**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**

**SEDE VIÑA DEL MAR - JOSÉ MIGUEL CARRERA**

# “APLICACIÓN PARA DISPOSITIVOS MÓVILES: VINCULATOR "

Trabajo de Titulación para optar al Título

de Técnico Universitario INFORMATICA

Alumnos:

Pedro Enrique Montenegro Aravena

Daniel Ignacio Astudillo Barrientos

Profesor Guía:

Cahe Cabach, Ricardo.

**2018**

**RESUMEN**

El objetivo principal de este proyecto es crear una aplicación para dispositivos móviles con sistema operativo android 4.0 o superior que almacene los gustos de los contactos del usuario y en base a ellos recomendar locales o eventos que puedan satisfacer esos gustos. Para lograr esto, primero se debe crear una base de datos externa (en un servidor) que contenga todos los datos referentes a locales, eventos y productos.

Ya que recopilar información de locales y eventos es un trabajo complicado de llevar a cabo se toma la decisión de solamente contemplar los datos de la ciudad de Valparaíso, pero también la aplicación contará con una sección donde el usuario puede enviar datos de locales o eventos que estime son necesarios ingresar en la base de datos, una vez confirmada que es cierta la información del usuario se procede a ingresarla al servidor, esto facilita mucho la recopilación de datos. Por último, al no contar con un servidor que comparta información fuera de la red local la aplicación estará limitada a funcionar dentro de la red en la que está el servidor.

El proyecto contempla dos partes, la primera es la aplicación móvil y la segunda es una aplicación de escritorio que utiliza el administrador de la base de datos para manejar los datos de los locales, eventos y productos. La aplicación móvil se desarrollará con el lenguaje de programación JAVA, con el ambiente de desarrollo Android Studio y su base de datos local será administrada por SQLite. Mientras que la aplicación de escritorio será desarrollada por el mismo lenguaje, pero se trabajará con el ambiente de desarrollo NetBeans y su base de datos será administrada en MYSQL.

En el capítulo 1 se muestran los aspectos relevantes del diseño lógico, desde el detalle de la situación actual con respecto de las aplicaciones móviles y su relevancia en la actualidad, el mercado en el cual se encuentra y del público destino. Además se detalla el sistema propuesto, dejando en claro los objetivos, beneficios y la estructura funcional de éste.

A continuación en el capítulo 2 se puede observar el medio ambiente computacional tanto de desarrollo como de ejecución, junto con la descripción detallada de los archivos del sistema propuesto.

Luego el capítulo 3, el cual trata de la descripción de los programas, consta del detalle de cada uno de los programas a traves de diagramas modulares y de menú, tanto desde el punto de vista del usuario como del administrador. Además contempla una descripción detallada de los programas y los archivos involucrados en cada una de las funcionalidades del sistema.

Finalmente se presentan las conclusiones del trabajo demostrando todas las dificultades que se tuvieron al realizar este proyecto.

**ÍNDICE**


**INTRODUCCIÓN** 1

**CAPÍTULO 1: ASPECTOS RELEVANTES DEL DISEÑO LÓGICO** 3

**1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO** 5

**1.1 SITUACIÓN ACTUAL** **5**

1.1.1 Datos del mercado 6

1.1.2 Datos del público 7

1.1.3 Datos de la competencia 7

**1.2 PROBLEMAS DETECTADOS** **8**

**1.3 DESCRIPCIÓN DEL SISTEMA PROPUESTO** **10**

1.3.1 Objetivo general 10

1.3.2 Objetivo específico 10

1.3.4 Beneficios 10

1.3.5 Descripción general del sistema 11

1.3.6 Requerimientos legales 12

1.3.7 Estructura funcional del sistema 13

1.3.8 Entradas 14

1.3.9 Salidas 15

1.3.10 Entidades 16

1.3.10.1 Descripción de tablas y campos: 16

1.3.11 Modelo relacional del sistema 17

1.3.12 Estructura de códigos 18

1.3.13 Condicionantes de diseño 18

**CAPÍTULO 2: MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS** 19

**2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS** 21

**2.1 DESCRIPCIÓN DEL RECURSO COMPUTACIONAL** **21**

2.1.1 Configuración del sistema 21

2.1.1.1 Herramientas de desarrollo 21

2.1.1.2 Herramientas de ejecución 21

2.1.2 Software utilizado 22

**2.2 DESCRIPCIÓN DE ARCHIVOS** **23**

2.2.1 Contacto 23

2.2.2 Preferencia 24
2.2.3 Encuesta 25
2.2.4 Evento 25
2.2.5 Local 26
2.2.6 Categoría 27
2.2.7 Producto 28
2.2.8 Local_Prod 28
2.2.9 Género 29
2.2.10 Even_Gen 30
**CAPÍTULO 3: DESCRIPCIÓN DE PROGRAMAS** 33
**3.- DESCRIPCIÓN DE PROGRAMAS** 35
**3.1 DIAGRAMA MODULAR** **35**
**3.2 DIAGRAMA DE MENÚ** **36**
3.2.1 Menú usuario 36
3.2.2 Menú administrador 37
**3.3 TABLA DE PROGRAMAS** **38**
**3.4 DESCRIPCIÓN DETALLADA DE LOS PROGRAMAS** **39**
3.4.1 Menú principal 39
3.4.2 Mantenedor de contacto 40
3.4.3 Mantenedor de preferencias 43
3.4.4 Recomendación 45
3.4.5 Solicitud de registro local/evento 47
3.4.6 Mantenedor de local 49
3.4.7 Mantenedor de Evento 51
3.4.8 Mantenedor de producto 53
3.4.9 Mantenedor de Local-Producto 55
**CONCLUSIONES** 57
**BIBLIOGRAFÍA** 59
**ÍNDICE DE FIGURAS**
Fig. 1.2 Modelo de datos 17
Fig. 3.1 Diagrama Modular 35
Fig. 3.2 Diagrama de Menú usuario 36
Fig. 3.3 Diagrama de Menú administrador 37

Fig. 3.5 Menù de solicitudes 39
Fig. 3.6 Pantalla de contacto 40
Fig. 3.7 Agregar contacto 41
Fig. 3.8 Información del contacto 42
Fig. 3.9 Modificar contacto 42
Fig. 3.10 Eliminar contacto 43
Fig. 3.11 Pantalla de preferencias 44
Fig. 3.12 Agregar preferencia 44
Fig. 3.13 Eliminar preferencia 45
Fig. 3.14 Pantalla de recomendación 46
Fig. 3.15 Solicitud agregar local 48
Fig. 3.16 Mensaje de solicitud de registro 48
Fig. 3.17 Mantenedor de locales 50
Fig. 3.18 Mantenedor de eventos 52
Fig. 3.19 Mantenedor de productos 54
Fig. 3.20 Mantenedor de relación local producto 56

**ÍNDICE DE GRÁFICO**

Gráfico 1.1 Aplicaciones disponibles por tiendas virtuales según sistema operativo 6

**ÍNDICE DE TABLA**

Tabla 2.1 Contacto 23
Tabla 2.2 Preferencia 24
Tabla 2.3 Encuesta 25
Tabla 2.4 Evento 25
Tabla 2.5 Local 26
Tabla 2.6 Categoría 27
Tabla 2.7 Datos predeterminados tabla Categoría 27
Tabla 2.8 Producto 28
Tabla 2.9 Local_Prod 28
Tabla 2.10 Género 29
Tabla 2.11 Datos predeterminados tabla Género 29
Tabla 2.12 Even_Gen 31

Tabla 3.1 Resumen de programas 36

**ANEXOS** 61

Anexo 1: código fuente del menú principal 61

Anexo 2: código fuente de la pantalla de contactos 633

Anexo 3: código fuente agregar contacto 666

Anexo 4: código fuente información del contacto 699

Anexo 5: código fuente modificar contacto 722

Anexo 6: código fuente pantalla de preferencias 777

Anexo 7: código fuente agregar preferencia 799

Anexo 8: código fuente eliminar preferencia 81

Anexo 9: código fuente pantalla de recomendación 822

Código fuente php utilizado para la pantalla de recomendación 91

Anexo 10: código fuente solicitud de registro local/evento/producto 922

Anexo 11: código fuente Mantenedor Local 944

Anexo 12: código fuente Mantenedor Evento 988

Anexo 13: código fuente Mantenedor Producto 1022

Anexo 14: código fuente Mantenedor Local-Producto 1044

## INTRODUCCIÓN

En la actualidad la mayor parte de la población tiene acceso a un teléfono inteligente o Smartphone, por esto existen millones de aplicaciones para ayudar al usuario a satisfacer sus necesidades en cualquier ámbito posible. Uno de los tipos más conocidos de aplicaciones son las redes sociales, aplicaciones que tienen como finalidad poder facilitar la comunicación entre personas desde distintos lugares del mundo, aunque el uso excesivo de éstas ha provocado que la gente ya no sienta la necesidad de juntarse de manera física con otras personas, además que las personas cuando están en la "red" suelen ser muy intolerantes y se dedican a criticar todo lo que puedan, esto lleva a que de a poco (o quizás no tan poco) la gente se vuelva más individualista.

Por lo anterior, se requiere el crear una aplicación que rompa lo convencional, no satisfacer directamente las necesidades del usuario, si no ver las necesidades de los amigos del usuario y ayudar a que éste las satisfaga, esto pretende que la relación entre usuario-circulo social mejore y así poder mejorar la calidad de vida del usuario y sus amigos. De aquí surge la idea de Vinculator, una aplicación que busca almacenar los gustos de los amigos del usuario y en base a ellos recomendar locales o eventos para satisfacer esos gustos, todo para apoyar a que las personas se reunan, se conozcan mejor y así sus relaciones interpersonales crezcan favorablemente.

Todos los componentes de la aplicación están bajo las métricas de diseño que Android sugiere para poder tener la mayor compatibilidad entre todo tipo de dispositivos, ya sean de distintos tamaños o tengan distintas versiones del sistema operativo.

# CAPÍTULO 1:

# ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

# 1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

## 1.1 SITUACIÓN ACTUAL

Los avances tecnológicos en el campo de la comunicación siempre han sido objeto de estudio de las ciencias sociales, puesto que las nuevas formas de relación social que generan provocan una transformación en los hábitos y costumbres de la sociedad. Ya en el siglo XIX, la invención del telégrafo y, posteriormente, del teléfono implicaron un cambio en las relaciones personales y comerciales, que contaban con detractores y partidarios de las mismas. Las preocupaciones en contra señalaban al aislamiento social y la falta de intimidad como algunos de los perjuicios de estas innovaciones, y las opiniones a favor abogaban por el aumento de la interacción y conexión entre las personas. Con Internet y la explosión de las redes sociales estos puntos de vista no han cambiado demasiado de los de aquella época. Pero el progreso también evoluciona, y la diferencia obvia es su gran difusión en periodos mucho más cortos.

Tal como lo hacen otras innovaciones tecnológicas, el uso de las redes sociales transforma los estilos de vida, cambia las prácticas y, también, crea nuevo vocabulario, pero todo esto se produce a un ritmo tan acelerado que genera confusión y desconocimiento de la usabilidad y los derechos en torno a su actividad. No sólo se usan las redes para comunicarse de una manera rápida y efectiva. Hace tiempo que las personas se tornaron dependientes y sienten la necesidad de vivir 'conectados' continuamente.

Mucha gente ya no concibe su vida sin compartir absolutamente todo lo que hace o sin exhibir sus fotografías a través de Facebook, Twitter e Instagram, que son las tres plataformas más utilizadas en todo el mundo.

A la larga este tipo de actitudes pueden terminar resultando perjudiciales. Abusar de estas herramientas o hacer un mal uso de ellas puede generar innumerables problemas. Por ejemplo, en la actualidad los jóvenes han dejado de establecer conversaciones presenciales o ya no se generan tantas instancias o reuniones en las que los jóvenes socialicen y a su vez han perdido esa capacidad de crear vínculos con otras personas.

## 1.1.1 Datos del mercado

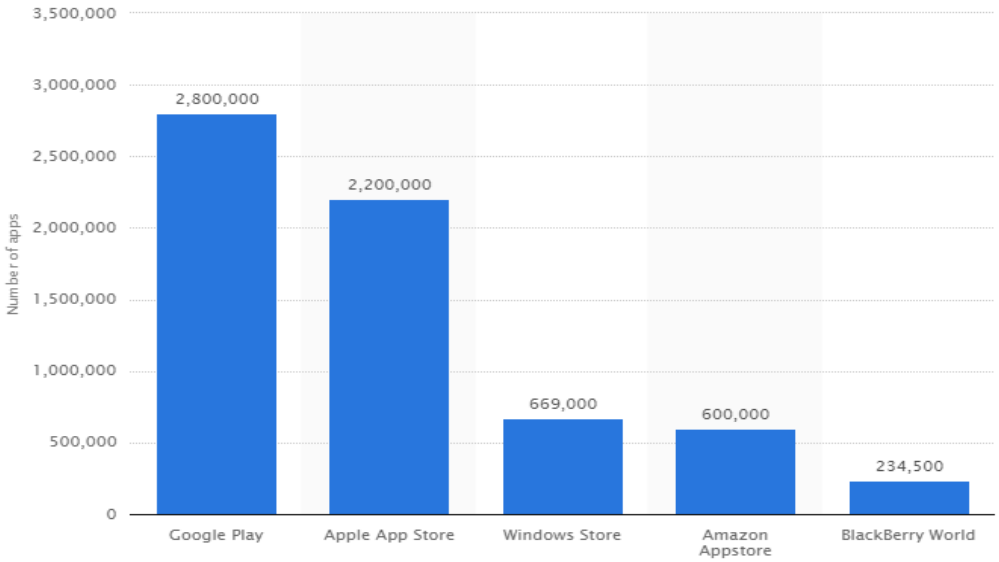
En la actualidad Chile encabeza la lista de países con mayor índice de uso de Smartphone o teléfonos inteligentes con respecto a Latinoamérica con un 52% aproximadamente y con un 78% con respecto a la gente que utiliza internet. Además, estos dispositivos móviles funcionan con su propio sistema operativo de los cuales el 85.3% corresponde al sistema operativo Android.

Claramente Android lidera este mercado y un estudio basado en la cantidad de usuarios y descargas que aumentan por año demostró que en la actualidad el número de usuarios activos supera los mil millones y las descargas anuales suman un total de 50 mil millones.

Fuente: https://iabtrends.cl/2016/08/09/chile-lidera-el-uso-de-smartphones-en-latinomaerica-con-7-9-millones-de-usuarios/

Fuente: http://www.emol.com/noticias/Tecnologia/2016/02/22/789497/Crece-el-uso-de-Internet-y-smarphones-en-paises-emergentes.html

Tomando en cuenta estos datos, es un gran momento para aprovechar este mercado el cual viene en aumento y desarrollar una aplicación para dispositivos móviles.




Fuente: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

Gráfico 1.1 Aplicaciones disponibles por tiendas virtuales según su sistema operativo.

### 1.1.2 Datos del público

Dentro de los usuarios de dispositivos móviles se destaca un público en particular que son los adolescentes y jóvenes de entre 14 a 34 años.

Es en esta etapa de la vida donde la amistad tiene una función muy importante en la integración de la sociedad. El hecho de sentirse integrado en el mundo y en la sociedad por medio de la amistad contribuye al mismo tiempo a reforzar el "Yo".

La amistad juvenil permite que se tome consciencia de la realidad del otro, se forman actitudes sociales y se toma experiencia en las relaciones interpersonales.

Al buscar el adolescente una forma de comunicarse y entablar amistad con otras personas se han insertado fuertemente en la sociedad las redes sociales como una herramienta para fortalecer sus vínculos.

Además dentro del público objetivo se encuentran todos aquellos locales que busquen promocionar sus servicios y ser ubicados de manera más simple a través de un sistema de geo localización.

Fuente: http://www.emol.com/noticias/Tecnologia/2017/05/04/856853/Chile-lidera-la-penetracion-de-internet-en-la-region-y-el-smartphone-continua-siendo-el-favorito.html

### 1.1.3 Datos de la competencia

Existen más de 2.8 millones de aplicaciones disponibles para sistemas Android por lo que hay varias aplicaciones que cumplen funciones similares o que cubren las mismas necesidades que la aplicación que será desarrollada. De las anteriores se destacan las siguientes:

- **FOURSQUARE:** Es una aplicación que con el uso de un mapa recomienda los lugares para comer cercanos a la posición del usuario y da los comentarios o reseñas de la gente sobre ese lugar. Esta aplicación actualmente cuenta con alrededor de 10 millones de descargas.
- **FOODSPOTTING:** Aplicación que destaca los mejores platos de la ciudad para que la gente al ir al restaurant cumpla sus expectativas. Esta aplicación cuenta con un millón

de descargas.

- **TINDER:** Aplicación que tiene por objetivo ayudar al usuario a establecer relaciones con personas cercanas a su ubicación. La cantidad de descargas sobrepasa los 50 millones.

Por lo descrito anteriormente, nace la idea este proyecto que busca que las personas puedan reunirse físicamente en algún local ya sea de comida, de eventos musicales y otros. Por lo anterior se propone construir una aplicación móvil que muestre ubicaciones de este tipo de locales cercanos a su ubicación dentro de la ciudad de Valparaíso.

## 1.2 PROBLEMAS DETECTADOS

Actualmente todas las aplicaciones están hechas para hacer que las personas pasen más tiempo conectadas al dispositivo móvil, esto se ve mayormente representado en las Redes Sociales, las cuales tienen por objetivo el rápido y sencillo contacto con personas que estén a una relativa larga distancia del usuario, pero que sin embargo está teniendo un efecto en el cual las personas se comunican tan seguido por estas redes, que no encuentran la necesidad de verse, o llamarse, ya que saben todo del otro sólo viendo sus perfiles en Facebook, fotos o historias en Instagram o simplemente hablando por WhatsApp. Además de esto, las aplicaciones solo buscan satisfacer las necesidades del usuario; esto provoca que el usuario se vuelva una persona individualista que se centra únicamente en satisfacerse a sí mismo, por todo esto se ve la necesidad de crear una aplicación que salga de lo tradicional, que busque satisfacer las necesidades de los amigos del usuario, a través de puntos de encuentro donde el usuario podrá ver, hablar y fortalecer su amistad con sus amigos compartiendo un grato momento juntos.

Como se menciona en el punto 1.1.3, las principales aplicaciones con funcionalidades parecidas al proyecto propuesto son: Foursquare, Foodspotting y Tinder.

Tanto Foursquare como Foodspotting se centran en satisfacer únicamente los gustos del usuario, por lo que si éste decide ir a un local de comida ofrecido por una de estas aplicaciones con otra persona, puede que ésta no se sienta cómoda en ese lugar, ya que sus gustos tal vez no sean satisfechos en ese lugar.

En el caso de la aplicación Tinder, si bien busca fomentar las relaciones sociales y

reunir a personas de manera física en espacios públicos, la manera en que lo hace es distinta a lo que busca “Vinculator”, ya que Tinder busca que el usuario conozca gente nueva, dejando de lado a los amigos, es una aplicación orientada a ayudar al usuario a encontrar pareja o algún tipo de relación sexual temporal.

Las principales diferencias entre Vinculator con las aplicaciones Foursquare y Foodspotting es que Vinculator busca lugares que cumplan las expectativas de los amigos (contactos) del usuario, para así mejorar su relación social. Además que no sólo se centra en lugares de comida, también busca otros panoramas como eventos musicales y cinematográficos.

A diferencia de Tinder, Vinculator no busca que el usuario conozca gente nueva, sino, que a la gente que ya conoce, la conozca mucho mejor y salgan a lugares donde ambos se sientan cómodos.

## 1.3 DESCRIPCIÓN DEL SISTEMA PROPUESTO

### 1.3.1 Objetivo general

Creación de una aplicación móvil la cual permita al usuario buscar y mostrar por pantalla de manera sencilla la ubicación de los locales (Restaurants, Cine) o eventos (Eventos musicales) cercanos, dentro de la cuidad de Valparaíso y que tengan relación con las preferencias de un contacto en específico.

### 1.3.2 Objetivo específico

- Crear una base de datos local para mantener información sobre las preferencias (Comida, Películas, Música) de una persona, para formar un perfil de ésta. Ej. Si le gustan sólo comidas con vegetales se puede concluir que esa persona, puede ser vegetariana, vegana o tiende a comer sano.
- Crear un software para computador con el fin de que el administrador pueda modificar la base de datos de manera sencilla.
- Crear una base de datos externa para mantener información sobre los locales y eventos que existan dentro de la ciudad de Valparaíso.
- Teniendo los gustos de una persona, la aplicación asistirá en la toma de decisiones al usuario, al ofrecerle distintas ideas de lugares donde posiblemente pueda satisfacer los gustos de un contacto.

### 1.3.4 Beneficios

- A diferencia de un software de escritorio, al ser una aplicación implementada en dispositivos móviles hace más fácil su acceso y más concurrente, ya que entregará información instantánea y en cualquier lugar.
- Sabiendo los gustos de los contactos y satisfaciéndolos, las relaciones de amistad entre usuario-contacto aumentarán y mientras más amigos se tengan mejor irá el bienestar emocional del usuario y tendrá mayor capacidad de establecer relaciones con personas, abriéndose cumplido el objetivo social

de la aplicación.

- Los locales u eventos que ingresen su ubicación en la base de datos se estarán haciendo publicidad al ser más fáciles de encontrar y al mostrar sus servicios, lo cual a su vez beneficiará a la aplicación al expandir la base de datos y la información que esta puede entregar.

### 1.3.5 Descripción general del sistema

Esta aplicación será implementada para dispositivos móviles y utilizará una base de datos local hecha con SQLite, contando también con un computador que se usará como servidor para tener información extra que será utilizada por esta aplicación. La información guardada en la base de datos local corresponderá a los contactos y sus preferencias (Gustos de los contactos, Música, Comida, etc.). Mientras que en el servidor se guardarán todos los datos referentes a Locales, Eventos y productos.

La recopilación de información para la base de datos local será a través de una encuesta al contacto del usuario. Esta consistirá en una pregunta (Ej. ¿Cuál es la comida favorita de "Contacto"?), que será desplegada por pantalla junto con un cuadro de texto para ingresar la respuesta. Esta pantalla se desplegará dos veces (una para usuario y otra para el contacto) con el fin de comparar respuestas y posteriormente entregar como salida un mensaje el cual mostrará que tanto conoce a su "entrevistado". A medida que la encuesta se desarrolla, sólo las respuestas del contacto se guardarán en la base de datos para su posterior uso.

La aplicación contará con un menú, el cual estará a disposición de los usuarios que deseen solicitar el registro de un local o evento, y de esta manera agregar su ubicación, contacto y otros detalles a la base de datos externa.

Los datos referentes a locales o eventos sólo podrán ser agregados, modificados o eliminados por el administrador de la base de datos. Los datos relacionados a un contacto podrán ser de igual manera modificados o eliminados por el usuario en cualquier momento.

Luego de la recopilación de datos y con la base de datos ya creada, se organizarán las preferencias de un contacto según categorías específicas (Ej.: Categoría de Comida abarcará platos preferidos, bebidas, etc.), con lo que el usuario tendrá la posibilidad de hacer una consulta en el momento que él desee sobre un contacto en específico, lo cual desplegará un menú de categorías. Al ingresar a la categoría deseada esta deberá desplegar por pantalla las ubicaciones de los lugares más cercanos al usuario según los datos de preferencia del contacto.

Las direcciones que se mostrarán por pantalla serán expuestas en un mapa para la mejor comprensión del usuario. Para este efecto se utilizará el servicio gratuito que ofrece Google Maps API, el cual permite la habilitación y uso de mapas en sistemas Android.

### 1.3.6 Requerimientos legales

**PERMISOS, LICENCIAS Y CONDICIONES DE USO**: Se debe ser claro y explícito a la hora de solicitar permisos al usuario para acceder a contactos de su dispositivo, realizar pagos o ceder datos. Además, es obligatorio desarrollar licencias y condiciones de uso. En todos los casos no basta con informar al usuario sino que éste tiene que aceptar, ya que en caso de reclamación se tendrá una mejor defensa.

**DERECHOS PROPIOS Y DE TERCEROS:** Es obligatorio disponer de licencias de los recursos que se vayan a utilizar. Para ello, hay que leer detenidamente las condiciones ya que hay casos en los que los recursos excluyen el uso comercial, no pudiéndose ejecutar en aplicaciones. Además, conviene proteger el contenido para evitar plagios y copias.

**PRIVACIDAD Y GEOLOCALIZACIÓN:** La recogida de información del usuario debe ser la indispensable para el funcionamiento de la App y éste debe tener la posibilidad de configurar la privacidad. Además, si la aplicación dispone de geolocalización, se tiene que contar con la aceptación del usuario para poder acceder a ella.

**INFORMACIÓN Y COOKIES:** Es fundamental informar al usuario de los aspectos regulados en la ley y mostrar los datos sobre los creadores y sobre quienes se encuentran tras la App. También es necesario que el usuario acepte las cookies, mediante un aviso informativo con la información básica y precisa sobre las mismas, y los aspectos exigidos por la ley.

**MARKETS:** Tienen condiciones muy estrictas para que se puedan publicar las aplicaciones por lo que hay que cumplir siempre lo que piden. De hecho, incluso cumpliendo las condiciones al colgar la app, éstas pueden cambiar y hacer que la aplicación no esté

disponible para usuarios nuevos. Un ejemplo que suelen alegar los Markets para rechazar una App es que su interfaz de usuario es compleja.

### 1.3.7 Estructura funcional del sistema

Las funciones contempladas en el sistema son:

**MANTENCIÓN DE UN CONTACTO:** Esta función permitirá agregar, modificar o eliminar los datos de un contacto ya existente.

**ENCUESTA:** Esta función despliega una pregunta por pantalla que debe ser respondida tanto por el usuario como por el contacto a quien se le realizará la encuesta, esto con el objetivo de saber los gustos de un contacto en específico.

**MANTENEDOR DE PREFERENCIAS:** Esta función permitirá al usuario de forma rápida, agregar o eliminar alguna preferencia de algún contacto en específico.

**ENLAZAR:** Esta función permite que al hacer click sobre el campo de contacto de algún local o evento, la aplicación lo dirija hacia su navegador predeterminado y busque la página web escrita en aquel campo de texto.

**MANTENER LOCAL:** Esta función le permite al administrador de la base de datos agregar, modificar y eliminar un local, mientras que al usuario solo le permitirá agregar.

**MANTENER EVENTO:** Esta función le permite al administrador de la base de datos agregar, modificar y eliminar un evento. También se eliminará un evento de manera automática luego de terminar dicho evento. Mientras que al usuario solo se le permite agregar un evento.

**MANTENER PRODUCTO:** Esta función le permite al administrador de la base de datos agregar, modificar o eliminar un producto, mientras que al usuario solo se le permite agregar.

**RECOMENDAR LUGARES:** Esta función muestra a través del sistema de Google Maps la localización de algún local que pueda satisfacer alguna preferencia del contacto, y además muestra todos los datos relacionados con dicho local (nombre, descripción, dirección, etc.).

**SOLICITAR REGISTRO DE LOCAL O EVENTO:** Esta función le permite al usuario mandar un correo directo al administrador de la base de datos externa con los datos de algún local o evento que el usuario estime que debe ser agregado.

1.3.8 Entradas

A continuación se describen los principales datos de entrada requeridos por el sistema:

- Datos del Contacto como número de teléfono, nombre del contacto, fecha de nacimiento.

- Datos de la encuesta, aquí se recopilan los gustos específicos de un contacto en las distintas categorías (comida, cinematográfico, música).

- Datos del local como el nombre del local, dirección, rubro (comida, música, cine), página oficial y descripción breve de los servicios que ofrece.

- Datos de los productos que ofrece un local como nombre del producto.

- Datos del evento como Nombre del evento, rubro (musical o cinematográfico), género al que pertenece (en caso de ser un evento musical puede ser pop, rock, románticas, etc. y si es uno cinematográfico puede ser una película romántica, de terror, comedia, etc.), Fecha, hora y dirección en la que se realizará, numero de contacto o página oficial de los organizadores del evento, descripción breve de lo que se hará.

1.3.9 Salidas

A continuación se muestran las salidas generadas por el sistema:

- Listado de Contactos: Muestra los contactos guardados por el usuario. Se compone de Nombre del contacto y número de teléfono.

- Información Contactos: Muestra por pantalla la información de contacto de un contacto en específico. Despliega el Número de teléfono, Nombre y fecha de nacimiento del contacto, además de las opciones de “Editar Información”, “Eliminar contacto” y “Preferencias”.

- Resultado de búsqueda Contacto: Muestra la información de un contacto en específico al hacer una consulta. Se compone de Nombre de contacto y Numero de contacto.

- Resultado de búsqueda Categoría: Muestra el nombre de la categoría junto con un listado de los gustos asociados a dicha categoría.

- Encuesta: Muestra por pantalla las preguntas de la encuesta asociadas a una categoría. Ej: ¿Cuál es la comida favorita del contacto?, ¿Cuál es la película favorita del contacto?, ¿Cuál es la canción favorita del contacto?

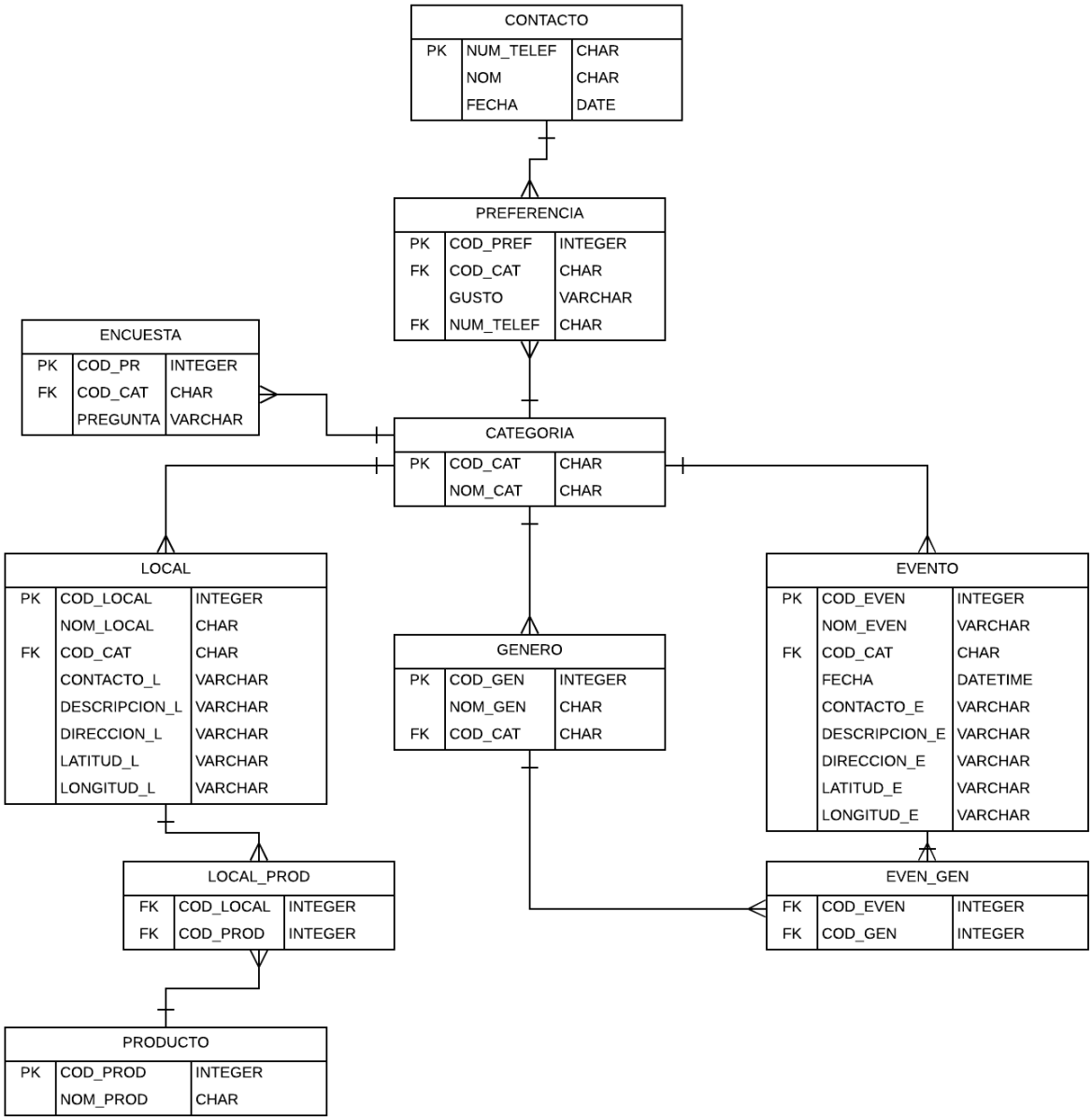
- Respuesta Encuesta: Despliega por pantalla un número del 0 al 100. Esto hace referencia al porcentaje de respuestas correctas del usuario durante la encuesta, tambien se muestra literalmente la cantidad de respuestas correctas e incorrectas.
- Recomendaciones: Muestra la información de algún local o evento especifico y su ubicación a través de un mapa.

## 1.3.10 Entidades

### 1.3.10.1 Descripción de tablas y campos:

- Contacto: Esta tabla almacenará los principales datos de los contactos.
- Categoría: Esta tabla contiene el nombre y código de las categorías disponibles en la aplicación. Estas pueden ser: Comida, Cinematográfico, Música y Otros.
- Preferencia: Esta tabla contiene los gustos o preferencias que tienen los contactos.
- Encuesta: Esta tabla almacena las preguntas que serán realizadas a los contactos.
- Local: Esta tabla guarda los nombres y direcciones de los locales que prestan algún servicio capaz de satisfacer la necesidad de algún contacto.
- Producto: Contiene los productos que puede ofrecer un local.
- Género: Contiene los posibles géneros dentro de una categoría musical (pop, rock, reggaetón, cumbia, etc.) o cinematográfica (terror, comedia, drama, romántica, etc.)
- Evento: Contiene los datos sobre los próximos eventos a ocurrir dentro de la ciudad.
- Local_Prod: Contiene los productos que ofrece un local determinado.
- Even_Gen: Contiene los distintos géneros que puede tener un evento; al igual que la tabla anterior, un evento puede tener uno o más géneros.

## 1.3.11 Modelo relacional del sistema




Fuente: Elaboración propia.

Fig 1.2 Modelo de datos.

1.3.12 Estructura de códigos

NUM_TELEF Es una clave primaria natural.

COD_PREF, COD_EVEN, COD_LOCAL, COD_PROD, COD_GEN, COD_PR Son claves primarias y son números correlativos.

COD_CAT: Clave primaria, puede ser “CO”,”CN”,”MU” y “OT” que corresponden a “Comida”, “Cinematográfico”, “Música” y “Otros”.

1.3.13 Condicionantes de diseño

Para el desarrollo de esta aplicación, es necesario contar con un lenguaje de programación capaz de trabajar con objetos, poseer herramientas necesarias para una óptima administración de los datos almacenados, así como también para una presentación amistosa y de calidad en la pantalla.

Se ha optado por el uso del entorno de desarrollo Android Studio, el cual utiliza el lenguaje de programación JAVA, pues este reúne todas las características necesarias para el manejo óptimo de archivos (base de datos). Genera como resultado presentaciones de calidad a un nivel competitivo con otros software en el mercado. Posee una biblioteca de herramientas poderosas que facilitan la programación. Además, se usará SQLite para gestionar la base de datos local.

Para el entorno de trabajo que utilizará el administrador de la base de datos tambien se optó por utilizar el lenguaje de programación JAVA a través del entorno de desarrollo llamado NetBeans y el paquete Wamp el cual incluye PHP, MYSQL y APACHE, que seran utilizados para la gestión de la base de datos externa y el enlace entre esta base de datos y la aplicación móvil.

Los tiempos de respuesta esperados dentro de la aplicación deben ser de tan solo segundos para que la experiencia del usuario sea grata y los resultados sean instantáneos.

La aplicación debe ser sin fines de lucro, ya que los servicios de Google Maps están disponibles de manera gratuita sólo para este tipo de aplicaciones, de lo contrario se dejarían de cumplir los requerimientos de la licencia gratuita de Google Maps.

Al ser una aplicación sin fines de lucro, no se cuenta con un presupuesto para comprar una base de datos ya hecha por otra empresa, por lo que en un comienzo el ingreso

de la información hacia la base de datos deberá ser de manera manual vía teclado, lo cual requiere una gran cantidad d ıda solamente en la ciudad de Valparaíso.

## CAPÍTULO 2:

## MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

# 2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

## 2.1 DESCRIPCIÓN DEL RECURSO COMPUTACIONAL

Para el desarrollo e implementación de esta aplicación, es necesario contar con ciertas características y herramientas básicas, tanto del equipamiento como del software a emplear.

A continuación se detallan las características, herramientas requeridas y empleadas para la parte de desarrollo y de ejecución de la aplicación.

### 2.1.1 Configuración del sistema

#### 2.1.1.1 Herramientas de desarrollo

La configuración usada para el desarrollo de esta aplicación son dos computadores, descritos a continuación.

PC modelo ONE-162a con un procesador APU A10 7860K 3.6 GHz / Radeon R7 (FM2+), 8 gb de memoria RAM y un disco duro de 1 terabytes.

PC modelo SVE14121CLB con un procesador Intel® Pentium ® CPU B980 2.4 GHz, 8 gb de memoria RAM y un disco duro de 271 gigabytes.

#### 2.1.1.2 Herramientas de ejecución

La aplicación podrá ser ejecutada por cualquier Smartphone que cumpla las siguientes características.

- Un procesador dual Core de 1,2 GHz
- Sistema Operativo Android 4.0 Ice Cream Sandwich o superior.
- Memoria RAM de 1 gb o superior.

### 2.1.2 Software utilizado

Uno de los computadores hará uso del Sistema Operativo Windows 10 de 64 bits y el otro usará Windows 8.1 de 64 bits. Adicionalmente se usarán para el desarrollo de la aplicación el entorno de desarrollo oficial de Android, llamado Android Studio, en el cual se programará la interfaz gráfica y las funcionalidades de la aplicación con el lenguaje de programación JAVA. Como complemento para la emulación del programa en un dispositivo móvil se utilizará el software Nox App Player. Para crear la interfaz gráfica del administrador se utilizará NetBeans.

Para la gestión de la base de datos local se utilizará el sistema SQLite, donde la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo, reduciendo la latencia en el acceso a la base de datos. Mientras que la base de datos remota será gestionada con MYSQL.

Ya que la aplicación necesitará en la base de datos externa la ubicación de los locales o eventos; no solo bastará con la dirección, sino que se debe conocer la latitud y longitud del lugar en concreto, para esto se utilizará la página web http://www.bufa.es/google-maps-latitud-longitud/. Este sitio web permite al usuario (en este caso al administrador) ingresar una dirección y este le entregará como resultado la latitud y longitud de ese lugar.

## 2.2 DESCRIPCIÓN DE ARCHIVOS

A continuación se indica la estructura de las tablas a utilizar, describiendo las tablas, claves primarias y foráneas, los campos junto con su tipo y longitud respectiva.

El formato que tendrán los campos de tipo DATETIME será dd/mm/aaaa hh:mm.

### 2.2.1 Contacto

Nombré Lógico: CONTACTO

Descripción: Almacena los datos del contacto.

Clave Primaria: NUM_TELEF

Clave Foránea: No tiene.

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| NUM_TELEF | CHAR | 12 | Número de teléfono del contacto. |
| NOM | CHAR | 50 | Nombre o apodo del contacto. |
| FECHA_NAC | DATE | 10 | Fecha de nacimiento del contacto. |

Fuente: Elaboración propia

Tabla 2.1 Contacto

## 2.2.2 Preferencia

Nombré Lógico: PREFERENCIA

Descripción: Almacena los gustos del contacto.

Clave Primaria: COD_PREF.

Clave Foránea: COD_CAT (Referencia a tabla CATEGORIA).

NUM_TELEF (Referencia a tabla CONTACTO).

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_PREF | SHORT | 3 | Identificador de la preferencia. |
| COD_CAT | CHAR | 2 | Identificador de la categoría del gusto de un contacto. |
| NUM_TELEF | CHAR | 12 | Número de teléfono del contacto. |
| GUSTO | CHAR | 40 | Dato del gusto. |

Fuente: Elaboración propia

Tabla 2.2 Preferencia

### 2.2.3 Encuesta

Nombré Lógico: ENCUESTA

Descripción: Almacena las preguntas de la encuesta.

Clave Primaria: COD_PR.

Clave Foránea: COD_CAT (Referencia a tabla CATEGORIA).

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_PR | INTEGER | 2 | Identificador de la pregunta. |
| COD_CAT | CHAR | 2 | Identificador de la categoría de la pregunta. |
| PREGUNTA | VARCHAR | 100 | Pregunta relacionada a una categoría. |

Fuente: Elaboración propia

Tabla 2.3 Encuesta

### 2.2.4 Evento

Nombré Lógico: EVENTO

Descripción: Almacena los datos de un evento musical o cinematográfico.

Clave Primaria: COD_EVEN.

Clave Foránea: COD_CAT (Referencia a tabla CATEGORIA).

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_EVEN | INTEGER | 3 | Identificador del evento. |
| NOM_EVEN | VARCHAR | 50 | Nombre del evento. |
| COD_CAT | CHAR | 2 | Identificador de la categoría del evento. |
| FECHA | DATETIME | 16 | Fecha y hora en que se realiza el evento. |
| CONTACTO_E | VARCHAR | 200 | Página oficial para contactar a los |

| | | | |
|---|---|---|---|
| | | | organizadores del evento. |
| DESCRIPCION_E | VARCHAR | 250 | Describe el evento. |
| DIRECCION_E | VARCHAR | 100 | Dirección del lugar donde se realiza el evento. |
| LONGITUD_E | VARCHAR | 100 | Coordenada que indica la posición del evento en el plano ESTE u OESTE del planeta. |
| LATITUD_E | VARCHAR | 100 | Coordenada que indica la posición del evento en el plano NORTE o SUR del planeta. |

Fuente: Elaboración propia

Tabla 2.4 Evento

## 2.2.5 Local

Nombré Lógico: LOCAL

Descripción: Almacena los datos de un local.

Clave Primaria: COD_LOCAL.

Clave Foránea: COD_CAT (Referencia a tabla CATEGORIA).

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_LOCAL | INTEGER | 3 | Identificador del local. |
| NOM_LOCAL | CHAR | 50 | Nombre del local. |
| COD_CAT | CHAR | 2 | Identificador de la categoría del local. |
| CONTACTO_L | VARCHAR | 200 | Página oficial, correo o número de teléfono para contactar al local. |
| DESCRIPCION_L | VARCHAR | 250 | Describe los servicios que presta el local. |
| DIRECCION_L | VARCHAR | 100 | Dirección del local. |
| LONGITUD_L | VARCHAR | 100 | Coordenada que |

| | | | |
|---|---|---|---|
| | | | indica la posición del local en el plano ESTE u OESTE del planeta. |
| LATITUD_L | VARCHAR | 100 | Coordenada que indica la posición del local en el plano NORTE o SUR del planeta. |

Fuente: Elaboración propia

Tabla 2.5 Local

## 2.2.6 Categoría

Nombré Lógico: CATEGORIA

Descripción: Almacena los nombres de las categorías.

Clave Primaria: COD_CAT.

Clave Foránea: No tiene.

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_CAT | CHAR | 2 | Identificador de la categoría del gusto de un contacto. |
| NOM_CAT | CHAR | 20 | Nombre de la categoría. |

Fuente: Elaboración propia

Tabla 2.6 Categoría

Datos Predeterminados:

| COD_CAT | NOM_CAT |
|---|---|
| CO | COMIDA |
| MU | MUSICA |
| CI | CINEMATOGRAFICO |
| OT | OTROS |

Fuente: Elaboración propia

Tabla 2.7 Datos predeterminados tabla Categoría

## 2.2.7 Producto

Nombré Lógico: PRODUCTO

Descripción: Almacena los nombres de los productos.

Clave Primaria: COD_PROD.

Clave Foránea: No tiene.

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_PROD | INTEGER | 3 | Identificador del producto. |
| NOM_PROD | CHAR | 30 | Nombre del producto. |

Fuente: Elaboración propia

Tabla 2.8 Producto

## 2.2.8 Local_Prod

Nombré Lógico: LOCAL_PROD

Descripción: Esta tabla actúa como intersección entre Local y Producto.

Clave Primaria: COD_LOCAL + COD_PROD.

Clave Foránea: No tiene.

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_LOCAL | INTEGER | 3 | Identificador del local. |
| COD_PROD | INTEGER | 3 | Identificador del producto. |

Fuente: Elaboración propia

Tabla 2.9 Local_Prod

## 2.2.9 Género

Nombré Lógico: GENERO

Descripción: Almacena los nombres de los géneros.

Clave Primaria: COD_GEN.

Clave Foránea: COD_CAT (Referencia a tabla CATEGORIA).

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_GEN | INTEGER | 3 | Identificador del género. |
| COD_CAT | CHAR | 2 | Identificador de la categoría del gusto del contacto. |
| NOM_GEN | CHAR | 30 | Nombre del género. |

Fuente: Elaboración propia

Tabla 2.10 Género

Datos Predeterminados:

| COD_GEN | COD_CAT | NOM_GEN |
|---|---|---|
| 1 | DRAMA | CI |
| 2 | COMEDIA | CI |
| 3 | ACCION | CI |
| 4 | CIENCIA FICCION | CI |
| 5 | FANTASIA | CI |
| 6 | TERROR | CI |
| 7 | ROMANCE | CI |
| 8 | MUSICAL | CI |
| 9 | MELODRAMA | CI |
| 10 | SUSPENSO | CI |
| 46 | SUPERHEROES | CI |
| 47 | FlAMENCO | MU |
| 48 | MUSICA CLASICA | MU |
| 49 | MUSICA ELECTROACUSTICA | MU |
| 50 | MUSICA ELECTRONICA | MU |
| 51 | MUSICA LIGERA | MU |
| 52 | BALADA | MU |
| 53 | BEAT | MU |
| 54 | BLUES | MU |
| 55 | BOLERO | MU |
| 56 | COROS | MU |

| | | |
|---|---|---|
| 57 | COUNTRY | MU |
| 58 | CUMBIA | MU |
| 59 | CHA CHA CHA | MU |
| 60 | DISCO | MU |
| 61 | FUNKY | MU |
| 62 | GOSPEL | MU |
| 63 | HEAVY METAL | MU |
| 64 | HIP HOP | MU |
| 65 | JAZZ | MU |
| 66 | MAMBO | MU |
| 67 | MERENGUE | MU |
| 68 | MUSICA CELTA | MU |
| 69 | MUSICA ESPAÑOLA | MU |
| 70 | BANDAS SONORAS | MU |
| 71 | MUSICA RELIGIOSA | MU |
| 72 | POP | MU |
| 73 | RAP | MU |
| 74 | REGGAE | MU |
| 75 | REGGAETON | MU |
| 76 | ROCK | MU |
| 77 | ROCK AND ROLL | MU |
| 78 | RUMBA | MU |
| 79 | SALSA | MU |
| 80 | SOUL | MU |
| 81 | SKA | MU |
| 82 | TANGO | MU |
| 83 | TECHNO | MU |
| 84 | VALS | MU |
| 85 | VILLANCICOS | MU |
| 86 | FOLK | MU |
| 87 | CYBER PUNK | CI |
| 88 | DOCUMENTAL | CI |
| 89 | DIBUJOS ANIMADOS | CI |
| 90 | ANIME | CI |

Fuente: Elaboración propia

Tabla 2.11 Datos predeterminados tabla Género

## 2.2.10 Even_Gen

Nombré Lógico: EVEN_GEN

Descripción: Esta tabla actúa como intersección entre Evento y Género.

Clave Primaria: COD_EVEN + COD_GEN.

Clave Foránea: No tiene.

Descripción de Registros:

| Nombre de Campo | Tipo | Longitud Posición | Descripción |
|---|---|---|---|
| COD_EVEN | INTEGER | 3 | Identificador del evento. |
| COD_GEN | INTEGER | 3 | Identificador del género. |

Fuente: Elaboración propia

Tabla 2.12 Even_Gen

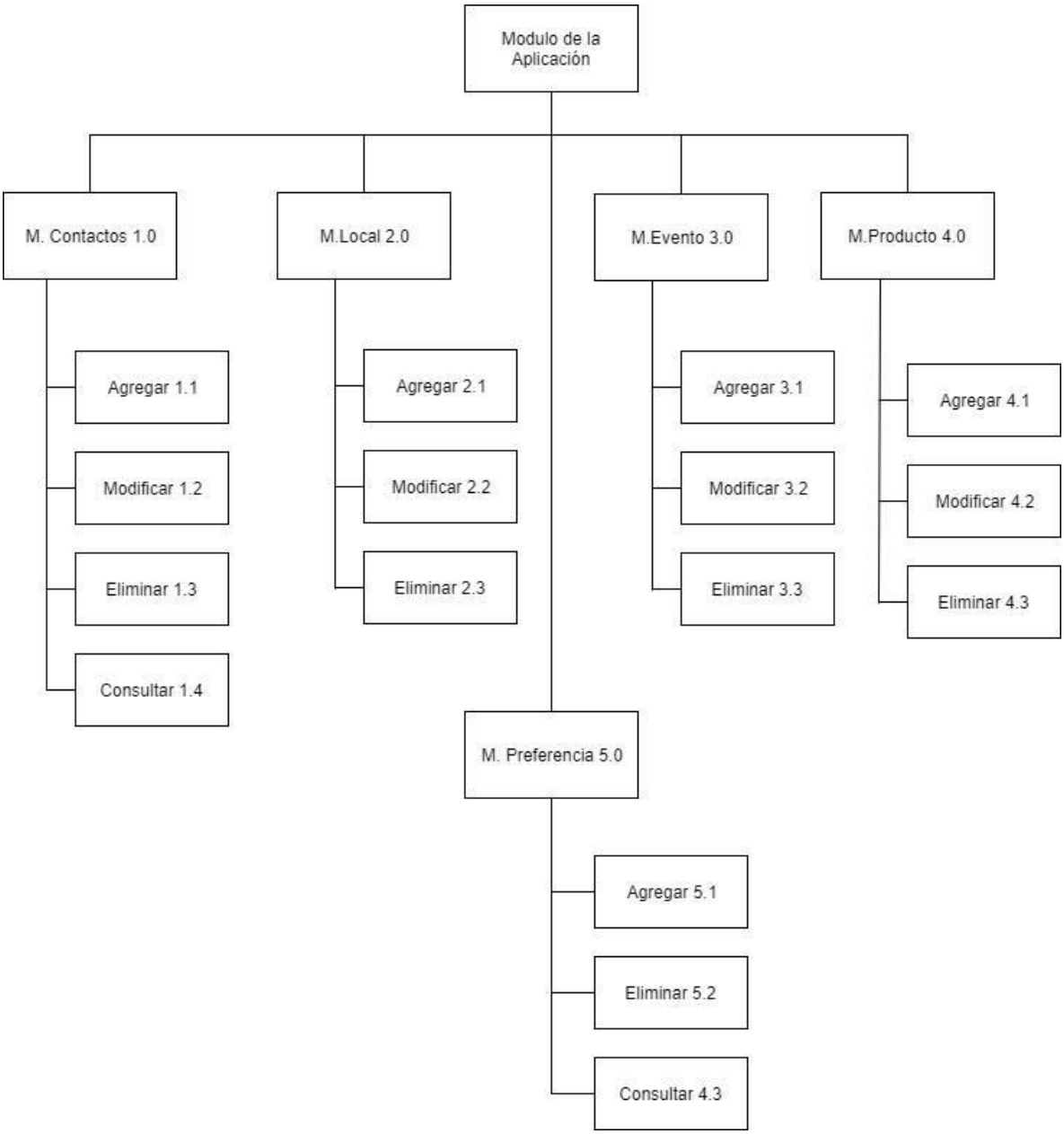
# CAPÍTULO 3:

# DESCRIPCIÓN DE PROGRAMAS

## 3.- DESCRIPCIÓN DE PROGRAMAS

### 3.1 DIAGRAMA MODULAR

A continuación se presenta el diagrama modular propuesto en el sistema.

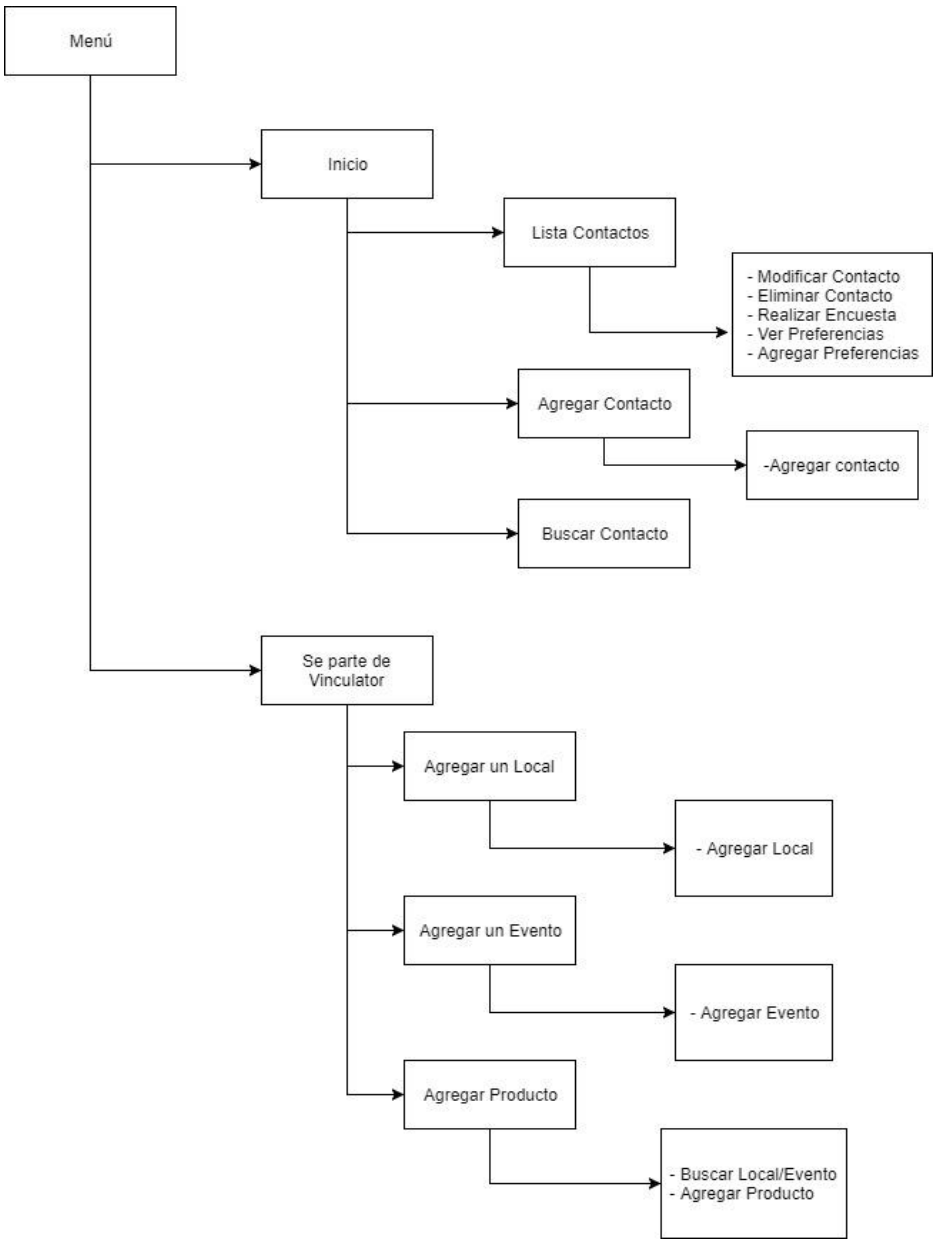



Fuente: Elaboración propia
Fig. 3.1 Diagrama Modular

## 3.2 DIAGRAMA DE MENÚ

### 3.2.1 Menú usuario

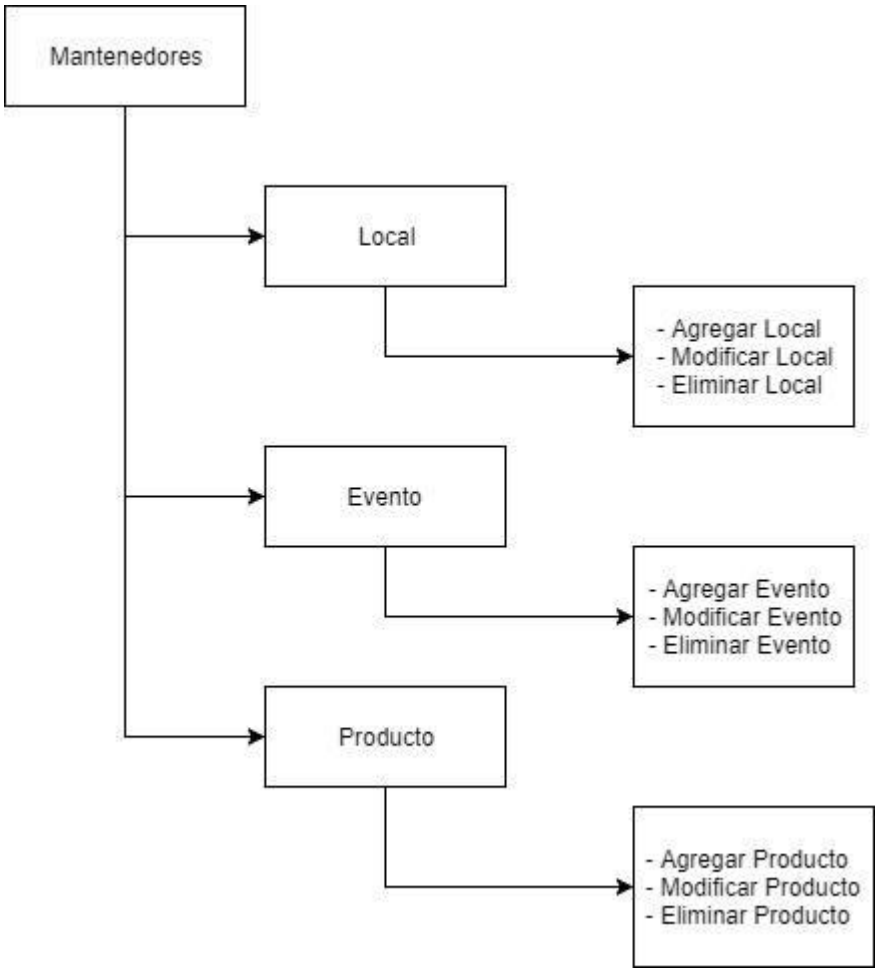
A continuación se presenta el diagrama de menú con las opciones correspondientes al usuario del sistema.




Fuente: Elaboración propia

Fig. 3.2 Diagrama de Menú usuario

## 3.2.2 Menú administrador




Fuente: Elaboración propia

Fig. 3.3 Diagrama de Menú administrador

## 3.3 TABLA DE PROGRAMAS

La siguiente tabla contiene todos los programas contenidos dentro del sistema propuesto y el objetivo de cada uno.

Tabla 3.1 Resumen de programas

| Tabla de Programas | |
|---|---|
| **Menú Principal** | Menú despegable que permite al usuario navegar entre las pantallas de la aplicación. |
| **Mantenedor Contacto(*)** | Permite Agregar, Listar, Modificar o Eliminar la información de un contacto almacenado en la tabla “CONTACTO”. |
| **Mantenedor Preferencia(*)** | Permite Agregar, Listar o Eliminar una preferencia asociada a un contacto, almacenada en la tabla “PREFERENCIA”. |
| **Mantenedor Local(*)** | Permite Agregar, Listar, Modificar o Eliminar la información de un local almacenado en la tabla “LOCAL”. |
| **Mantenedor Evento(*)** | Permite Agregar, Listar, Modificar o Eliminar la información de un evento almacenado en la tabla “EVENTO”. |
| **Mantenedor Producto(*)** | Permite Agregar, Listar, Modificar o Eliminar un producto almacenado en la tabla “PRODUCTO”. |
| **Mantenedor Local-Producto (*)** | Permite Agregar o Eliminar un producto asociado a un local en la tabla “LOCAL_PROD”. |
| **Encuesta** | Realiza una serie de preguntas para posteriormente agregar las respuestas a la tabla “PREFERENCIA”. |
| **Recomendación(*)** | Muestra por pantalla a través de un mapa las ubicaciones de locales o eventos. |
| **Enlazar** | Lleva al usuario a la página web oficial del local o evento. |
| **Solicitar Registro Local/Evento/Producto(*)** | Envía información de un local, los productos que este ofrece o evento al correo del administrador para su revisión. |

Fuente: Elaboración propia

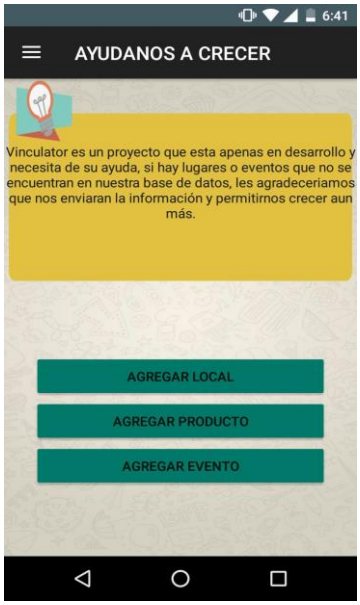
Los programas señalados con ‘*’ serán descritos en forma detallada a continuación.

## 3.4 DESCRIPCIÓN DETALLADA DE LOS PROGRAMAS

### 3.4.1 Menú principal

- Objetivo: Menú despegable que permite al usuario navegar entre las pantallas de la aplicación.
- Reglas de proceso:
  - El usuario cuenta con un menú con dos opciones (fig. 3.4), “inicio” y “se parte de vinculator”, de presionar inicio se muestra la fig. 3.6 y de presionar la otra opción se muestra la fig. 3.5 que muestra una pantalla con 3 botones, los cuales permiten al usuario enviar información al administrador de la base de datos (fig. 3.15).

Fuente: Elaboración propia
Fig. 3.4 Menú principal

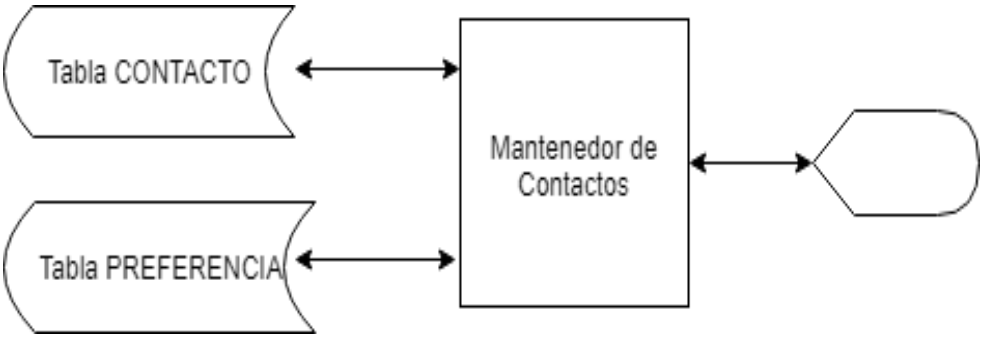



Fuente: Elaboración propia
Fig. 3.5 Menù de solicitudes

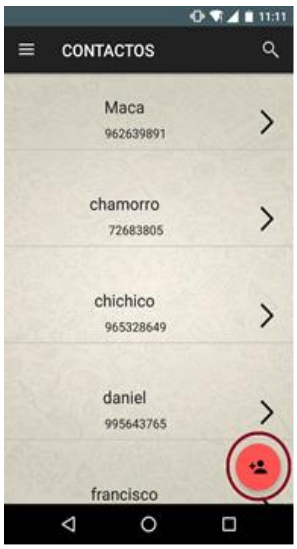
- Código fuente: Anexo 1 pág. 61.

### 3.4.2 Mantenedor de contacto

- Objetivo: Permite Agregar, Listar, Modificar o Eliminar la información de un contacto almacenado en la tabla "CONTACTO"
- Diagrama de bloque:




- Reglas de proceso:
  - Se despliega por pantalla un listado ordenado alfabéticamente con los nombres y números de teléfono de todos los contactos que tenga el usuario almacenado en la tabla CONTACTO (fig. 3.6).
  - El usuario dispone de un botón que lo lleva a la opción de agregar un nuevo contacto, este botón esta encerrado en un círculo **rojo**. También puede presionar el nombre de cualquier contacto para acceder a toda su información.




Fuente: Elaboración propia
Fig. 3.6 Pantalla de contacto

- De presionar el botón **rojo** el usuario accederá a la pantalla que se muestra en la fig. 3.7 y deberá completar todos los campos si desea agregar un nuevo contacto a la tabla CONTACTO, además, el número de teléfono no puede estar asociado a otro contacto ya existente y la fecha de nacimiento no puede ser mayor a la fecha actual.

Fuente: Elaboración propia
Fig. 3.7 Agregar contacto

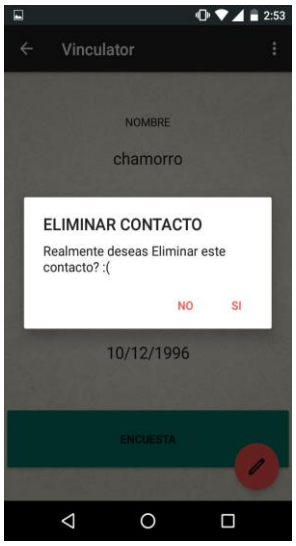
- Si por el contrario decide presionar sobre el nombre de contacto, la aplicación abrirá la pantalla mostrada en la fig. 3.8 que tendrá los detalles del contacto almacenados en la tabla CONTACTO y una serie de opciones, en un círculo **rojo** está la opción para modificar el contacto que esta actualmente seleccionado, en la esquina superior derecha se encuentran las opciones de preferencias que abre la pantalla de los gustos del contacto y la de eliminar el contacto. Por último el botón verde que abre la pantalla de encuesta.

Fuente: Elaboración propia
Fig. 3.8 Información del contacto

- Si presiona el botón **rojo** el usuario deberá ingresar los datos que desea modificar (fig. 3.9), no puede poner un número de teléfono relacionado a un contacto ya existente y no puede poner una fecha mayor a la fecha actual. Los datos modificados serán guardados en la tabla CONTACTO.

Fuente: Elaboración propia
Fig. 3.9 Modificar contacto

- Si presiona el botón eliminar contacto se presenta la fig. 3.10 en la cuál tendrá dos opciones:
  - SI: eliminará toda la información de este contacto de la tabla CONTACTO y también eliminará cada uno de sus gustos de la tabla PREFERENCIA.
  - NO: la aplicación quita el mensaje de eliminar y muestra la fig. 3.8.

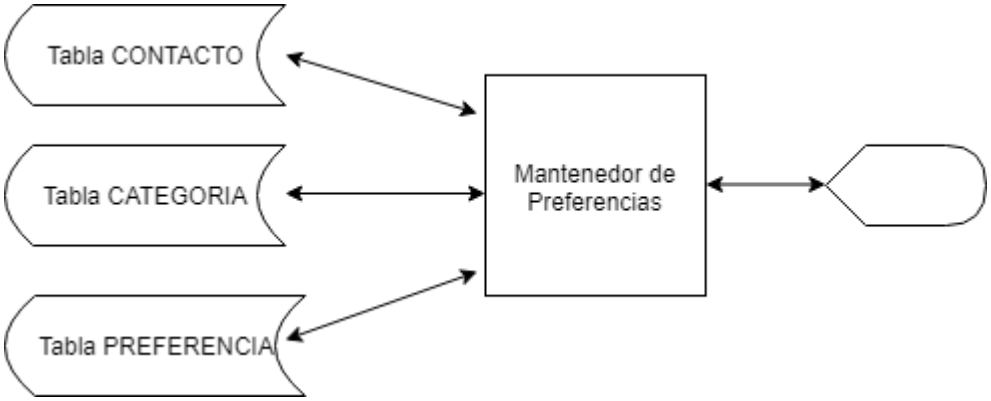



Fuente: Elaboración propia
Fig. 3.10 Eliminar contacto

- Código fuente: Anexo 2 pág. 63, Anexo 3 pág. 66, Anexo 4 pág. 71 y Anexo 5 pág. 74.

### 3.4.3 Mantenedor de preferencias

- Objetivo: Permite Agregar, Listar o Eliminar una preferencia asociada a un contacto, almacenada en la tabla "PREFERENCIA".
- Diagrama de bloque:




- Reglas de proceso:
  - Si en la fig. 3.8 el usuario presiona la opción de preferencias la aplicación buscará en la tabla CONTACTO el número del contacto actualmente visible y abrirá la pantalla de la fig. 3.11 donde se listan los gustos que el contacto tiene almacenados en la tabla PREFERENCIA, cada gusto se lista en 4 categorías, las cuales se leen desde la tabla (CATEGORÍA) mostradas en distintas pestañas de la pantalla, COMIDA, CINE, MÚSICA y OTROS.

Fuente: Elaboración propia
Fig. 3.11 Pantalla de preferencias

- Además en la fig. 3.11 existen dos opciones, presionar el botón **rojo** que abre la pantalla para agregar un nuevo gusto (fig. 3.12) o presionar alguno de los gustos que abre la pantalla de recomendación.

Fuente: Elaboración propia
Fig. 3.12 Agregar preferencia

- Si presiona el botón **rojo** el usuario deberá seleccionar obligatoriamente una categoría y escribir un gusto que serán almacenados en la tabla PREFERENCIA.
- Si presiona el gusto del contacto se abrirá la pantalla de recomendación la cuál tendrá en la esquina superior derecha una opción para eliminar el gusto

seleccionado (fig. 3.13).

Fuente: Elaboración propia
Fig. 3.13 Eliminar preferencia

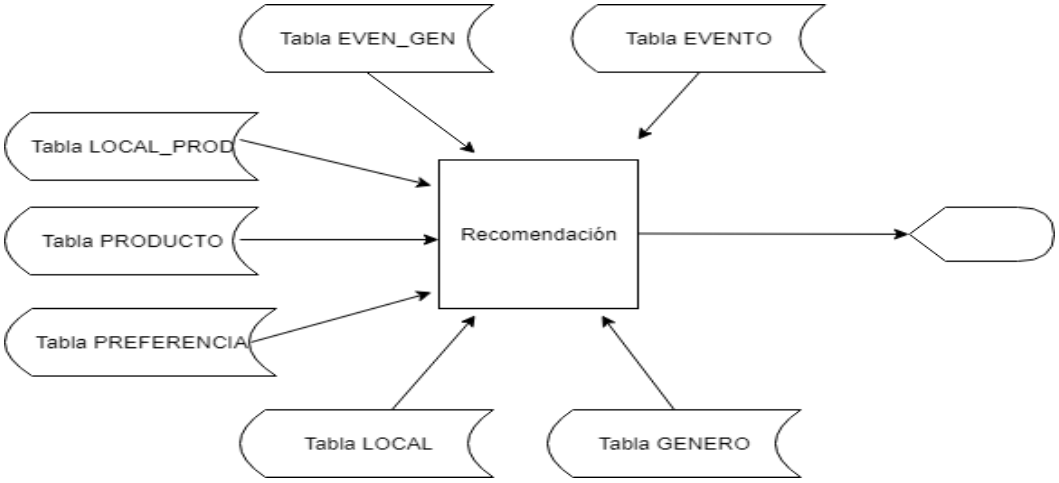
- Si presiona la opción de eliminar tendrá dos opciones:
  - SI: elimina el gusto seleccionado de la tabla PREFERENCIA que está relacionado al contacto seleccionado anteriormente.
  - NO: la aplicación quita el mensaje de eliminar.

➢ Código fuente: Anexo 6 pág. 77, Anexo 7 pág. 79 y Anexo 8 pág. 81.

### 3.4.4 Recomendación

➢ Objetivo: Muestra por pantalla a través de un mapa las ubicaciones de locales o eventos.

➢ Diagrama de bloque:



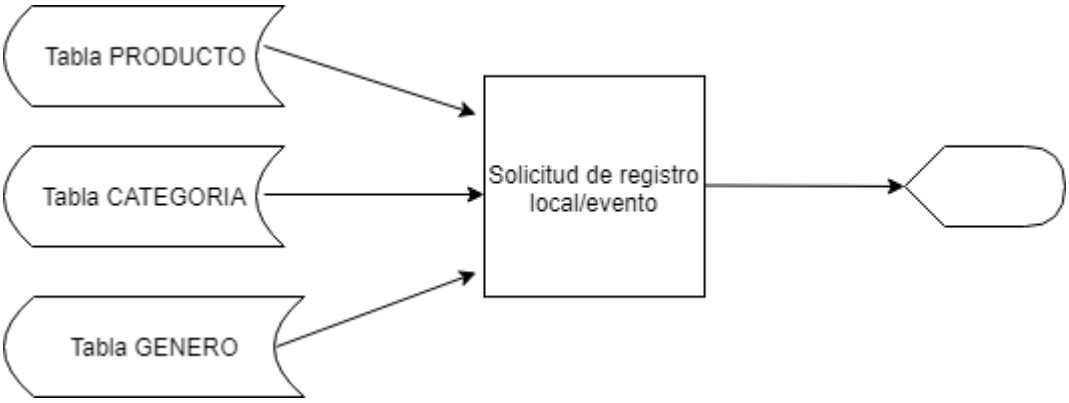
- Reglas de proceso:
  - Si en la fig. 3.11 el usuario presiona el gusto, entonces la aplicación abrirá la pantalla de la fig. 3.14 la cuál primero obtiene el gusto seleccionado de la tabla PREFERENCIA y su categoría, dependiendo de la categoría existen dos caminos, si pertenece a comida el gusto se compara con los productos de la tabla PRODUCTO y si es cinematográfico o musical se compara con los generos de la tabla GENERO y con las descripciones y nombres de eventos de la tabla EVENTO.
  - Si se encuentra algun producto de igual nombre al gusto seleccionado, se accede a la tabla LOCAL_PROD para saber los locales que ofrecen ese producto, hecho esto se accede a la tabla LOCAL para obtener los detalles de los locales y mostrarlos por pantalla.
  - Si se encuentra algún género de igual nombre al gusto seleccionado, se accede a la tabla EVEN_GEN para saber los eventos que sean de ese género. En ese caso se obtienen los detalles de los locales que cumplan con ese género. Por último si el nombre o descripción de un evento son iguales al nombre del gusto se accede directamente a la tabla EVENTO para obtener sus detalles y mostrarlos por pantalla.

Fuente: Elaboración propia
Fig. 3.14 Pantalla de recomendación

- Código fuente: Anexo 9 pág. 82.

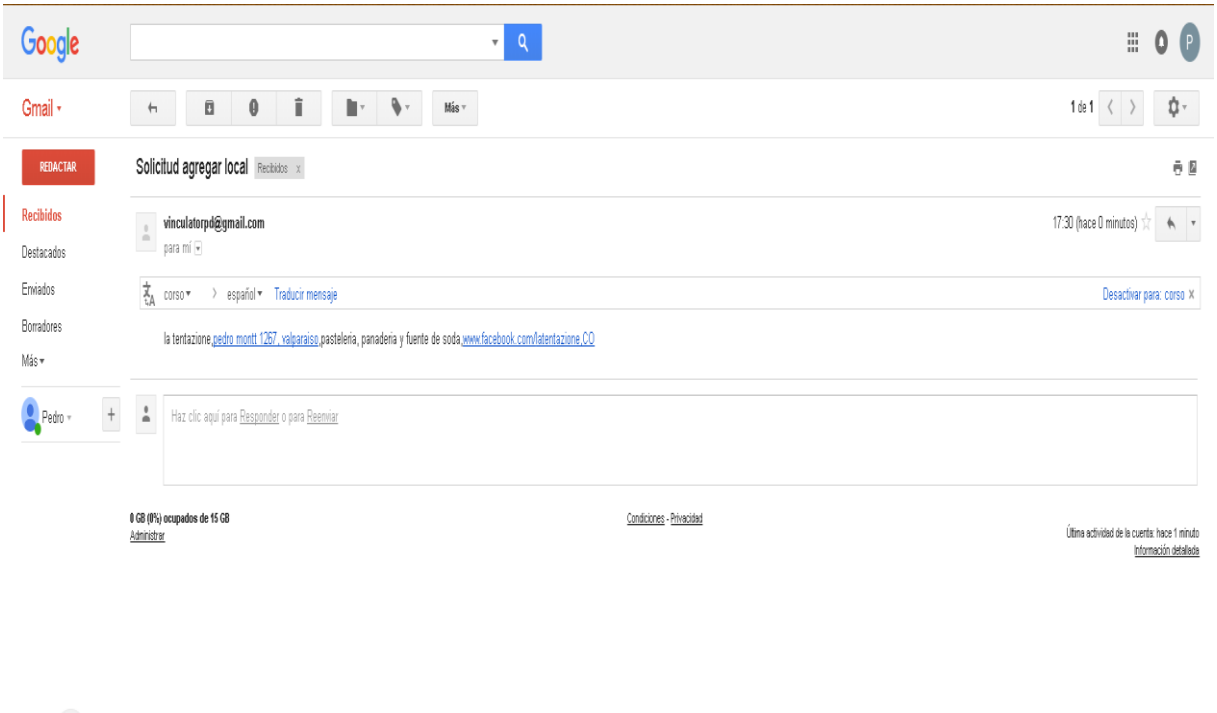
3.4.5 Solicitud de registro local/evento

- Objetivo: Envía información de un local, los productos que este ofrece o evento al correo del administrador para su revisión.
- Diagrama de bloque:




- Reglas de proceso:
  - Al presionar el usuario en alguna de las 3 opciones de la fig. 3.5 dependiendo de la seleccionada se abrirán tablas distintas para cargar datos, en el caso de presionar AGREGAR LOCAL no se cargarán datos de ningun lugar, pero en el caso de AGREGAR PRODUCTO, se abre la tabla PRODUCTO y en el caso de presionar AGREGAR EVENTO, se abre la tabla GÉNERO y CATEGORÍA.
  - Luego de cargar los datos correspondientes se pide al usuario que complete todos los campos que se requieren (fig. 3.15), luego al presionar el botón de enviar, la aplicación envía a través de un correo electrónico toda la información al administrador de la base de datos (fig. 3.16).
  - En el caso de solicitar que un evento se registre, el usuario obligatoriamente no puede seleccionar una fecha y hora anterior a la actual.

Fuente: Elaboración propia
Fig. 3.15 Solicitud agregar local



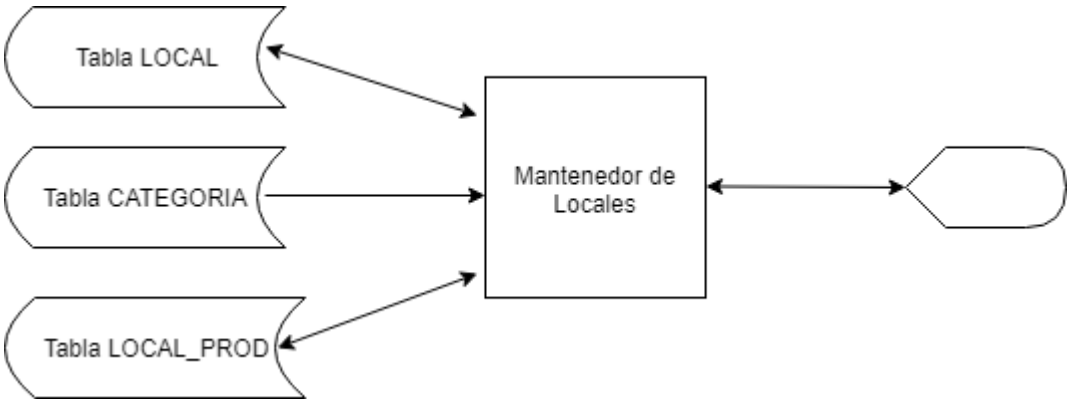

Fuente: https://mail.google.com/
Fig. 3.16 Mensaje de solicitud de registro

- Código fuente: Anexo 10 pág. 92.

3.4.6 Mantenedor de local

- Objetivo: Permite Agregar, Listar, Modificar o Eliminar la información de un local almacenado en la tabla “LOCAL”.

- Diagrama de bloque:




- Reglas de proceso:

Agregar local:

- El administrador ingresa por pantalla el nombre, dirección, categoría, contacto, descripción, latitud y longitud del local. Luego el presionar “Aceptar” se accesa a la tabla “LOCAL” según el nombre del local y la dirección de esta.
- Luego se valida lo ingresado anteriormente; si este existe se imprime por pantalla -“Error, Local Duplicado.”-, si los campos están vacios, -“No puede dejar campos en blanco.”-. Si no ocurre ninguno de estos casos, es decir cumple con los requisitos, entonces se genera un nuevo código de local de la tabla LOCAL, se obtiene un código de categoría de la tabla CATEGORIA y se ingresa a la tabla LOCAL un nuevo registro con los campos llenados por el administrador en pantalla.
- Finalmente un mensaje de “Local Agregado.” Indicará que la operación fue exitosa.

Modificar local:

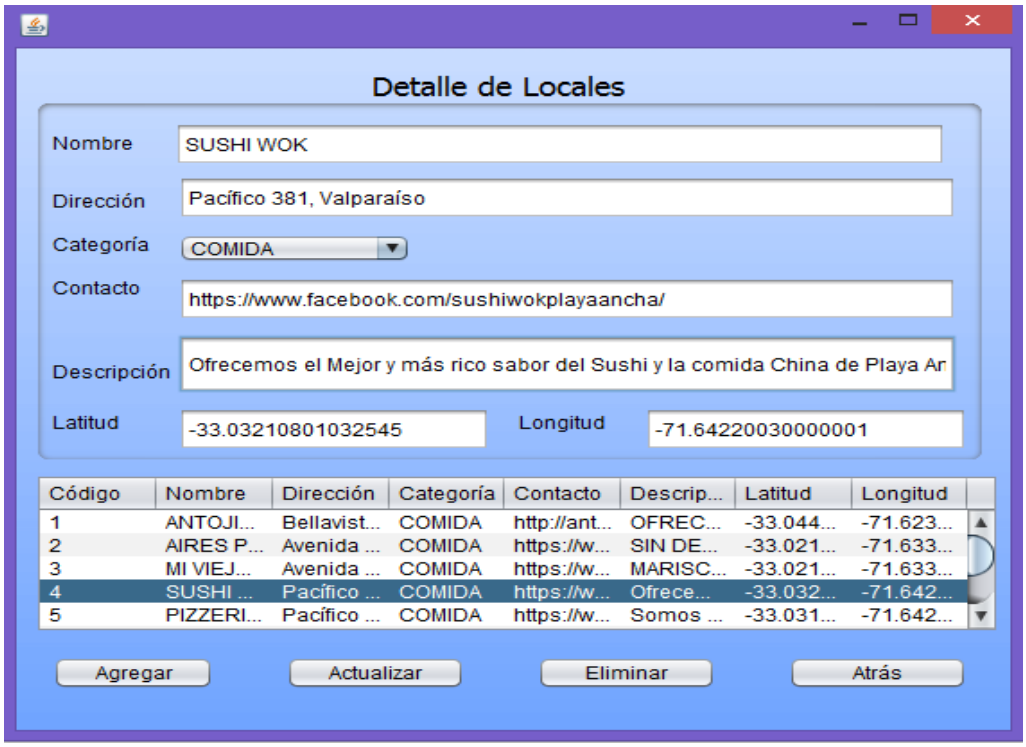
- Se selecciona un local por pantalla, obteniendo nombre, dirección, categoría, contacto, descripción, latitud y longitud de éste.

- Luego se podrá sobrescribir por pantalla los campos que desea modificar.
- Al presionar el botón “Actualizar” se accesará a la tabla LOCAL según nombre y dirección ingresados. Si este existe y tiene un código diferente al registro seleccionado, se imprime por pantalla -“Error, Local Duplicado.”-, si los campos están vacios, -“No puede dejar campos en blanco.”-. Sino ocurre ninguno de estos casos, es decir cumple con los requisitos, entonces se posicionará en el local seleccionado en la tabla LOCAL y se modificará el registro con los campos llenados por el administrador en pantalla.
- Finalmente un mensaje de “Local Actualizado.” Indicará que la operación fue exitosa.

Eliminar local:

- Se selecciona un local del listado por pantalla, obteniendo el código del local para accesar a la tabla LOCAL.
- Al presionar “Eliminar” se eliminará el registro según código local en la tabla LOCAL y se eliminara además de la tabla LOCAL_PROD.
- Un mensaje de “Local Eliminado” indicará que la operación fue exitosa.

➢ Diseño de Pantalla

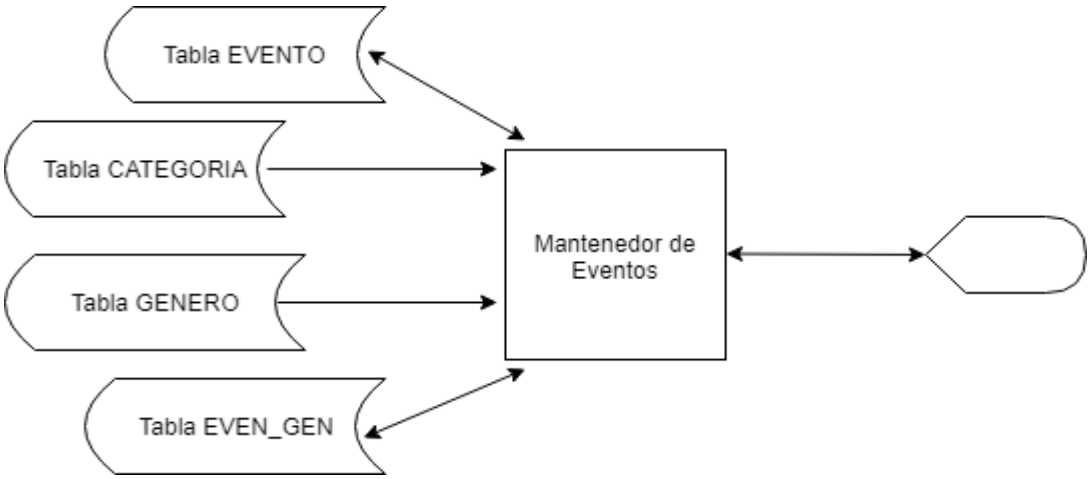



Fuente: Elaboración propia
Fig. 3.17 Mantenedor de locales

➢ Codigo Fuente Anexo 11 Pág. 94.

3.4.7 Mantenedor de Evento

- Objetivo: Permite Agregar, Listar, Modificar o Eliminar la información de un evento almacenado en la tabla “EVENTO”.
- Diagrama de bloque:




- Reglas de proceso:

Agregar evento:

- El administrador ingresa por pantalla el nombre, dirección, categoría, género(s), contacto, descripción, fecha, latitud y longitud del evento. Luego el presionar “Aceptar” se accesa a la tabla “EVENTO” según el nombre del evento, la dirección y la fecha de éste.
- Luego se valida lo ingresado anteriormente; si este existe se imprime por pantalla -“Error, Evento Duplicado.”-, si los campos están vacios, -“No puede dejar campos en blanco.”-. Sino ocurre ninguno de estos casos, es decir cumple con los requisitos, entonces se genera un nuevo código de evento de la tabla EVENTO, se obtiene un código de categoría de la tabla CATEGORIA, se obtiene un código de género de la tabla GENERO y se ingresa a la tabla EVENTO un nuevo registro con los campos llenados por el administrador en pantalla.
- Posteriormente se graba un la tabla EVEN_GEN un nuevo registro con el código del nuevo evento y el código del genero.
- Finalmente un mensaje de “Evento Agregado.” Indicará que la operación fue exitosa.

Modificar evento:

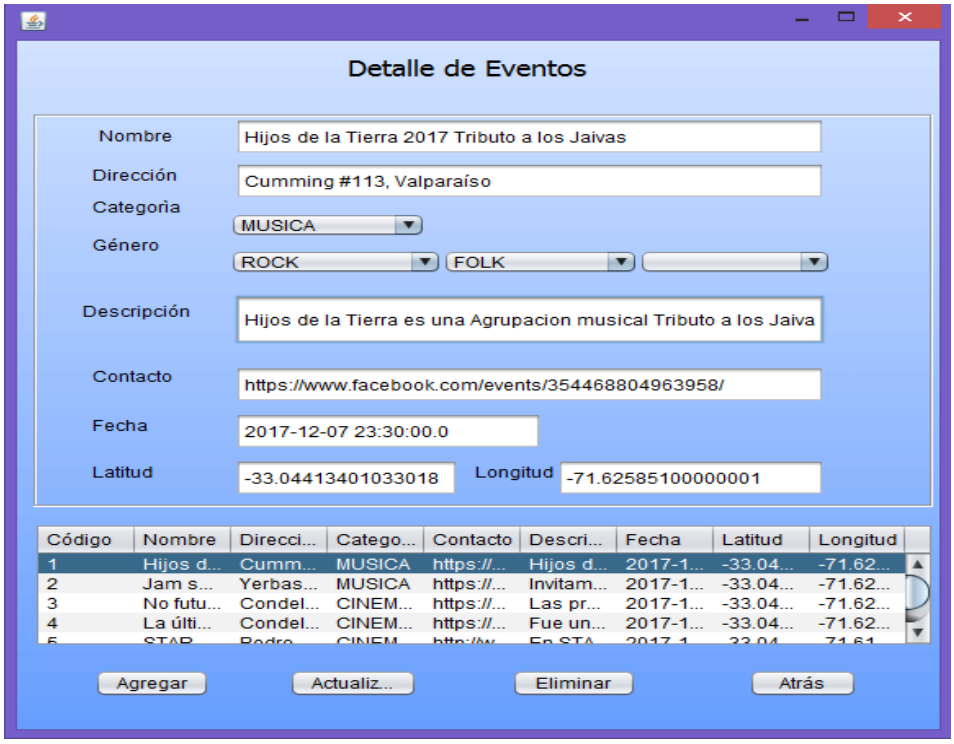
- Se selecciona un evento por pantalla, obteniendo nombre, dirección, categoría, género(s), contacto, descripción, fecha, latitud y longitud del evento.
- Luego se podrá sobrescribir por pantalla los campos que desea modificar.

- Al presionar el botón “Actualizar” se accesará a la tabla EVENTO según nombre, dirección y fecha ingresados. Si este existe y tiene un código diferente al registro seleccionado, se imprime por pantalla -“Error, Evento Duplicado.”-, si los campos están vacios, -“No puede dejar campos en blanco.”-. Si no ocurre ninguno de estos casos, es decir cumple con los requisitos, entonces se posicionará en el evento seleccionado en la tabla EVENTO y se modificará el registro con los campos llenados por el administrador en pantalla.
- Finalmente un mensaje de “Evento Actualizado.” Indicará que la operación fue exitosa.

Eliminar local:

- Se selecciona un evento del listado por pantalla, obteniendo el código del evento para accesar a la tabla EVENTO y EVEN_GEN.
- Luego al presionar “Eliminar” se eliminará el registro según código evento en la tabla EVENTO y se eliminara además de la tabla EVEN_GEN.
- Un mensaje de “Evento Eliminado” indicará que la operación fue exitosa.

➢ Diseño de Pantalla

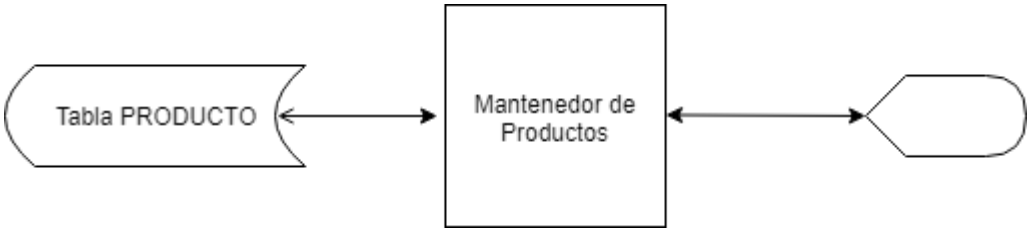



Fuente: Elaboración propia
Fig. 3.18 Mantenedor de eventos

➢ Codigo Fuente Anexo 12 Pág. 98

3.4.8 Mantenedor de producto

- Objetivo: Permite Agregar, Listar, Modificar o Eliminar la información de un producto almacenado en la tabla “PRODUCTO”.
- Diagrama de bloque:




- Reglas de proceso:

Agregar producto:

- Se ingresa por pantalla el nombre del nuevo producto y se presiona el botón “Agregar”.
- Se accesa a la tabla PRODUCTO según el nombre ingresado.
- Si el producto ya existe se muestra un mensaje por pantalla: “Producto Duplicado.”. Si el nombre se encuentra en blanco, se imprime por pantalla “No puede dejar el campo en blanco.”. Si no ocurre ninguno de estos casos, es decir cumple con las validaciones, entonces se genera un nuevo código de producto en la tabla PRODUCTO, y se ingresa un nuevo registro con el nombre ingresado por el administrador en pantalla.
- Finalmente un mensaje de “Producto Agregado.” Indicará que la operación fue exitosa

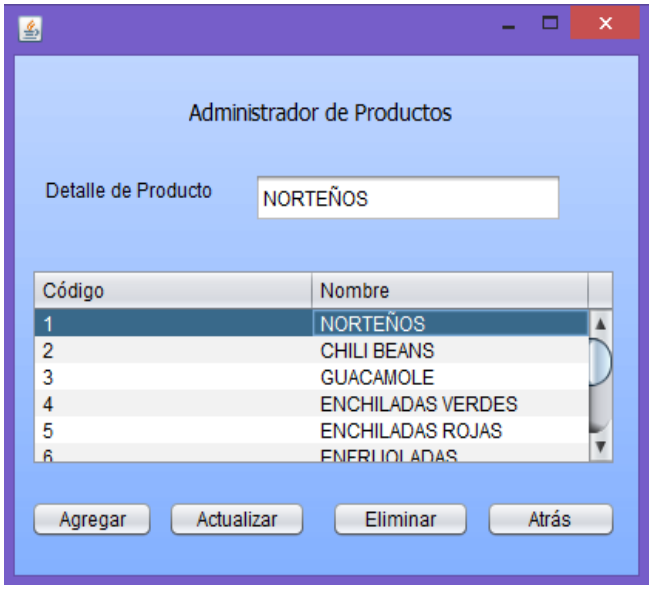
Modificar producto:

- Seleccionar producto por pantalla, obteniendo nombre.
- Luego se podrá sobrescribir por pantalla los campos que desea modificar.
- Al presionar el botón “Actualizar” se accesara a la tabla PRODUCTO según nombre. Si este existe y tiene un código diferente al registro seleccionado, se imprime por pantalla -“Error, Producto Duplicado.”-, si los campos están vacios, -“No puede dejar campos en blanco.”-. Si no ocurre ninguno de estos casos, es decir cumple con las validaciones, entonces se posicionará en el registro seleccionado de la tabla PRODUCTO y se modificara el registro con los campos llenados por el administrador en pantalla.
- Finalmente un mensaje de “Producto Actualizado.” Indicará que la operación fue exitosa.

Eliminar local:

- Se selecciona un producto del listado por pantalla, obteniendo el código de producto para accesar a la tabla PRODUCTO.
- Luego al presionar “Eliminar” se eliminará el registro según código producto en la tabla PRODUCTO.
- Un mensaje de “Producto Eliminado” indicará que la operación fue exitosa.

➢ Diseño de Pantalla



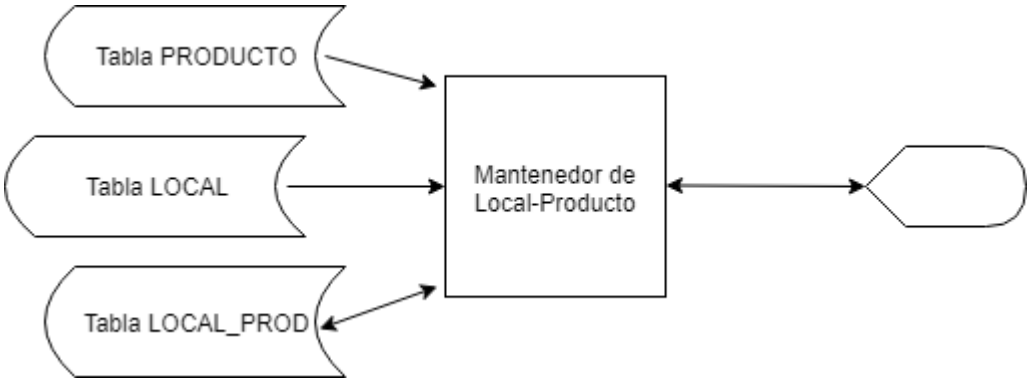

Fuente: Elaboración propia
Fig. 3.19 Mantenedor de productos

> Codigo Fuente Anexo 13 Pág. 102

3.4.9 Mantenedor de Local-Producto

- Objetivo: Permite Agregar o Eliminar un producto asociado a un local en la tabla "LOCAL_PROD".

- Diagrama de bloque:




- Reglas de proceso:

Agregar producto-local:

- Por pantalla se carga la tabla "LOCAL" y "PRODUCTO" y se muestra en un listado a traves de un combo box.
- El administrador puede seleccionar un local, lo cual cargará y mostrará en una lista los productos asociados a ese local.
- Ademas puede elegir un nuevo producto para asociar a un local seleccionado previamente.
- Luego de escoger un local y un producto, al presionar "Agregar" se valida si el producto ya existe en el local en la tabla LOCAL_PROD.
- Posteriormente se graba un nuevo registro en la tabla LOCAL_PROD con el código del local y el código del producto.
- Finalmente un mensaje de "Producto Agregado." Indicará que la operación fue exitosa.

Eliminar producto-local:

- Se selecciona un local y un producto del listado por pantalla, obteniendo el código de producto para accesar a la tabla LOCAL_PROD.
- Luego al presionar “Eliminar” se eliminará el registro según código producto y código local en la tabla LOCAL_PROD.
- Un mensaje de “Producto Eliminado” indicará que la operación fue exitosa.

➢ Diseño de pantalla

Fuente: Elaboración propia
Fig. 3.20 Mantenedor de relación local producto

➢ Codigo Fuente Anexo 14 Pág. 104

## CONCLUSIONES

El tener como proyecto de titulación el crear una aplicación móvil es realmente algo complicado, ya que si bien durante la carrera se prepara para programar y adecuarse a distintos ambientes de trabajo, en ninguna asignatura se muestran contenidos para desarrollar una aplicación móvil, esto causa que se tuviese que invertir una gran cantidad de tiempo en investigar desde cero todos los estándares de diseño. Para esta investigación se usó mayoritariamente videos de youtube y un curso web llamado “Curso Definitivo de Android” de la página platzi.com.

La única facilidad con la que se contó fue que se utilizó el lenguaje de programación JAVA, el cuál fue visto durante algunas asignaturas.

Aún con toda la falta de información inicial y las pocas esperanzas de lograr un proyecto bueno, se logró crear la aplicación de manera satisfactoria, aunque todavía tiene muchos puntos que pueden mejorar,por ejemplo el añadir fotos de perfil para cada contacto y asi poder hacer más atractiva la interfaz, también mejorar la interfaz gráfica de la parte de ENCUESTA de la app y por último conseguir una base de datos con información de locales, eventos y productos realmente grande y que mínimo abarque varias ciudades de chile, por supuesto esto también quiere decir que se debe contar con un servidor al que se debe ser capaz de acceder desde cualquier red y no sólo desde la red local como está configurado actualmente.

**BIBLIOGRAFÍA**


- Henao, C. [Christian Henao]. (2017, Febrero 17). *Curso  Android desde 0* [Lista de reproducción]. Recuperado de https://www.youtube.com/watch?v=2b7bXOiOA38&list=PLAg6Lv5BbjjdvIcLQdVg4ROZnfuuQcqXB
- Gómez, J. [pildorasinformaticas]. (2016, Febrero 23). *Curso de Android con Android Studio* [Lista de reproducción]. Recuperado de https://www.youtube.com/watch?v=pdYkmCcQFd8&list=PLU8oAlHdN5Bkn-KS1sRFlSEnXXcAtAJ9P
- Méndez, M. [codigofacilito]. (2016, Septiembre 14). *Colocar Google Maps en tu aplicación de Android – Tutorial* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=nybRpw2H7gg
- Méndez, M. [codigofacilito]. (2016, Octubre 4). *Enviar correos con JAVA Mail desde Android* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=pXl5CifFlM0
- Salgado, A. [Platzi]. (2017, Febrero). *Curso Definitivo de Android* [Lista de reproducción]. Recuperado de https://platzi.com/clases/android/
- Salazar, A. [Andrés Salazar]. (2017, Febrero 16). *WampServer 3 | Configuración para una intranet (red local) 2017* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=1XqjmeUoE-w
- Jorge. [codigofacilito]. (2011, Octubre 15). *Curso de Java* [Lista de reproducción]. Recuperado de https://www.youtube.com/watch?v=Z0F7sJaOQtw&list=PL602060AB32FC864B
- [Colores para el diseño de la aplicación]. Recuperado de https://www.materialpalette.com/
- [Herramienta para obtener las coordenadas geográficas de algun lugar en mapa de google Maps]. Recuperado de http://www.bufa.es/google-maps-latitud-longitud/

# ANEXOS

## Anexo 1: código fuente del menú principal

```
public class MainActivity extends AppCompatActivity
      implements NavigationView.OnNavigationItemSelectedListener,
Se_Parte_Fragment.OnFragmentInteractionListener,Inicio.OnFragmentInteractionListener {
      BDHelper conn;
      Toolbar toolbar;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      conn = new BDHelper(this,"bd_vinculator",null,1);// SE ABRE LA CONEXION A LA BASE
DE DATOS LOCAL
      toolbar = (Toolbar) findViewById(R.id.toolbar);// SE CREA UNA INSTANCIA QUE SE
VINCULA CON UN ELEMENTO GRAFICO DE TIPO TOOLBAR
      setSupportActionBar(toolbar);// SE CONFIGURA EL TOOLBAR PARA QUE SOPORTE
CIERTOS ELEMENTOS EN SISTEMAS ANDROIDS MAS ANTIGUOS
      toolbar.setTitle("CONTACTOS");//SE CONFIGURA EL TITULO DEL TOOLBAR
      DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
      ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
      drawer.setDrawerListener(toggle);
      toggle.syncState();
      Fragment fragment = new Inicio();// SE CREA UNA INSTANCIA QUE VINCULA LA
PANTALLA DE INICIO
getSupportFragmentManager().beginTransaction().add(R.id.content_main,fragment).commi
t();// SE CARGA EN LA PANTALLA PRINCIPAL LA PANTALLA DE INICIO QUE ESTA EN LA
INSTANCIA FRAGMENT
      NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);// SE
CREA UNA INSTANCIA VINCULADA A UN NAVEGADOR GRAFICO
      navigationView.setNavigationItemSelectedListener(this);
      navigationView.setItemIconTintList(null);    }

   @Override
   public void onBackPressed() {// CUANDO SE PRESIONA EL BOTON DE RETROCESO OCURRE
ALGO
      DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
      if (drawer.isDrawerOpen(GravityCompat.START)) {// SI EL NAVEGADOR ESTA ABIERTO
ENTONCES SE CIERRA
         drawer.closeDrawer(GravityCompat.START);
      } else {       }    }
   @SuppressWarnings("StatementWithEmptyBody")
```

```
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();// SE GUARDA EL ID DEL ITEM QUE EL USUARIO SELECCIONO EN
EL NAVEGADOR
        Fragment mifragment = null;
        boolean fragmentseleccionado = false;
        if (id == R.id.inicio) {// SI EL ID CORRESPONDE AL BOTON QUE LLEVA A LA PANTALLA
INICIO
            mifragment = new Inicio();// LA INSTANCIA FRAGMENT SE VINCULARA A LA
PANTALLA INICIO
            fragmentseleccionado = true;
            toolbar.setTitle("CONTACTOS");// EL TITULO DE LA TOOLBAR CAMBIA
        } else if (id == R.id.se_parte) {// SI EL ID CORRESPONDE AL BOTON QUE LLEVA A LA
SECCION DONDE EL USUARIO APORTA CON LOCALES O EVENTOS
            mifragment = new Se_Parte_Fragment();// LA INSTANCIA FRAGMENT SE
VINCULARA A LA PANTALLA SE_PARTE_FRAGMENT
            fragmentseleccionado = true;
            toolbar.setTitle("AYUDANOS A CRECER");// EL TITULO DE LA TOOLBAR CAMBIA
}
        if(fragmentseleccionado == true){// SI ALGUN ELEMENTO FUE SELECCIONADO
getSupportFragmentManager().beginTransaction().replace(R.id.content_main,mifragment).c
ommit();// SE CARGA EN PANTALLA EL FRAGMENTO GRAFICO DEL BOTON
CORRESPONDIENTE        }
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);// SE CIERRA EL NAVEGADOR
        return true    }
    public void agregarcontacto(View vista){
        Intent i = new Intent(this,AgregarContacto.class);// CREA UNA INSTANCIA QUE LLEVA A
LA PANTALLA PARA AGREGAR CONTACTOS
        startActivity(i);// SE EJECUTA LA INSTANCIA ANTERIOR    }
    @Override
    public void onFragmentInteraction(Uri uri) {    }
    public void iraAgregarLocal(View v){
        Intent i = new Intent(this,Agregar_Local.class);// CREA UNA INSTANCIA QUE LLEVA A LA
PANTALLA PARA AGREGAR UN LOCAL
        startActivity(i);// SE EJECUTA LA INSTANCIA ANTERIOR    }
    public void iraAgregarProd(View v){
        Intent i = new Intent(this,Agregar_Producto.class);// CREA UNA INSTANCIA QUE LLEVA
A LA PANTALLA PARA AGREGAR UN PRODUCTO
        startActivity(i);// SE EJECUTA LA INSTANCIA ANTERIOR    }
    public void iraAgregarEvento(View v){
        Intent i = new Intent(this,Agregar_Evento.class);// CREA UNA INSTANCIA QUE LLEVA A
LA PANTALLA PARA AGREGAR EVENTO
        startActivity(i);// SE EJECUTA LA INSTANCIA ANTERIOR    }}
```

Anexo 2: código fuente de la pantalla de contactos

```
public class Inicio extends Fragment implements SearchView.OnQueryTextListener {
  // DECLARACION DE VARIABLES
  private OnFragmentInteractionListener mListener;
  RecyclerView recyclerView;
  ArrayList<Contacto> list_contactos;
  BDHelper conn;
  String numero_telefono;
  Adaptador_Contactos adapter;
  public Inicio() {   }
  public static Inicio newInstance(String param1, String param2) {
    Inicio fragment = new Inicio();
    Bundle args = new Bundle();
    fragment.setArguments(args);
    return fragment;   }
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {      }   }
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                    Bundle savedInstanceState) {
    View vista = inflater.inflate(R.layout.fragment_inicio, container, false);
    list_contactos = new ArrayList<>(); //ARREGLO DE TIPO CONTACTO
    recyclerView = (RecyclerView) vista.findViewById(R.id.recycler_contactos);// SE
VINCULA LA VARIABLE RECYCLERVIEW A UN RECYCLERVIEW GRAFICO
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    conn = new BDHelper(getContext(),"bd_vinculator",null,1);// SE ABRE LA CONEXION A
LA BASE DE DATOS LOCAL
    adapter = new Adaptador_Contactos(list_contactos);// SE INSTANCIA UN ADAPTADOR
QUE TENDRA COMO PARAMETRO LA LISTA DE TIPO CONTACTO
    adapter.setOnclickListener(new View.OnClickListener() { // CUANDO SE HAGA CLICK
SOBRE LA LISTA DE TIPO CONTACTO SE EJECUTARA EL SIGUIENTE CODIGO
      @Override
      public void onClick(View v) {
        Intent i = new Intent(getContext(),Pantalla_Contacto.class); // SE CREA UNA
INSTANCIA QUE SIRVE PARA IR A LA VENTANA PANTALLA_CONTACTO
        numero_telefono =
list_contactos.get(recyclerView.getChildAdapterPosition(v)).getNum_telef();// SE GUARDA
EL NUMERO DE TELEFONO DEL ELEMENTO PRESIONADO
        SharedPreferences datos =
PreferenceManager.getDefaultSharedPreferences(getContext());// SE CREA LA VARIABLE
DATOS QUE SIRVE PARA PASAR DATOS DE ESTA CLASE A OTRA
```

```
            SharedPreferences.Editor editor = datos.edit();// SE CREA LA INSTANCIA PARA
PODER MODIFICAR EL CONTENIDO DE LA VARIABLE O PAQUETE CREADO ANTERIORMENTE
            editor.putString("numero_telefonito",numero_telefono);// SE AGREGA COMO
CONTENIDO EL NUMERO Y CON SU NOMBRE CLAVE "NUMERO_TELEFONITO"
            editor.apply();// SE APLICAN LOS CAMBIOS
            startActivity(i);// SE INICIA LA PANTALLA_CONTACTO
        }       });
    recyclerView.setAdapter(adapter);// EL RECYCLERVIEW SE ADAPTA AL CONTENIDO DEL
ADAPTER(QUE TIENE LA LISTA DE CONTACTOS)
    setHasOptionsMenu(true);
    return vista;    }
  public void onButtonPressed(Uri uri) {
    if (mListener != null) {
        mListener.onFragmentInteraction(uri);       }    }
  @Override
  public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnFragmentInteractionListener) {
        mListener = (OnFragmentInteractionListener) context;
    } else {
        throw new RuntimeException(context.toString()
                + " must implement OnFragmentInteractionListener");     }    }
  @Override
  public void onDetach() {
    super.onDetach();
    mListener = null;    }
  @Override
  public boolean onQueryTextSubmit(String query) {
    return false;    }
  @Override
  public boolean onQueryTextChange(String newText) {//FUNCION QUE BUSCA SI EXISTE
UNA CADENA DE CARACTERES DENTRO DE LA LISTA DE CONTACTOS
    newText = newText.toLowerCase(); // EL TEXTO PASADO COMO PARAMETRO (EN ESTE
CASO EL NOMBRE DEL CONTACTO) SE CONVIERTE EN MINUSCULA
    ArrayList<Contacto> newlist = new ArrayList<>();// SE INSTANCIA OTRA LISTA DE TIPO
CONTACTO
    for(Contacto contactito: list_contactos){// SE EJECUTA CADA HASTA QUE NO EXISTAN
MAS CONTACTOS EN LA LISTA
        String nombre = contactito.getNom().toLowerCase(); // SE CREA UNA VARIABLE QUE
CONTIENE EL NOMBRE DEL CONTACTO QUE ESTA REVISANDO ACTUALMENTE
        if(nombre.contains(newText)){ // SE  COMPARA SI EL NOMBRE QUE ESTA REVISANDO
CONTIENE LOS CARACTERES QUE SE PASARON COMO PARAMETRO
            newlist.add(contactito); // DE SER ASI SE AÑADE ESE CONTACTO A LA NUEVA LISTA
        }       }
    adapter.setFilter(newlist); // LA NUEVA LISTA ES PASADA COMO PARAMETROS AL
```

```
ADAPTADOR.
        return true;    }
    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(Uri uri);    }
    private void consultarContactos() {// consulta todos los contactos de la base de datos y los
agrega a la lista de contactos
        SQLiteDatabase db = conn.getReadableDatabase();// SE CREA UNA INSTANCIA DE TIPO
BASE DE DATOS QUE SE ABRE EN MODO DE LECTURA
        list_contactos.clear();// SE LIMPIA LA LISTA DE CONTACTOS ACTUAL
        Contacto contacto = null;
        Cursor cursor = db.rawQuery("SELECT "+ Utilidades.Campo_nom+",
"+Utilidades.Campo_telef+" FROM "+Utilidades.Tabla_Contacto+" ORDER BY nom",null); //
SE CREA UN CURSOR QUE EJECUTA UNA SENTENCIA SQL QUE BUSCA TODOS LOS
CONTACTOS Y LOS ORDENA POR NOMBRE
        while(cursor.moveToNext()){// SE MUEVE EL CURSOR HASTA QUE NO TENGA MAS
ELEMENTOS COMO RESULTADO DE LA CONSULTA SQL ANTERIOR
            contacto = new Contacto();// se crea una instancia de tipo contacto
            contacto.setNom(cursor.getString(0)); // en el campo nombre del contacto se le
agrega el primer campo del cursor en la posicion actual
            contacto.setNum_telef(cursor.getString(1));//en el campo numero de telefono del
contacto se le agrega el segundo campo del cursor en la posicion actual
            list_contactos.add(contacto);// se agrega el contacto a la lista de contactos        }
        db.close();// se cierra la base de datos    }
    public void onResume(){// cada vez que se recargue la pantalla inicio se ejecutara lo
siguiente
        super.onResume();
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
        consultarContactos();
        recyclerView.setAdapter(adapter);// se pasa como parametro el adaptador con la lista
de todos los contactos de la base de datos al recyclerview grafico    }
    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) { // se crea el menu
que sirve para buscar un contacto en especifico
        inflater.inflate(R.menu.buscar, menu);
        MenuItem menucito = menu.findItem(R.id.search_buscar);
        SearchView searchView = (SearchView) MenuItemCompat.getActionView(menucito);
        searchView.setOnQueryTextListener(this);    }}
```

Anexo 3: código fuente agregar contacto

```
public class AgregarContacto extends AppCompatActivity {
  //DECLARACION DE VARIABLES
  private int año;
  private int mes;
  private int dia;
  private int añoactual;
  private int mesactual;
  private int diaactual;
  private static DatePickerDialog.OnDateSetListener selectorfecha;
  private static final int Tipo_dialogo = 0;
  Button boton_agregar;
  Toolbar barra;
  EditText texto_nombre,texto_num,texto_fecha,text_hora;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.agregar_contacto2);
    barra = (Toolbar) findViewById(R.id.barra_agregarcontacto);// SE CREA UNA INSTANCIA
QUE VINCULA A UN COMPONENTE GRAFICO DE TIPO TOOLBAR
    setSupportActionBar(barra);// SE CONFIGURA EL TOOLBAR PARA SOPORTAR
ELEMENTOS EN SISTEMAS OPERATIVOS ANTIGUOS
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);//SE HABILITA LA OPCION DE
REGRESO EN LA TOOLBAR
    getSupportActionBar().setDisplayShowHomeEnabled(true);//SE HABILITA LA FUNCION
DE REGRESO EN LA TOOLBAR
    boton_agregar = (Button) findViewById(R.id.BTN_AGREGAR);// SE CREA UNA
INSTANCIA VINCULADA A UN COMPONENTE GRAFICO DE TIPO BUTTON QUE AGREGA LOS
DATOS A LA BASE DE DATOS LOCAL
    texto_nombre = (EditText) findViewById(R.id.TXT_NOMBRE_C);// SE CREA UNA
INSTANCIA VINCULADA A UN COMPONENTE GRAFICO DE TIPO EDITTEXT  QUE ALMACENA EL
NOMBRE DEL CONTACTO
    texto_num = (EditText) findViewById(R.id.TXT_NUM_TELEF);// SE CREA UNA INSTANCIA
VINCULADA A UN COMPONENTE GRAFICO DE TIPO EDITTEXT  QUE ALMACENA EL NUMERO
DE TELEFONO DEL CONTACTO
    texto_fecha = (EditText) findViewById(R.id.TXT_FECHA_C);// SE CREA UNA INSTANCIA
VINCULADA A UN COMPONENTE GRAFICO DE TIPO EDITTEXT  QUE ALMACENA LA FECHA DE
NACIMIENTO DEL CONTACTO
    Calendar calendario = Calendar.getInstance();// SE CREA UNA INSTANCIA DE TIPO
CALENDARIO
    año = calendario.get(Calendar.YEAR);//SE GUARDA EL AÑO ACTUAL
    mes = calendario.get(Calendar.MONTH);//SE GUARDA EL MES ACTUAL
    dia = calendario.get(Calendar.DAY_OF_MONTH);//SE GUARDA EL DIA ACTUAL
```

```
    añoactual = calendario.get(Calendar.YEAR);//SE GUARDA EL AÑO ACTUAL
    mesactual = calendario.get(Calendar.MONTH);//SE GUARDA EL MES ACTUAL
    diaactual = calendario.get(Calendar.DAY_OF_MONTH);//SE GUARDA EL DIA ACTUAL
    selectorfecha = new DatePickerDialog.OnDateSetListener() { //NOS MUESTRA UN
CALENDARIO CON LA FECHA ACTUAL SELECCIONADA
        @Override
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
            año = year;
            mes = month;
            dia = dayOfMonth;
            mostrarFecha();          }       };
    barra.setOnClickListener(new View.OnClickListener() {//SI SE PRESIONA SOBRE EL
BOTON DE REGRESO
        @Override
        public void onClick(View v) {
            finish();// SE CIERRA LA PANTALLA ACTUAL          }       });
    final BDHelper mDbHelper = new BDHelper(this,"bd_vinculator",null,1);// SE CREA UNA
INSTANCIA QUE CONECTA LA BASE DE DATOS LOCAL
    boton_agregar.setOnClickListener(new View.OnClickListener() {//CUANDO SE PRESIONA
EL BOTON AGREGAR
        @Override
        public void onClick(View v) {
            SQLiteDatabase db = mDbHelper.getWritableDatabase();// SE ABRE LA CONEXION
EN MODO DE ESCRITURA
            if ((texto_nombre.getText().toString().isEmpty()) ||
(texto_num.getText().toString().isEmpty()) ||
(texto_fecha.getText().toString().isEmpty())){//SI ALGUN CAMPO ESTA VACIO
                Toast.makeText(getApplicationContext(),"TODOS LOS CAMPOS SON
OBLIGATORIOS, VUELVA A INTENTARLO.",Toast.LENGTH_LONG).show();//SE MUESTRA
ESTE MENSAJE            }
            else if((año == añoactual)){//SI EL AÑO SELECCIONADO ES IGUAL AL AÑO ACTUAL
                if (mes == mesactual){// SI EL MES SELECCIONADO ES IGUAL AL MES ACTUAL
                    if (dia <= diaactual){// SI EL DIA SELECCIONADO ES MENOR O IGUAL AL DIA
ACTUAL
                        try{//SI LA CONEXION SE LLEVA A CABO CON EXITO
                            String sql = "INSERT INTO "+ Utilidades.Tabla_Contacto+"
("+Utilidades.Campo_telef+","+Utilidades.Campo_nom+","+Utilidades.Campo_fecha_nac+"
)
VALUES('"+texto_num.getText().toString()+"','"+texto_nombre.getText().toString()+"','"+tex
to_fecha.getText().toString()+"')";
                            //SE GUARDA LA SENTENCIA PARA INSERTAR EL NUEVO CONTACTO
                            db.execSQL(sql);// SE EJECUTA LA SENTENCIA SQL
                            Toast.makeText(getApplicationContext(),"SU CONTACTO FUE AGREGADO
CORRECTAMENTE",Toast.LENGTH_LONG).show();//SE MUESTRA ESTE MENSAJE
                            db.close();// SE CIERRA LA BASE DE DATOS
```

```
                    texto_nombre.setText("");// SE VACIA EL NOMBRE
                    texto_fecha.setText("");// SE VACIA LA FECHA
                    texto_num.setText("");// SE VACIA EL NUMERO                    }
                  catch (Exception e){
                    Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON
ESTE NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
MENSAJE
                    db.close();                    }                    }
                else{
                  Toast.makeText(getApplicationContext(),"LA FECHA NO PUEDE SER ESA,
VUELVA A INTENTARLO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
MENSAJE                    }            }
              else if (mes < mesactual){//SI EL MES SELECCIONADO ES MENOR AL MES ACTUAL
                try{
                  String sql = "INSERT INTO "+ Utilidades.Tabla_Contacto+"
("+Utilidades.Campo_telef+","+Utilidades.Campo_nom+","+Utilidades.Campo_fecha_nac+"
)
VALUES('"+texto_num.getText().toString()+"','"+texto_nombre.getText().toString()+"','"+tex
to_fecha.getText().toString()+"')";
                  db.execSQL(sql);
                  Toast.makeText(getApplicationContext(),"SU CONTACTO FUE AGREGADO
CORRECTAMENTE",Toast.LENGTH_LONG).show();
                  db.close();
                  texto_nombre.setText("");
                  texto_fecha.setText("");
                  texto_num.setText("");                    }
                catch (Exception e){
                  Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON
ESTE NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();
                  db.close();                    }
              }else{
                Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON ESTE
NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();                    }
            }else if(año < añoactual){// SI EL AÑO SELECCIONADO ES MENOR AL AÑO ACTUAL
              try{
                String sql = "INSERT INTO "+ Utilidades.Tabla_Contacto+"
("+Utilidades.Campo_telef+","+Utilidades.Campo_nom+","+Utilidades.Campo_fecha_nac+"
)
VALUES('"+texto_num.getText().toString()+"','"+texto_nombre.getText().toString()+"','"+tex
to_fecha.getText().toString()+"')";
                db.execSQL(sql);
                Toast.makeText(getApplicationContext(),"SU CONTACTO FUE AGREGADO
CORRECTAMENTE",Toast.LENGTH_LONG).show();
                db.close();
                texto_nombre.setText("");
```

```
                texto_fecha.setText("");
                texto_num.setText("");                }
            catch (Exception e){
                Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON ESTE
NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();
                db.close();                }            }
        else{
            Toast.makeText(getApplicationContext(),"LA FECHA NO PUEDE SER ESA, VUELVA
A INTENTARLO",Toast.LENGTH_LONG).show();            }        }      });   }
  @Override
  protected Dialog onCreateDialog(int id) {
    switch (id) {
      case 0: return new DatePickerDialog(this,selectorfecha, año,mes,dia);// SE CREA EL
CALENDARIO CON LA FECHA ACTUAL       }
    return null;   }
  public void mostrarCalendario(View v){
    showDialog(Tipo_dialogo);// SE MUESTRA EL CALENDARIO    }
  public void mostrarFecha(){
    texto_fecha.setText(dia+"/"+(mes+1)+"/"+año); // SE AÑADE LA FECHA SELECCIONADA
AL CAMPO FECHA    }}
```

Anexo 4: código fuente información del contacto

```
public class Pantalla_Contacto extends AppCompatActivity {
  //DECLARACION DE VARIABLES
  TextView nombre,numero,fecha;
  String numero1;
  BDHelper conn;
  String numero2;
  Toolbar barra;
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.pantalla_contacto);
    conn = new BDHelper(this,"bd_vinculator",null,1);//SE CREA INSTANCIA QUE CONECTA
LA BASE DE DATOS
    barra = (Toolbar) findViewById(R.id.barra_pantalla_contacto);// SE CREA INSTANCIA
QUE CONECTA CON UN COMPONENTE GRAFICO DE TIPO TOOLBAR
    setSupportActionBar(barra);//SE CONFIGURA EL TOOLBAR PARA SOPORTAR VERSIONES
DE ANDROID ANTERIORES
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);//SE HABILITA LA OPCION DE
REGRESAR EN LA TOOLBAR
    getSupportActionBar().setDisplayShowHomeEnabled(true);//SE MUESTRA
VISUALMENTE UNA FLECHA PARA REGRESAR
    barra.setOnClickListener(new View.OnClickListener() {
      @Override
```

```
        public void onClick(View v) {
            finish();// SI SE PRESIONA LA FLECHA SE CIERRA LA VENTANA ACTUAL          }       });
}
    private void buscar() {
        SQLiteDatabase db = conn.getReadableDatabase();//SE ABRE LA BASE DE DATOS EN
MODO LECTURA
        Cursor cursor = db.rawQuery("SELECT "+ Utilidades.Campo_nom+",
"+Utilidades.Campo_fecha_nac+" FROM "+Utilidades.Tabla_Contacto+" WHERE
"+Utilidades.Campo_telef+" = '"+numero1+"'",null);// SE EJECUTA LA QUERY
ALMACENANDO LOS RESULTADOS EN ESTE CURSOR
        cursor.moveToFirst();//SE OBTIENE EL PRIMER Y UNICO RESULTADO DE LA QUERY
ANTERIOR
        nombre.setText(cursor.getString(0));//SE MUESTRA EL NOMBRE
        fecha.setText(cursor.getString(1));//SE MUESTRA LA FECHA DE NACIMIENTO
        db.close();//SE CIERRA LA BASE DE DATOS    }
    public void iramodificar (View view){
        Intent i = new Intent(getApplicationContext(),Modificar_Contacto.class);//SE CREA UNA
INSTANCIA PARA LLAMAR LA PANTALLA DE MODIFICAR CONTACTO
        Contacto contacto = new Contacto();//SE CREA UN OBJETO CONTACTO
        contacto.setNum_telef(numero1);//SE AGREGA EL NUMERO DE TELEFONO
        contacto.setNom(nombre.getText().toString());//SE AGREGA SU NOMBRE

        Bundle mibundle = new Bundle();// SE CREA UN PAQUETE PARA ENVIAR DATOS A LA
SIGUIENTE PANTALLA
        mibundle.putSerializable("numero_telefonito",contacto);//SE ALMACENAN LOS DATOS
EN EL PAQUETE
        i.putExtras(mibundle);//SE ENVIAN
        startActivity(i);//SE EJECUTA LA PANTALLA MODIFICAR CONTACTO    }
    public void iraencuesta(View view){
        Intent i = new Intent(getApplicationContext(),Pantalla_Encuesta.class);//SE CREA UNA
INSTANCIA PARA LLAMAR LA PANTALLA DE ENCUESTA
        Contacto contacto = new Contacto();//SE CREA UN OBJETO CONTACTO
        contacto.setNum_telef(numero.getText().toString());//SE AGREGA EL NUMERO DE
TELEFONO
        contacto.setNom(nombre.getText().toString());//SE AGREGA SU NOMBRE
        Bundle mibundle = new Bundle();// SE CREA UN PAQUETE PARA ENVIAR DATOS A LA
SIGUIENTE PANTALLA
        mibundle.putSerializable("contactito",contacto);//SE ALMACENAN LOS DATOS EN EL
PAQUETE
        i.putExtras(mibundle);//SE ENVIAN
        startActivity(i);//SE EJECUTA LA PANTALLA MODIFICAR CONTACTO    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {//SE CREA UN MENU EN LA
TOOLBAR
        getMenuInflater().inflate(R.menu.menu_main, menu);
```

```
        return true;    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {//SE RECIBE LA OPCION DE LA
TOOLBAR QUE SEA SELECCIONADA
        int id = item.getItemId();
        if (id == R.id.opc_preferencia) {
            Intent i = new Intent(this,Pantalla_Preferencias.class);
            numero2 = (String) numero.getText();
            Bundle mibundle = new Bundle();
            mibundle.putString("numero_telefonito",numero2);
            i.putExtras(mibundle);
            startActivity(i);
            return true;       }
        if (id == R.id.opc_eliminar_c) {
            eliminarContacto();
            return true;       }
        return super.onOptionsItemSelected(item);    }
    private void eliminarContacto() {
        final SQLiteDatabase db = conn.getWritableDatabase();
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Realmente deseas Eliminar este contacto? :(");
        builder.setTitle("ELIMINAR CONTACTO");
        builder.setPositiveButton("SI", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                try{
                    String sql = "DELETE FROM "+Utilidades.Tabla_preferencia+" WHERE
"+Utilidades.Campo_telef+" = '"+numero1+"'";
                    db.execSQL(sql);
                    sql = "DELETE FROM "+Utilidades.Tabla_Contacto+" WHERE
"+Utilidades.Campo_telef+" = '"+numero1+"'";
                    db.execSQL(sql);
                    db.close();
                    finish();
                }catch (SQLException e){
                    db.close();            }          }    });
        builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();          }       });
        AlertDialog dialog = builder.create();
        dialog.show();    }
    public void onResume(){
        super.onResume();
        nombre = (TextView) findViewById(R.id.Txt_nombre_especifico);
```

```
        numero = (TextView) findViewById(R.id.txt_numero_especifico);
        fecha = (TextView) findViewById(R.id.txt_fecha_especifica);
        SharedPreferences datos =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        numero1 = datos.getString("numero_telefonito","");
        numero.setText(numero1);
        buscar();    }}
```

Anexo 5: código fuente modificar contacto

```
public class Modificar_Contacto extends AppCompatActivity {
    //DECLARACION DE VARIABLES
    private int año;
    private int mes;
    private int dia;
    private int añoactual;
    private int mesactual;
    private int diaactual;
    private static DatePickerDialog.OnDateSetListener selectorfecha;
    private static final int Tipo_dialogo = 0;
    Button boton_modificar;
    EditText texto_nombre,texto_num,texto_fecha;
    String numero;
    Contacto contacto;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.editar_contacto);
        boton_modificar = (Button) findViewById(R.id.BTN_MODIFICAR);// SE CREA UNA
INSTANCIA VINCULADA A UN COMPONENTE GRAFICO DE TIPO BUTTON QUE AGREGA LOS
DATOS A LA BASE DE DATOS LOCAL
        texto_nombre = (EditText) findViewById(R.id.TXT_EDITNOMBRE_C);// SE CREA UNA
INSTANCIA VINCULADA A UN COMPONENTE GRAFICO DE TIPO EDITTEXT  QUE ALMACENA EL
NOMBRE DEL CONTACTO
        texto_num = (EditText) findViewById(R.id.TXT_EDITNUM_TELEF);// SE CREA UNA
INSTANCIA VINCULADA A UN COMPONENTE GRAFICO DE TIPO EDITTEXT  QUE ALMACENA EL
NUMERO DE TELEFONO DEL CONTACTO
        texto_fecha = (EditText) findViewById(R.id.TXT_EDITFECHA_C);// SE CREA UNA
INSTANCIA VINCULADA A UN COMPONENTE GRAFICO DE TIPO EDITTEXT  QUE ALMACENA LA
FECHA DE NACIMIENTO DEL CONTACTO
        contacto = new Contacto();// SE CREA UN OBJETO QUE GUARDA LOS DATOS DEL
CONTACTO
        Bundle mibundle;
```

```
    mibundle = this.getIntent().getExtras();// SE CREA UNA INSTANCIA QUE RECIBE DATOS
DESDE OTRA PANTALLA
    contacto = (Contacto) mibundle.getSerializable("numero_telefonito");// SE GUARDAN
LOS DATOS DEL CONTACTO QUE VIENEN EN EL PAQUETE ANTERIOR
    texto_nombre.setText(contacto.getNom());// SE MUESTRA EL NOMBRE DEL CONTACTO
    texto_num.setText(contacto.getNum_telef());// SE MUESTRA EL NUMERO DE
TELEFONO
    numero = contacto.getNum_telef();//SE GUARDA EL NUMERO DE TELEFONO
    buscar();
    final BDHelper mDbHelper = new BDHelper(this,"bd_vinculator",null,1);// SE CREA UNA
INSTANCIA QUE CONECTA LA BASE DE DATOS LOCAL
    Calendar calendario = Calendar.getInstance();// SE CREA UNA INSTANCIA DE TIPO
CALENDARIO
    año = calendario.get(Calendar.YEAR);//SE GUARDA EL AÑO ACTUAL
    mes = calendario.get(Calendar.MONTH);//SE GUARDA EL MES ACTUAL
    dia = calendario.get(Calendar.DAY_OF_MONTH);//SE GUARDA EL DIA ACTUAL
    añoactual = calendario.get(Calendar.YEAR);//SE GUARDA EL AÑO ACTUAL
    mesactual = calendario.get(Calendar.MONTH);//SE GUARDA EL MES ACTUAL
    diaactual = calendario.get(Calendar.DAY_OF_MONTH);//SE GUARDA EL DIA ACTUAL
    selectorfecha = new DatePickerDialog.OnDateSetListener() {//NOS MUESTRA UN
CALENDARIO CON LA FECHA ACTUAL SELECCIONADA
        @Override
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
            año = year;
            mes = month;
            dia = dayOfMonth;
            mostrarFecha();         }       };
    boton_modificar.setOnClickListener(new View.OnClickListener() { //CADA VEZ QUE SE
PRESIONA EL BOTON MODIFICAR
        @Override
        public void onClick(View v) {
            SQLiteDatabase db = mDbHelper.getWritableDatabase();// SE ABRE LA CONEXION
EN MODO DE ESCRITURA
            if ((texto_nombre.getText().toString().isEmpty()) ||
(texto_num.getText().toString().isEmpty()) ||
(texto_fecha.getText().toString().isEmpty())){//SI ALGUN CAMPO ESTA VACIO
                Toast.makeText(getApplicationContext(),"TODOS LOS CAMPOS SON
OBLIGATORIOS, VUELVA A INTENTARLO.",Toast.LENGTH_LONG).show();//SE MUESTRA
ESTE MENSAJE            }
            else if((año == añoactual)){//SI EL AÑO SELECCIONADO ES IGUAL AL AÑO ACTUAL
                if (mes == mesactual){// SI EL MES SELECCIONADO ES IGUAL AL MES ACTUAL
                    if (dia <= diaactual){// SI EL DIA SELECCIONADO ES MENOR O IGUAL AL DIA
ACTUAL
                        try{//SI LA CONEXION SE LLEVA A CABO CON EXITO
                            String sql = "UPDATE "+Utilidades.Tabla_preferencia+" SET
```

```
"+Utilidades.Campo_telef+" = '"+texto_num.getText()+"' WHERE
"+Utilidades.Campo_telef+" = '"+numero+"'";
                    //SE GUARDA LA SENTENCIA PARA MODIFICAR LA TABLA PREFERENCIAS,
ACTUALIZANDO EL NUMERO NUEVO DEL CONTACTO
                    db.execSQL(sql);// SE EJECUTA LA SENTENCIA SQL
                    sql = "UPDATE "+ Utilidades.Tabla_Contacto+" set
"+Utilidades.Campo_telef+" = '"+texto_num.getText()+"', "
                         +Utilidades.Campo_nom+" = '"+texto_nombre.getText()+"',
"+Utilidades.Campo_fecha_nac+" = '"+texto_fecha.getText()+"' WHERE
"+Utilidades.Campo_telef +" = '"+numero+"'";
                    //SE GUARDA LA SENTENCIA PARA MODIFICAR EL CONTACO
                    db.execSQL(sql);// SE EJECUTA LA SENTENCIA SQL
                    Toast.makeText(getApplicationContext(),"SU CONTACTO FUE
MODIFICADO CON ÉXITO",Toast.LENGTH_LONG).show();//SE MUESTRA ESTE MENSAJE
                    db.close();// SE CIERRA LA BASE DE DATOS
                    SharedPreferences datos =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());// SE CREA UNA
INSTANCIA PARA ENVIAR EL NUMERO DE TELEFONO NUEVO A OTRA PANTALLA
                    SharedPreferences.Editor editor = datos.edit();// SE HABILITA LA EDICION
DE LA INSTANCIA

editor.putString("numero_telefonito",texto_num.getText().toString());//SE AÑADE EL
VALOR DEL NUMERO TELEFONICO AL PAQUETE
                    editor.apply();// EL PAQUETE SE GUARDA PARA PODER SER EXTRAIDO
DESPUES
                    finish();//SE CIERRA LA PANTALLA ACTUAL                    }
                  catch (Exception e){
                    Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON
ESTE NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
MENSAJE
                    db.close();// SE CIERRA LA BASE DE DATOS                    }                    }
                else{
                  Toast.makeText(getApplicationContext(),"LA FECHA NO PUEDE SER ESA,
VUELVA A INTENTARLO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
MENSAJE                    }                }
              else if(mes < mesactual){//SI EL MES SELECCIONADO ES MENOR AL MES ACTUAL
                try{
                  String sql = "UPDATE "+Utilidades.Tabla_preferencia+" SET
"+Utilidades.Campo_telef+" = '"+texto_num.getText()+"' WHERE
"+Utilidades.Campo_telef+" = '"+numero+"'";
                  db.execSQL(sql);
                  sql = "UPDATE "+ Utilidades.Tabla_Contacto+" set
"+Utilidades.Campo_telef+" = '"+texto_num.getText()+"', "
                       +Utilidades.Campo_nom+" = '"+texto_nombre.getText()+"',
"+Utilidades.Campo_fecha_nac+" = '"+texto_fecha.getText()+"' WHERE
```

```
"+Utilidades.Campo_telef +" = '"+numero+"'";
                db.execSQL(sql);
                Toast.makeText(getApplicationContext(),"SU CONTACTO FUE MODIFICADO
CON ÉXITO",Toast.LENGTH_LONG).show();
                db.close();
                SharedPreferences datos =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
                SharedPreferences.Editor editor = datos.edit();
                editor.putString("numero_telefonito",texto_num.getText().toString());
                editor.apply();
                finish();                }
            catch (Exception e){
                Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON
ESTE NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
MENSAJE
                db.close();              }
          }else{
            Toast.makeText(getApplicationContext(),"LA FECHA NO PUEDE SER ESA,
VUELVA A INTENTARLO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
MENSAJE
          }
        }else if(año < añoactual){// SI EL AÑO SELECCIONADO ES MENOR AL AÑO ACTUAL
          try{
            String sql = "UPDATE "+Utilidades.Tabla_preferencia+" SET
"+Utilidades.Campo_telef+" = '"+texto_num.getText()+"' WHERE
"+Utilidades.Campo_telef+" = '"+numero+"'";
            db.execSQL(sql);
            sql = "UPDATE "+ Utilidades.Tabla_Contacto+" set "+Utilidades.Campo_telef+"
= '"+texto_num.getText()+"', "
                +Utilidades.Campo_nom+" = '"+texto_nombre.getText()+"',
"+Utilidades.Campo_fecha_nac+" = '"+texto_fecha.getText()+"' WHERE
"+Utilidades.Campo_telef +" = '"+numero+"'";
            db.execSQL(sql);
            Toast.makeText(getApplicationContext(),"SU CONTACTO FUE MODIFICADO
CON ÉXITO",Toast.LENGTH_LONG).show();
            db.close();
            SharedPreferences datos =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
            SharedPreferences.Editor editor = datos.edit();
            editor.putString("numero_telefonito",texto_num.getText().toString());
            editor.apply();
            finish();              }
          catch (Exception e){
            Toast.makeText(getApplicationContext(),"YA EXISTE UN CONTACTO CON ESTE
NUMERO DE TELEFONO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE
```

```
MENSAJE
                db.close();                }
            }else{
              Toast.makeText(getApplicationContext(),"LA FECHA NO PUEDE SER ESA, VUELVA
A INTENTARLO",Toast.LENGTH_LONG).show();//SI FALLA SE MUESTRA ESTE MENSAJE
}         }      });   }
   @Override
   protected Dialog onCreateDialog(int id) {
      switch (id) {
         case 0: return new DatePickerDialog(this,selectorfecha, año,mes,dia);// SE CREA EL
CALENDARIO CON LA FECHA ACTUAL       }
      return null;   }
   public void mostrarCalendario(View v){
      showDialog(Tipo_dialogo);// SE MUESTRA EL CALENDARIO   }
   public void mostrarFecha(){
      texto_fecha.setText(dia+"/"+(mes+1)+"/"+año);// SE AÑADE LA FECHA SELECCIONADA
AL CAMPO FECHA   }
   private void buscar() {
      final BDHelper mDbHelper = new BDHelper(this,"bd_vinculator",null,1);//SE ABRE LA
CONEXION A LA BASE DE DATOS
      SQLiteDatabase db = mDbHelper.getReadableDatabase();//SE ABRE LA BASE DE DATOS
EN MODO LECTURA
      Cursor cursor = db.rawQuery("SELECT "+Utilidades.Campo_fecha_nac+" FROM
"+Utilidades.Tabla_Contacto+" WHERE "+Utilidades.Campo_telef+" =
'"+numero+"'",null);//SE EJECUTA LA QUERY QUE BUSCARA LA FECHA DE NACIMIENTO DEL
CONTACTO QUE ESTA SIENDO REVISADO
      cursor.moveToFirst();//SE VE EL PRIMER Y UNICO RESULTADO DE LA QUERY ANTERIOR
      texto_fecha.setText(cursor.getString(0));//SE MUESTRA LA FECHA
      db.close();//SE CIERRA LA BASE DE DATOS   }}
```

Anexo 6: código fuente pantalla de preferencias

```
public class Lista_PreferenciaFragment extends Fragment {
  //DECLARACION DE VARIABLES
  BDHelper conn;
  private OnFragmentInteractionListener mListener;
  RecyclerView recyclerViewPref;
  ArrayList<Preferencia> lista_pref;
  String numero;
  Detalle_Preferencia preferencias;
  Adaptador_Preferencias adapter;
  public Lista_PreferenciaFragment() {    }
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    conn = new BDHelper(getContext(),"bd_vinculator",null,1);// SE ABRE LA CONEXION A
LA BASE DE DATOS LOCAL
    if (getArguments() != null) {       }    }
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                    Bundle savedInstanceState) {
    View vista = inflater.inflate(R.layout.fragment_lista__preferencia, container, false);// SE
CARGA LA PANTALLA QUE CONTIENE LA LISTA DE PREFERENCIAS
    lista_pref = new ArrayList<>();// SE CREA UNA LISTA DE PREFERENCIAS
    preferencias = new Detalle_Preferencia();
    SharedPreferences dato =
PreferenceManager.getDefaultSharedPreferences(getContext());
    numero = dato.getString("numerito","");//se recibe el valor asignado a numerito en la
variable numero
    recyclerViewPref = (RecyclerView) vista.findViewById(R.id.recycler_pref);// SE VINCULA
EL OBJETO A UN COMPONENTE GRAFICO DE TIPO RECYCLERVIEW
    adapter = new Adaptador_Preferencias(lista_pref);//SE CREA UN ADAPTADOR QUE
CONTENGA LA LISTA DE PREFERENCIAS
    adapter.setOnclickListener(new View.OnClickListener(){//CUANDO EL RECYCLER SEA
PRESIONADO
      @Override
      public void onClick(View v) {
        Intent i = new Intent(getContext(),Recomendacion.class);// SE CREA UN OBJETO
QUE ABRIRA LA PANTALLA DE RECOMENDACION
preferencias.setGusto(lista_pref.get(recyclerViewPref.getChildAdapterPosition(v)).getGusto
());//SE GUARDA LA PREFERENCIA QUE ACABA DE SER SELECCIONADA
        preferencias.setNum_telef(numero);//SE GUARDA EL NUMERO DE TELEFONO DEL
CONTACTO QUE ESTA SIENDO REVISADO ACTUALMENTE
        preferencias.setCod_cat("CO");//SE GUARDA EL CODIGO DE LA CATEGORIA DE LA
```

```
PREFERENCIA
            Bundle mibundle = new Bundle();
            mibundle.putSerializable("detalle_pref",preferencias);//SE EMPAQUETAN LOS
DATOS DE LA PREFERENCIA GUARDADOS ANTERIORMENTE CON EL NOMBRE CLAVE
DETALLE_PREF
            i.putExtras(mibundle);//SE MANDA EL PAQUETE DE DATOS HASTA LA PANTALLA DE
RECOMENDACION
            startActivity(i);//SE ABRE LA PANTALLA DE RECOMENDACION          }       });
        return vista;    }
    private void llenarLista() {
        SQLiteDatabase db = conn.getReadableDatabase();//SE ABRE LA BASE DE DATOS EN
MODO DE LECTURA
        Preferencia pref = null;
        lista_pref.clear();//SE LIMPIA LA LISTA DE PREFERENCIAS
        Cursor cursor = db.rawQuery("SELECT gusto FROM preferencia WHERE num_telef =
'"+numero+"' AND cod_cat = 'CO'" ,null);//SE BUSCAN TODAS LAS PREFERENCIAS DE ESTE
CONTACTO QUE SEAN DE LA CATEGORIA COMIDA
        while(cursor.moveToNext()){//MIENTRAS EXISTAN FILAS POR REVISAR
            pref = new Preferencia();//SE CREA UN OBJETO DE TIPO PREFERENCIA
            pref.setGusto(cursor.getString(0));//SE GUARDA LA PREFERENCIA QUE ESTA SIENDO
REVISADA
           lista_pref.add(pref);//SE AÑADE DICHA PREFERENCIA A LA LISTA DE PREFERENCIAS
}
        db.close();//SE CIERRA LA BASE DE DATOS    }
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);       }    }
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof OnFragmentInteractionListener) {
            mListener = (OnFragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement OnFragmentInteractionListener");       }    }
    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;    }
    public void onResume(){//CADA VEZ QUE SE CARGA ESTA PANTALLA
        super.onResume();
        recyclerViewPref.setLayoutManager(new LinearLayoutManager(getContext()));
        llenarLista();
        recyclerViewPref.setAdapter(adapter);//SE CARGAN LOS DATOS DEL ADAPTADOR EN EL
RECYCLER    }
```

```
  public interface OnFragmentInteractionListener {
    void onFragmentInteraction(Uri uri);    }}
```

Anexo 7: código fuente agregar preferencia

```
public class Pantalla_Agregar_Preferencia extends AppCompatActivity {
  //DECLARACION DE VARIABLES
  CheckBox comida,cine,musica,otros;
  String numero;
  BDHelper conn;
  TextView texto_pref;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.pantalla_agregar_pref);
    comida = (CheckBox) findViewById(R.id.checkComida);// SE CREA UNA INSTANCIA DE
TIPO CHECKBOX QUE REPRESENTA LA CATEGORIA DE COMIDA
    cine = (CheckBox) findViewById(R.id.checkCine);// SE CREA UNA INSTANCIA DE TIPO
CHECKBOX QUE REPRESENTA LA CATEGORIA DE CINEMATOGRAFICO
    musica = (CheckBox) findViewById(R.id.checkMusica);// SE CREA UNA INSTANCIA DE
TIPO CHECKBOX QUE REPRESENTA LA CATEGORIA DE MUSICA
    otros = (CheckBox) findViewById(R.id.checkOtros);// SE CREA UNA INSTANCIA DE TIPO
CHECKBOX QUE REPRESENTA LA CATEGORIA DE OTROS
    texto_pref= (TextView) findViewById(R.id.txt_preferencia);// SE CREA UNA INSTANCIA
DE TIPO TEXTVIEW QUE ALMACENA LA PREFERENCIA QUE SE DESEA AGREGAR
    Bundle mibundle;
    mibundle = this.getIntent().getExtras();//SE CREA UNA INSTANCIA QUE RECIBE DATOS
DESDE OTRA PANTALLA
    numero = mibundle.getString("numero_telefonito");//SE RECIBEN LOS DATOS
    conn = new BDHelper(this,"bd_vinculator",null,1);// SE CREA UNA INSTANCIA QUE
PERMITE LA CONEXION A LA BASE DE DATOS    }
  public void aceptar_agregar_pref(View v){
    SQLiteDatabase db = conn.getWritableDatabase();//SE ABRE LA BASE DE DATOS EN
MODO DE ESCRITURA
    if(texto_pref.getText().toString().isEmpty() || (comida.isChecked() == false) &&
(cine.isChecked() == false) && (musica.isChecked()==false)
&&(otros.isChecked()==false)){//SE VALIDA QUE TODOS LOS CAMPOS NECESARIOS ESTEN
COMPLETOS
      Toast.makeText(getApplicationContext(),"DEBE SELECCIONAR UNA CATEGORÍA Y
ESCRIBIR UNA PREFERENCIA",Toast.LENGTH_LONG).show();
    }else {
      try{//SI LA CONEXION SE LLEVO A CABO CON EXITO
        if (comida.isChecked()) {
          String sql = "INSERT INTO " + Utilidades.Tabla_preferencia + " (" +
```

```
Utilidades.Campo_codcat + "," + Utilidades.Campo_gusto + "," + Utilidades.Campo_telef +
") VALUES('CO', '" + texto_pref.getText().toString() + "', '" + numero + "')";
            //SE ALMACENA LA SENTENCIA SQL PARA INSERTAR UNA NUEVA PREFERENCIA AL
CONTACTO QUE ESTA SIENDO REVISADO ACTUALMENTE
            comida.setChecked(false);
            texto_pref.setText("");
            db.execSQL(sql);//SE EJECUTA LA QUERY
            Toast.makeText(getApplicationContext(),"SU PREFERENCIA FUE AGREGADA
CORRECTAMENTE",Toast.LENGTH_LONG).show();
            db.close();// SE CIERRA LA BASE DE DATOS            }
          if (cine.isChecked()) {
            String sql = "INSERT INTO " + Utilidades.Tabla_preferencia + " (" +
Utilidades.Campo_codcat + "," + Utilidades.Campo_gusto + "," + Utilidades.Campo_telef +
") VALUES('CI', '" + texto_pref.getText().toString() + "', '" + numero + "')";
            cine.setChecked(false);
            texto_pref.setText("");
            db.execSQL(sql);
            Toast.makeText(getApplicationContext(),"SU PREFERENCIA FUE AGREGADA
CORRECTAMENTE",Toast.LENGTH_LONG).show();
            db.close();            }
          if (musica.isChecked()) {
            String sql = "INSERT INTO " + Utilidades.Tabla_preferencia + " (" +
Utilidades.Campo_codcat + "," + Utilidades.Campo_gusto + "," + Utilidades.Campo_telef +
") VALUES('MU', '" + texto_pref.getText().toString() + "', '" + numero + "')";
            musica.setChecked(false);
            texto_pref.setText("");
            db.execSQL(sql);
            Toast.makeText(getApplicationContext(),"SU PREFERENCIA FUE AGREGADA
CORRECTAMENTE",Toast.LENGTH_LONG).show();
            db.close();            }
          if (otros.isChecked()) {
            String sql = "INSERT INTO " + Utilidades.Tabla_preferencia + " (" +
Utilidades.Campo_codcat + "," + Utilidades.Campo_gusto + "," + Utilidades.Campo_telef +
") VALUES('OT', '" + texto_pref.getText().toString() + "', '" + numero + "')";
            otros.setChecked(false);
            texto_pref.setText("");
            db.execSQL(sql);
            Toast.makeText(getApplicationContext(),"SU PREFERENCIA FUE AGREGADA
CORRECTAMENTE",Toast.LENGTH_LONG).show();
            db.close();            }        }
        catch (Exception e){//SI FALLA LA CONEXION O LA QUERY
          Toast.makeText(getApplicationContext(),"YA EXISTE ESTA
PREFERENCIA",Toast.LENGTH_LONG).show();
          db.close();// SE CIERRA LA BASE DE DATOS        }      }   }
  public void ValidarChequeo(View v){
```

```
    if (comida.isChecked()){
        cine.setChecked(false);
        musica.setChecked(false);
        otros.setChecked(false);      }
    if(cine.isChecked()){
        comida.setChecked(false);
        musica.setChecked(false);
        otros.setChecked(false);      }
    if(musica.isChecked()){
        comida.setChecked(false);
        cine.setChecked(false);
        otros.setChecked(false);      }
    if(otros.isChecked()){
        comida.setChecked(false);
        cine.setChecked(false);
        musica.setChecked(false);      }  }}
```

Anexo 8: código fuente eliminar preferencia

```
private void eliminarPreferencia() {
   final SQLiteDatabase db = conn.getWritableDatabase();
   AlertDialog.Builder builder = new AlertDialog.Builder(this);
   builder.setMessage("Realmente deseas Eliminar esta preferencia? ");
   builder.setTitle("ELIMINAR PREFERENCIA");
   builder.setPositiveButton("SI", new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
         try{
            String sql = "DELETE FROM "+ Utilidades.Tabla_preferencia+" WHERE
"+Utilidades.Campo_telef+" = '"+preferencias.getNum_telef()+"' AND gusto
='"+preferencias.getGusto()+"'";
            db.execSQL(sql);
            db.close();
            finish();
         }catch (SQLException e){
            db.close();          }       }   });
   builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
         dialog.cancel();       }    });
   AlertDialog dialog = builder.create();
   dialog.show();}
```

Anexo 9: código fuente pantalla de recomendación

```
public class Recomendacion extends AppCompatActivity implements
Response.Listener<JSONObject>, Response.ErrorListener, OnMapReadyCallback,
ActivityCompat.OnRequestPermissionsResultCallback {
  //DECLARACION DE VARIABLES
  Detalle_Preferencia preferencias;
  Toolbar barra;
  BDHelper conn;
  RequestQueue request;
  JsonObjectRequest jsonObjectRequest;
  ArrayList<Locales> listalocal;
  ArrayList<Evento> listaevento;
  ArrayList<String> listagenero;
  int largo_evento, largo_local, largo_genero, actual, actual2;
  Locales local;
  Evento evento;
  MarkerOptions markerOptions;
  TextView nombre, descripcion, direccion, contacto, fecha, genero1, genero2, genero3,
titulo,textocontacto;
  String cadena, longitud_e, latitud_e, longitud_l, latitud_l;
  View view;
  private GoogleMap mMap;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.recomendacion);
    conn = new BDHelper(this, "bd_vinculator", null, 1);
    int status = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
    if (status == ConnectionResult.SUCCESS) {// SE VALIDA LA CORRECTA CONEXION CON EL
MAPA
      SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
            .findFragmentById(R.id.map);
      mapFragment.getMapAsync(this);
    } else {
      Dialog dialog = GooglePlayServicesUtil.getErrorDialog(status, this, 10);
      dialog.show();    }
    barra = (Toolbar) findViewById(R.id.barra_recomendacion);
    setSupportActionBar(barra);
    markerOptions = new MarkerOptions();//SE CREA UN MARCADOR
    actual = 0;
    actual2 = 0;
    largo_genero = 0;
```

```
    request = Volley.newRequestQueue(this);
    Bundle mibundle;
    mibundle = this.getIntent().getExtras();
    preferencias = new Detalle_Preferencia();
    preferencias = (Detalle_Preferencia) mibundle.getSerializable("detalle_pref");
    barra.setTitle(preferencias.getGusto());
    textocontacto = (TextView) findViewById(R.id.txt_contact);
    titulo = (TextView) findViewById(R.id.txt);
    nombre = (TextView) findViewById(R.id.txt_nombre);
    descripcion = (TextView) findViewById(R.id.txt_descripcion);
    direccion = (TextView) findViewById(R.id.txt_direccion);
    contacto = (TextView) findViewById(R.id.txt_contacto);
    fecha = (TextView) findViewById(R.id.txt_fecha);
    genero1 = (TextView) findViewById(R.id.txt_genero1);
    genero2 = (TextView) findViewById(R.id.txt_genero2);
    genero3 = (TextView) findViewById(R.id.txt_genero3);
    genero1.setText("GÉNERO: NINGUNO");
    genero2.setText("GÉNERO: NINGUNO");
    genero3.setText("GÉNERO: NINGUNO");
    titulo.setText("DESLICE HACIA ABAJO Y PRESIONE EN SIGUIENTE");
    textocontacto.setVisibility(View.INVISIBLE);
    nombre.setVisibility(View.INVISIBLE);
    descripcion.setVisibility(View.INVISIBLE);
    direccion.setVisibility(View.INVISIBLE);
    contacto.setVisibility(View.INVISIBLE);
    fecha.setVisibility(View.INVISIBLE);
    genero1.setVisibility(View.INVISIBLE);
    genero2.setVisibility(View.INVISIBLE);
    genero3.setVisibility(View.INVISIBLE);
    listalocal = new ArrayList<>();
    listaevento = new ArrayList<>();
    listagenero = new ArrayList<>();
    cadena = preferencias.getGusto().trim();
    cadena = cadena.replace(" ", "%20");
    if (preferencias.getCod_cat().equals("CO")) {// EN CASO DE QUE LA PREFERENCIA
ELEGIDA FUERA DE COMIDA
        consultarlocal_comida();//SE BUSCAN LOS LOCALES DE COMIDA
        consultarevento_comida();//SE BUSCAN LOS EVENTOS DE COMIDA        }
    if (preferencias.getCod_cat().equals("CI")) {// EN CASO DE QUE LA PREFERENCIA
ELEGIDA FUERA DE CINE
        consultarlocal_cine();//SE BUSCAN LOS LOCALES DE CINE
        consultarevento_cine();//SE BUSCAN LOS EVENTOS DE CINE        }
    if (preferencias.getCod_cat().equals("MU")) {
        consultarlocal_musica();//SE BUSCAN LOS LOCALES DE MUSICA
        consultarevento_musica();//SE BUSCAN LOS EVENTOS DE MUSICA        }    }
```

```
  public void buscar_siguiente(View v) {//BUSCA EL SIGUIENTE LOCAL EN LA LISTA QUE
PUEDA SATISFACER LA PREFERENCIA QUE ESTA SIENDO REVISADA ACTUALMENTE
      listagenero.clear();
      if ((largo_local == 0) && (largo_evento == 0)) {
         Toast.makeText(this, "NO EXISTEN LOCALES O EVENTOS QUE TENGAN ESTA
PREFERENCIA: " + preferencias.getGusto(), Toast.LENGTH_SHORT).show();
         finish();
      } else if (actual < largo_local) {
         nombre.setText("NOMBRE: " + listalocal.get(actual).getNom_local());
         descripcion.setText("DESCRIPCIÓN: " + listalocal.get(actual).getDescripcion_l());
         direccion.setText("DIRECCIÓN: " + listalocal.get(actual).getDireccion_l());
         contacto.setText(listalocal.get(actual).getContacto_l());
         latitud_l = listalocal.get(actual).getLatitud_l();
         longitud_l = listalocal.get(actual).getLongitud_l();
         titulo.setText("INFORMACIÓN DE LOS LOCALES O EVENTOS");
         textocontacto.setVisibility(View.VISIBLE);
         nombre.setVisibility(View.VISIBLE);
         descripcion.setVisibility(View.VISIBLE);
         direccion.setVisibility(View.VISIBLE);
         contacto.setVisibility(View.VISIBLE);
         fecha.setVisibility(View.INVISIBLE);
         genero1.setVisibility(View.INVISIBLE);
         genero2.setVisibility(View.INVISIBLE);
         genero3.setVisibility(View.INVISIBLE);
         LatLng marca1 = new LatLng(Double.parseDouble(latitud_l),
Double.parseDouble(longitud_l));// SE PONE UNA MARCA EN LA DIRECCION DEL LOCAL QUE
ESTA SIENDO REVISADO ACTUALMENTE
         movercamara(marca1, nombre.getText().toString());//SE MUEVE LA CAMARA DENTRO
DEL MAPA A LA MARCA ANTERIOR
         actual++;
      } else if (actual2 < largo_evento) {
         nombre.setText("NOMBRE: " + listaevento.get(actual2).getNom_even());
         descripcion.setText("DESCRIPCIÓN: " + listaevento.get(actual2).getDescripcion_e());
         direccion.setText("DIRECCIÓN: " + listaevento.get(actual2).getDireccion_e());
         contacto.setText(listaevento.get(actual2).getContacto_e());
         fecha.setText("FECHA Y HORA: " + listaevento.get(actual2).getFecha_e());
         latitud_e = listaevento.get(actual2).getLatitud_e();
         longitud_e = listaevento.get(actual2).getLongitud_e();
         titulo.setText("INFORMACIÓN DE LOS LOCALES O EVENTOS");
         nombre.setVisibility(View.VISIBLE);
         textocontacto.setVisibility(View.VISIBLE);
         descripcion.setVisibility(View.VISIBLE);
         direccion.setVisibility(View.VISIBLE);
         contacto.setVisibility(View.VISIBLE);
         fecha.setVisibility(View.VISIBLE);
```

```
        genero1.setVisibility(View.VISIBLE);
        genero2.setVisibility(View.VISIBLE);
        genero3.setVisibility(View.VISIBLE);
        LatLng marca2 = new LatLng(Double.parseDouble(latitud_e),
Double.parseDouble(longitud_e));// SE PONE UNA MARCA EN LA DIRECCION DEL LOCAL QUE
ESTA SIENDO REVISADO ACTUALMENTE
        movercamara(marca2, nombre.getText().toString());//SE MUEVE LA CAMARA DENTRO
DEL MAPA A LA MARCA ANTERIOR
        consultar_generoevento();//SE CONSULTAN LOS GENEROS QUE TENGA ESTE EVENTO
        actual2++;
      } else {
        titulo.setVisibility(View.INVISIBLE);
        nombre.setVisibility(View.INVISIBLE);
        descripcion.setVisibility(View.INVISIBLE);
        direccion.setVisibility(View.INVISIBLE);
        contacto.setVisibility(View.INVISIBLE);
        fecha.setVisibility(View.INVISIBLE);
        genero1.setVisibility(View.INVISIBLE);
        genero2.setVisibility(View.INVISIBLE);
        genero3.setVisibility(View.INVISIBLE);      }   }
    private void consultar_generoevento() {
      String url = "http://192.168.1.31:/conexiones/consultar_generoeventos.php?codigo="
+ listaevento.get(actual2).getCod_even();
      jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
          JSONArray json = response.optJSONArray("spinner");
          try {
            for (int i = 0; i < json.length(); i++) {
              JSONObject jsonobjecto = null;
              jsonobjecto = json.getJSONObject(i);
              listagenero.add(jsonobjecto.optString("nom_gen"));
              largo_genero = listagenero.size();                }
            if (largo_genero == 0) {
              genero1.setText("GÉNERO: NINGUNO");
              genero2.setText("GÉNERO: NINGUNO");
              genero3.setText("GÉNERO: NINGUNO");
            } else if (largo_genero == 1) {
              genero1.setText("GÉNERO: " + listagenero.get(0));
            } else if (largo_genero == 2) {
              genero1.setText("GÉNERO: " + listagenero.get(0));
              genero2.setText("GÉNERO: " + listagenero.get(1));
            } else if (largo_genero == 3) {
              genero1.setText("GÉNERO: " + listagenero.get(0));
```

```
                genero2.setText("GÉNERO: " + listagenero.get(1));
                genero3.setText("GÉNERO: " + listagenero.get(2));              }
            largo_genero = 0;
            listagenero.clear();
        } catch (JSONException e) {
            e.printStackTrace();            }        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            largo_genero = 0;
            listagenero.clear();
            genero1.setText("GÉNERO: NINGUNO");
            genero2.setText("GÉNERO: NINGUNO");
            genero3.setText("GÉNERO: NINGUNO");          }       });
    request.add(jsonObjectRequest);    }
  private void consultarevento_cine() {
    String url = "http://192.168.1.31:/conexiones/consultareventos_cine.php?gustos=" +
cadena;
    jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            JSONArray json = response.optJSONArray("spinner");
            try {
                for (int i = 0; i < json.length(); i++) {
                    evento = new Evento();
                    JSONObject jsonobjecto = null;
                    jsonobjecto = json.getJSONObject(i);
                    evento.setCod_even(jsonobjecto.optInt("cod_even"));
                    evento.setNom_even(jsonobjecto.optString("nom_even"));
                    evento.setCod_cat(jsonobjecto.optString("cod_cat"));
                    evento.setContacto_e(jsonobjecto.optString("contacto_e"));
                    evento.setDescripcion_e(jsonobjecto.optString("descripcion_e"));
                    evento.setDireccion_e(jsonobjecto.optString("direccion_e"));
                    evento.setFecha_e(jsonobjecto.optString("fecha"));
                    evento.setLatitud_e(jsonobjecto.optString("Latitud_e"));
                    evento.setLongitud_e(jsonobjecto.optString("Longitud_e"));
                    listaevento.add(evento);
                    largo_evento = listaevento.size();                }
            } catch (JSONException e) {
                e.printStackTrace();            }        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {        }       });
    request.add(jsonObjectRequest);    }
```

```
  private void consultarlocal_cine() {
    String url = "http://192.168.1.31/conexiones/consultarlocales_cine.php";
    jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
      @Override
      public void onResponse(JSONObject response) {
        JSONArray json = response.optJSONArray("spinner");
        try {
          for (int i = 0; i < json.length(); i++) {
            local = new Locales();
            JSONObject jsonobject = null;
            jsonobject = json.getJSONObject(i);
            local.setCod_local(jsonobject.optInt("cod_local"));
            local.setNom_local(jsonobject.optString("nom_local"));
            local.setCod_cat(jsonobject.optString("cod_cat"));
            local.setContacto_l(jsonobject.optString("contacto_l"));
            local.setDescripcion_l(jsonobject.optString("descripcion_l"));
            local.setDireccion_l(jsonobject.optString("direccion_l"));
            local.setLatitud_l(jsonobject.optString("Latitud_l"));
            local.setLongitud_l(jsonobject.optString("Longitud_l"));
            listalocal.add(local);
            largo_local = listalocal.size();              }
        } catch (JSONException e) {
          e.printStackTrace();            }        }
    }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {        }      });
    request.add(jsonObjectRequest);   }
  private void consultarevento_comida() {
    String url =
"http://192.168.1.31:/conexiones/consultareventos_comida.php?categorias=CO&gustos=
" + cadena;
    jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
      @Override
      public void onResponse(JSONObject response) {
        JSONArray json = response.optJSONArray("spinner");
        try {
          for (int i = 0; i < json.length(); i++) {
            evento = new Evento();
            JSONObject jsonobjecto = null;
            jsonobjecto = json.getJSONObject(i);
            evento.setCod_even(jsonobjecto.optInt("cod_even"));
            evento.setNom_even(jsonobjecto.optString("nom_even"));
            evento.setCod_cat(jsonobjecto.optString("cod_cat"));
```

```
                evento.setContacto_e(jsonobjecto.optString("contacto_e"));
                evento.setDescripcion_e(jsonobjecto.optString("descripcion_e"));
                evento.setDireccion_e(jsonobjecto.optString("direccion_e"));
                evento.setFecha_e(jsonobjecto.optString("fecha"));
                evento.setLatitud_e(jsonobjecto.optString("Latitud_e"));
                evento.setLongitud_e(jsonobjecto.optString("Longitud_e"));
                listaevento.add(evento);
                largo_evento = listaevento.size();              }
            } catch (JSONException e) {
                e.printStackTrace();            }         }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {          }       });
        request.add(jsonObjectRequest);    }
    private void consultarlocal_musica() {
        String url = "http://192.168.1.31/conexiones/consultarlocales_musica.php?gustos=" +
cadena;
        jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                JSONArray json = response.optJSONArray("spinner");
                try {
                    for (int i = 0; i < json.length(); i++) {
                        local = new Locales();
                        JSONObject jsonobject = null;
                        jsonobject = json.getJSONObject(i);
                        local.setCod_local(jsonobject.optInt("cod_local"));
                        local.setNom_local(jsonobject.optString("nom_local"));
                        local.setCod_cat(jsonobject.optString("cod_cat"));
                        local.setContacto_l(jsonobject.optString("contacto_l"));
                        local.setDescripcion_l(jsonobject.optString("descripcion_l"));
                        local.setDireccion_l(jsonobject.optString("direccion_l"));
                        local.setLatitud_l(jsonobject.optString("Latitud_l"));
                        local.setLongitud_l(jsonobject.optString("Longitud_l"));
                        listalocal.add(local);
                        largo_local = listalocal.size();                }
                } catch (JSONException e) {
                    e.printStackTrace();            }         }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {          }       });
        request.add(jsonObjectRequest);    }
    private void consultarevento_musica() {
        String url = "http://192.168.1.31/conexiones/consultareventos_musica.php?gustos="
```

```
+ cadena;
      jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, new
Response.Listener<JSONObject>() {
         @Override
         public void onResponse(JSONObject response) {
            JSONArray json = response.optJSONArray("spinner");
            try {
               for (int i = 0; i < json.length(); i++) {
                  evento = new Evento();
                  JSONObject jsonobjecto = null;
                  jsonobjecto = json.getJSONObject(i);
                  evento.setCod_even(jsonobjecto.optInt("cod_even"));
                  evento.setNom_even(jsonobjecto.optString("nom_even"));
                  evento.setCod_cat(jsonobjecto.optString("cod_cat"));
                  evento.setContacto_e(jsonobjecto.optString("contacto_e"));
                  evento.setDescripcion_e(jsonobjecto.optString("descripcion_e"));
                  evento.setDireccion_e(jsonobjecto.optString("direccion_e"));
                  evento.setFecha_e(jsonobjecto.optString("fecha"));
                  evento.setLatitud_e(jsonobjecto.optString("Latitud_e"));
                  evento.setLongitud_e(jsonobjecto.optString("Longitud_e"));
                  listaevento.add(evento);
                  largo_evento = listaevento.size();                }
            } catch (JSONException e) {
               e.printStackTrace();            }          }
      }, new Response.ErrorListener() {
         @Override
         public void onErrorResponse(VolleyError error) {          }       });
      request.add(jsonObjectRequest);    }
   private void consultarlocal_comida() {
      String url =
"http://192.168.1.31:/conexiones/consultarlocales_comida.php?categorias=CO&gustos="
+ cadena;
      jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null, this, this);
      request.add(jsonObjectRequest);    }
   @Override
   public void onErrorResponse(VolleyError error) {    }
   @Override
   public void onResponse(JSONObject response) {
      JSONArray json = response.optJSONArray("spinner");
      try {
         for (int i = 0; i < json.length(); i++) {
            local = new Locales();
            JSONObject jsonobject = null;
            jsonobject = json.getJSONObject(i);
            local.setCod_local(jsonobject.optInt("cod_local"));
```

```
            local.setNom_local(jsonobject.optString("nom_local"));
            local.setCod_cat(jsonobject.optString("cod_cat"));
            local.setContacto_l(jsonobject.optString("contacto_l"));
            local.setDescripcion_l(jsonobject.optString("descripcion_l"));
            local.setDireccion_l(jsonobject.optString("direccion_l"));
            local.setLatitud_l(jsonobject.optString("Latitud_l"));
            local.setLongitud_l(jsonobject.optString("Longitud_l"));
            listalocal.add(local);
            largo_local = listalocal.size();          }
        } catch (JSONException e) {
          e.printStackTrace();       }    }
    public void movercamara(LatLng coordenada, String nombre) {
        mMap.addMarker(new
MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.drawable.icono)).anchor(0.0
f, 1.0f).position(coordenada).title(nombre));
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(coordenada, 15));    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        UiSettings uiSettings = mMap.getUiSettings();
        uiSettings.setZoomControlsEnabled(true);
        LatLng coordenada = new LatLng(-33.050713942823286,-71.61104854999996);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(coordenada, 15));    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_preferencia, menu);
        return true;    }
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();
        if (id == R.id.opc_eliminar_pref) {
            eliminarPreferencia();
            return true;       }
        return super.onOptionsItemSelected(item);    }
    private void eliminarPreferencia() {
        final SQLiteDatabase db = conn.getWritableDatabase();
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Realmente deseas Eliminar esta preferencia? ");
        builder.setTitle("ELIMINAR PREFERENCIA");
        builder.setPositiveButton("SI", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                try{
                    String sql = "DELETE FROM "+ Utilidades.Tabla_preferencia+" WHERE
```

```
"+Utilidades.Campo_telef+" = '"+preferencias.getNum_telef()+"' AND gusto
='"+preferencias.getGusto()+"'";
                db.execSQL(sql);
                db.close();
                finish();
            }catch (SQLException e){
                db.close();            }        }      });
      builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();          }      });
      AlertDialog dialog = builder.create();
      dialog.show();   }
   public void hipervinculo(View v){
      if(contacto.getText().equals("")|| contacto.getText().equals("SIN INFORMACION")){
      }else{
         Intent hiperv = new Intent(Intent.ACTION_VIEW,
Uri.parse(contacto.getText().toString()));
         startActivity(hiperv);      }   }}
```

Código fuente php utilizado para la pantalla de recomendación

```
<?php

$hostname_localhost = "localhost";

$database_localhost = "vinculator_bd_externa";

$username_localhost = "root";

$password_localhost ="";

$categoria = $_GET["categorias"];

$preferencias = $_GET["gustos"];

$json=array();

                    $conexion =
mysqli_connect($hostname_localhost,$username_localhost,$password_localhost,$database_localho
st);

                    $consulta = "select * from local where cod_local = ANY(SELECT cod_local from
local_prod where cod_prod = (Select cod_prod from producto where nom_prod like
'%{$preferencias}%' LIMIT 0,1 ));";

                    mysqli_set_charset($conexion,"utf8");

                    $resultado = mysqli_query($conexion,$consulta);

                    while($registro=mysqli_fetch_array($resultado)){
```

```
                $json['spinner'][]=$registro;                }
            mysqli_close($conexion);
            echo json_encode($json)?>
```

Anexo 10: código fuente solicitud de registro local/evento/producto

```
public class Agregar_Local extends AppCompatActivity {
  //DECLARACION DE VARIABLES
  String Correo,Contraseña,mensaje;
  EditText nom,dir,descr,contactito;
  CheckBox comida,cine,musica;
  Button enviar;
  Session session;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.agregar_local);
    nom = (EditText) findViewById(R.id.txt_nomlocal);// SE CREA UNA INSTANCIA DE TIPO
EDITTEXT QUE ALMACENARA EL NOMBRE DEL LOCAL
    dir = (EditText) findViewById(R.id.txt_dirlocal);// SE CREA UNA INSTANCIA DE TIPO
EDITTEXT QUE ALMACENARA LA DIRECCION DEL LOCAL
    descr = (EditText) findViewById(R.id.txt_descripcion_local);// SE CREA UNA INSTANCIA
DE TIPO EDITTEXT QUE ALMACENARA LA DESCRIPCION DEL LOCAL
    contactito = (EditText) findViewById(R.id.txt_contacto_local);// SE CREA UNA
INSTANCIA DE TIPO EDITTEXT QUE ALMACENARA EL LINK DE LA PAGINA OFICIAL DEL LOCAL
    comida = (CheckBox) findViewById(R.id.checkcomidita);// SE CREA UNA INSTANCIA DE
TIPO CHECKBOX QUE REPRESENTA LA CATEGORIA DE COMIDA
    cine = (CheckBox) findViewById(R.id.checkcinecito);// SE CREA UNA INSTANCIA DE TIPO
CHECKBOX QUE REPRESENTA LA CATEGORIA DE CINEMATOGRAFICO
    musica = (CheckBox) findViewById(R.id.checkmusicacita);// SE CREA UNA INSTANCIA DE
TIPO CHECKBOX QUE REPRESENTA LA CATEGORIA DE MUSICA
    enviar = (Button) findViewById(R.id.BTN_enviarlocal);;// SE CREA UNA INSTANCIA DE
TIPO BUTTON QUE AL PRESIONAR ENVIA LOS DATOS DEL LOCAL AL ADMINISTRADOR DE LA
BASE DE DATOS EXTERNA
    Correo = "VinculatorPD@gmail.com";// SE ALMACENA EL CORREO QUE ENVIA LA
INFORMACION
    Contraseña = "bestideaever";// SE ALMACENA LA CONTRASEÑA DE ESTE CORREO    }
  public void enviarmensaje(View v){// SE ENVIAN LOS DATOS AL ADMINISTRADOR DE LA
BASE DE DATOS
    StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);
    Properties properties = new Properties();
```

```
properties.put("mail.smtp.host","smtp.googlemail.com");
properties.put("mail.smtp.socketFactory.port","465");
properties.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
properties.put("mail.smtp.auth","true");
properties.put("mail.smtp.port","465");
// TODAS LAS SENTENCIAS ANTERIORES SE ENCARGAN DE CONFIGURAR LOS PUERTOS,
PERMISOS Y POLITICAS NECESARIAS PARA ENVIAR UN EMAIL DESDE LA APLICACION
if((nom.getText().toString().isEmpty()) || (dir.getText().toString().isEmpty() ) ||
(descr.getText().toString().isEmpty())||(contactito.getText().toString().isEmpty())||
((comida.isChecked()==false) && (cine.isChecked()==false)
&&(musica.isChecked()==false))){// SI ALGUN CAMPO NO FUE LLENADO
    Toast.makeText(this,"TODOS LOS CAMPOS SON
OBLIGATORIOS",Toast.LENGTH_SHORT).show();// SE MUESTRA ESTE MENSAJE
}else{
    mensaje=
nom.getText().toString()+","+dir.getText().toString()+","+descr.getText().toString()+","+cont
actito.getText().toString();// EL MENSAJE SE ESTRUCTURA CON EL NOMBRE DEL LOCAL,
DIRECCION, DESCRIPCION, CONTACTO
    if (comida.isChecked()) {//SI SE ESCOGIO LA CATEGORIA COMIDA
        mensaje = mensaje+","+"CO";// SE AÑADE AL MENSAJE EL CODIGO DE LA
CATEGORIA SELECCIONADA
        comida.setChecked(false);// SE DESMARCA EL CHECKBOX          }
    if (cine.isChecked()) {//SI SE ESCOGIO LA CATEGORIA CINEMATOGRAFICO
        mensaje = mensaje+","+"CI";// SE AÑADE AL MENSAJE EL CODIGO DE LA
CATEGORIA SELECCIONADA
        cine.setChecked(false);// SE DESMARCA EL CHECKBOX          }
    if (musica.isChecked()) {//SI SE ESCOGIO LA CATEGORIA MUSICA
        mensaje = mensaje+","+"MU";// SE AÑADE AL MENSAJE EL CODIGO DE LA
CATEGORIA SELECCIONADA
        musica.setChecked(false);// SE DESMARCA EL CHECKBOX          }
    try{
        session = Session.getDefaultInstance(properties, new Authenticator() {// SE ABRE
LA SESION DE CORREO CON EL CORREO Y CONTRASEÑA ALMACENADAS ANTERIORMENTE
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(Correo,Contraseña);              }
});
        if (session != null){//SI LA SESION SE REALIZO CON EXITO ENTONCES
            Message message = new MimeMessage(session);//EL MENSAJE TENDRA LA
SESION ANTERIOR
            message.setFrom(new InternetAddress(Correo));// SE CONFIGURA EL CORREO
DESDE EL QUE SE ENVIA EL MENSAJE
            message.setSubject("Solicitud agregar local");// SE CONFIGURA EL ASUNTO DEL
MENSAJE
message.setRecipients(Message.RecipientType.TO,InternetAddress.parse("vinculatormante
```

```
nedores@gmail.com"));// SE CONFIGURA EL CORREO QUE RECIBIRA EL MENSAJE
            message.setContent(mensaje,"text/html; charset=utf-8");// SE CONFIGURA QUE
EL CONTENIDO ADMITA CARACTERES ESPECIALES
            Transport.send(message);// SE ENVIA EL MENSAJE
            nom.setText("");// SE VACIA EL NOMBRE
            dir.setText("");// SE VACIA LA DIRECCION
            descr.setText("");// SE VACIA LA DESCRIPCION
            contactito.setText("");// SE VACIA EL CONTACTO
            comida.setChecked(false);// SE DESMARCA EL CHECKBOX
            cine.setChecked(false);// SE DESMARCA EL CHECKBOX
            musica.setChecked(false);// SE DESMARCA EL CHECKBOX
            Toast.makeText(this,"SU SUGERENCIA DE LOCAL FUE ENVIADA EXITOSAMENTE,
UN ADMINISTRADOR SE ENCARGARÁ DE VERIFICAR LA INFORMACIÓN Y LUEGO ESTARÁ
DISPONIBLE PARA TODOS, MUCHAS GRACIAS",Toast.LENGTH_LONG).show();//SE MUESTRA
ESTE MENSAJE            }
        }catch (Exception e){
            Toast.makeText(this,"ERROR DE CONEXIÓN",Toast.LENGTH_LONG).show();// SE
MUESTRA ESTE MENSAJE SI ALGO SALE MAL
            finish();// SE CIERRA LA PANTALLA ACTUAL        }      }   }
  public void validarcheck(View v){
     if (comida.isChecked()){// SI SE SELECCIONA
        cine.setChecked(false);//SE DESMARCA
        musica.setChecked(false);//SE DESMARCA      }
     if(cine.isChecked()){// SI SE SELECCIONA
        comida.setChecked(false);//SE DESMARCA
        musica.setChecked(false);//SE DESMARCA      }
     if(musica.isChecked()){// SI SE SELECCIONA
        comida.setChecked(false);//SE DESMARCA
        cine.setChecked(false);//SE DESMARCA
     }  }}
```

## Anexo 11: código fuente Mantenedor Local

11.1 Agregar Local

```
private void BTN_ACEPTARActionPerformed(java.awt.event.ActionEvent evt) {

      String nombre = TXT_NOMBRE_L.getText();

     String direccion = TXT_DIRECCION_L.getText();

     validar_local(nombre,direccion);

     mostrardatos("");

     Limpiar(); }

String obtenercod_cat(String valor){
```

```
    String dato;
    try {
      String csql = "SELECT cod_cat FROM categoria WHERE nom_cat='"+valor+"'";
      Statement st = cn.createStatement();
      ResultSet rs = st.executeQuery(csql);
      while(rs.next()){
        dato=rs.getString(1);
        valor = dato;          }       }
    catch(Exception e){
      System.out.println("Falló la carga del combobox \n"+e);                    }
    return valor;}
String validar_local(String valor , String valor2){
    String vali = "";
    try {
      String csql = "SELECT * FROM local WHERE nom_local='"+valor+"' and direccion_l =
'"+valor2+"'";
      Statement st = cn.createStatement();
      ResultSet rs = st.executeQuery(csql);
      while(rs.next()){
        vali=rs.getString(1);          }       }
    catch(Exception e){
      System.out.println("Falló la carga del combobox \n"+e);                    }
    if (vali.equals("")){
      try {
        if (TXT_NOMBRE_L.getText().equals("") || TXT_DIRECCION_L.getText().equals("") ||
                TXT_LATITUD_L.getText().equals("")|| TXT_LONGITUD_L.getText().equals("")){
            JOptionPane.showMessageDialog(null, "No puede dejar campos en blanco.");
        }else{
    PreparedStatement pst = cn.prepareStatement("INSERT INTO
local(nom_local,cod_cat,contacto_l,
```

```
                    descripcion_l,direccion_l,latitud_l,longitud_l) VALUES (?,?,?,?,?,?,?)");
      pst.setString(1, TXT_NOMBRE_L.getText().toUpperCase());
      pst.setString(5, TXT_DIRECCION_L.getText());
      String a;
      a = obtenercod_cat(CB_CATEGORIA_L.getSelectedItem().toString());
      pst.setString(2,a);
      pst.setString(3, TXT_CONTACTO_L.getText());
      pst.setString(4, TXT_DESCRIPCION_L.getText());
      pst.setString(6,TXT_LATITUD_L.getText());
      pst.setString(7, TXT_LONGITUD_L.getText());
      pst.executeUpdate();
      JOptionPane.showMessageDialog(null, "Local Agregado");
      mostrardatos("");
      Limpiar();        }
   } catch (Exception e) {
      System.out.print(e.getMessage());    }
      }else {
       JOptionPane.showMessageDialog(null, "Local duplicado");       }
      return valor;}
```

11.2 Modificar Local

```
private void ActualizarActionPerformed(java.awt.event.ActionEvent evt) {
   try {
      if (TXT_NOMBRE_L.getText().equals("") || TXT_DIRECCION_L.getText().equals("") ||
            TXT_LATITUD_L.getText().equals("")|| TXT_LONGITUD_L.getText().equals("")){
         JOptionPane.showMessageDialog(null, "No puede dejar campos en blanco.");
      }else{
      String a;
      a = obtenercod_cat(CB_CATEGORIA_L.getSelectedItem().toString());
      PreparedStatement pst = cn.prepareStatement("UPDATE local SET
nom_local='"+TXT_NOMBRE_L.getText()+"',
            direccion_l='"+TXT_DIRECCION_L.getText()+"'" +
```

```
",cod_cat='"+a+"',contacto_l='"+TXT_CONTACTO_L.getText()+"',descripcion_l='"+
            TXT_DESCRIPCION_L.getText()+"'"+
",latitud_l='"+TXT_LATITUD_L.getText()+"',longitud_l='"+TXT_LONGITUD_L.getText()+
            "' WHERE cod_local='"+TXT_COD_LOCAL.getText()+"'");
        pst.executeUpdate();
        JOptionPane.showMessageDialog(null, "Local Actualizado");
        mostrardatos("");
        Limpiar();                    }
    } catch (Exception e) {
        System.out.print(e.getMessage());    }    }
```

11.3 Eliminar Local

```
private void BTN_EliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int fila = tb_local.getSelectedRow();
    String cod="";
    cod=tb_local.getValueAt(fila, 0).toString();
    try {
        PreparedStatement pst = cn.prepareStatement("DELETE FROM local WHERE
cod_local='"+cod+"'");
        pst.executeUpdate();
        PreparedStatement dpst = cn.prepareStatement("DELETE FROM local_prod WHERE
cod_local='"+cod+"'");
        dpst.executeUpdate();
        JOptionPane.showMessageDialog(null, "Local Eliminado");
        Limpiar();
        mostrardatos("");
    } catch (Exception e) {    }       }
```

## Anexo 12: código fuente Mantenedor Evento

12.1 Agregar Evento

```
private void BTN_ACEPTARActionPerformed(java.awt.event.ActionEvent evt) {
    String nombre = TXT_NOMBRE_E1.getText();
    String direccion = TXT_DIRECCION_E1.getText();
    String fecha = TXT_FECHA_E.getText();
    validar_evento(nombre,direccion,fecha);
    mostrardatose("");
    Limpiar();   }
  String validar_evento(String valor , String valor2, String valor3){
    String vali = "";
    try {
      String csql = "SELECT * FROM evento WHERE nom_even='"+valor+"' and direccion_e =
'"+valor2+"' and fecha = '"+valor3+"'";
      Statement st = cn.createStatement();
      ResultSet rs = st.executeQuery(csql);
      while(rs.next()){
        vali=rs.getString(1);          }     }
    catch(Exception e){
      System.out.println("Falló la carga del combobox \n"+e);                }
    if (vali.equals("")){
   try {
      if (TXT_NOMBRE_E1.getText().equals("") || TXT_DIRECCION_E1.getText().equals("") ||
TXT_LATITUD_E.getText().equals("")|| TXT_LONGITUD_E.getText().equals("")||
TXT_FECHA_E.getText().equals("")){
        JOptionPane.showMessageDialog(null, "No puede dejar campos en blanco.");
      }else{
    PreparedStatement pst = cn.prepareStatement("INSERT INTO
evento(nom_even,cod_cat,fecha,contacto_e,descripcion_e,direccion_e,latitud_e,longitud_e)
VALUES (?,?,?,?,?,?,?,?)");
    pst.setString(1, TXT_NOMBRE_E1.getText());
    pst.setString(6, TXT_DIRECCION_E1.getText());
```

```
String a;
a = obtenercod_cat(CB_CATEGORIA_E1.getSelectedItem().toString());
pst.setString(2,a);
pst.setString(3,TXT_FECHA_E.getText());
pst.setString(4, TXT_CONTACTO_E1.getText());
pst.setString(5, TXT_DESCRIPCION_E1.getText());
pst.setString(7,TXT_LATITUD_E.getText());
pst.setString(8, TXT_LONGITUD_E.getText());
pst.executeUpdate();
JOptionPane.showMessageDialog(null, "Evento Agregado");
String sql = "Select cod_even from evento where nom_even = '"+ TXT_NOMBRE_E1.getText()
+"'";
Statement st = cn.createStatement();
ResultSet rs = st.executeQuery(sql);
rs.next();
String ex = rs.getString(1);
PreparedStatement pstg = cn.prepareStatement("INSERT INTO even_gen(cod_even,cod_gen)
VALUES (?,?) ");
pstg.setString(1, ex);
String g;
g = obtenercod_gen(CB_GENERO1.getSelectedItem().toString());
pstg.setString(2,g);
pstg.executeUpdate();
String o1 = CB_GENERO1.getSelectedItem().toString();
String o2 = CB_GENERO2.getSelectedItem().toString();
String o3 = CB_GENERO3.getSelectedItem().toString();
if (o2.equals(o1)){
   System.out.print("son iguales");
   }else{
String sql2 = "Select cod_even from evento where nom_even = '"+ TXT_NOMBRE_E1.getText()
+"'";
```

```
Statement st2 = cn.createStatement();
ResultSet rs2 = st2.executeQuery(sql2);
rs2.next();
String ex2 = rs2.getString(1);
PreparedStatement pstg2 = cn.prepareStatement("INSERT INTO even_gen(cod_even,cod_gen)
VALUES (?,?) ");
pstg2.setString(1, ex2);
String g2;
g2 = obtenercod_gen(CB_GENERO2.getSelectedItem().toString());
pstg2.setString(2,g2);
pstg2.executeUpdate();          }
    if (o3.equals(o1) || o3.equals(o1) ){
  System.out.print("son iguales");
  }else{
    String sql3 = "Select cod_even from evento where nom_even = '"+
TXT_NOMBRE_E1.getText() +"'";
Statement st3 = cn.createStatement();
ResultSet rs3 = st3.executeQuery(sql3);
rs3.next();
String ex3 = rs3.getString(1);
PreparedStatement pstg3 = cn.prepareStatement("INSERT INTO even_gen(cod_even,cod_gen)
VALUES (?,?) ");
pstg3.setString(1, ex3);
String g3;
g3 = obtenercod_gen(CB_GENERO3.getSelectedItem().toString());
pstg3.setString(2,g3);
pstg3.executeUpdate();       }
mostrardatose("");
Limpiar();        }
} catch (Exception e) {
System.out.print(e.getMessage());    }
```

```
        }else {
         JOptionPane.showMessageDialog(null, "Evento duplicado");        }
        return valor;    }
```

12.2 Modificar Evento

```
private void ActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        if (TXT_NOMBRE_E1.getText().equals("") || TXT_DIRECCION_E1.getText().equals("") ||
TXT_LATITUD_E.getText().equals("")|| TXT_LONGITUD_E.getText().equals("") ||
TXT_FECHA_E.getText().equals("")){
            JOptionPane.showMessageDialog(null, "No puede dejar campos en blanco.");
        }else{
        String a;
        a = obtenercod_cat(CB_CATEGORIA_E1.getSelectedItem().toString());
        PreparedStatement pst = cn.prepareStatement("UPDATE evento SET
nom_even='"+TXT_NOMBRE_E1.getText()+"',direccion_e='"+TXT_DIRECCION_E1.getText()+"'"
            +
",cod_cat='"+a+"',contacto_e='"+TXT_CONTACTO_E1.getText()+"',descripcion_e='"+TXT_DESCRIPCIO
N_E1.getText()+"'"
                +
",latitud_e='"+TXT_LATITUD_E.getText()+"',longitud_e='"+TXT_LONGITUD_E.getText()+"' WHERE
cod_even='"+TXT_COD_EVENTO.getText()+"'");
        pst.executeUpdate();
        JOptionPane.showMessageDialog(null, "Evento Actualizado");
        mostrardatose("");
        Limpiar();                }
    } catch (Exception e) {
        System.out.print(e.getMessage()); }        }
```

12.3 Eliminar Evento

```
private void EliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int fila = tb_evento.getSelectedRow();
    String cod="";
    cod=tb_evento.getValueAt(fila, 0).toString();
```

```
   try {
      PreparedStatement pst = cn.prepareStatement("DELETE FROM evento WHERE
cod_even='"+cod+"'");
      pst.executeUpdate();
      PreparedStatement pste = cn.prepareStatement("DELETE FROM even_gen WHERE
cod_even='"+cod+"'");
      pste.executeUpdate();
      JOptionPane.showMessageDialog(null, "Evento Eliminado");
      Limpiar();
      mostrardatose("");
   } catch (Exception e) {    }         }
```

## Anexo 13: código fuente Mantenedor Producto

13.1 Agregar Producto

```
String validar_producto(String valor){
      String vali = "";
      valor = valor.toUpperCase();
      try {
         String csql = "SELECT * FROM producto WHERE nom_prod='"+valor+"'";
         Statement st = cn.createStatement();
         ResultSet rs = st.executeQuery(csql);
         while(rs.next()){
            vali=rs.getString(1);            }       }
      catch(Exception e){
         System.out.println("Falló la carga del combobox \n"+e);                    }
      if (vali.equals("")){
   try {
         if (TXT_NUEVO_P.getText().equals("")){
            JOptionPane.showMessageDialog(null, "No puede dejar el campo en blanco.");
         }else{
         PreparedStatement pst = cn.prepareStatement("INSERT INTO producto(nom_prod) VALUES
(?)");
```

```
        pst.setString(1, TXT_NUEVO_P.getText().toUpperCase());
        pst.executeUpdate();
        JOptionPane.showMessageDialog(null, "Producto Agregado");
        mostrardatos("");
        TXT_NUEVO_P.setText("");       }
    }catch (Exception e) {
      System.out.print(e.getMessage());   }
      }else {
       JOptionPane.showMessageDialog(null, "Producto duplicado");  }
      return valor;    }
```

13.2 Modificar Producto

```
private void ActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
      if (TXT_NUEVO_P.getText().equals("")){
        JOptionPane.showMessageDialog(null, "No puede dejar el campo en blanco.");
      }else{
        int fila = tb_productos.getSelectedRow();
        String coI="";
        coI=tb_productos.getValueAt(fila, 1).toString();
        String cod_p = obtenercod_producto(coI);
      PreparedStatement pst = cn.prepareStatement("UPDATE producto SET nom_prod=
'"+TXT_NUEVO_P.getText().toUpperCase()+"' WHERE cod_prod= '"+cod_p+"'");
      pst.executeUpdate();
      JOptionPane.showMessageDialog(null, "Producto Actualizado");
      mostrardatos("");
      TXT_NUEVO_P.setText("");       }
    } catch (Exception e) {
      System.out.print(e.getMessage());  }       }
```

13.3 Eliminar Producto

```
private void EliminarActionPerformed(java.awt.event.ActionEvent evt) {
```

```
int fila = tb_productos.getSelectedRow();
String cod="";
cod=tb_productos.getValueAt(fila, 0).toString();
try {
    PreparedStatement pst = cn.prepareStatement("DELETE FROM producto WHERE
cod_prod='"+cod+"'");
    pst.executeUpdate();
    PreparedStatement dpst = cn.prepareStatement("DELETE FROM local_prod WHERE
cod_prod='"+cod+"'");
    dpst.executeUpdate();
    JOptionPane.showMessageDialog(null, "Producto Eliminado");
    mostrardatos("");
} catch (Exception e) {  }      }
```

## Anexo 14: código fuente Mantenedor Local-Producto

14.1 Agregar Local-Producto

```
String validar_producto(String valor, String valor2){
    String vali = "";
    try {
        String csql = "SELECT * FROM local_prod WHERE cod_prod='"+valor2+"' and cod_local
='"+valor+"'";
        Statement st = cn.createStatement();
        ResultSet rs = st.executeQuery(csql);
        while(rs.next()){
            vali=rs.getString(1); } }
    catch(Exception e){
        System.out.println("Falló la carga del combobox \n"+e);              }
    if (vali.equals("")){
            try {
    String sql = "Select cod_prod from producto where nom_prod = '"+
CB_PRODUCTO.getSelectedItem().toString() +"'";
```

```
    Statement st = cn.createStatement();
    ResultSet rs = st.executeQuery(sql);
    rs.next();
    String ex = rs.getString(1);
    PreparedStatement pstg = cn.prepareStatement("INSERT INTO
LOCAL_PROD(cod_local,cod_prod) VALUES (?,?) ");
    String g;
    g = obtenercod_local(CB_NOMBRE_L.getSelectedItem().toString());
    pstg.setString(1,g);
    pstg.setString(2, ex);
    pstg.executeUpdate();
    JOptionPane.showMessageDialog(null, "Producto Agregado");
    mostrardatosp("");
  } catch (Exception e) {
    System.out.print(e.getMessage());    }
   }else {
     JOptionPane.showMessageDialog(null, "Producto duplicado");       }
    return valor;}
```

14.2 Eliminar Local Producto

```
private void EliminarActionPerformed(java.awt.event.ActionEvent evt) {
  int fila = tb_producto.getSelectedRow();    String cod="";
  cod=tb_producto.getValueAt(fila, 0).toString();
  String cod2="";
  cod2=obtenercod_producto(tb_producto.getValueAt(fila, 1).toString());
  try {
    PreparedStatement pst = cn.prepareStatement("DELETE FROM local_prod WHERE
cod_local='"+cod+"' and cod_prod = '" +cod2+ "'");
    pst.executeUpdate();
    JOptionPane.showMessageDialog(null, "Producto Eliminado");
```

```
    mostrardatosp("");
} catch (Exception e) {    }      }
```