PROCEDIMIETO PARA LA EDICION Y ADAPTACION DE PLUGINS ANDROID EN EL DESARROLLO DE APLICAIONES HIBRIDAS MOVILES

Autor
Juan Felipe Contreras Delgado

Director
EDAGAR ALEXIS ALBORNOZ ESPINEL
M.Sc en Computación

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS
PAMPLONA, NORTE DE SANTANDER – COLOMBIA
2017

Contenido

1 Introducción

Este capítulo contiene información de la importancia del presente trabajo de grado, da al lector un panorama de lo que trata este documento.

1.1 Resumen

El desarrollo de aplicaciones hibridas móviles está ligado a la integración de plugins que cumplen funcionalidades específicas, los desarrolladores eligen el plugin que mejor se adapte a sus requerimientos. Cuando el plugin elegido no es completamente satisfactorio estos recurren a descartar el requerimiento o a modificar el plugin por medio de prueba y error. Este proceso de edición tiene un coste en tiempo de edición bastante alto cuando el desarrollador no tiene la experiencia en ello. Por esta razón se plantea la creación de un procedimiento que pueda ser utilizado por desarrolladores interesados en la edición y adaptación de plugins que minimice los tiempos de edición y maximice los resultados esperados.

Palabras claves: Aplicación móvil, Adobe PhoneGap, plugin, edición de plugins, adaptación de plugins, aplicación hibrida.

1.2 PLANTEAMIENTO DEL PROBLEMA

El desarrollo de aplicaciones móviles es un mercado en crecimiento constante en nuestra era, diariamente se suben nuevas apps a las tiendas de aplicaciones de los diferentes sistemas operativos móviles. Estos sistemas operativos ofrecen sus propios lenguajes para desarrollar aplicaciones de forma nativa, lo que dificulta el desarrollo de apps multi plataforma. Por esta razón nace la iniciativa de las aplicaciones hibridas, apps desarrolladas para ser ejecutadas dentro de un WebView en los dispositivos móviles con soporte para varios sistemas operativos. Estas aplicaciones hibridas utilizan fragmentos de código (plugin) que contienen lenguaje de desarrollo nativo de cada una de las plataformas con la que es compatible. Estos plugin realizan tareas específicas como capturar una foto, verificar la conectividad del dispositivo, enviar un documento, ETC.

Sin embargo debido a que hay apps que requieren tareas específicas que no están cubiertas en su totalidad por los plugin existentes, se requiere la modificación y adaptación de estos plugins. Lo

que no es común en el desarrollo de aplicaciones hibridas, pero es de gran importancia para satisfacer requerimientos puntuales de clientes o de proyectos personales por parte de los desarrolladores móviles.

Los programadores de aplicaciones móviles para agilizar el desarrollo integran a sus proyectos plugins ya existentes y muchas veces deben conformarse con la funcionalidad preestablecida de estos, limitando el cumplimiento de los requerimientos exigidos por los clientes, cuando un plugin debe ser adaptado para mejorar sus funcionalidades el desarrollador debe a su criterio abordar el código e iniciar un proceso de prueba y error para lograr su adaptación. Si el desarrollador no cuenta con las habilidades necesarias el resultado será muy poco alentador o simplemente termine aceptando las funcionalidades preestablecidas y sacrificando los requerimientos del cliente. Es aquí donde surge la necesidad de proponer un procedimiento para la edición y adaptación de plugins, para la integración de estos en el desarrollo de aplicaciones hibridas basadas en Adobe PhoneGap.

1.3 JUSTIFICACION

La implementación de plugins en el desarrollo de aplicaciones hibridas es casi que necesario. Ya que la mayoría de estos plugins conectan el hardware del dispositivo móvil con la aplicación en el WebView, los desarrolladores de estos complementos crean funcionalidades generalizadas y

típicas para estos componentes de hardware. En la actualidad la demanda de aplicaciones hibridas crece constantemente al igual que los requerimientos que deben tener. Los plugins con funcionalidades limitadas y puntuales pueden detener el desarrollo de un proyecto, si se quiere cumplir con requerimientos específicos solicitados por los clientes. Los desarrolladores se ven obligados a decidir si toman un plugin limitado y se restringen en el cumplimiento de los requerimientos o se aventuran en adaptar los plugin sin ninguna guía. El procedimiento para abordar la edición y adaptación de plugins en Adobe PhoneGap tiene una utilidad práctica para los desarrolladores que se encuentren con la necesidad de modificar un plugin en su proyecto dando los lineamientos necesarios que faciliten la edición y adaptación, ahorrando tiempo de desarrollo y cumpliendo con los requerimientos planteados en el desarrollo de los aplicativos.

1.4 Objetivos

1.4.1 Objetivo General

Proponer un procedimiento para la edición y adaptación de plugins Android en el desarrollo de aplicaciones hibridas móviles.

1.4.2 Objetivos específicos

1.4.2.1 Realizar una revisión bibliográfica de los diferentes procedimientos y metodologías existentes para la adaptación de plugins en el desarrollo de software en dispositivos móviles.

1.4.2.2 Diseñar el procedimiento para la edición y adaptación de plugins en el desarrollo de aplicaciones hibridas para móviles basados en Android.

1.4.2.3 Desarrollar un aplicativo móvil en el cual se integre un plugin utilizando el procedimiento propuesto, donde se editara y adaptara el plugin para el sistema operativo Android.

1.4.2.4 Validar los resultados de la integración del plugin dentro del aplicativo desarrollado.

2 MARCO TEORICO

Este capítulo contiene información de la teoría y conceptos relacionados con los temas tratados en el presente documento.

2.1 Aplicación móvil o App:

Las aplicaciones móviles son uno de los segmentos del marketing móvil que mayor crecimiento ha experimentado en los últimos años. Se pueden encontrar en la mayoría de los teléfonos, incluso en los modelos más básicos (donde proporcionan interfaces para el envío de mensajería o servicios de voz), aunque adquieren mayor relevancia en los nuevos teléfonos inteligentes. El mundo de aplicaciones da un giro radical con la llegada de los smartphones y en especial con el lanzamiento del dispositivo iPhone a mediados de 2007. Con este lanzamiento, Apple cambia la manera de

interactuar con el teléfono, convirtiéndolo en un dispositivo intuitivo, potente, táctil y siempre online. Apple también desarrolla en profundidad el concepto de "tienda de aplicaciones": un mercado único y organizado donde la adquisición de las aplicaciones es transparente, fiable y directa. Además, incorpora un proceso definido y homogéneo de desarrollo para su sistema operativo: iOS, que ayuda a sacar el máximo potencial de las capacidades técnicas del teléfono, mejorando considerablemente la experiencia del usuario. Posteriormente, varios fabricantes, operadoras y empresas de software adoptan este modelo para ofrecer aplicaciones. Hasta el momento, Google y su sistema operativo Android libre y de código abierto ha logrado posicionarse como principal competidor. Adopta también el modelo de "mercado de aplicaciones" y con la ventajosa diferencia de que cualquier fabricante (HTC, Sony Ericsson o Samsung, entre otros) puede elegir libremente "Android" como sistema operativo para sus teléfonos. Este nuevo entorno de compra y las potentes funcionalidades, han propiciado un aumento de consumo de aplicaciones entre los usuarios, que ven cómo pueden satisfacer sus necesidades a través de las mismas. También ha influido en este crecimiento la llegada al mercado de las "tabletas", dispositivos móviles con una pantalla de mayores dimensiones y donde también pueden consumirse aplicaciones con algunas funcionalidades potenciadas. Por último, ha contribuido también la bajada en los precios de "Una aplicación móvil consiste en un software que funciona en un dispositivo móvil (teléfonos y tabletas) y ejecuta ciertas tareas para el usuario."

2.2 Aplicación nativa

Las apps nativas son desarrolladas en el lenguaje de programación de cada sistema operativo. Desde una aplicación nativa se puede acceder a todas las funcionalidades del dispositivo y el tiempo de acceso a la información es más rápido que en una web móvil. Por contra, debido a la complejidad de estos lenguajes de programación, el coste de desarrollo es más alto y para tener presencia en todos los sistemas operativos hay que multiplicar este coste por cada uno de ellos

2.3 Aplicación hibrida

Una aplicación híbrida es una aplicación nativa con HTML incrustado. Usando un framework de desarrollo común, las organizaciones pueden desarrollar aplicaciones multiplataforma que utilizan tecnologías web (como HTML, JavaScript y CSS), haciendo uso de las funciones del teléfono. Determinadas partes de la aplicación se programan utilizando tecnologías web. Esta opción permite a las empresas combinar los beneficios de las aplicaciones nativas y las webs móviles.

2.3 PhoneGap / Cordova

Es un marco de desarrollo móvil de código abierto. Se permite el uso de tecnologías web estándar - HTML5, CSS3 y JavaScript para el desarrollo multiplataforma. Las aplicaciones se ejecutan dentro de envoltorios específicos para cada plataforma, y se basan en los enlaces de la API compatibles con los estándares de acceso a las capacidades de cada dispositivo, tales como sensores, datos, estado de la red, etc.

2.4 WebView

Una visión que muestra las páginas web. Esta clase es la base sobre la que se puede rodar su propio navegador web o simplemente mostrar algunos contenidos en línea dentro de su actividad. Utiliza el motor de renderizado WebKit para mostrar páginas web e incluye métodos para navegar hacia adelante y hacia atrás a través de una historia, acercar y alejar, realizar búsquedas de texto y mucho más.

2.5 Plugin O Complemento

Los plugins son una parte integral del ecosistema Córdoba. Proporcionan una interfaz para Cordova y componentes nativos para comunicarse entre sí y los enlaces a las API del dispositivo estándar. Esto permite invocar código nativo de JavaScript.

Proyecto Apache Cordova mantiene un conjunto de plugins llamado el núcleo plugins. Estos complementos del núcleo proporcionan a su aplicación para acceder a las capacidades del dispositivo, tales como la batería, cámara, contactos, etc.

Además de los complementos del núcleo, hay varios plugins de terceros que proporcionan fijaciones adicionales para características no necesariamente disponibles en todas las plataformas.

3 ESTADO DEL ARTE

Este capítulo recopila información de investigaciones y estudios relacionados con los plugins de Cordova que se hayan realizados posteriormente.

3.1 Un plugin para el ambiente de trabajo PhoneGap permitiendo utilizar el SDK de la plataforma TestFlight

AÑO: 2013

AUTOR: Corella Pérez, Brian Alberto

NOMBRE: Plugin para PhoneGap: un plugin para el ambiente de trabajo PhoneGap permitiendo utilizar el SDK de la plataforma TestFlight

DESCRIPCION: Este proyecto se realizó con el objetivo de realizar el diseño y desarrollo de un plugin, para la empresa Avantica San Carlos, que permita integrar las funciones ofrecidas por el SDK de TestFlight en aplicaciones desarrolladas con PhoneGap. Avantica es una empresa que, desde hace varios años, brinda servicios a clientes de dispositivos móviles, y últimamente se ha dado la necesidad de implementar aplicaciones híbridas para facilitar el desarrollo multiplataforma; con el desarrollo de grandes proyectos, también es importante controlar de mejor forma la manera en que los usuarios interactúan con los sistemas.

BIBLIOGRAFIA:

Proyecto de Graduación (Bachillerato en Ingeniería en Computación) Instituto Tecnológico de Costa Rica, Escuela de Ingeniería en Computación, 2013.

URL:

http://repositoriotec.tec.ac.cr/bitstream/handle/2238/5687/Plugin%20para%20PhoneGap%20un%20plugin%20para%20el%20ambiente%20de%20trabajo.pdf?sequence=1&isAllowed=y

ABSTRACCION:

Este documento contiene información valiosa de cómo el autor elaboro el plugin que se propuso. El conocimiento de cómo elaborar un plugin permitirá conocer de una mejor manera como está estructurado y su orden jerárquico.

4 METODOLOGIA DE INVESTIGACION

Este capítulo describe la metodología que se empleara para desarrollar el proceso de investigación. Las características que se describen a continuación están basadas en (Hernández Sampieri, Fernández Collado, & Baptista Lucio, s.f).

4.1 PARADIGMA

El enfoque de este proyecto de investigación es cualitativo ya que no tiene una recolección de datos con medición numérica (uso de fórmulas estadísticas), sin algún tipo de predicción segura. Durante

la investigación se realiza la recolección de datos, basados en el análisis o el procesamiento de la información obtenida.

4.2 ALCANCE

El alcance de este proyecto de investigación es exploratorio ya que no se encuentran antecedentes de procedimientos para la edición de plugins en aplicaciones web hibridas.

4.3 TIPO

El tipo de este proyecto de investigación es no experimental debido a que no se hace uso de experimento alguno.

4.4 FUENTES DE INFORMACIÓN

Fue oportuno realizar una investigación en fuentes con cualidad virtual, utilizando documentos digitales y paginas oficiales las tecnologías, todos altamente confiables. Oportuno debido a que hacemos uso de la información a la que nos permite llegar la tecnología de forma más rápida.

4.5 POBLACIÓN

Este proyecto está dirigido a todos los desarrolladores independientes y/o a las empresas dedicadas al desarrollo de aplicaciones hibridas móviles que requieran adaptar plugins para cumplir con sus requerimientos funcionales que no son alcanzables con los plugins existentes. Por otro lado dirigido a los estudiantes de pregrado del programa de ingeniería de sistemas de la universidad de pamplona, que deseen editar los plugins de sus apps para volverlas más robustas o conocer el procedimiento.

4.6 INSTRUMENTOS

Como instrumento de recolección de información, se ha tomado como principal herramienta la observación y el seguimiento del flujo del código de los plugins para detectar sus características compartidas y en qué punto se debe editar el plugin.

4.7 CONSIDERACIONES ÉTICAS

El resultado principal de este proyecto (procedimiento para la edición y adaptación de plugins Android en el desarrollo de aplicaciones hibridas móviles) es libre para el uso de la persona que lo desee.

5 PROPUESTA DEL PROCEDIMIENTO

En el presente capitulo se diseñara el procedimiento para la edición de plugins en PhoneGap, donde se definirán las etapas y se aterrizar los diferentes pasos que se deben seguir para la edición de plugins.

El presente procedimiento está diseñado para ser aplicado cuando el desarrollador de la app agote todos los recursos de integrar diferentes plugins y no tener éxito en satisfacer su funcionamiento ideal. Es en ese momento cuando puede tomar un plugin y adaptarlo para que cumplan con los requerimientos y/o especificaciones que debe cumplir dentro de la aplicación que está desarrollando.

Aunque en el documento se describe un rol de Analista y un rol de desarrollador. Puede ser la misma persona la encargada de ejecutar todas las etapas.

5.1 PROCEDIMIENTO

A continuación se define la estructura del procedimiento **Hybrid Plugin Edit.** En sus diferentes etapas y estados (ver *FIGURA 5.1*).

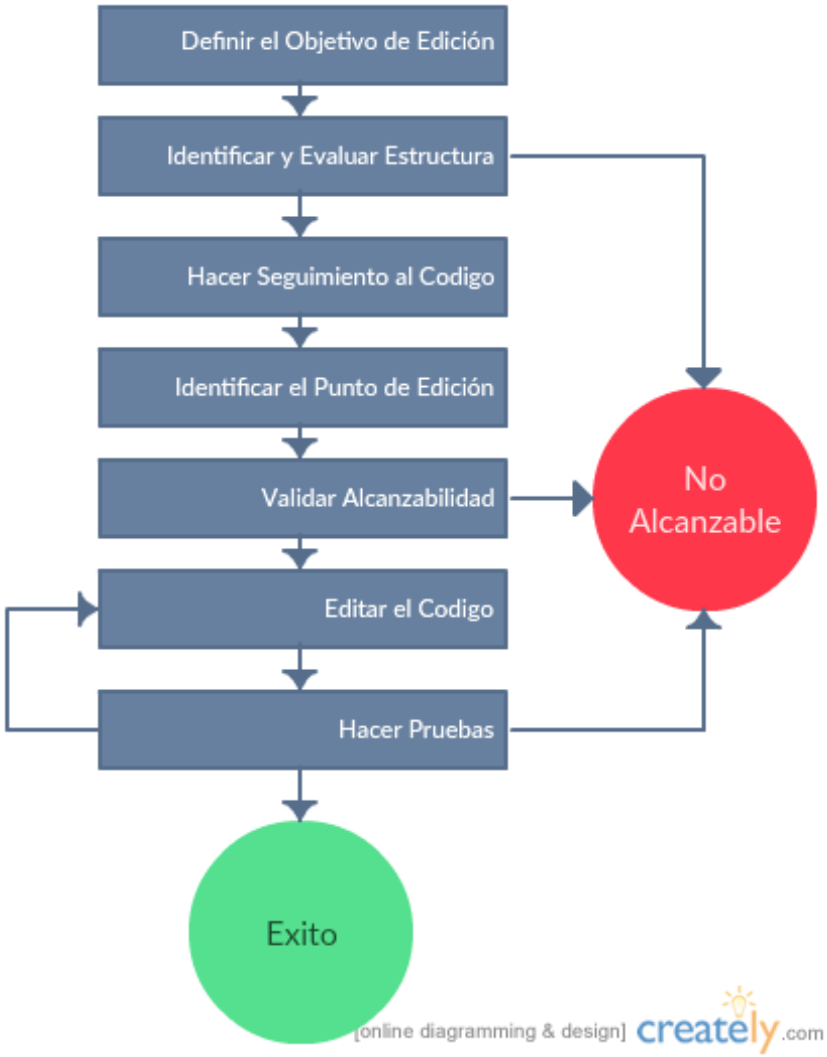

Figura 5.1: Procedimiento

5.1.1 Definir el Objetivo de edición

En esta etapa el analista, es el encargado de evaluar el estado actual. Describe el plugin que será sometido al procedimiento de edición y define cual será el estado requerido del plugin para satisfacer las funcionalidades ideales para el aplicativo en desarrollo.

Para el desarrollo de esta etapa, el analista hará uso del siguiente formato para describir el plugin y su funcionamiento esperado:

Formato 1. FORMATO DE OBJETIVO DE EDICION	
Analista:	**Fecha:**
Id del plugin:	**Versión del Plugin:**
Descripción del plugin:	

Tabla de requerimientos:

#	REQUERIMIENTO	ESTADO ACTUAL	ESTADO REQUERIDO

Observaciones:

Firma: _____ **C.C.**

En el formato encontramos los ítems:

- **Analista:** Nombre de la persona responsable de la ejecución de la etapa.
- **Fecha:** Fecha de diligenciamiento del formato.
- **Id del Plugin:** Identificador único del plugin en Phonegap/Cordova.
- **Versión del Plugin:** Numero de la versión del plugin utilizado
- **Tabla de requerimientos:** Describe cada uno de los requerimientos que se desea editar dentro del plugin, a continuación se especifica cada una de las columnas de la tabla:
  - **#:** Numeración incremental de los requerimientos para una rápida identificación.
  - **Requerimiento:** Nombre del requerimiento de edición, debe ser un nombre corto y que generalice la solución.
  - **Estado Actual:** Describe el funcionamiento actual del plugin con respecto al requerimiento de edición.

- **Estado Requerido:** Describe el funcionamiento esperado del plugin luego de la edición, con respecto al requerimiento de edición.
- **Observaciones:** Información de interés acerca del plugin o de los requerimientos de edición. La cual el analista considera importante.
- **Firma:** Firma del analista.
- **C.**C.: Numero de cedula de ciudadanía del analista.

Una vez que esté completo el Formato 1, el analista continua con la etapa dos del procedimiento "Identificar y Evaluar la Estructura".

5.1.2 Identificar y Evaluar la Estructura

Los plugins de Phonegap/Cordova deben cumplir una estructura estandarizada de Adobe, algunos desarrolladores hacen caso omiso a esta estructura. Lo que imposibilita leer y/o editar de manera clara el código.

En esta etapa el Analista validara si la estructura del plugin contiene los archivos necesarios para que el procedimiento pueda ser aplicado. A continuación se define la estructura básica que debe ser identificada al interior del plugin para poder ser estudiado (Ver Figura 5.1.2):

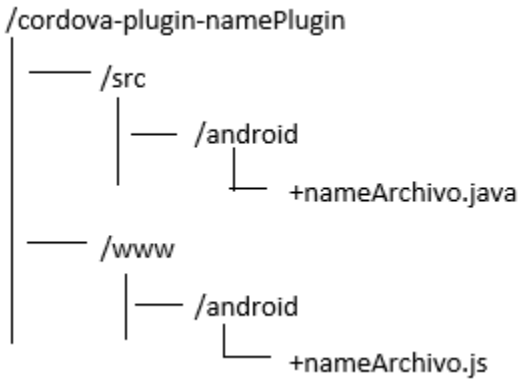

Figura 5.1.2: Árbol de direcciones

En el árbol de directorios que se encuentra en la parte superior podemos identificar:

- /cordova-plugin-namePlugin: Es el nombre de la carpeta que contiene el plugin que deseamos editar. En gran parte de los plugins, el nombre de la carpeta coincide con el id del plugin.
- /src: contiene los directorios de las plataformas soportadas por el plugin.
- /src/android: Contiene el o los archivos JAVA nativos del plugin seleccionado.

- +nameArchivo.java: Son los archivos de código JAVA que se ejecutaran de forma nativa en el dispositivo con sistema operativo android.
- /www: Contiene los directorios de las plataformas soportadas por el plugin.
- /www/android: Contiene el o los archivos JavaScript del plugin seleccionado.
- +nameArchivo.js: Son los archivos contenedores del código JavaScript que mediaran entre el WebView y los archivos de código nativos.

El objetivo de esta etapa es identificar el o los archivos nameArchivo.java así como el o los archivos nameArhivo.js. Para deducir que la estructura es **válida**, el número de archivos nativos (.java) y el número de archivos JavaScript (.js) deben ser mayor o iguales a uno. En caso contrario si el número de archivos (.js) o (.java) son igual a cero, la estructura se identifica como **invalida**.

Para esta etapa el analista deberá diligenciar el siguiente formato:

Formato 2. FORMATO DE VALIDACION DE ESTRUCTURA	
Analista:	**Fecha:**
Id del plugin:	**Versión del Plugin:**

Lista de chequeo:

Archivo	Existe		Observaciones
	SI	NO	
nameArchivo.js			
nameArchivo.java			

Resultado de estructura:

Valida		Invalida	

Firma:

_____ **C.C.**

En el formato 2, encontramos los ítems:

- **Analista:** Nombre de la persona responsable de la ejecución de la etapa.
- **Fecha:** Fecha del diligenciamiento del formato.
- **Id del plugin:** Identificador único del plugin en Phonegap/Cordova.
- **Versión del Plugin:** Numero de la versión del plugin.
- **Lista de chequeo:** Listado de archivos de código a identificar, contiene los siguientes columnas:
- **Archivo:** Hacer referencia a los dos tipos de archivos que se desean identificar dentro del árbol de direcciones.
- **nameArchivo.js:** Archivo de código JavaScript.
- **nameArchivo.java:** Archivo de código Java.
- **Existe:** Registra la existencia de los archivos.
- **SI:** Marcar con una X, si existe al menos un archivo.

- **NO:** Marcar con una X, si no existe ningún archivo.
- **Observaciones:** Información adicional que el analista considere pertinente acerca de la ubicación de los archivos dentro del árbol de directorios.
- **Resultado de estructura:** Proporciona información rápida si la estructura es Valida o Invalida. A través de la selección de uno de los siguientes dos ítems:
- **Valida:** Marcar con una X, si las columnas de "existe" de la "Lista de chequeo" están marcadas todas como "SI".
- **Invalida:** Marcar con una X, si al menos una de las columnas de "existe" de la "lista de chequeo" está marcada como "NO"
- **Firma:** Firma del Analista.
- **C.C.:** Numero de cedula de ciudadanía del Analista.

Si en el Formato 2, la casilla "**Valida"** en "Resultado de estructura" está marcada con una X. El Analista puede continuar a la siguiente etapa "Seguimiento de Código".

Si en el Formato 2, la casilla "**Invalida"** en "Resultado de estructura" está marcada con una X. El procedimiento estará en estado "No Alcanzable", lo que terminara con el procedimiento.

5.1.3 Seguimiento de Código

Esta etapa pretende encontrar el componente de código (Api, Clase, Método, Variable, ETC) encargado del comportamiento del estado actual de los requerimientos diligenciados en el Formato 1. Esta etapa puede ser ejecutada por el Analista valiéndose únicamente de su experiencia previa en el seguimiento, edición y creación de código.

El Analista hará un seguimiento al código para ver los posibles caminos de ejecución del plugin. Entendiéndose por camino de ejecución a las líneas de código que se ejecutan secuencialmente, desde la invocación del plugin hasta la respuesta del mismo. Es fundamental la experiencia y habilidad del Analista en el desarrollo y edición de código.

Una vez se tenga pleno entendimiento del flujo de ejecución del plugin, se procede a identificar el punto de edición de requerimiento.

Punto de edición de requerimiento: Es el API, clase, método, variable o componente de software. Encargado del funcionamiento del estado actual del requerimiento, identificado en la etapa "definir el objetivo de edición". Este punto se debe registrar en el siguiente formato:

Formato 3. FORMATO DE PUNTO DE EDICION

Encargado:	**Fecha:**
Requerimiento:	
Nombre:	**Tipo:**
Clase:	
Método:	
Ruta:	
Descripción:	

- **Encargado:** Nombre de la persona que diligencia el formato.
- **Fecha:** Fecha en la que se diligencia el formato.
- **Requerimiento:** Está compuesto por el número de requerimiento y el nombre del mismo.
- **Nombre:** es el nombre del API, clase, método, variable o componente de software.
- **Tipo:** Puede ser "API", "clase", "método", "variable" o el componente de software pertinente.
- **Clase:** Es la clase padre que contiene el componente de software.
- **Método:** Es el método que contiene al componente de software.
- **Ruta:** Dirección donde se encuentra el archivo.
- **Descripción:** Se realiza una descripción de lo observado en el punto de edición que el desarrollador considere puede ser de importancia para la edición deseable.

La documentación de Phonegap/Cordova nos provee de información acerca de una función JavaScript encontrada comúnmente en los plugins, haciendo uso de esto se presenta a continuación

una ruta de código que puede ser seguida por el Analista para encontrar el punto de edición del requerimiento.

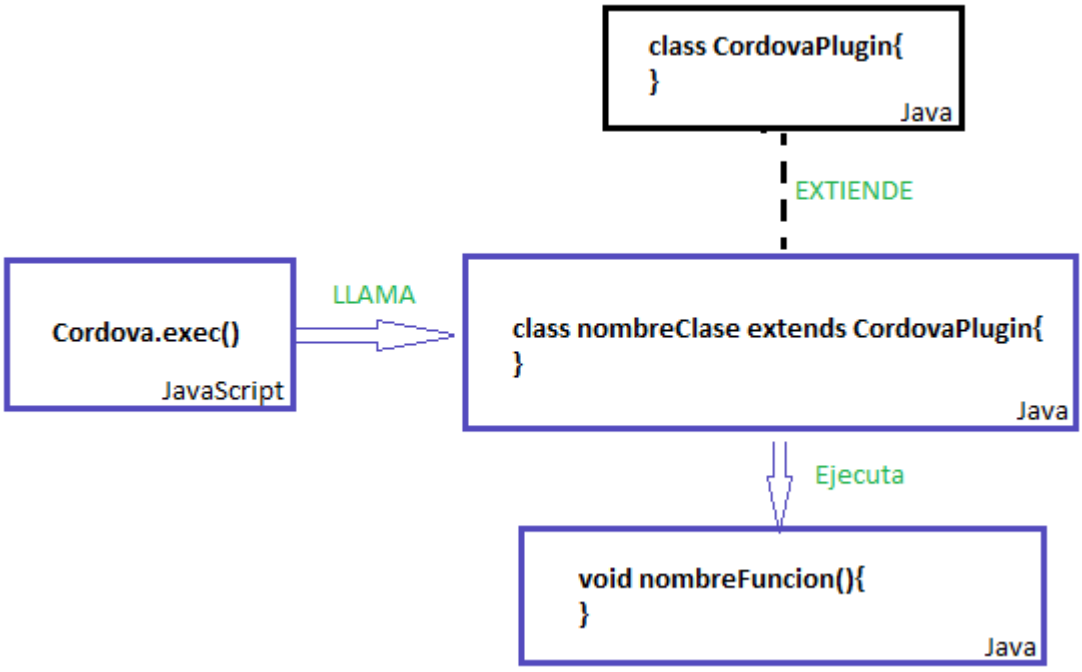

Figura 5.1.3: Línea de ejecución

En la Figura 5.1.3: Línea de ejecución, podemos identificar los métodos y clases que son llamados comúnmente en los plugins, a continuación se da una descripción generalizada de cada uno de ellos:

1 Cordova.exec(), es la función de JavaScript con la que PhoneGap se comunica con el código nativo Android.

El primer paso es encontrar esta función en el plugin que está contenida en uno de los archivos JavaScript de la ruta /www/android/.

Los parámetros de entrada de esta función son los siguientes:

```
cordova.exec (successCallback, failureCallback, class, method, [arguments]);
```

- **successCallback:** función que se ejecute cuando el resultado de la invocación sea satisfactorio.
- **failureCallback:** función que se ejecute cuando el resultado de la invocación no sea satisfactorio.
- **class:** nombre de la clase del código nativo, sin tener en cuenta el nombre del paquete.
- **method:** nombre de la acción que se va a ejecutar dentro de la clase invocada.
- **[arguments]:** Es un arreglo, generalmente en formato JSON, donde se le pasan todos los parámetros de entrada al método invocado.

2 Identificar la clase JAVA que se invoca en el método **exec()** en uno de los archivos de la ruta /src/android/. Esta clase tiene la particularidad que extiende de la clase CordovaPlugin.

3 Dentro de la clase es necesario identificar la acción invocada en el método **exec()**. Este método realiza la acción principal del plugin.

El éxito de esta etapa siempre estará basado en la destreza del analista en el desarrollo, lectura y edición de código de programación. Una vez el **Formato 3, Formato de punto de edición** este completo. Se procederá a la siguiente etapa "Validar la Alcanzabilidad de Requerimientos".

5.1.4 Validar la Alcanzabilidad de Requerimientos.

Esta etapa determina si los requerimientos del objetivo de edición son alcanzables o no. El Analista tiene la responsabilidad de evaluar si los requerimientos de edición pueden ser alcanzados, basado en la información conseguida previamente en los formatos de punto de edición. La capacidad técnica y experiencia del desarrollador es fundamental en esta etapa.

Si el Analista considera que el plugin puede ser editable para alcanzar el requerimiento de edición, dará una fundamentación en el lenguaje y/o palabras que el considere pertinente. En caso de considerar que el requerimiento no puede ser alcanzable deberá de igual manera fundamentar su determinación.

En la siguiente tabla se plasma la información de Alcanzabilidad de cada Requerimiento, es una variación a la tabla de la etapa "Definir el objetivo de edición":

Formato 4. FORMATO DE ALCANZABILIDAD	
Analista:	**Fecha:**
Id del plugin:	**Versión del Plugin:**
Tabla de Alcanzabilidad:	

#	REQUERIMIENTO	ALCANZABLE SI / NO	FUNDAMENTACION

Siguiente etapa:

Si		**No**	

Firma:

_____

C.C.

- **Analista:** Nombre de la persona responsable de la ejecución de la etapa.
- **Fecha:** Fecha del diligenciamiento del formato.
- **Id del plugin:** Identificador único del plugin en Phonegap/Cordova.
- **Versión del Plugin:** Numero de la versión del plugin.
- **Tabla de Alcanzabilidad:** Describe la Alcanzabilidad de cada uno de los requerimientos que se desea editar dentro del plugin, a continuación se especifica cada una de las columnas de la tabla:
- **#:** Numeración incremental, definida dentro del Formato 1 en la tabla de requerimientos.
- **Requerimiento:** Nombre del requerimiento de edición, definido dentro del Formato 1 en la tabla de requerimientos.
- **Alcanzable SI/NO:** El analista debe escribir "SI" al considerar que el plugin puede ser editado para satisfacer el requerimiento de edición y "NO" en caso que coincidiere lo contrario.
- **Fundamentación:** El analista debe escribir en palabras que considere pertinentes, la razón por la cual determina la Alcanzabilidad del requerimiento en la columna inmediatamente anterior.

- **Siguiente Etapa:** El analista marcara con una X si la edición continúa o no a la siguiente etapa del procedimiento, basado en que por lo menos uno de los requerimientos de edición deben ser alcanzable.
- **Firma:** Firma del Analista.
- **C.C.:** Numero de cedula de ciudadanía del Analista.

El formato aporta información específica en qué etapa se posicionara el procedimiento, si la casilla de "Siguiente Etapa" está marcada como "SI" el procedimiento continua a la etapa de "Edición de código". En caso de que este marcada como "NO" el procedimiento se posicionara en estado de inalcanzabilidad y el procedimiento termina.

Si tenemos requerimientos con valor SI en su columna de Alcanzabilidad, procederemos a la etapa de edición de código. Si por el contrario, la totalidad de requerimientos del objetivo de Alcanzabilidad tienen como valor NO el procedimiento entra en estado de inalcanzabilidad y el procedimiento termina.

5.1.5 Edición de código

El desarrollador procederá a realizar los cambios en el código basado en la tabla de Alcanzabilidad diligenciada en el Formato 4, su capacidad técnica y experiencia en la escritura, lectura y edición de código. Para documentar los cambios, el desarrollador debe diligenciar el siguiente formato:

Formato 5. FORMATO DE EDICION DE CODIGO

Desarrollador:	**Fecha:**
Requerimiento:	

Tabla de Modificaciones:

#	RUTA	MODIFICACION	FUNDAMENTACION

Observaciones:
Firma: _____ **C.C.**

- **Desarrollador:** Nombre de la persona que diligencia el formato.
- **Fecha:** Fecha en la que se diligencia el formato.
- **Requerimiento:** Está compuesto por el número de requerimiento y el nombre del mismo.
- **Tabla de Modificaciones:** Describe cada uno de los cambios que se harán en el plugin para alcanzar el objetivo de edición:
- **#:** Identificador, numeración incremental.

- **Ruta:** Ubicación del archivo que se modifica, la ruta debe ser relativa con respecto a la carpeta del plugin.
- **Modificación:** Descripción de la modificación realizada en el código, en términos que el desarrollador considere pertinentes.
- **Fundamentación:** El desarrollador debe escribir en palabras que considere pertinentes, la razón por la cual realiza la modificación.
- **Observaciones:** Información adicional que el desarrollador considera es pertinente para apoyar el proceso de la etapa de pruebas.
- **Firma:** Firma del desarrollador.
- **C.C.:** Numero de cedula de ciudadanía del desarrollador.

Una vez haya terminado la edición del código el procedimiento entrara en etapa de pruebas.

5.1.6 Pruebas

Las pruebas a realizar serán pruebas de caja negra, debido a que los requerimientos son funcionales y puntuales. El desarrollador debe diligenciar el siguiente formato:

Formato 6. PRUEBAS

Desarrollador:	**Fecha:**
Id del plugin:	**Versión del Plugin:**

# **Requerimiento:**	**Requerimiento:**

Oráculo:	**Bitácora:**

Determinación:

Éxito		**Inalcanzabilidad**		**Edición de Código**	

Observaciones:
Firma: _____ **C.C.**

- **Desarrollador:** Nombre de la persona responsable de la ejecución de la etapa.
- **Fecha:** Fecha del diligenciamiento del formato.
- **Id del plugin:** Identificador único del plugin en Phonegap/Cordova.
- **Versión del Plugin:** Numero de la versión del plugin.
- **# Requerimiento:** Numeración incremental, definida dentro del Formato 1 en la tabla de requerimientos.
- **Requerimiento:** Nombre del requerimiento de edición, definido dentro del Formato 1 en la tabla de requerimientos.

- **Oráculo:** Es el comportamiento y/o respuesta esperada por el desarrollador antes de ejecutar la prueba.
- **Bitácora:** Comportamiento y/o respuesta obtenida por el desarrollador luego de ejecutar la prueba.
- **Determinación:** Marcar con una X la opción que se ajuste a la prueba:
- **Éxito:** El desarrollador considera que se ha alcanzado el requerimiento de edición.
- **Inalcanzabilidad:** El desarrollador considera que no se alcanzó el objetivo de edición y no puede ser alcanzado.
- **Edición de código:** El desarrollador debe escribir "SI" al considerar que el plugin puede ser editado para satisfacer el requerimiento de edición y "NO" en caso que coincidiere lo contrario.
- **Observaciones:** El analista puede agregar en palabras que considere pertinentes cualquier agregado útil para el formato o el procedimiento.
- **Firma:** Firma del Desarrollador.
- **C.C.:** Numero de cedula de ciudadanía del Desarrollador.

El criterio del desarrollador definirá si el procedimiento regresara a la etapa de Edición de código, si entrara en estado de inalcanzabilidad o si estará en estado de Éxito. Basado en la determinación del formato 5.

El criterio del desarrollador, su destreza técnica y su experiencia será determinante para la correcta aplicación del procedimiento. Entendido por correcto no solo la edición del plugin si no también la determinación de estar en estado de inalcanzabilidad.

5.1.7 Estado de Éxito

El proceso estará en estado de éxito cuando el objetivo de edición sea cumplido.

4.1.8 Estado de No Alcanzabilidad

El proceso estará en estado de inalcanzabilidad si en alguna de sus etapas se determina que el objetivo de edición no puede ser alcanzado.

6 APLICACION DEL PROCEDIMIENTO

6.1 DESCRIPCION DE APLICACION

Esta etapa busca ejemplificar el procedimiento de edición de plugins, aplicándolo sobre un plugin instalado sin modificaciones en una app de prueba llamada **Cam Unipamplona**. Se hará paso a paso por cada una de las etapas del procedimiento. Esta app está desarrollada con Phonegap/Cordova para el sistema operativo Android. A continuación se describe la aplicación Cam Unipamplona y el plugin instalado a modificar.

Cam Unipamplona es una app de prueba desarrollada para mostrar el plugin cordova-plugin-media-custom de Phonegap/Cordova en ejecución, este plugin se encuentra hospedado en el repositorio de GitHub https://github.com/kmturley/cordova-plugin-media-custom y esta aceptado por Adobe como un plugin oficial. Su instalación se hace a través del comando:

```
Cordova plugin add cordova-plugin-media-custom
```

Este plugin permite grabar archivos de video, haciendo uso de la cámara del dispositivo Android. La particularidad de este plugin respecto a otros plugins de grabación de video, es que sobrepone una capa visual sobre la visualización de la grabación del video. Remplazando la capa que muestra el dispositivo Android por defecto.

La app Cam Unipamplona tiene una interfaz sencilla, en la cual podemos apreciar fácilmente su funcionamiento. (ver *FIGURA 6.1*).

Figura 6.1: interfaz app

- Identificamos en esta app en la parte superior un reproductor de videos, tiene un poster de pre visualización de la universidad de Pamplona para conservar la estética de la app. Una vez que sea grabado un video acá se podrá observar.
- Identificamos un botón rojo con la palabra "GRABAR" que nos servirá para iniciar la cámara de video y grabar un nuevo video.
- La sección "METADATA" la encontramos en la parte inferior, en ella se mostraran los detalles del video luego de ser grabado (Alto, ancho, duración).

6.2 APLICACIÓN DE PROCEDIMIENTO

A continuación se recorren cada una de las etapas del procedimiento **Hybrid Plugin Edit,** aplicándolo al plugin `cordova-plugin-media-custom`.

6.2.1 DEFINIR EL OBJETIVO DE EDICION

Como producto de esta etapa plasmamos los datos en el Formato 1 que nos ofrece el procedimiento.

Una vez diligenciamos el formato, avanzamos a la etapa dos del procedimiento "IDENTIFICAR Y EVALUAR LA ESTRUCTURA" como lo indica el procedimiento.

Formato 1. FORMATO DE OBJETIVO DE EDICION

Analista: Juan Felipe Contreras	**Fecha:** 30-05-2017
Id del plugin: Cordova-plugin-media-custom	**Versión del Plugin:** 1.0

Descripción del plugin:
Este plugin permite grabar archivos de video, haciendo uso de la cumara del dispositivo android. La particularidad de este plugin respecto a otros plugins de grabación de video, es que sobre pone una capa visual sobre la visualización de la grabación del video

Tabla de requerimientos:

#	REQUERIMIENTO	ESTADO ACTUAL	ESTADO REQUERIDO
1	Dimensiones	El plugin graba el video en la maxima resolucion posible por la cumara del dispositivo.	Se desea que las dimensiones del video siempre sean de 320x240 Pixeles.

Observaciones:
Se requere la adaptacion del plugin para hacer posible el envio del video sin consumir grandes cantidades de datos moviles y minimizar el coste de tiempo de transferencia.

Firma:
Juan Felipe Contreras

C.C. 1.044.266.399

6.2.2 IDENTIFICAR Y EVALUAR LA ESTRUCTURA

A continuación se muestra el árbol de direcciones del plugin `cordova-plugin-media-custom` (ver *Figura 6.2*)

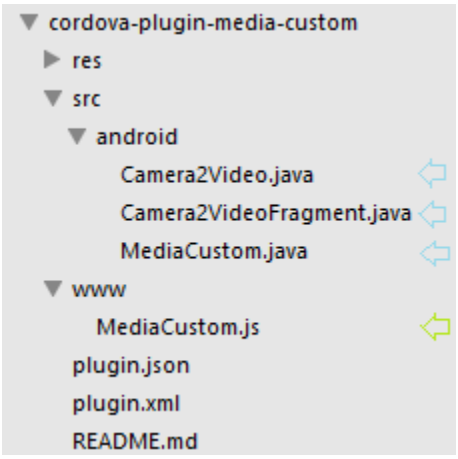

Figura 6.2

Con la ayuda de este árbol de direcciones plasmamos los datos en el Formato 2 que nos ofrece el procedimiento:

Formato 2. FORMATO DE VALIDACION DE ESTRUCTURA

Analista: Juan Felipe Contreras
Fecha: 30-05-2017
Id del plugin: cordova-plugin-media-custom
Versión del Plugin: 1.0

Lista de chequeo:

Archivo	Existe		Observaciones
	SI	NO	
nameArchivo.js	X		identifico 1 Archivo. MediaCustom.js
nameArchivo.java	X		identifico 3 Archivos. Camara2Video.java Camara2VideoFragment.java MediaCustom.java

Resultado de estructura:

Valida	X	Invalida	

Firma: Juan Felipe Contreras

C.C. 1.094.266.399

Podemos observar que la casilla "Valida" está marcada, por lo cual continuamos a la siguiente etapa "Seguimiento de código" como lo indica el procedimiento.

6.2.3 SEGUIMIENTO DE CODIGO

Para hacer seguimiento al código se agregaron impresiones de consola log.e(TAG, "Texto") como banderas como se muestra a continuación (Figura 6.2.3). Basado en mi experiencia como desarrollador como lo indicaba el procedimiento, considero que es la mejor forma para hacer seguimiento de código en ejecución.

```
521     private void setUpMediaRecorder() throws IOException {
522         final Activity activity = getActivity();
523         if (null == activity) {
524             return;
525         }
526         currentFile = getVideoFile();
527         mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
528         mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.SURFACE);
529         mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
530         mMediaRecorder.setOutputFile(currentFile.getAbsolutePath());
531         mMediaRecorder.setVideoEncodingBitRate(10000000);
532         mMediaRecorder.setVideoFrameRate(30);
533         Log.e(TAG, "mVideoSize: " + mVideoSize.getWidth() + " x " + mVideoSize.getHeight());
534         mMediaRecorder.setVideoSize(mVideoSize.getWidth(), mVideoSize.getHeight());
535         mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);
536         mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
537         int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
538         int orientation = ORIENTATIONS.get(rotation);
539         Log.e(TAG, "setOrientationHint: " + orientation);
540
541         mMediaRecorder.prepare();
542     }
```

Figura 6.2.3

Una vez fue identificado el punto de edición, plasmo los datos en el formato 3 como lo indica el plugin:

Formato 3. FORMATO DE PUNTO DE EDICION	
Encargado: Juan Felipe Contreras	**Fecha:** 30-05-2017
Requerimiento: Dimensiones de Video	
Nombre: mMediaRecorder.setVideoSize();	**Tipo:** Método
Clase: Camera2VideoFragment	
Método: SetupMediaRecorder()	
Ruta: /Cordova-plugin-media-custom/src/android/Camera2VideoFragment.java	
Descripción: identifico que en este punto el plugin asigna las dimensiones de ancho y Alto al video que sera grabado. Propongo que en este punto se asigne las dimensiones de 320 pixeles de ancho y 240 pixeles de alto	

Una vez diligenciado el formato prosigo a la siguiente etapa "Validar la Alcanzabilidad de Requerimientos" como lo indica el procedimiento.

6.2.4 VALIDAR LA ALCANZABILIDAD DE REQUERIMIENTO

Basado en el formato de punto de edición procedemos a evaluar la Alcanzabilidad del requerimiento, como producto de esta etapa diligenciamos el Formato 4:

Formato 4. FORMATO DE ALCANZABILIDAD

Analista: Juan Felipe Contreras	**Fecha:** 30-05-2017
Id del plugin: Cordova-plugin-media-custom	**Versión del Plugin:** 1.0

Tabla de Alcanzabilidad:

#	REQUERIMIENTO	ALCANZABLE SI / NO	FUNDAMENTACION
1	Dimensiones de Video	SI	Basado en el formato de punto de edición, se considera viable la solucion de pasar las dimensiones del video en el metodo mMediaRecorder.setVideoSize()

Siguiente etapa:

SI	X	No	

Firma: Juan Felipe Contreras

C.C. 1.094.266.394

La casilla de siguiente etapa está marcada en "Si" lo que indica que el procedimiento continua a la etapa de edición de código.

6.2.5 EDICION DE CODIGO

Con ayuda de los formatos generados en las etapas anteriores se realiza la edición del código (Ver Figura 6.2.5):

```
private void setUpMediaRecorder() throws IOException {
    Log.e(TAG, "primero19");
    final Activity activity = getActivity();
    if (null == activity) {
        return;
    }

    currentFile = getVideoFile();
    mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.SURFACE);
    mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    mMediaRecorder.setOutputFile(currentFile.getAbsolutePath());
    mMediaRecorder.setVideoEncodingBitRate(500000);
    mMediaRecorder.setVideoFrameRate(20);
    //Log.e(TAG, "mVideoSize: " + mVideoSize.getWidth() + " x " + mVideoSize.getHeight());
    mMediaRecorder.setVideoSize(320, 240);
    mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);
    mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
    //mMediaRecorder.setMaxDuration(10000);
    int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
    int orientation = ORIENTATIONS.get(rotation);

    mMediaRecorder.prepare();
    Log.e(TAG, "aca19");
}
```

Figura 6.2.5

Como producto de esta etapa se genera el diligenciamiento del formato 5:

Formato 5. FORMATO DE EDICION DE CODIGO

Desarrollador: Juan Felipe Contreras | **Fecha:** 30-05-2017

Requerimiento: Dimensiones de Video

Tabla de Modificaciones:

#	RUTA	MODIFICACION	FUNDAMENTACION
1	./Cordova-plugin-media-custom/src/android/Camera2VideoFragment.java	Se pasan los parametros de 320 y 240 al metodo mMediaRecorder.setVideoSize(320,240);	Con estos campos quemados como parametros de entrada al metodo, se asegura que la configuración de grabacion de video siempre estara en 320 x 240 pixeles en ancho y alto respectivamente.

Observaciones:
Se recomienda extraer los metadatos en el previsualizador del video, en las pruebas. Para obtener las dimensiones reales del video grabado.

Firma:
Juan Felipe Contreras

C.C. 1.094.266.399

El procedimiento avanza a la etapa de "Pruebas" una vez se ha diligenciado el formato 5.

6.2.6 PRUEBAS

Basado en el formato 5, se realiza una prueba de grabación de video con el plugin modificado. En el cual los metadatos serán traídos directamente de la etiqueta de video, con el objetivo de comprobar los datos reales de las dimensiones del video (Figura 6.2.6.1).

```
video.addEventListener('loadeddata', function (e) {
    window.setTimeout(function () {
        output.innerHTML += 'duration = ' + e.target.duration + '<br/>';
        output.innerHTML += 'videoWidth = ' + e.target.videoWidth + '<br/>';
        output.innerHTML += 'videoHeight = ' + e.target.videoHeight + '<br/>';
    }, 200);
});
```

Figura 6.2.6.1

A continuación se muestra la aplicación antes de grabar el video (Ver Figura 6.2.6.2)

Figura 6.2.6.2

Cuando el botón de grabar es accionado se inicia la grabación de video con el plugin modificado.

(Ver Figura 5.2.6.3)

Ver Figura 6.2.6.3

Una vez que el video ha finalizado se puede apreciar la extracción de los metadatos (Ver Figura 6.2.6.4)

Figura 6.2.6.4

Basado en la prueba realizada, se procede a diligenciar el formulario 6:

Formato 6. PRUEBAS

Desarrollador: Juan Felipe Contreras	**Fecha:** 30-05-2017
Id del plugin: Cordova-plugin-media-custom	**Versión del Plugin:** 1.0
# Requerimiento: 1	**Requerimiento:** Dimensiones de Video

Oráculo:
Se espera que el video tenga dimensiones de 320 y 240 pixeles es su Ancho y Alto respectivamente

Bitácora:
Como se puede apreciar en los metadatos del video:
VIDEOWIDTH = 320
VIDEOHEIGH = 240
Obtenemos un video de 320 y 240 pixeles de Ancho y Alto respectivamente en sus dimensiones.

Determinación:

Éxito	X	Inalcanzabilidad		Edición de Código	

Observaciones:
Se considera que la edicion se culmina con total exito.

Firma:
Juan Felipe Contreras

C.C. 1.094.266.399

La prueba arroja como "Determinación" el estado de "Éxito". Lo que posiciona al procedimiento en estado de éxito y se da por terminado el procedimiento con el requerimiento de edición cumplido.

6.3 Análisis de resultados

El resultado más importante y propósito de este trabajo de grado es el procedimiento de edición de plugins **Hybrid Plugin Edit,** aplicado de manera exitosa a nuestra app de prueba **Cam Unipamplona.** La aplicación de las etapas del procedimiento sobre la edición del plugin de cámara seleccionado, permite ver la efectividad del mismo para conseguir la edición de un plugin y su adaptación a la medida de los requerimientos del desarrollador. Esto permite reducir las limitaciones que tienen las aplicaciones hibridas al momento de ser seleccionadas como herramienta para el desarrollo de aplicaciones móviles.

El éxito de la edición del plugin está basado en la etapa de pruebas del procedimiento aplicado a Cam Unipamplona, donde se consigue un video grabado en las dimensiones deseadas de 320 x 240 Pixeles.

7 Conclusiones

Se realiza la revisión bibliográfica de los diferentes procedimientos y metodologías existentes para la adaptación de plugins en el desarrollo de software en dispositivos móviles. Dejando como resultado la inexistencia de ellos. Dando un valor agregado al presente trabajo de grado de gran aporte para desarrolladores y estudiantes.

El procedimiento de adaptación de plugins **Hybrid Plugin Edit** destaca en su facilidad de aplicación, obteniendo en cada una de sus etapas como producto un formato diligenciado por el encargado que facilita la conservación de la información y la evaluación de la misma. Así pueden distintos profesionales intervenir en la edición del mismo plugin.

El aplicativo Cam Unipamplona desarrollado para Android y basado en Phonegap / Cordova permite evidenciar la edición de un plugin sobre esta tecnología. El plugin editado es totalmente funcional y facilita la manipulación de archivos multimedia sobre dispositivos móviles. Disminuyendo su coste de almacenamiento y transferencia.

La etapa de pruebas del procedimiento permite evidenciar el éxito del procedimiento para alcanzar el objetivo de edición sobre el plugin seleccionado.

8 Referencias Bibliográficas

Mobile Marketing Association (MMA), 2011. Libro Blanco de apps / Guía de apps móviles. Recuperado el 20 de enero del 2016, de http://www.slideshare.net/mmaspain/libro-blanco-de-las-aplicaciones-mviles

Mobile Marketing Association (MMA), 2011. LIBRO BLANCO WEB MÓVILES. Recuperado el 18 de enero del 2016, de http://www.slideshare.net/mmaspain/libro-blanco-de-las-webs-mviles

Google Inc, 2016. *Documentación oficial Android - WebView recuperado el 31 de Marzo de 2016 de: http://developer.android.com/intl/es/reference/android/webkit/WebView.html*

Apache Software Foundation, 2015. Documentación oficial Apache. Recuperado el 22 de Enero de 2016 de: https://cordova.apache.org/docs/en/latest/guide/overview