama de secuencia – Caso de uso buscar usuarios

- **Caso de uso: Agregar nuevo amigo**

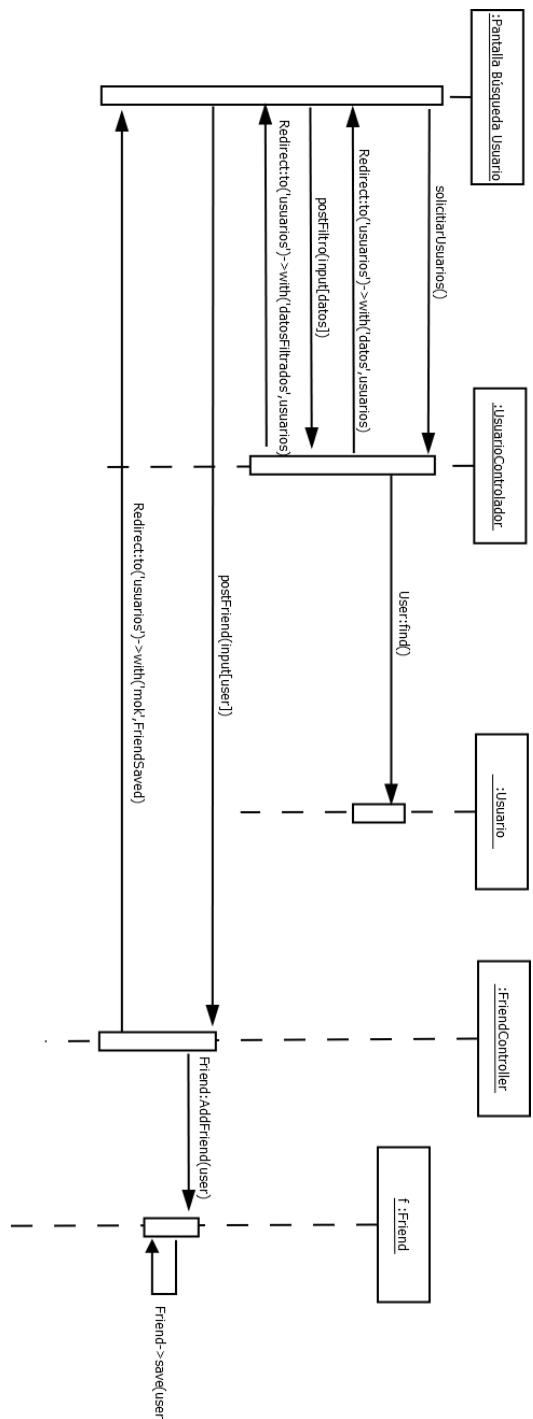


Figura 5.16 – Diagrama de secuencia – Caso de uso Agregar nuevo amigo

- **Caso de uso: Agregar nuevo amigo**

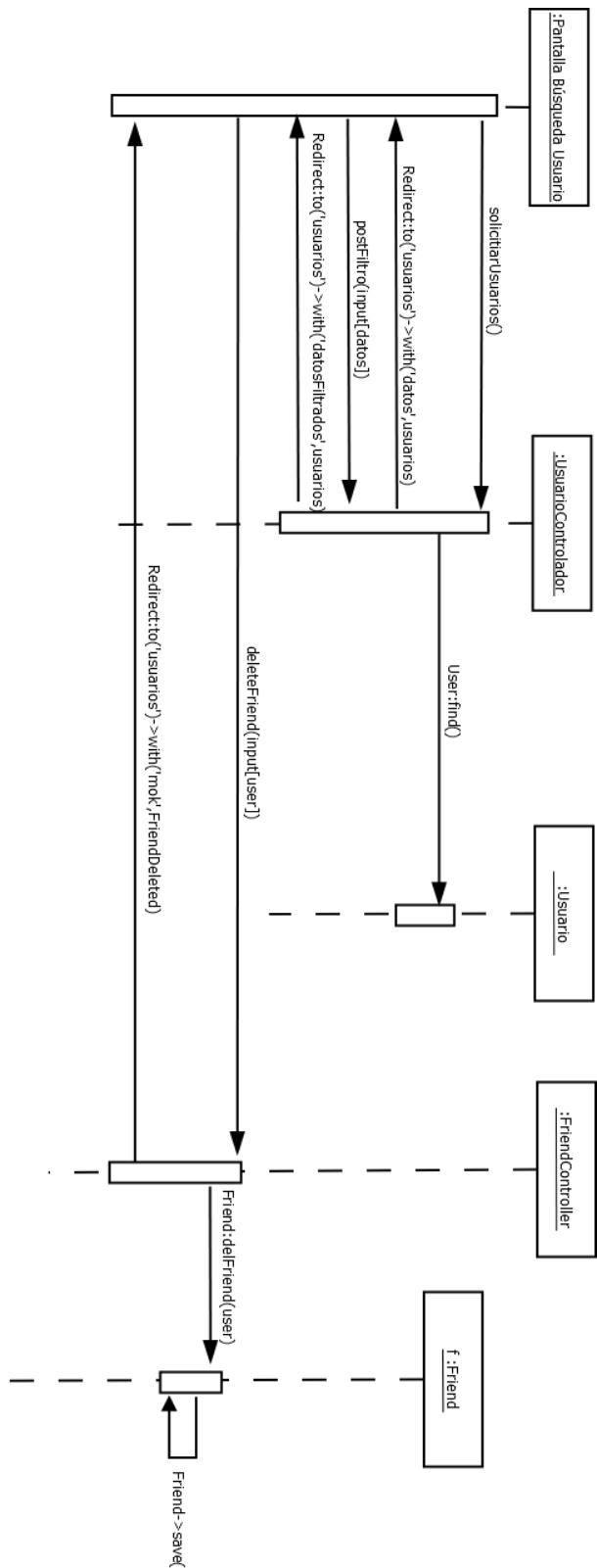


Figura 5.17 – Diagrama de secuencia – Caso de uso Agregar nuevo amigo

- **Caso de uso: Editar información personal**

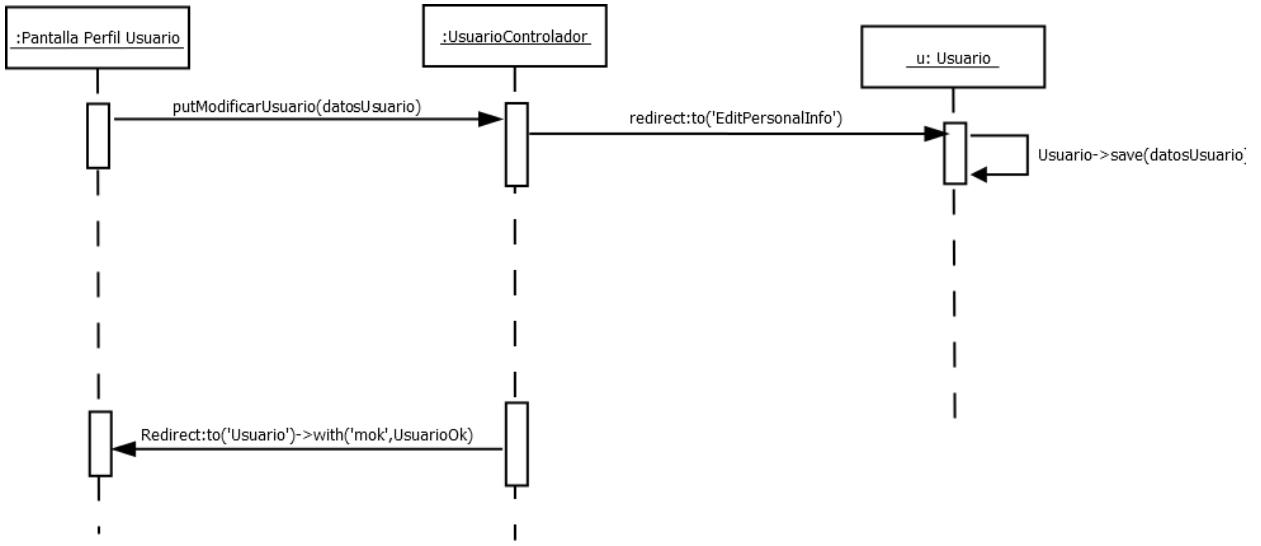


Figura 5.18 – Diagrama de secuencia – Caso de uso editar información personal

- **Caso de uso: Buscar lugares**

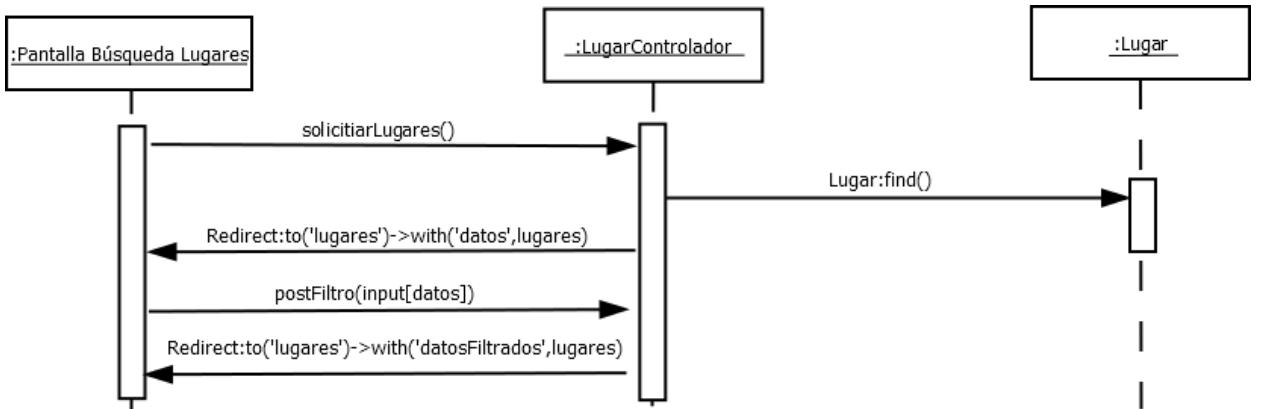


Figura 5.19 – Diagrama de secuencia – Caso de uso buscar lugares

- **Caso de uso:** Añadir lugar a favoritos

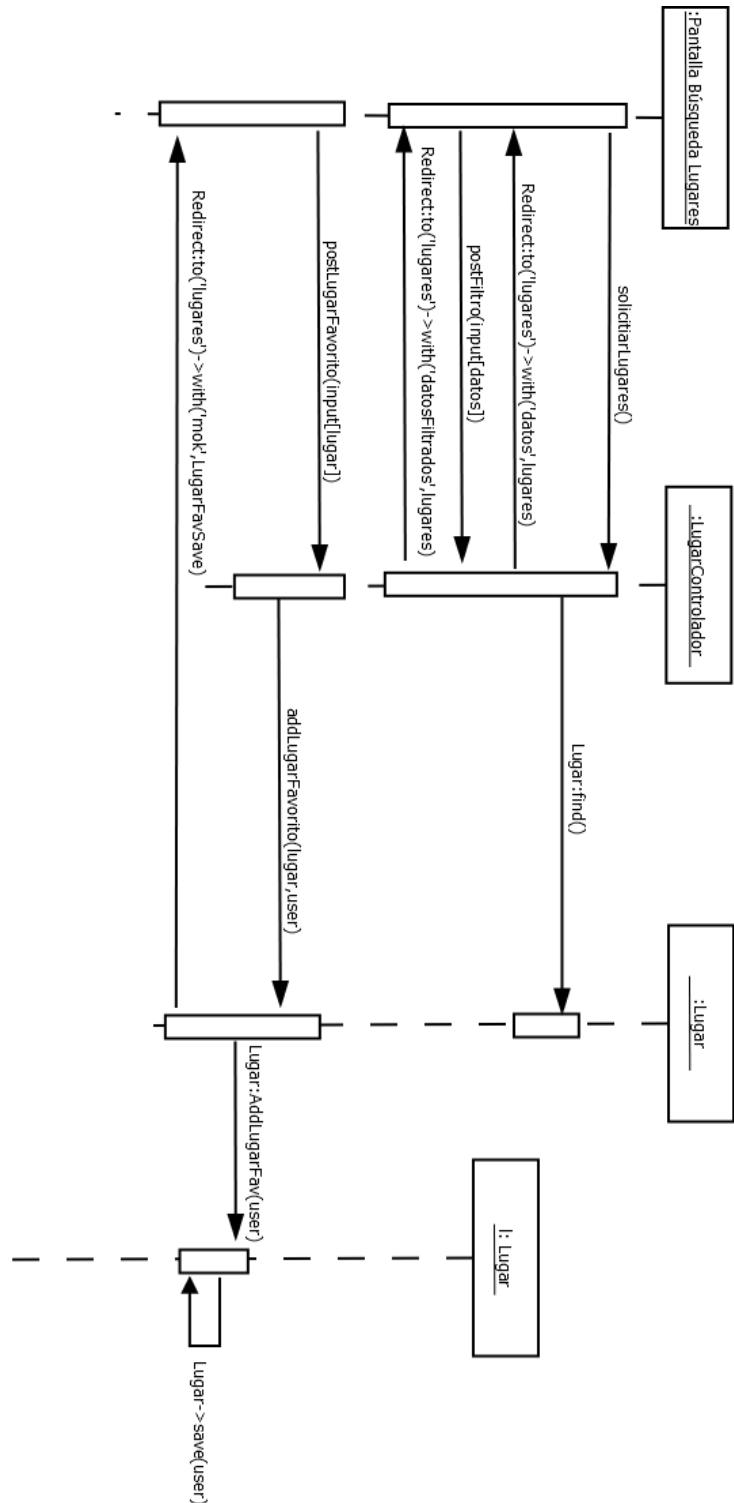


Figura 5.20 – Diagrama de secuencia – Caso de uso añadir lugar a favoritos

- **Caso de uso: Eliminar lugar de favoritos**

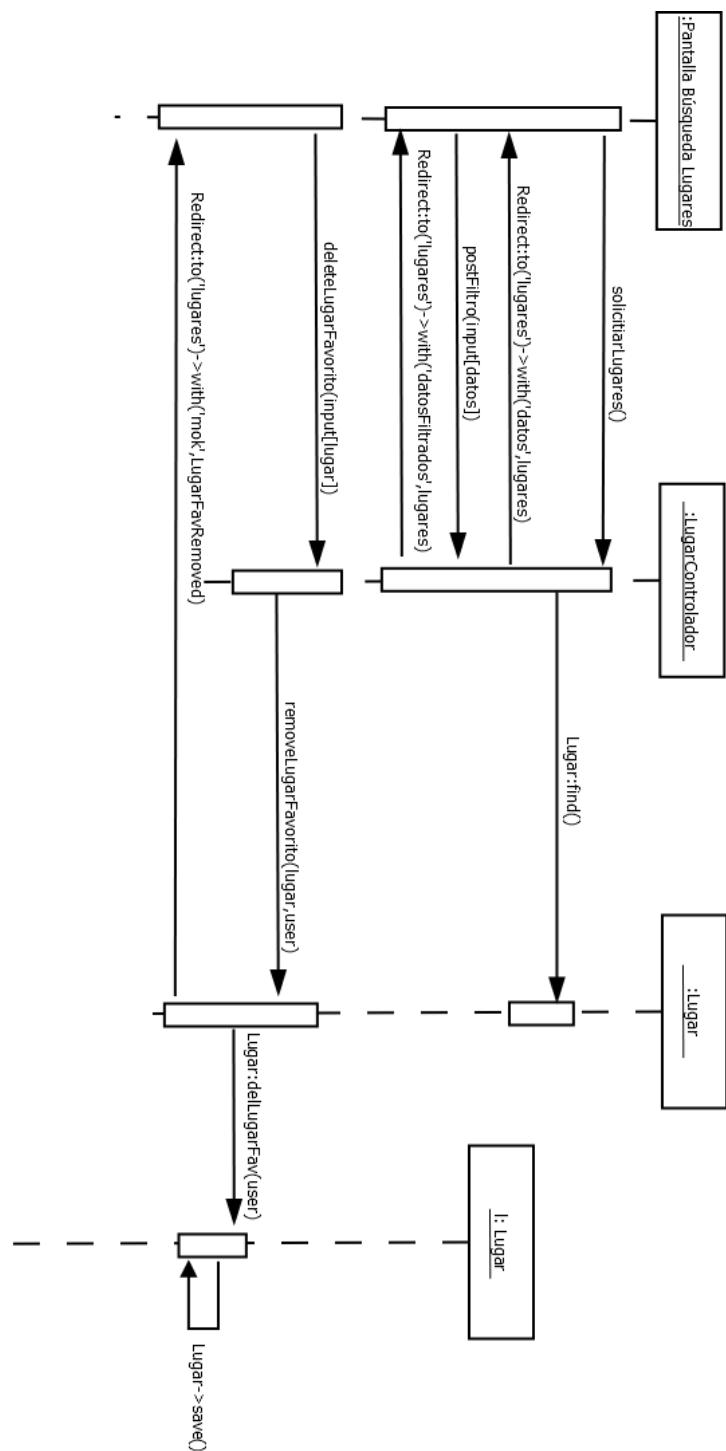


Figura 5.21 – Diagrama de secuencia – Caso de uso eliminar lugar de favoritos

- **Caso de uso: Crear lugar**

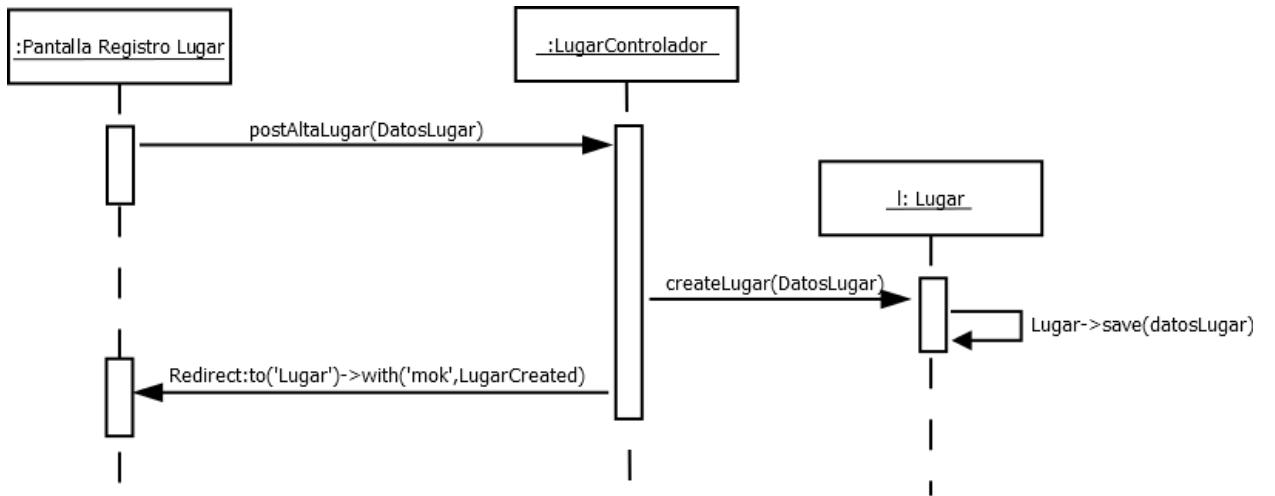


Figura 5.22 – Diagrama de secuencia – Caso de uso crear lugar

- **Caso de uso: Publicar mensaje**

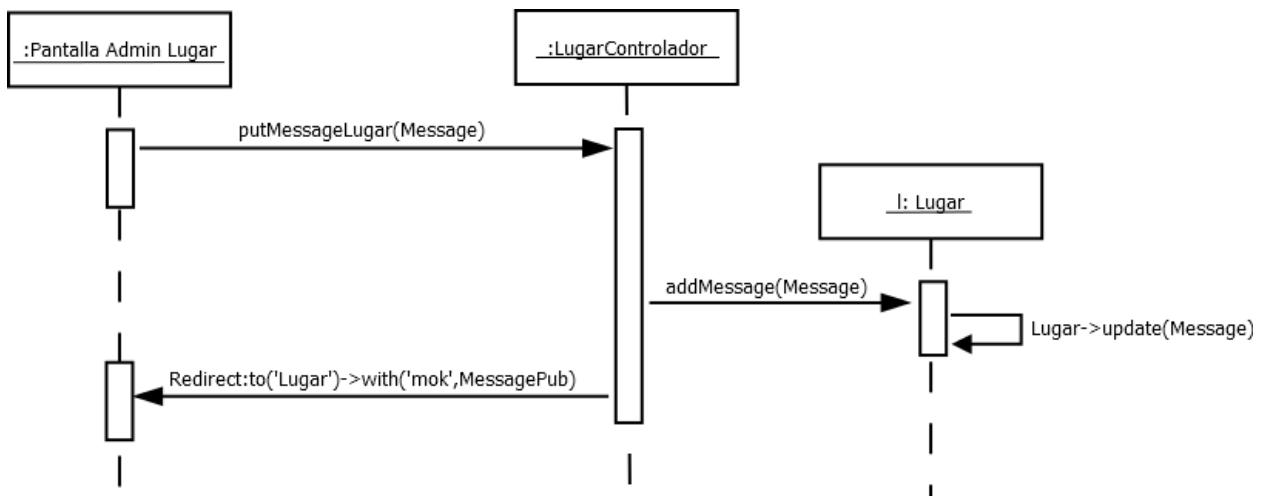


Figura 5.23 – Diagrama de secuencia – Caso de uso publicar mensaje

- **Caso de uso: Validar canciones**

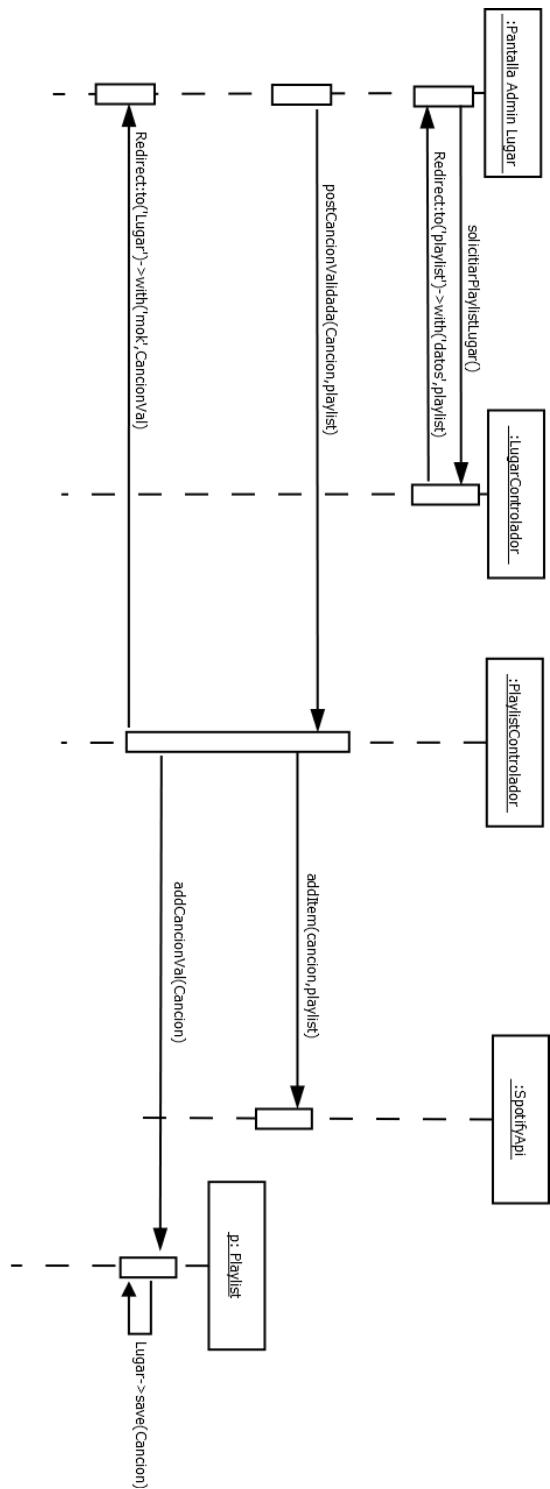


Figura 5.24 – Diagrama de secuencia – Caso de uso publicar mensaje

- **Caso de uso: Reproducir canciones**

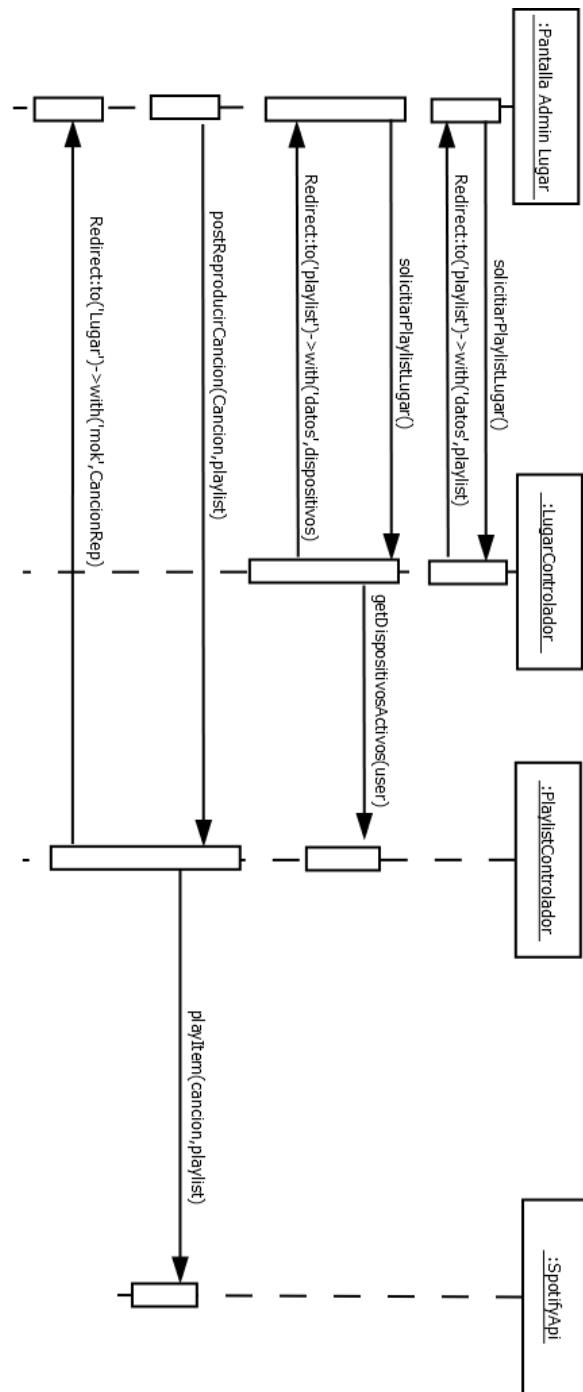


Figura 5.25 – Diagrama de secuencia – Caso de uso reproducir canciones

## 5.4. Diseño detallado de la Interfaz de Usuario

En esta sección se detallará la interfaz de usuario de la aplicación web, para ello se mostrará un ejemplo de la pantalla principal y el aspecto que tendrá con sus diferentes partes.



Figura 5.26 – Diseño de la pantalla principal jUCAbox

Se ha dividido la herramienta en 3 bloques diferentes. El bloque 1, muestra el menú principal de la aplicación. Este bloque es un menú lateral desarrollado de manera responsive, por lo que cuando se vea en una pantalla de menor tamaño se mostrará oculto. El bloque 2 muestra la información del usuario conectado y el bloque 3 muestra información sobre la página que estamos consultando.

En el capítulo del manual de usuario, se detallará con más profundidad todas las partes de la herramienta y su interacción con el resto de elementos.

La herramienta está pensada para ser utilizada desde cualquier dispositivo, por lo que está diseñada como un web Responsive. Utilizando bootstrap, para facilitar el diseño de la misma, y basándose en la plantilla OpenSource AdminLte. Se ha buscado una tonalidad neutra de colores, basándose en el gris para el menú y blanco para la parte central, para que la aplicación sea de aspecto visual atractivo pero sin utilizar colores llamativos que puedan causar fatiga visual.

# Capítulo 6

## Construcción del Sistema

Este capítulo trata sobre todos los aspectos relacionados con la implementación del sistema en código, haciendo uso de un determinado entorno tecnológico.

### 6.1. Entorno de Construcción

Para el desarrollo, como ya se ha comentado antes, se han utilizado herramientas de Mean Stack, el cual utiliza un desarrollo end-to-end basadas en JavaScript como lenguaje principal, tanto en el servidor como en el cliente. Las siglas MEAN significan:

- **MongoDB:** Sistema de base de datos no relacionales basada en JSON. Utiliza una versión propia llamada BSON.
- **Express:** Framework encargado de conectar MongoDB con el servidor WEB. Es el responsable de gestionar la API Rest. Capaz de manejar las solicitudes y peticiones Http.
- **Angular2:** Framework basado en MVC utilizado para la parte Cliente de la aplicación WEB. Se utiliza para el desarrollo de páginas SPA (Single Page Application). Se desarrolla con typescript, una versión libre de JavaScript creada por Microsoft.
- **Node.js:** Framework utilizado como servidor WEB. Se utiliza JavaScript para el desarrollo de las reglas del servidor.

En sus inicios MEAN comenzó con la primera versión de Angular, AngularJS, pero a raíz de la nueva versión de Angular publicada por Google, muchos programadores se decantaron por sustituirla por la nueva versión de Angular2. Son frameworks relativamente nuevos, pero con una gran comunidad detrás dando soporte.

Angular2 incorpora la inyección de dependencias con NPM, impulsando su rendimiento de forma exponencial.

Angular2 es el único framework del Mean Stack que no está basado en JavaScript. Se desarrolla en TypeScript, que al fin y al cabo, es un lenguaje que enmascara el JavaScript, siendo este su salida final. Su principal diferencia es el tipado de datos y clases. Fue creado por Microsoft, para obtener un código de JavaScript más limpio y ordenado. Cumple con la especificación ECMA6.

Al ser código JavaScript, existen infinidad de IDEs para su desarrollo, desde Visual Studio Code, un IDE gratuito de Microsoft, a los conocidos editores de código fuente SublimeText y Notepad++.

Tras tiempo sopesando las alternativas, descubrimos el nuevo producto del equipo de desarrollo de GitHub, el editor de código llamado Atom.io

Atom.io a través de su consola de comando, es bastante configurable. Pudiendo añadirle funcionales muy útiles, como la conexión directa con el repositorio GitHub, predictor de variables y funciones en tiempo real, y hasta Angular-Cli, una herramienta que facilita la

generación de código para Angular2. Finalmente nos decantamos por Atom, por su simplicidad y potencia.

Para el repositorio de información utilizamos GitHub, en su versión gratuita. Como herramientas de apoyo al desarrollo principal se encuentran, Mongoose, que es el conector para Node.js de MongoDB, el inyector de dependencias NPM, la librería de iconos FontAwesome y como plantilla de HTML5 y CSS3 AdminLTE 2, que es un dashboard open source.

Para el desarrollo de la memoria, se han utilizado herramientas como DIA para los diagramas, Balsamiq para el prototipado y Microsoft Word para la redacción de la misma.

## 6.2. Código Fuente

A continuación se describirá la estructura de los ficheros de desarrollo, así como la API Rest creada para el envío y recepción de peticiones con el servidor.

### 6.2.1. Estructura de ficheros

La estructura de los ficheros es la recomendada por el estándar de desarrollo de Mean Stack. iremos desde la estructura general, hasta profundizar en el resto. Centrándonos en las carpetas propias del desarrollo.

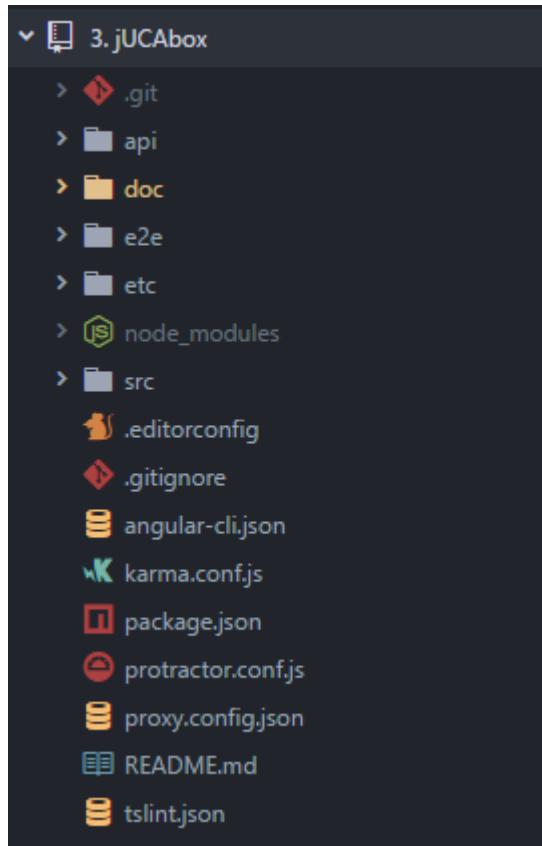


Figura 6.1 – Estructura de ficheros de jUCAblox

- **/.git**
  - Archivos de configuración del repositorio Github.
- **/api**
  - Carpeta de desarrollo de la Api Rest utilizada en el proyecto, es la parte del desarrollo basada en Node.js, Express y MongoDB. Su arquitectura es la siguiente:

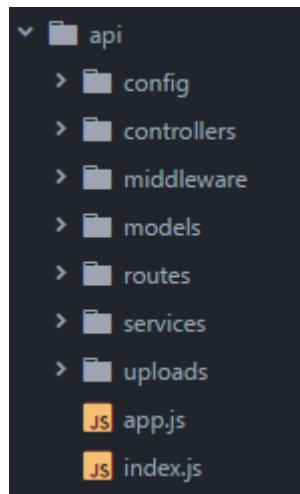


Figura 6.2 – Estructura de ficheros de la API de jUCAblox

- **/api/config**
  - Archivos generales de configuración.
- **/api/controllers**
  - Archivos donde se indica la lógica interna de las peticiones al servidor. Como ejemplo se mostrará una petición GET para solicitar la información de un lugar.

```

function getLugar(req,res){
    var lugarId = req.params.id;
    Lugar.findById(lugarId,(err,lugar)=>{
        if(err){
            res.status(500).send({message:'Error en la petición'});
        }else{
            if(!lugar){
                res.status(404).send({message:'no existe el lugar'});
            }else{
                res.status(200).send({lugar});
            }
        }
    })
}

```

Figura 6.3 – Petición GET Node.js

- **/api/middleware**

- Son los archivos utilizados como pasarela de validación. La herramienta está configurada, para que cualquier operación que comprometa los datos, sea necesario indicarle un token de autorización proporcionado por la misma. Para estos tokens se utiliza un método llamado JWT, que codifica la información del usuario, para comprobar que la petición es correcta.

- **/api/models**

- Representa los archivos donde se indican que campos va a contener cada entidad.

- **/api/routes**

- Contiene los ficheros de configuración de rutas. Siguiendo con el ejemplo anterior de obtener un lugar. Tendrías un fichero de rutas de lugares con la siguiente configuración. Donde se puede observar la instrucción api.get de getLugar.

```

var express = require('express');

//var spotifyToken = require('../spotify');
var LugarController = require('../controllers/lugares');

var api = express.Router();
var md_auth = require('../middleware/authenticate');
var multipart = require('connect-multiparty');
var resolve = __dirname

var md_upload = multipart({ uploadDir: resolve + '/../uploads/lugar' });

api.post('/addLugar',md_auth.ensureAuth,LugarController.saveLugar);
api.put('/updateLugar/:id',md_auth.ensureAuth,LugarController.updateLugar);
api.put('/updateLugarTM/:id',md_auth.ensureAuth,LugarController.updateTipoMusica);
api.get('/getLugar/:id',LugarController.getLugar);
api.get('/getLugar',LugarController.getLugares);
api.post('/upload-image-lugar/:id',[md_auth.ensureAuth,md_upload],LugarController.uploadImage);
api.post('/delete-image-lugar/:id',[md_auth.ensureAuth,md_upload],LugarController.deleteImageFile);
api.post('/delete-Allimage-lugar/:id',[md_auth.ensureAuth,md_upload],LugarController.deleteAllImageFile);
api.get('/get-image-lugar/:imageFile',LugarController.getImageFile);
api.delete('/deleteLugar/:id',md_auth.ensureAuth,LugarController.deleteLugar);
api.post('/getLugaresNombre',LugarController.getLugaresNombre);
api.get('/getTokenLugar/:id',md_auth.ensureAuth,LugarController.getTokenLugar);

module.exports = api;

```

Figura 6.4 – Rutas lugares Node.js

- **/api/services**
  - Contiene los ficheros para la generación del token JWT.
- **/api/uploads**
  - Carpeta donde se albergarán los ficheros que el usuario suba a la aplicación.
- **Apps.js e Index.js**
  - Son los ficheros principales de configuración de la Api.
- **/doc**
  - Todos los documentos de documentación de jUCAblox.
- **/e2e**
  - Ficheros internos de configuración del framework.
- **/etc**
  - Carpeta generada automáticamente por el framework, actualmente vacía.
- **/node\_modules**

- Carpeta donde se encuentran las dependencias inyectadas a través de NPM.
- **/src**
  - Archivos con el código del Front de la aplicación, sería la parte de Angular2.

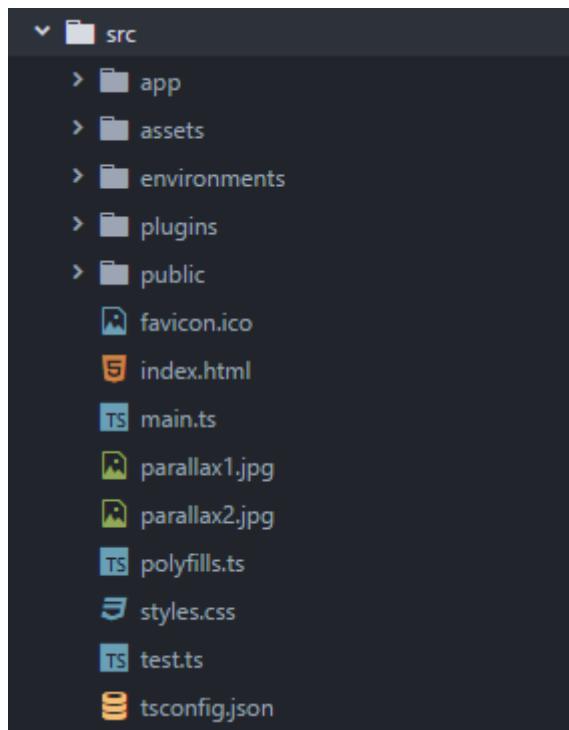


Figura 6.5 – Estructura de ficheros del Front de jUCAblox

- **/src/app**
  - Relación de carpetas donde se incluye el código fuente propio de la herramienta, como son los controladores, modelos y servicios.
- **/src/assets**
  - Carpeta que contiene todos los archivos relacionados con la parte visual de la herramienta, archivos css, js y fuentes entre otros.
- **/src/environments**
  - Archivos sobre la configuración del entorno. Si es en pruebas o si está en producción la herramienta.
- **/src/plugins**
  - Carpeta donde se incluirían plugins de terceros si se hicieran uso. En nuestro caso la carpeta está vacía.
- **/src/public**

- Ficheros públicos de la aplicación. Es la carpeta que utiliza fontawesome por defecto para dejar sus ficheros.
- /**src**/\*.\*
- El resto de ficheros son propios de Angular2, siendo el index.html el archivo central de la misma.
- /\*.\*
- El resto de los ficheros son propios del Mean Stack, donde se configuran las dependencias de manera automática a través de Angular-cli.

### 6.3. Scripts de Base de datos

Como ya hemos mencionado antes, la herramienta utiliza una base de datos NoSQL como es MongoDB. La configuración de base de datos se hace a través de los ficheros contenidos en la carpeta models de la API.

En ellos se describen el tipo de datos que va a contener la base de datos. En este caso, que información va a contener los documentos de las colecciones.

No existe un Script de creación de colecciones, puesto que cuando se inserta un documento en una colección que no existe, MongoDB interpreta ese documento y crea la colección automáticamente.

Como ejemplo, ya citado anteriormente, mostraremos uno de los ficheros modelo utilizado en la API como es el de usuario.

```

1  var UserSchema = Schema({
2    userID: String,
3    firstname: String,
4    lastname: String,
5    email: String,
6    avatarUrl: String,
7    creationDate: Date,
8    preferredLang: String,
9    city: String,
10   province: String,
11   country: String,
12   nickName: String,
13   LugaresFav: [{ type:Schema.ObjectId, ref:'Lugar'}]
14 });

```

Figura 6.6 – Modelo de Usuario



# Capítulo 7

## Pruebas del Sistema

En este capítulo se presenta el plan de pruebas del sistema de información, incluyendo los diferentes tipos de pruebas que se han llevado a cabo.

### 7.1. Estrategia

Para el testeo de la herramienta se ha seguido con un método de pruebas de manera manual. Pues la cantidad de funcionalidades de jUCABox no requería el desarrollo de un módulo específico para pruebas.

Al ser una herramienta orientada al mercado, no a algún cliente en particular, se ha realizado pruebas de aceptación con distintos perfiles potenciales de la herramienta, desde un usuario básico, a un administrador de un local de música.

Para probar la dependencia entre componentes, se realizaron pruebas de integración. Cuando se hacía alguna modificación en algún componente que interactuaba con otros, se comprobaban que cumplían con las necesidades propuestas. Cuando la herramienta ya estuvo lo suficientemente madura, las pruebas se realizaban en la capa visual de la herramienta. Era más sencillo realizar las pruebas conforme se iban desarrollando funcionalidades, que buscar los fallos según la dependencia de los componentes.

### 7.2. Entorno de Pruebas

El entorno utilizado para la realización de las pruebas, es el mismo que se ha utilizado para el desarrollo de la aplicación. Para la parte hardware un portátil Lenovo ideapad 100 con un procesador intel i5 con 8gb Ram y disco duro de 256 GB. El entorno software en el cual se hicieron las pruebas fue en el servidor local proporcionado por Node.js.

### 7.3. Roles

Para el testeo de la herramienta, fundamentalmente han interactuado dos roles. El propio desarrollador y usuario potencial de la herramienta. Al tener dos enfoques la herramienta, uno para la persona solicitante de canciones, y otro para el administrador de locales. Se ha contado con dos personas para este rol, cada una testeando las funcionalidades con las cuales se identificaba.

- **Desarrollador**

- Las pruebas realizadas por el desarrollador, están condicionadas a la propia naturaleza de la herramienta. Es inherente, probar algo para lo que se ha diseñado, con esto no quiere decir que las pruebas sean inútiles, sino todo lo contrario, el desarrollador conoce la funcionalidad de la herramienta pudiendo cubrir un amplio espectro de las posibles interacciones del usuario con la misma. Aun así es necesario realizar una batería de pruebas de un rol ajeno al desarrollo de la aplicación

- **Usuario**
  - Las pruebas por parte de los usuarios se enfocaban en las necesidades de cada uno. Al ser bastante sencillo el poder realizar una petición, el usuario no estaba formado en el uso de la herramienta y las pruebas concluyeron con éxito.
  - Para el administrador de locales, puesto que tiene una visión más amplia del negocio, a la vez que iba realizando el testeo de la herramienta, iba aportando nuevos requisitos que podrían incorporarse a la herramienta.

## 7.4. Niveles de Pruebas

En esta sección se documentan los diferentes tipos de pruebas que se han llevado a cabo.

### 7.4.1. Pruebas Unitarias

Para la realización de pruebas unitarias, como se ha comentado antes, se han realizado en la capa visual de la herramienta. Para cada apartado se comprobaba que la apariencia y respuesta de las acciones era la esperada.

Para la API Rest, se comprobaba si las peticiones hacia el servidor devolvían una respuesta satisfactoria. Para las pruebas de la API, se utilizó una herramienta gratuita llamada Postman, que gestiona las llamadas al servidor a través del protocolo http.

### 7.4.2. Pruebas de Integración

Una vez comprobado que la aplicación Web y la API respondían correctamente por separado, se procedió a la integración de las mismas. Haciendo llamadas al servidor desde la herramienta Web. Para el cual se utilizó la herramienta RoboMongo, una aplicación de escritorio para gestionar de manera visual la base de datos MongoDB. Pudiendo así comprobar que la respuesta de las peticiones realizadas desde la parte Web eran correctas y coherentes.

### 7.4.3. Pruebas de Sistema

En esta sección se realizan las pruebas de sistema de modo que se asegure que el sistema cumple con todos los requisitos establecidos.

#### Pruebas funcionales

Para realizar las pruebas funcionales, se han utilizado los requisitos descritos en la sección 3.4.1, lo cual fue necesario probar tanto los flujos normales como los flujos alternativos.

Se comenzó realizando el acceso de los usuarios al sistema, ya que fue la parte inicial del proyecto. Se comprobó que el acceso era correcto, y al conectarse a través de inicio de sesión de terceros, además de que los datos que se obtenían eran correctos. Una vez realizada la comprobación, se testeó el correcto funcionamiento de la protección de rutas, ya que cada perfil puede acceder a diferentes funcionalidades.

Una vez realizada estas pruebas, se procede a realizar búsquedas de otros usuarios, así como la interacción con los mismos. Se comprobó que los datos registrados, correspondían con los datos proporcionados por terceros.

Para el apartado de gestión de lugares, en primer lugar se comprobó la creación de los mismos. Si los datos introducidos se registraban correctamente. Una vez realizada la creación de lugares, se

procedía a la búsqueda del mismo, desde el módulo específico para ello.

Se realizó una búsqueda del lugar creado, y el usuario previamente logado, comprobaba si la inserción del lugar a la lista de favoritos era correcta. Una vez concluida con la prueba inicial de creación de lugares, se probó el resto de funciones del apartado. Como son la inserción de mensajes en el tablón y la creación de playlists para el mismo.

A raíz de este punto, se comprobó que la conexión con la API de Spotify era satisfactoria, detectando que los datos obtenidos, se correspondían con la información solicitada. Una vez comprobada que la conexión era correcta, se procedió a enviar una canción para el proceso de validación de la misma, y validar el flujo de información desde la herramienta a Spotify.

Para ello se procedió a comprobar la interacción de la herramienta con la API de Spotify. Enviando y validando las canciones introducidas previamente por el usuario.

Por último se validaron que las estadísticas proporcionadas por el reporte de QlikView eran correctas y que los datos mostrados eran reales.

## **Pruebas no funcionales**

A continuación se describen las pruebas funcionales realizadas en la aplicación Web.

- **Portabilidad**

- A través de diferentes navegadores se comprueba la correcta visualización de los elementos, así como desde diferentes dispositivos móviles. Ya que el desarrollo de la herramienta se ha basado en un desarrollo responsive, para que pueda ser visualizada desde cualquier dispositivo con conexión a internet. A través de google Chrome, se realizaron varias visualizaciones en distintos dispositivos.

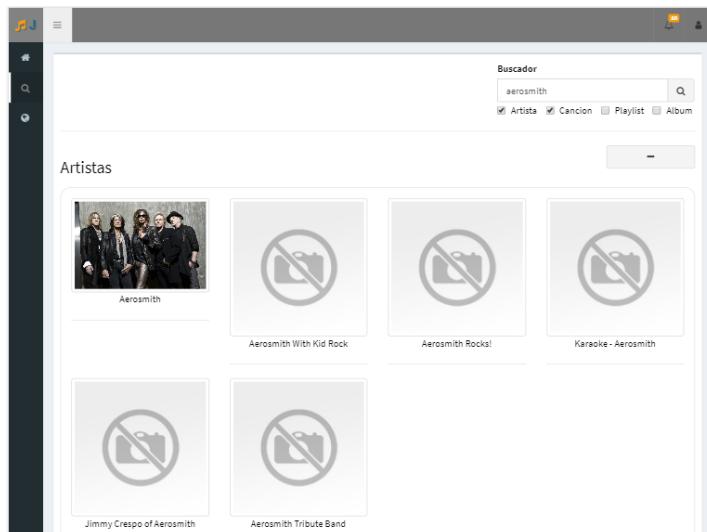


Figura 7.1 – jUCAbbox Ipad

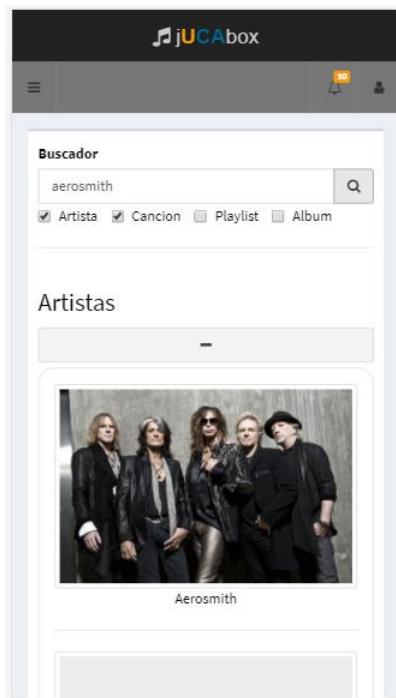


Figura 7.2 – jUCAbbox Iphone 6

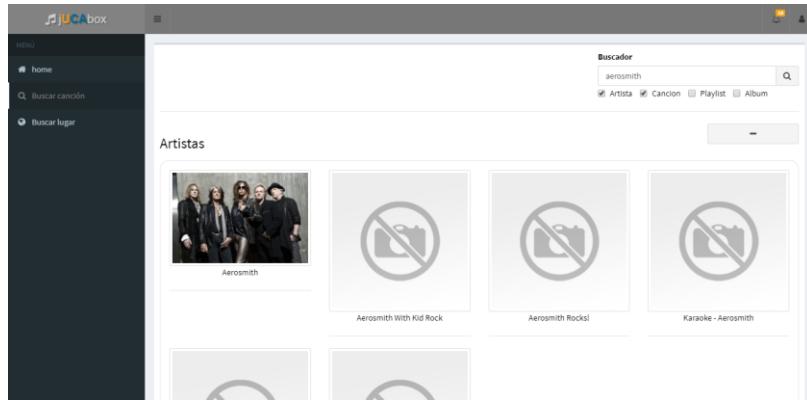


Figura 7.3 – Navegador Web

- **Mantenibilidad**

- Al utilizar herramientas novedosas, siempre se corre el riesgo de que dejen de ser mantenidas. Pero en este caso el fork MEAN Stack, tienen detrás a una gran comunidad de usuarios, incluyendo el propio desarrollador de Angular, Google, ampliando cada vez más la funcionalidad del mismo.
- Se han realizado pruebas con la herramienta SonarQube, para comprobar duplicidades de código y reglas de codificación. Aunque es de mencionar, que el lenguaje utilizado para desarrollar esta web, TypeScript, aún no es soportado 100% por la herramienta por lo que no se pueden obtener todos los estudios posibles.
- Aún así se ha pasado el test con éxito, sin duplicidades, ni vulnerabilidades en el código.

SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files... July 11, 2017 5:29 PM Version 1.0

jucabox Project

Issues Measures **Code** Activity Administration ▾

Search

		Lines of Code	Bugs	Vulnerabilities	Code Smells	Coverage	Duplications
⌚	⌚ jucabox Project	5.1k	0	0	0		0.0%
⌚	⌚ src	45	0	0	0		0.0%
⌚	⌚ src/app	244	0	0	0		0.0%
⌚	⌚ src/app/components/amigo-detalle	227	0	0	0		0.0%
⌚	⌚ src/app/components/artista	474	0	0	0		0.0%
⌚	⌚ src/app/components/callback-spotify	51	0	0	0		0.0%
⌚	⌚ src/app/components/crearlugar	204	0	0	0		0.0%
⌚	⌚ src/app/components/home	48	0	0	0		0.0%
⌚	⌚ src/app/components/luqar	622	0	0	0		0.0%
⌚	⌚ src/app/components/luqueres	129	0	0	0		0.0%
⌚	⌚ src/app/components/search	112	0	0	0		0.0%
⌚	⌚ src/app/components/shared/navbar	12	0	0	0		0.0%
⌚	⌚ src/app/components/usuario	325	0	0	0		0.0%
⌚	⌚ src/app/helpers	6	0	0	0		0.0%
⌚	⌚ src/app/interfaces	6	0	0	0		0.0%
⌚	⌚ src/app/models	88	0	0	0		0.0%
⌚	⌚ src/app/pages/client	0	0	0	0		0.0%
⌚	⌚ src/app/pages/home	53	0	0	0		0.0%
⌚	⌚ src/app/pages/layouts/auth	106	0	0	0		0.0%
⌚	⌚ src/app/pages/login	34	0	0	0		0.0%
⌚	⌚ src/app/pages/page-num	44	0	0	0		0.0%
⌚	⌚ src/app/pages/register	11	0	0	0		0.0%
⌚	⌚ src/app/pipes	125	0	0	0		0.0%
⌚	⌚ src/app/services	1.8k	0	0	0		0.0%
⌚	⌚ src/app/widgets/app-footer	55	0	0	0		0.0%
⌚	⌚ src/app/widgets/app-header	18	0	0	0		0.0%
⌚	⌚ src/app/widgets/breadcrumb	22	0	0	0		0.0%
⌚	⌚ src/app/widgets/control-sidebar	11	0	0	0		0.0%
⌚	⌚ src/app/widgets/menu-aside	72	0	0	0		0.0%
⌚	⌚ src/app/widgets/messages-box	26	0	0	0		0.0%
⌚	⌚ src/app/widgets/notification-box	17	0	0	0		0.0%
⌚	⌚ src/app/widgets/tasks-box	17	0	0	0		0.0%
⌚	⌚ src/app/widgets/user-box	21	0	0	0		0.0%
⌚	⌚ src/environments	15	0	0	0		0.0%

33 of 33 shown

Figura 7.4 – Test SonarQube líneas de código

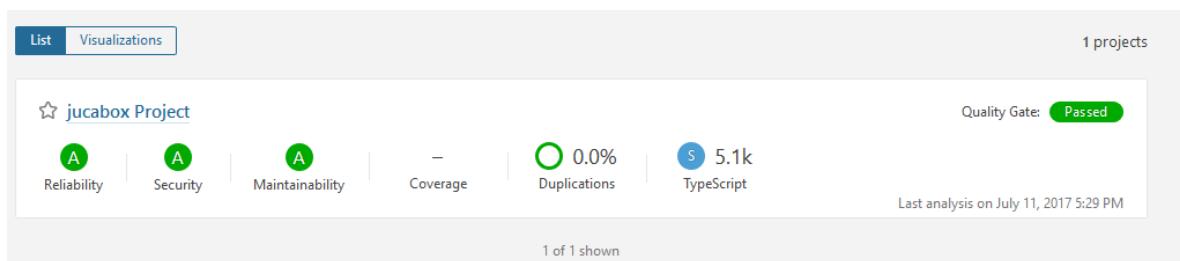


Figura 7.5 – Test Global SonarQube

- **Seguridad**

- Las pruebas realizadas para comprobar la fiabilidad de la herramienta han sido varias
  - Comprobar el acceso a sitios restringidos por el perfil a través de la url. La cual dio resultado satisfactorio, redirigiendo la petición a la página principal.
  - Realizando peticiones a la API Rest de jUCABox desde fuera de la herramienta, a través de la aplicación Postman. Resultado satisfactorio, ya que para poder realizar una petición, es necesario un token de autenticación interno de la herramienta.
  - Y la última prueba de seguridad, utilizando la herramienta SonarQube, fue satisfactoria, pudiéndose apreciar en la figura 7.5

- **Fiabilidad**

- Despues de realizar toda la batería de pruebas y usar sistemas de confianza, nos aseguran el correcto funcionamiento de la herramienta. Esto añadido a la confianza de elegir un servidor adecuado, como Amazon Web Service, garantiza que el sistema se encontrará en correcto funcionamiento.

- **Escalabilidad**

- Para garantizar la escalabilidad se ha utilizado un sistema de bases de datos MongoDB, la cual permite particionar de manera transparente para el usuario, aumentando así el rendimiento de acceso y procesado de los datos.

- **Rendimiento**

- Para garantizar el rendimiento de la aplicación se ha seguido con el estándar marcado por las herramientas utilizadas. Se ha utilizado un medidor de velocidad web llamado Performance-Analyser, la cual arroja los siguientes resultados:

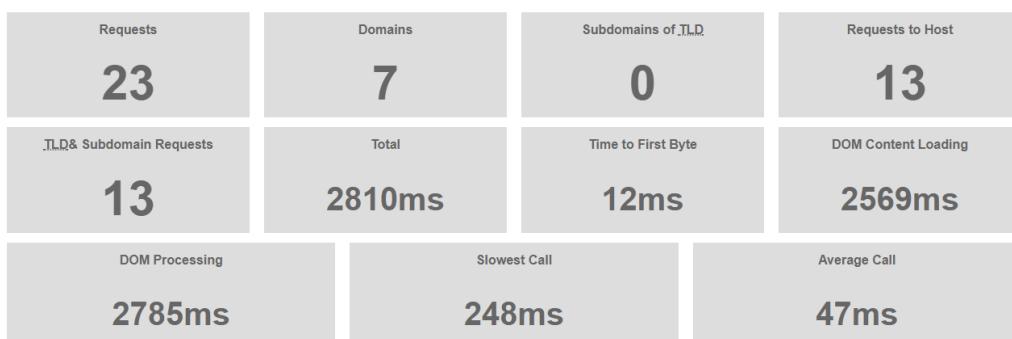


Figura 7.6 – Peticiones y Tiempos Performance-Analyser

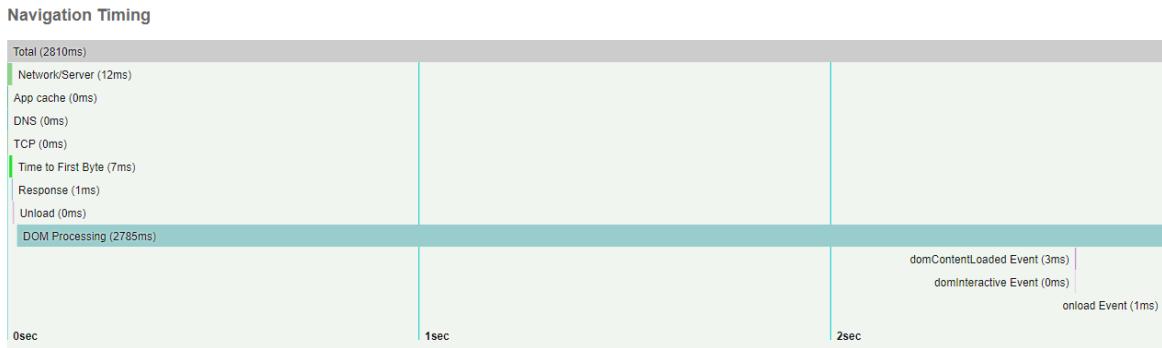


Figura 7.7 – Tiempo de navegación Performance-Analyser

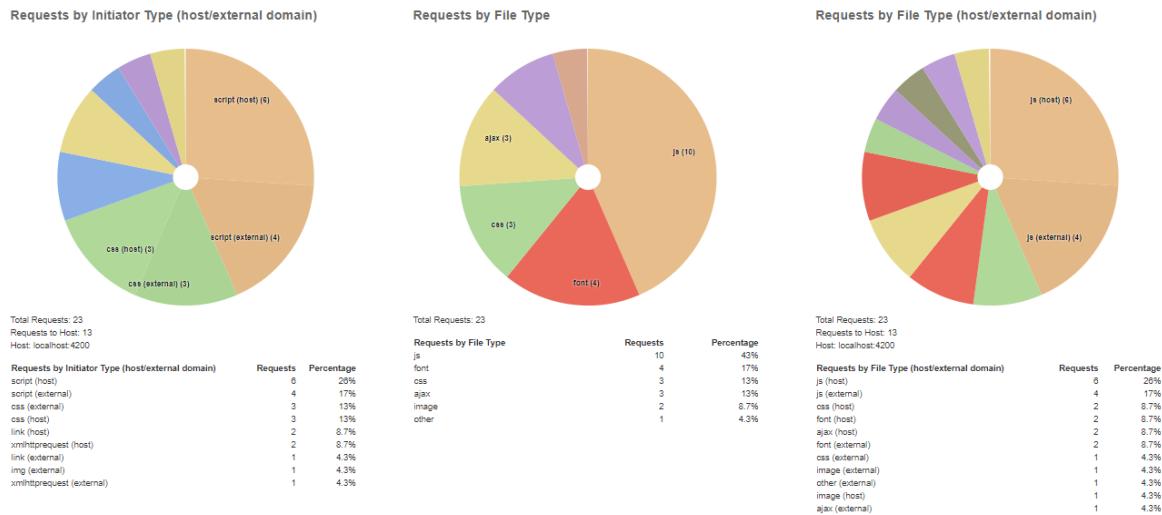


Figura 7.8 – Clasificación de peticiones Performance-Analyser

#### 7.4.4. Pruebas de Aceptación

Una vez realizadas todas las pruebas oportunas, se hicieron las pruebas finales de aceptación de la herramienta. Para estas pruebas son necesarias las personas integrantes en los roles mencionados anteriormente.

Básicamente, estas pruebas dejan el sistema a disposición del usuario, para que lo utilice el producto desarrollado. El feedback obtenido fue positivo, obteniendo también impresiones y críticas para futuras mejoras.

Las pruebas se realizaron sin supervisión del desarrollador, esperando ver reacciones y comportamientos de usuarios finales potenciales.

## **Parte III**

# **Epílogo**



# **Capítulo 8**

# **Manual de implantación y explotación**

Las instrucciones de implantación y explotación del sistema se detallan a continuación.

## **8.1. Introducción**

La herramienta se ha desarrollado como canal de comunicación de canciones entre clientes de un local y el encargado de la música. Consiguiendo trazabilidad de la información y mayor control del flujo de datos. Pudiendo realizarse estudios y análisis de las peticiones obtenidas. A su vez tiene como objetivo potenciar las capacidades de un lugar a nivel musical, es decir, poderse dar a conocer, así como enviar mensajes a sus clientes potenciales.

La otra visión para la que está ideada la aplicación, es para eventos sociales, donde el organizador puede crear listas de reproducción a petición de sus invitados.

## **8.2. Requisitos previos**

Para poder instalar la herramienta en un servidor es necesario tener instalado:

- Node.js siendo su versión igual o superior a 7.4.0
- El inyector de dependencias NPM, siendo su versión igual o superior a 4.2.0
- Angular-cli, con versión igual o superior a 1.0.0-beta 28.3
- Express, con versión igual o superior a 4.15.3
- Y una versión de MongoDB de 3.4.2 o superior
- Además de las dependencias ya incluidas en el package.json de la herramienta

## **8.3. Inventario de componentes**

No es necesario ningún componente específico hardware, para el despliegue de la aplicación se ha utilizada una instancia de AWS con 1 CPU Virtual y 1 GB de memoria. Para el lado del cliente, no existe ninguna limitación de hardware.

Los componentes software se le proporcionará al usuario un fichero con la relación de carpetas ya mencionada anteriormente, donde estarán todos los ficheros necesarios para su implantación. Además se incluirá el fichero package.json, con el cual se realiza la instalación de dependencias.

Por la parte de la base de datos, no es necesario ningún archivo con los Scripts, puesto que al iniciarse el proyecto, automáticamente se creará la base de datos y las tablas necesarias a petición de la aplicación. En el fichero, index.js, de la carpeta api, se indicará puerto y ruta de la base de datos.

## 8.4. Procedimientos de instalación

Para el despliegue de la aplicación hemos utilizado Amazon Web Service (AWS) y su servicio Amazon Elastic Compute Cloud ec2, que se basa en una colección de servidores remotos.

Suponiendo que ya tenemos creada una cuenta en AWS, nos dirigimos al apartado EC2

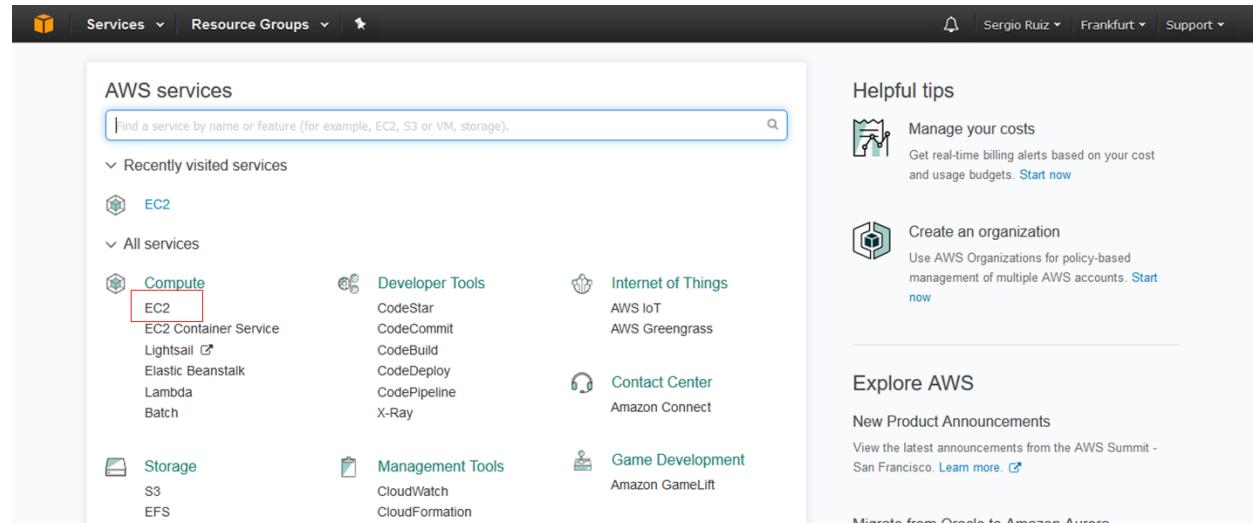


Figura 8.1 – Amazon Web Service Ec2

Antes de crear la instancia, es necesario crearnos un EC2 Key Pair, referidas a las claves públicas y privadas usadas para cifrar y descifrar la información para logarnos en el servidor.

Para ello, en el menú de la izquierda, en el apartado NETWORK & SECURITY, elegimos Key Pairs.

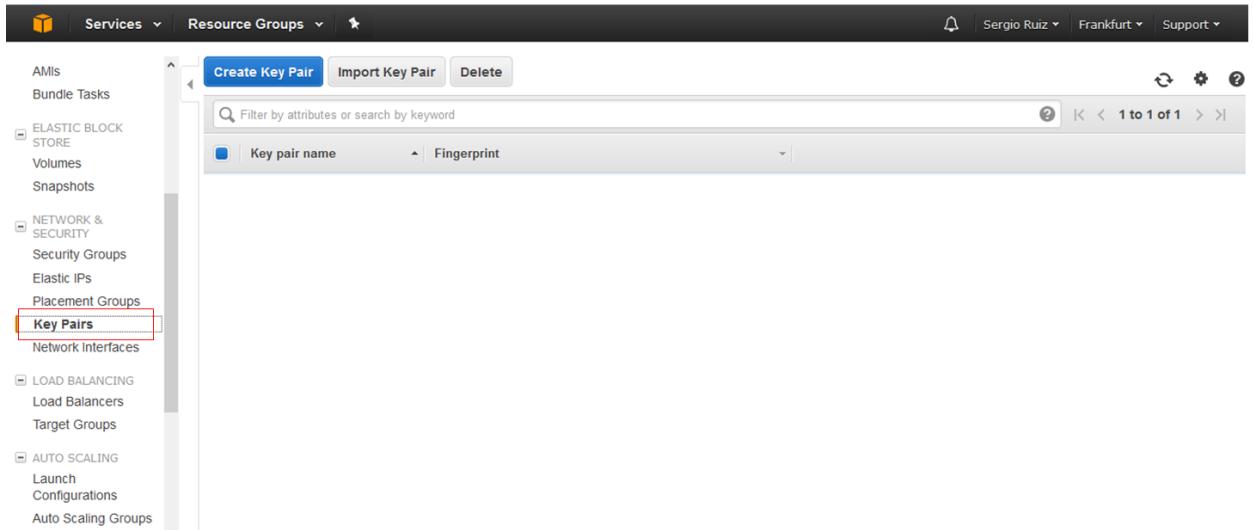


Figura 8.2 – Key Pairs AWS

Y hacemos click en Create Key Pair.

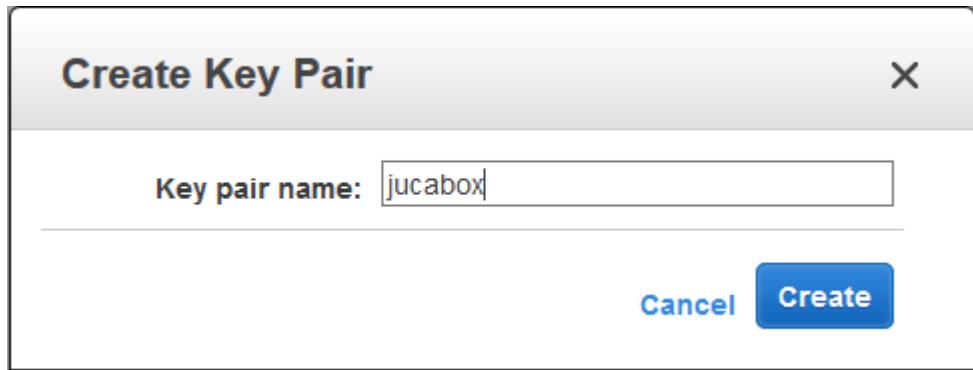


Figura 8.3 – Create Key Pairs AWS

Nos aparecerá la clave creada y se nos descargará un fichero llamada jucabox.pem , que será el fichero necesario para conectarnos a través de ssh al servidor.

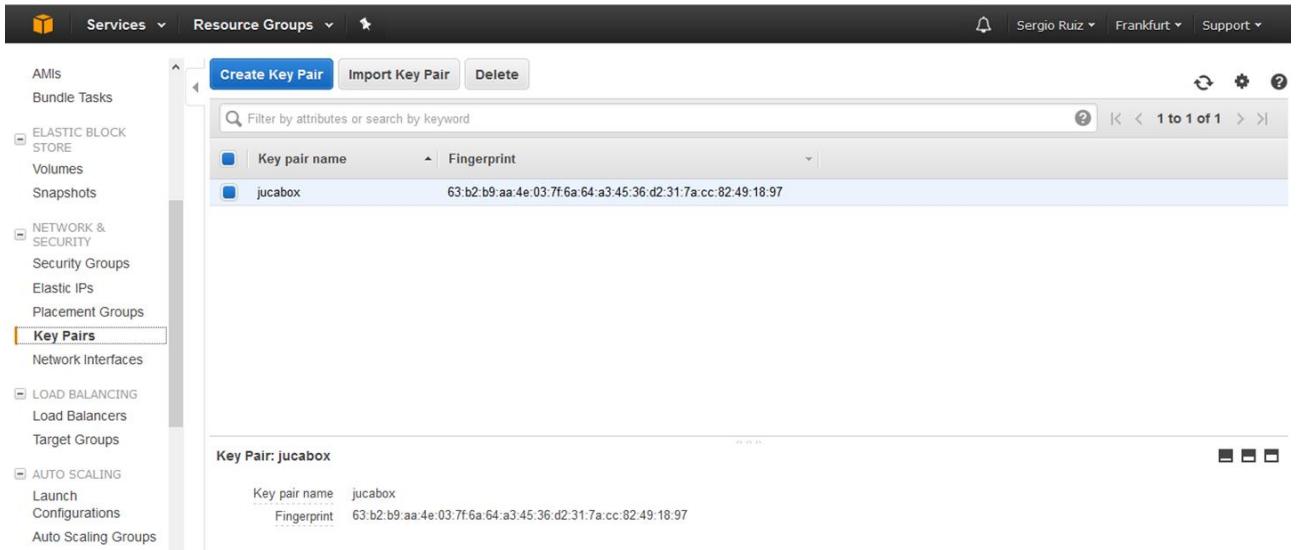


Figura 8.4 – Key Pairs Creada AWS

Una vez creado el par de claves, tendremos que dirigirnos al AWS Marketplace, el cual incluye imágenes preconfiguradas con herramientas. En nuestro caso vamos a buscar Bitnami MEAN image from BitRock, la cual incluye todas las herramientas necesarias para ejecutar aplicaciones basadas en MEAN Stack.

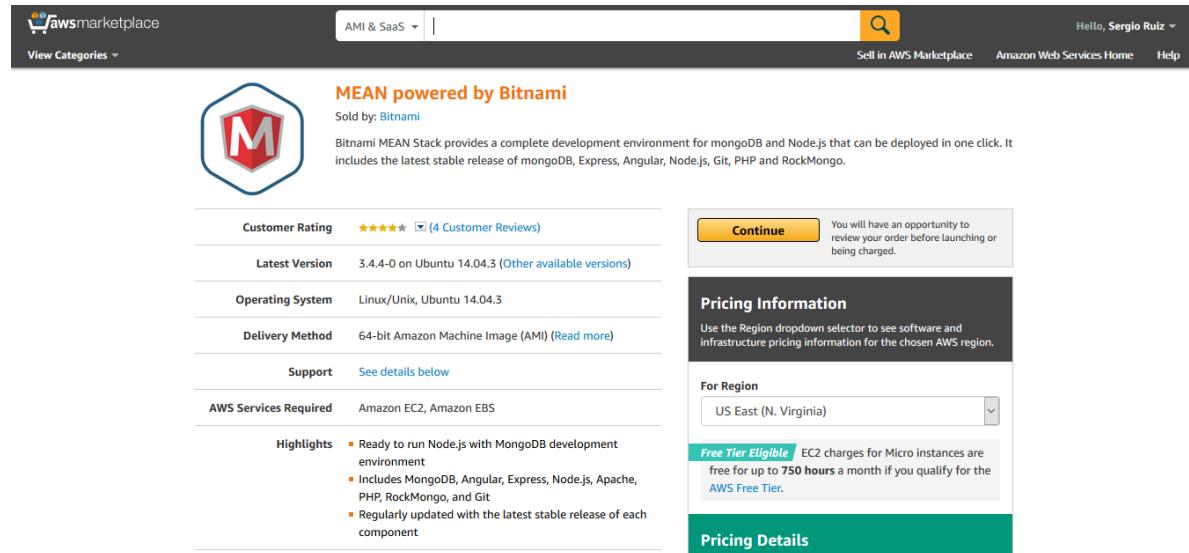


Figura 8.5 – Bitnami AWS Marketplace

Pulsamos Continue, y elegimos la instancia deseada, en este caso t2.micro y añadimos la Key Pair creada anteriormente.

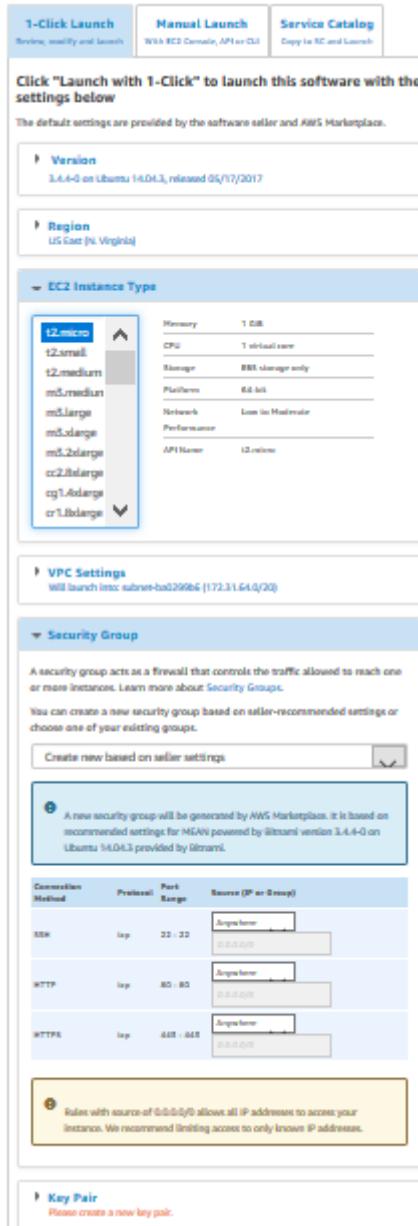


Figura 8.6 – Configuración Bitnami AWS Marketplace

Una vez instalado todo, volvemos al Dashboard de configuración de AWS, y veremos que tenemos activa una instancia.

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with links for Events, Tags, Reports, Limits, Instances (Instances, Spot Requests, Reserved Instances, Dedicated Hosts), Images (AMIs, Bundle Tasks), and Elastic Block Store (Volumes, Snapshots). The main area is titled "Resources" and displays the following summary for the EU Central (Frankfurt) region:

Category	Value
Running Instances	1
Dedicated Hosts	0
Volumes	1
Key Pairs	1
Placement Groups	0
Elastic IPs	0
Snapshots	0
Load Balancers	0
Security Groups	2

A callout box highlights a message: "Just need a simple virtual private server? Get everything you need to jumpstart your project - compute, storage, and networking – for a low, predictable price. Try Amazon Lightsail for free." Below this, a "Create Instance" button is visible.

Figura 8.7 – AWS Instancia

Accedemos a Running Instances, y deberemos esperar hasta que el Status Check, muestre 2/2 checks.

The screenshot shows the AWS EC2 Instances page. The sidebar includes links for Events, Tags, Reports, Limits, Instances (Instances, Spot Requests, Reserved Instances, Dedicated Hosts), Images (AMIs, Bundle Tasks), and Elastic Block Store (Volumes, Snapshots). The main table lists one instance:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4
i-000173710a0e05b9a	i-000173710a0e05b9a	t2.micro	eu-central-1a	running	2/2 checks ...	None	ec2-35-157-8-132.eu-central-1.compute.amazonaws.com	35.1

The instance details page for "i-000173710a0e05b9a" shows the following information:

Description		Status Checks		Monitoring		Tags		Usage Instructions	
Instance ID	i-000173710a0e05b9a	Public DNS (IPv4)	ec2-35-157-8-132.eu-central-1.compute.amazonaws.com	IPv4 Public IP	35.157.8.132	IPv6 Public IP	-	Private DNS	ip-172-31-26-220.eu-central-1.compute.internal
Instance state	running	Private DNS	172.31.26.220	Elastic IPs		Private IPs		Scheduled events	No scheduled events
Instance type	t2.micro	Secondary private IPs		AMI ID	bitnami-meanstack-3.4-0-linux-ubuntu-	VPC ID	vpc-23512d4b	Subnet ID	subnet-f38d889h
Availability zone	eu-central-1a								
Security groups	MEAN powered by Bitnami-3.4-4.0 on Ubuntu 14.04.3-AutogenByAWSMP-, view inbound rules								

Figura 8.8 – AWS Instancia Running

Una vez levantada la instancia, con el botón derecho encima de la instancia, seleccionamos Get System Log.

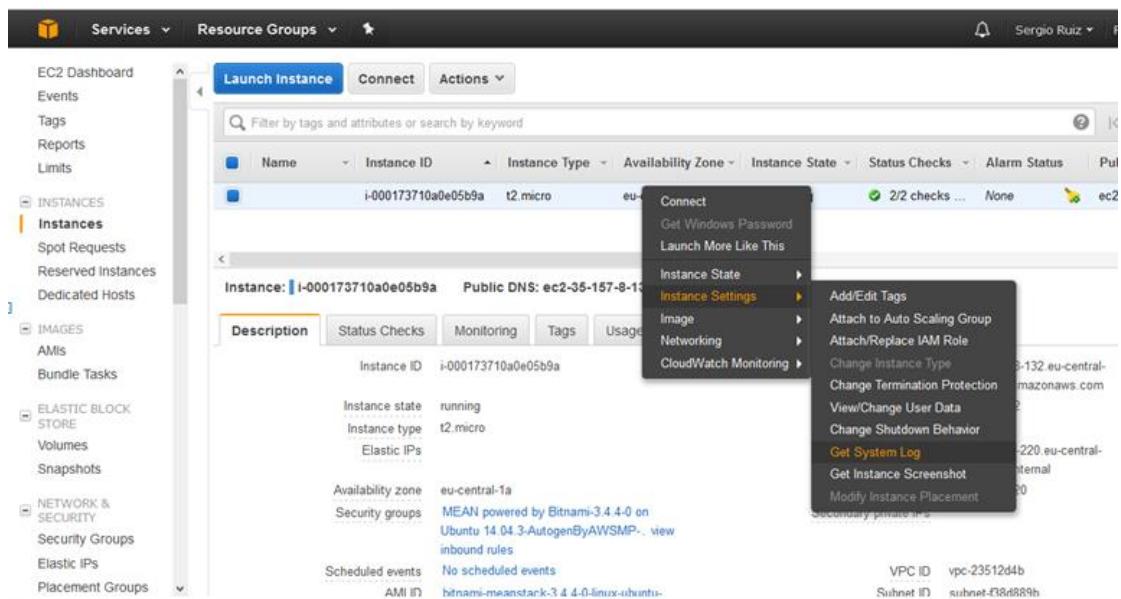


Figura 8.9 – AWS Instancia Get System Log

Y buscamos **Setting Bitnami application password to** y guardamos la contraseña.

The screenshot shows the 'System Log' window for instance i-000173710a0e05b9a. The log output is as follows:

```

en_US.UTF-8... * Starting OpenSSH server[74G[ OK ]
up-to-date
Generation complete.
open-vm-tools: not starting as this is not a VMware VM
* Restoring resolver state... [80G [74G[ OK ]
resize2fs 1.42.9 (4-Feb-2014)
The filesystem is already 2618595 blocks long. Nothing to do!

* Stopping Handle applying cloud-config[74G[ OK ]
650000+0 records in
650000+0 records out
665600000 bytes (666 MB) copied, 12.0076 s, 55.4 MB/s
Setting up swapspace version 1, size = 649996 KiB
no label, UUID=1957ec4d-5d4d-4d9e-9a36-05effc427c6e
micro

#####
#      Setting Bitnami application password to 'uOSMGpDdXPf4'
#
#####

[Thu Jul  6 10:03:47 UTC 2017] Regenerating keys for *
/opt/bitnami/var/init/pre-start/120_regenerate_keys: 27: /opt/bitnami/var/init/pre-start
[Thu Jul  6 10:03:47 UTC 2017] Regenerating keys for * finished
[Thu Jul  6 10:03:47 UTC 2017] Finished regenerating keys
[Thu Jul  6 10:03:52 UTC 2017] Setting up password for mongodb service
[Thu Jul  6 10:04:27 UTC 2017] Setting up password for mongodb service finished
[Thu Jul  6 10:04:32 UTC 2017] Setting up password for * application
/opt/bitnami/var/init/pre-start/130_configure_default_passwords: 50: /opt/bitnami/var/i
[Thu Jul  6 10:04:32 UTC 2017] Setting up password for * application finished
[Thu Jul  6 10:04:32 UTC 2017] Finished setting password
root@bitnami:/opt/bitnami/init#1 cd /var/mongodb/ laid 2251 already running

```

A 'Close' button is visible at the bottom right of the window.

Figura 8.10 – AWS Log Bitnami

Ahora necesitamos subir y configurar nuestra aplicación en el servidor AWS. Por lo que nos descargamos Putty, una herramienta gratuita para conectarnos a través de SSH. A su vez nos descargamos también, Putty Key Generator para transformar el fichero PEM previamente descargado a un fichero compatible con Putty.

Para ello abrimos el fichero jUCAbbox.pem con Putty Key Generator y seleccionamos Save private key.

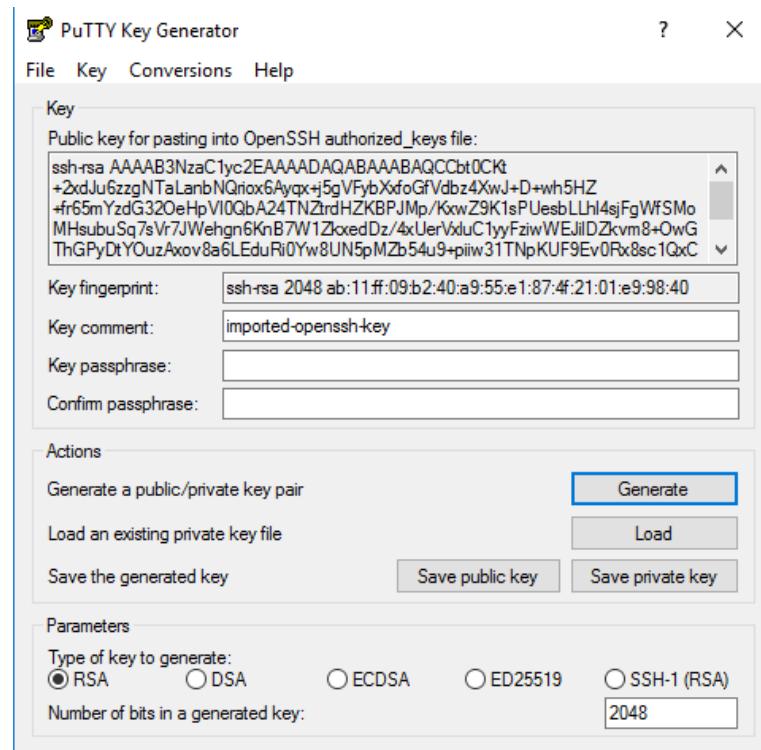


Figura 8.11 – Putty Key Generator

Se nos generará un fichero compatible con Putty. Con lo que abrimos Putty e introducimos los datos necesarios para la configuración. Obtenemos la dns de AWS.

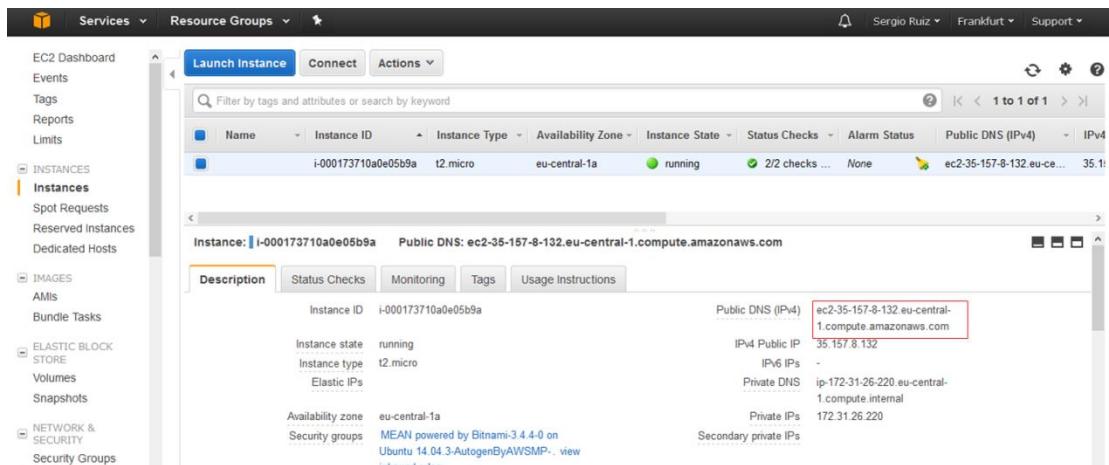


Figura 8.12 – AWS Dns

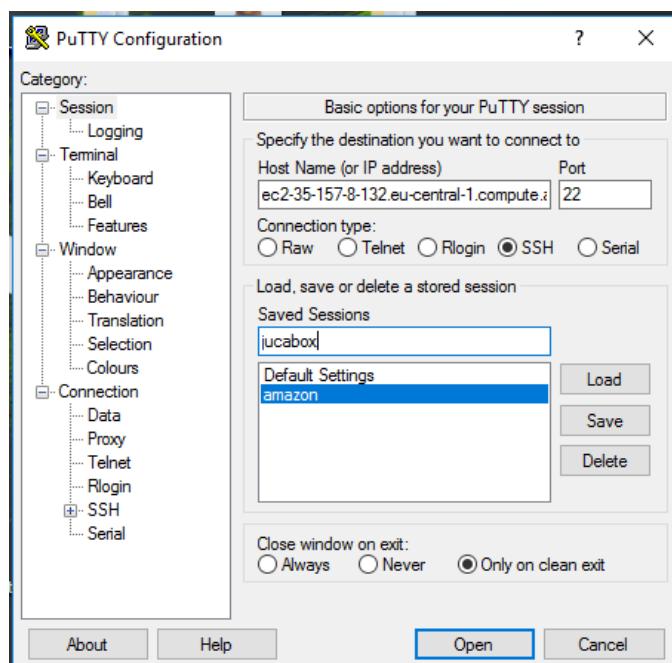


Figura 8.13 – Putty Session

Y añadimos en el apartado Connection → SSH →Auth , el fichero generado por Putty Gen.

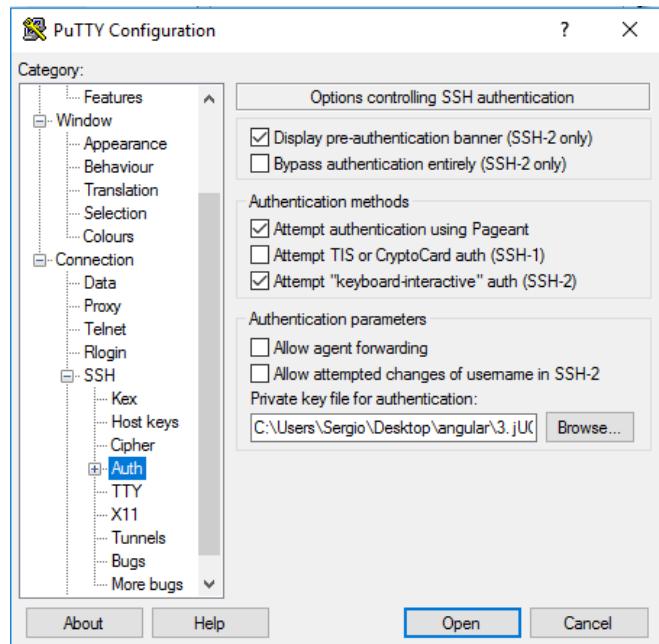


Figura 8.14 – Putty Session SSH Auth

Y nos logamos como bitnami.

```
bitnami@ip-172-31-26-220: ~
login as: bitnami
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-117-generic x86_64)

           _.-` \   -. 
          / \  \  \  \  \  \  \  \  \  \
*** Welcome to the Bitnami MEAN 3.4.4-0 ***
*** Documentation:  https://docs.bitnami.com/aws/infrastructure/mean/ ***
***                      https://docs.bitnami.com/aws/ ***
*** Bitnami Forums: https://community.bitnami.com/ ***
Last login: Thu Jul  6 13:17:09 2017 from 203.red-79-155-132.dynamicip.rima-tde.net
bitnami@ip-172-31-26-220:~$
```

A screenshot of a PuTTY terminal window. The title bar says 'bitnami@ip-172-31-26-220: ~'. The window displays a terminal session where the user has logged in as 'bitnami'. The system is identified as 'Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-117-generic x86\_64)'. It then displays a welcome message for 'Bitnami MEAN 3.4.4-0' with links to documentation and forums. At the bottom, it shows the last login information and the prompt 'bitnami@ip-172-31-26-220:~\$'. The background of the terminal window is black with white text.

Figura 8.15 – Putty Inicio Sesion

Una vez dentro tendremos varias carpetas, entre ellas apps, así que accedemos a ella y clonamos nuestro repositorio GIT. En este caso los comandos serían:

```

cd
cd apps
sudo git clone https://github.com/kodorniz/jUCAbbox_PFC.git
cd jUCAbbox_PFC

```

Una vez terminado de clonar el repositorio

```

bitnami@ip-172-31-26-220:~/apps$ sudo git clone
https://github.com/kodorniz/jUCAbbox_PFC.git
Cloning into 'jUCAbbox_PFC'...
remote: Counting objects: 1458, done.
remote: Compressing objects: 100% (374/374), done.
remote: Total 1458 (delta 250), reused 365 (delta 131), pack-reused 948
Receiving objects: 100% (1458/1458), 11.98 MiB | 5.84 MiB/s, done.
Resolving deltas: 100% (791/791), done.
Checking connectivity... done.
bitnami@ip-172-31-26-220:~/apps$

```

Una vez terminado de clonar el repositorio, realizamos la opción **npm install** para instalar las dependencias del proyecto.

Una vez finalizado, deberemos de configurar los puertos que hayamos puesto para la aplicación, en este caso para Angular el puerto es el 4200, para el API es el 3000 y para MongoDB es el 27017

Por lo que accedemos al Dashboard de AWS, en el apartado de NETWORK & SECURITY, accedemos a Security Groups.

En el grupo creado por MEAN powered by bitnami, editamos los puertos habilitados.

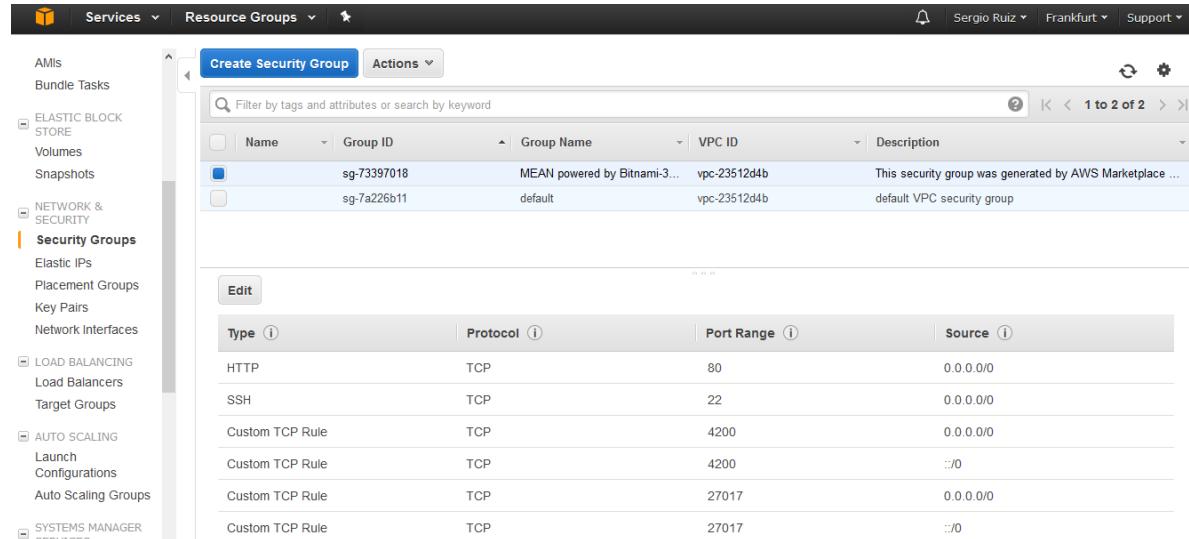


Figura 8.16 – Editar puertos AWS

Una vez abierto los puertos, probamos que nuestra aplicación está corriendo correctamente. Para ello ejecutamos el comando **npm start**.

Y obtendríamos el siguiente resultado si todo ha ido ok:

```

** NG Live Development Server is running **
10% building modules 4/4 modules o active[HPM] Proxy created: /api -> port:3000
[HPM] Subscribed to http-proxy events: [ 'error', 'close' ]

```

```

11% building modules 14/19 modules 5 active
...\\bootstrap\\dist\\css\\bootstrap.min.csswebpack: wait until bundle finished: /
Hash: a89f75a53bd541960297
Time: 31565ms
chunk {0} polyfills.bundle.js, polyfills.bundle.map (polyfills) 232 kB {5} [initial] [rendered]
chunk {1} main.bundle.js, main.bundle.map (main) 405 kB {4} [initial] [rendered]
chunk {2} styles.bundle.js, styles.bundle.map (styles) 401 kB {5} [initial] [rendered]
chunk {3} scripts.bundle.js, scripts.bundle.map (scripts) 994 kB {5} [initial] [rendered]
chunk {4} vendor.bundle.js, vendor.bundle.map (vendor) 5.44 MB [initial] [rendered]
chunk {5} inline.bundle.js, inline.bundle.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.

```

Ya tendríamos nuestra aplicación funcionando correctamente.

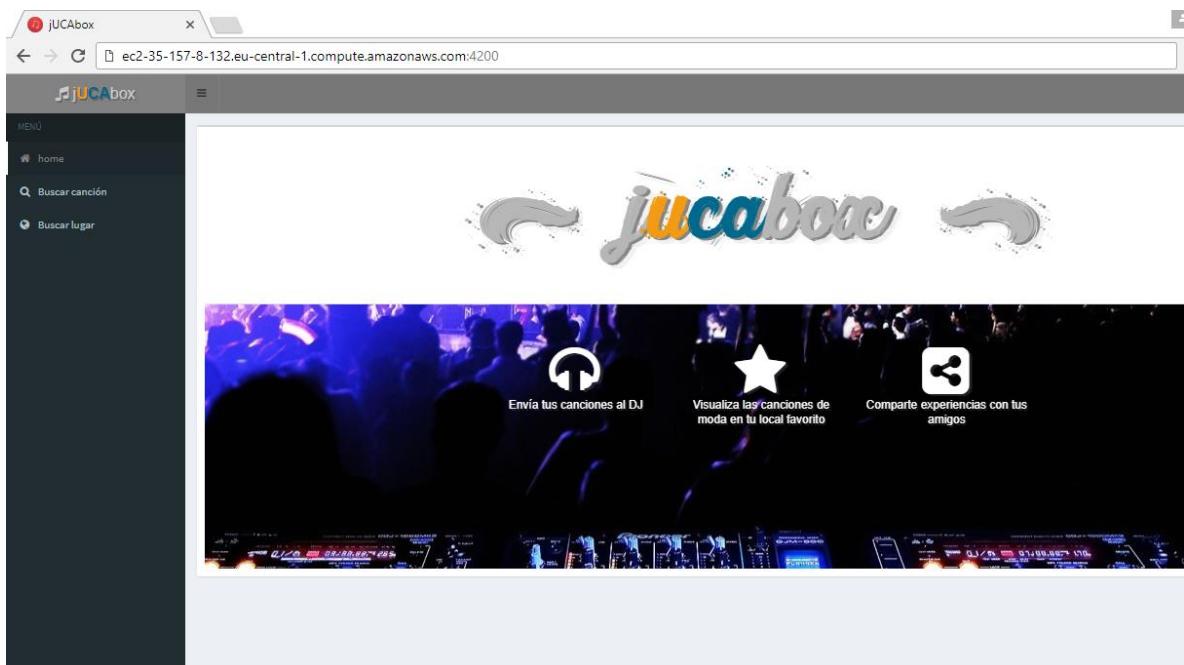


Figura 8.17 – jUCAbbox en AWS

## 8.5. Pruebas de implantación

Para comprobar el correcto funcionamiento simplemente es necesario acceder a la dirección creada, y verificar desde el Dashboard de AWS que todo funciona correctamente. Si existiera algún problema, AWS tiene reportes y controles de seguridad donde nos informan del estado de la aplicación en todo momento.

## 8.6. Procedimientos de operación y nivel de servicio

Para garantizar y asegurar los datos del servidor es necesario realizar copias periódicas de seguridad, para ello AWS nos ofrece un amplio abanico para la configuración de estas copias. Aun así, existen herramientas para poder hacer copias de seguridad de la base de datos de manera manual, como es la herramienta mongodump, que podemos instalar en el servidor y ejecutar el siguiente comando:

```
mongodump -h localhost --directoryperdb -o alldump
```

Para realizar una copia de base de datos consistente, sería necesario para la instancia, lo

que tendríamos que buscar un rango de tiempo adecuado para no intervenir en altos picos de uso de la herramienta.



# **Capítulo 9**

## **Manual de usuario**

### **9.1. Introducción**

Este capítulo describe el uso de la herramienta desarrollada. Uno de los principales objetivos que se quería cumplir, fue el de emular un jukebox, una máquina que permite elegir al usuario que canción va a sonar en el local.

Además se buscaba que el sistema fuera sencillo, intuitivo y ágil para el usuario.

### **9.2. Características**

La herramienta tiene como principales características:

- La petición, desde cualquier dispositivo con conexión a internet, de una canción en un local al encargado de la música.
- La gestión de locales, pudiendo enviar mensajes a los distintos usuarios que sean afines al local.
- La posibilidad de chatear entre los distintos usuarios de la herramienta.
- Tener trazabilidad de la información obtenida.

### **9.3. Requisitos previos**

Los requisitos previos para poder visualizar y utilizar la herramienta, es disponer de un dispositivo con conexión a internet y un navegador web.

## 9.4. Uso del Sistema

A continuación se van a describir las funcionalidades del sistema, así como su uso. Se detallarán los pasos necesarios que el usuario debe seguir para poder realizar las funciones principales de la herramienta.

### 9.4.1. Aspecto visual

En la siguiente imagen podemos observar el aspecto de la herramienta. Un menú en la parte izquierda con las funcionalidades principales de la herramienta. Otro menú superior con la información referente al usuario. Y una parte central, la cual va a ser cambiante, dependiendo de la funcionalidad que se esté utilizando en ese momento.

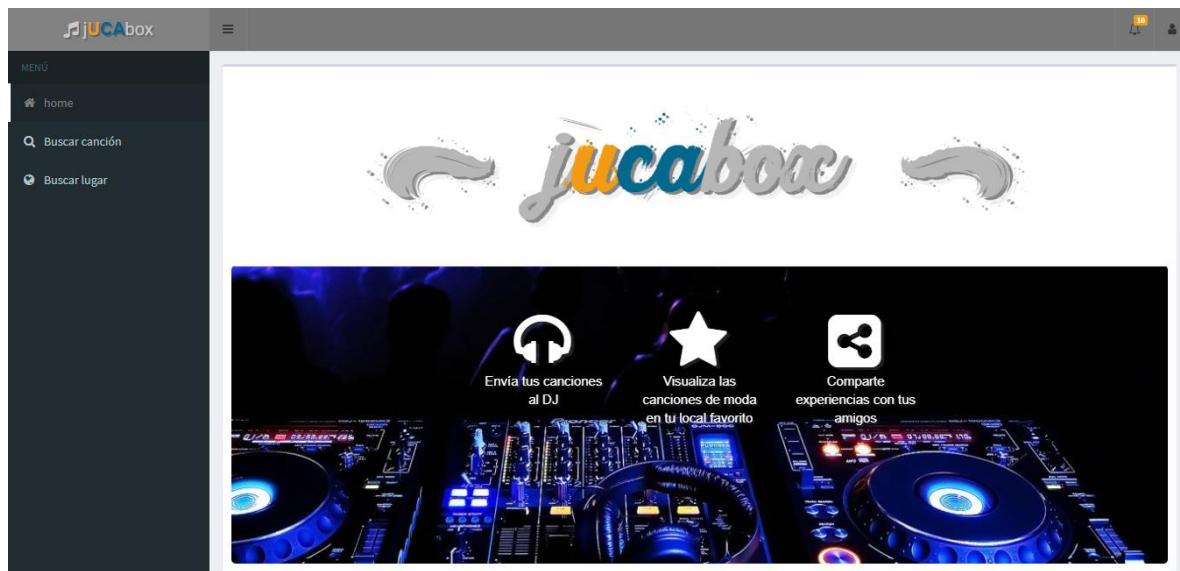


Figura 9.1 – Aspecto visual jUCAbbox

### 9.4.2. Búsqueda de canciones y envío de peticiones

Para poder realizar una búsqueda de alguna canción, no es necesario estar logado en la herramienta. Accedemos al apartado del menú de la izquierda *Buscar canción*.

En el cuadro de texto podemos escribir el nombre del artista, nombre de la canción, del álbum o de la playlist, dependiendo que tengamos marcado.

En este caso hemos buscado un artista, pero además como tenemos marcado el check de canciones también nos mostrará sus canciones relacionadas.

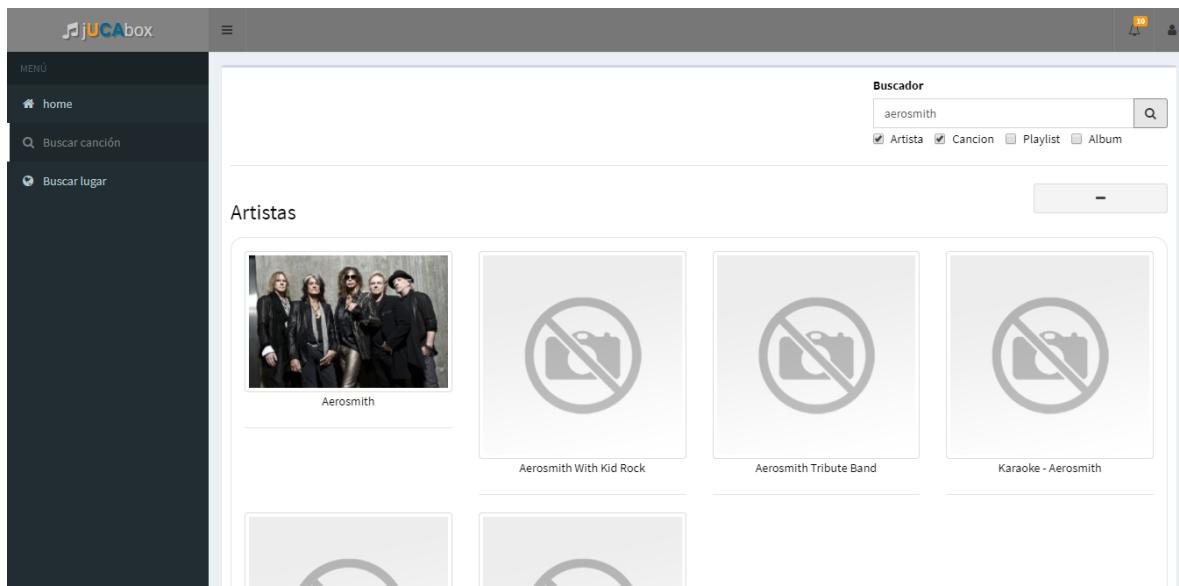


Figura 9.2 – Búsqueda de canción jUCAbbox

Haciendo click en el artista, podremos ver la ficha completa con el top de canciones asociadas al mismo.

This screenshot shows the detailed artist profile for 'Aerosmith'. It features a large circular photo of the band. The artist's name 'Aerosmith' is prominently displayed in bold text. Below it, the genres listed are 'album rock, alternative rock, classic rock, hard rock, mellow gold, rock'. The number of followers is shown as 'Seguidores: 1905258'. There are two buttons: 'Ir a spotify' and 'Buscar otros artistas'. To the right is a search bar for songs ('Buscar canciones...'). The main content area is titled 'Top Canciones' and lists four songs with their preview and share icons:

#	Canción	Album	Preview	Enviar
1	Dream On	Aerosmith		
2	I Don't Want to Miss a Thing - From the Touchstone film, "Armageddon"	I Don't Want To Miss A Thing		
3	Sweet Emotion	Toys In The Attic		
4	Walk This Way	Toys In The Attic		

Figura 9.3 – Ficha Artista jUCAbbox

Volviendo al apartado anterior de búsqueda de canciones. Podemos observar que tienen dos iconos.



Para escuchar un preview de la canción de 30 segundos.



Para realizar una petición al local que seleccionemos.

The screenshot shows the jUCAblox application interface. On the left is a dark sidebar with a menu: 'MENÚ', 'home', 'Buscar canción', and 'Buscar lugar'. The main area has a header 'Buscador' with a search input containing 'aerosmith' and checkboxes for 'Artista' (checked), 'Canción' (checked), 'Playlist' (unchecked), and 'Album' (unchecked). Below the header are two sections: 'Artistas' and 'Canciones'. The 'Canciones' section contains a table with the following data:

#	Preview	Enviar	Canción	Album	Artista
1			Dream On	Aerosmith	Aerosmith
2			Sweet Emotion	Toys In The Attic	Aerosmith
3			I Don't Want to Miss a Thing	Armageddon - The Album	Various Artists
4			Dream On	The Essential Aerosmith	Aerosmith
5			Walk This Way	Aerosmith's Greatest Hits	Aerosmith

Figura 9.4 – Búsqueda de canción 2 jUCAblox

Al pulsar en el ícono de enviar se abrirá una ventana donde podremos elegir el local o lugar al cual queremos hacer la petición.

Existen dos tipos de locales. Con token de validación y sin token de validación. El token de validación es un código que define el administrador del lugar para poder mostrar en vivo y no tener peticiones de canciones de usuarios que no estén presentes.

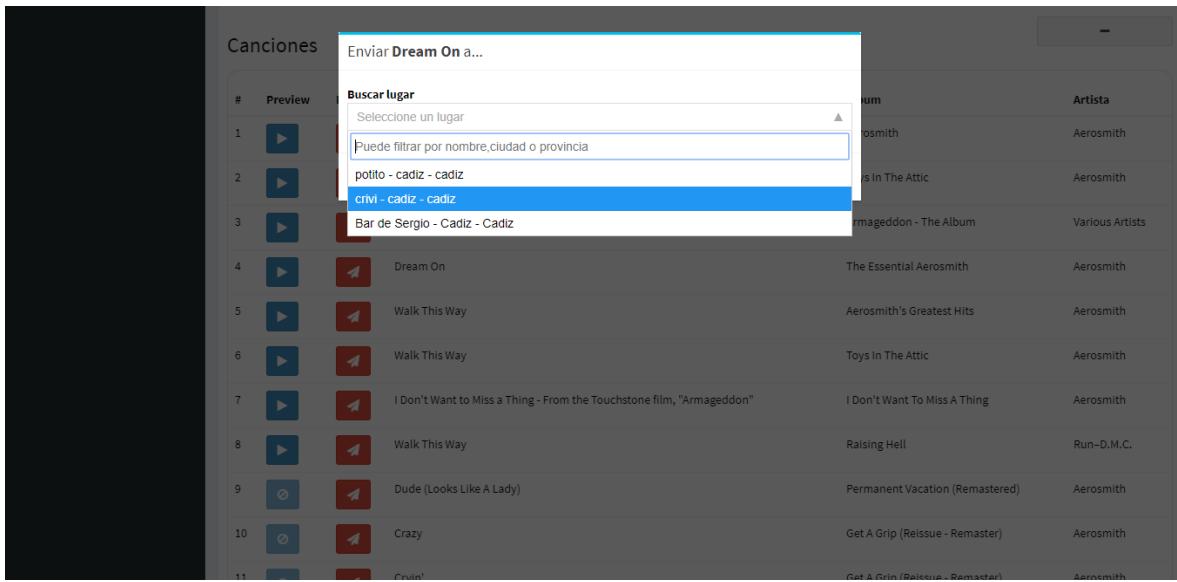


Figura 9.5 – Enviar canción local jUCAbbox

En primer lugar se deberá escoger el local al cual vamos a realizar la petición. Si ese lugar necesita token, mostrará el campo, si no lo necesita no lo mostrará. Dejará hacer la petición si el token introducido es válido.

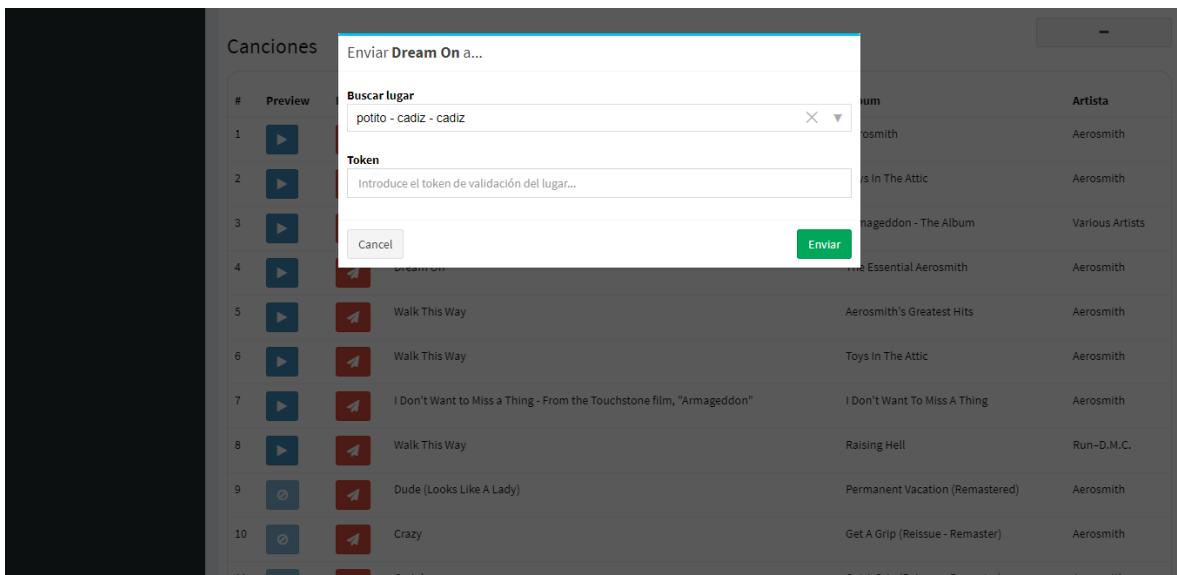


Figura 9.6 – Enviar canción local jUCAbbox

Si la canción se ha enviado correctamente aparecerá un mensaje abajo a la izquierda.

The screenshot shows a list of songs in a table format. The columns are labeled '#', 'Preview', 'Enviar', 'Canción', 'Album', and 'Artista'. The songs listed are:

#	Preview	Enviar	Canción	Album	Artista
1			Dream On	Aerosmith	Aerosmith
2			Sweet Emotion	Toys In The Attic	Aerosmith
3			I Don't Want to Miss a Thing	Armageddon - The Album	Various Artists
4			Dream On	The Essential Aerosmith	Aerosmith
5			Walk This Way	Aerosmith's Greatest Hits	Aerosmith
6			Walk This Way	Toys In The Attic	Aerosmith
7			I Don't Want to Miss a Thing - From the Touchstone film, "Armageddon"	I Don't Want To Miss A Thing	Aerosmith
8			Walk This Way	Raising Hell	Run-D.M.C.
9			Dude (Looks Like A Lady)	Permanent Vacation (Remastered)	Aerosmith
10			Crazy	Get A Grip (Reissue - Remaster)	Aerosmith
11			Cryin'	Get A Grip (Reissue - Remaster)	Aerosmith

A green bar at the bottom left displays the song 'Dream On' and the message 'Enviada a validación del local' with a checkmark icon.

Figura 9.7 – Confirmación de petición de canción jUCAbbox

#### 9.4.3. Búsqueda de lugares

Para la búsqueda de lugares, no es necesario estar registrado en la herramienta, pero hay funcionalidades no disponibles si el usuario no está logado. Accedemos al apartado del menú de la izquierda *Buscar lugar*.

The screenshot shows the search results for 'potito'. The result is displayed in a card format with a thumbnail of a person, the name 'potito', a short description ('Local situado en muñoz arenillas, cercano a la playa. Gran ambiente y la mejor música de la ciudad.'), and location details ('cadiz - cadiz'). Below the description are buttons for 'Tipo de música:' (Pop, Dance, Reggaeton) and a 'Ver Lugar...' button. There is also a second, partially visible result for 'crívi'.

Figura 9.8 – Búsqueda de lugares jUCAbbox

Se nos mostrarán todos los lugares disponibles de la herramienta. Y podremos buscarlo por nombre, provincia, ciudad y/o tipo de música.

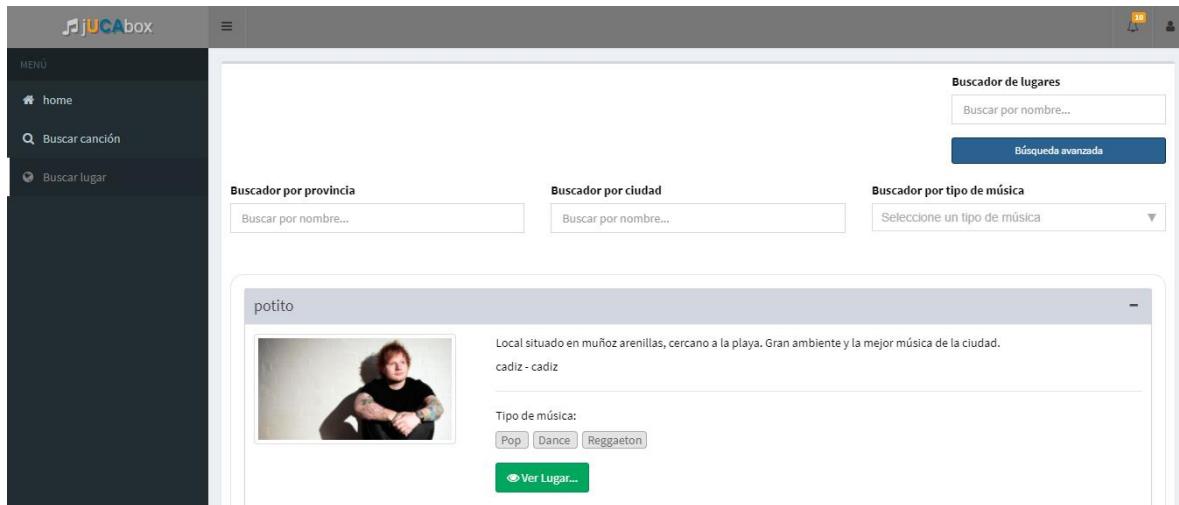


Figura 9.9 – Búsqueda avanzada de lugares jUCAbbox

Una vez encontrado el lugar buscado, podemos acceder al mismo haciendo click en el botón *ver lugar*.

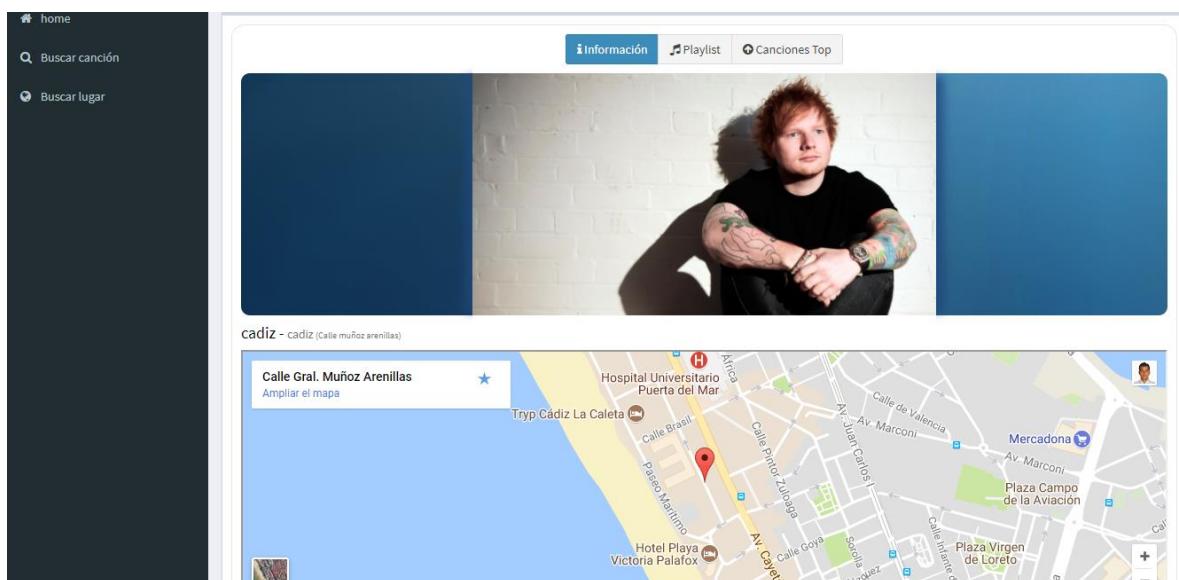


Figura 9.9 – Detalle de lugar jUCAbbox

Podemos observar que en la pantalla principal del lugar, tenemos la dirección, descripción y tipo de música, así como una foto.

Tenemos 3 opciones para los usuarios no registrados en la herramienta.

- **Información**

- Donde se pueden observar los datos más relevantes del local.

- **Playlist**

- Donde podrá observar que canciones hay actualmente en fase de validación por

parte del local.

- **Canciones Top**

- Lista de canciones más sonadas en el local en un rango de fechas determinado.

The screenshot shows the jUCAblox web application. On the left is a dark sidebar with a menu: 'home', 'Buscar canción', and 'Buscar lugar'. The main area has a header 'potito' and tabs for 'Información', 'Playlist', and 'Canciones Top'. Below is a section titled 'Pending to validate' with two entries:

Preview	Song	Petitions	Last Petition
	Dream On - Aerosmith	1	17 minutes ago
	Me Enamoré De Ti - David Bisbal	1	24/06/2017 12:52

At the bottom is a 'Back to places...' button.

Figura 9.10 – Canciones solicitadas al local jUCAblox

Podemos ver que aparece la canción que hemos solicitado, y el tiempo que hace que se ha realizado, además del número de peticiones realizadas.

This screenshot shows the same jUCAblox interface as Figure 9.10, but with a date range filter applied. The 'Fecha Inicio' is set to '08/06/2017' and the 'Fecha Fin' is set to '08/07/2017'. A 'Filtrar' (Filter) button is visible. The results show the top songs for that period:

Preview	Song	Number of petitions
	Me Enamoré De Ti - David Bisbal	1
	Hijos Del Mar - David Bisbal	1
	Dígale - David Bisbal	1

Figura 9.11 – Top Canciones local jUCAblox

En esta sección se pueden ver cuáles son las canciones más escuchadas en el local. Pudiendo filtrarse por fechas.

Estas son las funciones que pueden realizar los usuarios sin estar registrados. Una versión simplificada y sencilla de la herramienta.

#### 9.4.4. Registro e inicio de sesión

Para facilitar el registro e inicio de sesión en la herramienta, se ha dotado de la posibilidad de registro e inicio con las credenciales de Google, Facebook o Twitter, además de las de la propia herramienta.

En el caso de credenciales externas, el sistema redirige a la página del proveedor y una vez introducido el usuario y contraseña, devuelve las credenciales del usuario.

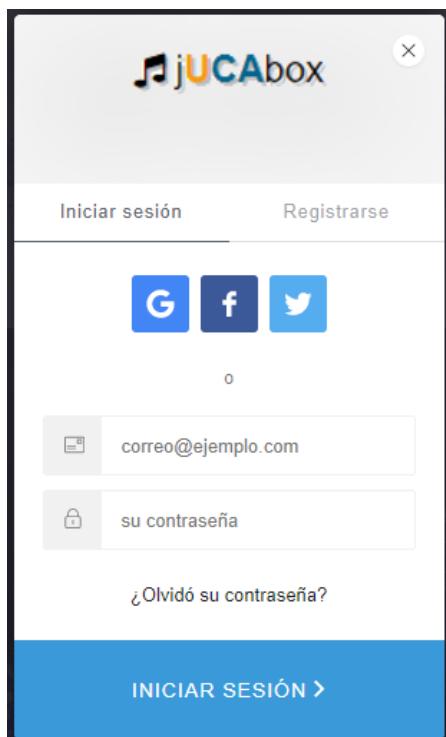


Figura 9.12 – Registro / Inicio de sesión jUCAbbox

Al logarnos en la herramienta, la pantalla principal cambia respecto al invitado. Aparece la lista de mensajes de los lugares a los que tenemos añadido como favorito, así como la opción de logarnos con Spotify si tenemos cuenta. Esta opción solo es interesante si queremos registrar un lugar nuevo en la herramienta.



Figura 9.13 – Pantalla principal usuario Logado jUCAbbox

#### 9.4.5. Búsqueda de canciones y envío de peticiones usuarios registrados

Respecto a las diferencias frente a los usuarios no registrados en los puntos mencionados anteriormente, es la posibilidad de agregar como favorito a algún artista seleccionado.

The screenshot shows the jUCAbbox application displaying an artist profile for 'Aerosmith'. The profile includes a circular thumbnail of the band members, the artist's name 'Aerosmith', a genre list ('Géneros: album rock, alternative rock, classic rock, hard rock, mellow gold, rock'), and a follower count ('Seguidores: 1905258'). Below the profile, there are three buttons: 'Ir a spotify', 'Buscar otros artistas', and 'Añadir a favoritos'. To the right, there is a search bar labeled 'Buscador de canciones' with the placeholder 'Buscar canciones...'. Below the search bar, there is a table titled 'Top Canciones' listing three songs: 'Dream On', 'I Don't Want to Miss a Thing - From the Touchstone film, "Armageddon"', and 'Sweet Emotion'. Each song entry includes the album name ('Aerosmith', 'I Don't Want To Miss A Thing', 'Toys In The Attic') and two action buttons: 'Preview' (blue arrow) and 'Enviar' (red arrow).

Figura 9.13 – Artista usuario registrado jUCAbbox

En el envío de peticiones al local, podremos escoger entre 2 listados, uno el de los locales añadidos a favoritos y luego otro con todos los locales.

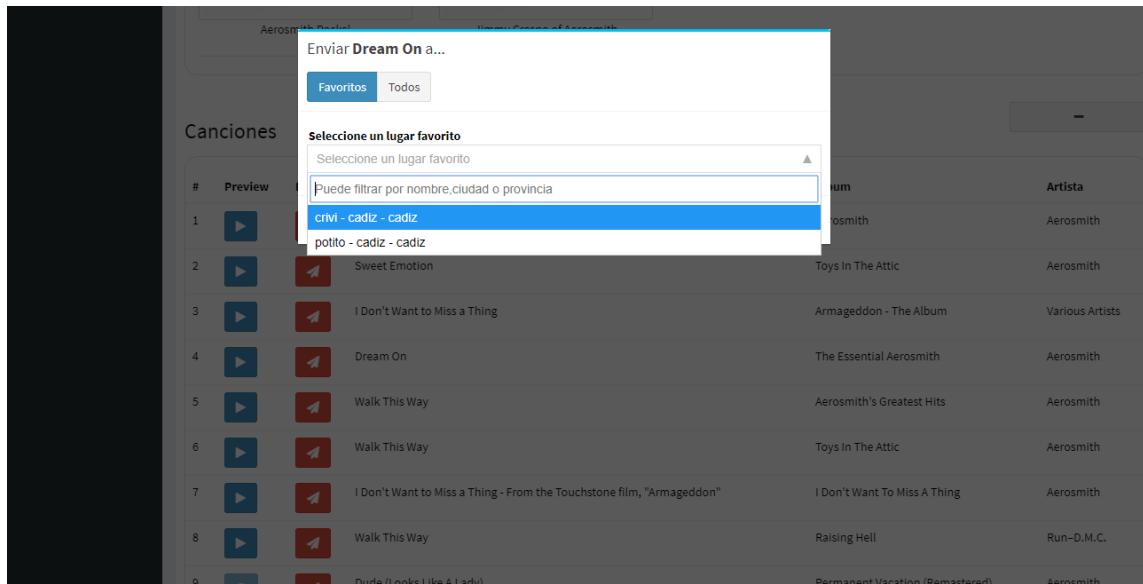


Figura 9.13 – Envío de canciones de usuarios registrados jUCAbbox

#### 9.4.6. Búsqueda de lugares de usuarios registrados

La gran diferencia con los usuarios no registrados, es el poder añadir como favoritos a cualquier lugar disponible en la herramienta, y el poder crear nuevos lugares.

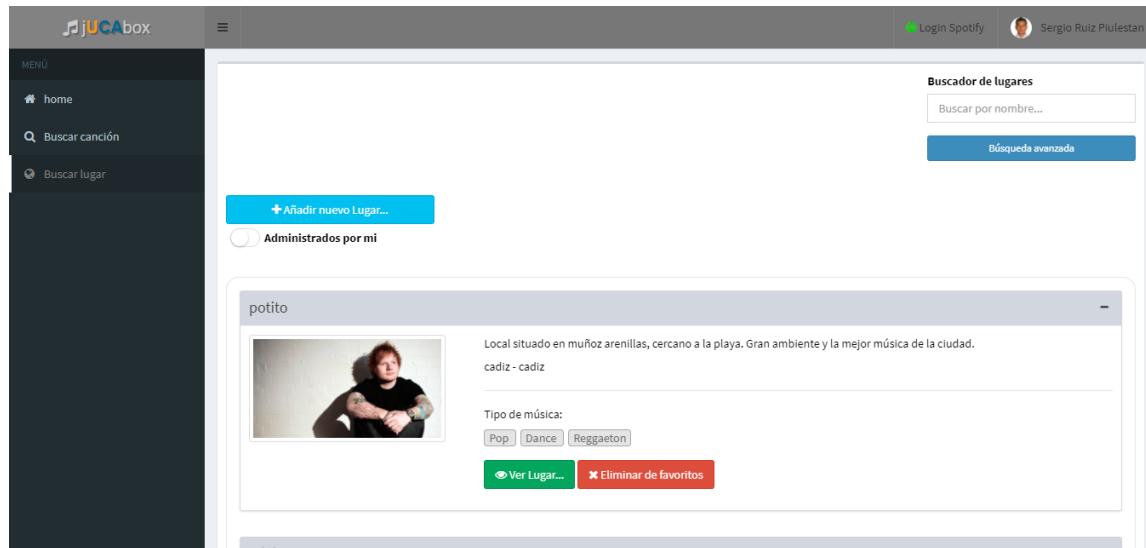


Figura 9.14 – Búsqueda de lugares de usuarios registrados jUCAbbox

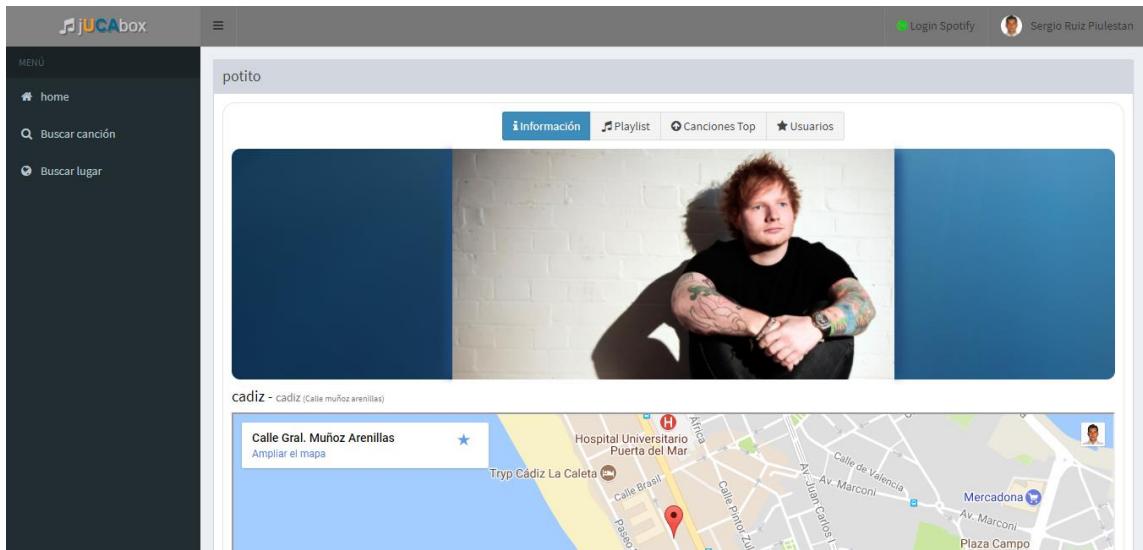


Figura 9.15 – Detalle de lugar usuarios registrados jUCAbbox

Podemos observar que se ha habilitado una nueva opción del menú *Usuarios*. Donde se podrán ver los usuarios que tienen el lugar como favorito.

#### 9.4.7. Área de usuarios

Para acceder al área de usuarios, deberemos hacer click a la foto situada en la esquina superior derecha. Donde podremos acceder al perfil o hacer salir de la sesión.

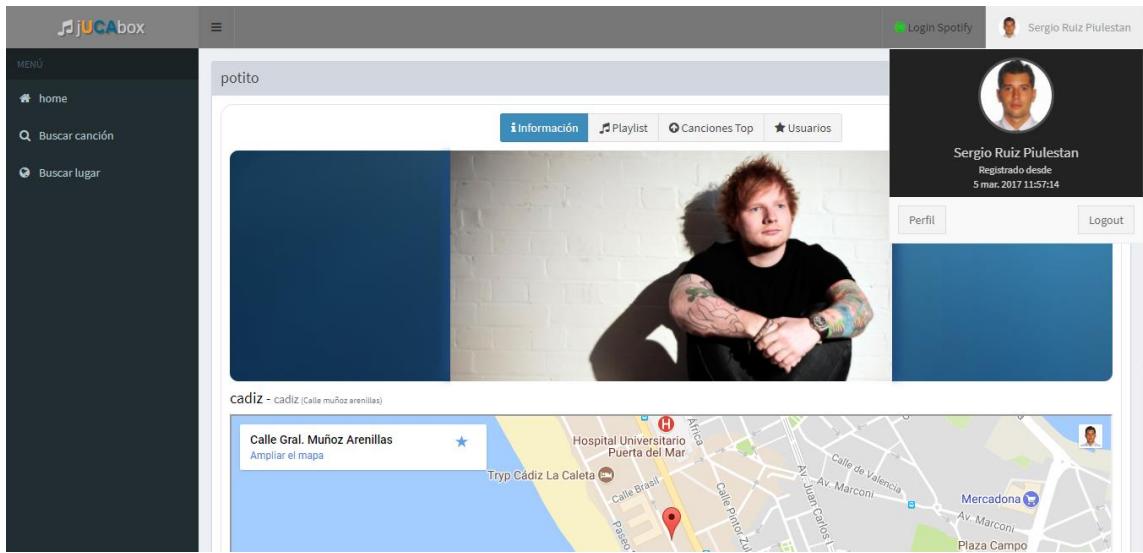


Figura 9.16 – Acceso área de usuarios jUCAbbox

### 9.4.8. Perfil usuario

Una vez dentro, tenemos varias opciones disponibles en el menú superior.

- Historial de acciones
  - El usuario puede ver las interacciones realizadas con la herramienta.
- Usuarios
  - El usuario puede buscar otros usuarios, agregarlos como amigos e incluso iniciar una conversación. Podrá visualizar el perfil de otros usuarios agregados como amigos.
- Artistas Favoritos
  - El usuario podrá ver un listado de los usuarios que tenga seleccionado como favoritos.
- Lugares Favoritos
  - El usuario podrá visualizar un listado de los lugares que haya seleccionado como favorito
- Información personal
  - El usuario podrá modificar la información personal incluida en la herramienta.

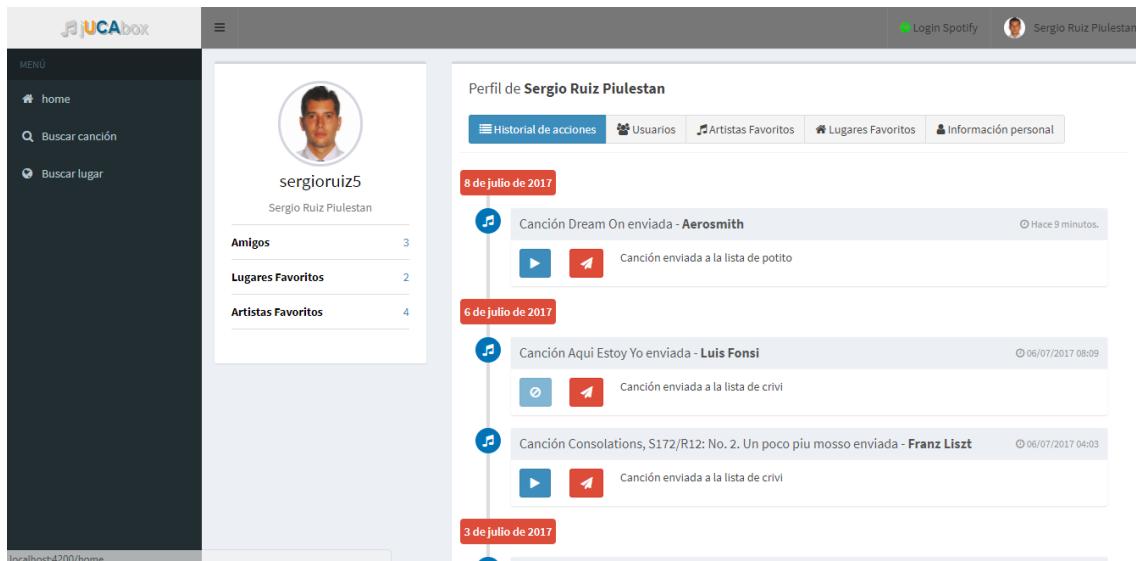


Figura 9.17 – Perfil usuario jUCAbbox

The screenshot shows the jUCAbbox user profile for 'sergioruiz5'. The left sidebar has a dark theme with white text. It includes a navigation menu with 'home', 'Buscar canción', 'Buscar lugar', 'Amigos' (3), 'Lugares Favoritos' (2), and 'Artistas Favoritos' (4). The main content area displays the user's profile picture and name 'sergioruiz5' with the subtitle 'Sergio Ruiz Piulestan'. Below this are sections for 'Amigos', 'Lugares Favoritos', and 'Artistas Favoritos'. The 'Artistas Favoritos' section is currently active, showing a list of favorite artists.

Figura 9.18 – Lista de usuarios jUCAbbox

The screenshot shows the jUCAbbox artist favorites page for 'sergioruiz5'. The left sidebar is identical to Figure 9.18. The main content area is titled 'Perfil de Sergio Ruiz Piulestan' and shows the 'Artistas Favoritos' tab selected. It lists three artists: 'Metallica', 'Various Artists - Metallica Tribute', and 'Metallica Tribute Band'. Each entry includes a small thumbnail image, the artist name, a genre list, and two buttons: 'Eliminar artista favorito' (red) and 'Ver canciones TOP' (blue).

Figura 9.19 – Lista de artistas favoritos jUCAbbox

Figura 9.20 – Lista de lugares favoritos jUCAbbox

Figura 9.21 – Información personal usuario jUCAbbox

Estas son las funciones para usuarios registrados, que no son administradores de algún lugar. En el siguiente capítulo iniciaremos sesión con el usuario administrador del lugar utilizado en ejemplos anteriores.

#### 9.4.9. Búsqueda de lugares de usuarios administradores

La búsqueda es exactamente igual que en casos anteriores, con la única diferencia que si somos administradores de algún lugar, nos aparecerá un botón *Administrar*.

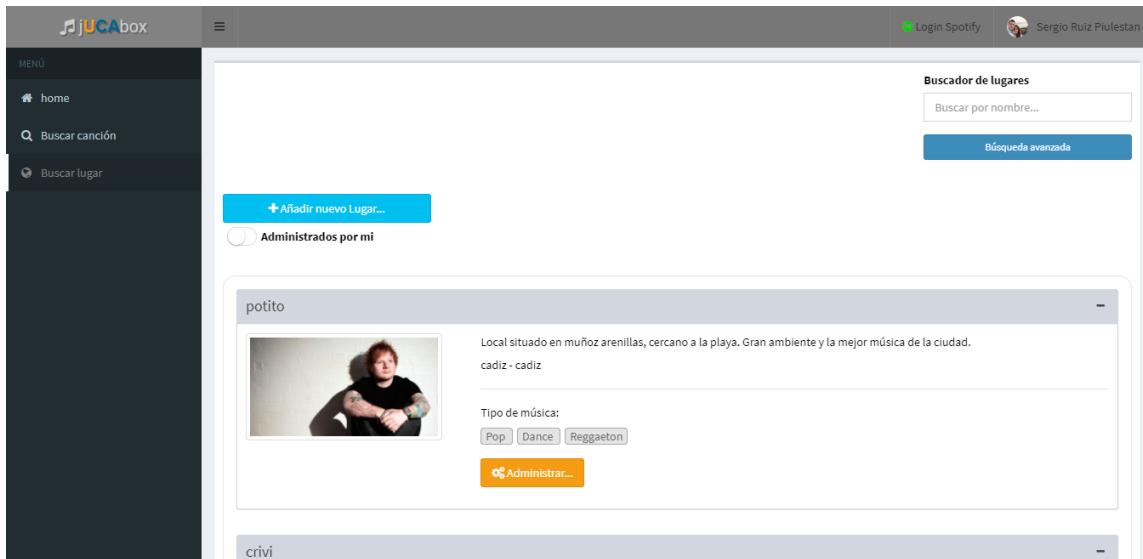


Figura 9.14 – Búsqueda de lugares de usuario administrador jUCAbbox

Una vez dentro del lugar, podremos observar que tenemos dos nuevas opciones en el menú superior.

- Mensajes
  - Podremos escribir mensajes y aparecerán en el tablón principal de usuarios que tengan el lugar añadidos a favoritos.
  - Se podrán añadir todo tipo de mensajes, desde texto plano a imágenes y enlaces a videos.
- Editar Información
  - Podremos editar la información relacionada con el lugar.

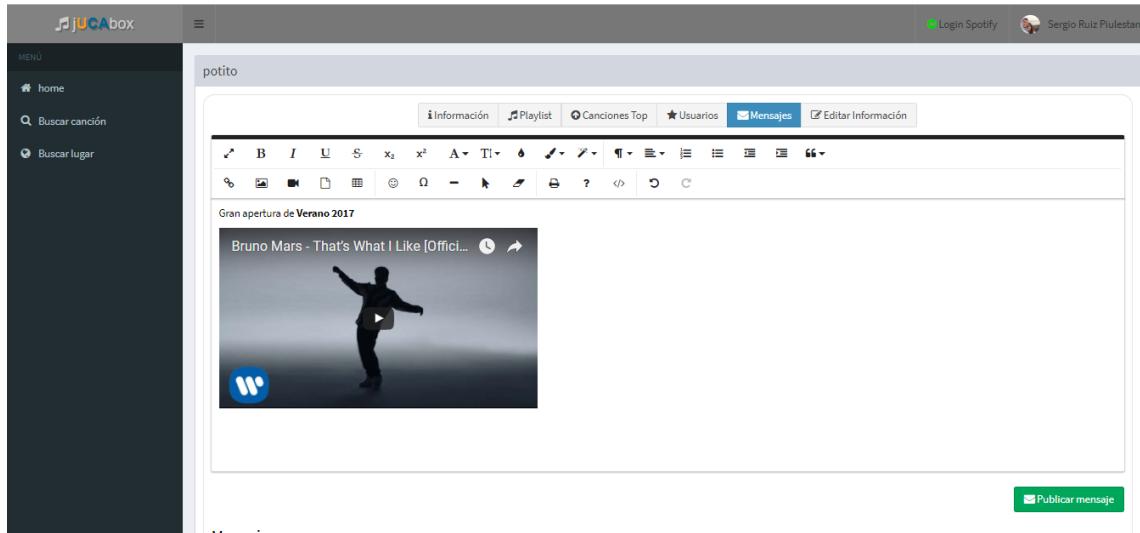


Figura 9.15 – Publicar mensaje desde lugar administrador jUCAbbox

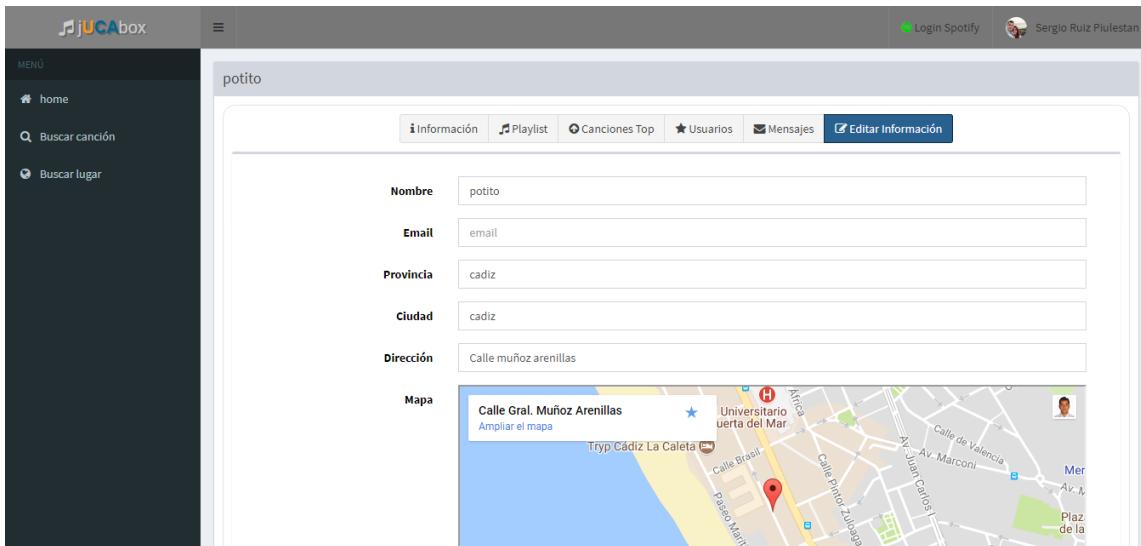


Figura 9.16 – Editar información acerca del lugar jUCAbbox

Además se han añadido en la parte de la playlist la función de administrar las canciones enviadas para su validación. En esta sección es necesario tener una cuenta de Spotify y logarnos con ella. Si no estuviéramos logados nos saldría la siguiente pantalla.

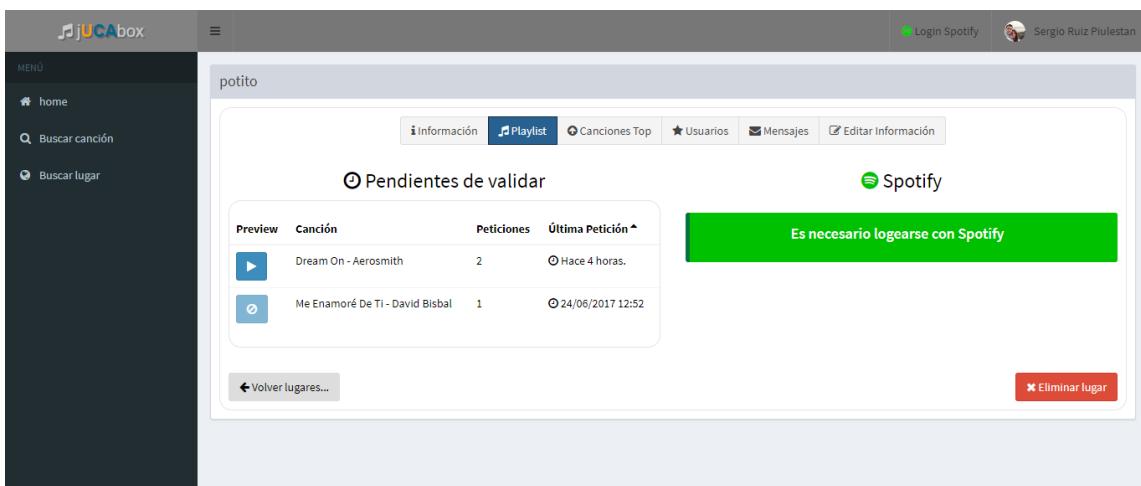


Figura 9.16 – Playlist lugar administrador jUCAbbox

Al lado de la foto de perfil deberemos iniciar sesión con Spotify. Una vez realizado el login, podremos ver unas nuevas opciones. La posibilidad de aceptar o rechazar la canción, y la elección de las playlists que tengamos asociadas a nuestra cuenta de Spotify. Si no existiera ninguna podríamos crearla a través de la herramienta.

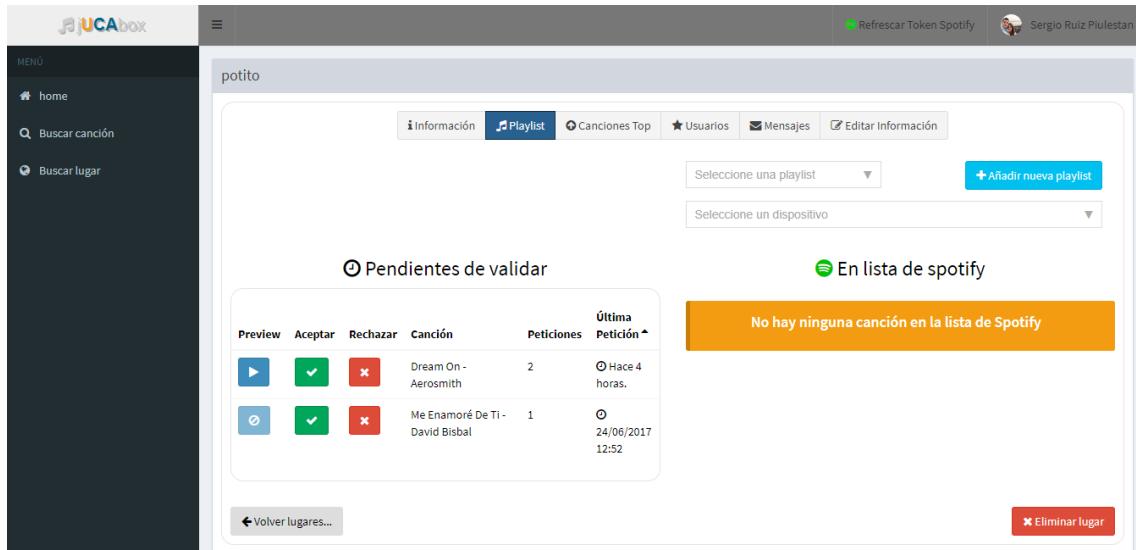


Figura 9.17 – Playlist lugar administrador jUCABox

También aparece un desplegable de dispositivos. Esto es opcional, si queremos reproducir la lista de canciones directamente desde la herramienta, deberemos elegir desde que dispositivo. Se nos habilitará un Player con la información de la canción reproducida.

Al seleccionar la playlist, se nos mostrarán las canciones incluidas en esa playlist, pudiendo borrarlas, o moverlas de lugar.

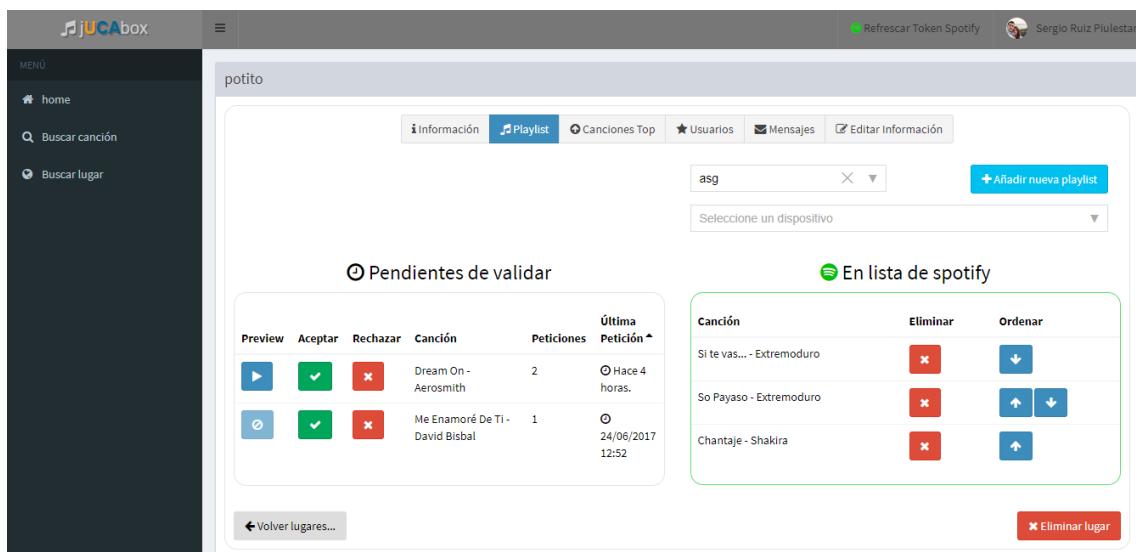


Figura 9.18 – Playlist lugar administrador Spotify jUCABox

Si validamos una canción, automáticamente pasa a la lista de Spotify. Si la rechazamos, desaparece del listado.

Figura 9.19 – Validación canción playlist lugar administrador Spotify jUCAbbox

Si desplegamos la lista de dispositivos y elegimos alguno, podremos reproducirlo haciendo doble click sobre la canción seleccionada.

Figura 9.20 – Reproducción de canción desde jUCAbbox



# **Capítulo 10**

## **Conclusiones**

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

### **10.1. Objetivos alcanzados**

Se han cumplido con todos los objetivos propuestos en la fase de análisis, profundizando aún más de lo propuesto en las funciones propuestas para la conexión de la herramienta con la API de Spotify. Además de conseguir que la herramienta sea más social de lo previsto, añadiendo una interacción entre usuarios.

Se ha conseguido hacer una aplicación sencilla pero robusta, pudiendo ser utilizada fácilmente desde cualquier dispositivo con conexión a internet.

### **10.2. Lecciones aprendidas**

En lo referente al proyecto uno de los principales objetivos ha sido aplicar los conocimientos obtenidos a lo largo de los años en la carrera de Ingeniera Informática. Gracias a estos conocimientos, se ha sido capaz de crear una herramienta desde cero, realizando paso por paso todas las funciones y procedimientos necesarios.

Además añadiendo el valor de la realización de la documentación, necesaria para todos los proyectos. Siendo útil para futuros mantenimientos y ampliaciones del software, incluso para posibles incidencias que puedan ocurrir. Por lo que resultado muy importante el aprendizaje obtenido a la hora de elaborar diagramas, casos de usos, diseño, así como la correcta redacción de un documento técnico.

Por el lado del software utilizado, ha sido muy enriquecedor aprender nuevos lenguajes de programación, así como todas las herramientas utilizadas a lo largo del proyecto. Siendo un trabajo fundamental investigar sobre nuevas técnicas y soluciones.

### **10.3. Trabajo futuro**

Una vez concluido con los objetivos propuestos en el proyecto, para trabajos futuros sería interesante crear una aplicación móvil nativa. Aunque la herramienta se adapta de forma adecuada al tamaño de cualquier dispositivo, es mucho más sencillo y ágil acceder directamente desde una aplicación instalada en el móvil.

De cara a nuevas funcionalidades propuestas, sería la posibilidad de crear distintos roles par un mismo lugar, es decir, varios administradores a distinto nivel. O poder crear lugares privados, que solo los usuarios puedan acceder y enviar canciones con invitaciones previas.



# Bibliografía

[Philippe Kruchten, 1999] Philippe Kruchten (1999). The Rational Unified Process: An Introduction (3rd Edition)

[Grady Booch, 1999] Grady Booch (1999). The unified modeling language user guide.

[Russ Miles et al., 2006] Russ Miles, Kim Hamilton (2006). Learning UML 2.0.

[Ari Lerner et al., 2016] Ari Lerner, Nate Murray, Felipe Coury, Carlos Taborda (2016). Ng-Book 2: The Complete Book on Angular 2.

[Nathan Rajlich et al., 2013] Nathan Rajlich, Mike Cantelon, T. J. Holowaychuk, Marc Harter (2013). Node.js in Action.

[Simon Holmes, 2015] Simon Holmes (2015). Getting MEAN With Mongo, Express, Angular, and Node.

[Tim Hawkins et al., 2010] Tim Hawkins, Eelco Plugge, Peter Membrey (2010). The Definitive Guide to MongoDB: A Complete Guide to Dealing with Big Data.



# GNU Free Documentation License

## GNU Free Documentation License

Version 1.3, 3 November 2008

© 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial,

philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, TeXinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## **2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## **3. COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## **4. MODIFICATIONS**

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Trans-parent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their WarrantyDisclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated

and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## **10. FUTUR REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## **11. RELICENSING**

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## **ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with . . . Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.