(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 613 831**

(51) Int. CI.:

H04L 29/06 (2006.01)
G06F 21/33 (2013.01)
H04L 9/08 (2006.01)
H04L 29/08 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **26.02.2014 PCT/US2014/018459**

(87) Fecha y número de publicación internacional: **04.09.2014 WO2014134081**

(96) Fecha de presentación y número de la solicitud europea: **26.02.2014 E 14712392 (1)**

(97) Fecha y número de publicación de la concesión europea: **04.01.2017 EP 2962440**

(54) Título: **Comunicaciones en tiempo real usando una API RESTLIKE**

(30) Prioridad:

28.02.2013 US 201361771073 P
26.06.2013 US 201313927116

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
26.05.2017

(73) Titular/es:

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

(72) Inventor/es:

TAINE, STEPHANE;
RAO, DEEPAK;
YOUNIS, SHAHZAIB;
GANESAN, KRISHNAMURTHY y
EDELSBURG, ALEX

(74) Agente/Representante:

UNGRÍA LÓPEZ, Javier

DESCRIPCIÓN

Comunicaciones en tiempo real usando una API RESTLIKE

Solicitud relacionada

Esta solicitud reivindica prioridad por la Solicitud de Patente Provisional de Estados Unidos número 61/771.073, presentada el 28 de Febrero de 2013, y titulada “Tecnologías de comunicaciones unificadas”, cuya totalidad se incorpora aquí por referencia.

Antecedentes

Las aplicaciones de comunicaciones unificadas (UC) implementadas por ordenador permiten a sus usuarios recibir y transmitir comunicaciones en tiempo real por medio de una pluralidad de diferentes modalidades de comunicaciones. Por ejemplo, una aplicación UC ejemplar puede estar configurada para soportar mensajería instantánea, comunicaciones de voz, teleconferencia, vídeo conferencia, recuperación de correo de voz, sincronización de calendarios, y compartición de contenido, entre otras modalidades de comunicaciones.

Además, la señalización de puntos finales con instancias de lado de cliente de una aplicación UC instalada en ellas ha sido movida por protocolos dedicados, de propiedad y estándar (a base de sesión, orden y control, o hop-by-hop), tal como SIMPLE, SIP, H.323, XMPP, Jingle, Skinny, y análogos. Cada uno de estos protocolos ha sido finamente ajustado para afrontar un aspecto de comunicaciones respectivo soportado por la aplicación UC; por ejemplo, se puede emplear un primer protocolo para establecimiento de llamada telefónica, se puede emplear un segundo protocolo para determinar o firmar la presencia de una persona, etc. Una consecuencia de este acercamiento es que las instancias de lado de cliente de aplicaciones UC convencionales son relativamente complejas, puesto que la lógica de núcleo es impulsada a tales instancias de lado de cliente (denominadas aquí “clientes”). Efectivamente, entonces los clientes actúan como respectivos puntos de coordinación de varios flujos de señalización - si un cliente es incapaz de soportar un cierto protocolo, puede no ser considerado por los usuarios como una aplicación UC en toda regla. En un ejemplo, los clientes web convencionales soportan mensajería instantánea y presencia, pero no comunicaciones de voz y vídeo a nivel de empresa, y así pueden no ser considerados como una aplicación UC en toda regla por los usuarios.

Varias deficiencias están asociadas con las aplicaciones UC convencionales debido a la arquitectura descrita anteriormente. En primer lugar, la evolución de clientes ligeros es difícil, puesto que los protocolos de llamada de procedimiento remoto son relativamente inflexibles al cambio, puesto que tales protocolos fuerzan un acoplamiento explícito o implícito (acoplamiento ajustado) entre clientes y el código correspondiente que se ejecuta en servidores. Así, si se añaden elementos nuevos a la aplicación UC, los clientes de legado pueden ser inoperativos. Además, el acercamiento convencional está asociado con una huella de memoria alta, ruido, y alta utilización de anchura de banda. Un protocolo de procedimiento remoto (RPC) adoptado para alimentar clientes web convencionales en aplicaciones UC puede producir código de lado de servidor (denominado aquí “servidores”) para sincronizar a menudo 100% de un estado de cliente, incluso cuando una interfaz de usuario del cliente necesita una cantidad relativamente pequeña de información. Además, tal acercamiento está asociado con una falta de herramientas estándar/de supervisión, puesto que el RPC antes indicado emplea el protocolo de transferencia de hipertexto (HTTP) como una capa de transporte, con un localizador unificado de recursos (URL) que representa el servicio - esto da lugar a una falta de herramientas estándar para supervisar la salud de la aplicación UC.

WO 2010/042733 describe un método de gestionar las conexiones entre un cliente, y aplicación intermedia y un servidor, de modo que se pueda transmitir mensajes asíncronos por HTTP desde el servidor al cliente.

US 7.647.329 describe una arquitectura de servicio de distribución de teclado para un sistema de almacenamiento distribuido. El sistema implementa nodos de almacenamiento configurados para almacenar réplicas de objetos de datos, donde cada una de las réplicas es accesible mediante un valor de localizador respectivo, e instancias de distribución de teclado, cada una configurada para almacenar entradas de distribución de teclado correspondientes respectivamente a los objetos de datos.

Resumen

Lo siguiente es un breve resumen de la materia que se describe con más detalle aquí. Este resumen no pretende ser limitativo del alcance de las reivindicaciones.

Aquí se describen varias tecnologías pertenecientes a una interfaz de programa de aplicación (API) que generalmente es conforme a una arquitectura de transferencia de estado representacional (REST). Tal API, cuando es empleada por una aplicación UC, proporciona soporte para comunicaciones síncronas (por ejemplo, Voice-Over-IP (VoIP), vídeo conferencia, compartición de aplicación, colaboración de datos, …). Además, la aplicación UC también puede soportar comunicaciones asíncronas (por ejemplo, planificación de encuentros online, presencia, y sincronización de contactos e información de grupo, …). Además, la aplicación UC que utiliza la API aquí descrita

puede facilitar las comunicaciones asíncronas y síncronas en tiempo real (en el orden de milisegundos o menos).

En una realización ejemplar, tal aplicación puede ser una aplicación de comunicaciones (UC) unificadas. Una arquitectura de sistema que soporta tal aplicación incluye dispositivos informáticos de cliente en comunicación con dispositivos informáticos de servidor, donde los dispositivos informáticos de cliente ejecutan instancias de lado de cliente de la aplicación (clientes) y los dispositivos informáticos de servidor ejecutan código de lado de servidor (servidores). Los dispositivos informáticos de cliente pueden incluir dispositivos informáticos de sobremesa así como dispositivos informáticos portátiles como teléfonos móviles, tabletas (pizarras), phablets, wearables, etc. Según aspectos aquí descritos, las comunicaciones síncronas y asíncronas en tiempo real entre clientes y/o entre un cliente y un servidor pueden realizarse mediante el empleo de una API que es conforme a la arquitectura REST (por ejemplo, una API RESTLIKE).

La API es modelada en términos de identificadores y órdenes, donde los identificadores representan recursos y las órdenes representan métodos de interactuar con recursos, tal como métodos convencionales de protocolo de transferencia de hipertexto (HTTP). La aplicación UC puede transmitir datos por medio de la API usando una variedad de formatos de carga, tal como tipos de hipermedios convencionales. Un recurso puede ser o incluir datos, tal como un documento, imagen, entrada de base de datos, etc. En otro ejemplo, un recurso puede ser o incluir código ejecutable por ordenador, tal como un servicio, un hilo ejecutable, etc. Además, los recursos pertenecen por lo general a un concepto concreto descrito por sus respectivos identificadores. Por ejemplo, un recurso “contactos” puede incluir una lista de contactos de un usuario, o puede ser un servicio que recupere la lista de contactos de otros recursos. Cada recurso es identificado por un identificador global respectivo, tal como un localizador universal de recursos (URL) que puede ser usado por el cliente para acceder a un recurso respectivo. Los recursos también pueden incluir identificadores globales de otros recursos, con el fin de indicar una relación de enlace. Por ejemplo, el recurso “contactos” también puede incluir un identificador global de un recurso “perfil” que incluye un perfil de un contacto del usuario. El cliente, después de acceder al recurso “contactos” y localizar el identificador global del recurso “perfil”, puede usar entonces el identificador global para el recurso “perfil” para recuperar el perfil del contacto. Tales relaciones de enlace pueden permitir que un cliente navegue por recursos de manera similar a como un usuario puede navegar por páginas web usando hiperenlaces.

Así, una API ejemplar empleada por una aplicación UC puede usar relaciones de enlace para expresar capacidades soportadas por la aplicación UC, donde el uso de relaciones de enlace es conforme al principio HATEOAS (hipertexto como el motor de estado de aplicación). Pueden introducirse nuevas capacidades añadiendo nuevas relaciones de enlace.

Como se ha indicado anteriormente, las órdenes representan métodos respectivos de interactuar con recursos. Los métodos ejemplares incluyen recuperar datos de un recurso, añadir datos a un recurso, sustituir o añadir un recurso, y borrar un recurso. Combinando un identificador y una orden en una petición, un cliente puede indicar un recurso al que se desea acceder así como una operación que se desea ejecutar. Por ejemplo, con el fin de recuperar una lista de contactos de un usuario, un cliente puede generar una petición que incluya una orden que represente una operación de recuperación y un identificador que represente el recurso “contactos”.

Sin embargo, puede ser deseable añadir, modificar o borrar recursos soportados por el servidor, y consiguientemente puede ser deseable actualizar clientes. Así, en un ejemplo, un cliente puede enviar una “petición pendiente” al servidor con respecto a eventos (por ejemplo, un listado de actualizaciones de recursos), donde una petición pendiente es una a la que el servidor no tiene que responder inmediatamente. Por ejemplo, más bien que responder que no hay disponibles nuevas actualizaciones de recursos usados por el cliente, el servidor puede esperar hasta que tenga lugar una actualización para generar/transmitir una respuesta.

Además, el servidor puede apalancar una estructura de eventos para agregar eventos y actualizaciones a recursos en el servidor y determinar cómo y cuándo las notificaciones de tales actualizaciones son distribuidas al cliente. Más bien que enviar todas las actualizaciones, por ejemplo, sincronizando 100% del estado, el servidor puede transmitir al cliente metadatos que describen las actualizaciones que están disponibles, y el cliente puede pedir actualizaciones deseadas. En una realización ejemplar, el servidor puede asignar a cada evento/actualización una categoría, donde las categorías ejemplares incluyen sensible a tiempo real, urgencia alta y urgencia baja.

Cuando una actualización es sensible a tiempo real, el servidor puede embeber la actualización en los metadatos transmitidos al cliente, permitiendo que el cliente reciba la actualización sensible a tiempo real sin tener que determinar que la actualización es necesaria, hacer una petición al servidor, y esperar que se reciba la actualización, reduciendo por ello la latencia para enviar actualizaciones sensibles a tiempo real. Las actualizaciones de urgencia alta pueden indicarse en los metadatos, y el cliente puede elegir cuándo pedir la distribución de tales actualizaciones al servidor. Las actualizaciones pueden recuperar con prontitud (por ejemplo, de forma relativamente rápida), lentamente (cuando sea conveniente o cuando el costo de recuperar la actualización esté por debajo de un umbral), o una actualización puede ser ignorada. Las actualizaciones de urgencia baja pueden ser agregadas y retenidas por el servidor hasta que haya pasado un período de tiempo predeterminado (por ejemplo, hasta que la petición pendiente expire) o hasta que la actualización de urgencia baja sea de urgencia más alta. Con respecto a actualizaciones recuperadas de forma precoz, la provisión de una actualización de urgencia alta puede ser retardada

durante un período de tiempo relativamente corto (por ejemplo, del orden de 50 ms) para permitir la agregación de actualizaciones de urgencia alta.

Además, una aplicación puede incorporar una operación asíncrona, donde uno o varios puntos finales y/o un servidor no tienen que operar simultáneamente. Por ejemplo, en mensajería electrónica, un receptor no tiene que estar activo para que un emisor envíe un mensaje. Una operación asíncrona puede ser iniciada por un cliente, tal como un emisor que desea enviar un mensaje, o por un servidor, tal como cuando el servidor desea distribuir actualizaciones de archivos compartidos entre múltiples clientes que trabajan de forma asíncrona. Dado que porciones de la operación asíncrona pueden tener lugar en posiciones diferentes (por ejemplo, en el cliente o servidor), y dado que tales porciones pueden tener lugar en tiempos diferentes, un recurso puede persistir en el servidor después de haberse completado una petición de cliente. En una arquitectura REST convencional (por ejemplo, sin estado), el recurso persistente puede no ser servido al cliente en una petición posterior. Sin embargo, apalancando la API RESTLIKE, el cliente, pidiendo realizar tal operación asíncrona, puede hacer que el servidor cree un recurso operativo que puede mantener actualizado el cliente en cuanto al progreso de la operación asíncrona, pero respectando el principio HATEOAS.

El resumen anterior presenta un resumen simplificado con el fin de ofrecer una comprensión básica de algunos aspectos de los sistemas y/o métodos aquí explicados. Este resumen no es una visión general amplia de los sistemas y/o métodos aquí explicados. No se ha previsto identificar elementos clave/críticos o delinear el alcance de tales sistemas y/o métodos. Su única finalidad es presentar algunos conceptos de una forma simplificada como un preludio a la descripción más detallada que se presenta más adelante.

Breve descripción de los dibujos

Las figuras 1 y 2 son diagramas de bloques funcionales de respectivos sistemas ejemplares que facilitan la ejecución de una aplicación que soporta comunicación bidireccional síncrona en tiempo real mediante el uso de una API RESTLIKE mientras que también soporta comunicación asíncrona.

La figura 3 es un diagrama de bloques funcionales de un sistema ejemplar que facilita la ejecución de una aplicación que soporta una operación asíncrona mediante el uso de una API RESTLIKE.

Las figuras 4-9 son diagramas de flujo que ilustran respectivas metodologías ejemplares para comunicación asíncrona y síncrona bidireccional en tiempo real mediante el uso de una API RESTLIKE.

La figura 10 es un diagrama de flujo que ilustra una metodología ejemplar para comunicaciones bidireccionales para una operación asíncrona mediante el uso de una API RESTLIKE.

La figura 11 es un sistema informático ejemplar.

Descripción detallada

Varias tecnologías pertenecientes a comunicaciones en tiempo real entre dispositivos informáticos mediante el uso de una API que es conforme a arquitectura de transferencia de estado representacional (REST) se describen ahora con referencia a los dibujos, donde se usan números de referencia análogos para hacer referencia a elementos análogos en todos ellos. En la descripción siguiente, a efectos de explicación, se exponen numerosos detalles específicos con el fin de proporcionar una comprensión completa de uno o varios aspectos. Puede ser evidente, sin embargo, que tal(es) aspecto(s) se puede(n) poner en práctica sin estos detalles específicos. En otros casos, estructuras y dispositivos conocidos se representan en forma de diagrama de bloques con el fin de facilitar la descripción de uno o arios aspectos. Además, se ha de entender que la funcionalidad que se describe como realizada por algunos componentes del sistema puede ser realizada por múltiples componentes. Igualmente, por ejemplo, un componente puede estar configurado para realizar una funcionalidad que se describe como realizada por múltiples componentes.

Además, el término “o” pretende significar un “o” inclusivo más bien que un “o” exclusivo. Es decir, a no ser que se especifique lo contrario, o sea claro por el contexto, la expresión “X emplea A o B” pretende significar cualquiera de las permutaciones inclusivas naturales. Es decir, la expresión “X emplea A o B” se cumple en cualquiera de los casos siguientes: X emplea A; X emplea B; o X emplea tanto A como B. Además, los artículos “uno/una” usados en esta solicitud y en las reivindicaciones anexas deberán interpretarse en general en el sentido de “una o varios” a no ser que se especifique lo contrario o sea claro por el contexto que se refiere a una forma singular.

Además, en el sentido en que se usa aquí, los términos “componente” y “sistema” pretenden abarcar almacenamiento de datos legibles por ordenador que está configurado con instrucciones ejecutables por ordenador que hacen que cierta funcionalidad sea realizada cuando sea ejecutada por un procesador. Las instrucciones ejecutables por ordenador pueden incluir una rutina, una función, o análogos. También se ha de entender que un componente o sistema puede estar localizado en un solo dispositivo o distribuido por varios dispositivos. Además, en el sentido en que se usa aquí, el término “ejemplar” pretende significar que sirve como una ilustración o ejemplo de

algo, y no se prevé que indique preferencia.

Con referencia ahora a la figura 1 se ilustra un sistema ejemplar 100 que facilita comunicaciones síncronas y asíncronas en tiempo real entre dispositivos informáticos. El sistema 100 incluye una pluralidad de dispositivos informáticos de cliente 102-104. Los dispositivos informáticos de cliente 102-104 pueden incluir dispositivos informáticos de sobremesa, dispositivos informáticos personales, netbooks, ultrabooks, tabletas (pizarras), phablets, teléfonos móviles, etc. El sistema 100 incluye además un dispositivo informático de servidor 106, que está en comunicación con los dispositivos informáticos de cliente 102-104 por medio de respectivas conexiones de red adecuadas. Además, se puede establecer canales de comunicaciones directamente entre dispositivos informáticos en los dispositivos informáticos de cliente 102-104 para facilitar comunicaciones entre ellos.

Los dispositivos informáticos de cliente 102-104 y el dispositivo informático de servidor 106 están configurados para soportar comunicaciones síncronas y asíncronas en tiempo real entre ellos por medio de una aplicación de comunicaciones distribuida entre los dispositivos informáticos de cliente 102-104 y el dispositivo componente de servidor 106. Con más detalle, el primer dispositivo informático 102 incluye un primer módulo procesador 108 que ejecuta instrucciones retenidas en una primera memoria 110. La primera memoria 110 puede incluir una primera instancia de lado de cliente de una aplicación de comunicaciones, denominada un primer cliente 112. Igualmente, el enésimo dispositivo informático 104 puede incluir un enésimo módulo procesador 114 que esté configurado para ejecutar instrucciones en una enésima memoria 116. La enésima memoria 116 incluye una enésima instancia de lado de cliente de la aplicación de comunicaciones, denominada un enésimo cliente 118.

Aunque el dispositivo informático de servidor 106 se ilustra como un solo dispositivo en la figura 1, el dispositivo informático de servidor 106 puede ser una pluralidad de dispositivos informáticos de servidor (por ejemplo, un servidor distribuido) y puede ser o incluir una o varias máquinas virtuales. El dispositivo informático de servidor 106 incluye un módulo procesador de servidor 120 que está configurado para ejecutar código en una memoria de lado de servidor 122. Por ejemplo, la memoria de lado de servidor 122 puede incluir código de lado de servidor de la aplicación de comunicaciones, donde el código de lado de servidor está configurado para soportar al menos una modalidad de comunicaciones de la aplicación de comunicaciones. El código de lado de servidor se denomina aquí un servidor 124. Por ejemplo, el servidor 124 puede soportar mensajería instantánea entre dispositivos informáticos de cliente, mantenimiento y provisión de información de presencia a dispositivos informáticos de cliente, etc.

La aplicación de comunicaciones distribuida entre los dispositivos informáticos de cliente 102-104 y el dispositivo componente de servidor 106 puede soportar comunicaciones asíncronas en tiempo real y comunicaciones síncronas en tiempo real entre dispositivos informáticos. Comunicación asíncrona en tiempo real se refiere a transmisión de datos a un dispositivo receptor (por ejemplo, uno de los dispositivos informáticos de cliente 102-104) en tiempos o intervalos que son desconocidos por el dispositivo receptor. Por ejemplo, una comunicación asíncrona puede ser al menos uno de una transferencia de archivo, un mensaje electrónico (por ejemplo, correo electrónico, mensajería instantánea, …), una petición de programa de evento, una indicación de presencia, una actualización de una lista de contactos o grupos, etc. Comunicación síncrona se refiere a cuando tanto un emisor como un receptor comunican y están sincronizados uno con otro, aunque en algunos casos la sincronización puede ser aproximada. Por ejemplo, comunicación síncrona puede incluir vídeo streaming, vídeo conferencia, comunicación de voz, compartición de aplicación entre dispositivos informáticos, compartición de sobremesa, etc.

En una realización ejemplar, la aplicación de comunicaciones distribuida entre los dispositivos informáticos de cliente 102-104 y el dispositivo informático de servidor 106 puede ser una aplicación de comunicaciones (UC) unificadas que soporte múltiples modalidades de comunicaciones, incluyendo al menos una de las modalidades síncrona y asíncrona antes referenciadas. En otro ejemplo, la aplicación de comunicaciones puede ser o estar incluida en una aplicación de red social que soporte mensajería instantánea, presencia y grupos. En otro ejemplo, tal aplicación de comunicaciones puede ser una aplicación de mensajería instantánea que soporte mensajería instantánea, presencia y comunicaciones de voz. También se contemplan otros tipos de aplicaciones de comunicaciones.

Los clientes 112-118 y el servidor 124 pueden usar una API RESTLIKE para comunicar sincrónicamente en tiempo real. Para ello, el cliente 112 puede hacer que se visualice una interfaz gráfica de usuario (GUI) que incluya, por ejemplo, contactos para un usuario del cliente 112, mensajes recibidos por el cliente 112, etc. El servidor 124 puede acceder, por ejemplo, a una lista de contactos del usuario, mensajes enviados y recibidos mediante el cliente 112, y otros datos con el fin de poblar, por ejemplo, la interfaz de usuario. Cuando, por ejemplo, el usuario del primer dispositivo informático de cliente 102 desea enviar una comunicación a un contacto (por ejemplo, el usuario del enésimo dispositivo informático de cliente 104), el primer cliente 112 puede comunicar con el enésimo cliente 118 mediante el servidor 124 o mediante un canal establecido por el servidor 124. El servidor 124, además de dirigir la comunicación, también puede enviar otra información al primer cliente 112, tal como que el contacto está disponible para comunicar en tiempo real.

Como se describirá con más detalle aquí, la API RESTLIKE es modelada en términos de identificadores y órdenes, donde los identificadores representan recursos almacenados en la memoria de servidor 122, y las órdenes representan métodos de interactuar con recursos, tal como los métodos convencionales de protocolo de transferencia de hipertexto (HTTP) y tipos de hipermedios. Un recurso puede ser o incluir datos, tal como un

documento, una imagen, etc. Además, un recurso puede ser o incluir un servicio ejecutable por ordenador u otras instrucciones ejecutables por ordenador. Además, un recurso puede ser o incluir una combinación de datos e instrucciones ejecutables por ordenador. Además, cada recurso puede ser dirigido hacia un respectivo concepto descrito por su identificador. Por ejemplo, un recurso “contactos” puede incluir una lista de contactos de un usuario. Cada recurso es identificado por un identificador global respectivo, tal como un localizador universal de recursos (URL) que puede ser usado por el cliente para acceder a un recurso respectivo. Un recurso también puede incluir un identificador global de otro recurso (o múltiples recursos), con el fin de indicar una relación de enlace entre el recurso y el otro recurso. Por ejemplo, el recurso “contactos” también puede incluir un identificador global de un recurso “perfil”, donde el recurso perfil incluye un perfil de un contacto del usuario. El primer cliente 112, después de acceder al recurso “contactos” (por ejemplo, desde el servidor 124) y localizar el identificador global para el recurso “perfil” en él, puede usar entonces el identificador global para que el recurso “perfil” acceda al perfil del contacto. Tales relaciones de enlace pueden permitir que el primer cliente 112 navegue por recursos de una manera similar a como un usuario puede navegar por páginas web usando hiperenlaces. Más adelante se exponen detalles adicionales pertenecientes a la API RESTLIKE.

Con referencia ahora a la figura 2 se ilustra un sistema ejemplar 200 que facilita comunicación bidireccional síncrona en tiempo real y asíncrona en tiempo real (por ejemplo, entre un cliente y servidor o un cliente y cliente) mediante utilización de una API RESTLIKE. El sistema 200 incluye el primer cliente 112 que está en comunicación con el servidor 124 mediante la utilización de la API RESTLIKE. En una realización alternativa, el servidor 124 puede ser sustituido por el enésimo cliente 118, de tal manera que el primer cliente 112 y el enésimo cliente 118 estén en comunicación directa uno con otro. La API RESTLIKE, como se ha indicado anteriormente, se modela usando identificadores y órdenes. Tales identificadores y órdenes pueden ser universales, porque pueden ser usadas sin consideración a una aplicación específica que se ejecute, un dispositivo específico con el que se comunique, o una operación específica que se emprenda, y proporcionar así una interfaz generalizada entre dispositivos.

Ahora se describen recursos correspondientes a la API RESTLIKE. Como se representa, el servidor 124 puede incluir o tener acceso a una pluralidad de recursos (por ejemplo, un primer recurso 202 a través de un emésimo recurso 204). El formato de un recurso es arbitrario, puesto que un recurso puede estar estructurado como una base de datos, un archivo, etc. Los identificadores pueden representar respectivamente los recursos 202-204, donde un recurso puede incluir al menos uno de datos (tal como un documento, una imagen, una entrada o entradas de base de datos, etc), código ejecutable por ordenador, y análogos. Cada recurso pertenece a un concepto respectivo (por ejemplo, que puede describirse por un identificador para el recurso). Por ejemplo, el primer recurso 202 puede ser un recurso “contactos”, y así puede incluir una lista de contactos de un usuario del primer dispositivo informático de cliente 102 que incluye el primer cliente 112. Cada recurso es identificado por un identificador global respectivo, donde un identificador global puede ser un localizador universal de recursos (URL) que puede ser usado por el primer cliente 112 para acceder al recurso en el servidor 124 correspondiente al identificador global. Por ejemplo, un identificador global para el primer recurso 202 puede ser el siguiente:

/messagingapp/user/contacts

Un recurso también puede incluir identificador(es) global(es) de otro(s) recurso(s), con el fin de indicar una relación de enlace al (a los) otro(s) recurso(s). Por ejemplo, el primer recurso 202 puede incluir un identificador global para el emésimo recurso 204, donde el emésimo recurso 204 es un recurso “perfil” e incluye un perfil de un contacto incluido en el recurso “contactos”. Así, por ejemplo, el primer recurso 202 puede incluir una entrada de base de datos relativa a un contacto del usuario del primer cliente 112 (por ejemplo, “Contact1”) con un identificador global del emésimo recurso 204, donde el emésimo recurso incluye el perfil de Contact1, como se representa en el ejemplo siguiente:

Contactos	Perfiles
Contact1	/messagingapp/contact1/profile

Los recursos 202-204 pueden incluir una variedad de diferentes tipos de recursos, incluyendo, aunque sin limitación, un recurso dinámico, un recurso de operación, un recurso emisor, un recurso de evento, un recurso de capacidad, un recurso de depósito, un recurso de factoría de operación, un recurso de depósito dinámico, y/o un recurso que combina aspectos de dichos recursos. Un recurso dinámico es uno que probablemente se cambia o actualiza a menudo, tal como un indicador de presencia para un contacto; un recurso operativo representa una operación asíncrona; un recurso emisor está configurado para reportar actualizaciones de un grupo de recursos dinámicos y de operación; un recurso de evento incluye un lote o lista de actualizaciones agrupadas por emisor; un recurso de capacidad representa una capacidad de un cliente concreto; un recurso de depósito incluye recursos de otros tipos; un recurso de factoría de operación es un recurso que puede ser usado primariamente para iniciar un recurso de operación; un recurso de depósito dinámico incluye otros recursos y puede generar un evento cuando un recurso es insertado, modificado o cambiado. También se ha de entender que un recurso puede incluir o combinar elementos de dichos tipos de recursos. Por ejemplo, un recurso de operación también puede actuar como un recurso emisor y enviar actualizaciones.

Ahora se describen métodos soportados por la API RESTLIKE (representada por órdenes) para interactuar con recursos. Los métodos ejemplares soportados por la API RESTLIKE incluyen métodos convencionales de protocolo

de transferencia de hipertexto (HTTP) y tipos de hipermedios. Así, con más especificidad, los métodos ejemplares incluyen un método para recuperar datos de un recurso (por ejemplo, mediante una orden “GET”), un método para añadir datos a un recurso (por ejemplo, mediante una orden “POST”), un método para sustituir o añadir un recurso (por ejemplo, mediante una orden “PUT”), y un método para borrar un recurso (por ejemplo, mediante una orden “DELETE”). El primer cliente 112 puede generar una petición que incluye un identificador y una orden, identificando por ello un recurso al que se desea acceder y un método a ejecutar en conexión con el recurso. Consiguientemente, el primer cliente 112 incluye un componente solicitante 206 que está configurado para acceder a un recurso en los recursos 202-204 enviando una petición al servidor 124 que incluye una identificación de un método (tal como un método para recuperar datos de un método) y una identificación de un identificador global del recurso al que se desea acceder.

Por ejemplo, el componente solicitante 206, con el fin de recuperar la lista de contactos del recurso “contactos”, puede exponer la petición siguiente:

GET /messagingapp/user/contacts

donde “GET” identifica el método para recuperar datos, y

/messagingapp/user/contacts es un identificador global que identifica el primer recurso 202. Así, más bien que tener que almacenar una lista de contactos localmente, el primer cliente 112 puede pedir el contenido del primer recurso 202 del servidor 124. El uso de identificadores globales y métodos permite al primer cliente 112 hacer peticiones de un recurso sin tener que tener ningún conocimiento de cómo está almacenado el recurso en el dispositivo informático de servidor 106 o la forma en que el servidor 124 implementa el método pedido.

En respuesta a recibir la petición para el primer recurso 202 del primer cliente 112, el servidor 124 puede transmitir el primer recurso 202 al cliente 112. En otro ejemplo, el servidor 124 puede generar un paquete de datos en base al contenido del primer recurso 202 y transmitir tal paquete de datos al primer cliente 112. Por ejemplo, tal paquete de datos puede ser de tamaño más pequeño en comparación con el primer recurso 202 propiamente dicho. Además, el paquete de datos puede estar en un formato que sea analizado más fácilmente por el primer cliente 112 en comparación con el formato del primer recurso 202. En otro ejemplo, donde un recurso es un servicio o incluye código ejecutable, el paquete de datos puede incluir datos que representan el servicio o código ejecutable. Además, el paquete de datos puede estar en un formato que sea reconocible por el cliente.

Consiguientemente, más bien que devolver el primer recurso 202 identificado en la petición expuesta por el componente solicitante 206, el servidor 124 puede generar y devolver un paquete de datos basado en el contenido del primer recurso 202, donde tal paquete de datos puede ser o incluir un documento, una lista, o análogos, y puede estar en un formato de archivo estructurado tal como un archivo de marcación (por ejemplo, XML) o un archivo de notación de objeto (por ejemplo, JSON). Por ejemplo, un paquete de datos que puede ser generado por el servidor 124 en base a un recurso puede ser una representación JSON de una base de datos que incluya una lista de contactos, tal como sigue:

Var ContactsList = [

{“name”: “Contact1”, “Profile”: “/messagingapp/contact1profile.”}

]

El primer cliente 112 incluye además un componente receptor 208 que recibe datos transmitidos al primer cliente 112 desde el servidor 124, tal como el primer recurso 202 o el paquete de datos basado en él, que puede almacenarse en un cache que sea accesible al primer cliente 112.

En una realización ejemplar, como se ha indicado anteriormente, el primer recurso 202 incluye una lista de contactos de un usuario del primer cliente 112, y puede incluir además un identificador global del emésimo recurso 204, donde el emésimo recurso 204 incluye un perfil de un contacto en la lista de contactos del primer recurso 202. Así, en respuesta a recibir el primer recurso 202 o un paquete de datos correspondiente a él, el primer cliente 112 puede identificar el identificador global del emésimo recurso. El componente solicitante 206 puede transmitir entonces una petición para recuperar el emésimo recurso 204 (o datos incluidos en él) del servidor 124 (por ejemplo, usando la orden “GET” y el identificador global para el emésimo recurso 204). Efectivamente, entonces, el primer recurso 202 se enlaza al emésimo recurso 204 mediante una relación de enlace. Las relaciones de enlace entre los recursos 202-204 permiten que el primer cliente 112 navegue por recursos igual a como un usuario puede navegar por páginas web usando hiperenlaces.

El primer recurso 202 (incluyendo la lista de contactos del usuario del primer cliente 112) puede no existir antes de que el servidor 124 reciba la petición para tal primer recurso 202. En tal caso, el servidor 124 puede reunir datos necesarios para generar el primer recurso 202 en respuesta a recibir la petición, y a continuación puede transmitir el primer recurso 202 (o un paquete de datos correspondiente) al primer cliente 112. Se ha de entender, sin embargo,

que el primer cliente 112 no tiene que tener conocimiento de cómo/cuándo se generan los recursos pedidos, puesto que la API RESTLIKE proporciona una interfaz generalizada entre el primer cliente 112 y el servidor 124. Por lo tanto, se puede considerar que el primer cliente 112 y el servidor 124 están flojamente acoplados, porque el primer cliente 112 y el servidor 124 no tienen que tener un conocimiento específico de los componentes o métodos ejecutados por el otro.

La API RESTLIKE empleada por el primer cliente 112 y el servidor 124 puede facilitar la actualización del primer cliente 112 con funcionalidad adicional (o funcionalidad reducida). Por ejemplo, un servicio o información ofrecido por el servidor 124 puede ser actualizado deseablemente, y además puede ser deseable proporcionar tales actualizaciones al primer cliente 112.

Por ejemplo, la aplicación de comunicaciones puede ser actualizada con un nuevo elemento que proporcione a un usuario una lista de contactos y datos de posición de cada contacto en la lista de contactos. Así, el primer recurso 202, que en este ejemplo incluye la lista de contactos del usuario del primer cliente 112, se puede modificar de manera que incluya un identificador global de un /ésimo recurso (no representado) en la pluralidad de recursos 202-204. El primer cliente 112 puede ignorar la nueva relación de enlace, y por ello la petición expuesta por el primer cliente 112 para el primer recurso 202 sigue siendo válida. Por lo tanto, el primer recurso 202 puede ser modificado sin impactar en el comportamiento del primer cliente 112. En un ejemplo particular, el primer recurso 202 puede ser modificado de manera que incluya lo siguiente:

Contactos	Perfiles	Datos de posición
Contact1	/messagingapp/contact 1 /profile.	/messagingapp/contact 1 /location.

El servidor 124 puede generar un paquete de datos basado en el primer recurso 202 de la siguiente manera:

Var ContactsList = [

{

Name": "Contact1",

Continuando con este ejemplo, el emésimo recurso 204 es identificado por el identificador global messagingapp/contact1/profile.html, y una petición para tal emésimo recurso 204 sigue siendo válida. Por lo tanto, la actualización en el servidor 124 no tiene que afectar al comportamiento del primer cliente 112.

Igualmente, el primer cliente 112 puede ser actualizado teniendo en cuenta cambios en los recursos sin afectar a la operación del servidor 124. Por ejemplo, el primer cliente 112 puede ser actualizado contemplando el identificador global para el /ésimo recurso, de tal manera que el componente solicitante 206 transmita una petición para el /ésimo recurso cuando el cliente 112 analice el primer recurso 202 (o un paquete de datos correspondiente).

Detectar estos tipos de actualizaciones puede incluir, por ejemplo, navegar por identificadores globales de recursos, construir un mapa o gráfico basado en la navegación, y comparar el mapa o gráfico con un mapa o gráfico previo. El primer cliente 112 puede navegar por los identificadores globales en el primer recurso 202, tal como los identificadores globales del /ésimo recurso y el emésimo recurso 204, y puede navegar igualmente por cualesquiera identificadores globales abarcados por el /ésimo recurso y el emésimo recurso 204. Tal navegación puede repetirse para recursos adicionales identificados por identificadores globales adicionales. La navegación por tales identificadores globales puede ser mapeada o representada entonces, y el mapa o gráfico se puede comparar con un mapa o gráfico previo con el fin de detectar cambios.

Las actualizaciones del cliente 112 y/o el servidor 124 no tienen que implementarse en una secuencia particular o en un tiempo concreto, puesto que la API RESTLIKE tiene alta compatibilidad directa e inversa entre clientes y servidores. Así, la funcionalidad soportada por clientes y servidores en la aplicación distribuida de comunicaciones puede evolucionar independientemente sin rotura o inutilización. Este tipo de comportamiento es especialmente adecuado para arquitecturas distribuidas relativamente grandes, donde no es posible redesplegar un entorno de programación completo cada vez que la funcionalidad soportada cambie.

Además, los recursos 202-204 en el servidor 124 pueden ser modificados o cambiados. En un ejemplo, el primer cliente 112 puede pedir un recurso al servidor 124, y el servidor 124 puede generar una respuesta respectiva cada vez que se reciba una petición del primer cliente 112. Sin embargo, las peticiones frecuentes pueden impactar negativamente en las comunicaciones en tiempo real. Consiguientemente, puede usarse una estructura de eventos en conexión con determinar cuándo y cómo el primer cliente 112 es informado acerca de actualizaciones de recursos respectivos, así como cuándo y cómo tales actualizaciones son distribuidas desde el servidor 124 al primer cliente 112. Para ello, el primer cliente 112 puede incluir un componente solicitante de actualización 210 que está

configurado para transmitir una petición al servidor 124 que está configurado para recuperar identificaciones de actualizaciones disponibles a uno o más de los recursos 202-204. En un ejemplo, el servidor 124 puede incluir o tener acceso a un recurso de evento 212, que está configurado para hacer que el servidor 124 identifique categorías asignadas a actualizaciones y transmita identidades de actualizaciones (o actualizaciones propiamente dichas) al primer cliente 112. Por ejemplo, las categorías que pueden asignarse a actualizaciones pueden 1) incluir sensibles a tiempo real; 2) de prioridad alta; o 3) de prioridad baja. Como se describirá con más detalle más adelante, el recurso de evento 212 puede hacer que el servidor 124 transmita inmediatamente actualizaciones sensibles a tiempo real al primer cliente 112, puede hacer que el servidor 124 notifique inmediatamente al primer cliente 112 una actualización de prioridad alta disponible, y puede hacer que el servidor 124 notifique al primer cliente 112 actualizaciones de prioridad baja disponibles cuando alguna de 1) una actualización sensible a tiempo real sea identificada; 2) una actualización de prioridad alta sea identificada; o 3) la petición pendiente del recurso de evento 212 expire (por ejemplo, 30 segundos, 45 segundos, 60 segundos).

El uso de peticiones pendientes reduce el ruido en comunicaciones entre el primer cliente 112 y el servidor 124, puesto que las respuestas del servidor 124 al primer cliente 112 serán menos frecuentes. El componente solicitante de actualización 210 puede estar configurado para transmitir peticiones pendientes periódicamente o a la recepción de una respuesta a una petición pendiente (por ejemplo, de tal manera que al menos una petición para el recurso de evento 212 siempre esté activa cuando el primer cliente 112 esté siendo ejecutado.

Como se ha indicado anteriormente, el servidor 124 puede asignar una categoría respectiva a cada actualización identificada en el recurso de evento 212, donde las categorías incluyen sensibles a tiempo real, de urgencia alta, o de urgencia baja. Cuando el recurso de evento 212 identifica una actualización a la que se le asigna una categoría sensible a tiempo real, el servidor 124 puede generar inmediatamente una respuesta a la petición pendiente del primer cliente 112, donde la actualización está incluida en la respuesta. En respuesta al componente receptor 208 del primer cliente 112 que recibe la respuesta, el componente solicitante de actualización 210 puede transmitir otra petición pendiente (por ejemplo, una GET pendiente) para el recurso de evento 212 al servidor 124. Consiguientemente, las actualizaciones sensibles a tiempo real de recursos son recibidas en el primer cliente 112 en tiempo real, puesto que tales actualizaciones son identificadas en el servidor 124. Además, en respuesta a una actualización de urgencia alta identificada en el recurso de evento 212, el servidor 124 puede generar una respuesta a la petición pendiente recibida del primer cliente 112, donde la actualización de urgencia alta es identificada para el primer cliente 112 en la respuesta, pero la actualización propiamente dicha no está incluida en la respuesta. El primer cliente 112, en respuesta a recibir la identificación de la actualización de urgencia alta, puede pedir la actualización del servidor 124 (por ejemplo, donde la petición identifica el recurso a actualizar). Para actualizaciones clasificadas como de urgencia baja, el recurso de evento 212 puede agregar tales actualizaciones, donde el servidor 124 transmite una respuesta a una petición pendiente cuando la petición pendiente expira o cuando una actualización de urgencia baja se cambia a una actualización de urgencia alta. La respuesta generada por el servidor 124 identifica las actualizaciones disponibles, pero no incluye las actualizaciones en la respuesta.

Con más especificidad, como se ha indicado anteriormente, las actualizaciones disponibles para uno de más de los recursos 202-204 (por ejemplo, actualizaciones de urgencia baja) pueden agregarse en el recurso de eventos 212. Así, cuando uno de los recursos 202-204 es actualizado, el servidor 124 hace que la información perteneciente a tal actualización sea incluida en el recurso de eventos 212. Las actualizaciones ejemplares de los recursos 202-204 incluyen adición de un recurso a los recursos 202-204, modificación de datos en un recurso de los recursos 202-204 (por ejemplo, relativos a un estado anterior del recurso), borrado de un recurso de los recursos 202-204, etc. Así, si el primer recurso 202 es modificado de manera que incluya una referencia al /ésimo recurso, la información perteneciente a tal modificación se incluye en el recurso de evento 212.

Así, el recurso de evento 212 puede incluir una identificación de al menos una actualización (o varias actualizaciones de urgencia baja) que esté disponible para el primer cliente 112. Consiguientemente, el recurso de evento 212 puede incluir una lista de eventos que identifiquen actualizaciones disponibles, un tipo respectivo de actualización para un evento, y un identificador global respectivo para el recurso que se actualiza. Entonces, los datos ejemplares incluidos en el recurso de evento 212, pueden ser como sigue:

Eventos	ID de recurso actualizado	Tipo de actualización
Update1	/messagingapp/user/contactslist	Modificación
Lote de eventos siguiente	/messagingapp/nextevents	Nuevo recurso

Cuando una actualización es clasificada como sensible a tiempo real, es deseable que el primer cliente 112 reciba la actualización tan pronto como sea posible (por ejemplo, en tiempo real). Así, cuando una actualización es clasificada como sensible a tiempo real por el servidor 124, la actualización puede ser incluida en la respuesta transmitida al cliente 112 por el servidor 124 cuando responda a la petición pendiente para el recurso de evento 212 - así, la actualización está disponible para el primer cliente 112 inmediatamente después de que el servidor 124 reconoce la actualización. Igualmente, cuando el recurso de evento 212 incluye datos que indican que una actualización concreta es clasificada como de urgencia alta, el servidor 124 puede responder a una petición pendiente para el

recurso de evento 212 con metadatos que indiquen que la actualización está disponible para el cliente 112 (sin proporcionar la actualización propiamente dicha). La respuesta puede ser distribuida al cliente 112 a través de una toma VoIP, por ejemplo.

Sin embargo, las actualizaciones clasificadas como de urgencia baja no tienen que ser distribuidas al primer cliente 112 de inmediato. En cambio, las actualizaciones de identificación de urgencia baja pueden ser agregadas en el recurso de evento 212 durante una cantidad de tiempo umbral, o hasta que la urgencia de una actualización se cambie (por ejemplo, de urgencia baja a urgencia alta). Tal agregación puede impactar positivamente en las comunicaciones entre el primer cliente 112 y el servidor 124. Por ejemplo, si se añade otro recurso y luego es actualizado, pueden agregarse conjuntamente lo que pueden haber sido dos actualizaciones separadas. Esto no solamente puede simplificar el procesado para el primer cliente 112, dado que el otro recurso solamente tiene que ser pedido y procesado una vez, sino que también el ruido de la aplicación se puede reducir, puesto que múltiples peticiones y/o actualizaciones son reducidas a una sola comunicación para las actualizaciones agregadas.

También se puede apreciar que las actualizaciones de urgencia alta y de urgencia baja también pueden estar embebidas en la respuesta a la petición pendiente. Por ejemplo, el tamaño de una actualización puede ser pequeño y no tendría un impacto negativo en la transmisión de la respuesta. En otro ejemplo, el número total de actualizaciones disponibles puede ser bajo, de tal manera que es más eficiente transmitir la actualización más bien que esperar una petición.

La respuesta del servidor 124 a una petición pendiente del primer cliente 112 puede incluir metadatos que incluyen una indicación de un tipo de la actualización y un identificador global de un recurso que ha sido actualizado. Los metadatos ejemplares en forma de un archivo XML que pueden ser transmitidos desde el servidor 124 al cliente 112 cuando el servidor responde a una petición pendiente para el recurso de evento 212, y una actualización no es una actualización sensible a tiempo real, pueden ser como sigue:

```
<EVENTS>
        <UPDATE1>
                <ID> /messagingapp/user/contactslist </ID>
                <TYPE>modification</TYPE>
        </UPDATE1>
        <NEXTEVENTSID> /messagingapp/nextevents</NEXTEVENTSID>
</EVENTS>
```

Para actualizaciones clasificadas como de urgencia alta o baja, el primer cliente 112 puede analizar la respuesta del servidor 124 e identificar actualizaciones que se recuperen deseablemente del servidor 124. El primer cliente 112 puede transmitir entonces una petición al servidor 124 de al menos una actualización deseada. En respuesta a la petición, el servidor 124 proporciona la actualización al primer cliente 112.

Como se ha indicado anteriormente, el componente receptor 206 del cliente 112 recibe la respuesta del servidor 124, en respuesta a la recepción de la respuesta, el primer cliente 112 transmite una recepción de reconocimiento al servidor 124. En algunos casos, este reconocimiento también puede incluir otra petición pendiente de un lote siguiente de actualizaciones disponibles. Dado que el reconocimiento también incluye una petición pendiente siguiente, el ruido de la comunicación se puede reducir más. Cuando las respuestas transmitidas desde el servidor 124 al primer cliente 112 (excepto las respuestas que incluyen actualizaciones clasificadas como sensibles a tiempo real) incluyen metadatos pero no actualizaciones, el primer cliente 112 puede determinar cuándo y cómo recuperar actualizaciones, y así puede ordenar al servidor 124 que distribuya solamente las actualizaciones que el primer cliente 112 esté preparado para recibir y/o sea capaz de recibir.

El primer cliente 112 también puede incluir un componente decisor 214 que está configurado para determinar un tipo de recuperación de una actualización identificada en una respuesta recibida del servidor 124. El componente decisor 214 puede hacer tal determinación en base al menos en parte a al menos una de una capacidad del primer cliente 112, una urgencia de la actualización, o una condición de tráfico de red. Por ejemplo, el primer cliente 112 puede ser capaz solamente de recibir adecuadamente una cierta cantidad de datos a la vez, y así no puede recibir todas las actualizaciones que estén disponibles; el primer cliente 112 puede diseñarse de manera que reciba un cierto número de actualizaciones en un tiempo; o un alto tráfico de red puede producir retardos de comunicación que impacten en la fidelidad de los datos o produzcan de otro modo problemas de comunicación, cuellos de botella o errores. El tipo de recuperación puede ser uno de recuperación rápida, recuperación lenta, u o recuperación.

Con respecto a recuperación rápida, el componente decisor 214 hace que el componente solicitante de actualización

210 transmita inmediatamente una petición para una actualización identificada como disponible por el servidor 124 al servidor 124 (por ejemplo, una petición para el primer recurso actualizado 202). Cuando el tipo de recuperación es recuperación lenta, también denominada recuperación oportunista, el componente decisor 214 hace que el componente solicitante de actualización 210 transmita una petición para una actualización identificada como disponible por el servidor 124 al servidor 124 cuando se cumpla una condición predefinida. La condición puede ser, por ejemplo, cuando el costo de la recuperación de la actualización sea inferior a un umbral; cuando no entren otras actualizaciones o tráfico; cuando una cierta velocidad de comunicaciones o tasa de bits esté disponible entre el primer cliente 112 y el servidor 124, etc.

El componente decisor 214 también puede determinar que, para un estado actual del primer cliente 112 y/o en un punto de tiempo actual, la actualización identificada en la respuesta recibida del servidor 124 no es necesaria, y no se hace ninguna petición de la actualización (por ejemplo, el tipo de recuperación es no recuperación). Sin embargo, el componente decisor 214 puede estar configurado para poner un señalizador indicando que un recurso concreto (por ejemplo, el primer recurso 202) ha sido actualizado, de tal manera que cuando el primer cliente 112 determine que se desea acceder al recurso, el primer cliente 112 haga que el componente solicitante de actualización 210 transmita una petición de actualización del recurso. En respuesta a recibir una petición de una actualización, el servidor 124 transmite la actualización al primer cliente 112.

Con referencia ahora a la figura 3 se ilustra un sistema ejemplar 300 que facilita realizar una operación asíncrona durante comunicación bidireccional en tiempo real entre el primer cliente 112 y el servidor 124 usando una API RESTLIKE. El sistema 300 incluye el primer cliente 112 que está en comunicación en tiempo real con el servidor 124 usando la API RESTLIKE.

Una operación asíncrona puede incluir al menos un paso de realizar por al menos uno del primer cliente 112 o el servidor 124. Por ejemplo, una operación realizada por el servidor 124 de hacer una llamada a través de una aplicación de comunicaciones de voz que puede ser dirigida a múltiples clientes antes de ser aceptada o declinada puede incluir los hechos de: 1) recibir una petición de hacer la llamada al primer cliente 112; 2) identificar un receptor previsto; 3) identificar clientes del receptor previsto que estén disponibles para recibir la llamada; y 4) completar la petición de llamada. El servidor 124 puede estar configurado para recibir/generar una indicación de que un paso en la operación ha sido realizado o su realización ha fallado. Además, la operación puede modelarse con una estructura que puede hacer que se generen eventos intermedios. Por ejemplo, para que el primer cliente 112 entienda el estado de una llamada (por ejemplo, sonando, conectada o fallida), se sirven deseablemente al primer cliente 112 eventos que indican el estado de la operación al primer cliente 112.

Para ello, el primer cliente 112 incluye el componente solicitante 206, donde el componente solicitante 206 transmite una petición de realizar una operación asíncrona al servidor 124. En respuesta a recibir la petición de realizar la operación asíncrona, el servidor 124 crea un recurso de operación 302. El recurso de operación 302 puede ser usado para modelar la operación asíncrona, y también puede ser modelado después de un recurso transitorio dedicado que sigue a una configuración de empezado, actualizado y completado.

El recurso de operación 302, como se ha descrito anteriormente con respecto a los recursos 202-204, puede ser accedido e interactuado usando órdenes incluyendo métodos tales como los métodos HTTP “GET”, “PUT”, “POST” y “DELETE”. Sin embargo, el recurso de operación 302 también puede estar configurado para hacer que el servidor 124 añada (en este ejemplo) el primer recurso 202, modifique el primer recurso 202 o borre el primer recurso 202. Tal adición, modificación o borrado se basa, al menos en parte, en un paso en la operación asíncrona. Por ejemplo, en un ejemplo donde el primer cliente 112 desea hacer una llamada a un contacto de un usuario del segundo dispositivo informático de cliente 104, la petición puede ser transmitida al servidor 124, que crea el recurso de operación 302 en respuesta a recibir la petición. El recurso de operación 302 puede crear, por ejemplo, el primer recurso 202 que sea un recurso de “dispositivos disponibles”, que incluye datos que indican los dispositivos del contacto que son capaces de recibir la llamada.

El recurso de operación 302 también está configurado para hacer que el servidor 124 transmita eventos que representen la evolución de estados de la operación asíncrona hasta el último éxito o fallo de la operación asíncrona. Cuando la operación asíncrona efectúa una transición hacia la terminación, el recurso de operación 302 transmite actualizaciones/eventos de urgencia alta al recurso de evento 212, y el servidor 124 puede transmitir una respuesta que identifica una actualización de urgencia alta al primer cliente 112 como se ha descrito anteriormente. Por ejemplo, cuando la petición de hacer una llamada ha sido recibida por el servidor 124, el servidor puede hacer que actualizaciones de prioridad alta sean distribuidas al primer cliente 112 con el fin de indicar que la llamada se está haciendo, y que se ha detectado un dispositivo disponible. Otras actualizaciones identificadas por el recurso de operación 302 pueden incluir: que la operación está comenzando; que la operación está en curso; y que la operación ha concluido. Tales tipos de actualizaciones también pueden incluir información adicional acerca del estado de la operación. En un ejemplo, cuando el tipo de actualización es que la operación ha concluido, el tipo de actualización también puede incluir información de que la operación, por ejemplo, tuvo éxito o falló, y las razones del éxito o fallo de la operación.

El recurso de operación 302 también puede hacer que el primer recurso 202 sea añadido, modificado o borrado del

servidor 124. El servidor 124 puede hacer que tal actualización del primer recurso 202 sea transmitida al recurso de evento 212 y, como las actualizaciones del recurso de operación 302, la actualización es clasificada como de urgencia alta. Categorizando actualizaciones pertenecientes a la operación asíncrona como de urgencia alta, los metadatos acerca de la actualización son transmitidos inmediatamente al primer cliente 112, como se ha descrito anteriormente. Así, el primer cliente 112 se mantiene informado acerca del estado de la operación asíncrona sin tener que esperar a que se complete la operación asíncrona.

El cliente 112 incluye además el componente solicitante de actualización 210 que está configurado para enviar una petición pendiente al servidor 124 para el recurso de evento 212, como se ha descrito anteriormente. Cuando es apropiado, el servidor 124 genera una respuesta a la petición pendiente, de la manera expuesta anteriormente con respecto a la descripción del sistema 200 en la figura 2. Como se ha explicado anteriormente, los metadatos en la respuesta pueden indicar que el primer recurso 202 ha sido actualizado. El componente solicitante de actualización 210 puede transmitir entonces una petición para la actualización del primer recurso 202 (por ejemplo, una petición para el primer recurso actualizado 202) al servidor 124 usando la API RESTLIKE. Así, cuando el recurso de operación 302 ha hecho que el primer recurso 202 sea añadido, modificado o borrado, al primer cliente 112 se le notifica dicha actualización y puede pedir dicha actualización cuando lo desee.

En respuesta a que el componente solicitante de actualización 210 pide la actualización del primer recurso 202, el servidor 124 puede transmitir tal actualización, y el componente receptor 208 del primer cliente 112 puede recibir tal actualización (y, consiguientemente, el primer cliente 112 es actualizado). Se deberá entender que cuando el recurso de operación 302 hace que otros recursos sean actualizados, tal como el primer recurso 202, tales actualizaciones son servidas al cliente 112 antes de completar la operación asíncrona, independientemente de si la operación completa satisfactoriamente o sin éxito la operación. Haciéndolo así, las actualizaciones de recursos son distribuidas antes de que cualesquiera recursos dinámicos asociados con la operación asíncrona sean borrados.

Las figuras 4-10 ilustran metodologías ejemplares relativas a comunicación bidireccional asíncrona y síncrona en tiempo real entre dispositivos informáticos mediante el uso de una API RESTLIKE. Aunque las metodologías se representan y describen como una serie de hechos que son realizados en una secuencia, se ha de entender y apreciar que las metodologías no quedan limitadas por el orden de la secuencia. Por ejemplo, algunos hechos pueden tener lugar en un orden diferente al aquí descrito. Además, un hecho puede tener lugar simultáneamente con otro hecho. Además, en algunos casos, puede no ser necesario que todos los hechos implementen una metodología aquí descrita.

Además, los hechos aquí descritos pueden ser instrucciones ejecutables por ordenador que pueden ser implementadas por uno o varios procesadores y/o almacenadas en un medio o medios legibles por ordenador. Las instrucciones ejecutables por ordenador pueden incluir una rutina, una subrutina, programas, un hilo de ejecución, y/o análogos. Además, los resultados de hechos de las metodologías pueden almacenarse en un medio legible por ordenador, visualizarse en un dispositivo de visualización, y/o análogos.

Con referencia ahora a la figura 4 se ilustra una metodología ejemplar 400 que facilita la comunicación asíncrona y síncrona bidireccional en tiempo real entre dispositivos informáticos mediante el uso de una API RESTLIKE. La metodología 400 empieza en 402, y en 404 se ejecuta una instancia de lado de cliente de una aplicación en un dispositivo informático de cliente, donde la aplicación soporta comunicación síncrona en tiempo real y comunicación asíncrona en tiempo real con un segundo dispositivo informático mediante el uso de una API RESTLIKE. En 406, la instancia de lado de cliente comunica, en tiempo real, de forma síncrona y/o asíncrona con el servidor usando la API. La metodología 400 termina en 408.

Con referencia ahora a la figura 5 se ilustra otra metodología ejemplar 500 ejecutada en un primer dispositivo informático que facilita la comunicación, en tiempo real, de forma síncrona y asíncrona, con un segundo dispositivo informático (usando una API RESTLIKE). La metodología 500 empieza 502, y en 504, una petición para un recurso en el servidor es transmitida al servidor, donde la petición incluye un identificador global para el recurso. En 506, los datos pertenecientes al recurso son recibidos del servidor, donde los datos pertenecientes al recurso pueden ser un archivo XML, una actualización para el recurso, etc. La metodología 500 concluye en 508.

Con referencia ahora a la figura 6 se ilustra otra metodología ejemplar 600 ejecutada por un primer dispositivo informático (cliente) que facilita la comunicación bidireccional en tiempo real con un segundo dispositivo informático (por ejemplo, un dispositivo informático de servidor). La metodología 600 empieza en 602, y en 604, una petición pendiente incluyendo un identificador global es transmitida desde el dispositivo informático de cliente al segundo dispositivo informático, identificando el identificador global un recurso en el servidor. En 606 se reciben los metadatos pertenecientes a una actualización del recurso. Los metadatos pueden identificar una actualización disponible para el recurso, o pueden incluir la actualización del recurso. La metodología 600 finaliza en 618.

Con referencia ahora a la figura 7 se ilustra otra metodología ejemplar 700 ejecutada por un primer dispositivo informático (cliente) que facilita la comunicación con un segundo dispositivo informático (servidor), por medio de una API RESTLIKE, bidireccionalmente, sincrónicamente y de forma asíncrona, y en tiempo real. La metodología 700 comienza en 702, y en 704 se transmite una petición pendiente incluyendo un identificador global de un recurso al

segundo dispositivo informático, donde el identificador global identifica el recurso. En 706 se recibe del primer dispositivo informático una respuesta a la petición pendiente, donde la respuesta incluye metadatos que identifican una actualización disponible del recurso. En 708 se determina un tipo de recuperación para recuperar la actualización, donde el tipo de recuperación puede ser una de recuperación rápida, recuperación lenta, o no recuperación. En 710 se transmite una petición de la actualización al segundo dispositivo informático según el tipo de recuperación determinado en 710. En 712 se recibe la actualización del servidor, y la metodología 700 finaliza en 714.

Con referencia ahora a la figura 8 se ilustra una metodología ejemplar 800 que es ejecutada por un primer dispositivo informático (servidor) que facilita la comunicación bidireccional asíncrona y síncrona en tiempo real entre el primer dispositivo informático y un segundo dispositivo informático (cliente) por medio de una API RESTLIKE. La metodología 800 comienza en 802, y en 804 se recibe una petición pendiente para un recurso de evento, donde el recurso de evento es retenido en un medio de almacenamiento legible por ordenador. En 806, el dispositivo informático de servidor determina que un recurso (distinto del recurso de evento) ha sido actualizado. En 808, los datos pertenecientes a la actualización del recurso son enviados al recurso de evento, y en 810, la actualización se clasifica como de urgencia alta. En 812, el dispositivo informático de servidor transmite una respuesta a la petición pendiente al dispositivo informático de cliente, incluyendo la respuesta metadatos que identifican que la actualización está disponible. En 814, después de que el dispositivo informático de servidor transmite la respuesta en 812, el dispositivo informático de servidor recibe una petición del dispositivo informático de cliente para la actualización. En 816, el dispositivo informático de servidor transmite la actualización al dispositivo informático de cliente, y la metodología 800 finaliza en 818.

Con referencia ahora a la figura 9 se ilustra otra metodología ejemplar 900, ejecutada en un dispositivo informático de servidor, que facilita la comunicación bidireccional en tiempo real entre el dispositivo informático de servidor y el dispositivo informático de cliente. La metodología 900 comienza en 902, y en 904 se recibe una petición para un recurso en el dispositivo informático de servidor. Por ejemplo, la petición puede incluir un identificador global para el recurso. En 906 se transmite una respuesta a la petición al cliente, donde la respuesta incluye un paquete de datos pertenecientes al recurso (por ejemplo, datos formateados en una forma que pueda ser analizada por el dispositivo informático de cliente). En 908 se recibe una petición pendiente del dispositivo informático de cliente, identificando la petición pendiente un recurso de evento que se usa para supervisar actualizaciones realizadas en el recurso. En 912, se transmite al cliente una respuesta a la petición pendiente, donde la respuesta incluye metadatos pertenecientes al recurso. La metodología 900 finaliza en 914.

Con referencia ahora a la figura 10 se ilustra una metodología ejemplar 1000 ejecutada en un dispositivo informático de cliente que facilita la realización de una operación asíncrona con un dispositivo informático de servidor (por ejemplo, mediante el uso de una API RESTLIKE). La metodología 1000 empieza en 1002, y en 1004 se transmite una petición de realización de una operación asíncrona al servidor. En un ejemplo, la petición puede hacer que el servidor cree un recurso de operación para la operación asíncrona pedida. En 1006 se transmite una petición pendiente para un recurso de evento al dispositivo informático de servidor. En 1008, para cada paso de la operación asíncrona representada en el recurso de operación que se realiza en el dispositivo informático de servidor, se recibe una notificación respectiva de actualización de urgencia alta del dispositivo informático de servidor. En 1010, los metadatos que identifican una actualización disponible para otro recurso se reciben del dispositivo informático de servidor. Por ejemplo, la actualización disponible puede ser una actualización producida por la realización de un paso de la operación. En 1012 se transmite una petición al dispositivo informático de servidor para la actualización, y en 1014 se recibe la actualización en el dispositivo informático de cliente. La metodología 1000 finaliza en 1016.

Con referencia ahora a la figura 11 se ilustra una ilustración de nivel alto de un dispositivo informático 1100 que se puede usar según los sistemas y metodologías aquí descritas. Por ejemplo, el dispositivo informático 1100 puede ser usado en un sistema que facilite comunicación bidireccional asíncrona y síncrona en tiempo real usando una API RESTLIKE. Como otro ejemplo, el dispositivo informático 1100 puede ser usado en un sistema que facilite la realización de una operación asíncrona usando una API RESTLIKE. El dispositivo informático 1100 incluye al menos un procesador 1102 que ejecuta instrucciones almacenadas en una memoria 1104. Las instrucciones pueden ser, por ejemplo, instrucciones para implementar la funcionalidad descrita como realizada por uno o varios componentes explicados anteriormente o instrucciones para implementar uno o varios de los métodos descritos anteriormente. El procesador 1102 puede acceder a la memoria 1104 por medio de un bus de sistema 1106. Además de almacenar instrucciones ejecutables, la memoria 1104 también puede almacenar una instancia de la aplicación, una cache local, un recurso o una representación de un recurso, actualizaciones, metadatos pertenecientes a actualizaciones, o notificaciones o indicaciones de actualizaciones.

El dispositivo informático 1100 incluye además un almacenamiento de datos 1108 que es accesible por el procesador 1102 por medio del bus de sistema 1106. El almacenamiento de datos 1108 puede incluir instrucciones ejecutables, datos o metadatos pertenecientes a ejecutar un cliente o servidor de una aplicación, recursos, instrucciones API o métodos, etc. El dispositivo informático 1100 también incluye una interfaz de entrada 1110 que permite que dispositivos externos comuniquen con el dispositivo informático 1100. Por ejemplo, la interfaz de entrada 1110 puede ser usada para recibir instrucciones de un dispositivo informático externo, de un usuario, etc. El dispositivo informático 1100 también incluye una interfaz de salida 1112 que está en interfaz con el dispositivo

informático 1100 con uno o más dispositivos externos. Por ejemplo, el dispositivo informático 1100 puede presentar texto, imágenes, etc, por medio de la interfaz de salida 1112.

Se contempla que los dispositivos externos que comunican con el dispositivo informático 1100 mediante la interfaz de entrada 1110 y la interfaz de salida 1112 puedan estar incluidos en un entorno que proporcione sustancialmente cualquier tipo de interfaz de usuario con la que un usuario pueda interactuar. Los ejemplos de tipos de interfaz de usuario incluyen interfaces gráficas de usuario, interfaces naturales de usuario, etc. Por ejemplo, una interfaz gráfica de usuario puede aceptar entrada de un usuario que emplee dispositivo(s) de entrada tales como un teclado, ratón, control remoto, o análogos y proporcionar salida en un dispositivo de salida tal como una pantalla. Además, una interfaz natural de usuario puede permitir que un usuario interactúe con el dispositivo informático 1100 sin las limitaciones impuestas por dispositivos de entrada tales como teclados, ratones, controles remotos, y análogos. Más bien, una interfaz natural de usuario puede basarse en reconocimiento de voz, reconocimiento de toque y lápiz, reconocimiento de gestos tanto en pantalla como junto a la pantalla, gestos al aire, seguimiento de la cabeza y los ojos, voz y habla, visión, toque, gestos, inteligencia de máquina, etc.

Además, aunque se ha ilustrado como un solo sistema, se ha de entender que el dispositivo informático 1100 puede ser un sistema distribuido. Así, por ejemplo, varios dispositivos pueden estar en comunicación por medio de una conexión de red y pueden realizar colectivamente tareas descritas como realizadas por el dispositivo informático 1100.

Varias funciones aquí descritas pueden implementarse en hardware, software, o cualquier combinación de los mismos. Si se implementan en software, las funciones pueden almacenarse o transmitirse como una o varias instrucciones o código en un medio legible por ordenador. Los medios legibles por ordenador incluyen medios de almacenamiento legibles por ordenador. Un medio de almacenamiento legible por ordenador puede ser cualquier medio de almacenamiento disponible al que pueda acceder un ordenador. A modo de ejemplo, y no de limitación, tal medio de almacenamiento legible por ordenador puede incluir RAM, ROM, EEPROM, CD-ROM u otro almacenamiento en disco óptico, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda ser usado para llevar o almacenar código de programa deseado en forma de instrucciones o estructuras de datos y al que se puede acceder con un ordenador. Disco, en el sentido en que se usa aquí, incluye disco compacto (CD), disco láser, disco óptico, disco digital versátil (DVD), disco flexible, y disco blu-ray (BD), donde los discos reproducen en general datos magnéticamente y los discos reproducen en general datos ópticamente con láseres. Además, una señal propagada no está incluida dentro del alcance de medio de almacenamiento legible por ordenador. El medio legible por ordenador también incluye medios de comunicación incluyendo cualquier medio que facilite la transferencia de un programa de ordenador de un lugar a otro. Una conexión, por ejemplo, puede ser un medio de comunicación. Por ejemplo, si el software es transmitido desde una página web, servidor, u otra fuente remota usando un cable coaxial, cable de fibra óptica, par trenzado, línea digital de abonado (DSL), o tecnologías inalámbricas como infrarrojos, radio, y microondas, entonces el cable coaxial, cable de fibra óptica, par trenzado, DSL o tecnologías inalámbricas tales como infrarrojos, radio y microondas se incluyen en la definición de medio de comunicación. Las combinaciones de los anteriores también se deberán incluir dentro del alcance de medios legibles por ordenador.

Alternativamente, o además, lo funcionalmente descrito aquí puede ser realizado, al menos en parte, por uno o varios componentes lógicos de hardware. Por ejemplo, y sin limitación, los tipos ilustrativos de componentes lógicos de hardware que pueden ser usados incluyen matrices de puertas programables in situ (FPGAs),circuitos integrados específicos de programa (ASICs), productos estándar específicos de programa (ASSPs), Sistemas de sistema en chip (SOCs), dispositivos lógicos programables complejos (CPLDs), etc. El módulo procesador y/o una combinación del módulo procesador y una memoria asociada aquí descrita pretenden abarcar cualquiera de tales componentes lógicos de hardware.

Lo descrito anteriormente incluye ejemplos de una o varias realizaciones. Naturalmente, no es posible describir cada modificación y alteración concebibles de los dispositivos o metodologías anteriores al objeto de describir dichos aspectos, pero los expertos en la técnica pueden reconocer que muchas más modificaciones y permutaciones de varios aspectos son posibles. Consiguientemente, se prevé que los aspectos descritos abarquen todas las alteraciones, modificaciones y variaciones que caigan dentro del alcance de las reivindicaciones anexas. Además, en la medida en que se usa el término “incluye” en la descripción detallada o las reivindicaciones, tal término se considera inclusivo de manera similar al término “incluyendo” puesto que “incluyendo” se interpreta en el sentido de un término de transición en una reivindicación.

REIVINDICACIONES

1. Un dispositivo informático (102) incluyendo:

un módulo procesador (108); y

una memoria (110) que incluye una instancia de lado de cliente de una aplicación (112), siendo ejecutada la instancia de lado de cliente de la aplicación por el módulo procesador, soportando la instancia de lado de cliente de la aplicación comunicación asíncrona en tiempo real entre el dispositivo informático y un segundo dispositivo informático (104, 106) usando una interfaz de programa de aplicación (API) que es conforme a una arquitectura de transferencia de estado representacional (REST);

caracterizado porque:

la instancia de lado de cliente de la aplicación también soporta comunicación síncrona en tiempo real entre el dispositivo informático y el segundo dispositivo informático usando la API conforme a la arquitectura REST;

la instancia de lado de cliente de la aplicación incluye un componente solicitante de actualización (210) que transmite una petición pendiente al segundo dispositivo informático para un recurso de evento (212), incluyendo el recurso de evento datos pertenecientes a actualizaciones de recursos (202-204) mantenidos por el segundo dispositivo informático, e incluyendo la petición pendiente un identificador global para el recurso de evento;

la instancia de lado de cliente de la aplicación incluye además un componente receptor (208) que recibe una respuesta a la petición pendiente, donde la respuesta no incluye la actualización, pero incluye metadatos basados en los datos incluidos en el recurso de evento, incluyendo los metadatos una indicación de un tipo de la actualización y un identificador global para el recurso; y

la instancia de lado de cliente de la aplicación está configurada para analizar la respuesta para identificar al menos una de las actualizaciones a recuperar del segundo dispositivo informático.

2. El dispositivo informático de la reivindicación 1, donde la instancia de lado de cliente de la aplicación incluye además un componente decisor (214) que determina un tipo de recuperación para la actualización en base al menos en parte a al menos una de una capacidad del dispositivo informático, una urgencia de la actualización, o una condición de tráfico de red, el componente decisor hace que el componente solicitante de actualización transmita la petición pendiente en base al tipo de recuperación.

3. El dispositivo informático de la reivindicación 2, donde el tipo de la actualización es uno de una adición del recurso en el segundo dispositivo informático, una modificación del recurso, o un borrado del recurso del segundo dispositivo informático.

4. El dispositivo informático de la reivindicación 1, incluyendo además la instancia de lado de cliente de la aplicación un componente solicitante que transmite una petición para realizar una operación asíncrona al segundo dispositivo informático usando la API, donde la petición está configurada para hacer que el segundo dispositivo informático cree un recurso de operación en el segundo dispositivo informático, siendo utilizable el recurso de operación para actualizar un recurso en el segundo dispositivo informático en base a al menos un paso de la operación asíncrona, siendo utilizable además el recurso de operación para hacer que el segundo dispositivo informático actualice un recurso de evento en el segundo dispositivo informático.

5. El dispositivo informático de la reivindicación 1, donde la comunicación asíncrona en tiempo real es al menos uno de una transferencia de archivo, un mensaje electrónico, una petición de programa de evento, una indicación de presencia, una actualización de una lista de contactos, o una actualización de una lista de grupos.

6. El dispositivo informático de la reivindicación 1, donde la comunicación síncrona en tiempo real incluye al menos uno de una transmisión de vídeo, una videoconferencia, una comunicación de voz, o datos pertenecientes a compartición de aplicación.

7. Un método ejecutado por un módulo procesador de un dispositivo informático, incluyendo el método:

ejecutar una instancia de lado de cliente de una aplicación que soporta comunicación asíncrona en tiempo real entre el dispositivo informático y un segundo dispositivo informático por medio de una interfaz de programa de aplicación (API) que es conforme a una arquitectura de transferencia de estado representacional (REST); y

transmitir y recibir asíncronamente datos en tiempo real por medio de la aplicación usando dicha API;

caracterizado porque:

la instancia de lado de cliente de la aplicación también soporta comunicación síncrona en tiempo real entre el dispositivo informático y el segundo dispositivo informático usando la API conforme a la arquitectura REST;

el método incluye además transmitir y recibir asíncronamente datos en tiempo real por medio de la aplicación usando dicha API;

la instancia de lado de cliente de la aplicación incluye un componente solicitante de actualización que transmite una petición pendiente al segundo dispositivo informático para un recurso de evento, incluyendo el recurso de evento datos pertenecientes a actualizaciones de recursos mantenidos por el segundo dispositivo informático, e incluyendo la petición pendiente un identificador global para el recurso de evento;

la instancia de lado de cliente de la aplicación incluye además un componente receptor que recibe una respuesta a la petición pendiente, donde la respuesta no incluye la actualización, pero incluye metadatos basados en los datos incluidos en el recurso de evento, incluyendo los metadatos una indicación de un tipo de la actualización y un identificador global para el recurso; y

la instancia de lado de cliente de la aplicación está configurada para analizar la respuesta para identificar al menos una de las actualizaciones a recuperar del segundo dispositivo informático.

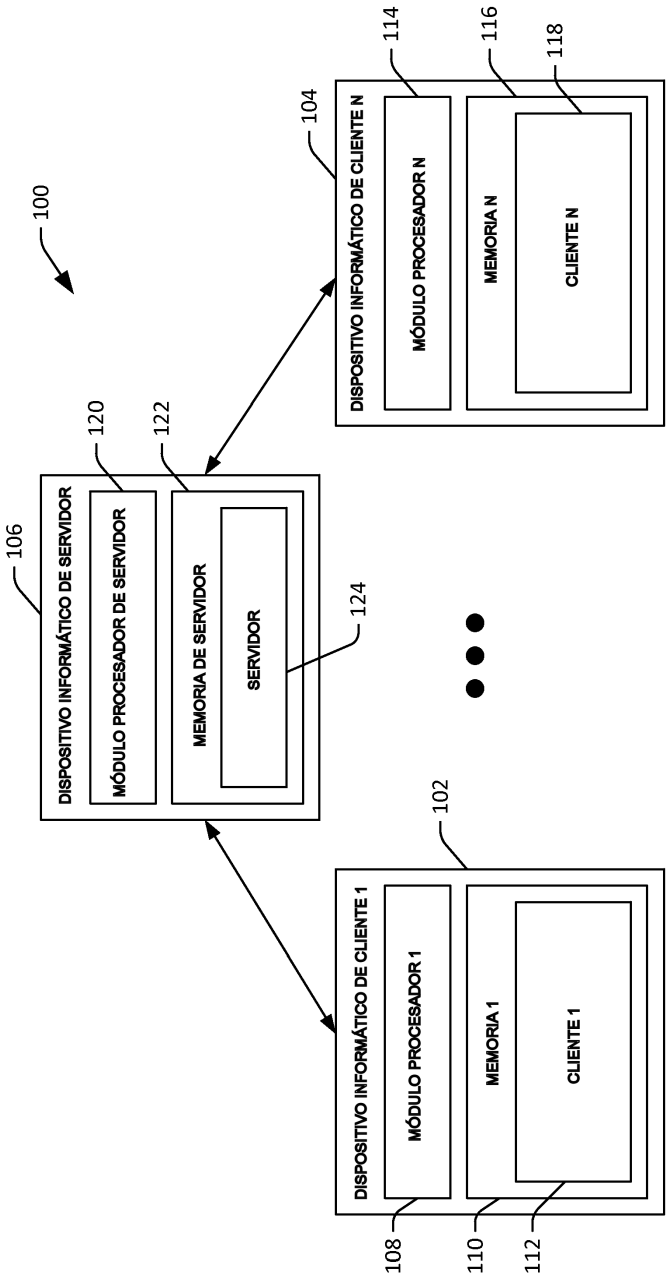
100
DISPOSITIVO INFORMÁTICO DE CLIENTE 1
MÓDULO PROCESADOR 1
MEMORIA 1
CLIENTE 1
102
108
110
112
DISPOSITIVO INFORMÁTICO DE SERVIDOR
MÓDULO PROCESADOR DE SERVIDOR
MEMORIA DE SERVIDOR
SERVIDOR
106
120
122
124
DISPOSITIVO INFORMÁTICO DE CLIENTE N
MÓDULO PROCESADOR N
MEMORIA N
CLIENTE N
104
114
116
118

FIG. 1

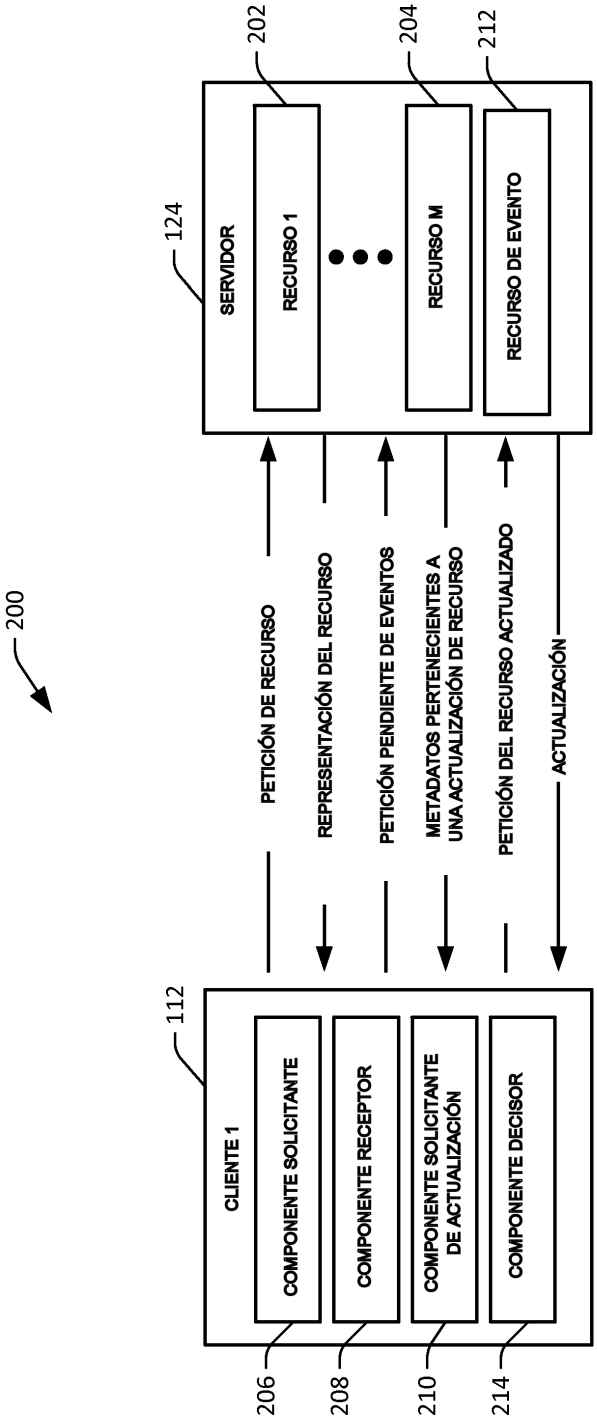
200
124
SERVIDOR
202
RECURSO 1
204
RECURSO M
212
RECURSO DE EVENTO
112
CLIENTE 1
COMPONENTE SOLICITANTE
COMPONENTE RECEPTOR
COMPONENTE SOLICITANTE DE ACTUALIZACIÓN
COMPONENTE DECISOR
206
208
210
214
PETICIÓN DE RECURSO
REPRESENTACIÓN DEL RECURSO
PETICIÓN PENDIENTE DE EVENTOS
METADATOS PERTENECIENTES A UNA ACTUALIZACIÓN DE RECURSO
PETICIÓN DEL RECURSO ACTUALIZADO
ACTUALIZACIÓN

FIG. 2

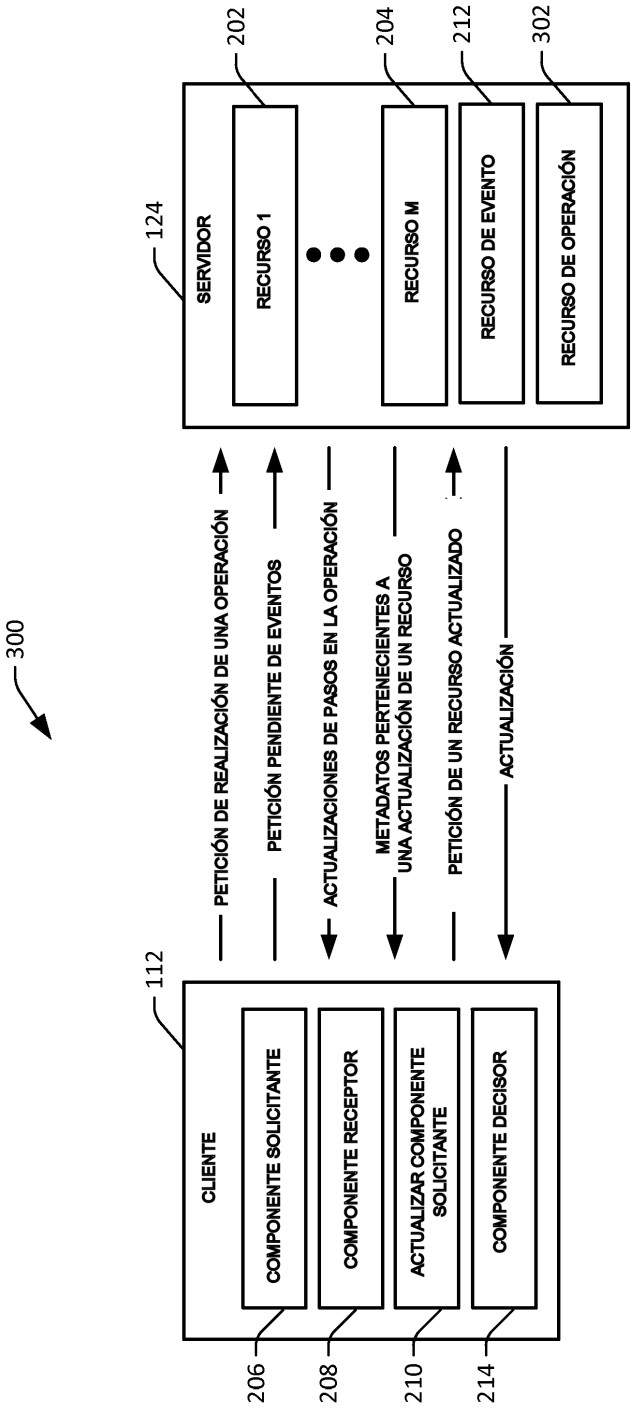
300
112
CLIENTE
COMPONENTE SOLICITANTE
COMPONENTE RECEPTOR
ACTUALIZAR COMPONENTE SOLICITANTE
COMPONENTE DECISOR
206
208
210
214
PETICIÓN DE REALIZACIÓN DE UNA OPERACIÓN
PETICIÓN PENDIENTE DE EVENTOS
ACTUALIZACIONES DE PASOS EN LA OPERACIÓN
METADATOS PERTENECIENTES A UNA ACTUALIZACIÓN DE UN RECURSO
PETICIÓN DE UN RECURSO ACTUALIZADO
ACTUALIZACIÓN
124
SERVIDOR
RECURSO 1
RECURSO M
RECURSO DE EVENTO
RECURSO DE OPERACIÓN
202
204
212
302

FIG. 3

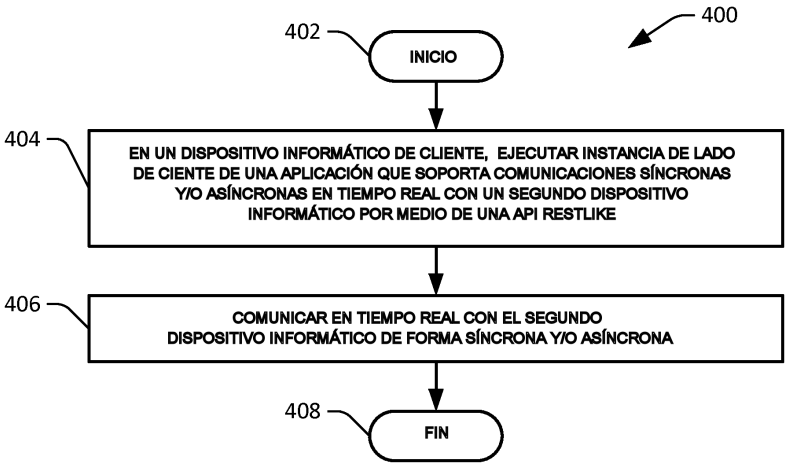
400
402
INICIO
404
EN UN DISPOSITIVO INFORMÁTICO DE CLIENTE, EJECUTAR INSTANCIA DE LADO DE CIENTE DE UNA APLICACIÓN QUE SOPORTA COMUNICACIONES SÍNCRONAS Y/O ASÍNCRONAS EN TIEMPO REAL CON UN SEGUNDO DISPOSITIVO INFORMÁTICO POR MEDIO DE UNA API RESTLIKE
406
COMUNICAR EN TIEMPO REAL CON EL SEGUNDO DISPOSITIVO INFORMÁTICO DE FORMA SÍNCRONA Y/O ASÍNCRONA
408
FIN

FIG. 4

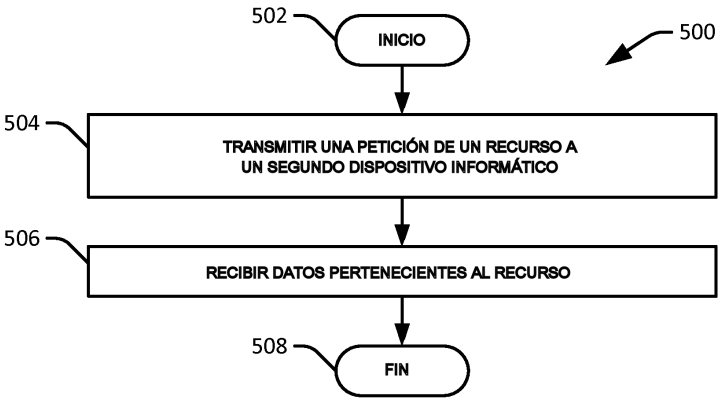
502
INICIO
500
504
TRANSMITIR UNA PETICIÓN DE UN RECURSO A UN SEGUNDO DISPOSITIVO INFORMÁTICO
506
RECIBIR DATOS PERTENECIENTES AL RECURSO
508
FIN

FIG. 5

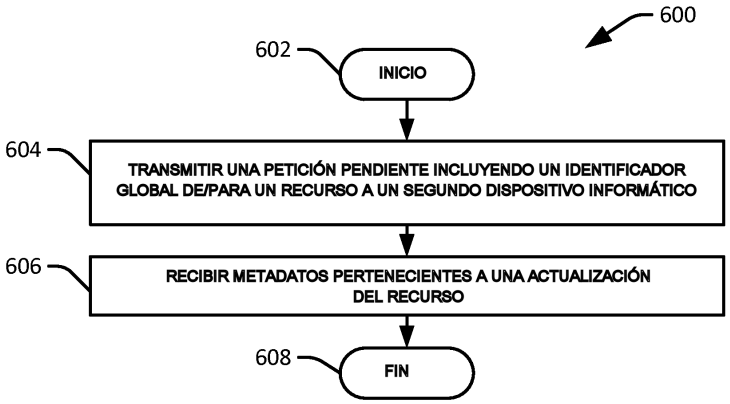
600
602
INICIO
604
TRANSMITIR UNA PETICIÓN PENDIENTE INCLUYENDO UN IDENTIFICADOR GLOBAL DE/PARA UN RECURSO A UN SEGUNDO DISPOSITIVO INFORMÁTICO
606
RECIBIR METADATOS PERTENECIENTES A UNA ACTUALIZACIÓN DEL RECURSO
608
FIN

FIG. 6

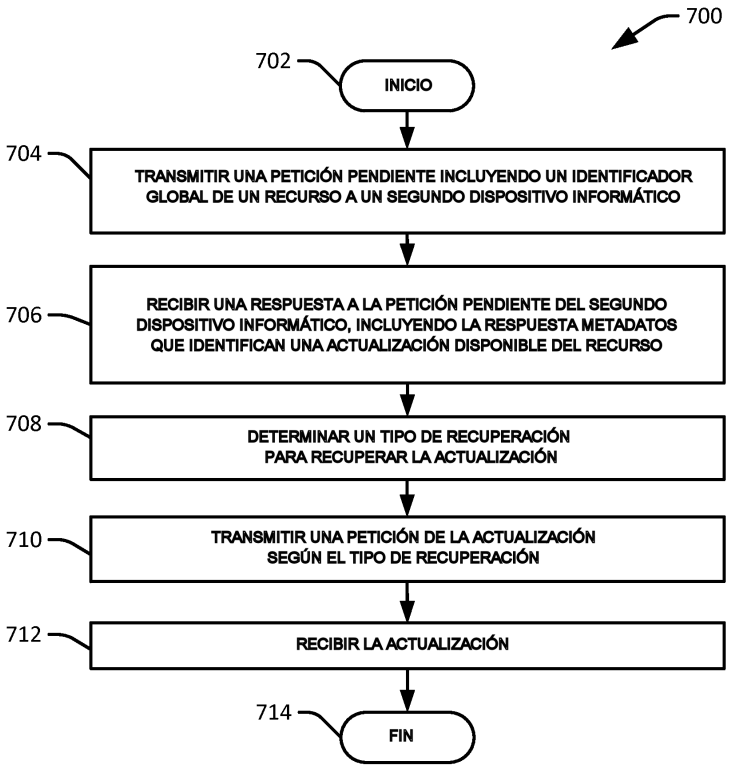
700
702
INICIO
704
TRANSMITIR UNA PETICIÓN PENDIENTE INCLUYENDO UN IDENTIFICADOR GLOBAL DE UN RECURSO A UN SEGUNDO DISPOSITIVO INFORMÁTICO
706
RECIBIR UNA RESPUESTA A LA PETICIÓN PENDIENTE DEL SEGUNDO DISPOSITIVO INFORMÁTICO, INCLUYENDO LA RESPUESTA METADATOS QUE IDENTIFICAN UNA ACTUALIZACIÓN DISPONIBLE DEL RECURSO
708
DETERMINAR UN TIPO DE RECUPERACIÓN PARA RECUPERAR LA ACTUALIZACIÓN
710
TRANSMITIR UNA PETICIÓN DE LA ACTUALIZACIÓN SEGÚN EL TIPO DE RECUPERACIÓN
712
RECIBIR LA ACTUALIZACIÓN
714
FIN

FIG. 7

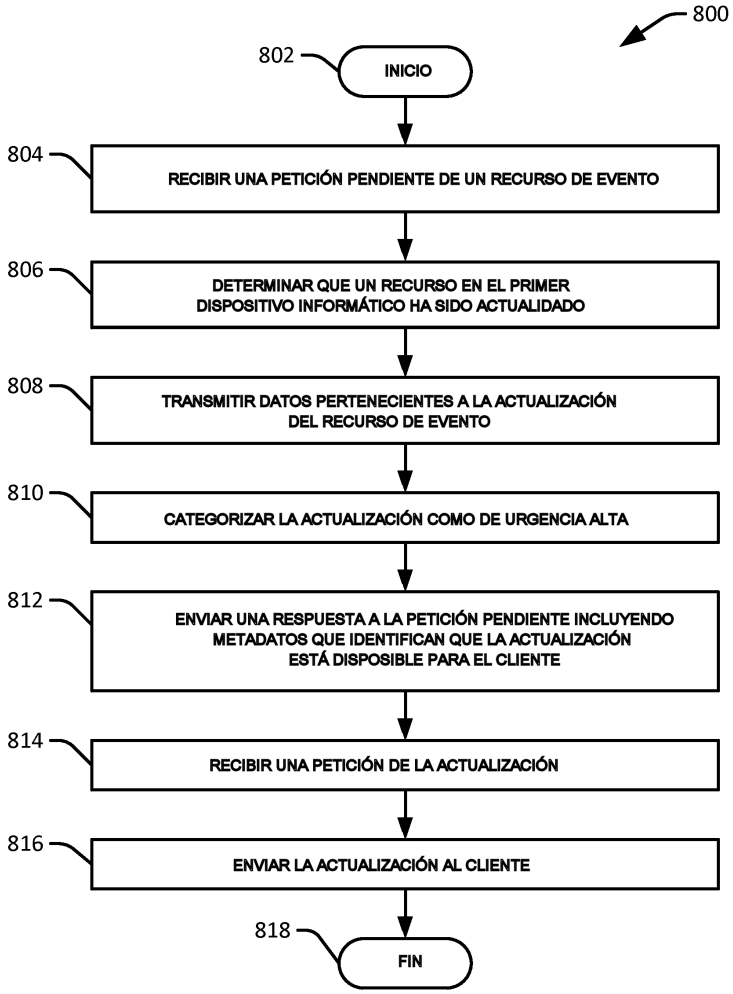
800
802
INICIO
804
RECIBIR UNA PETICIÓN PENDIENTE DE UN RECURSO DE EVENTO
806
DETERMINAR QUE UN RECURSO EN EL PRIMER DISPOSITIVO INFORMÁTICO HA SIDO ACTUALIDADO
808
TRANSMITIR DATOS PERTENECIENTES A LA ACTUALIZACIÓN DEL RECURSO DE EVENTO
810
CATEGORIZAR LA ACTUALIZACIÓN COMO DE URGENCIA ALTA
812
ENVIAR UNA RESPUESTA A LA PETICIÓN PENDIENTE INCLUYENDO METADATOS QUE IDENTIFICAN QUE LA ACTUALIZACIÓN ESTÁ DISPOSIBLE PARA EL CLIENTE
814
RECIBIR UNA PETICIÓN DE LA ACTUALIZACIÓN
816
ENVIAR LA ACTUALIZACIÓN AL CLIENTE
818
FIN

FIG. 8

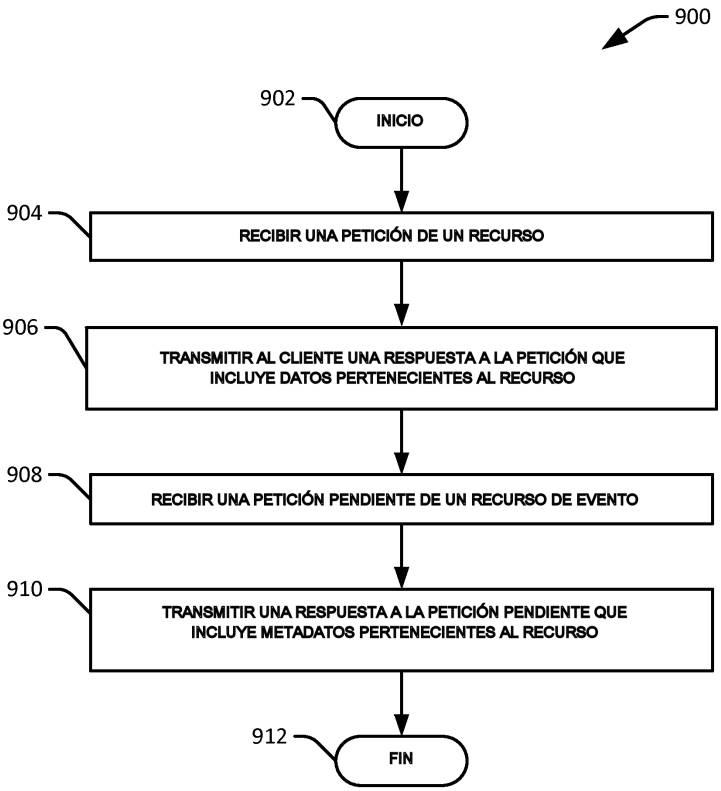
900
902
INICIO
904
RECIBIR UNA PETICIÓN DE UN RECURSO
906
TRANSMITIR AL CLIENTE UNA RESPUESTA A LA PETICIÓN QUE INCLUYE DATOS PERTENECIENTES AL RECURSO
908
RECIBIR UNA PETICIÓN PENDIENTE DE UN RECURSO DE EVENTO
910
TRANSMITIR UNA RESPUESTA A LA PETICIÓN PENDIENTE QUE INCLUYE METADATOS PERTENECIENTES AL RECURSO
912
FIN

FIG. 9

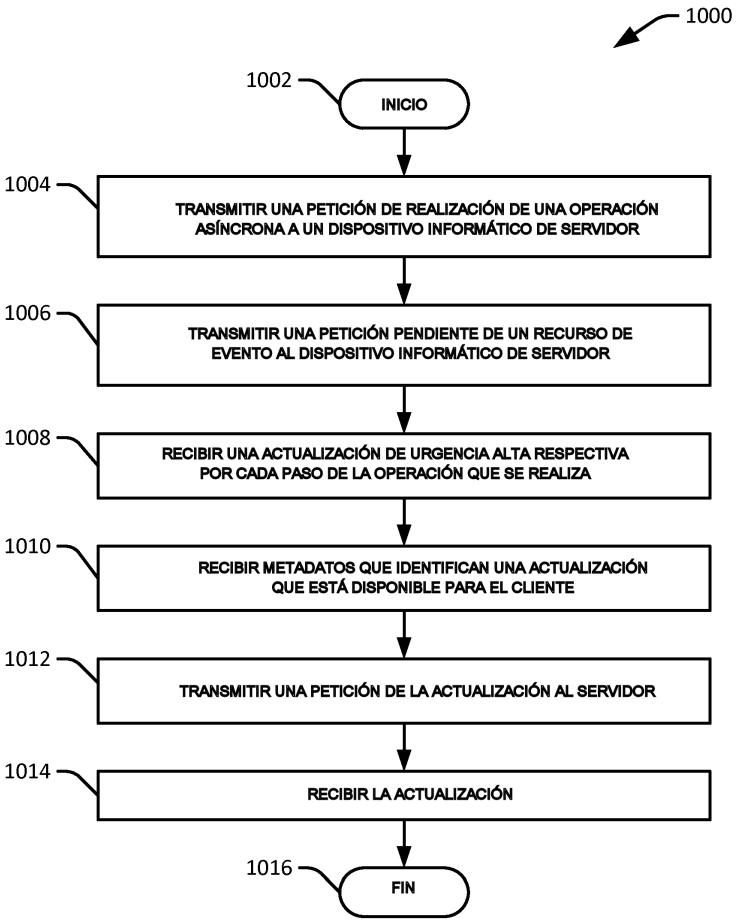
1000
1002
INICIO
1004
TRANSMITIR UNA PETICIÓN DE REALIZACIÓN DE UNA OPERACIÓN ASÍNCRONA A UN DISPOSITIVO INFORMÁTICO DE SERVIDOR
1006
TRANSMITIR UNA PETICIÓN PENDIENTE DE UN RECURSO DE EVENTO AL DISPOSITIVO INFORMÁTICO DE SERVIDOR
1008
RECIBIR UNA ACTUALIZACIÓN DE URGENCIA ALTA RESPECTIVA POR CADA PASO DE LA OPERACIÓN QUE SE REALIZA
1010
RECIBIR METADATOS QUE IDENTIFICAN UNA ACTUALIZACIÓN QUE ESTÁ DISPONIBLE PARA EL CLIENTE
1012
TRANSMITIR UNA PETICIÓN DE LA ACTUALIZACIÓN AL SERVIDOR
1014
RECIBIR LA ACTUALIZACIÓN
1016
FIN

FIG.10

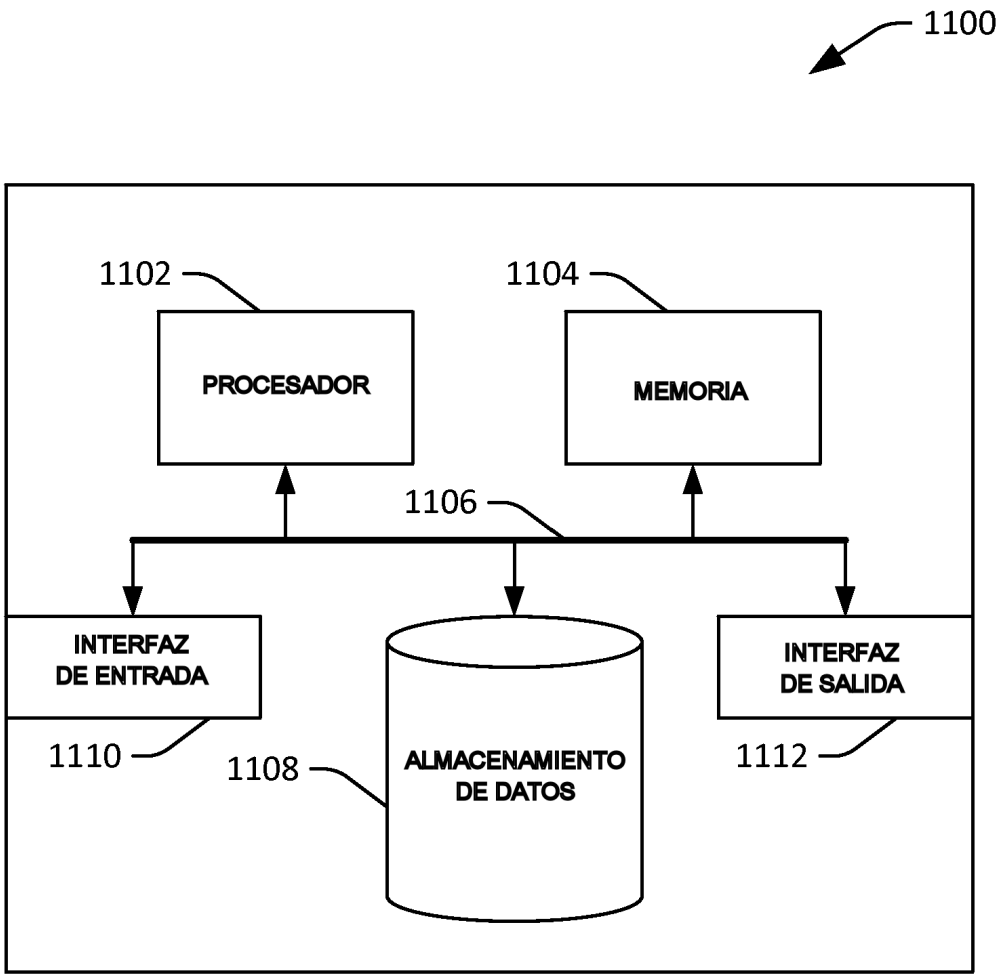
1100
1102
PROCESADOR
1104
MEMORIA
1106
INTERFAZ DE ENTRADA
1110
1108
ALMACENAMIENTO DE DATOS
INTERFAZ DE SALIDA
1112

FIG. 11