"DESARROLLO DE APLICACIÓN MÓVIL PARA LA POSTPRODUCCIÓN Y COMERCIALIZACIÓN DE LA MORA EN LA REGIÓN DE SANTUARIO, ASOCIACIÓN LA AMOROSA"

PRESENTADO POR:

JULIÁN ANDRÉS TAMAYO VALENCIA

ANDRÉS MAURICIO AGUDELO GUEVARA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
2016

“DESARROLLO DE APLICACIÓN MÓVIL PARA LA POSTPRODUCCIÓN Y COMERCIALIZACIÓN DE LA MORA EN LA REGIÓN DE SANTUARIO, ASOCIACIÓN LA AMOROSA”

PRESENTADO POR:

JULIÁN ANDRÉS TAMAYO VALENCIA

ANDRÉS MAURICIO AGUDELO GUEVARA

DIRECTOR:

CESAR AUGUSTO JARAMILLO ACEVEDO

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA

2016

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En Colombia, la mora sin espinas (*Rubus glaucus Benth*) tiene demanda de consumo por parte de la industria de jugos, la cual exhibe potencial de exportación por sus propiedades nutritivas y antioxidantes[1]. En el territorio nacional, las plantaciones de mora se concentran en su orden, en los departamentos de: Cundinamarca, Santander, Antioquia, Boyacá, Huila y Risaralda. En el departamento de Risaralda a 2013 se contaba con 576 hectáreas (ha) cultivadas de mora de Castilla, con una producción anual de 5138 toneladas (ton) y con un rendimiento de 9.9 ton/ha, por encima del promedio nacional[2].

Teniendo en cuenta el documento "Estrategia de cooperación internacional del sector agropecuario 2013-2015"[3], se encuentran las líneas estratégicas de Productividad dentro de la cual la Comercialización estaría enmarcada en el Fortalecimiento de procesos productivos para la denominación de origen, el Fortalecimiento de sistemas de información para la postproducción y comercialización de vegetales y agroalimentos, y el Fortalecimiento de sistemas de información y de productores. Por otra parte el Ministerio de Tecnologías de la Información y las Comunicaciones MINTIC, en conjunto con el Ministerio de Agricultura y Desarrollo Rural, está llevando a cabo estrategias tales como la AGROTÓN, la cual es una convocatoria de desarrollo dedicada a la generación de soluciones web y móviles para el campo colombiano. Para estos ministerios es de vital importancia generar un ambiente de agregación de valor en el sector agropecuario apoyado en la tecnología, con el fin de atender la demanda de los mercados nacionales e internacionales[4].

A partir de la información anterior, y de las actuales propuestas tecnológicas presentadas por el Ministerio de Agricultura en su Iniciativa con Vive Digital, se pretende realizar un trabajo conjunto entre la parte científica aportada por el grupo de investigación de Oleoquímica y los estudiantes de Ingeniería de Sistemas, ambos de la Universidad Tecnológica de Pereira, mediante los estudios y desarrollo de la iniciativa de la aplicación móvil como medio de postproducción de la región.

Con base en una serie de estudios sobre la mora sin espinas (Rubus glaucus Benth) realizados en la Universidad Tecnológica de Pereira sobre selección de materiales promisorios de mora en

---

[1] Tafur, R., et al., Plan frutícola nacional (PFN). Ministerio de Agricultura y Desarrollo Rural, Fondo Nacional de Fomento Hortofrutícola, Asohofrucol, SAG, 2006: p. 43

[2] Ministerio de Agricultura y Desarrollo Rural. Cadena agroalimentaria de la mora. Bullets Mora 2015.

[3] Martínez, A., Jácome, R., Ayala, G., & Villares, M., Consejos técnicos para el manejo de la mora de castilla (Rubus glaucus Benth). 2014, Bolívar, INIAP (Instituto Nacional de Investigaciones Agropecuarias, EC). p. 114.

[4] AGROTÓN: MARATÓN DE DESARROLLO DE APLICACIONES PARA EL AGRO COLOMBIANO EN CORABASTOS ver:http://estrategia.gobiernoenlinea.gov.co/623/w3article8332.html

Risaralda, su reproducción in vitro (procesos patentados por la UTP), distribución en la mayoría de los municipios de Risaralda (grupo de Biodiversidad y biotecnología) y estudios sobre bioprospección de ingredientes activos con actividad antioxidante realizados por el grupo de investigación de Oleoquímica, se establece que la mora de Castilla producida en el municipio de Santuario presenta características excepcionales[5].

A partir de esto, se tiene como punto de referencia la asociación de productores de mora de Santuario, en la cual se puede evidenciar que los planes de acción con respecto al producto muestran una necesidad latente de postproducción. Para esto se plantea que la aplicación móvil es un mecanismo por medio del cual se haría visible la producción de mora de la Asociación Amorosa en un marco comercial y productivo, que podría tener un alto valor agregado debido a la localización de los cultivos en la reserva natural de Planes de San Rafael.

Basado en la postproducción, se puede lograr que esta mora sea el punto de referencia para dar a conocer la región, fomentando un ecoturismo creciente en la misma. El mercado de la mora podría mejorar en cuanto al aspecto de que los cultivadores, al saber cuál puede ser el destino final de esta mora, pueden darle un tratamiento especial al cultivo según lo exigido por el cliente.

El proyecto de la aplicación móvil empleará los resultados de los estudios de bioprospección de la mora de Santuario (ver Informe técnico Final del grupo de investigación de Oleoquímica), por lo que se demostró que este producto tiene características que lo hacen excepcional, teniendo en cuenta sus perfiles sensoriales, nutricionales y nutracéuticos. Con lo cual se contribuye a que esta asociación pueda ser un Slogan en la región de una posible Denominación de Origen con la mora de Castilla.

2. METODOLOGÍA

- Para hacer análisis de requerimientos para el aplicativo móvil de la postproducción y comercialización de la mora en la región de santuario, asociación la amorosa, se obtuvieron

[5] Informe Técnico Final: BIOPROSPECCION DE METABOLITOS SECUNDARIOS CON VALOR NEUTRACÉUTICO EN LOS MATERIALES CULTIVADOS DE MORA EN EL DEPARTAMENTO DE RISARALDA, 26p

los requerimientos funcionales y no funcionales del sistema mediante el desarrollo de entrevistas.

- Se generan formatos y un análisis de datos, los cuales serán especificados mediante casos de uso. El diagrama de casos de uso describe el comportamiento UML pero mejorado. UML define una notación gráfica para representar estos casos. Dicha notación gráfica describe el comportamiento del sistema al afrontar una regla de negocio o requisito de negocio; esta descripción se enfoca en el valor suministrado por el sistema a entidades externas como los usuarios Humanos o pueden ser otros sistemas.

  Un conjunto de casos de uso deben ser consistentes y coherentes, además deben promover una imagen sencilla de comprender sobre el comportamiento del sistema. En otras palabras los diagramas de casos de uso proveen un entendimiento común entre el cliente, usuario y equipo de desarrollo.

  Un inconveniente que se puede presentar al momento de implementar los casos de uso, se debe a la falta de estandarización de un único modelo o diagrama específico. Debido a que UML define los casos de uso como estandarización de la descripción gráfica del sistema, pero no se define con precisión cual podría ser un diagrama estándar. Dado esto es común que algunas especificaciones complementarias del sistema caigan fuera del ámbito de las descripciones de los casos de uso, como lo son el rendimiento, escalabilidad, gestión, entre otras.

- Para elaborar el diseño y la arquitectura del aplicativo móvil correspondiente a la postproducción y comercialización de la mora en la región de santuario, asociación la amorosa, se adoptó la metodología de desarrollo Scrum para la fase de diseño del sistema, el cual es un proceso de la metodología Ágil la cual se usa para minimizar los riesgos y optimizar la retroalimentación de un proyecto durante su realización, pero teniendo en cuenta que es de manera colaborativa. Entre las ventajas encontradas, están la productividad, calidad y que se realiza un seguimiento de los avances del proyecto, y que las personas interesadas en dicho desarrollo, puedan verificar los avances.

En este sentido podemos considerar Scrum como el paradigma de metodología de desarrollo ágil, definiendo la forma de abordar un proceso de desarrollo de software de forma rápida y liviana, a través de la descripción de un conjunto de roles, componentes y organización de la actividad diaria.

Las características de esta metodología son:

- Ágil: La división del trabajo en pequeñas unidades funcionales (Sprint) permite mantener una política de entregas frecuentes de software que ofrecen una visión clara del estado del proceso y permite la introducción de modificaciones.
- Simple: Se centra especialmente en facilitar el desarrollo rápido, por lo que su complejidad (por ejemplo desde el punto de vista de la documentación a generar o de la organización de equipos) se ha tratado de reducir al máximo.
- Flexible: Todo el desarrollo se contempla como un ciclo de iteraciones continuas de desarrollo, lo que facilita la introducción de modificaciones "sobre la marcha", mejorando continuamente el proceso.
- Colaborativa: El planteamiento, desde el punto de vista de la organización del equipo, resulta bastante horizontal (en contraposición a una organización jerárquica férrea), otorgando a los miembros del equipo de desarrollo una elevado grado de autonomía y auto-organización de su trabajo.

Dentro del desarrollo de la metodología Scrum se deben destacar los siguientes roles que se presentan:

- Product Owner: Es quien tiene contacto directo con el cliente y habla por el cliente y sus necesidades con respecto al proyecto y sus avances, también asegura que el equipo cumpla las expectativas.
- Scrum Master: Lidera las reuniones y ayuda al equipo si es que tienen dificultades. Además, minimiza los obstáculos para cumplir el objetivo del Sprint, siendo un facilitador más no un gestor.
- Scrum Team: Son los encargados de desarrollar y cumplir lo que les asigna el Product Owner.
- Cliente: Recibe el producto y puede influir en el proceso, entregando sus ideas o comentarios respecto al desarrollo.

El proceso dentro de la metodología Scrum se puede representar mediante los siguientes puntos:

- Product Backlog: Es elaborado por el Product Owner y las funciones están priorizadas según los requerimientos más y menos importantes para el proyecto. El objetivo es responder a la pregunta: "¿Qué hay que hacer?".
- Sprint Backlog: Es un subconjunto de ítems del Product Backlog, que son seleccionados por el equipo para realizar durante el Sprint sobre el que se va a trabajar.
- Sprint Planning Meeting: Es una reunión que se hace al comienzo de cada Sprint y se define cómo se va a enfocar el proyecto que viene del Product Backlog las etapas y los plazos. Cada Sprint está compuesto por diferentes etapas de desarrollo.
- Daily Scrum o Stand-up Meeting: Es una reunión breve que se realiza a diario mientras dura el periodo de Sprint. Se responden individualmente tres preguntas: ¿Qué hice ayer?, ¿Qué voy a hacer hoy?, ¿Qué ayuda necesito? El Scrum Master debe tratar de solucionar los problemas u obstáculos que se presenten.
- Sprint Review: Se revisa el sprint terminado, y ya debería haber un avance claro y tangible para presentárselo al cliente.
- Sprint Retrospective: El equipo revisa los objetivos cumplidos del Sprint terminado. Se resalta lo bueno y lo malo, para no volver a repetir los errores. Esta etapa sirve para implementar mejoras desde el punto de vista del proceso del desarrollo.

- Para implementar el aplicativo móvil, basados en la fase de análisis y diseño, el cual se desarrolló con las características modeladas en las fases anteriores bajo la plataforma Android Studio, cuyo lenguaje es basado en Java y JavaScript, junto con esté el framework otorgado por My GoodBarber, framework en la web. Posteriormente el sistema se prueba para verificar la correcta satisfacción de los requerimientos.

3. DESARROLLO

Para el análisis de datos del aplicativo se partió de la información recolectada en la entrevista realizada el día 15 de Febrero de 2016 con el señor Ovidio representante de la Asociación La Amorosa, y en acompañamiento de la Doctora Gloria Guerrero. De esta se toma un texto representativo el cual se tiene cómo presentación de lo que el cliente final desea en el aplicativo:

"Asociación La Amorosa: Eco-conócenos!!

Para empezar a contar sobre nuestra Asociación La Amorosa es conveniente hablar primero de la región, ubicándonos en Santuario y una breve introducción de dos puntos especiales!

***Santuario** es un municipio del departamento de Risaralda (Colombia), ubicado a 64 km de la capital del departamento, en el lado oriental de la cordillera Occidental. Además limita con los municipios de Pueblo Rico, Apía, La Celia y Balboa y los departamentos de Valle del Cauca y Caldas. Fue fundado en 1886, y nació bajo tutela administrativa de Anserma, haciendo parte del estado del Cauca. En 1892 se adhirió al nuevo municipio de San Antonio de Apía, un año más tarde se creó la primera escuela, y en 1894 dejó de ser caserío para ser declarado corregimiento, se convirtió en parroquia en 1906. Fue erigido Municipio en 1907.*

*Cuenta con una población superior a los 15.000 habitantes y en su territorio se encuentra parte del Parque Nacional Natural Tatamá y el **Parque Municipal Natural Planes de San Rafael,** Planes de San Rafel es nuestro punto principal de referencial y por ello una breve introducción: Planes de San Rafael se encuentra a unos 64 km. de la ciudad de Pereira y 11 km. de Santuario en zona rural con una temperatura media de 20C y a una altura de 1600 msnm. Los llamados Planes de San Rafael han sido tradicionalmente la puerta de entrada al Parque Nacional Natural Tatamá, por esta razón se le conoce como "La Perla de Tatamá".*

Un poco de Historia de Planes de San Rafael y el Parque Natural Nacional de Tatamá: El parque Nacional de Tatamá tiene alturas desde 2500 hasta 4250 msnm y fue declarado en 1987 como Parque Nacional. En tanto Planes de San Rafael pasa de tener denominación de Parque Natural Municipal Planes de san Rafael, a Distrito de Manejo integrado y el Parque Nacional de Tatamá además es también un Distrito de Conservación, en el cual se cuenta con una extensión de 51.900 hectáreas en las cuales no debe existir la práctica de la agricultura. Se debe destacar que El Parque Nacional es una de las áreas más conservadas del país y del planeta, tanto así que en el

año 2014 ganó un reconocimiento en La Lista verde del mundo por su excelente estado de conservación.

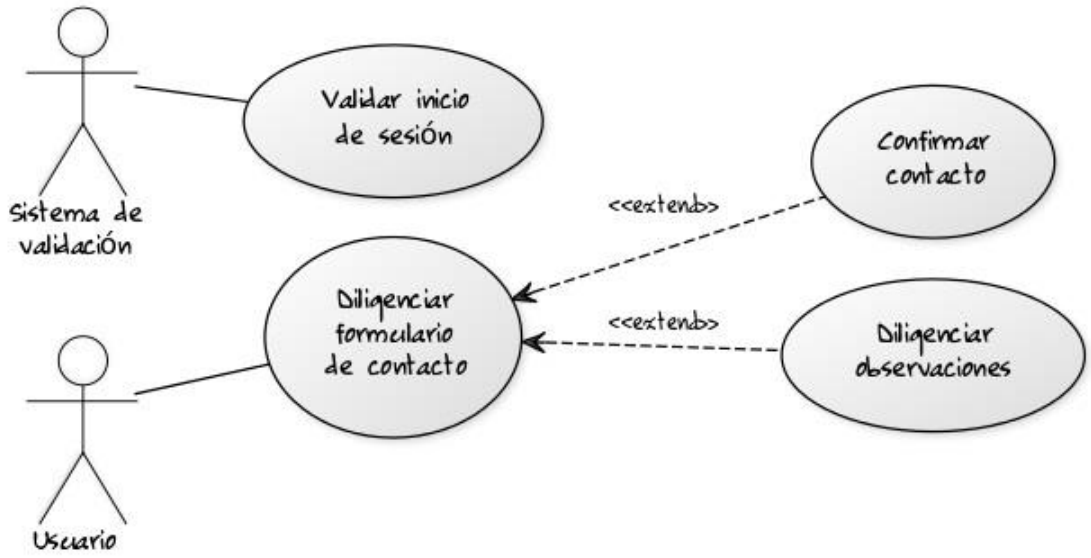
Planes de San Rafael se declara como área protegida en el año de 1995 en cuyo momento contaba con 107 hectáreas (Hoy por hoy el Distrito de manejo Integrado Planes de San Rafael cuenta con 510 hectáreas). Se debe destacar como punto muy importante que Planes de San Rafael es la única entrada y salida al parque, lo que convierte a Planes en uno de los factores para el alto nivel de conservación del Páramo, teniendo claro que Planes de San Rafael está estratégicamente ubicado para el control a la zona del mismo, siendo su objetivo principal el de conservar la cuenca media del río San Rafael que aflora en el parque Tatamá. La iniciativa de conservación de esta cuenca nace debido a que en ese momento se tenían cultivos a los 2.800 msnm dentro del páramo, y al volverse Tatamá un Parque Nacional Natural, se debe frenar el proceso de extensión de la agricultura.

La reserva natural Planes de San Rafael está entre los 2.000 y los 2.500 msnm (zona amortiguadora) el objetivo de la figura de Parque Municipal de conservación, es la protección, educación: Eco-turismo e investigación. Teniendo esto claro, en el paso del tiempo se presenta una propuesta a la CARDER para que la zona sea nombrada como Distrito de Manejo Integrado, porque dicha figura permite realizar las mismas actividades que la figura de Parque Municipal (mencionadas anteriormente) y además agregarle la agricultura sostenible. Esto porque se tiene la convicción entre los habitantes de Planes de San Rafael de que un área protegida no debe ser la causal del desplazamiento de las personas que allí habiten, si no permitir que sean autosostenibles y acordes con la conservación teniendo en cuenta el equilibrio entre ésta y la producción.

Teniendo en cuenta el nombramiento de Planes de San Rafael como Distrito de Manejo Integrado, Nace la asociación La Amorosa en la zona Ecoturística de Planes, y se da pie a un proceso de agricultura autosostenible y amigable con la reserva. Para ello se piensa y se lleva a cabo plantaciones con La Mora. Este fruto de características excepcionales se convierte en un cultivo que, en comparación con otros de la misma región, es el menos afectado por las plagas e insectos, los cuales no han asediado de manera considerable el fruto. Gracias a esto ha desencadenado que los productos agroquímicos que se utilizan para los cuidados del cultivo, sean los más amigables con el medio ambiente. Como consecuencia de todo lo anterior se podría destacar a la Mora de

Planes de San Rafael como un fruto amigable con el ambiente que posee características que lo hacen excepcional.”[6]

Teniendo como punto de partida lo encontrado en la entrevista realizada, los siguientes son los casos de uso que se pudieron obtener para la aplicación:



Como se concluye, el aplicativo es altamente informativo donde lo que se busca es que la Asociación, la zona de Planes de san Rafael y el producto principal que son las moras, sean presentadas hacia un nuevo mercado, tanto para la mora como para el ecoturismo que se encuentra en Planes de San Rafael.

Para estos casos de uso se replantea la validación de usuario, que en conjunto con el cliente final, se concluye que no es necesario, puesto que al ser informativa, debe llegar a la mayor cantidad de público posible; encontrando en la investigación la necesidad de abarcar la mayor cantidad de consumidores de la aplicación posible, por esta razón se descarta la validación de usuario, acordando con el cliente final que es mejor dejar abierta a todo público. Por ende se deja el sistema

[6] Entrevista a Ovidio Ledesma representante Asociación Amorosa-Planes de san Rafael

de validación como un sistema externo al actual, que se podrá tener en cuenta en futuras actualizaciones de la aplicación

Teniendo esto en cuenta la estructura de los casos de uso es la siguiente:

Código:	CU_02		
Nombre:	Diligenciar formulario de contacto		
Creado por:		**Actualizado por:**	
Fecha de Creación:		**Fecha de última actualización:**	
Actores:	Usuario		
Descripción:	El Usuario completa los campos requeridos para el contacto		
Disparador:	El usuario ingresa en la ventana de contacto		
Pre-condiciones:	1. El usuario ingresa en la ventana de contacto		
Post-condiciones:	1. El usuario envía datos y observaciones de contacto de forma exitosa		
Flujo Normal:	1. El usuario ingresa en la ventana de contacto 2. El usuario diligencia campo de Nombre 3. El usuario diligencia campo de Email 4. El usuario diligencia número de contacto 5. El usurario diligencia el campo de mensaje 6. El usuario da clic en enviar		

Flujos Alternativos:	4a. En el paso 4 del flujo normal cuando el sistema valida el campo E-mail 1. El sistema valida que el correo sea valido 2. Se retorna al paso 3 del flujo normal
Excepciones:	6a. En el paso 6 del flujo normal cuando el usuario da clic en enviar: 1. El sistema valida que estén llenos los campos requeridos 3. El sistema no permite el envío del contacto
Asunciones o supuestos:	1. El usuario debe contar con un E-mail
Notas and Comentarios:	

Código:	CU_03		
Nombre:	Confirmar contacto		
Creado por:		**Actualizado por:**	
Fecha de Creación:		**Fecha de última actualización:**	
Actores:	Usuario		
Descripción:	El Usuario completa los campos requeridos para el contacto		
Disparador:	El usuario ingresa en la ventana de contacto		
Pre-condiciones:	1. El usuario ingresa en la ventana de contacto		

Post-condiciones:	1. El usuario envía datos y observaciones de contacto de forma exitosa
Flujo Normal:	1. El usuario ingresa en la ventana de contacto 2. El usuario diligencia campo de Nombre 3. El usuario diligencia campo de Email 4. El usuario diligencia número de contacto 5. El usurario diligencia el campo de mensaje 6. El usuario da clic en enviar
Flujos Alternativos:	4a. En el paso 4 del flujo normal cuando el sistema valida el campo E-mail 1. El sistema valida que el correo sea valido 2. Se retorna al paso 3 del flujo normal
Excepciones:	6a. En el paso 6 del flujo normal cuando el usuario da clic en enviar: 1. El sistema valida que estén llenos los campos requeridos 3. El sistema no permite el envío del contacto
Asunciones o supuestos:	1. El usuario debe contar con un E-mail
Notas and Comentarios:	

Para el desarrollo del aplicativo se utilizó la herramienta web My GoodBarber como framework y herramienta para el diseño de la aplicación web, tanto en vistas como en la información en general que se presenta en la aplicación.

Uno de los aspectos más importantes de todas las aplicaciones es el diseño. Precisamente por él empieza el proceso de construcción de aplicaciones en GoodBarber. Se posee una biblioteca de más de 100 temas, plantillas personalizables y estilos distintos. La interfaz es muy intuitiva y permite cambiar muchísimos detalles hasta conseguir una app bonita, sencilla. La herramienta dispone de un CMS interno rehecho desde cero que permite crear todo el contenido directamente desde el back office de la aplicación. Con el nuevo CMS se pueden crear: artículos, vídeos, imágenes y ahora audio. Para ofrecer una navegación completa, sin necesidad de conectar con fuentes externas. En el cual se puede añadir contenido exclusivo para la aplicación y/o conectar fuentes externas y ordenar todo esto de forma coherente por secciones y cada sección cuenta con su plantilla especifica adaptada al contenido.

Adicional, la aplicación tiene su propio sistema de estadísticas y de igual forma se puede añadir Google Analytics o Flurry. También se puede elegir añadir o no publicidad externa y/o interna, en formato banner o splash screen y especificar el orden de aparición. La aplicación tiene el formato nativo para IOs y Android pero también se crea una versión HTML5 el cual se puede vincular con un nombre de dominio propio para que sea accesible directamente desde la web como versión móvil de una página web. GoodBarber termina siendo una herramienta para crear una aplicación de estilo profesional en Android (o en iPad como aplicación nativa). Gracias al uso de API’s de Google, se logra que al tener estos puntos de georreferenciación, también se puedan generar las diferentes rutas de acceso, que la aplicación maps de Google tiene implementada.

Otra herramienta que se tuvo para el desarrollo del aplicativo móvil, es Android Studio, herramienta para el desarrollo de estos aplicativos de forma nativa. En la etapa de desarrollo del aplicativo móvil, se tiene como referente el framework My GoodBarber, el cual es una plataforma online, utilizado para el diseño de las diferentes vistas y contenidos que se presentaron en el desarrollo de la misma.

Al finalizar el proceso con este framework, se descargó la ".apk" que se genera, y mediando el uso de herramientas como Java Decompiler, se le hizo ingeniería inversa a la misma, esto con el fin de obtener el código en java de la aplicación.

Para realizar el proceso de ingeniería inversa a la .apk generada, se debe tener en cuenta cuales son las clases que se ven afectadas en el proceso. Gracias a la herramienta de Java Decompiler, se obtiene un archivo ".dex", el cual contiene todo el paquete de las clases y donde se realizaron los cambios oportunos en programación, los cuales estuvieron enfocados principalmente en el uso de Georreferenciación. En estas vistas que estarán en los anexos, se colocaron referencias de GPS de los diferentes cultivos que hacen parte de la Asociación La Amorosa, y de igual forma se tuvo en cuenta la referenciación de Santuario como punto inicial.

La ingeniería inversa se pudo verificar mediante Android (IDE desarrollado por Google), el cual está especializado para el desarrollo de aplicaciones en Android, basado en IntelliJ IDEA, el cual es un IDE de java. Este IDE ofrece la posibilidad de ver los cambios al diseño de las aplicaciones en las diferentes resoluciones que puede soportar Android. Al igual que información útil del código como iconos o colores que relacionan la mención de algún apartado en cualquier parte del mismo código.

Esto se logra mediante Java y JavaScript; en el cual Java es un lenguaje de programación orientado a objetos. El objetivo era utilizarlo en un set-top box, un tipo de dispositivo que se encarga de la recepción y la descodificación de la señal televisiva. El primer nombre del lenguaje fue Oak, luego se conoció como Green y finamente adoptó la denominación de Java. La intención de Sun era crear un lenguaje con una estructura y una sintaxis similar a C y C++, aunque con un modelo de objetos más simple y eliminando las herramientas de bajo nivel. Los pilares en los que se sustenta Java son cinco: la programación orientada a objetos, la posibilidad de ejecutar un mismo programa en diversos sistemas operativos, la inclusión por defecto de soporte para trabajo en red, la opción de ejecutar el código en sistemas remotos de manera segura y la facilidad de uso. JavaScript Al igual que HTML, es un lenguaje de programación que se puede utilizar para construir sitios Web y para hacerlos más interactivos. Aunque comparte muchas de las características y de las estructuras del lenguaje Java, fue desarrollado independientemente. El lenguaje Javascript puede interactuar con el código HTML, permitiendo a los programadores web utilizar contenido dinámico.

En el desarrollo del aplicativo móvil se presenta el siguiente código fuente con las clases más representativas, las demás van como anexo:

MULTIDEX.JAVA

```
package android.support.multidex;

import android.content.Context;

import android.content.pm.ApplicationInfo;

import android.content.pm.PackageManager;

import android.content.pm.PackageManager.NameNotFoundException;

import android.os.Build.VERSION;

import android.support.v4.app.NotificationCompat;

import android.util.Log;

import dalvik.system.DexFile;

import java.io.File;

import java.io.IOException;

import java.lang.reflect.Array;

import java.lang.reflect.Field;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashSet;

import java.util.Iterator;

import java.util.List;

import java.util.ListIterator;

import java.util.Set;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.util.zip.ZipFile;


public final class MultiDex {

    private static final boolean IS_VM_MULTIDEX_CAPABLE;

    private static final String SECONDARY_FOLDER_NAME;

    private static final Set<String> installedApk;


    private static final class V14 {

        private static void install(ClassLoader loader, List<File> additionalClassPathEntries, File optimizedDirectory) throws IllegalArgumentException, IllegalAccessException, NoSuchFieldException, InvocationTargetException, NoSuchMethodException {

            Object dexPathList = MultiDex.findField(loader, "pathList").get(loader);


            MultiDex.expandFieldArray(dexPathList, "dexElements", makeDexElements(dexPathList, new ArrayList(additionalClassPathEntries), optimizedDirectory));

        }


        private static Object[] makeDexElements(Object dexPathList, ArrayList<File> files, File optimizedDirectory) throws IllegalAccessException, InvocationTargetException, NoSuchMethodException {

            return (Object[]) MultiDex.findMethod(dexPathList, "makeDexElements", ArrayList.class, File.class).invoke(dexPathList, new Object[]{files, optimizedDirectory});

        }

    }
```

```
    private static final class V19 {

        private static void install(ClassLoader loader, List<File> additionalClassPathEntries, File optimizedDirectory) throws IllegalArgumentException, IllegalAccessException, NoSuchFieldException, InvocationTargetException, NoSuchMethodException {

            Object dexPathList = MultiDex.findField(loader, "pathList").get(loader);

            ArrayList<IOException> suppressedExceptions = new ArrayList();


MultiDex.expandFieldArray(dexPathList, "dexElements", makeDexElements(dexPathList, new ArrayList(additionalClassPathEntries), optimizedDirectory, suppressedExceptions));

            if (suppressedExceptions.size() > 0) {

                Iterator i$ = suppressedExceptions.iterator();

                while (i$.hasNext()) {

                    Log.w("MultiDex", "Exception in makeDexElement", (IOException) i$.next());

                }

                Field suppressedExceptionsField = MultiDex.findField(loader, "dexElementsSuppressedExceptions");

                IOException[] dexElementsSuppressedExceptions = (IOException[]) suppressedExceptionsField.get(loader);

                if (dexElementsSuppressedExceptions == null) {


dexElementsSuppressedExceptions = (IOException[]) suppressedExceptions.toArray(new IOException[suppressedExceptions.size()]);

                } else {

                    IOException[] combined = new IOException[(suppressedExceptions.size() + dexElementsSuppressedExceptions.length)];


suppressedExceptions.toArray(combined);


System.arraycopy(dexElementsSuppressedExceptions, 0, combined, suppressedExceptions.size(), dexElementsSuppressedExceptions.length);


dexElementsSuppressedExceptions = combined;

                }


suppressedExceptionsField.set(loader, dexElementsSuppressedExceptions);

            }

        }


        private static Object[] makeDexElements(Object dexPathList, ArrayList<File> files, File optimizedDirectory, ArrayList<IOException> suppressedExceptions) throws IllegalAccessException, InvocationTargetException, NoSuchMethodException {

            return (Object[]) MultiDex.findMethod(dexPathList, "makeDexElements", ArrayList.class, File.class, ArrayList.class).invoke(dexPathList, new Object[]{files, optimizedDirectory, suppressedExceptions});

        }

    }


    private static final class V4 {

        private static void install(ClassLoader loader, List<File> additionalClassPathEntries) throws IllegalArgumentException, IllegalAccessException,
```

```
NoSuchFieldException,
IOException {

        int extraSize =
additionalClassPathEntries.size();

        Field pathField =
MultiDex.findField(loader, "path");

        StringBuilder path = new
StringBuilder((String)
pathField.get(loader));

        String[] extraPaths = new
String[extraSize];

        File[] extraFiles = new
File[extraSize];

        ZipFile[] extraZips = new
ZipFile[extraSize];

        DexFile[] extraDexs = new
DexFile[extraSize];

        ListIterator<File> iterator =
additionalClassPathEntries.listIterat
or();

        while (iterator.hasNext()) {

            File additionalEntry =
(File) iterator.next();

            String entryPath =
additionalEntry.getAbsolutePath();


path.append(':').append(entryPath);

            int index =
iterator.previousIndex();

            extraPaths[index] =
entryPath;

            extraFiles[index] =
additionalEntry;

            extraZips[index] = new
ZipFile(additionalEntry);

            extraDexs[index] =
DexFile.loadDex(entryPath,
entryPath + ".dex", 0);

        }

        pathField.set(loader,
path.toString());


MultiDex.expandFieldArray(loader,
"mPaths", extraPaths);


MultiDex.expandFieldArray(loader,
"mFiles", extraFiles);


MultiDex.expandFieldArray(loader,
"mZips", extraZips);


MultiDex.expandFieldArray(loader,
"mDexs", extraDexs);

    }

  }


  static {


SECONDARY_FOLDER_NAME =
"code_cache" + File.separator +
"secondary-dexes";

    installedApk = new HashSet();


IS_VM_MULTIDEX_CAPABLE =
isVMMultidexCapable(System.getP
roperty("java.vm.version"));

  }


  public static void install(Context
context) {

    Log.i("MultiDex", "install");

    if
(IS_VM_MULTIDEX_CAPABLE)
{

        Log.i("MultiDex", "VM has
multidex support, MultiDex support
library is disabled.");

    } else if (VERSION.SDK_INT
< 4) {

        throw new
RuntimeException("Multi dex
installation failed. SDK " +
VERSION.SDK_INT + " is
unsupported. Min SDK version is " +
4 + ".");

    } else {

        try {

            ApplicationInfo
applicationInfo =
getApplicationInfo(context);

            if (applicationInfo != null)
{

                synchronized
(installedApk) {
```

```
            String apkPath = applicationInfo.sourceDir;

            if (installedApk.contains(apkPath)) {
                return;
            }

            installedApk.add(apkPath);

            if (VERSION.SDK_INT > 20) {

                Log.w("MultiDex", "MultiDex is not guaranteed to work in SDK version " + VERSION.SDK_INT + ": SDK version higher than " + 20 + " should be backed by " + "runtime with built-in multidex capabilty but it's not the " + "case here: java.vm.version=\"" + System.getProperty("java.vm.version") + "\"");
            }

            try {
                ClassLoader loader = context.getClassLoader();
                if (loader == null) {

                    Log.e("MultiDex", "Context class loader is null. Must be running in test mode. Skip patching.");
                    return;
                }
                try {
                    clearOldDexDir(context);
                } catch (Throwable t) {

                    Log.w("MultiDex", "Something went wrong when trying to clear old MultiDex extraction, continuing without cleaning.", t);
                }

                File dexDir = new File(applicationInfo.dataDir, SECONDARY_FOLDER_NAME);

                List<File> files = MultiDexExtractor.load(context, applicationInfo, dexDir, false);

                if (checkValidZipFiles(files)) {

                    installSecondaryDexes(loader, dexDir, files);

                } else {

                    Log.w("MultiDex", "Files were not valid zip files.  Forcing a reload.");

                    files = MultiDexExtractor.load(context, applicationInfo, dexDir, true);

                    if (checkValidZipFiles(files)) {

                        installSecondaryDexes(loader, dexDir, files);

                    } else {

                        throw new RuntimeException("Zip files were not valid.");

                    }

                }

                Log.i("MultiDex", "install done");

                return;

            } catch (RuntimeException e) {

                Log.w("MultiDex", "Failure while trying to obtain Context class loader. Must be running in test mode. Skip patching.", e);

                }

            }

        } catch (Exception e2) {

            Log.e("MultiDex", "Multidex installation failure", e2);

            throw new RuntimeException("Multi dex installation failed (" + e2.getMessage() + ").");

        }

    }
```

```
    private static ApplicationInfo getApplicationInfo(Context context) throws NameNotFoundException {

        try {

            PackageManager pm = context.getPackageManager();

            String packageName = context.getPackageName();

            if (pm == null || packageName == null) {

                return null;

            }

            return pm.getApplicationInfo(packageName, NotificationCompat.FLAG_HIGH_PRIORITY);

        } catch (RuntimeException e) {

            Log.w("MultiDex", "Failure while trying to obtain ApplicationInfo from Context. Must be running in test mode. Skip patching.", e);

            return null;

        }

    }



    static boolean isVMMultidexCapable(String versionString) {

        boolean isMultidexCapable = false;

        if (versionString != null) {

            Matcher matcher = Pattern.compile("(\\d+)\\.(\\d+)(\\.\\d+)?").matcher(versionString);

            if (matcher.matches()) {

                try {

                    int major = Integer.parseInt(matcher.group(1));

                    isMultidexCapable = major > 2 || (major == 2 && Integer.parseInt(matcher.group(2)) >= 1);

                } catch (NumberFormatException e) {

                }

            }

        }

        Log.i("MultiDex", "VM with version " + versionString + (isMultidexCapable ? " has multidex support" : " does not have multidex support"));

        return isMultidexCapable;

    }



    private static void installSecondaryDexes(ClassLoader loader, File dexDir, List<File> files) throws IllegalArgumentException, IllegalAccessException, NoSuchFieldException, InvocationTargetException, NoSuchMethodException, IOException {

        if (!files.isEmpty()) {

            if (VERSION.SDK_INT >= 19) {

                V19.install(loader, files, dexDir);

            } else if (VERSION.SDK_INT >= 14) {

                V14.install(loader, files, dexDir);

            } else {

                V4.install(loader, files);

            }

        }

    }



    private static boolean checkValidZipFiles(List<File> files) {

        for (File file : files) {

            if (!MultiDexExtractor.verifyZipFile(file)) {

                return false;

            }

        }

        return true;

    }
```

```
    private static Field findField(Object instance, String name) throws NoSuchFieldException {
        Class<?> clazz = instance.getClass();
        while (clazz != null) {
            try {
                Field field = clazz.getDeclaredField(name);
                if (!field.isAccessible()) {
                    field.setAccessible(true);
                }
                return field;
            } catch (NoSuchFieldException e) {
                clazz = clazz.getSuperclass();
            }
        }
        throw new NoSuchFieldException("Field " + name + " not found in " + instance.getClass());
    }

    private static Method findMethod(Object instance, String name, Class<?>... parameterTypes) throws NoSuchMethodException {
        Class<?> clazz = instance.getClass();
        while (clazz != null) {
            try {
                Method method = clazz.getDeclaredMethod(name, parameterTypes);
                if (!method.isAccessible()) {
                    method.setAccessible(true);
                }
                return method;
            } catch (NoSuchMethodException e) {
                clazz = clazz.getSuperclass();
            }
        }
        throw new NoSuchMethodException("Method " + name + " with parameters " + Arrays.asList(parameterTypes) + " not found in " + instance.getClass());
    }

    private static void expandFieldArray(Object instance, String fieldName, Object[] extraElements) throws NoSuchFieldException, IllegalArgumentException, IllegalAccessException {
        Field jlrField = findField(instance, fieldName);
        Object[] original = (Object[]) jlrField.get(instance);
        Object[] combined = (Object[]) Array.newInstance(original.getClass().getComponentType(), original.length + extraElements.length);
        System.arraycopy(original, 0, combined, 0, original.length);
        System.arraycopy(extraElements, 0, combined, original.length, extraElements.length);
        jlrField.set(instance, combined);
    }

    private static void clearOldDexDir(Context context) throws Exception {
        File dexDir = new File(context.getFilesDir(), "secondary-dexes");
        if (dexDir.isDirectory()) {
            Log.i("MultiDex", "Clearing old secondary dex dir (" + dexDir.getPath() + ").");
```

```
        File[] files = dexDir.listFiles();
        if (files == null) {
            Log.w("MultiDex", "Failed to list secondary dex dir content (" + dexDir.getPath() + ").");
            return;
        }
        for (File oldFile : files) {
            Log.i("MultiDex", "Trying to delete old file " + oldFile.getPath() + " of size " + oldFile.length());
            if (oldFile.delete()) {
                Log.i("MultiDex", "Deleted old file " + oldFile.getPath());
            } else {
                Log.w("MultiDex", "Failed to delete old file " + oldFile.getPath());
            }
        }
        if (dexDir.delete()) {
            Log.i("MultiDex", "Deleted old secondary dex dir " + dexDir.getPath());
        } else {
            Log.w("MultiDex", "Failed to delete secondary dex dir " + dexDir.getPath());
        }
    }
}
```

MULTIDEXAPPLICATION.JAVA

```
package android.support.multidex;

import android.app.Application;

import android.content.Context;


public class MultiDexApplication extends Application {

   protected void attachBaseContext(Context base) {

      super.attachBaseContext(base);

      MultiDex.install(this);

   }

}
```

MULTIDEXEXTRACTOR.JAVA

```
package android.support.multidex;

import android.content.Context;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.content.pm.ApplicationInfo;

import android.os.Build.VERSION;

import android.util.Log;

import com.facebook.GraphResponse;
```

```
import java.io.BufferedOutputStream;
import java.io.Closeable;
import java.io.File;
import java.io.FileFilter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
import java.util.zip.ZipEntry;
import java.util.zip.ZipFile;
import java.util.zip.ZipOutputStream;

final class MultiDexExtractor {
    private static Method sApplyMethod;

    /* renamed from: android.support.multidex.MultiDexExtractor.1 */
    static class C00001 implements FileFilter {
        final /* synthetic */ String val$extractedFilePrefix;

        C00001(String str) {
            this.val$extractedFilePrefix = str;
        }

        public boolean accept(File pathname) {
            return !pathname.getName().startsWith(this.val$extractedFilePrefix);
        }
    }

    static List<File> load(Context context, ApplicationInfo applicationInfo, File dexDir, boolean forceReload) throws IOException {
        List<File> files;
        Log.i("MultiDex", "MultiDexExtractor.load(" + applicationInfo.sourceDir + ", " + forceReload + ")");
        File sourceApk = new File(applicationInfo.sourceDir);
        long currentCrc = getZipCrc(sourceApk);
        if (forceReload || isModified(context, sourceApk, currentCrc)) {
            Log.i("MultiDex", "Detected that extraction must be performed.");
            files = performExtractions(sourceApk, dexDir);
            putStoredApkInfo(context, getTimeStamp(sourceApk), currentCrc, files.size() + 1);
        } else {
            try {
                files = loadExistingExtractions(context, sourceApk, dexDir);
            } catch (IOException ioe) {
                Log.w("MultiDex", "Failed to reload existing extracted secondary dex files, falling back to fresh extraction", ioe);
                files = performExtractions(sourceApk, dexDir);
                putStoredApkInfo(context, getTimeStamp(sourceApk), currentCrc, files.size() + 1);
            }
        }
        Log.i("MultiDex", "load found " + files.size() + " secondary dex files");
        return files;
    }

    private static List<File> loadExistingExtractions(Context context, File sourceApk, File dexDir) throws IOException {
        Log.i("MultiDex", "loading existing secondary dex files");
        String extractedFilePrefix = sourceApk.getName() + ".classes";
        int totalDexNumber = getMultiDexPreferences(context).getInt("dex.number", 1);
        List<File> files = new ArrayList(totalDexNumber);
        int secondaryNumber = 2;
```

```
        while (secondaryNumber <= totalDexNumber) {
            File extractedFile = new File(dexDir, extractedFilePrefix + secondaryNumber + ".zip");
            if (extractedFile.isFile()) {
                files.add(extractedFile);
                if (verifyZipFile(extractedFile)) {
                    secondaryNumber++;
                } else {
                    Log.i("MultiDex", "Invalid zip file: " + extractedFile);
                    throw new IOException("Invalid ZIP file.");
                }
            }
            throw new IOException("Missing extracted secondary dex file '" + extractedFile.getPath() + "'");
        }
        return files;
    }

    private static boolean isModified(Context context, File archive, long currentCrc) {
        SharedPreferences prefs = getMultiDexPreferences(context);
        return (prefs.getLong("timestamp", -1) == getTimeStamp(archive) && prefs.getLong("crc", -1) == currentCrc) ? false : true;
    }

    private static long getTimeStamp(File archive) {
        long timeStamp = archive.lastModified();
        if (timeStamp == -1) {
            return timeStamp - 1;
        }
        return timeStamp;
    }

    private static long getZipCrc(File archive) throws IOException {
        long computedValue = ZipUtil.getZipCrc(archive);
        if (computedValue == -1) {
            return computedValue - 1;
        }
        return computedValue;
    }

    private static List<File> performExtractions(File sourceApk, File dexDir) throws IOException {
        String extractedFilePrefix = sourceApk.getName() + ".classes";
        prepareDexDir(dexDir, extractedFilePrefix);
        List<File> files = new ArrayList();
        ZipFile apk = new ZipFile(sourceApk);
        int secondaryNumber = 2;
        try {
            ZipEntry dexFile = apk.getEntry("classes" + 2 + ".dex");
            while (dexFile != null) {
                File extractedFile = new File(dexDir, extractedFilePrefix + secondaryNumber + ".zip");
                files.add(extractedFile);
                Log.i("MultiDex", "Extraction is needed for file " + extractedFile);
                int numAttempts = 0;
                boolean isExtractionSuccessful = false;
                while (numAttempts < 3 && !isExtractionSuccessful) {
                    numAttempts++;
                    extract(apk, dexFile, extractedFile, extractedFilePrefix);
                    isExtractionSuccessful = verifyZipFile(extractedFile);
                    Log.i("MultiDex", "Extraction " + (isExtractionSuccessful ? GraphResponse.SUCCESS_KEY : "failed") + " - length " + extractedFile.getAbsolutePath() + ": " + extractedFile.length());
                    if (!isExtractionSuccessful) {
                        extractedFile.delete();
                        if (extractedFile.exists()) {
```

```
                    Log.w("MultiDex",
"Failed to delete corrupted secondary dex
'" + extractedFile.getPath() + "'");
                }
            }
        }
        if (isExtractionSuccessful) {
            secondaryNumber++;
            dexFile =
apk.getEntry("classes" +
secondaryNumber + ".dex");
        } else {
            throw new
IOException("Could not create zip file "
+ extractedFile.getAbsolutePath() + " for
secondary dex (" + secondaryNumber +
")");
        }
      }
      return files;
    } finally {
      try {
        apk.close();
      } catch (IOException e) {
        Log.w("MultiDex", "Failed to
close resource", e);
      }
    }
  }

  private static void
putStoredApkInfo(Context context, long
timeStamp, long crc, int
totalDexNumber) {
    Editor edit =
getMultiDexPreferences(context).edit();
    edit.putLong("timestamp",
timeStamp);
    edit.putLong("crc", crc);
    edit.putInt("dex.number",
totalDexNumber);
    apply(edit);
  }

  private static SharedPreferences
getMultiDexPreferences(Context
context) {
    return
context.getSharedPreferences("multidex.
version", VERSION.SDK_INT < 11 ? 0 :
4);
  }

  private static void prepareDexDir(File
dexDir, String extractedFilePrefix)
throws IOException {

mkdirChecked(dexDir.getParentFile());
    mkdirChecked(dexDir);
    File[] files = dexDir.listFiles(new
C00001(extractedFilePrefix));
    if (files == null) {
      Log.w("MultiDex", "Failed to list
secondary dex dir content (" +
dexDir.getPath() + ").");
      return;
    }
    for (File oldFile : files) {
      Log.i("MultiDex", "Trying to
delete old file " + oldFile.getPath() + " of
size " + oldFile.length());
      if (oldFile.delete()) {
        Log.i("MultiDex", "Deleted old
file " + oldFile.getPath());
      } else {
        Log.w("MultiDex", "Failed to
delete old file " + oldFile.getPath());
      }
    }
  }

  private static void mkdirChecked(File
dir) throws IOException {
    dir.mkdir();
    if (!dir.isDirectory()) {
      File parent = dir.getParentFile();
      if (parent == null) {
        Log.e("MultiDex", "Failed to
create dir " + dir.getPath() + ". Parent file
is null.");
      } else {
        Log.e("MultiDex", "Failed to
create dir " + dir.getPath() + ". parent file
is a dir " + parent.isDirectory() + ", a file
" + parent.isFile() + ", exists " +
parent.exists() + ", readable " +
parent.canRead() + ", writable " +
parent.canWrite());
      }
```

```
            throw new IOException("Failed to create cache directory " + dir.getPath());
        }
    }

    private static void extract(ZipFile apk, ZipEntry dexFile, File extractTo, String extractedFilePrefix) throws IOException, FileNotFoundException {
        Throwable th;
        InputStream in = apk.getInputStream(dexFile);
        File tmp = File.createTempFile(extractedFilePrefix, ".zip", extractTo.getParentFile());
        Log.i("MultiDex", "Extracting " + tmp.getPath());
        try {
            ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(new FileOutputStream(tmp)));
            try {
                ZipEntry classesDex = new ZipEntry("classes.dex");

                classesDex.setTime(dexFile.getTime());
                out.putNextEntry(classesDex);
                byte[] buffer = new byte[16384];
                for (int length = in.read(buffer); length != -1; length = in.read(buffer)) {
                    out.write(buffer, 0, length);
                }
                out.closeEntry();
                out.close();
                Log.i("MultiDex", "Renaming to " + extractTo.getPath());
                if (tmp.renameTo(extractTo)) {
                    closeQuietly(in);
                    tmp.delete();
                    return;
                }
                throw new IOException("Failed to rename \"" + tmp.getAbsolutePath() + "\" to \"" + extractTo.getAbsolutePath() + "\"");
            } catch (Throwable th2) {
                th = th2;
                ZipOutputStream zipOutputStream = out;
                closeQuietly(in);
                tmp.delete();
                throw th;
            }
        } catch (Throwable th3) {
            th = th3;
            closeQuietly(in);
            tmp.delete();
            throw th;
        }
    }

    /* JADX WARNING: inconsistent code. */
    /* Code decompiled incorrectly, please refer to instructions dump. */
    static boolean verifyZipFile(java.io.File r6) {
        /*
        r2 = new java.util.zip.ZipFile; Catch:{ ZipException -> 0x0029, IOException -> 0x004d }
        r2.<init>(r6); Catch:{ ZipException -> 0x0029, IOException -> 0x004d }
        r2.close(); Catch:{ IOException -> 0x000a, ZipException -> 0x0029 }
        r3 = 1;
    L_0x0009:
        return r3;
    L_0x000a:
        r0 = move-exception;
        r3 = "MultiDex";
        r4 = new java.lang.StringBuilder; Catch:{ ZipException -> 0x0029, IOException -> 0x004d }
        r4.<init>(); Catch:{ ZipException -> 0x0029, IOException -> 0x004d }
        r5 = "Failed to close zip file: ";
        r4 = r4.append(r5); Catch:{ ZipException -> 0x0029, IOException -> 0x004d }
        r5 = r6.getAbsolutePath(); Catch:{ ZipException -> 0x0029, IOException -> 0x004d }
```

```
        r4 = r4.append(r5);             Catch:{
ZipException -> 0x0029, IOException ->
0x004d }

        r4 = r4.toString();             Catch:{
ZipException -> 0x0029, IOException ->
0x004d }

        android.util.Log.w(r3, r4);     Catch:{
ZipException -> 0x0029, IOException ->
0x004d }

    L_0x0027:

        r3 = 0;

        goto L_0x0009;

    L_0x0029:

        r1 = move-exception;

        r3 = "MultiDex";

        r4 = new java.lang.StringBuilder;

        r4.<init>();

        r5 = "File ";

        r4 = r4.append(r5);

        r5 = r6.getAbsolutePath();

        r4 = r4.append(r5);

        r5 = " is not a valid zip file.";

        r4 = r4.append(r5);

        r4 = r4.toString();

        android.util.Log.w(r3, r4, r1);

        goto L_0x0027;

    L_0x004d:

        r1 = move-exception;

        r3 = "MultiDex";

        r4 = new java.lang.StringBuilder;

        r4.<init>();

        r5 = "Got an IOException trying to
open zip file: ";

        r4 = r4.append(r5);

        r5 = r6.getAbsolutePath();

        r4 = r4.append(r5);

        r4 = r4.toString();

        android.util.Log.w(r3, r4, r1);

        goto L_0x0027;

        */

        throw new
UnsupportedOperationException("Metho
d not decompiled:
android.support.multidex.MultiDexExtra
ctor.verifyZipFile(java.io.File):boolean")
;

    }


    private static void
closeQuietly(Closeable closeable) {

        try {

            closeable.close();

        } catch (IOException e) {

            Log.w("MultiDex", "Failed to
close resource", e);

        }

    }

    static {

        try {

            sApplyMethod =
Editor.class.getMethod("apply", new
Class[0]);

        } catch (NoSuchMethodException
e) {

            sApplyMethod = null;

        }

    }


    private static void apply(Editor editor)
{

        if (sApplyMethod != null) {

            try {

                sApplyMethod.invoke(editor,
new Object[0]);

                return;

            } catch
(InvocationTargetException e) {

            } catch (IllegalAccessException
e2) {

            }

        }

        editor.commit();

    }

}
```

ZIPUTIL.JAVA

```
package android.support.multidex;

import java.io.File;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.util.zip.CRC32;

import java.util.zip.ZipException;

final class ZipUtil {

   static class CentralDirectory {

      long offset;

      long size;



      CentralDirectory() {

      }

   }

   static long getZipCrc(File apk) throws IOException {

      RandomAccessFile raf = new RandomAccessFile(apk, "r");

      try {

         long computeCrcOfCentralDir = computeCrcOfCentralDir(raf, findCentralDirectory(raf));

         return computeCrcOfCentralDir;

      } finally {

         raf.close();

      }

   }

   static CentralDirectory findCentralDirectory(RandomAccessFile raf) throws IOException, ZipException {

      long scanOffset = raf.length() - 22;

      if (scanOffset < 0) {

         throw new ZipException("File too short to be a zip file: " + raf.length());

      }

      long stopOffset = scanOffset - 65536;

      if (stopOffset < 0) {

         stopOffset = 0;

      }

      int endSig = Integer.reverseBytes(101010256);

      do {

         raf.seek(scanOffset);

         if (raf.readInt() == endSig) {

            raf.skipBytes(2);

            raf.skipBytes(2);

            raf.skipBytes(2);

            raf.skipBytes(2);

            CentralDirectory dir = new CentralDirectory();

            dir.size = ((long) Integer.reverseBytes(raf.readInt())) & 4294967295L;

            dir.offset = ((long) Integer.reverseBytes(raf.readInt())) & 4294967295L;

            return dir;

         }

         scanOffset--;

      } while (scanOffset >= stopOffset);

      throw new ZipException("End Of Central Directory signature not found");

   }

   static long computeCrcOfCentralDir(RandomAccessFile raf, CentralDirectory dir) throws IOException {

      CRC32 crc = new CRC32();

      long stillToRead = dir.size;

      raf.seek(dir.offset);

      byte[] buffer = new byte[16384];

      int length = raf.read(buffer, 0, (int) Math.min(16384, stillToRead));

      while (length != -1) {

         crc.update(buffer, 0, length);

         stillToRead -= (long) length;

         if (stillToRead == 0) {

            break;

         }

         length = raf.read(buffer, 0, (int) Math.min(16384, stillToRead));

      }

      return crc.getValue();

   }

}
```

C0063R.JAVA

```
package android.support.v7.cardview;

import com.facebook.C0415R;


/* renamed from: android.support.v7.cardview.R */

public final class C0063R {


    /* renamed from: android.support.v7.cardview.R.color */

    public static final class color {

        public static final int cardview_dark_background = 2131361792;

        public static final int cardview_light_background = 2131361793;

        public static final int cardview_shadow_end_color = 2131361794;

        public static final int cardview_shadow_start_color = 2131361795;

    }


    /* renamed from: android.support.v7.cardview.R.dimen */

    public static final class dimen {

        public static final int cardview_compat_inset_shadow = 2131230784;

        public static final int cardview_default_elevation = 2131230785;

        public static final int cardview_default_radius = 2131230786;

    }


    /* renamed from: android.support.v7.cardview.R.style */

    public static final class style {

        public static final int CardView = 2131165186;

        public static final int CardView_Dark = 2131165187;

        public static final int CardView_Light = 2131165188;

    }


    /* renamed from: android.support.v7.cardview.R.styleable */

    public static final class styleable {

        public static final int[] CardView;

        public static final int CardView_cardBackgroundColor = 0;

        public static final int CardView_cardCornerRadius = 1;

        public static final int CardView_cardElevation = 2;

        public static final int CardView_cardMaxElevation = 3;

        public static final int CardView_cardPreventCornerOverlap = 5;

        public static final int CardView_cardUseCompatPadding = 4;

        public static final int CardView_contentPadding = 6;

        public static final int CardView_contentPaddingBottom = 10;

        public static final int CardView_contentPaddingLeft = 7;

        public static final int CardView_contentPaddingRight = 8;

        public static final int CardView_contentPaddingTop = 9;


        static {

            CardView = new int[]{C0415R.attr.cardBackgroundColor, C0415R.attr.cardCornerRadius, C0415R.attr.cardElevation, C0415R.attr.cardMaxElevation, C0415R.attr.cardUseCompatPadding, C0415R.attr.cardPreventCornerOverlap, C0415R.attr.contentPadding, C0415R.attr.contentPaddingLeft, C0415R.attr.contentPaddingRight, C0415R.attr.contentPaddingTop, C0415R.attr.contentPaddingBottom};

        }

    }

}
```

C0064R.JAVA

```
package android.support.v7.gridlayout;


/* renamed from: android.support.v7.gridlayout.R */

public final class C0064R {


    /* renamed from: android.support.v7.gridlayout.R.dimen */

    public static final class dimen {

        public static final int default_gap = 2131230859;

    }
```

```
    /* renamed from:
android.support.v7.gridlayout.R.styleable
*/

    public static final class styleable {

        public static final int[] GridLayout;

        public static final int[]
GridLayout_Layout;

        public static final int
GridLayout_Layout_android_layout_hei
ght = 1;

        public static final int
GridLayout_Layout_android_layout_mar
gin = 2;

        public static final int
GridLayout_Layout_android_layout_mar
ginBottom = 6;

        public static final int
GridLayout_Layout_android_layout_mar
ginLeft = 3;

        public static final int
GridLayout_Layout_android_layout_mar
ginRight = 5;

        public static final int
GridLayout_Layout_android_layout_mar
ginTop = 4;

        public static final int
GridLayout_Layout_android_layout_wid
th = 0;

        public static final int
GridLayout_Layout_layout_column =
10;

        public static final int
GridLayout_Layout_layout_columnSpan
= 11;

        public static final int
GridLayout_Layout_layout_columnWei
ght = 12;

        public static final int
GridLayout_Layout_layout_gravity =
13;

        public static final int
GridLayout_Layout_layout_row = 7;

        public static final int
GridLayout_Layout_layout_rowSpan =
8;

        public static final int
GridLayout_Layout_layout_rowWeight
= 9;

        public static final int
GridLayout_alignmentMode = 4;

        public static final int
GridLayout_columnCount = 2;

        public static final int
GridLayout_columnOrderPreserved = 6;

        public static final int
GridLayout_orientation = 0;

        public static final int
GridLayout_rowCount = 1;

        public static final int
GridLayout_rowOrderPreserved = 5;

        public static final int
GridLayout_useDefaultMargins = 3;



        static {

            GridLayout = new
int[]{2130772001, 2130772002,
2130772003, 2130772004, 2130772005,
2130772006, 2130772007};

            GridLayout_Layout = new
int[]{16842996, 16842997, 16842998,
16842999, 16843000, 16843001,
16843002, 2130772008, 2130772009,
2130772010, 2130772011, 2130772012,
2130772013, 2130772014};

        }

    }

}
```

C0065R.JAVA

```
package
android.support.v7.recyclerview;



import com.facebook.ads.C0448R;



/* renamed from:
android.support.v7.recyclerview.R */

public final class C0065R {



    /* renamed from:
android.support.v7.recyclerview.R.stylea
ble */

    public static final class styleable {

        public static final int[]
RecyclerView;

        public static final int
RecyclerView_android_orientation = 0;

        public static final int
RecyclerView_layoutManager = 1;

        public static final int
RecyclerView_reverseLayout = 3;

        public static final int
RecyclerView_spanCount = 2;

        public static final int
RecyclerView_stackFromEnd = 4;

        static {

            RecyclerView = new
int[]{16842948,
C0448R.attr.layoutManager,
C0448R.attr.spanCount,
C0448R.attr.reverseLayout,
C0448R.attr.stackFromEnd};

        }

    }

}
```

ADAPTERHELPER.JAVA

```
package android.support.v7.widget;



import
android.support.v4.util.Pools.Pool;

import
android.support.v4.util.Pools.SimplePool
;

import
android.support.v7.widget.RecyclerView
.ViewHolder;

import
com.swelen.ads.SwelenAdConversion;

import com.swelen.ads.SwelenAdView;

import java.util.ArrayList;

import java.util.List;



class AdapterHelper implements
Callback {

    final Callback mCallback;

    final boolean mDisableRecycler;

    Runnable
mOnItemProcessedCallback;

    final OpReorderer mOpReorderer;

    final ArrayList<UpdateOp>
mPendingUpdates;

    final ArrayList<UpdateOp>
mPostponedList;

    private Pool<UpdateOp>
mUpdateOpPool;



    interface Callback {

        ViewHolder findViewHolder(int i);



        void markViewHoldersUpdated(int
i, int i2, Object obj);

        void offsetPositionsForAdd(int i, int
i2);



        void offsetPositionsForMove(int i,
int i2);



        void
offsetPositionsForRemovingInvisible(int
i, int i2);



        void
offsetPositionsForRemovingLaidOutOrN
ewView(int i, int i2);



        void
onDispatchFirstPass(UpdateOp
updateOp);



        void
onDispatchSecondPass(UpdateOp
updateOp);

    }



    static class UpdateOp {

        int cmd;

        int itemCount;

        Object payload;

        int positionStart;



        UpdateOp(int cmd, int positionStart,
int itemCount, Object payload) {

            this.cmd = cmd;

            this.positionStart = positionStart;

            this.itemCount = itemCount;

            this.payload = payload;

        }

        String cmdToString() {

            switch (this.cmd) {

                case
SwelenAdConversion.UNIQUE /*0*/:

                    return "add";

                case
SwelenAdView.ERR_CANT_CONNEC
T /*1*/:

                    return "rm";

                case
SwelenAdView.ERR_BAD_DATA
/*2*/:

                    return "up";

                case
SwelenAdView.ERR_NO_AD_FOUND
/*3*/:

                    return "mv";

                default:

                    return "??";

            }

        }



        public String toString() {

            return
Integer.toHexString(System.identityHash
Code(this)) + "[" + cmdToString() + ",s:"
+ this.positionStart + "c:" +
this.itemCount + ",p:" + this.payload +
"]";

        }



        public boolean equals(Object o) {

            if (this == o) {

                return true;

            }
```

```
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        UpdateOp op = (UpdateOp) o;
        if (this.cmd != op.cmd) {
            return false;
        }
        if (this.cmd == 3 && Math.abs(this.itemCount - this.positionStart) == 1 && this.itemCount == op.positionStart && this.positionStart == op.itemCount) {
            return true;
        }
        if (this.itemCount != op.itemCount) {
            return false;
        }
        if (this.positionStart != op.positionStart) {
            return false;
        }
        if (this.payload != null) {
            if (this.payload.equals(op.payload)) {
                return true;
            }
            return false;
        } else if (op.payload != null) {
            return false;
        } else {
            return true;
        }
    }

    public int hashCode() {
        return (((this.cmd * 31) + this.positionStart) * 31) + this.itemCount;
    }
}


AdapterHelper(Callback callback) {
    this(callback, false);
}


AdapterHelper(Callback callback, boolean disableRecycler) {
    this.mUpdateOpPool = new SimplePool(30);
    this.mPendingUpdates = new ArrayList();
    this.mPostponedList = new ArrayList();
    this.mCallback = callback;
    this.mDisableRecycler = disableRecycler;
    this.mOpReorderer = new OpReorderer(this);
}


void reset() {
    recycleUpdateOpsAndClearList(this.mPendingUpdates);
    recycleUpdateOpsAndClearList(this.mPostponedList);
}


void preProcess() {
    this.mOpReorderer.reorderOps(this.mPendingUpdates);
    int count = this.mPendingUpdates.size();
    for (int i = 0; i < count; i++) {
        UpdateOp op = (UpdateOp) this.mPendingUpdates.get(i);
        switch (op.cmd) {
            case SwelenAdConversion.UNIQUE /*0*/:
                applyAdd(op);
                break;
            case SwelenAdView.ERR_CANT_CONNECT /*1*/:
                applyRemove(op);
                break;
            case SwelenAdView.ERR_BAD_DATA /*2*/:
                applyUpdate(op);
                break;
            case SwelenAdView.ERR_NO_AD_FOUND /*3*/:
                applyMove(op);
                break;
        }
        if (this.mOnItemProcessedCallback != null) {
            this.mOnItemProcessedCallback.run();
        }
    }
    this.mPendingUpdates.clear();
}
```

```
    void consumePostponedUpdates() {
        int count =
this.mPostponedList.size();
        for (int i = 0; i < count; i++) {

this.mCallback.onDispatchSecondPass((
UpdateOp) this.mPostponedList.get(i));
        }

recycleUpdateOpsAndClearList(this.mP
ostponedList);
    }


    private void applyMove(UpdateOp op)
{

postponeAndUpdateViewHolders(op);
    }


    private void applyRemove(UpdateOp
op) {
        int tmpStart = op.positionStart;
        int tmpCount = 0;
        int tmpEnd = op.positionStart +
op.itemCount;
        int type = -1;
        int position = op.positionStart;
        while (position < tmpEnd) {
            boolean typeChanged = false;
            if
(this.mCallback.findViewHolder(positio
n) != null ||
canFindInPreLayout(position)) {
                if (type == 0) {

dispatchAndUpdateViewHolders(obtain
UpdateOp(1, tmpStart, tmpCount, null));
                    typeChanged = true;
                }
                type = 1;
            } else {
                if (type == 1) {

postponeAndUpdateViewHolders(obtain
UpdateOp(1, tmpStart, tmpCount, null));
                    typeChanged = true;
                }
                type = 0;
            }
            if (typeChanged) {
                position -= tmpCount;
                tmpEnd -= tmpCount;
                tmpCount = 1;
            } else {
                tmpCount++;
            }
            position++;
        }
        if (tmpCount != op.itemCount) {
            recycleUpdateOp(op);
            op = obtainUpdateOp(1,
tmpStart, tmpCount, null);
        }
        if (type == 0) {

dispatchAndUpdateViewHolders(op);
        } else {

postponeAndUpdateViewHolders(op);
        }
    }


    private void applyUpdate(UpdateOp
op) {
        int tmpStart = op.positionStart;
        int tmpCount = 0;
        int tmpEnd = op.positionStart +
op.itemCount;
        int type = -1;
        int position = op.positionStart;
        while (position < tmpEnd) {
            if
(this.mCallback.findViewHolder(positio
n) != null ||
canFindInPreLayout(position)) {
                if (type == 0) {

dispatchAndUpdateViewHolders(obtain
UpdateOp(2, tmpStart, tmpCount,
op.payload));
                    tmpCount = 0;
                    tmpStart = position;
                }
                type = 1;
            } else {
                if (type == 1) {

postponeAndUpdateViewHolders(obtain
UpdateOp(2, tmpStart, tmpCount,
op.payload));
                    tmpCount = 0;
                    tmpStart = position;
                }
                type = 0;
            }
            tmpCount++;
            position++;
        }
```

```
        if (tmpCount != op.itemCount) {

            Object payload = op.payload;

            recycleUpdateOp(op);

            op = obtainUpdateOp(2,
tmpStart, tmpCount, payload);

        }

        if (type == 0) {


dispatchAndUpdateViewHolders(op);

        } else {


postponeAndUpdateViewHolders(op);

        }

    }



    private void
dispatchAndUpdateViewHolders(Update
Op op) {

        if (op.cmd == 0 || op.cmd == 3) {

            throw new
IllegalArgumentException("should not
dispatch add or move for pre layout");

        }

        int positionMultiplier;

        int tmpStart =
updatePositionWithPostponed(op.positio
nStart, op.cmd);

        int tmpCnt = 1;

        int offsetPositionForPartial =
op.positionStart;

        switch (op.cmd) {

            case
SwelenAdView.ERR_CANT_CONNEC
T /*1*/:

                positionMultiplier = 0;

                break;

            case
SwelenAdView.ERR_BAD_DATA
/*2*/:

                positionMultiplier = 1;

                break;

            default:

                throw new
IllegalArgumentException("op should be
remove or update." + op);

        }

        for (int p = 1; p < op.itemCount;
p++) {

            int updatedPos =
updatePositionWithPostponed(op.positio
nStart + (positionMultiplier * p),
op.cmd);

            boolean continuous = false;

            switch (op.cmd) {

                case
SwelenAdView.ERR_CANT_CONNEC
T /*1*/:

                    continuous = updatedPos ==
tmpStart;

                    break;

                case
SwelenAdView.ERR_BAD_DATA
/*2*/:

                    if (updatedPos == tmpStart
+ 1) {

                        continuous = true;

                    } else {

                        continuous = false;

                    }

                    break;

            }

            if (continuous) {

                tmpCnt++;

            } else {

                UpdateOp tmp =
obtainUpdateOp(op.cmd, tmpStart,
tmpCnt, op.payload);


dispatchFirstPassAndUpdateViewHolder
s(tmp, offsetPositionForPartial);

                recycleUpdateOp(tmp);

                if (op.cmd == 2) {

                    offsetPositionForPartial +=
tmpCnt;

                }

                tmpStart = updatedPos;

                tmpCnt = 1;

            }

        }

        Object payload = op.payload;

        recycleUpdateOp(op);

        if (tmpCnt > 0) {

            tmp = obtainUpdateOp(op.cmd,
tmpStart, tmpCnt, payload);


dispatchFirstPassAndUpdateViewHolder
s(tmp, offsetPositionForPartial);

            recycleUpdateOp(tmp);

        }

    }



    void
dispatchFirstPassAndUpdateViewHolder
s(UpdateOp op, int offsetStart) {


this.mCallback.onDispatchFirstPass(op);

        switch (op.cmd) {

            case
SwelenAdView.ERR_CANT_CONNEC
T /*1*/:


this.mCallback.offsetPositionsForRemov
ingInvisible(offsetStart, op.itemCount);
```

```
        case
SwelenAdView.ERR_BAD_DATA
/*2*/:


this.mCallback.markViewHoldersUpdate
d(offsetStart, op.itemCount, op.payload);

        default:

            throw new
IllegalArgumentException("only remove
and update ops can be dispatched in first
pass");

      }

   }



   private int
updatePositionWithPostponed(int pos,
int cmd) {

      int i;

      for (i = this.mPostponedList.size() -
1; i >= 0; i--) {

         UpdateOp postponed =
(UpdateOp) this.mPostponedList.get(i);

         if (postponed.cmd == 3) {

            int start;

            int end;

            if (postponed.positionStart <
postponed.itemCount) {

               start =
postponed.positionStart;

               end = postponed.itemCount;

            } else {

               start =
postponed.itemCount;

               end =
postponed.positionStart;

            }

            if (pos < start || pos > end) {

               if (pos <
postponed.positionStart) {

                  if (cmd == 0) {


postponed.positionStart++;


postponed.itemCount++;

                  } else if (cmd == 1) {


postponed.positionStart--;

                     postponed.itemCount--;

                  }

               }

            } else if (start ==
postponed.positionStart) {

               if (cmd == 0) {

                  postponed.itemCount++;

               } else if (cmd == 1) {

                  postponed.itemCount--;

               }

               pos++;

            } else {

               if (cmd == 0) {


postponed.positionStart++;

               } else if (cmd == 1) {

                  postponed.positionStart--;

               }

               pos--;

            }

         } else if (postponed.positionStart
<= pos) {

            if (postponed.cmd == 0) {

               pos -=
postponed.itemCount;

            } else if (postponed.cmd == 1)
{

               pos +=
postponed.itemCount;

            }

         } else if (cmd == 0) {

            postponed.positionStart++;

         } else if (cmd == 1) {

            postponed.positionStart--;

         }

      }

      for (i = this.mPostponedList.size() -
1; i >= 0; i--) {

         UpdateOp op = (UpdateOp)
this.mPostponedList.get(i);

         if (op.cmd == 3) {

            if (op.itemCount ==
op.positionStart || op.itemCount < 0) {


this.mPostponedList.remove(i);

               recycleUpdateOp(op);

            }

         } else if (op.itemCount <= 0) {


this.mPostponedList.remove(i);

            recycleUpdateOp(op);

         }

      }

      return pos;

   }



   private boolean
canFindInPreLayout(int position) {

      int count =
this.mPostponedList.size();

      for (int i = 0; i < count; i++) {

         UpdateOp op = (UpdateOp)
this.mPostponedList.get(i);
```

```
        if (op.cmd == 3) {
            if (findPositionOffset(op.itemCount, i + 1) == position) {
                return true;
            }
        } else if (op.cmd == 0) {
            int end = op.positionStart + op.itemCount;
            for (int pos = op.positionStart; pos < end; pos++) {
                if (findPositionOffset(pos, i + 1) == position) {
                    return true;
                }
            }
            continue;
        } else {
            continue;
        }
    }
    return false;
  }

  private void applyAdd(UpdateOp op) {

postponeAndUpdateViewHolders(op);

  }

  private void postponeAndUpdateViewHolders(UpdateOp op) {
    this.mPostponedList.add(op);
    switch (op.cmd) {
      case SwelenAdConversion.UNIQUE /*0*/:
this.mCallback.offsetPositionsForAdd(op.positionStart, op.itemCount);
        case SwelenAdView.ERR_CANT_CONNECT /*1*/:

this.mCallback.offsetPositionsForRemovingLaidOutOrNewView(op.positionStart, op.itemCount);
        case SwelenAdView.ERR_BAD_DATA /*2*/:

this.mCallback.markViewHoldersUpdated(op.positionStart, op.itemCount, op.payload);
        case SwelenAdView.ERR_NO_AD_FOUND /*3*/:

this.mCallback.offsetPositionsForMove(op.positionStart, op.itemCount);
        default:
          throw new IllegalArgumentException("Unknown update op type for " + op);
      }
  }

  boolean hasPendingUpdates() {
    return this.mPendingUpdates.size() > 0;
  }

  int findPositionOffset(int position) {
    return findPositionOffset(position, 0);
  }

  int findPositionOffset(int position, int firstPostponedItem) {
    int count = this.mPostponedList.size();
    for (int i = firstPostponedItem; i < count; i++) {
      UpdateOp op = (UpdateOp) this.mPostponedList.get(i);
      if (op.cmd == 3) {
        if (op.positionStart == position) {
          position = op.itemCount;
        } else {
          if (op.positionStart < position) {
            position--;
          }
          if (op.itemCount <= position) {
            position++;
          }
        }
      } else if (op.positionStart > position) {
        continue;
      } else if (op.cmd == 1) {
        if (position < op.positionStart + op.itemCount) {
          return -1;
        }
        position -= op.itemCount;
      } else if (op.cmd == 0) {
        position += op.itemCount;
      }
    }
    return position;
  }
```

```
    boolean onItemRangeChanged(int positionStart, int itemCount, Object payload) {

this.mPendingUpdates.add(obtainUpdateOp(2, positionStart, itemCount, payload));

        if (this.mPendingUpdates.size() == 1) {

            return true;

        }

        return false;

    }



    boolean onItemRangeInserted(int positionStart, int itemCount) {

this.mPendingUpdates.add(obtainUpdateOp(0, positionStart, itemCount, null));

        if (this.mPendingUpdates.size() == 1) {

            return true;

        }

        return false;

    }



    boolean onItemRangeRemoved(int positionStart, int itemCount) {

this.mPendingUpdates.add(obtainUpdateOp(1, positionStart, itemCount, null));

        if (this.mPendingUpdates.size() == 1) {

            return true;

        }

        return false;

    }



    boolean onItemRangeMoved(int from, int to, int itemCount) {

        boolean z = true;

        if (from == to) {

            return false;

        }

        if (itemCount != 1) {

            throw new IllegalArgumentException("Moving more than 1 item is not supported yet");

        }

this.mPendingUpdates.add(obtainUpdateOp(3, from, to, null));

        if (this.mPendingUpdates.size() != 1) {

            z = false;

        }

        return z;

    }



    void consumeUpdatesInOnePass() {

        consumePostponedUpdates();

        int count = this.mPendingUpdates.size();

        for (int i = 0; i < count; i++) {

            UpdateOp op = (UpdateOp) this.mPendingUpdates.get(i);

            switch (op.cmd) {

                case SwelenAdConversion.UNIQUE /*0*/:

this.mCallback.onDispatchSecondPass(op);

this.mCallback.offsetPositionsForAdd(op.positionStart, op.itemCount);

                    break;

                case SwelenAdView.ERR_CANT_CONNECT /*1*/:

this.mCallback.onDispatchSecondPass(op);

this.mCallback.offsetPositionsForRemovingInvisible(op.positionStart, op.itemCount);

                    break;

                case SwelenAdView.ERR_BAD_DATA /*2*/:

this.mCallback.onDispatchSecondPass(op);

this.mCallback.markViewHoldersUpdated(op.positionStart, op.itemCount, op.payload);

                    break;

                case SwelenAdView.ERR_NO_AD_FOUND /*3*/:

this.mCallback.onDispatchSecondPass(op);

this.mCallback.offsetPositionsForMove(op.positionStart, op.itemCount);

                    break;

            }

            if (this.mOnItemProcessedCallback != null) {

this.mOnItemProcessedCallback.run();

            }

        }

recycleUpdateOpsAndClearList(this.mPendingUpdates);

    }
```

```
    public int
applyPendingUpdatesToPosition(int
position) {

        int size =
this.mPendingUpdates.size();

        for (int i = 0; i < size; i++) {

            UpdateOp op = (UpdateOp)
this.mPendingUpdates.get(i);

            switch (op.cmd) {

                case
SwelenAdConversion.UNIQUE /*0*/:

                    if (op.positionStart >
position) {

                        break;

                    }

                    position += op.itemCount;

                    break;

                case
SwelenAdView.ERR_CANT_CONNEC
T /*1*/:

                    if (op.positionStart <=
position) {

                        if (op.positionStart +
op.itemCount <= position) {

                            position -=
op.itemCount;

                            break;

                        }

                        return -1;

                    }

                    continue;

                case
SwelenAdView.ERR_NO_AD_FOUND
/*3*/:

                    if (op.positionStart !=
position) {

                        if (op.positionStart <
position) {

                            position--;

                        }

                        if (op.itemCount >
position) {

                            break;

                        }

                        position++;

                        break;

                    }

                    position = op.itemCount;

                    break;

                default:

                    break;

            }

        }

        return position;

    }


    public UpdateOp obtainUpdateOp(int
cmd, int positionStart, int itemCount,
Object payload) {

        UpdateOp op = (UpdateOp)
this.mUpdateOpPool.acquire();

        if (op == null) {

            return new UpdateOp(cmd,
positionStart, itemCount, payload);

        }

        op.cmd = cmd;

        op.positionStart = positionStart;

        op.itemCount = itemCount;

        op.payload = payload;

        return op;

    }


    public void
recycleUpdateOp(UpdateOp op) {

        if (!this.mDisableRecycler) {

            op.payload = null;

            this.mUpdateOpPool.release(op);

        }

    }


    void
recycleUpdateOpsAndClearList(List<Up
dateOp> ops) {

        int count = ops.size();

        for (int i = 0; i < count; i++) {

            recycleUpdateOp((UpdateOp)
ops.get(i));

        }

        ops.clear();

    }

}
```

CUSTOMEVENT.JAVA

```
package com.adsdk.sdk.customevents;

public class CustomEvent {

    private String className;

    private String optionalParameter;

    private String pixelUrl;
```

```
    public CustomEvent(String
className, String optionalParam, String
pixelUrl) {

        this.className = className;

        this.optionalParameter =
optionalParam;

        this.pixelUrl = pixelUrl;
    }

    public String getClassName() {

        return this.className;

    }

    public String getOptionalParameter() {

        return this.optionalParameter;

    }

    public String getPixelUrl() {

        return this.pixelUrl;

    }
}
```

CUSTOMEVENTBANNER.JAVA

```
package com.adsdk.sdk.customevents;

import android.content.Context;

public abstract class CustomEventBanner
{

    public interface
CustomEventBannerListener {

        void onBannerFailed();

    }

    public abstract void
loadBanner(Context context,
CustomEventBannerListener
customEventBannerListener, String str,
String str2, int i, int i2);

}
```

CUSTOMEVENTBANNERFACTORY.JAVA

```
package com.adsdk.sdk.customevents;

import java.lang.reflect.Constructor;

public class CustomEventBannerFactory
{

    private static
CustomEventBannerFactory instance;

    static {

        instance = new
CustomEventBannerFactory();
    }

    public static CustomEventBanner
create(String className) throws
Exception {

        return
instance.internalCreate(className);

    }

    protected CustomEventBanner
internalCreate(String className) throws
Exception {

        Constructor<?> bannerConstructor
=
Class.forName("com.adsdk.sdk.custome
vents." + className +
"Banner").asSubclass(CustomEventBann
er.class).getDeclaredConstructor(null);

bannerConstructor.setAccessible(true);

        return (CustomEventBanner)
bannerConstructor.newInstance(new
Object[0]);

    }
}
```

CUSTOMEVENTFULLSCREEN.JAVA

```
package com.adsdk.sdk.customevents;

import android.content.Context;

public abstract class
CustomEventFullscreen {

    public interface
CustomEventFullscreenListener {

    }

    public abstract void
loadFullscreen(Context context,
CustomEventFullscreenListener
customEventFullscreenListener, String
str, String str2);
```

```
    public abstract void showFullscreen();
}
```

CUSTOMEVENTFULLSCREENFACTORY.JAVA

```
package com.adsdk.sdk.customevents;

import java.lang.reflect.Constructor;

public class CustomEventFullscreenFactory {

    private static CustomEventFullscreenFactory instance;

    static {
        instance = new CustomEventFullscreenFactory();
    }

    public static CustomEventFullscreen create(String className) throws Exception {
        return instance.internalCreate(className);
    }

    protected CustomEventFullscreen internalCreate(String className) throws Exception {
        Constructor<?> fullscreenConstructor = Class.forName("com.adsdk.sdk.customevents." + className + "Fullscreen").asSubclass(CustomEventFullscreen.class).getDeclaredConstructor(null);

        fullscreenConstructor.setAccessible(true);

        return (CustomEventFullscreen) fullscreenConstructor.newInstance(new Object[0]);
    }

}
```

CLICKTYPE.JAVA

```
package com.adsdk.sdk.data;

public enum ClickType {

    INAPP,

    BROWSER;

    public static ClickType getValue(String value) {

        for (ClickType clickType : values()) {

            if (clickType.name().equalsIgnoreCase(value)) {

                return clickType;
            }

        }

        return null;

    }

}
```

NATIVEAD.JAVA

```
package com.adsdk.sdk.nativeads;

public class NativeAd {

    public static class Tracker {

        String type;

        String url;

    }

}
```

NATIVEADLISTENER.JAVA

```
package com.adsdk.sdk.nativeads;

public interface NativeAdListener {

    void impression();

}
```

NATIVEADVIEW.JAVA

```
package com.adsdk.sdk.nativeads;

import android.annotation.SuppressLint;

import android.graphics.Canvas;

import android.os.AsyncTask;

import android.os.Handler;

import android.widget.FrameLayout;

import com.adsdk.sdk.nativeads.NativeAd.Tracker;
```

```
import java.io.IOException;

import java.net.URI;

import java.net.URISyntaxException;

import java.util.List;

import
org.apache.http.client.ClientProtocolExc
eption;

import org.apache.http.client.HttpClient;

import
org.apache.http.client.methods.HttpGet;

import
org.apache.http.impl.client.DefaultHttpC
lient;



@SuppressLint({"ViewConstructor"})

public class NativeAdView extends
FrameLayout {

   private Handler handler;

   private boolean impressionReported;

   private NativeAdListener listener;

   private List<Tracker> trackers;



   /* renamed from:
com.adsdk.sdk.nativeads.NativeAdView.
1 */

   class C01471 implements Runnable {

      C01471() {

      }



      public void run() {


NativeAdView.this.listener.impression();

      }

   }

   /* renamed from:
com.adsdk.sdk.nativeads.NativeAdView.
2 */

   class C01482 extends
AsyncTask<Void, Void, Void> {

      private final /* synthetic */ String
val$url;



      C01482(String str) {

         this.val$url = str;

      }



      protected Void
doInBackground(Void... params) {

         try {

            HttpClient client = new
DefaultHttpClient();

            HttpGet request = new
HttpGet();

            request.setHeader("User-
Agent",
System.getProperty("http.agent"));

            request.setURI(new
URI(this.val$url));

            client.execute(request);

         } catch (URISyntaxException e)
{

            e.printStackTrace();

         } catch (ClientProtocolException
e2) {

            e2.printStackTrace();

         } catch (IOException e3) {

            e3.printStackTrace();

         }

         return null;

      }

   }



   protected void dispatchDraw(Canvas
canvas) {

      if (!this.impressionReported) {

         this.impressionReported = true;

         notifyImpression();

         for (Tracker t : this.trackers) {

            if (t.type.equals("impression"))
{

               trackImpression(t.url);

            }

         }

      }

      super.dispatchDraw(canvas);

   }



   private void notifyImpression() {

      if (this.listener != null) {

         this.handler.post(new C01471());

      }

   }



   private void trackImpression(String
url) {

      new C01482(url).execute(new
Void[0]);

   }

}
```

SASMOTIONLEADADAPTER.JAVA

```
package com.smartadserver.android.library.mediation;

import java.util.HashMap;

import android.app.Activity;
import android.content.Context;
import android.util.Log;
import android.view.View;

import com.google.android.gms.ads.identifier.AdvertisingIdClient;
import com.motionlead.listeners.MotionleadHelper;
import com.motionlead.motionlead.MotionleadHandler;
import com.smartadserver.android.library.SASBannerView;
import com.smartadserver.android.library.SASInterstitialView;
import com.smartadserver.android.library.controller.mraid.SASMRAIDState;
import com.smartadserver.android.library.ui.SASAdView;
import com.smartadserver.android.library.util.SASUtil;

/**
 * Adapter class for MotionLead SDK
 * @hide
 */
public class SASMotionleadAdapter implements SASMediationSDKAdapter {

    private static final String TAG = "SASMotionleadAdapter";

    private static final String PLACEMENT_ID_KEY="placementId";
    private static final String PUBLISHER_ID="publisherId";

    MotionleadHandler motionleadHandler;
    private AdRequestHandler adRequestHandler;
    private SASAdView sasAdView = null;

    /*
     * Implementation of MotionLead helper that forward events to the SAS SDK via the AdRequestHandler
     */
    private class MotionleadHelperImpl implements MotionleadHelper {

        @Override
        public void onError(String arg0) {

            Log.d(TAG, "MotionLead Ad onError");
            adRequestHandler.adRequestFailed(arg0);

        }

        @Override
        public void onMLClick() {

            Log.d(TAG, "MotionLead Ad onMLClick()");
            adRequestHandler.adWasClicked();

        }

        @Override
        public void onMLDismiss() {
            if (sasAdView != null) {
                sasAdView.executeOnUIThread(new Runnable() {
                    public void run() {
                        sasAdView.close();
                    }
                });
            }
        }

        @Override
        public void onMotionleadLoaded() {
```

```
            Log.d(TAG, "MotionLead Ad onMotionleadLoaded");


        if (adRequestHandler != null) {


            sasAdView.getMRAIDController().setState(SASMRAIDState.DEFAULT);


            boolean proceed = adRequestHandler.adRequestSucceeded();


            if (sasAdView instanceof SASInterstitialView) {

                if (proceed) {

                    if (sasAdView != null) {

                        // need to hide close button


                        sasAdView.getMRAIDController().setExpandUseCustomCloseProperty(true);

                    }


                    motionleadHandler.displayAd();

                }

            }

        }

                    }


                    @Override

                    public void onShow() {


            Log.d(TAG, "MotionLead Ad onShow");

                    }

    }


                @Override

                public void requestAd(final SASAdView sasAdView, HashMap<String, String> placementParameters, AdRequestHandler adRequestHandler) {


                    // abort immediately if the SASAdView is a banner (support for interstitial only)

                    if (sasAdView instanceof SASBannerView) {


                adRequestHandler.adRequestFailed("MotionLead mediation is not supported for banner placement, only interstitial");

                        return;

                    }


                    Context viewContext = sasAdView.getContext();

                    if (!(viewContext instanceof Activity)) {


                adRequestHandler.adRequestFailed("MotionLead ad mediation requires that the SASInterstitialView be created with an Activity as context parameter");

                        return;

                    }


                    Activity viewActivity = (Activity)sasAdView.getContext();


                    String placementID = placementParameters.get(PLACEMENT_ID_KEY);

                    String publisherId = placementParameters.get(PUBLISHER_ID);


                    // store adRequestHandler and SASADView


                this.adRequestHandler = adRequestHandler;

                    this.sasAdView = sasAdView;


                    // clean previous ad views


                cleanPreviousMotionLeadSetup();


                    // init ad listener if need be

                    if (motionleadHandler == null) {


                motionleadHandler = new MotionleadHandler(viewActivity,new MotionleadHelperImpl());

                    }


                    motionleadHandler.displayAd();


                    String advertisingID = "";


                AdvertisingIdClient.Info advertisingIDInfo = SASUtil.getAdvertisingIdClientInfo(null, false);
```

```
                if (advertisingIDInfo != null) {

                    advertisingID = advertisingIDInfo.getId();

                }

                motionleadHandler.onRemoteAd(publisherId, placementID, advertisingID);

            }



            private void cleanPreviousMotionLeadSetup() {

//                motionleadHandler = null; // ?

            }

            @Override

            public View getAdView() {

                // no View as MotionLead displays ads over the Activity

                return null;

            }



            @Override

            public void destroy() {

                sasAdView = null;

                cleanPreviousMotionLeadSetup();

            }



            @Override

            public boolean isSDKAvailable() {

                boolean available = true;

                try {

                    Class.forName("com.motionlead.motionlead.MotionleadHandler");

                } catch (ClassNotFoundException e) {

                    available = false;

                }

                return available;

            }

        }
```

BUILDER.JAVA

```
package org.simpleframework.xml.stream;


import org.simpleframework.xml.util.Cache;

import org.simpleframework.xml.util.ConcurrentCache;


class Builder implements Style {

   private final Cache<String> attributes;

   private final Cache<String> elements;

   private final Style style;


   public Builder(Style style) {

      this.attributes = new ConcurrentCache();

      this.elements = new ConcurrentCache();

      this.style = style;

   }


   public String getAttribute(String name) {

      String value = (String) this.attributes.fetch(name);

      if (value != null) {

         return value;

      }

      value = this.style.getAttribute(name);

      if (value != null) {

         this.attributes.cache(name, value);

      }

      return value;

   }


   public String getElement(String name) {

      String value = (String) this.elements.fetch(name);

      if (value != null) {

         return value;

      }

      value = this.style.getElement(name);

      if (value != null) {

         this.elements.cache(name, value);

      }

      return value;

   }
```

```
    public void setAttribute(String name,
String value) {
        this.attributes.cache(name, value);
    }

    public void setElement(String name,
String value) {
        this.elements.cache(name, value);
    }
}
```

SERVICEBUILDER.JAVA

```
package org.scribe.builder;

import java.io.OutputStream;

import org.scribe.builder.api.Api;

import
org.scribe.exceptions.OAuthException;

import org.scribe.model.OAuthConfig;

import org.scribe.model.SignatureType;

import org.scribe.oauth.OAuthService;

import org.scribe.utils.Preconditions;


public class ServiceBuilder {

    private Api api;

    private String apiKey;

    private String apiSecret;

    private String callback;

    private OutputStream debugStream;

    private String scope;

    private SignatureType signatureType;


    public ServiceBuilder() {

        this.callback = "oob";

        this.signatureType =
SignatureType.Header;

        this.debugStream = null;

    }


    public ServiceBuilder
provider(Class<? extends Api> apiClass)
{

        this.api = createApi(apiClass);

        return this;

    }


    private Api createApi(Class<? extends
Api> apiClass) {

        Preconditions.checkNotNull(apiClass,
"Api class cannot be null");

        try {

            return (Api)
apiClass.newInstance();

        } catch (Exception e) {

            throw new
OAuthException("Error while creating
the Api object", e);

        }

    }


    public ServiceBuilder callback(String
callback) {

        Preconditions.checkNotNull(callback,
"Callback can't be null");

        this.callback = callback;

        return this;

    }


    public ServiceBuilder apiKey(String
apiKey) {

        Preconditions.checkEmptyString(apiKey
, "Invalid Api key");

        this.apiKey = apiKey;

        return this;

    }


    public ServiceBuilder apiSecret(String
apiSecret) {

        Preconditions.checkEmptyString(apiSecr
et, "Invalid Api secret");

        this.apiSecret = apiSecret;

        return this;

    }


    public OAuthService build() {

        Preconditions.checkNotNull(this.api,
"You must specify a valid api through
the provider() method");

        Preconditions.checkEmptyString(this.api
Key, "You must provide an api key");

        Preconditions.checkEmptyString(this.api
Secret, "You must provide an api
secret");

        return this.api.createService(new
OAuthConfig(this.apiKey, this.apiSecret,
this.callback, this.signatureType,
this.scope, this.debugStream));

    }

}
```

4. ANÁLISIS DE RESULTADOS

Una vez terminado el desarrollo de la aplicación móvil para Android, se puede tener en cuenta lo siguiente:

- Tomando como inicio los requerimientos que el cliente solicita, es claro cuál es el cumplimiento y la satisfacción del cliente con el mismo. Partimos de los requerimientos funcionales y no funcionales, siendo una aplicación informativa, donde la operatividad, la coherencia de datos y la cohesión de los mismos son parte fundamental del desarrollo de la aplicación
- Dar a conocer la mora producida en el municipio de Santuario, en la región de Planes de San Rafael, mediante el apoyo a la comercialización de la mora, a partir de la postproducción y los datos obtenidos por los análisis de los respectivos perfiles tanto del fruto como del suelo en el cual se produce
- Destacar dentro de la información presentada en la aplicación móvil, las características excepcionales que hacen ser al fruto producido en planes de San Rafael, un fruto de alta calidad, el cual se espera aumente sus niveles de comercialización a los presentados actualmente
- Dar conocimiento de la región en la cual se realiza la producción de la mora y la ubicación de los productores
- El aporte que se puede generar al gremio de la región, y que a largo tiempo puede impulsar la misma a través de un mercado como son las frutas, si este se llega a expandir
- La familiarización del desarrollo con el apoyo de un framework, esto con el fin de facilitar el diseño de la aplicación, de forma que sea agradable e intuitiva para el cliente final
- La reingeniería que se realiza a la aplicación, al tenerla como .apk, esto con el fin de encontrar las funciones principales, sobre las cuales se deben realizar cambios para llevar a cabo la funcionalidad, dependiendo de los requerimientos que se levantaron desde el incio con el cliente
- El aporte al inicio de una marca de origen para la región, siempre y cuando la asociación lo considere pertinente para avanzar con el proyecto de marca de origen, dando el plus que la fruta demanda.

5. CONCLUSIONES, APORTES Y RECOMENDACIONES

- Según los resultados de los estudios realizados con anterioridad, se puede concluir por parte del Grupo de Investigación de Oleoquímica que el fruto (mora), tiene características que la hacen excepcional, para lo cual se da a conocer con mayor profundidad en la sección “Nuestra Mora” de la aplicación móvil
- Se concluye, luego de la encuesta y entrevista realizadas a Ovidio Ledesma (Representante de la Asociación La Amorosa) que la comercialización actual del producto, podría aumentar considerablemente si se llega por medios electrónicos a nuevos y potenciales clientes
- Mediante el desarrollo del aplicativo móvil, se aporta tecnológicamente al reconocimiento de la zona en la cual se produce la fruta, y además es el primer paso para que esta se pueda convertir en la insignia representativa de Planes de San Rafael (Marca de Origen)
- Es recomendable que la aplicación sea actualizada constantemente
- Se recomienda que se realice una retroalimentación de los resultados que puede generar la aplicación a un mediano plazo (de 6 a 8 meses), y de ser positivos, adicionar y llevar a cabo actualizaciones pertinentes sobre los demás productos y servicios presentados por la Asociación y la Zona.

6. BIBLIOGRAFÍA

- Grupo de Investigación de Olequímica Universidad tecnológica de Pereira, (2013), Informe Técnico final: BIOPROSPECCIÓN DE METABOLITOS SECUNDARIOS CON VALOR NUTRACÉUTICO EN LOS MATERIALES CULTIVADOS DE MORA EN EL DEPARTAMENTO DE RISARALDA, Universidad Tecnológica de Pereira, Pereira, Risaralda
- Barrero Meneses, L.S. (2009), Caracterización, evaluación y producción de material limpio de la mora con alto valor agregado. Cundinamarca – Colombia. Editorial Produmedios, Corpoica, Colombia.
- Doctora Martha Leonor Marulanda y Ana María López, BIODIVERSIDAD Y BIOTECNOLOGÍA EN LA EVALUACIÓN Y SELECCIÓN DE CULTIVARES PROMISORIOS DE MORA DE CASTILLA, Grupo de Biodiversidad y biotecnología.
- Agrotón: maratón de desarrollo de aplicaciones para el agro colombiano en Corabastos, http://estrategia.gobiernoenlinea.gov.co/623/w3-article-8332.html
- Crea tu propia App para Android: http://www.elandroidelibre.com/2014/07/crea-tu-propia-app-para-android-con-goodbarber-3.html
- Good barber el mejor servicio 'web to app': http://orizhial.com/good-barber-el-mejor-servicio-web-app-del-universo/
- Alcaldía de Santuario – Risaralda: http://www.santuario-risaralda.gov.co/index.shtml
- Municipio de Santuario, Risaralda, Colombia: http://municipiodesantuario.blogspot.com.co/2006/04/parque-municipal-natural-planes-de-san.html
- Organización Colparques: http://www.colparques.net/RAFAEL
- Blog Vass digital: SCRUM la metodología de desarrollo ágil por excelencia http://vassdigital.com/blog/scrum-la-metodologia-de-desarrollo-agil-por-excelencia/
- I2B Intelligence to Bussiness: ¿Para qué sirve el Scrum en la metodología ágil? http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metogologia-agil/

- Ayuda de Developer Console: https://support.google.com/googleplay/android-developer/answer/113469?hl=es-419
- Definición de java: http://definicion.de/java/
- ¿Qué es JavaScript? - Definición de Javascript: http://www.masadelante.com/faqs/javascript

7. ANEXOS

1. Encuesta Mora, realizada en la Asociación La Amorosa, Ovidio Ledesma
2. Requerimientos.xls, Matriz de requerimientos
3. Casos de uso.doc, Diagramas de casos de uso y especificaciones de cada uno
4. Vista 1: Carga Aplicación
5. Vista 2: Menú principal
6. Vista 3: Nuestra Región
7. Vista 4: Eco-Conócenos
8. Vista 5: Nuestra mora
9. Vista 5.1: Conociendo nuestra mora
10. Vista 5.2: Perfil Nutracéutico
11. Vista 5.3: Perfil Sensorial
12. Vista 5.4: Perfil Nutricional
13. Vista 6: Nuestros cultivadores
14. Vista 6.1: Mapa de ubicaciones
15. Vista 7: Contáctanos
16. Vista 8: Acerca de…