(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 736 252**

(51) Int. CI.:

G06F 9/44 (2008.01)
G06F 9/22 (2006.01)
G06F 9/48 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **10.10.2011 PCT/US2011/055630**

(87) Fecha y número de publicación internacional: **07.03.2013 WO13032504**

(96) Fecha de presentación y número de la solicitud europea: **10.10.2011 E 11871515 (0)**

(97) Fecha y número de publicación de la concesión europea: **08.05.2019 EP 2751674**

(54) Título: **Agregación de eventos para una ejecución de trabajo en segundo plano**

(30) Prioridad:

01.09.2011 US 201113224154

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
27.12.2019

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

SCHWARTZ, JAMES A., JR.;
KISHAN, ARUN U.;
NEVES, RICHARD K.;
PROBERT, DAVID B.;
PULAPAKA, HARI y
GEFFLAUT, ALAIN F.

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Agregación de eventos para una ejecución de trabajo en segundo plano

Antecedentes

Los sistemas operativos convencionales implementan algunos modelos de ejecución diferentes para gestionar el trabajo en primer plano y en segundo plano de las aplicaciones. Ejemplos de tareas de procesamiento que se puede considerar que son trabajo de primer plano incluyen, sin limitación, la presentación de una interfaz gráfica de usuario, la respuesta a una indicación del usuario, y cualquier otro tipo de trabajo relacionado con la interacción con el usuario. Ejemplos de tareas de procesamiento que se puede considerar que son trabajo de segundo plano incluyen, sin limitación, la descarga y la instalación de actualizaciones del soporte lógico, la sincronización con un servidor y cualquier otro tipo de trabajo que pueda no implicar la atención del usuario.

En un primer modelo, se permite que se ejecuten de forma concurrente un número cualquiera de aplicaciones, tanto en primer plano como en segundo plano, y que compitan por recursos tales como ciclos del procesador y memoria. Este modelo es implementado comúnmente en ordenadores de sobremesa y portátiles.

En un segundo modelo, se bloquea la ejecución de todas las aplicaciones, salvo una aplicación “activa”; concretamente, una aplicación con la cual el usuario está interactuando de forma activa. Este modelo es implementado comúnmente en dispositivos móviles, en los que, en cualquier momento dado, un usuario normalmente interactúa con una sola aplicación que es mostrada a pantalla completa.

En un tercer modelo, se trata el trabajo de primer plano y el trabajo de segundo plano para la misma aplicación como mutuamente excluyente, de modo que no se planifica que una aplicación que se ejecuta en primer plano haga ningún trabajo de segundo plano a la vez, y viceversa.

En un cuarto modelo, una aplicación puede ser instanciada por separado para trabajo de primer plano y para trabajo de segundo plano, ejecutándose las dos instancias concurrentemente y compitiendo por recursos.

El documento US5694603 da a conocer un sistema informático de múltiples subprocesos para la ejecución concurrente asíncrona preferencial de varios subprocesos de instrucciones de un programa de aplicación de múltiples subprocesos en el que una rutina de servicio de interrupción activa reiteradamente el subproceso editor para proporcionar un proceso de edición asíncrona preferencial ya sea para la introducción de nuevo código fuente o para la modificación del código introducido previamente, mientras el subproceso compilador procesa concurrentemente el código fuente durante los intervalos de tiempo entre pulsaciones de tecla.

Sumario

Lo anterior es un sumario no limitante de la invención, que está definida por las reivindicaciones adjuntas.

Los inventores han reconocido y apreciado varias desventajas que surgen de los modelos de ejecución de los sistemas operativos existentes, y han desarrollado nuevos modelos de ejecución que proporcionan ventajas tales como mayor vida de la batería y mejor experiencia de usuario.

En algunas realizaciones, un sistema operativo puede cargar por separado trabajo de primer plano y trabajo de segundo plano de una misma aplicación o de aplicaciones diferentes, y puede tratar el trabajo de primer plano y el trabajo de segundo plano de manera diferente cuando se adoptan decisiones de planificación. Por ejemplo, los componentes de la aplicación pueden ser designados como componentes de primer plano o como componentes de segundo plano, para que un sistema operativo pueda aplicar directrices diferentes a los componentes en función de su designación. Las directrices pueden estar diseñadas para promover un uso eficaz de los recursos a la vez que para brindar al usuario una abundante experiencia multitarea.

En algunas realizaciones adicionales, un sistema operativo puede determinar cuándo ha de realizarse trabajo de segundo plano en función de información facilitada por una aplicación que especifica ciertas circunstancias en las que se pretende la realización del trabajo de segundo plano. Sin embargo, el sistema operativo puede anular la especificación de una aplicación en un esfuerzo por evitar que el trabajo de segundo plano consuma demasiada energía y/o tenga un impacto en la capacidad de respuesta de las aplicaciones activas.

En una realización, se proporciona un procedimiento para ser usado por un sistema operativo que se ejecuta en al menos un ordenador. El procedimiento comprende las etapas de: identificar al menos un componente de segundo plano de una aplicación; determinar si se satisface al menos una condición; y ejecutar el al menos un componente de segundo plano únicamente cuando se determina que se satisface la al menos una condición.

En un ejemplo adicional, se proporciona al menos un soporte legible por ordenador que tiene codificado en el mismo, al menos: al menos un primer elemento de código ejecutable por ordenador que implementa al menos un componente de primer plano de una aplicación; una especificación que identifica al menos un segundo elemento de código ejecutable por ordenador para realizar trabajo de segundo plano para la aplicación; y al menos un segundo elemento

de código ejecutable para especificar, a un sistema operativo, al menos una condición que ha de ser satisfecha antes de iniciar el trabajo de segundo plano.

En otro ejemplo adicional, se proporciona un procedimiento implementado por ordenador que comprende las etapas de: recibir, en un sistema operativo, una solicitud de que se ejecute al menos un componente de segundo plano de una aplicación en respuesta a al menos un evento; en respuesta a la detección de una incidencia del al menos un evento, determinar, por el sistema operativo, si se satisface al menos una primera condición establecida por la aplicación y si se satisface al menos una segunda condición establecida por el sistema operativo; y ejecutar el al menos un componente de segundo plano cuando se determina que las al menos una primera condición y una segunda condición tras la incidencia del al menos un evento.

Debería apreciarse que se contempla que todas las combinaciones de los conceptos precedentes y conceptos adicionales expuestas con mayor detalle posteriormente (siempre y cuando tales conceptos no sean mutuamente incoherentes) formen parte del contenido inventivo dado a conocer en la presente memoria. En particular, se contempla que todas las combinaciones del contenido reivindicado que aparece al final de esta divulgación formen parte de del contenido inventivo dado a conocer en la presente memoria.

Breve descripción de los dibujos

Los dibujos adjuntos no están necesariamente dibujados a escala.

La FIG. 1 muestra una aplicación ilustrativa 100 que tiene lógica desacoplada para trabajo de primer plano y trabajo de segundo plano, según algunas realizaciones.
La FIG. 2 muestra un diagrama ilustrativo de estados para una aplicación que tiene un componente de primer plano y un componente de segundo plano que pueden ser ejecutados concurrentemente, según algunas realizaciones.
La FIG. 3A muestra un ejemplo en el cual un sistema operativo carga un componente de primer plano y un componente de segundo plano en procesos separados, según algunas realizaciones.
La FIG. 3B muestra un ejemplo en el cual un sistema operativo carga un componente de primer plano y un componente de segundo plano en el mismo proceso, según algunas realizaciones.
La FIG. 4 muestra un ejemplo de un sistema operativo adaptado para gestionar el trabajo de segundo plano y el trabajo de primer plano por separado, según algunas realizaciones.
La FIG. 5 es un diagrama de flujo de un proceso ilustrativo que puede ser realizado por un componente agente en respuesta a una solicitud de disponer un evento negociado, según algunas realizaciones.
La FIG. 6 muestra una estructura ilustrativa de datos que puede ser usada por un componente agente para almacenar información relativa a eventos negociados, según algunas realizaciones.
La FIG. 7 es un diagrama de flujo de un proceso ilustrativo que puede realizar un componente agente para determinar cuándo señalizar un evento negociado según algunas realizaciones.
La FIG. 8A es un diagrama de flujo de un proceso ilustrativo que puede realizar una infraestructura de agentes para asociar un componente de segundo plano con un evento negociado, según algunas realizaciones.
La FIG. 8B muestra una estructura ilustrativa de datos que puede ser usada para almacenar eventos negociados en asociación con componentes de segundo plano, según algunas realizaciones.
La FIG. 9 muestra un proceso ilustrativo 900 que puede realizar una infraestructura de agentes para determinar cuándo ejecutar un componente de segundo plano y cómo gestionar la ejecución del componente de segundo plano, según algunas realizaciones.
La FIG. 10 muestra, esquemáticamente, un ordenador ilustrativo en el que pueden implementarse diversos aspectos de la presente divulgación.

Descripción detallada

Los inventores han reconocido y apreciado varias desventajas que surgen de los modelos existentes de ejecución de sistemas operativos, y han desarrollado nuevos modelos de ejecución que proporcionan ventajas tales como una mayor duración de la batería y una mejor experiencia de usuario.

Por ejemplo, el primer modelo de ejecución que permite que un número cualquiera de aplicaciones se ejecute de forma concurrente puede llevar a demasiados procesos que compiten por recursos tales como ciclos de procesador y memoria. A su vez, esto puede conducir a un rendimiento lento en un entorno con recursos limitados, como en un dispositivo móvil con velocidad de procesamiento y memoria limitadas. Si el dispositivo móvil funciona con energía de batería, la ejecución indiscriminada de procesos también puede reducir la vida útil de la batería.

Por otro lado, el segundo modelo de ejecución de bloqueo de todas las aplicaciones menos la activa puede impedir escenarios deseables de multitarea, como que un usuario componga un correo electrónico mientras escucha música, recibe actualizaciones de ubicación del sistema de posicionamiento global (GPS), y/o recibir llamadas entrantes de voz sobre protocolo de Internet (VoIP). Estos escenarios de multitarea requieren que una aplicación pueda ejecutarse hasta cierto punto incluso cuando el usuario está interactuando activamente con otra aplicación.

El tercer modelo de ejecución, concretamente, ejecutar el trabajo de primer plano y trabajo de segundo plano de la misma aplicación de manera mutuamente excluyente, puede impedir de manera similar los escenarios deseables de tareas múltiples, como que un usuario lea una página electrónica a través de un navegador de Internet, mientras aguarda que se complete una descarga de Internet.

Aunque el cuarto modelo de ejecución de instanciación separada puede permitir que un sistema operativo ejecute simultáneamente el trabajo de segundo plano y el trabajo de primer plano para la misma aplicación, siguen existiendo varios inconvenientes potenciales. Por ejemplo, puede haber demasiada dependencia de los desarrolladores de aplicaciones para escribir código que se comporte de una manera coherente con la distinción de primer plano-segundo plano. Aunque una aplicación exponga un parámetro de llamada para permitir que un sistema operativo indique si la aplicación está instanciada para trabajo de primer plano o trabajo de segundo plano, un desarrollador puede omitir sin querer la lógica del programa que impida que el trabajo de segundo plano se ejecute cuando la aplicación está instanciada para el trabajo de primer plano, o viceversa. Esto puede socavar la capacidad del sistema operativo para controlar efectivamente el consumo de recursos de control en el trabajo de segundo plano, porque el sistema operativo puede tener dificultades para determinar si una supuesta instancia de primer plano está realizando solo el trabajo de primer plano (por ejemplo, responder a una entrada recibida a través de una interfaz de usuario) y no realizar ningún trabajo de segundo plano innecesario o ineficiente (por ejemplo, sincronizar periódicamente con un servidor).

El planteamiento de la instanciación separada también puede imponer una carga sobre los desarrolladores de aplicaciones para gestionar las complejidades relacionadas con la ejecución concurrente. Por ejemplo, cuando una instancia de primer plano y una instancia de segundo plano de la misma aplicación se ejecutan simultáneamente, las dos instancias pueden ejecutarse en diferentes contextos y pueden no compartir el estado. En consecuencia, puede que las dos instancias se comuniquen solo a través de mecanismos proporcionados explícitamente por el sistema operativo, lo que puede complicar el código del programa. Los inventores han reconocido y apreciado que, si bien tal desacoplamiento entre trabajo de segundo plano y trabajo de primer plano puede ser deseable, también puede ser conveniente dar a los desarrolladores una opción para no admitir la ejecución concurrente de múltiples instancias en contextos diferentes.

Por consiguiente, en algunas realizaciones, se proporcionan modelos de ejecución más flexibles para permitir que un sistema operativo gestione el trabajo de segundo plano de manera diferente al trabajo de primer plano.

En un aspecto, la lógica de la aplicación para el trabajo de primer plano y el trabajo de segundo plano puede estar desacoplada de tal modo que permita a un sistema operativo cargar separadamente el trabajo de primer plano y el trabajo de segundo plano, y tratar el trabajo de primer plano y el trabajo de segundo plano de forma diferente cuando se adoptan decisiones de planificación.

Por ejemplo, en algunas realizaciones, una aplicación puede incluir componentes separados, algunos designados como componentes de primer plano y otros designados como componentes de segundo plano. Un sistema operativo puede aplicar diferentes directrices a los componentes en función de sus designaciones, pudiendo estar diseñadas las directrices para promover el uso eficiente de los recursos y, al mismo tiempo, brindar al usuario una abundante experiencia multitarea.

Por ejemplo, pueden aplicarse una o más directrices de gestión de recursos a los componentes de segundo plano. Estas directrices pueden diseñarse para limitar cuándo se ejecuta un componente de segundo plano en función de los recursos que ha de usar ese componente de segundo plano. La imposición de tales directrices puede contribuir a conservar la energía y/o reservar recursos suficientes para una aplicación activa (es decir, una aplicación con la que un usuario está interactuando activamente). Dichas directrices pueden imponerse, aunque no es preciso que lo hagan, mediante la ejecución de componentes de segundo plano en un entorno de ejecución controlada, como un “enclave de seguridad” o *sandbox*, como se conoce en la técnica, aislando con ello los componentes de segundo plano de los componentes de primer plano.

En otro aspecto, un desarrollador de aplicaciones puede tener una opción para no admitir la ejecución concurrente de componentes de primer plano y de segundo plano de la misma aplicación en diferentes procesos. La selección de tal opción puede dar como resultado un código de programa más simple y eficiente.

En algunas realizaciones, los componentes de primer plano y de segundo plano pueden proporcionarse en trozos separados de código ejecutable que pueden ejecutarse independientemente entre sí. En algunas realizaciones adicionales, los componentes de primer plano y de segundo plano pueden tener distintos puntos de entrada, de modo que un sistema operativo pueda cargar los componentes de primer plano y de segundo plano en el mismo proceso pero en diferentes subprocesos, lo que permite que los componentes de primer plano y de segundo plano se ejecuten junto con el estado compartido. De esta manera, el sistema operativo sigue siendo capaz de distinguir los subprocesos de ejecución asociados con el trabajo de segundo plano de los subprocesos de ejecución asociados con el trabajo de primer plano, sin requerir que la aplicación gestione diversas complejidades de ejecución concurrente en diferentes procesos.

Si una aplicación soporta la ejecución concurrente, un sistema operativo puede ser capaz de elegir cargar los componentes de primer plano y de segundo plano ya sea en el mismo proceso o en procesos separados. Se pueden

tener en cuenta varios factores al elegir entre estas opciones. Por ejemplo, según se ha mencionado anteriormente, separar el trabajo de segundo plano del trabajo de primer plano puede permitir que un sistema operativo ejecute el trabajo de segundo plano en un entorno designado sujeto a ciertas restricciones de recursos (por ejemplo, un enclave de seguridad construido para alojar componentes de segundo plano). Además, separar el trabajo de segundo plano del trabajo de primer plano puede reducir la ocupación de memoria, ya que no es preciso cargar el código ejecutable (por ejemplo, bibliotecas) relacionado únicamente con las funcionalidades de la interfaz de usuario (UI) cuando la aplicación solo realiza trabajo de segundo plano. Por otro lado, si una aplicación realiza con frecuencia trabajo de segundo plano, o si uno o más usuarios interactúan frecuentemente con la aplicación, puede ser ventajosa la colocalización del trabajo de primer plano y el trabajo de segundo plano en el mismo proceso para compartir los costes de puesta en marcha del proceso entre los componentes de primer plano y de segundo plano. Sin embargo, el sistema operativo también puede reducir los costes de puesta en marcha del proceso ejecutando el trabajo de segundo plano en un entorno separado (por ejemplo, un contenedor de soporte lógico) y almacenando en memoria intermedia ese entorno.

En otro aspecto adicional, un sistema operativo puede determinar cuándo se realiza el trabajo de segundo plano en función de información proporcionada por una aplicación que especifica ciertas circunstancias en las cuales se pretende la ejecución del trabajo de segundo plano. Por ejemplo, en algunas realizaciones, el sistema operativo puede planificar el trabajo de segundo plano para su ejecución solo en respuesta a uno o más eventos especificados (también denominados “activadores”). Ejemplos de activadores incluyen, sin limitación, eventos relacionados con actividades de la red (por ejemplo, un paquete que llega de una red), eventos relacionados con actividades del usuario (por ejemplo, que un usuario se conecte o se desconecte), eventos planificados (por ejemplo, que caduque un temporizador), eventos relacionados con dispositivos periféricos (por ejemplo, una notificación de impresora) y similares.

En algunas realizaciones adicionales, un sistema operativo puede anular la especificación de una aplicación en cuanto a cuándo ha de ejecutarse el trabajo de segundo plano. Por ejemplo, el sistema operativo puede actuar como un “agente” que filtra los eventos previstos para desencadenar el trabajo del segundo plano. En tal modelo de “ejecución negociada”, una aplicación puede dar de alta en el sistema operativo uno o más activadores para hacer que un componente de segundo plano se ejecute; por ejemplo, para realizar una determinada unidad de trabajo de segundo plano (también denominada “tarea de segundo plano” o “elemento de trabajo”). En lugar de permitir que un activador de este tipo provoque directamente la ejecución del componente de segundo plano, el sistema operativo puede introducir una capa adicional de toma de decisiones. Por ejemplo, cuando se desencadena un activador especificado por la aplicación, el sistema operativo puede decidir si enviar la señal de lo que se denomina “evento negociado”, lo que, a su vez puede hacer que se ejecute el componente de segundo plano. De esta manera, el sistema operativo puede decidir en último término cuándo se ejecuta el componente de segundo plano. Por ejemplo, un cliente de correo electrónico puede dar de alta una tarea de sincronización en segundo plano para su ejecución periódica (por ejemplo, cada 15 minutos), y el sistema operativo puede configurar un temporizador apropiado y, cuando el temporizador caduque, decidir si ejecutar la tarea en función de las condiciones operativas que existan en ese momento. Alternativamente, o adicionalmente, el sistema operativo puede implementar una directriz que limite la cantidad de activadores de cierto tipo que una aplicación puede dar de alta en el sistema operativo. En un ejemplo, el sistema operativo puede limitar el número de veces que una aplicación puede dar de alta un evento de alarma como activador, pudiendo ser el límite cualquier valor debidamente definido. Una directriz de este tipo puede ser beneficiosa, porque un evento de alarma puede “despertar” a un ordenador partiendo de un modo de “reposo”, de modo que la limitación de los eventos de alarma pueda mejorar la vida útil de la batería.

Como se ha expuesto anteriormente, un sistema operativo puede imponer una o más directrices de gestión de recursos diseñadas para controlar el consumo de recursos por el trabajo de segundo plano. En algunas realizaciones, se puede aplicar una directriz de gestión de recursos usando un modelo de ejecución negociada controlando inteligentemente la señalización de eventos negociados, evitando con ello que el trabajo asociado de segundo plano consuma demasiada energía y/o que afecte a la capacidad de respuesta de las aplicaciones activas. En un ejemplo, cuando se determina que la energía restante de la batería está por debajo de un umbral seleccionado, el sistema operativo puede posponer la señalización de uno o más eventos negociados y, por lo tanto, posponer cierto trabajo de segundo plano no crítico. En otro ejemplo, cuando se detecta una latencia alta en la red, el sistema operativo puede disminuir la frecuencia a la que se señalizan uno o más eventos negociados y, con ello, disminuir la frecuencia con la que una aplicación se sincroniza con un servidor. En otro ejemplo adicional, el sistema operativo puede posponer la señalización de uno o más eventos negociados porque una aplicación asociada ha agotado una cuota de procesamiento de segundo plano aplicable, que puede ser especificada, aunque no es preciso que así sea, en términos de una cierta cantidad de procesamiento en segundo plano permitida durante cierto periodo de tiempo. En otro ejemplo adicional, el sistema operativo puede posponer la señalización de uno o más eventos negociados porque ciertas actividades de servicio pueden estar en marcha, por lo que los ficheros ejecutables relevantes pueden no estar disponibles temporalmente. En otro ejemplo adicional, múltiples elementos del trabajo de segundo plano pueden solicitar un recurso común (por ejemplo, una conexión de red, una unidad de procesamiento de gráficos, una cámara web, etc.) que puede consumir una cantidad significativa de energía cuando se enciende y se apaga. En consecuencia, el sistema operativo puede controlar la señalización de uno o más eventos negociados para consolidar la ejecución de estos elementos de trabajo y reducir el número de veces que se activa y desactiva el recurso común, con lo que se conserva energía.

Los inventores han reconocido y apreciado, además, que algunos tipos de trabajo de segundo plano pueden ser innecesarios y/o ineficientes. Por ejemplo, un cliente de correo electrónico que se despierta según lo planificado para sincronizarse con un servidor de correo electrónico puede descubrir que no hay conexión de red, y simplemente puede volver a un estado suspendido para ser despertado nuevamente en el futuro. En esta situación, se puede gastar energía solo para reactivar brevemente al cliente de correo electrónico para verificar la disponibilidad de la red, pero no se puede realizar ningún trabajo útil. Por lo tanto, puede ser conveniente posponer la activación del cliente de correo electrónico hasta que ciertas condiciones del sistema sean tales que sea probable que se pueda realizar un trabajo útil. Ejemplos de tales condiciones del sistema incluyen, sin limitación, la disponibilidad de un recurso como una conexión a Internet y que esté encendido un componente de soporte físico (por ejemplo, pantalla, altavoz, etc.).

Por consiguiente, en algunas realizaciones, un sistema operativo puede asociar un componente de segundo plano con una o más condiciones, además de con uno o más activadores. Cuando se desencadena un activador, el sistema operativo puede verificar las condiciones asociadas y puede planificar la ejecución del componente de segundo plano solo si se satisfacen todas las condiciones asociadas. Así, al especificar una o más condiciones para un componente de segundo plano, un desarrollador de aplicaciones puede identificar en el sistema operativo un conjunto de circunstancias en las que es probable que el componente de segundo plano se ejecute de manera productiva y el sistema operativo puede planificar en consecuencia el componente de segundo para su ejecución.

En algunas realizaciones adicionales, un sistema operativo puede configurarse para retener un activador si el sistema operativo determina que una o más condiciones asociadas no se satisfacen en el momento en que se desencadena el activador. De esta manera, el sistema operativo puede seguir verificando las condiciones asociadas, y puede ejecutar el componente de segundo plano si y cuando se satisfagan todas las condiciones asociadas. Alternativamente, o adicionalmente, el sistema operativo puede simplemente descartar un activador desencadenado si no se satisfacen una o más condiciones asociadas en el momento en que se desencadena el activador.

En otras realizaciones adicionales, el sistema operativo puede configurarse para que mantenga información de estado que puede usarse para evaluar una o más condiciones asociadas con un componente de segundo plano. El sistema operativo también puede configurarse para que detecte cambios en la información de estado mantenida y, en respuesta a la detección de un cambio, para que evalúe una condición asociada. Esto puede permitir que el sistema operativo detecte cuándo ha de ejecutarse un componente de segundo plano después de que se haya retenido un activador para el componente de segundo plano (por ejemplo, porque no se satisficieron todas las condiciones para el componente de segundo plano en el momento en que se desencadenó el activador). Alternativamente, o adicionalmente, el sistema operativo puede obtener información relevante necesaria para evaluar una condición, sin almacenar dicha información como información de estado.

A continuación, se incluyen descripciones más detalladas de diversos conceptos relacionados con sistemas, procedimientos y aparatos inventivos, y realizaciones de los mismos, para la gestión del trabajo de segundo plano y del trabajo de primer plano. Debe apreciarse que los diversos conceptos introducidos anteriormente y expuestos con mayor detalle a continuación pueden ser implementados en cualquiera de varias maneras, ya que los conceptos divulgados no están limitados a ninguna forma particular de implementación. Por ejemplo, la presente divulgación no se limita a las disposiciones particulares de los componentes mostrados en las diversas figuras, ya que otras disposiciones también pueden ser adecuadas. Tales ejemplos de implementaciones y aplicaciones específicas se proporcionan únicamente con fines ilustrativos. Además, aunque diversos conceptos descritos en este documento pueden ser útiles para la conservación de recursos en un entorno informático con recursos limitados, los conceptos no están limitados a ser empleados en tales entornos. Por ejemplo, puede emplearse cualquier combinación de técnicas divulgadas no solo en dispositivos móviles, sino también en ordenadores de sobremesa, ordenadores servidores y similares.

La FIG. 1 muestra una arquitectura de aplicación con lógica desacoplada para el trabajo de primer plano y el trabajo de segundo plano, según algunas realizaciones. En tal arquitectura, el trabajo de primer plano y el trabajo de segundo plano pueden ser proporcionados como componentes separados de la aplicación con interfaces adaptadas para permitir que un sistema operativo gestione el trabajo de primer plano y el trabajo de segundo plano por separado. Posteriormente, se describe, en conexión con la FIG. 4, un ejemplo de sistema operativo adaptado a gestionar por separado el trabajo de primer plano y el trabajo de segundo plano.

En el ejemplo mostrado en la FIG. 1, la aplicación 100 incluye varios componentes separados, como uno o más componentes de primer plano (por ejemplo, el componente 105 de primer plano), uno o más componentes de segundo plano (por ejemplo, los elementos de trabajo 110A, 110B, 110C, …), y uno o más componentes (por ejemplo, los objetos compartidos 115A, 115B, …). Estos componentes pueden implementarse de tal modo que estén desacoplados lógicamente entre sí. Por ejemplo, en algunas realizaciones, los componentes pueden ser proporcionados en partes separadas de código ejecutable que pueden ejecutarse independientemente entre sí. Estos trozos separados de código ejecutable pueden tener puntos de entrada distintos para ser utilizados por un sistema operativo para cargar por separado los correspondientes componentes.

Por ejemplo, en algunas realizaciones, un sistema operativo puede poner en marcha la aplicación 100 en primer plano en respuesta a una acción del usuario. El lanzamiento de la aplicación 100 en primer plano puede incluir el inicio del

componente 105 de primer plano a través de un correspondiente punto de entrada. Por otro lado, el sistema operativo puede poner en marcha la aplicación 100 en segundo plano en respuesta a la detección de una incidencia de uno o más eventos (por ejemplo, uno o más activadores, como se describió anteriormente). La puesta en marcha de la aplicación 100 en segundo plano puede incluir el inicio de un componente de segundo plano, como el elemento de trabajo 110A, a través de un punto de entrada correspondiente que es diferente del punto de entrada para el componente 105 de primer plano. Por ejemplo, en una realización, el elemento de trabajo 110A puede implementarse como una biblioteca de enlace dinámico (DLL), y el sistema operativo puede ejecutar el elemento de trabajo 110A en segundo plano al cargar la DLL y ejecutar una función de punto de entrada de la DLL.

Ejemplos de acciones del usuario que pueden hacer que una aplicación se ponga en marcha en primer plano incluyen, sin limitación, que un usuario haga clic en un mosaico o icono correspondiente a la aplicación, que el usuario inicie sesión, que el usuario intente acceder a un fichero u objeto para el cual la aplicación está identificada como aplicación por defecto, y similares. Ejemplos de eventos que se pueden usar como activadores para hacer que una aplicación se inicie en segundo plano incluyen, sin limitación, un temporizador que caduca, un cambio en la conectividad de la red, la recepción de notificaciones, desde un servidor, de un nuevo contenido (por ejemplo, mensajes de correo electrónico, contenido multimedia, etc.) está disponible para su descarga, y similares. Sin embargo, debe apreciarse que estas acciones y estos eventos son meramente ilustrativos, ya que otras acciones y otros eventos también pueden hacer que se ponga en marcha una aplicación. Por ejemplo, un desarrollador de sistemas operativos, un desarrollador de aplicaciones y/o un usuario pueden especificar cualquier acción adecuada para hacer que una aplicación se inicie en primer plano, y cualquier evento adecuado para hacer que una aplicación se inicie en segundo plano. Ejemplos de tales eventos incluyen, sin limitación, eventos sintéticos creados por un sistema operativo para organizar el trabajo para un uso económico eficiente de los recursos, como un evento que indica el inicio de un periodo de operación designado para el mantenimiento, un evento que indica la alta disponibilidad de uno o más servidores externos, y un evento que indica la disponibilidad de una red de bajo coste (por ejemplo, una red inalámbrica gratuita).

En algunas realizaciones, los componentes de primer plano y de segundo plano se pueden desacoplar lógicamente, de tal manera que faciliten la ejecución concurrente. En un ejemplo, se pueden implementar los componentes de primer plano y de segundo plano para evitar la comunicación directa, lo que puede reducir la probabilidad de cuelgue (por ejemplo, un componente de primer plano y un componente de segundo plano que se esperan simultáneamente el uno al otro para completar una respectiva tarea, o un componente de segundo plano que se ejecuta en el mismo subproceso que un componente de primer plano que evita el procesamiento inmediato de la actividad en primer plano). Así, en algunas realizaciones, un componente de segundo plano puede depender de un sistema operativo para informar del progreso a un correspondiente componente de primer plano, como para notificar al componente de primer plano que el componente de segundo plano ha finalizado su ejecución. El sistema operativo puede hacerlo a través de una interfaz de informe de estado de segundo plano proporcionada por el componente de primer plano, como la interfaz de “Estado de SP” del componente 105 de primer plano mostrado en la FIG. 1. Tal mecanismo de notificación puede permitir que los componentes de primer plano y de segundo plano trabajen juntos para proporcionar una experiencia unificada única a un usuario. Como ejemplo específico, cuando un componente de segundo plano descarga un libro electrónico desde un servidor, el progreso de la descarga se puede comunicar al usuario a través del correspondiente componente de primer plano cuando el usuario pone en marcha el componente de primer plano.

Otro planteamiento ilustrativo del desacoplamiento es evitar la operación directa en datos compartidos, lo que puede reducir la probabilidad de corrupción de datos. Por ejemplo, en una realización, un componente de primer plano y un componente de segundo plano pueden usar el almacenamiento persistente de algunas o todas las comunicaciones de datos para reducir la probabilidad de corrupción de datos. En algunas realizaciones adicionales, un componente de primer plano y un componente de segundo plano pueden operar en un objeto compartido, tal como los objetos compartidos 115A y 115B mostrados en la FIG. 1, pero solo invocando una o más interfaces (no mostradas) en el objeto compartido. Dicho objeto compartido puede estar adaptado para serializar el acceso a los datos, evitando así la corrupción de los datos.

Debería apreciarse que las comunicaciones a través de un sistema operativo y/u objetos compartidos son formas meramente ilustrativas que se pueden usar, ya sea solas o en combinación, para desacoplar lógicamente componentes de segundo plano de componentes de primer plano. También se pueden usar otras formas de desacoplamiento, ya que los aspectos de la presente divulgación no están limitados a ninguna forma particular de desacoplamiento.

En el ejemplo mostrado en la FIG. 1, la aplicación 100 incluye, además, una especificación 120 que contiene información que puede ser usada por un sistema operativo para cargar y administrar por separado componentes de primer plano y de segundo plano. La especificación 120 se puede escribir en cualquier formato adecuado que un sistema operativo esté configurado para procesar. Por ejemplo, la especificación 120 puede tener la forma de un manifiesto de aplicación escrito en un lenguaje de marcación adecuado, tal como el lenguaje de marcación ampliable (XML).

La especificación 120 puede contener cualquier combinación adecuada de información útil para un sistema operativo. En un ejemplo, la especificación 120 puede identificar el código ejecutable correspondiente a un componente de segundo plano (por ejemplo, uno de los elementos de trabajo 110A, 110B, 110C, …) y las correspondientes interfaces

de “Iniciar” y "Cancelar" para uso de un sistema operativo para, respectivamente, iniciar y detener la ejecución del componente de segundo plano.

En otro ejemplo, la especificación 120 puede especificar uno o más activadores en respuesta a los cuales un sistema operativo debe ejecutar un componente de segundo plano. En algunas realizaciones, la especificación 120 puede especificar, además, una o más condiciones que han de ser verificadas antes de que el sistema operativo ejecute el componente de segundo plano. Como se expone con mayor detalle a continuación en relación con las FIGURAS 5-7, estos activadores y/o estas condiciones pueden ser usados por un sistema operativo para determinar cuándo ha de ejecutarse el componente de segundo plano, por ejemplo, para controlar el consumo de recursos por el componente de segundo plano.

En otro ejemplo adicional, la especificación 120 puede especificar una o más interfaces de informe de estado de trabajo de segundo plano implementadas por un componente de primer plano (por ejemplo, la interfaz de “Estado de SP” del componente 105 de primer plano mostrado en la FIG. 1). Según se mencionó anteriormente, un sistema operativo puede usar tal interfaz para informar del progreso de ejecución de un componente de segundo plano.

De nuevo, los elementos de información descritos anteriormente son meramente ilustrativos. Un desarrollador de aplicaciones puede elegir incluir cualquier combinación adecuada de información en una especificación para ayudar a un sistema operativo a cargar y administrar por separado componentes de primer plano y de segundo plano. Alternativamente, o adicionalmente, un proveedor del sistema operativo puede solicitar que se proporcionen algunos elementos seleccionados de información y/o herramientas en una especificación para que un sistema operativo los use para cargar y gestionar por separado los componentes de primer plano y de segundo plano.

En algunas realizaciones, un componente de primer plano y un componente de segundo plano de una aplicación pueden estar tan suficientemente desacoplados que un sistema operativo pueda mover por separado los componentes de primer plano y de segundo plano durante sus respectivos ciclos de vida. La FIG. 2 muestra un diagrama ilustrativo de estados para una aplicación que tiene un componente de primer plano y un componente de segundo plano que pueden ser ejecutados de manera concurrente.

En el ejemplo mostrado en la FIG. 2, el diagrama de estado para la aplicación incluye dos diagramas, 200 y 250, de estado de componentes separados. El diagrama 200 de estado muestra un ciclo de vida ilustrativo para un componente de primer plano de la aplicación, y el diagrama 250 de estado muestra un ciclo de vida ilustrativo para un componente de segundo plano de la aplicación.

En el ciclo 200 de vida ilustrativo, el componente de primer plano puede estar en uno de tres estados diferentes: estado activo 205, estado suspendido 210 y estado 215 de no ejecución. En el estado activo 205, el componente de primer plano reside en la memoria o en un fichero de paginación activo, y está siendo planificado por un sistema operativo para su ejecución. En el estado suspendido 210, el componente de primer plano sigue residiendo en la memoria o en un fichero de paginación activo, pero ya no está planificado por el sistema operativo para su ejecución. En el estado 215 de no ejecución, el componente de primer plano ya no reside en la memoria, ni en ningún fichero de paginación activo.

Por otro lado, en el ciclo 250 de vida ilustrativo mostrado en la FIG. 2, el componente de segundo plano puede estar en uno de dos estados diferentes: estado 255 de ejecución y estado 260 de no ejecución. En el estado 255 de ejecución, el componente de segundo plano reside en la memoria o en un fichero de paginación, y está siendo planificado por un sistema operativo para su ejecución. En el estado 260 de no ejecución, el componente de segundo plano puede no residir ya en la memoria, ni en ningún fichero de paginación. Alternativamente, un componente de segundo plano puede ejecutarse en un entorno de alojamiento (por ejemplo, un contenedor) y el entorno de alojamiento puede almacenarse en la memoria intermedia cuando el componente de segundo plano es pasado al estado 260 no en ejecución.

Así, en este ejemplo, la aplicación en su conjunto, que incluye tanto el componente de primer plano como el componente de segundo plano, puede tener seis estados diferentes: <primer plano activo, segundo plano ejecutándose>, <primer plano suspendido, segundo plano ejecutándose> , <primer plano no ejecutándose, segundo plano ejecutándose>, <primer plano activo, segundo plano no ejecutándose>, <primer plano suspendido, segundo plano no ejecutándose> y <primer plano no ejecutándose, segundo plano no ejecutándose>. La aplicación en su conjunto puede pasar por estos seis estados a medida que los componentes de primer plano y de segundo plano atraviesan sus respectivos ciclos de vida, que son descritos con mayor detalle a continuación. En el último estado, <primer plano no ejecutándose, segundo plano no ejecutándose>, la aplicación puede considerarse finalizada, ya que la aplicación no se está ejecutando ni en el primer plano ni en el segundo.

Como se mencionó anteriormente, debido a que los componentes de primer plano y de segundo plano están lógicamente desacoplados, el sistema operativo puede ser capaz de pasar un componente de un estado a otro independientemente del estado del otro componente y sin afectar al otro componente. Por ejemplo, en algunas realizaciones, el componente de primer plano puede pasar del estado activo al estado suspendido, o viceversa, independientemente del estado del componente de segundo plano y sin afectarlo. Como un ejemplo más específico, cuando el componente de primer plano pasa del estado activo 205 al estado suspendido 210, la aplicación en su

conjunto puede pasar ya sea de <primer plano activo, segundo plano ejecutándose> a <primer plano suspendido, segundo plano ejecutándose>, o de <primer plano activo, segundo plano no ejecutándose> a <primer plano suspendido, segundo plano no ejecutándose>, dependiendo de si el componente de segundo plano está en el estado 255 de ejecución o el estado 260 de no ejecución.

Debe apreciarse que, aunque los componentes de primer plano y segundo plano pueden implementarse de manera lógica desacoplada, un sistema operativo puede (aunque no es preciso que así sea) usar información sobre la ejecución del componente de primer plano para gestionar la ejecución del componente de segundo plano, o viceversa. Por ejemplo, en algunas realizaciones, el sistema operativo puede planificar la ejecución del componente de segundo plano únicamente cuando el componente de primer plano está suspendido o no se está ejecutando. En resumen, el desacoplamiento lógico de los componentes del primer plano y del segundo plano puede dar a un sistema operativo la opción ya sea de crear dependencias entre las ejecuciones de los componentes o no crear tales dependencias.

Volviendo ahora al ciclo 200 de vida ilustrativo mostrado en la FIG. 2, el sistema operativo puede emplear cualquier conjunto adecuado de reglas para determinar cuándo pasar el componente de primer plano de un estado a otro. Tales reglas pueden estar diseñadas para evitar que un componente de primer plano que no está captando la atención de un usuario consuma recursos como energía de la batería, ciclos del procesador y memoria. Imponer estas reglas puede mejorar la vida útil de la batería y/o la capacidad de respuesta de un componente de primer plano que esté captando la atención de un usuario. En un ejemplo, el sistema operativo puede pasar un componente de primer plano que no está recibiendo la atención de un usuario de un estado suspendido a un estado sin ejecución para recuperar recursos de memoria.

En algunas realizaciones, el componente de primer plano puede estar inicialmente en el estado de no ejecución, y el sistema operativo puede iniciar el componente de primer plano en respuesta a una acción del usuario (por ejemplo, que un usuario haga clic en un icono o mosaico correspondiente a la aplicación). Iniciar el componente de primer plano puede implicar cargar en la memoria el componente de primer plano y cualquier componente auxiliar al que pueda acceder el componente de primer plano, inicializar el componente de primer plano, y/o asignar los recursos apropiados. Si ya se está ejecutando un componente de segundo plano de la aplicación en el momento de iniciar el componente de primer plano y el componente de primer plano se carga en el mismo proceso que el componente de segundo plano, algunos componentes auxiliares pueden ya estar cargados y algunos recursos pueden ya estar asignados. Si no, se puede realizar una puesta en marcha completa. En cualquier caso, el componente de primer plano puede iniciarse en el estado activo 205, y puede ser planificado por el sistema operativo para su ejecución.

En algunas realizaciones, el sistema operativo puede ejecutar una función de punto de entrada del componente de primer plano invocando una función API proporcionada por el componente de primer plano para inicializar el componente de primer plano una vez que el componente de primer plano se carga desde el almacenamiento. Más específicamente, en el ejemplo mostrado en las FIGURAS 1-2, el sistema operativo puede llamar lo que se denomina procedimiento de “Inicio” implementado por el componente de primer plano.

El sistema operativo puede mantener al componente de primer plano en el estado activo 205 cuando un usuario está interactuando activamente con el componente de primer plano, para evitar interferir en la experiencia del usuario. Un usuario puede interactuar activamente con el componente de primer plano de varias maneras diferentes; por ejemplo, proporcionando información al componente de primer plano a través de una interfaz de usuario. Con el fin de determinar si el componente de primer plano ha de permanecer en el estado activo, también se puede considerar que un usuario interactúe activamente con el componente de primer plano si el componente de primer plano está generando una salida (por ejemplo, reproduciendo audio y/o vídeo) o completando una tarea solicitada por el usuario (por ejemplo, enviar datos a través de una conexión de red, escribir en un fichero, imprimir, etc.).

En algunas realizaciones, el sistema operativo puede pasar al componente de primer plano del estado activo 205 al estado suspendido 210 cuando el usuario ya no está interactuando activamente con el componente de primer plano. Esto sucede, por ejemplo, cuando el usuario cambia el foco a una ventana de una aplicación diferente, minimiza una ventana del componente de primer plano, no proporciona ninguna entrada al componente de primer plano durante un periodo de tiempo umbral, etc. Alternativamente, o adicionalmente, el sistema operativo puede pasar al componente de primer plano del estado activo 205 al estado suspendido 210 cuando el componente de primer plano ya no realiza ninguna actividad que pueda considerarse como una actividad de primer plano. Los ejemplos de actividades que pueden considerarse actividades de primer plano incluyen, sin limitación, reproducción de audio, sincronización de dispositivos, etc.

Antes de pasar al componente de primer plano del estado activo 205 al estado suspendido 210, el sistema operativo puede informar al componente de primer plano que el componente de primer plano está a punto de quedar en suspenso, por lo que el componente de primer plano puede prepararse en consecuencia; por ejemplo, guardando el estado y liberando los recursos asignados. Por ejemplo, el sistema operativo puede informar al componente de primer plano invocando una función de interfaz de programación de aplicaciones (API) proporcionada por el componente de primer plano. En un ejemplo más específico, mostrado en las FIGURAS 1 y 2, el sistema operativo puede llamar a un procedimiento llamado “Inactivación” implementado por el componente de primer plano, que puede hacer que el componente de primer plano guarde el estado y libere recursos (por ejemplo, ficheros, conexiones de red, etc.). En

algunas implementaciones, el procedimiento Inactivación puede incluir el código del sistema operativo para bloquear explícitamente el subproceso que invocó el procedimiento Inactivación (es decir, un subproceso del componente de primer plano) para que no sea planificado para su ejecución. Esto puede hacerse además de suspender (por ejemplo, deshabilitar la planificación de) todo el componente de primer plano en su conjunto. El componente del primer plano puede permanecer bloqueado hasta que el sistema operativo pase el componente de primer plano al estado activo 205.

En algunas realizaciones, el sistema operativo puede pasar al componente de primer plano del estado 210 suspendido al estado activo 205 cuando el usuario comienza a interactuar nuevamente con el componente de primer plano; por ejemplo, cambiando el foco de nuevo a una ventana del componente de primer plano, restaurando una ventana del componente de primer plano desde un estado minimizado, empezando a proporcionar indicaciones al componente de primer plano, o similares. Para hacerlo, el sistema operativo puede simplemente planificar el componente de primer plano para su ejecución. En algunas realizaciones, el sistema operativo puede invocar adicionalmente una función API proporcionada por el componente de primer plano para ayudar al componente de primer plano a restaurar un estado anterior. Más específicamente, en los ejemplos mostrados en las FIGURAS 1 y 2, el sistema operativo puede llamar a un procedimiento denominado “Reanudar” implementado por el componente de primer plano, que puede hacer que el componente de primer plano recupere el estado guardado y solicite los recursos apropiados (por ejemplo, ficheros, conexiones de red, etc.).

En algunas realizaciones adicionales, el sistema operativo puede pasar al componente de primer plano del estado suspendido 210 al estado 215 de no ejecución eliminando al componente de primer plano de la memoria. Esto se puede hacer, por ejemplo, para reducir el uso de la memoria o para permitir una actualización del código ejecutable para el componente de primer plano. Si la aplicación no se ejecuta en segundo plano en el momento en que el sistema operativo pasa al componente de primer plano al estado 215 de no ejecución, la aplicación puede terminar por completo. Si no, el componente de segundo plano puede seguir ejecutándose, independientemente del estado de cambio del componente de primer plano.

Si el usuario se pone a interactuar con el componente de primer plano nuevamente cuando el componente de primer plano ha pasado al estado 215 de no ejecución, el sistema operativo puede iniciar nuevamente el componente de primer plano, lo que puede ser similar a la puesta en marcha inicial del componente de primer plano, como se expuso anteriormente.

Debería apreciarse que el sistema operativo puede elegir cualquier conjunto adecuado de reglas y técnicas para determinar cuándo pasar al componente de primer plano del estado activo 205 al estado suspendido 210 y cuándo pasar al componente de primer plano del estado suspendido 210 al estado 215 de no ejecución. Estas reglas pueden reflejar un equilibrio deseado entre los beneficios previstos y los costes previstos asociados con la suspensión del componente de primer plano, y un equilibrio deseado entre los beneficios previstos y los costes previstos asociados con la eliminación del componente de primer plano de la memoria. Por ejemplo, aunque suspender el componente de primer plano puede llevar a algunos ahorros en recursos, el componente de primer plano puede incurrir en una sobrecarga de procesamiento adicional para prepararse para la suspensión y para la reanudación a partir de la misma. Además, el usuario puede experimentar lentitud cuando el componente de primer plano se reanuda por vez primera desde la suspensión e intenta recuperar recursos (por ejemplo, volver a abrir ficheros, restablecer las conexiones de red, etc.). De manera similar, aunque eliminar el componente de primer plano de la memoria puede reducir el uso de memoria, el componente de primer plano puede incurrir en una sobrecarga de procesamiento adicional al reiniciarse, y el usuario puede experimentar lentitud cuando el componente de primer plano se está recargando y reinicializando.

Volviendo al ciclo 250 de vida ilustrativo mostrado en la FIG. 2, el componente de segundo plano puede estar inicialmente en el estado 260 de no ejecución. El sistema operativo puede, en algunas realizaciones, iniciar el componente de segundo plano en respuesta a la detección de una incidencia de uno o más eventos. Como se explica con mayor detalle a continuación en relación con las FIGURAS 5-7, se pueden tomar en cuenta dos tipos de eventos (es decir, activadores y condiciones) para determinar cuándo iniciar el componente de segundo plano, en algunas realizaciones ilustrativas.

La puesta en marcha del componente de segundo plano puede implicar cargar en la memoria el componente de segundo plano y cualquier componente auxiliar al que pueda acceder el componente de segundo plano, inicializar el componente de segundo plano, y/o asignar los recursos apropiados. Si ya se está ejecutando un componente de primer plano de la aplicación en el momento de iniciar el componente de segundo plano y el componente de segundo plano se carga en el mismo proceso que el componente de primer plano, es posible que algunos componentes auxiliares ya estén cargados y algunos recursos pueden estar ya asignados. Si no, se puede realizar una puesta en marcha completa. En cualquier caso, el componente de segundo plano puede pasar al estado 255 de ejecución.

En algunas realizaciones, el sistema operativo puede ejecutar una función de punto de entrada del componente de segundo plano invocando una función API proporcionada por el componente de segundo plano para inicializar el componente de segundo plano una vez que el componente de segundo plano se carga desde el almacenamiento. En un ejemplo más específico, mostrado en las FIGURAS 1-2, el sistema operativo puede llamar al procedimiento denominado “Inicio” implementado por el componente de segundo plano.

En algunas realizaciones, el sistema operativo puede monitorizar el progreso del componente de segundo plano después de que se inicie el componente de segundo plano. Algunos componentes de segundo plano (por ejemplo, los elementos de trabajo 110A, 110B, 110C, … mostrados en la FIG. 1) pueden corresponder a una tarea de segundo plano diferenciada (por ejemplo, descargar un contenido de un servidor, verificar la disponibilidad de una actualización de soporte lógico, realizar un escaneo rutinario de virus, etc.). Como tales, estos componentes de segundo plano no están previstos para ejecutarse indefinidamente. Si el componente de segundo plano no ha progresado lo suficiente después de ejecutarse durante un periodo de tiempo umbral, el sistema operativo puede detener la ejecución del componente de segundo plano, pasando así al componente de segundo plano al estado 260 de no ejecución.

El sistema operativo puede monitorizar el progreso del componente de segundo plano usando cualquier técnica conocida para monitorizar el progreso de las tareas informáticas. Alternativamente, o adicionalmente, una aplicación puede proporcionar una o más herramientas para que el sistema operativo las use para monitorizar el progreso. Por ejemplo, un componente de segundo plano puede implementar una interfaz de “verificación de progreso” que un sistema operativo puede invocar para obtener información de progreso (por ejemplo, porcentaje completado de la tarea y/o si el componente de segundo plano está esperando la respuesta de otro componente o que algún recurso esté disponible). Esta interfaz puede ser especificada en una especificación asociada con la aplicación, como la especificación 120 mostrada en la FIG. 1.

En algunas realizaciones, el sistema operativo puede informar al componente de segundo plano de que el componente de segundo plano está a punto de quedar detenido, por lo que el componente de segundo plano se puede preparar en consecuencia, por ejemplo, guardando el estado y liberando los recursos asignados. Esto se puede hacer, por ejemplo, invocando una función API proporcionada por el componente de segundo plano. Más específicamente, en el ejemplo mostrado en las FIGURAS 1-2, el sistema operativo puede llamar a un procedimiento llamado “Cancelar” implementado por el componente de segundo plano, después de lo cual se le puede dar al componente de segundo plano un poco de tiempo (por ejemplo, unos segundos) antes de que el sistema operativo haga pasar al componente de segundo plano al estado 260 de no ejecución.

Si el componente de segundo plano continúa progresando a una velocidad suficiente, el sistema operativo puede permitir que el componente de segundo plano se ejecute hasta su finalización. En ese caso, el sistema operativo también puede pasar al componente de segundo plano al estado 260 de no ejecución; por ejemplo, eliminando al componente de segundo plano de la memoria.

Si la aplicación no se está ejecutando en primer plano en el momento en que el sistema operativo pasa el componente de segundo plano al estado 215 de no ejecución, la aplicación puede terminar por completo. Si no, el componente de primer plano puede seguir ejecutándose (por ejemplo, en el estado activo 205 o el estado suspendido 210), independientemente del estado de cambio del componente de segundo plano.

Aunque en la FIG. 2 se muestran ejemplos detallados de los ciclos de vida de los componentes de la aplicación, y son descritos anteriormente, debe apreciarse que estos ejemplos se proporcionan únicamente con fines ilustrativos. Los aspectos de la presente divulgación no están limitados a ninguna definición particular de los ciclos de vida, ni a ninguna manera particular de administrar los ciclos de vida. Por ejemplo, un componente de primer plano puede atravesar un conjunto de estados diferentes de los mostrados en la FIG. 2, y un sistema operativo puede emplear directrices diferentes para determinar cuándo pasar el componente de primer plano de un estado a otro, y de manera similar para un componente de segundo plano. Como otro ejemplo, un sistema operativo puede cargar una o más instancias de un mismo componente de segundo plano o diferentes componentes de segundo plano en un mismo entorno (por ejemplo, un contenedor de soporte lógico), y puede eliminar el entorno de la memoria solo cuando ninguna de instancias se esté ejecutando. Como otro ejemplo más, un sistema operativo puede mantener un componente de segundo plano cargado indefinidamente para mejorar la eficiencia.

Como se expuso anteriormente, un sistema operativo puede, en algunas realizaciones, elegir cargar los componentes de primer plano y de segundo plano de una aplicación, ya sea en el mismo proceso o en procesos separados, si los componentes de primer plano y de segundo plano son capaces de ser ejecutados de cualquiera de las dos maneras. En un ejemplo, el sistema operativo puede cargar por defecto los componentes de primer plano y de segundo plano en procesos separados, a no ser que una aplicación o un usuario especifique lo contrario. En otro ejemplo, el sistema operativo puede cargar un componente de segundo plano proporcionado por una aplicación por separado del correspondiente componente de primer plano, pero puede colocalizar un componente de segundo plano proporcionado por el sistema operativo con una aplicación para la cual el componente de segundo plano esta realizando trabajo. También se pueden aplicar otras reglas adecuadas, ya que los aspectos de la presente divulgación no están limitados de esta manera.

La FIG. 3A muestra un ejemplo en el que un sistema operativo 305 carga en la memoria 300 un componente de primer plano 310 en el proceso 320 y un componente 315 de segundo plano en un proceso separado 325. Aislar el trabajo de segundo plano del trabajo de primer plano de esta manera puede facilitar la gestión de recursos. Por ejemplo, en algunas realizaciones, el sistema operativo puede ejecutar el proceso 325 en un entorno de ejecución controlada (por ejemplo, un enclave de seguridad, como se denomina en la técnica) construido para permitir que los procesos que se ejecutan en él accedan solo a los recursos designados. De esta manera, el sistema operativo puede asignar una

cantidad específica de recursos (por ejemplo, algunos porcentajes especificados de tiempo de CPU, memoria, etc.) al entorno de ejecución controlada para garantizar que el componente 315 de segundo plano no consuma más que la cantidad de recursos especificada. El entorno de ejecución controlada puede, en algunas implementaciones, estar designado para ejecutar componentes de segundo plano, de modo que los componentes de segundo plano de otras aplicaciones también se puedan ejecutar en el enclave de seguridad y estén sujetos a las mismas restricciones de recursos.

El aislamiento del trabajo de segundo plano con respecto al trabajo de primer plano en procesos separados también puede ayudar a reducir la ocupación de memoria, ya que el sistema operativo puede evitar cargar código ejecutable (por ejemplo, bibliotecas) relacionado solo con las funcionalidades de la interfaz de usuario (UI) cuando la aplicación únicamente está realizando trabajo de segundo plano. Por ejemplo, en una realización en la que el trabajo de primer plano y el trabajo de segundo plano se implementan como componentes separados, las funcionalidades relacionadas con la IU pueden estar presentes únicamente en un componente de primer plano, que no es preciso que se cargue en un proceso que aloja un componente de segundo plano.

La FIG. 3B muestra un ejemplo en el que un sistema operativo 355 carga en la memoria 350 un componente 360 de primer plano y un componente 365 de segundo plano en el mismo proceso 370. La colocalización del trabajo de primer plano y el trabajo de segundo plano de esta manera puede permitir el componente 360 de primer plano y el componente 365 de segundo plano compartan los costes de la puesta en marcha. Por lo tanto, puede ser beneficioso colocalizar los componentes si uno de los componentes se ejecuta con frecuencia; por ejemplo, si uno o más usuarios interactúan con frecuencia con la aplicación, o si la aplicación realiza con frecuencia el trabajo de segundo plano.

En el ejemplo mostrado en la FIG. 3B, el componente 360 de primer plano y el componente 365 de segundo plano se inician en el sistema operativo 355 como subprocesos separados: los subprocesos 380 y 385, respectivamente. De esta manera, el sistema operativo 355 sigue pudiendo gestionar las ejecuciones del componente 360 de primer plano y del componente 365 de segundo plano por separado. Por ejemplo, el sistema operativo puede bloquear la ejecución del subproceso 380 (y, por lo tanto, del componente 360 de primer plano), mientras que permite que se ejecute el subproceso 385 (y, por lo tanto, el componente 365 de segundo plano), y viceversa.

Así, en algunas realizaciones, un sistema operativo puede adaptarse para decidir si colocalizar o separar los componentes de primer plano y de segundo plano en función de una solución de compromiso deseada entre una gestión de recursos fácil de implementar y costes reducidos de puesta en marcha. También se pueden tener en cuenta otras consideraciones (por ejemplo, condiciones operativas dinámicas, como el uso del procesador y/o de la memoria, el número de procesos activos, etc.), ya que los aspectos de la presente divulgación no están limitados a ninguna razón particular para la colocalización o la separación de los componentes de primer plano y de segundo plano. En un ejemplo, el consumo de memoria y/o de ciclos del procesador puede ser tenido en cuenta al decidir si colocalizar o separar los componentes de primer plano y de segundo plano en una implementación en la que se pueden evitar los interbloqueos manteniendo algunos o todos los subprocesos de un componente de primer plano ejecutable mientras se ejecuta un componente de segundo plano. En otro ejemplo, se puede tener en cuenta la fiabilidad del soporte lógico, por ejemplo, al separar un componente de segundo plano con errores para reducir la interferencia con otros componentes de primer plano y/o componentes de segundo plano.

La FIG. 4 muestra un ejemplo de un sistema operativo 400 adaptado para administrar el trabajo de segundo plano y el trabajo de primer plano por separado. En este ejemplo, el sistema operativo 400 implementa un modelo de ejecución negociada para los componentes de segundo plano, dando de alta cada aplicación en el sistema operativo 400 uno o más componentes de la aplicación que está previsto que se ejecuten en segundo plano, y el sistema operativo 400 controla en último término cuándo, dónde, y/o por cuánto tiempo se ejecuta cada componente de segundo plano. De esta manera, el sistema operativo 400 puede controlar efectivamente el consumo de recursos por los componentes de segundo plano.

En el ejemplo mostrado en la FIG. 4, el sistema operativo 400 implementa el modelo de ejecución negociada usando una infraestructura 405 de agentes (BI) y varios agentes. En este ejemplo, el sistema operativo 400 está preprogramado con agentes, cada uno configurado para gestionar un tipo respectivo de recursos y/o procesar un tipo respectivo de eventos. Por ejemplo, el sistema operativo 400 puede incluir, según se ilustra, el agente 415A de temporización, el agente 415B de conexiones de red, el agente 415C de activación remota y el agente 415D de eventos del sistema. Tal marco puede ser fácilmente extensible, por ejemplo, al permitir que terceros proporcionen nuevos agentes adaptados para administrar nuevos tipos de recursos y/o procesar nuevos tipos de eventos (por ejemplo, se puede proporcionar un agente de ubicación para que la ejecución del trabajo de segundo plano pueda depender de la ubicación geográfica). Sin embargo, debe apreciarse que un modelo de ejecución negociada no requiere el uso de un marco que tenga una infraestructura de agentes y varios agentes. Más bien, en realizaciones alternativas, las funcionalidades de la infraestructura de agentes y los agentes individuales pueden ser proporcionadas mediante un solo componente del sistema operativo o uno o más componentes fuera del sistema operativo.

Como se analiza con mayor detalle a continuación en relación con las FIGURAS 8-9, la infraestructura 405 de agentes puede ser responsable de gestionar la ejecución de los componentes de segundo plano, tales como los elementos de trabajo. Por ejemplo, en algunas realizaciones, la infraestructura 405 de agentes puede proporcionar una interfaz (por

ejemplo, la interfaz de “Dar de alta” que se muestra en la FIG. 4) para ser usada por una aplicación para dar de alta un componente de segundo plano para la ejecución negociada. El alta de un componente de segundo plano puede incluir recibir de la aplicación una identificación del componente de segundo plano y una indicación de cuándo ha de ejecutarse el componente de segundo plano (por ejemplo, especificando uno o más activadores y/o condiciones). Por ejemplo, la aplicación puede proporcionar un identificador de objeto que identifique un objeto de soporte lógico que implementa un componente de segundo plano, y puede identificar un evento en respuesta al cual ha de ejecutarse el componente de segundo plano.

En algunas realizaciones, la infraestructura 405 de agentes puede ejecutar un componente de segundo plano siempre que se detecte una incidencia de un evento correspondiente. Alternativamente, la infraestructura 405 de agentes puede imponer restricciones y/o limitaciones adicionales según las directrices apropiadas de gestión de recursos. Por ejemplo, las restricciones y/o limitaciones adicionales pueden reflejar directrices diseñadas para evitar que una sola aplicación consuma demasiados recursos y/o para evitar que las aplicaciones instaladas pero no en uso consuman cualquier recurso. En una realización, los recursos pueden asignarse a diferentes aplicaciones en función de la cantidad de aplicaciones que un usuario utilice. Por ejemplo, una cantidad de recursos asignados a una aplicación puede ser proporcional al uso de la aplicación como porcentaje de uso de todas las aplicaciones. Dicha directriz puede ser deseable en un escenario común en el que un usuario instala una gran cantidad (por ejemplo, cientos) de aplicaciones, pero solo usa una pequeña cantidad (por ejemplo, 10, 15 o 20) de forma continua.

En algunas realizaciones adicionales, la infraestructura 405 de agentes puede monitorizar el progreso y/o el consumo de recursos de un componente de segundo plano durante la ejecución. Esto se puede hacer usando cualquier técnica conocida para monitorizar el progreso y/o el consumo de recursos asociado con un componente de soporte lógico. Alternativamente, o adicionalmente, la infraestructura de agentes puede invocar una interfaz proporcionada por el componente de segundo plano para informar del progreso y/o del consumo de recursos. La infraestructura 405 de agentes puede descubrir tal interfaz a partir de una especificación de la aplicación, como la especificación 120 mostrada en la FIG. 1.

Si el componente de segundo plano no avanza a una velocidad suficiente, o si el componente de segundo plano consume demasiados recursos, la infraestructura 405 de agentes puede detener la ejecución del componente de segundo plano. Nuevamente, la decisión de detener a un componente de segundo plano puede basarse en cualquier directriz apropiada de administración de recursos.

En otras realizaciones adicionales, la infraestructura 405 de agentes puede ser responsable de informar del progreso del trabajo de segundo plano a un componente de primer plano. Esto puede contribuir a evitar la comunicación directa entre un componente de segundo plano y un componente de primer plano, lo que, como se ha explicado anteriormente, puede facilitar el desacoplamiento lógico entre los componentes de primer plano y de segundo plano.

En un ejemplo, la infraestructura 405 de agentes puede notificar al componente de primer plano cuándo se completa con éxito el componente de segundo plano, de modo que el componente de primer plano pueda realizar cualquier procesamiento adecuado. Por ejemplo, cuando una descarga se completa en segundo plano, la infraestructura 405 de agentes puede notificar a un componente correspondiente de primer plano, el cual, a su vez, puede notificar a un usuario que hay nuevo contenido disponible. En otro ejemplo, la infraestructura de agentes 405 puede notificar a un componente de primer plano correspondiente cuándo se detiene un componente de segundo plano por un progreso insuficiente o un consumo excesivo de recursos. Aunque no es preciso que lo haga, el componente de primer plano puede notificar el fallo a un usuario, el cual puede optar por reiniciar el componente de segundo plano con una exención a la directriz de gestión de recursos que fue violada por el componente de segundo plano. Sin embargo, debe tenerse en cuenta que el componente de primer plano puede no estar activo en el momento en que se genera una notificación. En tal escenario, la infraestructura 405 de agentes puede notificar al componente de primer plano cuando el componente de primer plano se active en el futuro (por ejemplo, cuando un usuario haga clic en un mosaico o icono correspondiente al componente de primer plano). Alternativamente, la infraestructura 405 de agentes puede gestionar la notificación; por ejemplo, notificando a un usuario el éxito o el fracaso de una tarea en segundo plano.

Volviendo a la FIG. 4, una aplicación puede, en algunas realizaciones, especificar lo que se denomina “evento negociado” como un evento en respuesta al cual ha de ejecutarse el componente de segundo plano. Un evento negociado puede ser un evento señalado por un agente de componente, como el agente 415A de temporización, el agente 415B de conexiones de red, el agente 415C de activación remota, el agente 415D de eventos del sistema o algún otro agente. Cada uno de dichos agentes puede ser adaptado para señalizar eventos en ciertas circunstancias específicas. Por ejemplo, el agente 415A de temporización puede estar adaptado para señalizar eventos en tiempos absolutos, a intervalos regulares o a ciertas horas “convenientes” relacionadas con la operación del sistema (por ejemplo, tiempo de mantenimiento para realizar actividades opcionales), el agente 415B de conexiones de red puede estar adaptado para indicar eventos cuando hay datos disponibles en una interfaz de conexión de red, el agente 415C de activación remota puede estar adaptado para reenviar eventos que llegan desde uno o más servidores en la nube, y el agente 415D de eventos del sistema puede estar adaptado para indicar eventos, como el inicio de sesión o la desconexión de un usuario.

En algunas implementaciones, un agente puede proporcionar una interfaz (por ejemplo, las interfaces de “Solicitud” mostradas en la FIG. 4) a través de las cuales una aplicación puede solicitar que el agente señalice un evento negociado en ciertas circunstancias especificadas por la aplicación. De esta manera, la aplicación puede informar al sistema operativo cuándo está previsto que se ejecute un componente de segundo plano. Por ejemplo, como se explica con mayor detalle a continuación en relación con las FIGURAS 5-6, un agente puede permitir que una aplicación solicite que se señalice un evento negociado cuando se hayan desencadenado uno o más “activadores”. Además, la aplicación puede especificar una o más “condiciones”, además del desencadenamiento de un activador, que deben satisfacerse antes de señalizar el evento negociado.

Como la infraestructura 405 de agentes, un agente puede, en algunas realizaciones, imponer una o más directrices apropiadas de gestión de recursos; por ejemplo, a través de una o más restricciones y/o limitaciones. Por ejemplo, en algunos casos, un agente puede no señalizar un evento negociado, ni siquiera cuando se haya desencadenado un activador y se cumplan todas las condiciones, ya que la señalización del evento negociado puede violar alguna directriz de gestión de recursos. Como ejemplo más específico, el agente 415A de temporización puede limitar el número de reiteraciones de un evento negociado por día, de modo que la aplicación que solicita el evento negociado no se ejecute en segundo plano más de esa cantidad de veces al día.

Aunque en lo que antecede se han expuesto varios ejemplos de agentes, se debe apreciar que estos agentes son meramente ilustrativos. En diversas realizaciones, un sistema operativo puede incluir cualquier combinación adecuada de agentes adaptados para imponer las directrices adecuadas de gestión de recursos. Algunas de las directrices pueden ser de naturaleza estática; por ejemplo, especificar una cantidad fija de recursos que una aplicación puede consumir en un periodo de tiempo fijo. Otras directrices pueden ser dinámicas y pueden imponer restricciones en el consumo según las condiciones de operación (por ejemplo, si el sistema se está ejecutando con batería, si el uso de la memoria y/o de la CPU se ha mantenido alto durante un periodo de tiempo sostenido, etc.). No obstante, otras directrices pueden combinar aspectos estáticos y dinámicos.

Volviendo de nuevo al ejemplo mostrado en la FIG. 4, el sistema operativo 400 incluye adicionalmente varios componentes de segundo plano, como los elementos de trabajo 410A, 410B, 410C, … Al igual que los elementos de trabajo 110A-C que se muestran en la FIG. 1, los elementos de trabajo 410A-C pueden ejecutarse en segundo plano para realizar el trabajo de una aplicación, y pueden ser gestionados por el sistema operativo 400 de la misma manera que los componentes de segundo plano proporcionados por una aplicación. Por ejemplo, el sistema operativo 400 puede imponer las mismas directrices de gestión de recursos contra los elementos de trabajo 410A-C y contra los elementos de trabajo 110A-C. Sin embargo, en realizaciones alternativas, el sistema operativo 400 puede confiar en los elementos de trabajo 410A-C pero no en los elementos de trabajo 110A-C, porque los elementos de trabajo 410A-C son parte del sistema operativo 400, mientras que los elementos de trabajo 110A-C contienen código proporcionado por la aplicación. En consecuencia, el sistema operativo 400 puede relajar las directrices aplicadas contra los elementos de trabajo 410A-C, porque los elementos de trabajo 410A-C son fiables. Por ejemplo, el sistema operativo 400 puede cargar los elementos de trabajo 110A-C en un enclave de seguridad separado de los componentes de primer plano, pero puede cargar los elementos de trabajo 410A-C en los mismos procesos que los respectivos componentes de primer plano.

De nuevo, debería apreciarse que los aspectos de la presente divulgación no están limitados al uso de componentes de segundo plano de cualquier fuente particular. Los componentes de segundo plano pueden, en diversas realizaciones, estar escritos por desarrolladores del sistema operativo, desarrolladores de aplicaciones y/o cualquier otro tercer proveedor de soporte lógico.

La FIG. 5 muestra un proceso ilustrativo 500 que puede ser realizado por un componente agente en respuesta a una solicitud de organización de un evento negociado según algunas relacionadas, y la FIG. 6 muestra una estructura 600 de datos ilustrativa que puede ser utilizada por el componente agente en relación con el proceso 500. Por ejemplo, el proceso 500 puede ser realizado por uno de los agentes 415A-D mostrados en la FIG. 4 en respuesta a una aplicación que invoque la interfaz de “Solicitud”.

En la etapa 505, el componente agente puede recibir una solicitud de un evento negociado. En diversas realizaciones, la solicitud puede incluir cualquier combinación de elementos de información que el agente pueda usar para decidir cuándo señalizar el evento negociado solicitado. Por ejemplo, la API del agente puede indicar ciertos elementos de la información deseada para permitir que el agente imponga una o más directrices de gestión de recursos. Ejemplos de tales elementos de información incluyen, sin limitación, una identificación del solicitante (por ejemplo, una aplicación), uno o más activadores en respuesta a los cuales se ha de señalizar el evento negociado solicitado, una o más condiciones que han de ser satisfechas cuando se señalice el evento negociado solicitado, una indicación de un propósito previsto para el evento negociado solicitado (por ejemplo, para hacer que un sistema operativo ejecute un componente de segundo plano seleccionado), y cualquier otro elemento de información adecuado.

En la etapa 510, el componente agente puede asignar un ID de evento negociado al evento negociado solicitado. Este ID de evento negociado puede ser único para cada solicitud de un evento negociado. Por ejemplo, si dos aplicaciones solicitan un evento negociado a través de una invocación diferente de la API del agente, el agente puede asignar diferentes ID de evento negociado, aunque los activadores subyacentes sean los mismos. De esta manera, el agente

puede señalizar dos eventos negociados diferentes cuando se desencadena el activador subyacente, teniendo cada evento negociado un ID de evento negociado diferente.

Aunque no se requiere que así sea, el ID único de eventos negociados puede ser beneficioso, porque diferentes aplicaciones pueden especificar diferentes condiciones para un evento negociado, aunque los activadores sean los mismos. Además, una o más directrices de gestión de recursos aplicadas por el agente pueden ser aplicadas de manera diferente dependiendo de la aplicación en particular. Por lo tanto, el ID único de eventos negociados puede ser usado para garantizar que se aplique un conjunto apropiado de condiciones, ya sea impuestas por una aplicación correspondiente o por el sistema operativo, para controlar la señalización de un evento negociado.

En la etapa 515, el agente puede identificar uno o más activadores en respuesta a los cuales ha de señalizarse el evento negociado solicitado. Un activador puede ser cualquier evento que el agente esté programado para detectar. En un ejemplo, un activador puede ser un evento relacionado con el usuario, tal como que un usuario inicie o cierre la sesión, que un usuario pase a estar activo (por ejemplo, al proporcionar cualquier indicación, como escribir, mover el ratón, hablar, etc.), que un usuario pase a estar inactivo (por ejemplo, al no proporcionar ninguna indicación durante al menos un periodo de tiempo umbral), que un usuario inicie una sesión de usuario, que un usuario solicite una descarga, que un usuario que inicie un trabajo de impresión o similar. En otro ejemplo, un activador puede ser un evento relacionado con el sistema, como la expiración de un temporizador, la llegada de un mensaje o una notificación (por ejemplo, un mensaje SMS o una notificación de mensaje de la empresa explotadora), un cambio en el estado de la red, que se restablezcan los canales de red, que Internet pase a estar disponible o no disponible, la finalización de una actualización de soporte lógico o similar. En otro ejemplo adicional, un activador puede ser un evento relacionado con la aplicación, como que una aplicación solicite que un evento negociado sea señalizado de inmediato para que el sistema operativo ponga en marcha un componente de segundo plano de inmediato. En otro ejemplo, un activador puede ser un evento sintético creado por un componente del sistema operativo (por ejemplo, un agente) al combinar el conocimiento de uno o más eventos de diferentes fuentes, una o más condiciones del sistema, y/o una o más directrices del sistema. Un ejemplo específico de tal evento sintético puede ser el “tiempo de mantenimiento”, que puede indicar el comienzo de un periodo de operación designado para el mantenimiento.

Aunque una aplicación puede especificar un activador para el evento negociado solicitado, en algunos casos se pueden especificar múltiples activadores. Por ejemplo, el agente puede tratar los activadores como alternativas, de modo que la activación de cualquier activador pueda provocar la señalización del evento negociado solicitado, siempre y cuando se cumplan todas las condiciones especificadas. Alternativamente, el agente puede tratar algunos o todos los activadores como conjuntos, o puede permitir que una aplicación especifique cualquier combinación lógica adecuada de activadores. En otras realizaciones adicionales, se pueden proporcionar múltiples agentes, cada uno de los cuales responde a un único activador, y pueden implementar colectivamente una combinación de múltiples activadores.

En la etapa 520, el agente puede almacenar uno o más activadores identificados en la etapa 515 en asociación con el ID del evento negociado asignado en la etapa 510. Los activadores pueden almacenarse de cualquier manera adecuada. En algunas realizaciones, cada activador puede almacenarse como una configuración de activador y un tipo de activador. Como otro ejemplo, el agente puede almacenar los activadores y el ID del evento negociado en la estructura 600 de datos ilustrativa mostrada en la FIG. 6. En esta realización, la estructura 600 de datos puede incluir varias entradas, cada una correspondiente a un evento negociado solicitado y que tiene al menos tres campos: el campo 605 para almacenar los ID de eventos negociados, el campo 610 para almacenar activadores y el campo 615 para condiciones de almacenamiento. Por ejemplo, en la primera entrada ilustrativa, un ID de evento negociado “BE_1” se asocia con varios activadores alternativos “$E_{1,1}$”, “$E_{1,2}$”, … y una única condición “$C_{1,1}$”. Aunque no se muestran, las entradas 600 en la estructura de datos pueden tener campos adicionales; por ejemplo, para almacenar otros elementos de información que pueden permitir que el agente aplique una o más directrices de administración de recursos.

Volviendo a la FIG. 5, en la etapa 525, el agente puede determinar si se especifican una o más condiciones que han de satisfacerse cuando se señala el evento negociado solicitado. Si se especifican una o más condiciones, el agente puede proceder a identificar las condiciones en la etapa 530. Si no, el agente puede omitir las etapas 530 y 535 y proceder directamente a la etapa 540.

Según se ha expuesto anteriormente, las condiciones pueden ser especificadas por la aplicación que solicita el evento negociado; por ejemplo, para informar al sistema operativo de un conjunto de circunstancias bajo las cuales es probable que un componente de segundo plano asociado realice un trabajo útil. Por ejemplo, un componente de segundo plano para descargar contenido de Internet solo podrá realizar un trabajo útil cuando haya disponible una conexión a Internet. En consecuencia, la aplicación puede solicitar que el evento negociado se señalice únicamente cuando Internet esté disponible.

Otros ejemplos de condiciones incluyen, sin limitación, condiciones relacionadas con el usuario (por ejemplo, que el usuario esté activo o inactivo), condiciones relacionadas con el sistema (por ejemplo, que la red esté disponible/no disponible), condiciones relacionadas con la aplicación (por ejemplo, que la aplicación haya sido utilizada activamente por el usuario recientemente en el pasado —por ejemplo, según lo determinado por una cierta cantidad de tiempo

umbral—, que la aplicación no haya superado cierta asignación de recursos otorgada por el sistema operativo, etc.), y similares.

Una aplicación puede especificar un número cualquiera de condiciones, ya que los aspectos de la presente divulgación no están limitados a un número particular de condiciones que se especifiquen para un evento negociado. Por ejemplo, una aplicación puede especificar una o más condiciones, o ninguna condición en absoluto. El agente puede tratar las condiciones como si fueran conjuntas, de modo que el evento negociado solicitado no se señalice a menos que se satisfagan todas las condiciones especificadas, aunque se haya desencadenado el activador correspondiente. Alternativamente, el agente puede tratar las condiciones como alternativas, de modo que el evento negociado solicitado sea señalizado si se satisface alguna de las condiciones especificadas cuando se desencadene un activador correspondiente. En algunas realizaciones adicionales, el agente puede permitir que la aplicación especifique cualquier combinación lógica adecuada de condiciones.

Alternativamente, o adicionalmente, un sistema operativo o una aplicación (por ejemplo, una miniaplicación) distinta de la aplicación que solicita el evento negociado también puede especificar una o más condiciones para asociarse con un evento negociado. Por ejemplo, en algunas realizaciones, un agente puede implementar una interfaz para que sea invocada por un sistema operativo o una aplicación para especificar y/o modificar las condiciones asociadas con un evento negociado; por ejemplo, proporcionando el ID del evento negociado asignado en la etapa 510.

Si se identifican una o más condiciones en las etapas 525 y 530, el agente puede almacenar, en la etapa 535, las condiciones identificadas en asociación con el ID del evento negociado asignado en la etapa 510. Por ejemplo, el agente puede almacenar las condiciones en la estructura 600 de datos ilustrativa mostrada en la FIG. 6 y explicada anteriormente.

En la etapa 540, el agente puede devolver el ID del evento negociado asignado en la etapa 510 a la aplicación que solicita el evento negociado, de modo que la aplicación pueda invocar otra funcionalidad del sistema operativo para asociar el evento negociado con un componente deseado de segundo plano. Como se expone con mayor detalle en relación con la FIG. 9, la aplicación puede, en algunas realizaciones, dar de alta en una infraestructura de agentes (o en cualquier otro componente adecuado del sistema operativo) el componente de segundo plano para su ejecución, pudiendo aquella ejecutar el componente de segundo plano en respuesta a un agente que señaliza un ID de evento negociado identificado por el ID del evento negociado.

En la etapa 545, el agente puede especificar a la infraestructura de agentes (o a cualquier otro destinatario deseado de eventos negociados señalizados) cualquier restricción adicional asociada con el evento negociado solicitado. Por ejemplo, en algunas realizaciones, el agente puede ser un componente especializado para la gestión de un tipo particular de recursos, y puede transmitir a la infraestructura de agentes cualquier directriz especializada que haya de aplicarse en la gestión de los eventos negociados relacionados con ese tipo particular de recursos. Esta información se puede transmitir de cualquier manera adecuada; por ejemplo, a través de la comunicación directa entre el agente y la infraestructura de agentes utilizando el ID del evento negociado asignado en la etapa 510, o proporcionando la información a la solicitud que ha de ser enviada a la infraestructura de agentes junto con el ID del evento negociado asignado en la etapa 510.

Ejemplos de restricciones adecuadas que pueden ser especificadas por el agente incluyen, sin limitación, requerir que la infraestructura de agentes notifique al agente antes de y/o después de ejecutar uno o más componentes de segundo plano en respuesta al evento negociado solicitado, exención para cualquier componente de segundo plano asociado con el evento negociado solicitado de las reglas de regulación de la infraestructura de agentes, una fecha límite en la cual uno o más componentes de segundo plano hayan de ser ejecutados y/o completarse después de que se señalice el evento negociado solicitado, ejecutar todos los componentes de segundo plano asociados con eventos negociados de este agente en el mismo proceso y finalizar ese proceso solo como último recurso, serializar los componentes de segundo plano asociados con eventos negociados de este agente, hacer que un evento negociado señalizado sobreviva a un reinicio del sistema y/o notificar al agente después del reinicio del sistema del evento negociado señalizado persistente, y similares.

Aunque anteriormente se han expuesto ejemplos específicos de activadores, condiciones y restricciones en relación con la FIG. 5, debería apreciarse que los aspectos de la presente divulgación no están limitados a ninguna combinación particular de activadores, condiciones y/o restricciones especificados por una aplicación y/o sistema operativo. En algunas realizaciones, los activadores, las condiciones, y/o las restricciones se pueden representar como programas generales que han de ser interpretados por un componente del sistema operativo (por ejemplo, un agente) para determinar cuándo ejecutar un elemento de trabajo de segundo plano. Además, aunque se describen realizaciones ilustrativas en las que una infraestructura de agentes y agentes individuales forman una estructura de dos niveles, otras realizaciones pueden incluir una jerarquía de múltiples niveles de agentes, pudiendo un agente de un nivel combinar eventos de uno o más agentes en niveles inferiores.

La FIG. 7 muestra un proceso ilustrativo 700 que puede ser realizado por un componente agente para determinar cuándo señalizar un evento negociado según algunas realizaciones. Por ejemplo, el proceso 700 puede ser realizado por uno de los agentes 415A-D mostrados en la FIG. 4.

En la etapa 705, el agente puede detectar el desencadenamiento de un activador. Como se mencionó anteriormente, esto puede incluir la recepción de un evento del sistema operativo que indique que algo acaba de suceder, como que un usuario inicie o cierre la sesión, que la red esté disponible o no esté disponible, que expire un temporizador y similares.

En respuesta a la detección del desencadenamiento del activador, el agente puede, en la etapa 707, identificar uno o más eventos negociados asociados que han de ser señalizados. Esto se puede hacer, por ejemplo, accediendo a una estructura de datos como la mostrada en la FIG. 6. Por ejemplo, el agente puede identificar una o más entradas en la estructura 600 de datos que tienen el activador desencadenado almacenado en el campo “Activador(es)” 610. Si se identifican varias de estas entradas, el agente puede procesar las entradas una por una.

Para cada entrada que coincida con el activador desencadenado, el agente puede, en la etapa 710, determinar si se especifican condiciones para el correspondiente evento negociado y, si es así, si se satisfacen todas esas condiciones. En algunas realizaciones, esto puede hacerse verificando el correspondiente campo “Condición(es)” 615 mostrado en la FIG. 6 y evaluando todas las condiciones almacenadas. En realizaciones alternativas, el agente puede mantener, para cada condición almacenada, una variable booleana indicativa de si la condición almacenada está satisfecha en ese momento, y cualquier variable auxiliar para evaluar la condición. El agente puede actualizar continuamente la variable booleana; por ejemplo, en respuesta a cambios en las variables auxiliares. De esta manera, el agente puede simplemente verificar la variable booleana en la etapa 710, sin tener que evaluar la condición nuevamente.

Si se satisfacen todas las condiciones asociadas, o si no se almacenan condiciones en el campo 615, el agente puede proceder a la etapa 715. De lo contrario, el agente puede bloquear el activador en la etapa 720 y continuar monitorizando las condiciones asociadas. En una realización en la que el agente mantiene una variable booleana para cada condición almacenada, monitorizar las condiciones puede incluir simplemente actualizar continuamente las variables booleanas. Si y cuando se detecta un cambio para una o más de las condiciones asociadas en la etapa 725, el agente puede volver a la etapa 710 para verificar si se cumplen todas las condiciones asociadas. El agente puede hacer un bucle alrededor de las etapas 710, 720 y 725 de forma indefinida, a no ser que se asocie un plazo con el activador bloqueado, en cuyo caso el agente puede descartar el activador bloqueado cuando expire el plazo.

Si en algún momento se satisfacen todas las condiciones asociadas, el agente puede, en la etapa 715, verificar si alguna limitación del agente es aplicable al evento negociado y, de ser así, si se satisfacen las limitaciones del agente. Como se mencionó anteriormente, el agente puede imponer una o más directrices de gestión de recursos, y puede abstenerse de señalizar un evento negociado incluso después de que se haya desencadenado un activar y se satisfagan todas las condiciones especificadas por la aplicación, como una forma de controlar el consumo de recursos mediante una aplicación. Las limitaciones del agente pueden ser un mecanismo para aplicar estas directrices de gestión de recursos. En un ejemplo, se puede establecer una limitación en la frecuencia con la que una sola aplicación puede hacer uso de los recursos gestionados por el agente (por ejemplo, en términos de la cantidad de veces por día o algún otro periodo de tiempo adecuado, o de la cantidad de recursos gestionados que ya ha sido consumida por las actividades en segundo plano de la aplicación, o el sistema en su conjunto, dentro del periodo de tiempo). También se pueden especificar otros tipos de limitaciones de los agentes, ya que los aspectos de la presente divulgación no están limitados a ningún tipo particular de limitaciones de agentes.

Si se satisfacen todas las limitaciones aplicables del agente, el agente puede señalizar el evento negociado en la etapa 720; si no, el agente simplemente puede descartar el activador desencadenado. En algunas realizaciones, el agente puede solicitar que el sistema operativo haga notar que un evento negociado no fue señalizado porque no se satisficieron ciertas condiciones y/o limitaciones. El sistema operativo puede a su vez notificar a la aplicación que solicitó el evento negociado (por ejemplo, a través de una función API, como la interfaz de “Estado SP” del componente 105 de primer plano mostrado en la Figura 1), aunque no se requiere tal notificación.

Aunque anteriormente se han descrito ejemplos específicos de agentes y funcionalidades de agentes en relación con las FIGURAS 5-7, debería apreciarse que los aspectos de la presente divulgación no están limitados al uso de un marco que tenga una infraestructura de agentes y a varios agentes, ni a una división específica de responsabilidades entre una infraestructura de agentes y los agentes individuales. Por ejemplo, en realizaciones alternativas, cualquier subconjunto adecuado de las funcionalidades de agente descritas en este documento puede ser implementado por una infraestructura de agentes. En un ejemplo, una infraestructura de agentes puede realizar un seguimiento de las asociaciones entre eventos negociados y las aplicaciones que solicitaron los eventos de agentes, de modo que las aplicaciones y/o los agentes puedan obtener esa información de la infraestructura de agentes. Esto se puede hacer como una forma de hacer dicha información persistente. En otro ejemplo, una infraestructura de agentes puede estar programada para bloquear un activador y/o monitorizar las condiciones asociadas para determinar cuándo se debe señalizar un evento negociado. De esta manera, una aplicación puede solicitar un evento negociado para el cual los agentes implementan algunos activadores y/o condiciones, ya que la infraestructura de agentes puede gestionar información de estado de múltiples agentes individuales. Además, puede facilitarse el desarrollo modular en una implementación en la que la infraestructura de agentes gestiona la monitorización de las condiciones, puesto que ya no es preciso que los agentes individuales “entiendan” o aborden las condiciones.

La FIG. 8A muestra un proceso ilustrativo 800 que puede realizar una infraestructura de agentes (BI), o algún otro componente adecuado del sistema operativo, para asociar un componente de segundo plano con un evento negociado según algunas realizaciones. La FIG. 8B muestra una estructura 850 de datos ilustrativa que se puede usar en relación con el proceso 800. Una asociación creada por el proceso 800 puede permitir que el sistema operativo cargue y ejecute el componente negociado en el evento negociado que se está señalizando. En un ejemplo, el proceso 800 puede ser efectuado por la infraestructura 405 de agentes mostrada en la FIG. 4. Sin embargo, los aspectos de la presente divulgación no están limitados a que un componente del sistema operativo en particular realice un proceso de asociación. En algunas realizaciones alternativas, un agente responsable de señalar el evento negociado puede asociar un componente de segundo plano con un evento negociado; por ejemplo, como parte de un proceso para establecer el evento negociado en respuesta a una solicitud de una aplicación.

En la etapa 805, la BI puede recibir una solicitud para asociar un componente de segundo plano, tal como un elemento de trabajo, con un evento negociado. La solicitud puede ser recibida de cualquier entidad adecuada, ya que los aspectos de la presente divulgación no están así limitados. Por ejemplo, en algunas realizaciones, una solicitud puede presentar dicha solicitud para que se ejecute el componente de segundo plano para que haga un trabajo de segundo plano para la aplicación. En algunas otras realizaciones, un agente responsable de señalar el evento negociado puede enviar dicha solicitud en nombre de una aplicación; por ejemplo, cuando la aplicación solicita el evento negociado al agente o cuando la aplicación se instala o es actualizada.

En la etapa 810, la BI puede almacenar un ID de componente de segundo plano en asociación con un ID de evento negociado; por ejemplo, en la estructura 850 de datos ilustrativa mostrada en la FIG. 8B. En esta realización, la estructura 850 de datos puede incluir varias entradas, cada una correspondiente a un evento negociado y que tiene al menos dos campos: el campo 855 para almacenar ID de eventos negociados y el campo 860 para almacenar varios ID de componentes de segundo plano. Por ejemplo, en la primera entrada ilustrativa, un ID de evento negociado "BE_1" está asociado con un ID de componente de segundo plano "$ObjectActivationClass_1$", que puede identificar un objeto de soporte lógico que implementa el componente de segundo plano. Aunque no se muestran, las entradas en la estructura 850 de datos pueden tener campos adicionales; por ejemplo, para almacenar información que indica una ubicación de almacenamiento desde la cual se puede cargar el componente de segundo plano y/o información que puede permitir que la BI aplique una o más directrices de gestión de recursos al gestionar el componente de segundo plano. En un ejemplo, la BI puede almacenar una identificación de una aplicación solicitando que se ejecute el componente de segundo plano, de modo que la BI pueda imponer directrices aplicables a esa aplicación al gestionar el componente de segundo plano. En otro ejemplo, la BI puede almacenar las restricciones especificadas por el agente responsable de señalizar el evento negociado; por ejemplo, como se explicó anteriormente en conexión con la etapa 545 de la FIG. 5.

La FIG. 9 muestra un proceso ilustrativo 900 que puede ser realizado por un componente de una infraestructura de agentes (BI), o algún otro componente adecuado del sistema operativo, para determinar cuándo ejecutar un componente de segundo plano y cómo gestionar la ejecución del componente de segundo plano según algunas realizaciones. Por ejemplo, el proceso 900 puede ser realizado por el componente 405 de BI mostrado en la FIG. 4.

En la etapa 905, la BI puede detectar un evento negociado señalizado por un agente. En respuesta a la detección del evento negociado señalizado, la BI puede, en la etapa 910, identificar uno o más componentes asociados de segundo plano que han de ser ejecutados. Esto se puede hacer, por ejemplo, accediendo a una estructura de datos como la mostrada en la FIG. 8B. Por ejemplo, la BI puede identificar una o más entradas en la estructura 850 de datos que tienen el evento negociado señalizado almacenado en el campo "ID de evento negociado" 855. Si se identifican varias de estas entradas, la BI puede procesar las entradas una por una.

Para cada entrada que coincida con el evento negociado señalado, la BI puede, en la etapa 915, verificar si son aplicables y se satisfacen las restricciones y/o limitaciones. Como se expuso anteriormente en relación con la etapa 545 de la FIG. 5, una restricción puede ser una regla de gestión de recursos especializada transmitida por un agente al BI para ayudar a la BI a gestionar los recursos asociados con el evento negociado señalizado. Una limitación, por otro lado, puede capturar cualquier directriz de gestión de recursos aplicada por la BI. Estas directrices pueden superponerse, aunque no es preciso que lo hagan, a las aplicadas por agentes individuales. En algunas implementaciones, una BI puede ser un componente general del sistema operativo (a diferencia de un agente, que puede estar especialmente adaptado para gestionar un tipo específico de recursos), y puede ser capaz de aplicar limitaciones agregadas que se relacionan con múltiples aspectos de la gestión de recursos.

En un ejemplo, una BI puede imponer un límite en la cantidad de tiempo de procesador que se puede asignar a la ejecución del trabajo de segundo plano; por ejemplo, en términos de un porcentaje del tiempo del procesador o una cantidad específica de tiempo del procesador por día o algún otro periodo de tiempo adecuado. Esta limitación se puede imponer en todo momento, o solo en ciertas circunstancias, como cuando el sistema funciona con batería. En otro ejemplo, una BI puede asignar un porcentaje de una previsión de ejecución en segundo plano a una aplicación en función de la cantidad de "tiempo de contacto cara a cara" que la aplicación obtiene con un usuario, es decir, la frecuencia con que el usuario utiliza la aplicación en relación con otras aplicaciones (por ejemplo, en términos de un porcentaje del tiempo real empleado por el usuario que interactúa con la aplicación, o un porcentaje del tiempo del procesador utilizado por un componente de primer plano de la aplicación). En otro ejemplo adicional, una BI puede

evitar que se ejecute un componente de segundo plano si un usuario no ha ejecutado ningún componente de primer plano asociado durante un tiempo prolongado (por ejemplo, durante un periodo de tiempo umbral seleccionado, como un mes, dos meses, tres meses, etc.).

Si se satisfacen todas las restricciones y las limitaciones aplicables de BI, la BI puede proceder a la etapa 920 para cargar y ejecutar el componente de segundo plano. Si no, la BI puede proceder a la etapa 935 para informar de un fallo a un correspondiente componente de primer plano. En algunas realizaciones, el informe puede identificar una o más razones del error, por lo que el componente de primer plano puede tomar las medidas apropiadas, como pedir a un usuario que apruebe una exención a una restricción o una limitación violada de BI.

Si el BI determina en la etapa 915 que se satisfacen todas las restricciones estáticas aplicables y las limitaciones de BI e inicia el componente de segundo plano en la etapa 920, la BI puede supervisar, en la etapa 925, si se satisfacen todas las restricciones dinámicas aplicables y las limitaciones de BI mientras se ejecuta el componente de segundo plano. Ejemplos de limitaciones dinámicas de BI incluyen, sin limitación, una limitación de la cantidad de tiempo de procesador que un componente de segundo plano puede usar en relación con otros componentes de segundo plano (por ejemplo, en términos de un porcentaje de la previsión de tiempo del procesador asignado para la ejecución en segundo plano), y una velocidad mínima a la que un componente de segundo plano ha de avanzar hacia su finalización.

Si se viola cualquier restricción dinámica o limitación de BI durante la ejecución, el componente de segundo plano puede detenerse en la etapa 930, y la BI puede proceder a la etapa 935 para informar de un error en la finalización de la ejecución del componente de segundo plano. Nuevamente, en algunas realizaciones, el informe puede identificar las razones del fallo, por lo que el componente de primer plano puede adoptar las medidas apropiadas. Si no se infringen las restricciones dinámicas o las limitaciones de BI durante la ejecución, el componente de segundo plano puede completarse y la BI puede proceder a la etapa 935 para informar de una finalización satisfactoria del componente de segundo plano. Sin embargo, debe apreciarse que diversas técnicas descritas en este documento para gestionar los componentes de segundo plano pueden ser usadas para gestionar aplicaciones sin componentes de primer plano, como los servicios en segundo plano que se ejecutan en cualquier tipo de dispositivos informáticos.

Debería apreciarse que diversos modelos de ejecución negociada descritos anteriormente en relación con las FIGURAS 4-9 pueden, en algunas realizaciones, ser aplicados por un sistema operativo para gestionar la ejecución del trabajo de segundo plano, pero no la ejecución del trabajo de primer plano, lo que resulta en una gestión diferenciada del trabajo de primer plano y de segundo plano. Sin embargo, en realizaciones alternativas, la ejecución negociada también puede ser usada para gestionar el trabajo de primer plano; por ejemplo, para controlar el consumo de recursos. Por ejemplo, la ejecución de trabajo de primer plano puede ser causada por eventos negociados señalizados en respuesta a las indicaciones del usuario. En tal realización, sin embargo, un usuario puede percibir más lentitud debido a una o más capas de mecanismos de gestión de recursos que se aplican en un modelo de ejecución negociada. Por lo tanto, al decidir si usar la ejecución negociada para el trabajo de primer plano, se puede hacer una compensación entre la conservación de recursos y el mantenimiento de la capacidad de respuesta de los componentes de primer plano activos.

La FIG. 10 muestra, esquemáticamente, un ordenador ilustrativo 1000 en el que pueden implementarse diversos aspectos inventivos de la presente divulgación. Por ejemplo, el ordenador 1000 puede ser un dispositivo móvil en el que se puede implementar cualquiera de las características descritas en este documento.

Como se usa en el presente documento, un “dispositivo móvil” puede ser cualquier dispositivo informático que sea suficientemente pequeño para que pueda ser llevado por un usuario (por ejemplo, sostenido en la mano del usuario). Ejemplos de dispositivos móviles incluyen, entre otros, teléfonos móviles, buscapersonas, reproductores multimedia portátiles, lectores de libros electrónicos, consolas portátiles de juegos, agendas electrónicas (PDA) y tabletas. En algunos casos, el peso de un dispositivo móvil puede ser como máximo de 450 gramos, 680 gramos, o 900 gramos, y/o la dimensión más grande de un dispositivo móvil puede ser como máximo de quince centímetros, veintitrés centímetros o treinta centímetros. Además, un dispositivo móvil puede incluir características que permitan al usuario utilizar el dispositivo en diversas ubicaciones. Por ejemplo, un dispositivo móvil puede incluir un almacenamiento de energía (por ejemplo, una batería) de modo que se pueda usar por algún tiempo sin estar conectado a una toma de corriente. Como otro ejemplo, un dispositivo móvil puede incluir una interfaz de red inalámbrica configurada para proporcionar una conexión de red sin estar físicamente conectado a un punto de conexión de red.

En el ejemplo mostrado en la FIG. 10, el ordenador 1000 incluye un procesador o una unidad 1001 de procesamiento y una memoria 1002 que puede incluir una memoria volátil y/o no volátil. El ordenador 1000 también puede incluir almacenamiento 1005 (por ejemplo, una o más unidades de disco), además de la memoria 1002 del sistema. La memoria 1002 puede almacenar una o más instrucciones para programar la unidad 1001 de procesamiento para que realice cualquiera de las funciones descritas en este documento. La memoria 1002 también puede almacenar uno o más programas de aplicación y/o funciones de la interfaz de programación de aplicaciones (API).

El ordenador 1000 puede tener uno o más dispositivos de entrada y/o dispositivos de salida, tales como los dispositivos 1006 y 1007 ilustrados en la FIG. 10. Estos dispositivos pueden utilizarse, entre otras cosas, para presentar una interfaz de usuario. Ejemplos de dispositivos de salida que se pueden usar para proporcionar una interfaz de usuario

incluyen impresoras o pantallas de visualización para la presentación visual de la salida y altavoces u otros dispositivos generadores de sonido para la presentación audible de la salida. Ejemplos de dispositivos de entrada que se pueden usar para una interfaz de usuario incluyen teclados y dispositivos señaladores, como ratones, almohadillas táctiles y tabletas digitalizadoras. Como otro ejemplo, un ordenador puede recibir información de entrada a través del reconocimiento de voz o en otro formato audible.

Como se muestra en la FIG. 10, el ordenador 1000 también puede comprender una o más interfaces de red (por ejemplo, la interfaz 1010 de red) para permitir la comunicación a través de varias redes (por ejemplo, la red 1020). Ejemplos de redes incluyen una red de área local o una red de área amplia, como una red empresarial o Internet. Tales redes pueden basarse en cualquier tecnología adecuada y pueden operar de acuerdo con cualquier protocolo adecuado y pueden incluir redes inalámbricas, redes cableadas o redes de fibra óptica.

Habiendo así descrito varios aspectos de al menos una realización de esta invención, ha de apreciarse que a los expertos en la técnica se les ocurrirán fácilmente diversas alteraciones, modificaciones y mejoras.

Además, aunque se indican ventajas de la presente invención, debe apreciarse que no todas las realizaciones de la invención incluirán todas las ventajas descritas. Algunas realizaciones pueden no implementar ninguna de las características descritas como ventajosas en este documento y en algunos casos. Por consiguiente, la descripción y los dibujos anteriores son a título de ejemplo únicamente.

Las realizaciones de la presente invención descritas anteriormente pueden implementarse de cualquiera de numerosas maneras. Por ejemplo, las realizaciones pueden implementarse utilizando soporte físico, soporte lógico o una combinación de los mismos. Cuando se implementa en soporte lógico, el código del soporte lógico puede ejecutarse en cualquier procesador o colección de procesadores adecuados, ya sea que se proporcionen en un solo ordenador o se distribuyan entre varios ordenadores. Dichos procesadores pueden implementarse como circuitos integrados, con uno o más procesadores en un componente de circuito integrado. Sin embargo, un procesador puede implementarse utilizando circuitos en cualquier formato adecuado.

Además, debería apreciarse que un ordenador puede estar implementado en cualquiera de varias formas, tales como un ordenador montado en bastidor, un ordenador de sobremesa, un ordenador portátil o una tableta. Además, un ordenador puede estar integrado en un dispositivo que generalmente no se considera un ordenador pero con capacidades de procesamiento adecuadas, incluyendo una agenda electrónica (PDA), un teléfono inteligente, una caja registradora, un cajero automático (ATM) o cualquier otro dispositivo electrónico portátil o fijo adecuado.

Además, los diversos procedimientos o procesos esbozados en este documento pueden codificarse como soporte lógico que se puede ejecutar en uno o más procesadores que emplean cualquiera de diversos sistemas operativos o plataformas. Además, dicho soporte lógico puede escribirse utilizando cualquiera de una serie de lenguajes de programación adecuados y/o herramientas de programación, y también se puede compilar como un código de lenguaje de máquina ejecutable o código intermedio que se ejecuta en un marco o máquina virtual.

En este sentido, la invención puede realizarse como un soporte de almacenamiento legible por ordenador (o múltiples soportes legibles por ordenador) (por ejemplo, una memoria de ordenador, uno o más disquetes flexibles, discos compactos (CD), discos ópticos, discos de video digital (DVD), cintas magnéticas, memorias flash, configuraciones de circuitos en matrices de puertas programables *in situ* u otros dispositivos semiconductores u otro soporte tangible de almacenamiento de ordenador) codificados con uno o más programas que, cuando se ejecutan en uno o más ordenadores u otros procesadores, llevan a cabo procedimientos que implementan las diversas realizaciones de la invención expuestas anteriormente. Como es evidente por los ejemplos anteriores, un soporte de almacenamiento legible por ordenador puede retener información durante un tiempo suficiente para proporcionar instrucciones ejecutables por ordenador en una forma no transitoria. Tal medio o tales medios de almacenamiento legibles por ordenador pueden ser transportables, de manera que el programa o los programas almacenados en ellos se pueden cargar en uno o más ordenadores u otros procesadores diferentes para implementar diversos aspectos de la presente invención expuesta en lo que antecede. Según se usa en el presente documento, la expresión “soporte de almacenamiento legible por ordenador” abarca solo un soporte legible por ordenador que pueda considerarse una fabricación (es decir, un artículo fabricado) o una máquina. Alternativa o adicionalmente, la invención puede implementarse como un soporte legible por ordenador distinto de un soporte de almacenamiento legible por ordenador, tal como una señal que se propague.

Los términos “programa” o “soporte lógico” son usados en la presente memoria en un sentido genérico para referirse a cualquier tipo de código informático o conjunto de instrucciones ejecutables por ordenador que pueden emplearse para programar un ordenador u otro procesador para implementar diversos aspectos de la presente invención como se expuso anteriormente. Además, debería apreciarse que, según un aspecto de esta realización, uno o más programas de ordenador que, cuando se ejecutan, realizan los procedimientos de la presente invención no necesitan residir en un solo ordenador o procesador, sino que pueden distribuirse de manera modular entre varios ordenadores o procesadores diferentes para implementar diversos aspectos de la presente invención.

Las instrucciones ejecutables por ordenador pueden tener muchas formas, como módulos de programa, ejecutados por uno o más ordenadores u otros dispositivos. Generalmente, los módulos de programa incluyen rutinas, programas,

objetos, componentes, estructuras de datos, etc. que llevan a cabo tareas particulares o implementan tipos de datos abstractos particulares. Normalmente, la funcionalidad de los módulos del programa puede combinarse o distribuirse según se desee en diversas realizaciones.

Además, las estructuras de datos pueden almacenarse en soportes legibles por ordenador en cualquier forma adecuada. Para simplificar la ilustración, se puede mostrar que las estructuras de datos tienen campos que están relacionados a través de la ubicación en la estructura de datos. Tales relaciones también se pueden lograr asignando almacenamiento para los campos con ubicaciones en un soporte legible por ordenador que exprese la relación entre los campos. Sin embargo, se puede usar cualquier mecanismo adecuado para establecer una relación entre la información de los campos de una estructura de datos, incluso mediante el uso de punteros, etiquetas u otros mecanismos que establezcan la relación entre los elementos de datos.

Diversos aspectos de la presente invención pueden ser usados solos, en combinación o en diversas disposiciones no expuestas específicamente en las realizaciones descritas anteriormente y, por lo tanto, no están limitados en su aplicación a los detalles y la disposición de los componentes expuestos en la descripción anterior o ilustrados en los dibujos. Por ejemplo, los aspectos descritos en una realización pueden combinarse de cualquier manera con los aspectos descritos en otras realizaciones.

Además, la invención se puede implementar como un procedimiento, del cual se ha proporcionado un ejemplo. Las etapas llevadas a cabo como parte del procedimiento se pueden ordenar de cualquier manera adecuada. Por consiguiente, se pueden construir realizaciones en las que las etapas se realicen en un orden diferente al ilustrado, que puede incluir realizar algunas etapas simultáneamente, aunque se muestren como etapas secuenciales en realizaciones ilustrativas.

El uso de términos ordinales como “primero”, “segundo”, “tercero”, etc., en las reivindicaciones para modificar un elemento de reivindicación no implica por sí mismo ninguna prioridad, precedencia u orden de un elemento de reclamación sobre otro o el orden temporal en el que se realizan las etapas de un procedimiento, sino que se usan meramente como etiquetas para distinguir un elemento de reclamación que tiene un nombre determinado de otro elemento que tiene un mismo nombre (salvo por el uso del término ordinal) para distinguir los elementos de reivindicación.

Además, la fraseología y la terminología utilizadas en el presente documento tienen una finalidad descriptiva y no deben considerarse limitativas. El uso de “que incluye”, “que comprende” o “que tiene”, “que contiene”, “que implica”, y las variaciones de los mismos en el presente documento, pretende abarcar los elementos enumerados a continuación y sus equivalentes, así como elementos adicionales.

REIVINDICACIONES

1. Un procedimiento para ser usado por un sistema operativo configurado para cargar separadamente componentes de primer plano y componentes de segundo plano de una aplicación y para su ejecución en al menos un ordenador, comprendiendo el procedimiento las etapas de:

detectar (705), en un componente agente del sistema operativo, una incidencia de al menos un activador asociado con un evento negociado, siendo el evento negociado un evento en respuesta al cual ha de ejecutarse al menos un componente de segundo plano de la aplicación;
en respuesta a la detección de la incidencia del al menos un activador, determinar (710), en el componente agente del sistema operativo, si se satisface al menos una condición operativa especificada por la aplicación o por el sistema operativo;
caracterizándose el procedimiento **porque**, además, comprende:

señalizar (730), por el componente agente del sistema operativo, el evento negociado únicamente cuando se determina que se satisface al menos una condición operativa; y
ejecutar el al menos un componente de segundo plano de la aplicación únicamente en respuesta al evento negociado señalizado por el componente agente del sistema operativo, estando asociado por el sistema operativo el al menos un componente de segundo plano con la al menos una condición operativa.

2. El procedimiento de la reivindicación 1 que, además, comprende las etapas de:

identificar al menos un componente de primer plano de la aplicación;
y gestionar la ejecución del componente de segundo plano de forma diferente a la ejecución del al menos un componente de primer plano, comprendiendo la ejecución del al menos un componente de primer plano con independencia de la al menos una condición operativa.

3. El procedimiento de la reivindicación 2 en el que la etapa de gestión de la ejecución del componente de segundo plano de forma diferente a la ejecución del al menos un componente de primer plano, además, comprende:

determinar si un usuario está interactuando de manera activa con la aplicación;
ejecutar el al menos un componente de primer plano únicamente cuando se determina que el usuario está interactuando de manera activa con la aplicación; y
ejecutar el manos un componente de segundo plano con independencia de si el usuario está interactuando de manera activa con la aplicación.

4. El procedimiento de la reivindicación 1 que, además, comprende las etapas de:

determinar, por el sistema operativo, si se satisface al menos una primer condición operativa establecida por la aplicación y si se satisface al menos una segunda condición operativa establecida por el sistema operativo; y
ejecutar el al menos un componente de segundo plano cuando se determina que se satisfacen las al menos una primera condición operativa y una segunda condición operativa.

5. El procedimiento de la reivindicación 4 que, además, comprende las etapas de:

mantener, por el componente agente del sistema operativo, información de estado para ser usada en la evaluación de la al menos una segunda condición operativa;
retener, por el componente agente del sistema operativo, el al menos un activador en respuesta a la determinación de que la al menos una segunda condición operativa no es satisfecha cuando se desencadena el al menos un activador;
reevaluar, por el componente agente del sistema operativo, la al menos una segunda condición operativa en respuesta a un cambio en la información de estado mantenida para determinar si debe señalizarse el evento negociado.

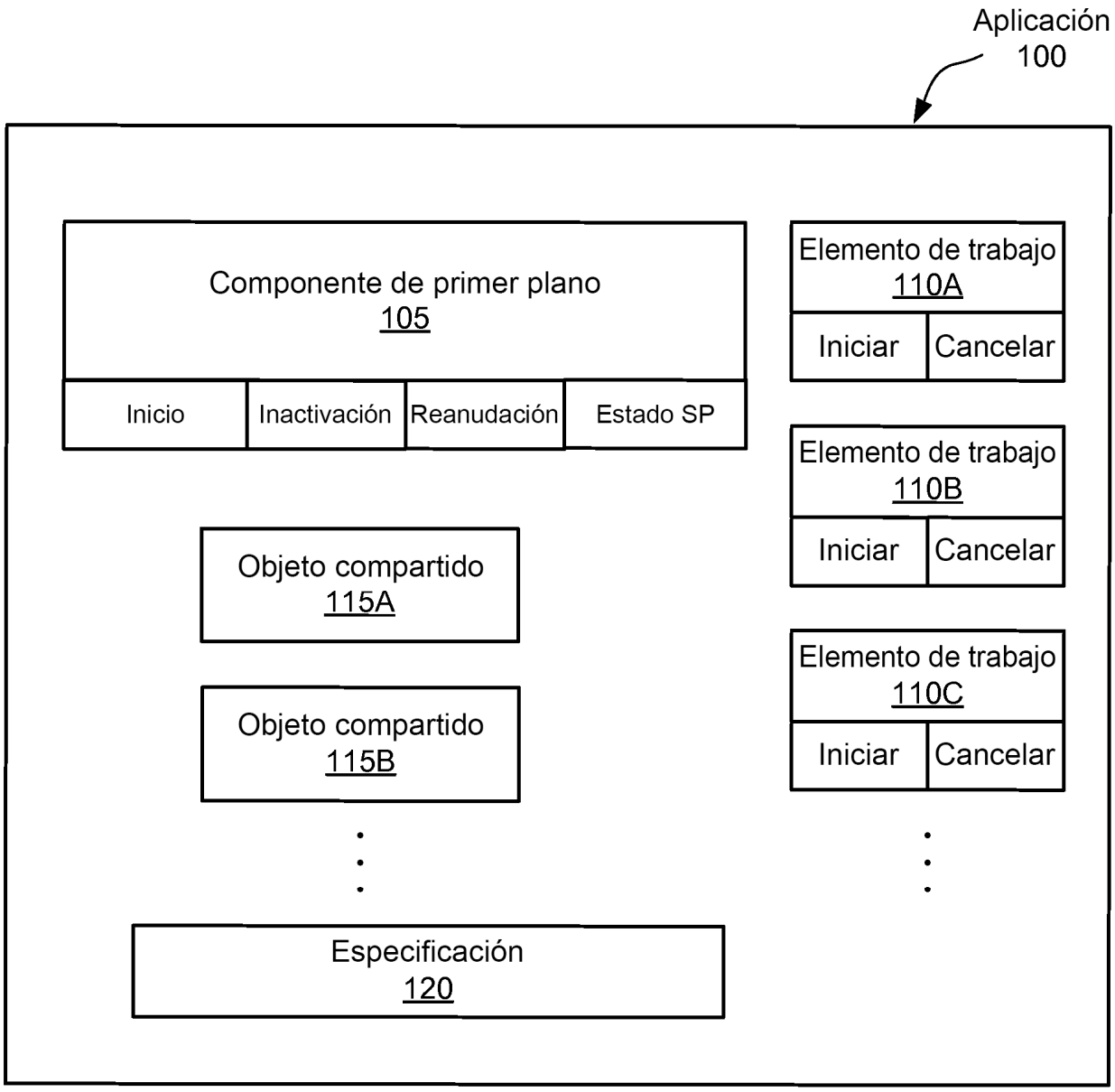
Aplicación
100
Componente de primer plano
105
Inicio
Inactivación
Reanudación
Estado SP
Objeto compartido
115A
Objeto compartido
115B
Especificación
120
Elemento de trabajo
110A
Iniciar
Cancelar
Elemento de trabajo
110B
Iniciar
Cancelar
Elemento de trabajo
110C
Iniciar
Cancelar

FIG. 1

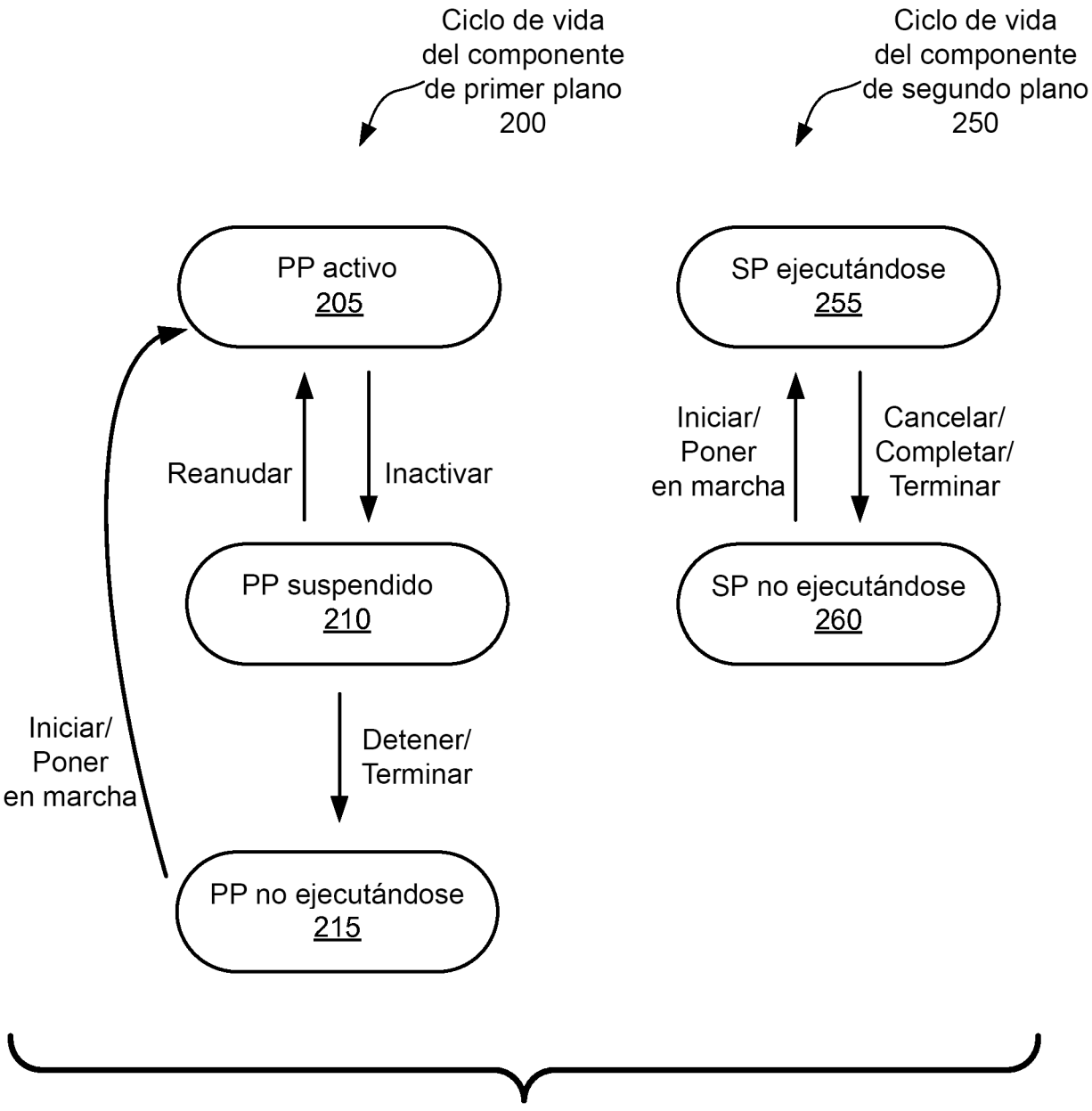
Ciclo de vida del componente de primer plano 200
Ciclo de vida del componente de segundo plano 250
PP activo 205
SP ejecutándose 255
Reanudar
Inactivar
Iniciar/ Poner en marcha
Cancelar/ Completar/ Terminar
PP suspendido 210
SP no ejecutándose 260
Detener/ Terminar
Iniciar/ Poner en marcha
PP no ejecutándose 215

FIG. 2

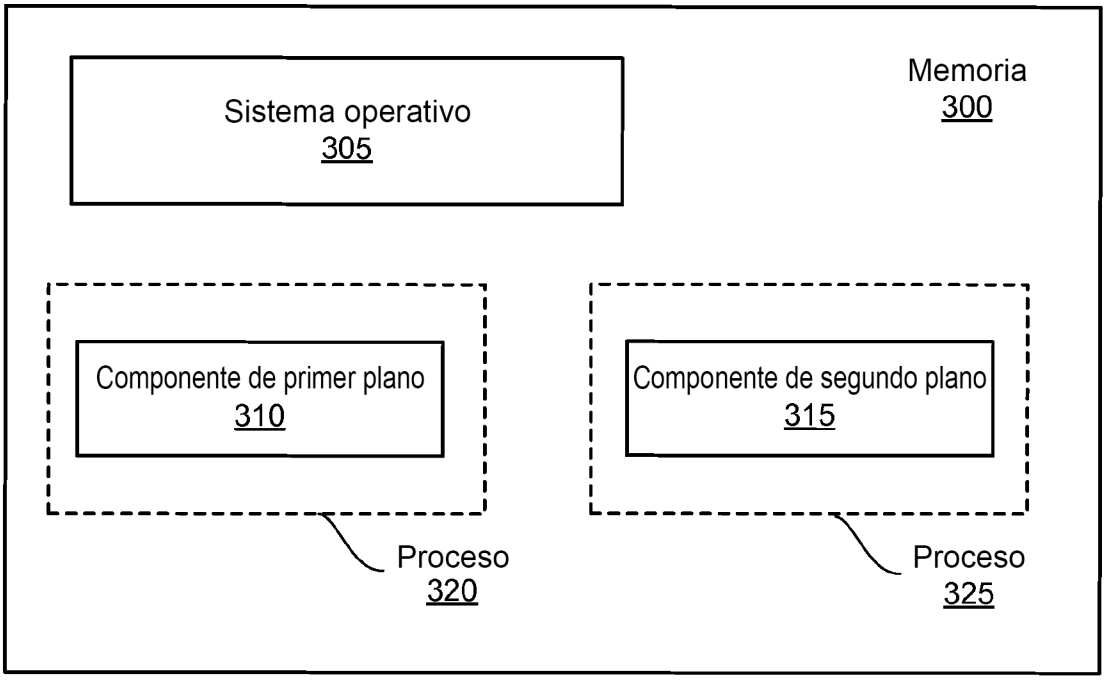
Memoria
300
Sistema operativo
305
Componente de primer plano
310
Componente de segundo plano
315
Proceso
320
Proceso
325

FIG. 3A

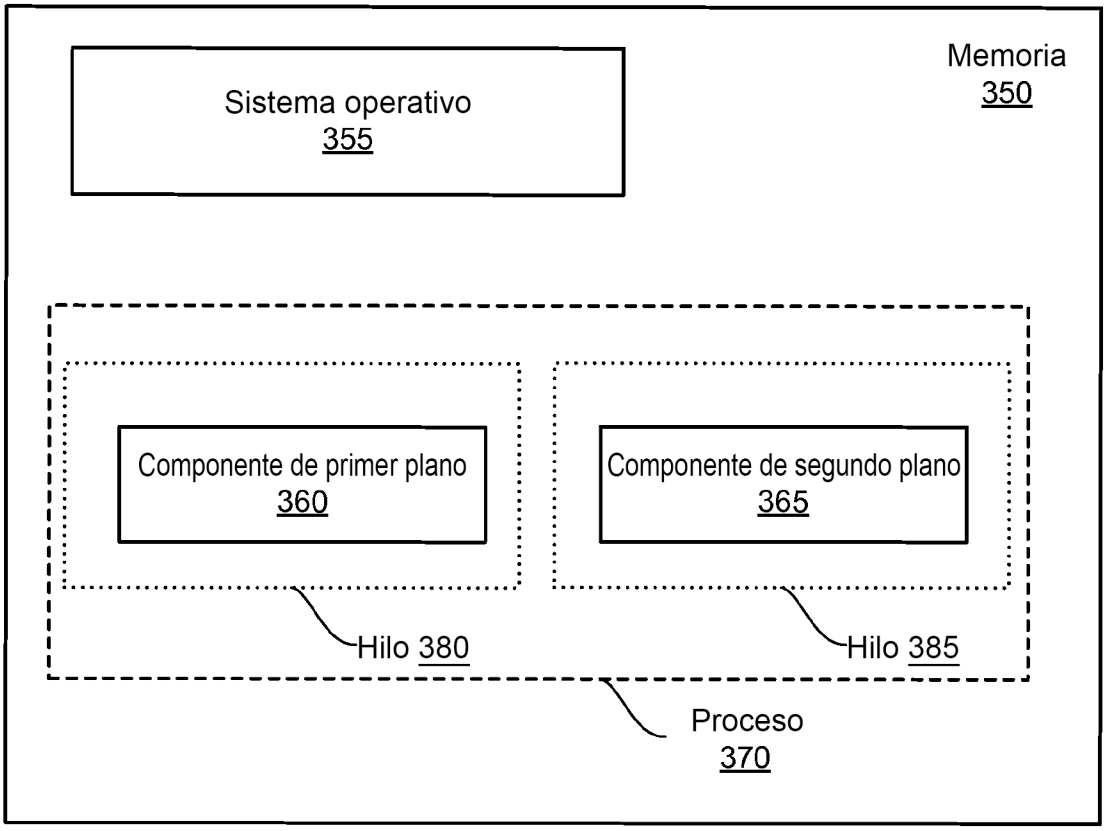
Memoria
350
Sistema operativo
355
Componente de primer plano
360
Componente de segundo plano
365
Hilo 380
Hilo 385
Proceso
370

FIG. 3B

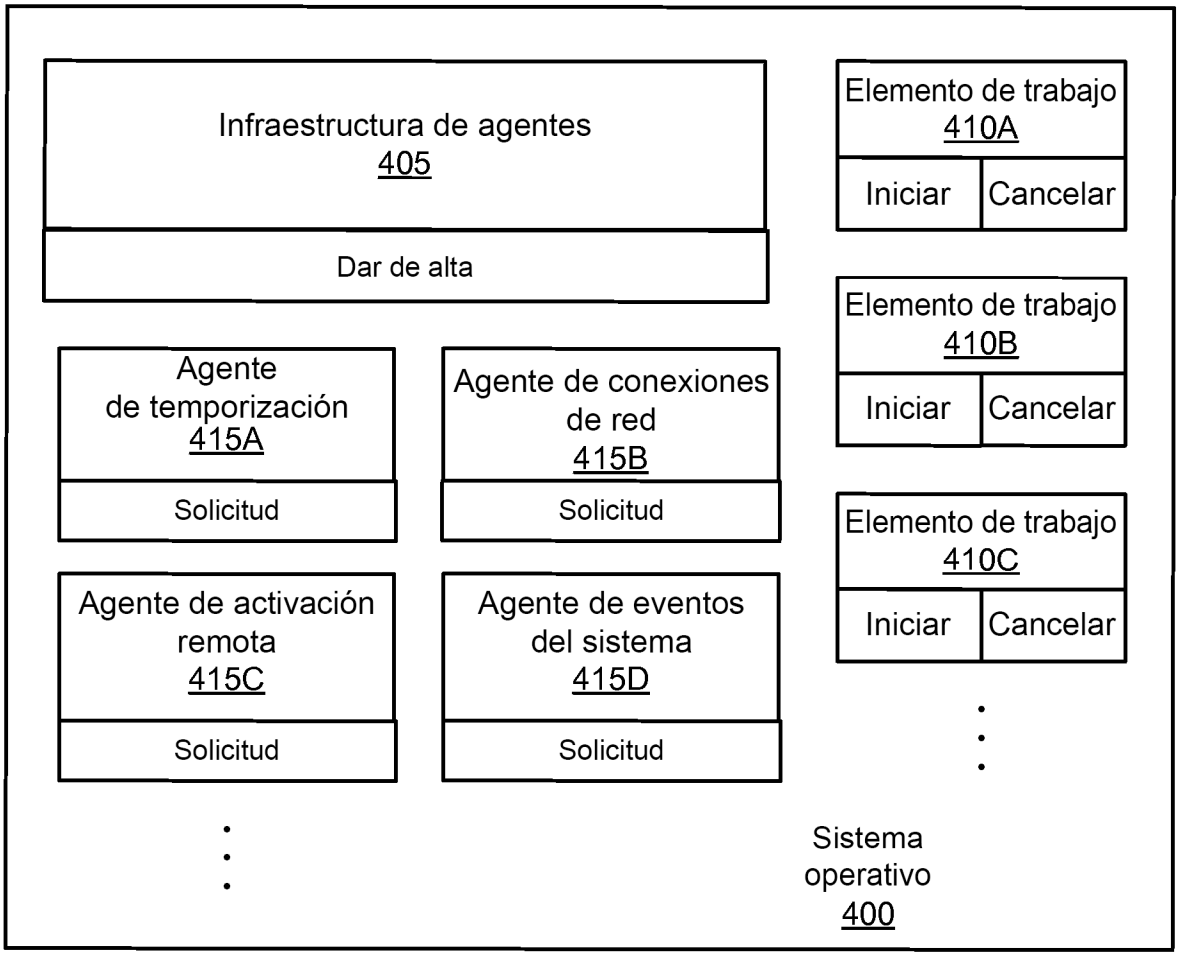
Infraestructura de agentes
405
Dar de alta
Agente de temporización
415A
Solicitud
Agente de conexiones de red
415B
Solicitud
Agente de activación remota
415C
Solicitud
Agente de eventos del sistema
415D
Solicitud
Elemento de trabajo
410A
Iniciar
Cancelar
Elemento de trabajo
410B
Iniciar
Cancelar
Elemento de trabajo
410C
Iniciar
Cancelar
Sistema operativo
400

FIG. 4

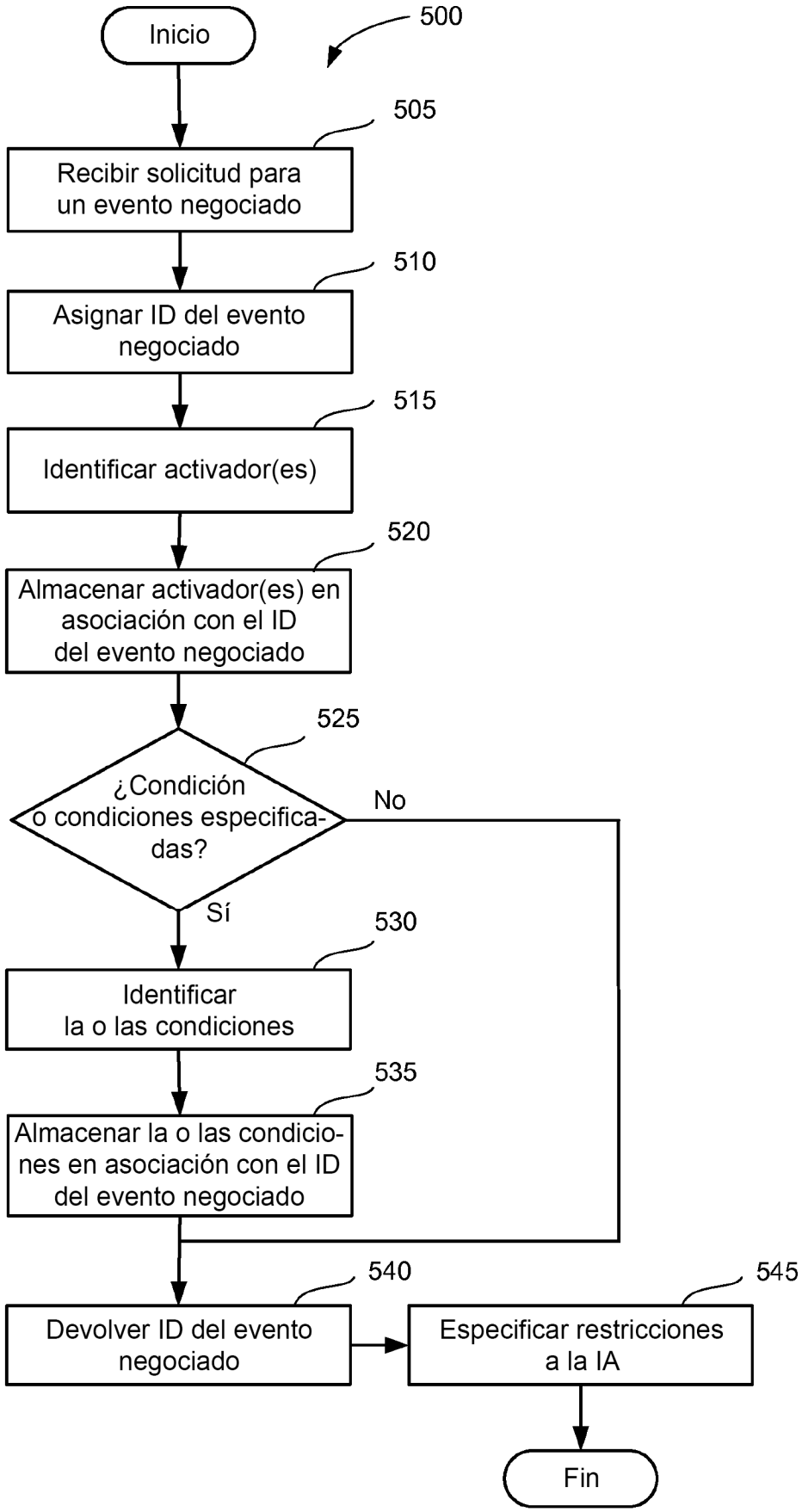
Inicio
500
505
Recibir solicitud para un evento negociado
510
Asignar ID del evento negociado
515
Identificar activador(es)
520
Almacenar activador(es) en asociación con el ID del evento negociado
525
¿Condición o condiciones especificadas?
No
Sí
530
Identificar la o las condiciones
535
Almacenar la o las condiciones en asociación con el ID del evento negociado
540
Devolver ID del evento negociado
545
Especificar restricciones a la IA
Fin

FIG. 5
Agente

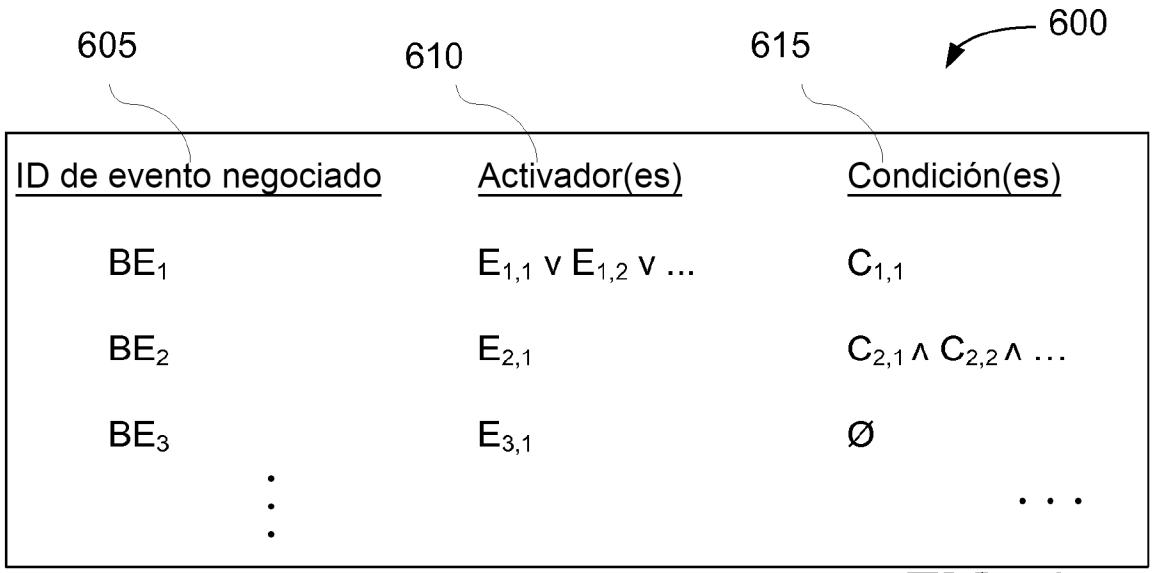

FIG. 6

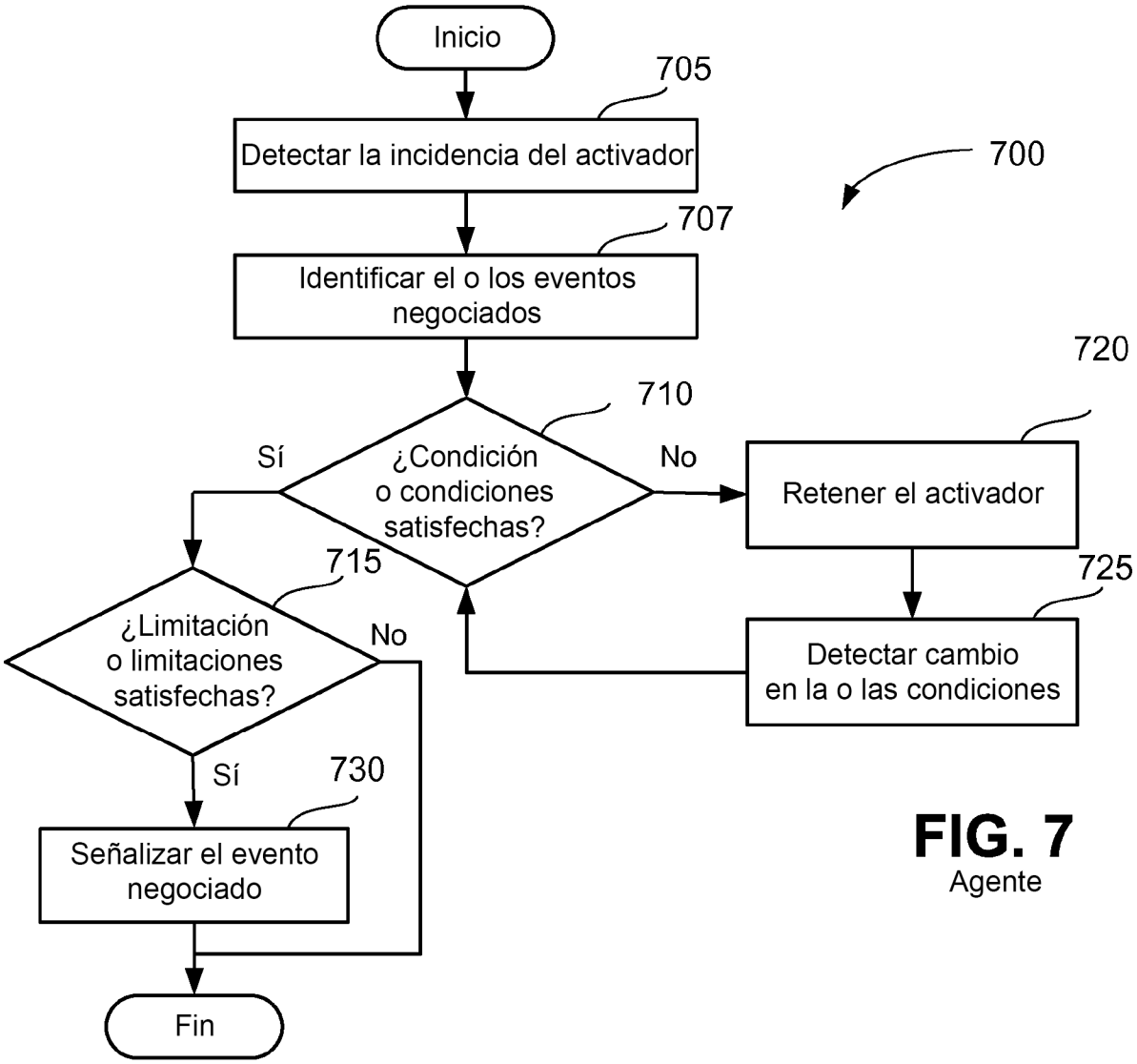

FIG. 7
Agente

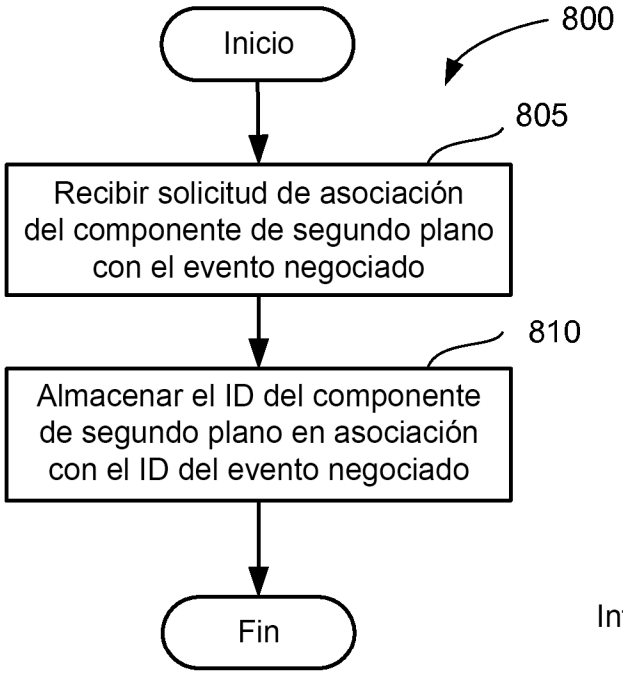

FIG. 8A

Infraestructura de agentes

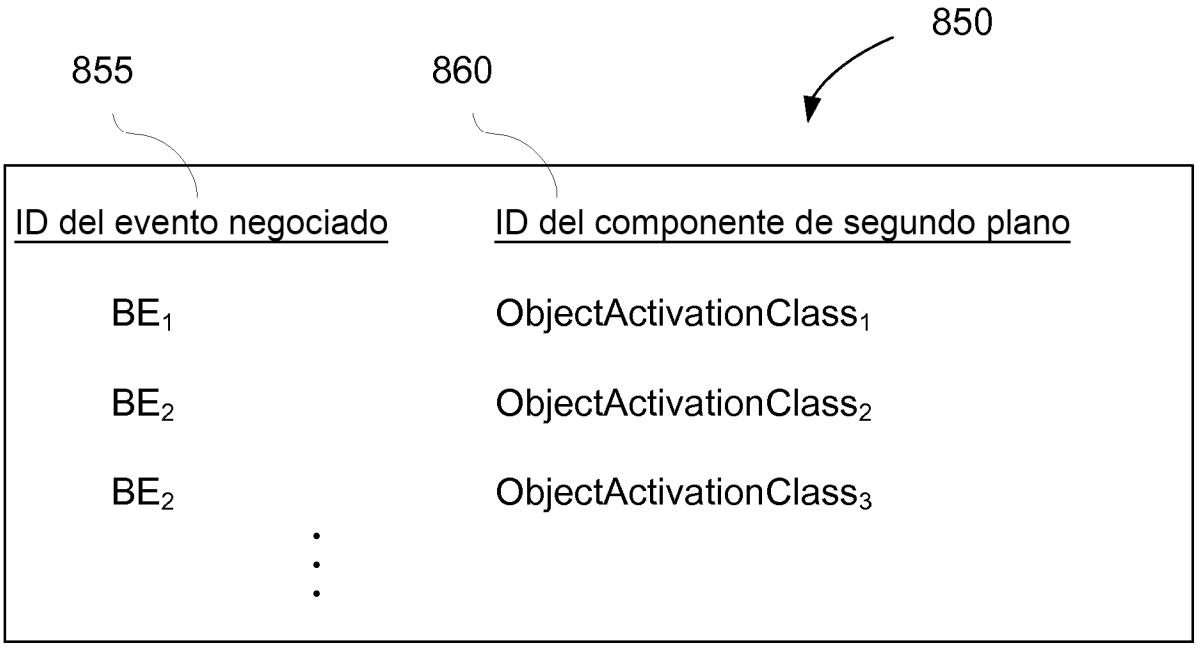

FIG. 8B

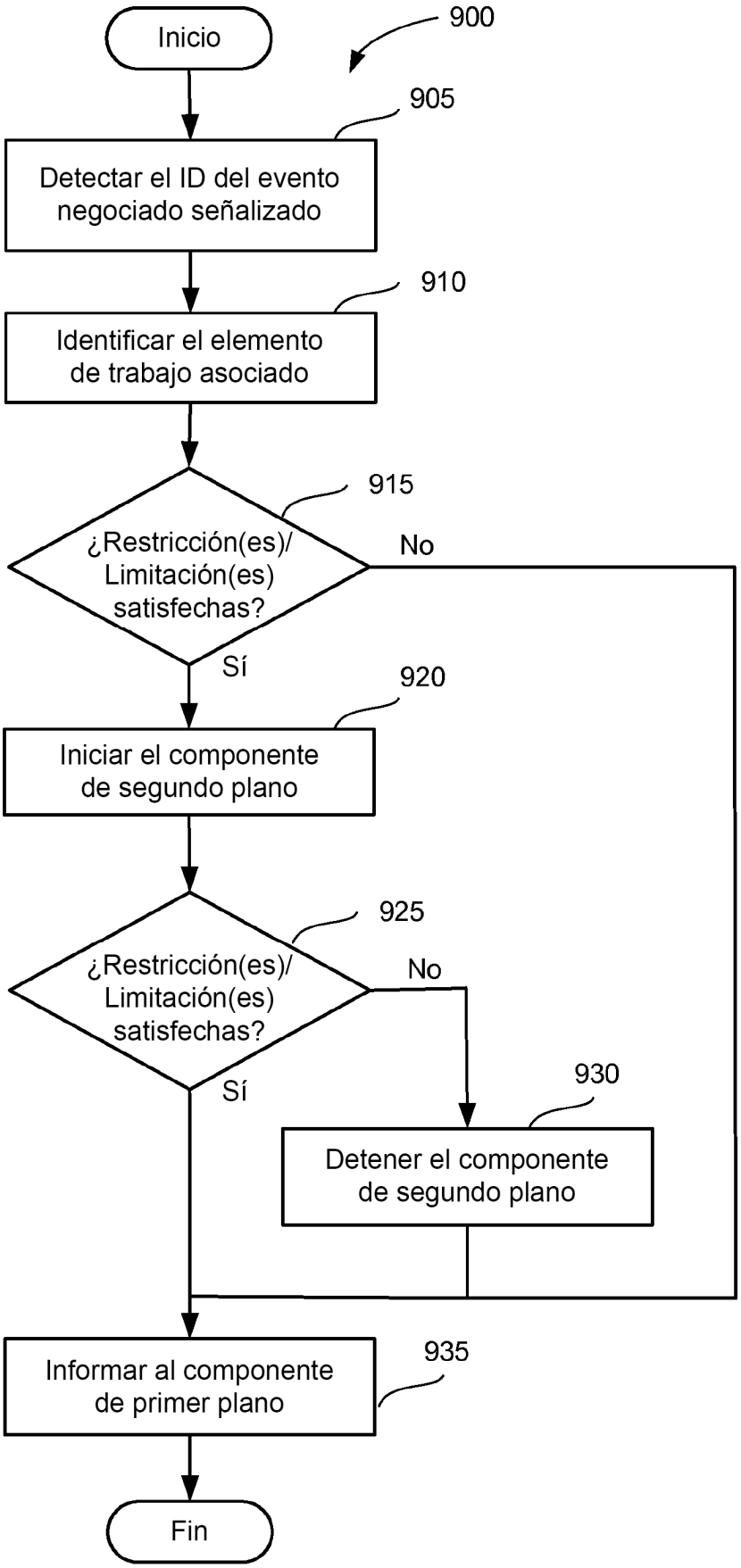
Inicio
900
905
Detectar el ID del evento negociado señalizado
910
Identificar el elemento de trabajo asociado
915
¿Restricción(es)/ Limitación(es) satisfechas?
No
Sí
920
Iniciar el componente de segundo plano
925
¿Restricción(es)/ Limitación(es) satisfechas?
No
Sí
930
Detener el componente de segundo plano
935
Informar al componente de primer plano
Fin

FIG. 9

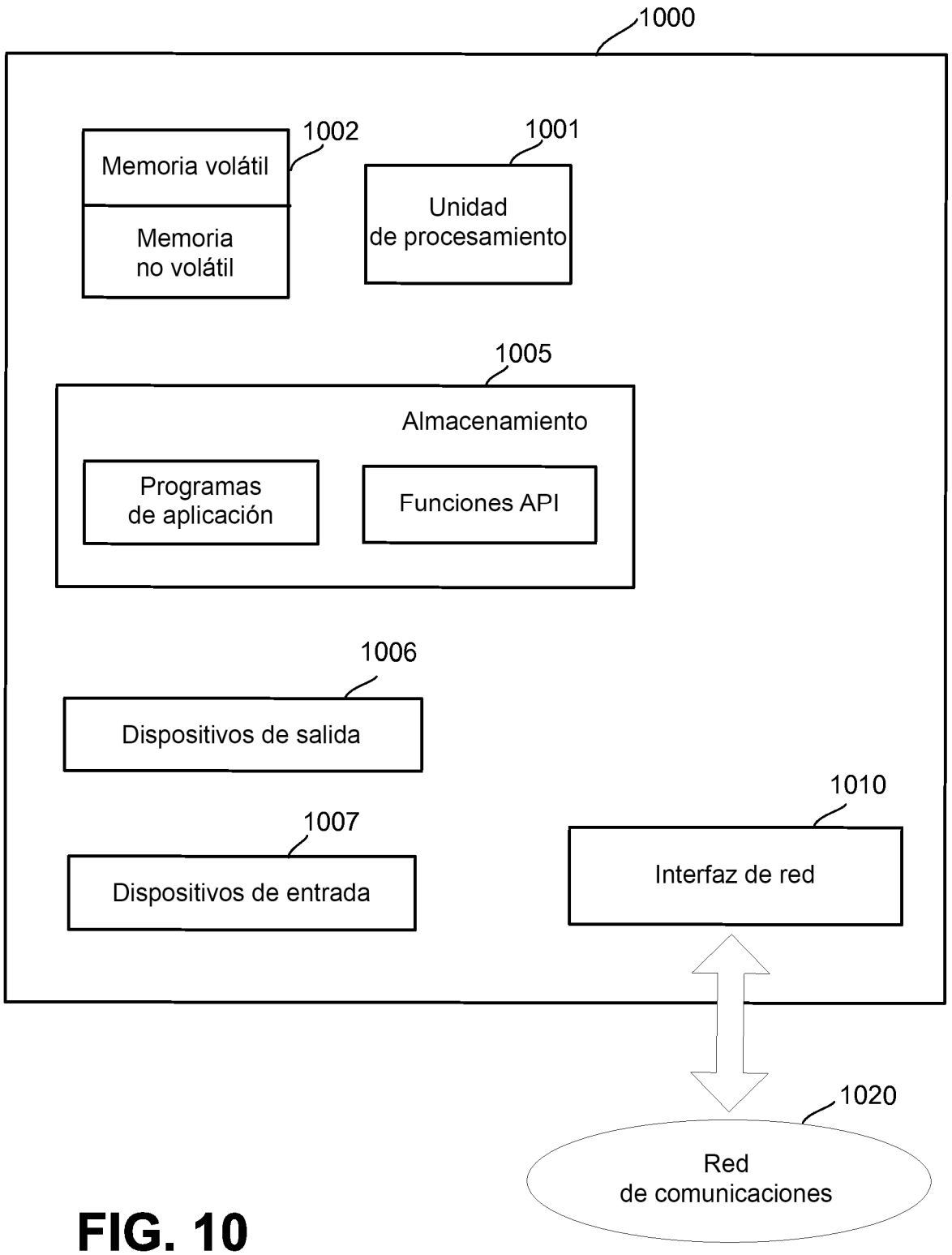
1000
1002
Memoria volátil
Memoria
no volátil
1001
Unidad
de procesamiento
1005
Almacenamiento
Programas
de aplicación
Funciones API
1006
Dispositivos de salida
1007
Dispositivos de entrada
1010
Interfaz de red
1020
Red
de comunicaciones

FIG. 10