(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 655 207**

(51) Int. CI.:

G06F 9/445 (2006.01)
G06F 9/44 (2006.01)
G06Q 10/00 (2012.01)
G06Q 30/00 (2012.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **18.12.2014 E 14198847 (7)**

(97) Fecha y número de publicación de la concesión europea: **15.11.2017 EP 2889813**

(54) Título: **Método y sistema para implementar conjuntos de herramientas de desarrollo de software en aplicación**

(30) Prioridad:

27.12.2013 US 201361921287 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
19.02.2018

(73) Titular/es:

BUONGIORNO S.P.A. (100.0%)
Borgo Omero Masnovo, 2
43121 Parma, IT

(72) Inventor/es:

PIUNNO, SIMONE

(74) Agente/Representante:

TEMIÑO CENICEROS, Ignacio

DESCRIPCIÓN

Método y sistema para implementar conjuntos de herramientas de desarrollo de software en aplicación

Campo de la invención

La presente invención se refiere a métodos para el desarrollo de conjuntos de herramientas de desarrollo de software propios (denominados a continuación en el presente documento, “SDK”) para permitir las aplicaciones de software existentes (denominadas a continuación en el presente documento, “aplicaciones”), preferiblemente aplicaciones de Android, se conectan a métodos de pago de aplicación alternativos, y específicamente funcionalidades de facturación de portador directas, por tanto, pudiendo distribuir dichas aplicaciones fuera de una única tienda de aplicaciones (por ejemplo, en mercados alternativos a la tienda de Google Play) con una integración fácil. La invención también se refiere a sistemas configurados para implementar dichos métodos.

Antecedentes de la invención

Google Play es una tienda de aplicaciones gestionada por Google en la que usuarios pueden cargar sus aplicaciones compatibles con Android. Desarrolladores de terceros que usan sistemas de herramientas de desarrollo de software de Android pueden construir aplicaciones como archivos de paquete de aplicación binaria empaquetada (“APK”) y cargarlos a la tienda de Google Play para su distribución a usuarios finales.

Mediante Google Play, estas aplicaciones pueden encontrarse por usuarios finales, descargarse e instalarse en dispositivos basados en Android. Google proporciona adicionalmente un conjunto de servicios útiles para implementar funcionalidades comunes en aplicaciones de Android, tales como mensajería, notificación asíncrona, actualización de aplicaciones, y pago en aplicaciones. Estas se denominarán “servicios de Google propios” o PGS. Los mismos normalmente solicitan otros servicios propios (“PS”) propiedad de otras empresas.

Google Play no es la única plataforma de distribución posible para aplicaciones de Android. Existen tiendas alternativas tales como la tienda de Amazon y métodos de pago alternativos. Desarrolladores que quieren distribuir también mediante estos canales alternativos tienen prohibido usar PGS y, por tanto, tienen que construir un paquete APK diferente vinculado a implementaciones alternativas de estos servicios. Esto requiere esfuerzos de desarrollo adicionales que desaconsejan habitualmente el uso de plataformas alternativas y pasa a ser una barrera técnica para los competidores de Google. Un ejemplo de cómo puede desarrollarse código de programa de aplicación según requisitos y especificaciones proporcionados por una plantilla puede encontrarse en el documento US5671415.

Breve descripción de la invención

La presente invención proporciona un método de desarrollo de SDK propios para construir APK de archivos alternativos con algoritmos automatizados para cambiar de manera dinámica el comportamiento dependiendo del contexto (tal como país, operador de móvil o flujo de “participación” requerido) y esfuerzo mínimo para el desarrollador, simplificando el proceso técnico para publicar aplicaciones de Android en canales de distribución alternativos.

Un objeto de la invención se refiere a un método para implementar conjuntos de herramientas de desarrollo de software en aplicación que comprende:

- implementar lógicas de aplicación en un lenguaje de programación compatible con un sistema operativo de máquina virtual;

- proporcionar una o más bibliotecas de compatibilidad propios proporcionadas por un primer proveedor de servicios;

- usar un entorno de desarrollo de software para vincular las lógicas de aplicación y las bibliotecas de compatibilidad propios entre sí, y empaquetar las mismas en un binario APK;

- cargar el binario APK a una tienda de aplicaciones asociada al primer proveedor de servicios;

en el que el método comprende además al menos las etapas siguientes:

- incluir bibliotecas de compatibilidad propios adicionales proporcionadas por segundos proveedores de servicios que implementan la integración con canales de distribución alternativos en lugar de las proporcionadas por el primer proveedor de servicios;

- usar dicho entorno de desarrollo de software para vincular las lógicas de aplicación y las bibliotecas de compatibilidad propios adicionales entre sí en binarios APK adicionales que pueden cargarse a tiendas de aplicaciones alternativas.

En una realización preferida de la invención, las bibliotecas de compatibilidad propias adicionales incluidas comprenden uno o más de los métodos de función siguientes:

- método de función para determinar si se soporta un pago dependiendo del país, el operador de móvil y/o el tipo de compra;

- método de función para volver a una lista de identificadores de productos de artículos ya comprados y sus detalles, y/o para saber si ya se ha comprado un artículo;

- método de función para volver a una lista de todos los artículos que pueden comprarse y sus detalles, que incluyen precio, título, descripción y tipo de compra;

- método de función para iniciar un flujo de compra y permitir la interacción de usuario dentro de la aplicación;

- método de función para marcar una compra como consumida y/o permitir compras adicionales del mismo artículo.

Un objeto adicional de la invención se refiere a un sistema para implementar conjuntos de herramientas de desarrollo de software en aplicación que comprende software, hardware y/o medios de software intermedio configurados para llevar a cabo un método según cualquiera de las realizaciones descritas en el presente documento.

En una realización preferida de la invención, el sistema comprende un software intermedio de servidor implementado en una agrupación de servidores de aplicaciones, configurado para mandar instrucciones a uno o más de los componentes siguientes:

- base de datos de artículos de inventario para una aplicación en una tienda;

- base de datos de artículos comprados para cada aplicación y usuario en una tienda;

- medios de definición de flujo de compra para un operador de móvil y método de pago asociado;

- máquina de estados finitos que implementa el flujo de compra durante una sesión de compra;

- base de datos de sesiones de compra en curso y sus estados;

- conectores y adaptadores de protocolo para un operador de móvil y método de pago asociado;

- API de JSON usada por SDK;

- registro de notificación para estadísticas;

- traza de auditoría.

Como ejemplo relacionado con servicios de Google propios (aunque también puede aplicarse a otros servicios propios de aplicaciones introduciendo las modificaciones correspondientes relacionadas con lenguajes de programación, sistemas operativos y especificaciones de tienda específica y requisitos de APK), el proceso técnico implicado en la producción de un archivo APK la carga a Google Play incluye las etapas siguientes:

a) El desarrollador tiene que implementar la lógica de aplicación en lenguaje de programación Java compatible con el sistema operativo de máquina virtual Dalvik.

b) El desarrollador tiene que incluir bibliotecas de compatibilidad propios proporcionadas por Google.

c) Al usar un entorno de desarrollo de software, las dos partes se vinculan entre sí y empaquetan en un binario APK.

d) El binario APK se carga a Google Play.

El método de la invención incluye también al menos las etapas siguientes:

e) El desarrollador incluye bibliotecas de compatibilidad propios adicionales que implementan la integración con un canal de distribución alternativo en lugar de las proporcionadas por Google en la etapa (b)

f) Al usar el mismo entorno de desarrollo de software de la etapa (c), las partes se vinculan de nuevo entre sí en un nuevo binario APK que puede cargarse a una tienda alternativa.

Mediante este método, un competidor de Google en el campo de distribución de aplicaciones de Android puede

convencer fácilmente a los desarrolladores de aplicaciones de comprometerse a la distribución en la tienda del competidor.

Descripción de las figuras

La figura 1 muestra un diagrama de secuencia que representa las etapas de un proceso de compra según una realización de la invención.

La figura 2 muestra una realización de la arquitectura de software intermedio y SDK según una realización de la presente invención.

La figura 3 muestra un diagrama de secuencia que representa una sesión de compra según una realización de la invención.

Descripción detallada de la invención

Como ejemplo relacionado con una realización de la invención aplicado a servicios de Google propios, el método propuesto puede implementarse mediante al menos una biblioteca de soporte que es compatible en el nivel de interfaz de programación de aplicación ("API") con la proporcionada por Google Play, además de una plataforma de lado de servidor que implementa funcionalidades similares a las proporcionadas por Google, por ejemplo, para el pago en aplicaciones. Estas funcionalidades pueden conectarse a diferentes servidores finales y, por ejemplo, el pago en aplicaciones puede implementarse con facturación de operador en lugar de transacciones de tarjeta de crédito. La biblioteca de soporte comprenderá preferiblemente los siguientes métodos de función (véase la figura 1 para un ejemplo de un proceso de compra completo según la invención):

• IsBillingSupported() - Método de función para determinar si se soporta un pago dependiendo del contexto (país, operador de móvil, tipo de compra).

• GetPurchases() - Método de función para volver a una lista de ID de productos de artículos ya comprados y sus detalles, o para saber si un artículo ya se compró.

• GetSkuDetails() - Método de función para volver a una lista de todos los artículos que pueden comprarse y sus detalles, que incluyen precio, título, descripción y tipo de compra.

• GetBuyIntent() - Método de función para iniciar un flujo de compra y permitir la interacción de usuario dentro de la aplicación.

• ConsumePurchase() - Método de función para marcar una compra como consumida y permitir compras adicionales del mismo artículo (cuando el artículo se configura como consumible).

Mediante los métodos de función anteriores, la API de soporte interactuará con un software intermedio de servidor global (véase la figura 2 para un ejemplo de la arquitectura de software intermedio y SDK según la invención) que es un sistema de servidor conectado por medio de consultas JSON/REST por la red. Este sistema se implementa en una agrupación de servidores de aplicaciones que mandan instrucciones a diferentes componentes de servidor y proveedores de pago. Todas las comunicaciones se protegerán y cifrarán con protocolo HTTPS, que usa un par de certificados generados por claves públicas en los que la clave pública formará parte de los archivos de biblioteca de soporte y la clave privada almacenada en los servidores de software intermedio globales.

La función IsBillingSupported() puede implementarse recogiendo información del sistema operativo de un teléfono (tal como el código IMSI de una tarjeta SIM) o la dirección IP y con estos parámetros llamar al software intermedio para computar a qué país y operador pertenece el usuario, quién es el usuario (por ejemplo, por medio de reconocimiento de número 3G o mapeo de IMSI a MSISDN o a dirección de correo electrónico con una búsqueda de base de datos) y si finalmente la clase de transacción (única o subscripción) está permitida en este contexto. Esta llamada permite también identificar al usuario y tener códigos de autorización necesarios para los otros métodos.

El software intermedio global gestiona tablas con el inventario de artículos que pueden comprarse para cada aplicación, tablas de precios independientes para cada operador de móvil, listas de artículos ya comprados para cada usuario de cada aplicación, listas de aplicaciones instaladas.

Los métodos GetPurchases() y GetSkuDetails() pueden implementarse solicitando al software intermedio que busque las tablas de inventario y los detalles de artículos y notificar un documento de JSON con la lista de todos los artículos propiedad del usuario. Los usuarios se identifican por su cuenta de correo electrónico (proporcionada por un sistema operativo Android) y las aplicaciones se identifican por su certificado y cadena de nombre de paquete. La combinación de los tres artículos define el conjunto de artículos que va a extraerse por las tablas.

El software intermedio global también mantiene conexiones con la red central y los sistemas de pago de diversos

operadores de móvil en muchos países, implementando sus protocolos de comunicación propios y adaptando automáticamente la comunicación de modo que todos los operadores se presentan a la biblioteca de soporte como una API uniforme y única. El flujo de compra se describe en estas API como máquina de estados finitos con número arbitrario de etapas, en la que se representa la etapa de una acción de introducción solicitada al usuario en el flujo de participación. En una realización de la invención, el software intermedio contiene los siguientes componentes:

• Base de datos de artículos de inventario para cada aplicación.

• Base de datos de artículos comprados para cada aplicación y usuario.

• Definición de flujo de compra para cada operador y método de pago.

• Máquina de estados finitos que implementa el flujo de compra durante una sesión.

• Base de datos de sesiones de compra en curso y sus estados.

• Conectores y adaptadores de protocolo para cada operador y método de pago.

• API de JSON usada por SDK.

• Registro de notificación para estadísticas.

• Traza de auditoría.

El GetBuyIntent() llama al software intermedio para empezar una sesión de compra, pasando la referencia de almacén (SKU) del artículo que va a comprarse y recibiendo una sesión. La interacción de usuario se gestiona con una actividad de Android implementada en la biblioteca de soporte como o bien un formato nativo o bien una vista web. La sesión se contextualiza para el método de pago y operador de móvil particulares y por consiguiente la apariencia del formato adaptado. La biblioteca de soporte puede llamar al software intermedio en cada etapa para enviar información adquirida al servidor y devuelva instrucciones sobre qué es necesario realizar en la siguiente etapa (qué clase de formato viene después, qué información va a adquirirse con el fin de avanzar). La información obtenida se almacena de manera incremental en la terminación del software intermedio en una tabla de sesión y solo cuando el conjunto está completo se realiza la transacción, enviando instrucciones apropiadas (diferentes para cada método de pago y operador de móvil) a los servidores relevantes. Cuando la sesión está completa, el software intermedio devolverá un código de éxito y la actividad de compra se cerrará, devolviendo el control al juego.

El método ConsumePurchase() puede implementarse con una llamada al software intermedio con el ID de Sku y hará que el artículo usado de modo que en la tabla de artículos propiedad del usuario no aparezca más como en propiedad y pasará a ser posible una nueva compra para este ID de Sku. Esto es particularmente útil para aquellos artículos que son “consumibles” o “repetibles”, lo que significa que usuarios pueden comprarlos más de una vez.

Una interfaz de administración de lado de servidor puede proporcionar al desarrollador la capacidad de configurar parámetros del método de distribución adicional, o la capacidad de proporcionar por sí mismo el binario APK en la tienda alternativa. Una versión de marca blanca de esta interfaz de administración puede impulsar diferentes tiendas bajo diferentes marcas.

Algunos ejemplos de las acciones que pueden realizarse mediante la interfaz de administración son:

• Extraer notificaciones sobre las aplicaciones instaladas, actividad de los usuarios, transacciones de pago por hora o día.

• Leer la lista de aplicaciones configurada por un desarrollador.

• Configurar la lista de países en los que una aplicación dada está aprobada para su distribución.

• Leer y cambiar la tabla de inventario de artículos que pueden comprarse para una aplicación y país dados, con parámetros tales como tipo de artículo (subscripción, repetible, no repetible), precio, título, descripción, SKU.

• Crear un nuevo perfil de aplicación (con título, ID de aplicación, nombre de paquete, par de certificados)

• Cargar un nuevo APK para una aplicación dada.

• Publishing un APK para una URL dada.

• Enviar mensajes a aplicaciones instaladas, por ejemplo para iniciar una actualización.

En particular, para el pago en aplicaciones basado en la facturación de operador, tiene que superarse una complejidad adicional debido a que operadores diferentes requieren diferentes flujos de experiencia de compra y cambian frecuentemente los requisitos a cumplir con políticas y regulaciones nacionales en constante cambio.

Una aplicación pensada para distribución global se conectará con más de 100+ operadores diferentes, requiriendo todos flujos ligeramente diferentes. Cada uno de estos operadores actualizará habitualmente los requisitos locales anualmente, lo que significa que el conjunto de flujos acumulados en la aplicación tiene que actualizarse muy a menudo.

La biblioteca de soporte puede superar esta complejidad cargando de manera dinámica instrucciones etapa por etapa en el flujo de experiencia de compra específico requerido por el operador al que el teléfono que está ejecutando la aplicación está conectado y adaptando automáticamente la experiencia de compra para cumplir la regulación local. Una sesión habitual (figura 3) puede suceder mediante estas etapas:

1. Solicitar al software intermedio que identifique el contexto (país, operador, métodos de pago disponibles).

2. Solicitar al software intermedio que identifique parámetros del artículo que va a comprarse (nombre, descripción y precio).

3. Iniciar una sesión de compra para rastrear las etapas y conseguir la lista de parámetros para las primeras etapas como tipo de formato que va a visualizarse, cadenas y logos de imagen que va a rellenarse en la plantilla, nombre del campo de introducción que va a recogerse, por ejemplo un formato para mostrar al usuario su número de teléfono, nombre y precio del artículo que va a comprarse, indicación de los términos y condiciones legales de la compra, aviso legal de privacidad y un botón de OK para aprobar.

4. Procesar el formulario según se define dinámicamente en la etapa previa.

5. Esperar que el usuario interaccione con el formato realizando la acción solicitada (hacer clic en un botón, pulsar una zona de la pantalla, rellenar un campo, etc), por ejemplo pulsar en el botón de OK.

6. Enviar de vuelta al software intermedio el campo de introducción adquirido y conseguir información sobre la siguiente etapa que va a realizarse, volviendo a la etapa n.º 4 o, si no existe etapa siguiente, moverse a la etapa n.º 7

7. Visualizar respuesta apropiada al usuario sobre el éxito o el fallo de la transacción, con cadenas e imágenes seleccionadas de manera dinámica en la etapa n.º 6.

De esta manera, la experiencia de compra se localiza automáticamente, se actualiza con las últimas políticas y trabaja con un número variable y tipo de etapas dependiendo del contexto.

Todos los PGS pueden implementarse en la biblioteca de soporte alternativa según la implementación alternativa completamente compatible que hace a la aplicación totalmente funcional fuera de la tienda de Google Play.

La biblioteca de soporte detectará automáticamente país y operador de móvil del teléfono móvil consultando la API de sistema operativo para recuperar código IMSI y descomponer el mismo en código de país de móvil y código de red de móvil según normas de la ITU. Al usar esta información, la biblioteca de soporte relacionará automáticamente el teléfono móvil con un operador de móvil particular y reconfigurará la experiencia de compra según las capacidades técnicas y los requisitos normativos de este operador de móvil. Esto puede incluir, por ejemplo:

- Cambiar diseño gráfico del formato de participación
- Añadir logo y nombre del operador de móvil
- Cambiar lenguaje del texto visualizado
- Visualizar un texto diferente para términos y condiciones legales y privacidad
- Seleccionar un precio diferente indicado para el artículo que está comprándose
- Ofrecer un flujo de compra diferente (un clic, dos clics, etc.)
- Seleccionar un método de identificación diferente (3G, por medio de cabeceras, con triangulación SMS, etc.)

Cuando sea necesario, la identificación de usuario se realizará automáticamente mediante la biblioteca de soporte sin ninguna necesidad de interacción de usuario, usando estas etapas:

• La biblioteca de soporte llama al servidor de software intermedio para abrir una sesión de triangulación y recibe un único código de sesión generado aleatoriamente

• La biblioteca de soporte envía un SMS de MT a un número de teléfono especial seleccionado específicamente dependiendo del operador de móvil, con el código de sesión incluido en el texto de mensaje.

• El software intermedio global que controla el número de teléfono especial, recibe el SMS y almacena el número de teléfono asociado en una tabla junto con el código de sesión recuperado del texto de mensaje.

• La biblioteca de soporte sondea el software intermedio global que pasa el código de sesión, esperando la tabla que va a sondearse y recibiendo de vuelta finalmente el número de teléfono

• Una tarea periódica limpia la tabla eliminando entradas antiguas y liberando códigos de sesión para reutilización.

La biblioteca de soporte puede conectarse en el código de aplicación original por el desarrollador de aplicación y el nuevo APK construido usando la misma serie de herramientas usada para compilaciones de APK normales, por ejemplo dentro de Eclipse. Mediante este método, el desarrollador original puede crear versiones alternativas del APK y usarlas para la distribución fuera de Google Play o para proporcionar a los usuarios métodos de pago en aplicaciones alternativos.

En una realización alternativa, al usar un algoritmo de software único, un procedimiento automatizado puede modificar automáticamente el binario APK original para sustituir la biblioteca de soporte de Google con la biblioteca de soporte alternativa, construyendo de manera eficaz la nueva APK automáticamente sin la intervención de los desarrolladores originales. Este algoritmo requiere las etapas siguientes:

a) Desempaquetar el archivo APK original para extraer componentes internos.

b) Decompilar el archivo DEX binario en archivos de clase independientes.

c) Añadir archivos de clase propios que implementan la biblioteca de soporte.

d) Sustituir cadenas de nombres de paquete para referirse a un archivo de clase añadido.

e) Recompilar archivos de clase en un archivo DEX.

f) Añadir archivos de recursos.

g) Modificar el archivo de manifiesto con artículos adicionales que se refieren a la biblioteca de soporte.

h) Reconstruir el archivo APK, que incluye la generación de una nueva firma.

Un procedimiento de construcción automatizado similar puede usarse también para conectar automáticamente el archivo de manifiesto con un código de identificación que puede usarse para etiquetar diferentes circunstancias del archivo APK y rastrear las mismas a canales de distribución diferentes.

La biblioteca de soporte puede proporcionar también un método para rastrear las instalaciones de campañas promocionales diferentes y para referirse a cada compra en aplicación para el patrocinador publicitario específico que generó la instalación de aplicación. Cuando la instalación del APK se promociona, un ID de campaña específico se asocia con la fuente de tráfico y el patrocinador publicitario. Cuando un usuario entra en la campaña promocional, buscando una URL específica, un conjunto de parámetros que caracterizan el teléfono móvil se recogen y combinan en un valor de huella digital que se almacena en una tabla temporal, gestionada por el software intermedio global, junto con el ID de campaña.

Este método de huella digital no se refiere a *cookies*, almacenamiento local ni otros métodos de etiquetado que solo funcionan dentro del buscador y, por tanto, una vez que la aplicación se descarga, instala y abre, el método puede obtenerse de nuevo obteniendo la misma huella digital. La primera vez que la aplicación se abre la huella digital se genera de nuevo y se usa para recuperar el ID de campaña de la tabla temporal. Este ID de campaña se almacena entonces dentro de la aplicación instalada y se usa para rastrear de vuelta al software intermedio global cada suceso de interés, incluyendo la instalación completada de la aplicación, el inicio de una sesión de compra y cada etapa de la misma sesión. Una tabla de sucesos se mantiene en el software intermedio global y se usa para rastrear el historial de cada usuario y notificar a los patrocinadores publicitarios de los sucesos que les pertenecen.

La biblioteca de soporte puede proporcionar también un receptor para notificaciones no solicitadas entrantes generadas por el software intermedio. Este receptor sondea periódicamente el software intermedio, incluso cuando la aplicación no está abierta, y recupera una lista de nuevas notificaciones. Una vez que una nueva notificación llega, esta notificación se visualiza inmediatamente al usuario a través del sistema de notificación de sistema

operativo. Este canal mensajería puede usarse para alertar al usuario y propiciar una interacción adicional, para proporcionar información útil, para mandar mensajes de aviso, para señalar que una nueva versión del APK está disponible.

El software intermedio mantendrá una tabla de aplicaciones ya instaladas, junto con la versión instalada actualmente. Cuando un nuevo APK para esa aplicación se carga en el software intermedio, el software intermedio creará automáticamente notificaciones para cada teléfono móvil que tiene una versión anterior de la aplicación instalada. La biblioteca de soporte puede interceptar automáticamente notificaciones relacionadas con versiones de APK nuevas y iniciar una descarga del nuevo binario APK para actualizar automáticamente la aplicación.

REIVINDICACIONES

1. Método para implementar conjuntos de herramientas de desarrollo de software en aplicación para aplicaciones de Android, que comprende:

- implementar lógicas de aplicación en un lenguaje de programación compatible con un sistema operativo de máquina virtual;

- proporcionar una o más bibliotecas de compatibilidad propios proporcionadas por un primer proveedor de servicios;

- usar un entorno de desarrollo de software para vincular las lógicas de aplicación y las bibliotecas de compatibilidad propios entre sí, y empaquetar las mismas en a primer binario APK;

- cargar el primer binario APK a una tienda de aplicaciones asociada al primer proveedor de servicios;

y caracterizado porque comprende además al menos las etapas siguientes:

- que incluyen bibliotecas de compatibilidad propios adicionales proporcionadas por segundos proveedores de servicios que implementan la integración con canales de distribución alternativos en lugar de las proporcionadas por el primer proveedor de servicios;

- usar dicho entorno de desarrollo de software para vincular las lógicas de aplicación y las bibliotecas de compatibilidad propios adicionales entre sí en binarios APK adicionales que pueden cargarse a tiendas de aplicaciones alternativas;

en el que dichos binarios APK adicionales se obtienen aplicando al menos las etapas siguientes:

a) desempaquetar el primer binario APK para extraer componentes internos;

b) decompilar archivo DEX binario en archivos de clase independientes;

c) añadir los archivos de clase que implementan las bibliotecas de compatibilidad propios adicionales;

d) sustituir cadenas de nombres de paquete para referirse a archivos de clase añadidos;

e) recompilar archivos de clase en un archivo DEX;

f) añadir archivos de recursos;

g) modificar el archivo de manifiesto con artículos adicionales que se refieren a las bibliotecas de compatibilidad propios adicionales;

h) reconstruir el primer archivo APK como binarios APK adicionales, que incluyen la generación de nuevas firmas.

2. Método según la reivindicación 1, en el que las bibliotecas de compatibilidad propios adicionales comprenden uno o más de los métodos de función siguientes:

- método de función para determinar si se soporta un pago dependiendo del país, el operador de móvil y/o el tipo de compra;

- método de función para volver a una lista de identificadores de productos de artículos ya comprados y sus detalles, y/o para saber si ya se ha comprado un artículo;

- método de función para volver a una lista de todos los artículos que pueden comprarse y sus detalles, que incluyen precio, título, descripción y tipo de compra;

- método de función para iniciar un flujo de compra y permitir la interacción de usuario dentro de la aplicación;

- método de función para marcar una compra como consumida y/o permitir compras adicionales del mismo artículo.

3. Sistema para implementar conjuntos de herramientas de desarrollo de software en aplicación que comprende software, hardware y/o medios de software intermedio configurados para llevar a cabo un

método según la reivindicación 1.

4. Sistema según la reivindicación 3, que comprende un software intermedio de servidor implementado en una agrupación de servidores de aplicaciones, configurado para mandar instrucciones a uno o más de los componentes siguientes:

- base de datos de artículos de inventario para una aplicación en una tienda;

- base de datos de artículos comprados para cada aplicación y usuario en una tienda;

- medios de definición de flujo de compra para un operador de móvil y método de pago asociado;

- máquina de estados finitos que implementa el flujo de compra durante una sesión de compra;

- base de datos de sesiones de compra en curso y sus estados;

- conectores y adaptadores de protocolo para un operador de móvil y método de pago asociado;

- API de JSON usada por SDK;

- registro de notificación para estadísticas;

- traza de auditoría.

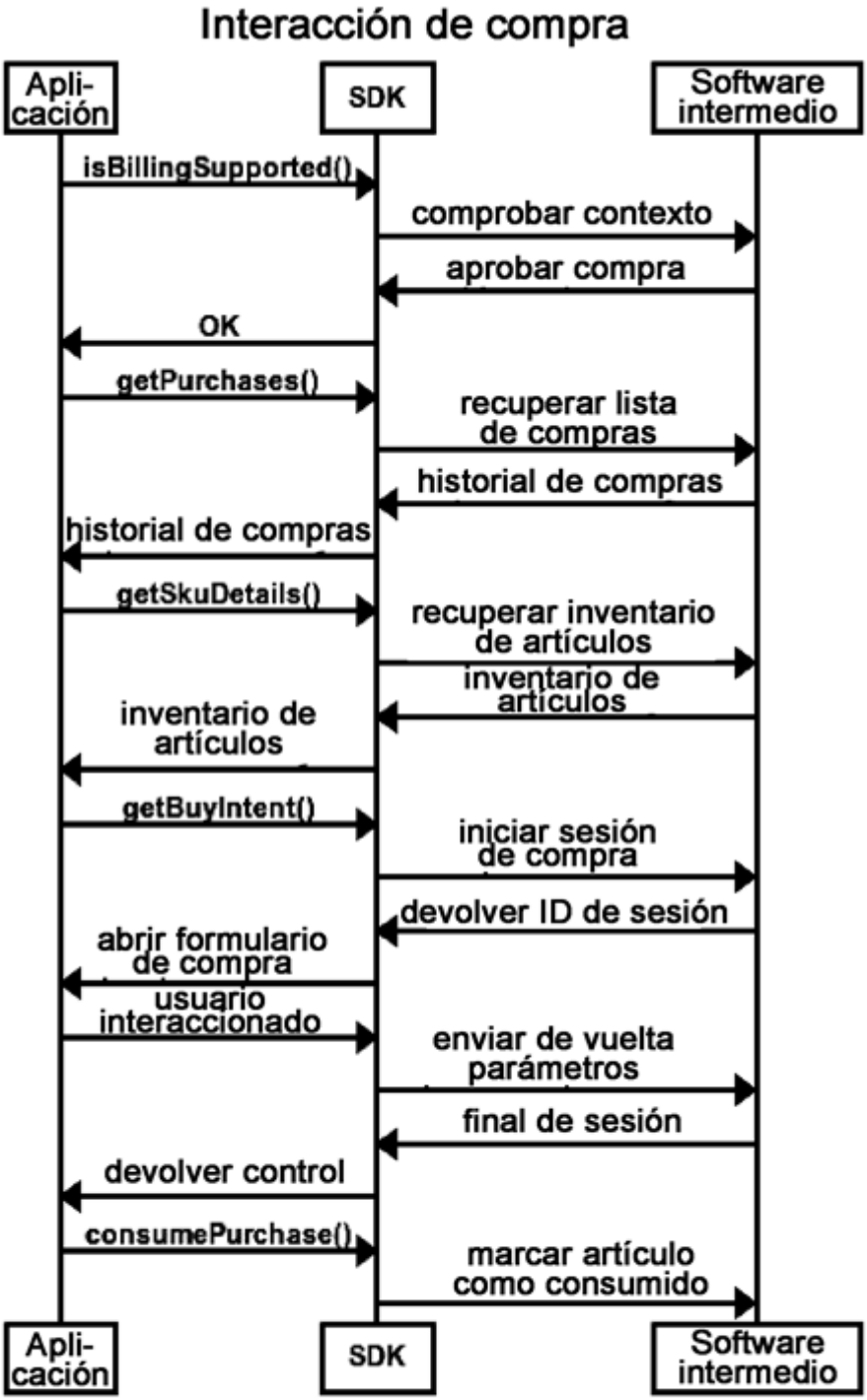
Interacción de compra
Apli-cación
SDK
Software intermedio
isBillingSupported()
comprobar contexto
aprobar compra
OK
getPurchases()
recuperar lista de compras
historial de compras
historial de compras
getSkuDetails()
recuperar inventario de artículos
inventario de artículos
inventario de artículos
getBuyIntent()
iniciar sesión de compra
devolver ID de sesión
abrir formulario de compra
usuario interaccionado
enviar de vuelta parámetros
final de sesión
devolver control
consumePurchase()
marcar artículo como consumido
Apli-cación
SDK
Software intermedio

FIG. 1

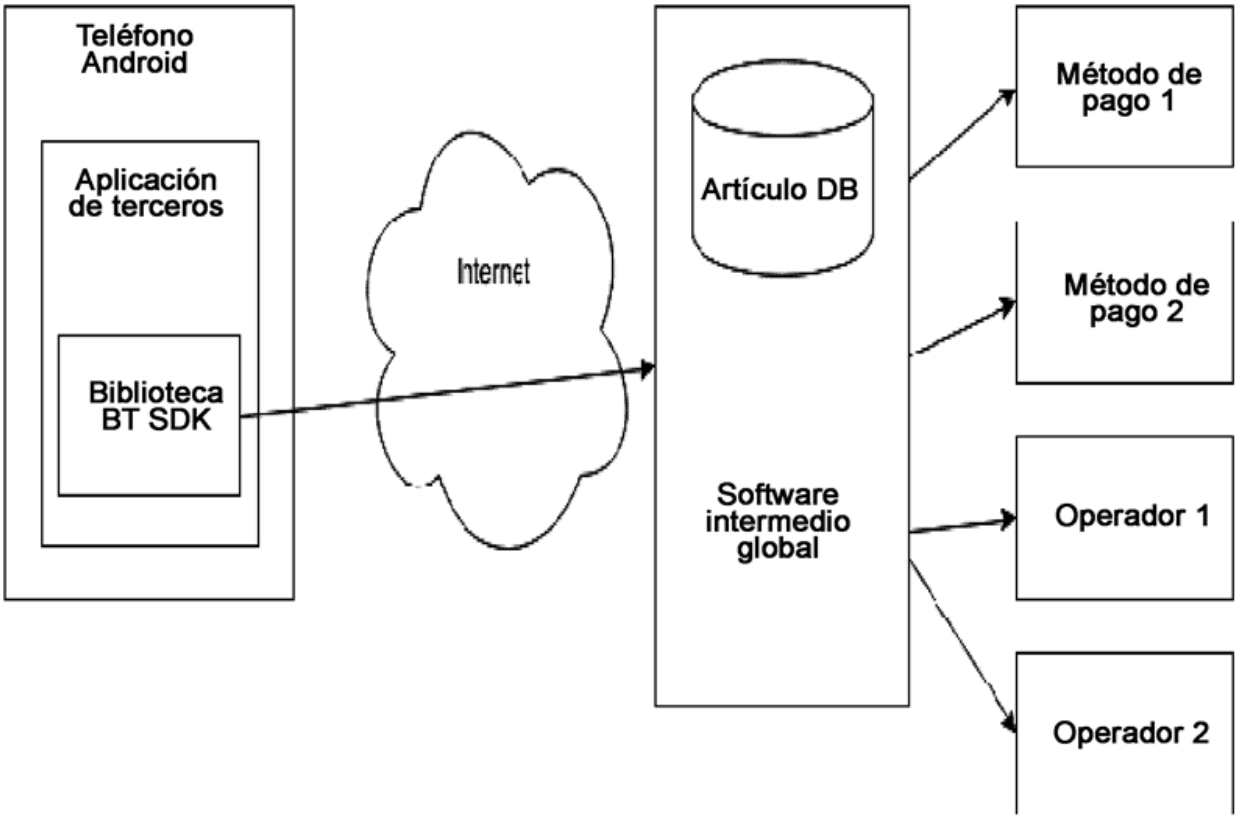
Teléfono Android
Aplicación de terceros
Biblioteca BT SDK
Internet
Artículo DB
Software intermedio global
Método de pago 1
Método de pago 2
Operador 1
Operador 2

FIG. 2

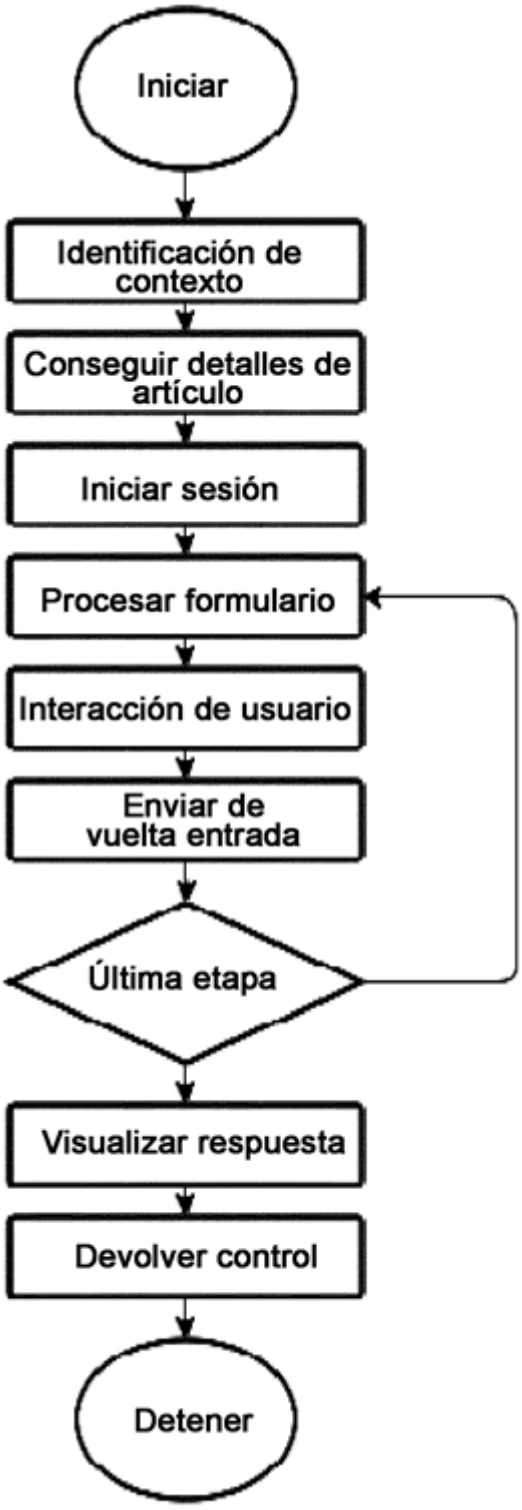
Iniciar
Identificación de contexto
Conseguir detalles de artículo
Iniciar sesión
Procesar formulario
Interacción de usuario
Enviar de vuelta entrada
Última etapa
Visualizar respuesta
Devolver control
Detener

FIG. 3