(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 435 735**

(21) Número de solicitud: 201231288

(51) Int. CI.:

G06F 9/44 (2006.01)
A63F 9/24 (2006.01)

(12) SOLICITUD DE PATENTE A2

(22) Fecha de presentación:

09.08.2012

(30) Prioridad:

10.08.2011 US 61/521,796
10.05.2012 US 61/645,187

(43) Fecha de publicación de la solicitud:

23.12.2013

(71) Solicitantes:

PLAYTECH SOFTWARE LIMITED (100.0%)
St George's Court, 2nd Floor, Upper Church Street
IM1 1EE Douglas GB

(72) Inventor/es:

GAVISH, Roei

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

(54) Título: **Gestor de widgets**

(57) Resumen:

Gestor de widgets.

Un gestor de widgets para gestionar por lo menos un proceso de widget que está configurado para ejecutarse como una parte de un proceso de cliente, dicho proceso de cliente está configurado para comunicarse con una plataforma de juego a través de un canal de comunicación de cliente, el gestor de widgets está caracterizado por comprender una interfaz de widget para comunicar dicho por lo menos un proceso de widget con un proceso de juego que está configurado para ejecutarse como una parte de dicho proceso de cliente independientemente del proceso de widget, una interfaz de servicio para comunicar dicho por lo menos un proceso de widget con la plataforma de juego y un módulo de procesamiento, el módulo de procesamiento, la interfaz de widget y la interfaz de servicio están acopladas funcionalmente entre sí.

ES 2 435 735 A2

DESCRIPCIÓN

Gestor de *widgets*

Objeto de la invención

Este tema de asunto presentado está relacionado en general con juegos de ordenador y, más particularmente, con la gestión de *widgets* en juegos de ordenador.

Antecedentes de la invención

Los *widgets* son una tecnología cada vez más popular. Los *widgets* o chismes son mini-aplicaciones que soportan un conjunto limitado de información y/o funcionalidad. Un *widget* típico comprende metadatos (por ejemplo, XML), código (por ejemplo, Javascript (secuencia de comandos Java) y una interfaz de usuario (por ejemplo, HTML). Los metadatos proporcionan al *widget* información de configuración tal como una identificación, un nombre, la versión, una descripción y el autor, entre otras cosas. El código especifica la funcionalidad asociada con un *widget* y la interfaz de usuario proporciona un mecanismo para presentar los datos y recibir los datos para y desde un usuario.

Los *widgets* pueden ser implementados en juegos de ordenador de varias maneras conocidas en la técnica como se describe, por ejemplo, en las siguientes solicitudes de patente:

La solicitud de patente de EE.UU. nº. 2011/159966 describe un método que comprende: validar una petición para establecer una introducción de datos entre una máquina de juego de apuestas en un establecimiento de juegos de apuestas y un servidor de una comunidad social en línea contra reglas de importación/exportación de datos de un establecimiento de juegos de apuestas; establecer los datos introducidos por un enlace de datos asociado con la comunidad social en línea y un controlador de importación/exportación de datos del establecimiento de juegos de apuestas según las reglas de importación/exportación de datos; y expurgar unidades de la introducción de datos según las reglas de importación/exportación al tiempo que se mantiene la introducción de datos y se mantiene la ocultación de la máquina de juegos de apuestas del servidor de comunidad social en línea. Una petición que se origina en una comunidad social en línea o un derivado de la comunidad social en línea puede ser un *widget* creado para y/o distribuido desde una comunidad social en línea.

La solicitud de patente de EE.UU. nº. 2011/136569 describe un método y un aparato para generar juegos de apuestas. El método comprende: analizar el código de un juego de apuestas para determinar uno o más activos estéticos, uno o más motores de presentación de juego de apuestas y la lógica de juego del juego de apuestas; generar un código reutilizable ejecutable para cada uno del uno o más activos estéticos para indicar el uno o más activos estéticos en un entorno diferente de un entorno electrónico de máquina de juego de apuestas; generar un código reutilizable ejecutable para cada uno del uno o más motores de presentación de juego de apuestas para implementar el uno o más motores de presentación de juego de apuestas en el entorno diferente del entorno electrónico de máquina de juego de apuestas; y generar un código reutilizable ejecutable que implementa la lógica de juego en el entorno diferente del entorno electrónico de máquina de juego de apuestas.

La solicitud de patente de EE.UU. nº. 2009/249282 describe mecanismos para soportar *widgets* de plataformas cruzadas. Un *widget* genérico puede ser convertido en un *widget* especializado de una plataforma correspondiente. Doblemente, un *widget* especializado puede ser generalizado a un *widget* genérico para el despliegue subsiguiente en el mismo anfitrión o uno diferente. Además, se proporciona soporte para *widgets* composicionales a través de plataformas.

Breve descripción de la invención

Según determinados aspectos del tema de asunto descrito por la presente, se proporciona un gestor de *widgets* . El gestor de *widgets* permite la integración de aplicaciones de *widget* con un proceso de cliente sin la necesidad de realizar modificaciones (del inglés “hack”) o conectar (del inglés “hook”) forzosamente los módulos de cliente o los canales de comunicación cliente-servidor. El gestor de *widgets* permite además utilizar la infraestructura del juego (p. ej. servicios de *back-end*, servicios de almacenamiento y de bases de datos, etc.) para las necesidades de las aplicaciones de *widgets*.

Según determinados aspectos del tema de asunto descrito por la presente, el gestor de *widgets* está configurado para gestionar por lo menos un proceso de *widget* que está configurado para ejecutarse como una parte de un proceso de cliente. El proceso de cliente está configurado para comunicarse con una plataforma de juegos a través de un canal de comunicaciones de clientes. El gestor de *widgets* se caracteriza por comprender una interfaz de *widget* para comunicar el por lo menos un proceso de *widget* con un proceso de juego que está configurado para ejecutarse como una parte del proceso de cliente independientemente del proceso de *widget*, una interfaz de servicio para comunicar el por lo menos un proceso de *widget* con la plataforma de juego y un

módulo de procesamiento. El módulo de procesamiento, la interfaz de *widget* y la interfaz de servicio están acoplados funcionalmente entre sí.
Según aspectos adicionales del tema de asunto descrito por lo presente, la interfaz del servicio está configurada para comunicar dicho por lo menos un proceso de *widget* con la plataforma de juego a través del canal de comunicación de cliente. La interfaz de *widget* está configurada para comunicar el por lo menos un proceso de *widget* con el proceso de juego que utiliza eventos de publicación/suscripción, y en donde el proceso de juego está configurado para actuar como un publicador y el proceso de *widget* está configurado para actuar como un suscriptor. La interfaz del servicio está configurada para comunicar dicho por lo menos un proceso de *widget* y la plataforma de juego y partes de la misma utilizando órdenes de petición/respuesta, y en donde el proceso de *widget* está configurado para actuar como un solicitante y la plataforma de juego y partes de la misma están configuradas para actuar como un ejecutor.

La interfaz de *widget* puede comprender por lo menos una interfaz de programación de aplicación seleccionada de un grupo que comprende: interfaz de programación de aplicación tipo *pull* (extraer) configurada para llamar a determinada funcionalidad; interfaz de programación de aplicación tipo *push* (empujar)configurada para registrar uno o más eventos predefinidos; interfaz de programación de aplicación tipo *get* (obtener) configurada para permitir al proceso de *widget* solicitar información relacionada con el proceso del juego; e interfaz de programación de aplicación tipo *set* (establecer) configurada para permitir al proceso de *widget* transferir ordenes a través de la interfaz de servicio.

La interfaz de servicio puede comprender por lo menos una interfaz de programación de aplicación seleccionada de un grupo que comprende: interfaz de programación de aplicación de órdenes configurada para enviar y recibir órdenes en tiempo real relacionadas con transacciones financieras asociadas con el proceso de *widget*; interfaz de programación de aplicación de canal asegurado configurada para permitir la comunicación con la plataforma de juego y/o partes de la misma a través de un canal seguro; e interfaz de programación de aplicación de canal en tiempo real configurada para permitir la comunicación con la plataforma de juego y/o partes de la misma a través de un canal en tiempo real.

Según un aspecto adicional del tema de asunto descrito por la presente se proporciona un programa informático que comprende medios de código de programa informático que implementa el gestor de *widgets* como se ha descrito anteriormente, en donde dicho programa está en un soporte no transitorio legible por ordenador y configurado para ejecutarse en un dispositivo electrónico programable que ejecuta un proceso de cliente.

Entre las ventajas de determinadas realizaciones del tema de asunto descrito está proporcionar a los desarrolladores de *widgets* de terceros un Juego de Desarrollo de Software (SDK: del inglés *Software Development Kit*) que incluye una Interfaz de Programación de Aplicación (API: del inglés *Application Programming Interface*), permitiendo de este modo la integración con la infraestructura de juego, incluyendo el uso de la unidad de servicios *back-end* para la gestión de usuario y de transacciones financieras relacionadas con las aplicaciones de *widget*.

Breve descripción de los dibujos

Para entender la descripción y para ver cómo puede ser llevada a la práctica, ahora se describirán unas realizaciones, solo a modo de ejemplo no limitativo, haciendo referencia a los dibujos adjuntos, en los que:
La Fig. 1 ilustra un entorno generalizado de juego según determinadas realizaciones del tema de asunto descrito por la presente;

La Fig. 2 ilustra un gestor de *widgets* según determinadas realizaciones del tema de asunto descrito por la presente;

Las Figs. 3a – 3c ilustran esquemáticamente un modelo de Publicación/Suscripción y ejemplos del mismo;
La Fig. 4 ilustra un diagrama de flujo generalizado de un proceso de gestión de eventos según determinadas realizaciones del tema de asunto descrito por la presente; y

La Fig. 5 es un diagrama de flujo generalizado de un proceso de transacción relacionado con un *widget* según determinadas realizaciones del tema de asunto descrito por la presente.

Realización preferente de la invención

En la siguiente descripción detallada, se establecen numerosos detalles específicos con el fin de proporcionar una comprensión profunda del tema de asunto presentado. Sin embargo, los expertos en la técnica entenderán que el tema de asunto presentado puede ponerse en práctica sin estos detalles específicos. En otros casos, no se han descrito con detalle métodos, procedimientos y componentes bien conocidos para no obstruir el tema de asunto presentado. En los dibujos y la descripción, los números de referencia idénticos indican los componentes que son comunes a diferentes realizaciones o configuraciones.

A menos que específicamente se indique de otro modo, como será evidente a partir de las siguientes explicaciones, se aprecia que a través de la memoria descriptiva, las explicaciones que utilizan términos como "procesar", "analizar", "calcular", "emparejar", "generar", "establecer", "configurar" o similares, se refieren a la acción y/o los procesos de un ordenador que manipula y/o transforma datos en otros datos, dichos datos están representados como cantidades físicas, tales como por ejemplo electrónicas y/o dichos datos representan los objetos físicos. El término “ordenador” debería interpretarse ampliamente como que cubre cualquier clase de dispositivo electrónico con capacidades de procesamiento de datos.

Las operaciones según las enseñanzas de esta memoria descriptiva pueden ser realizadas por un ordenador construido especialmente para los propósitos deseados o por un ordenador de uso general configurado especialmente para el propósito deseado por un programa informático almacenado en soporte de almacenamiento legible por ordenador.

Además, las realizaciones del tema de asunto descrito por la presente no se describen haciendo referencia a ningún lenguaje de programación en particular. Se apreciará que puede utilizarse una variedad de lenguajes de programación para aplicar las enseñanzas del tema de asunto descrito por la presente.
Las referencias citadas en los antecedentes enseñan muchos principios de aplicación y gestión de *widgets* en juegos de ordenador que son aplicables al tema de asunto presentado. Por lo tanto el contenido completo de estas publicaciones se incorporan a modo de referencia en esta memoria para las enseñanzas apropiadas de detalles, características y/o antecedente técnico adicionales o alternativos.

Teniendo esto en mente, se dirige la atención a la Fig. 1 que ilustra un entorno generalizado de juego según determinadas realizaciones del tema de asunto descrito por la presente y que muestra un ejemplo no limitativo de una pantalla de cliente que representa una parte de mesa 11 de póquer/casino, interfaz de usuario (UI) 12 del gestor de *widgets* y un contenedor 13 de *widgets*. La UI del gestor de *widgets* está configurada para presentar los *widgets* disponibles y para permitir lanzar, activar y gestionar los mismos. Los *widgets* pueden ser presentados a un jugador como una parte de una tienda de *widgets* accesible a través de la interfaz de usuario de gestor de *widgets*. Los *widgets* pueden ser arrastrados y soltados en la interfaz de barra de *widgets*, clasificados, buscados, valorados, etc.

A modo de ejemplo no limitativo, la UI ilustrada de gestor de *widgets* presenta un *widget* de nube de tema (una aplicación que permite el cambio externo de temas de mesa de iPoker por parte de los jugadores); un *widgets* de mini-juegos (aplicación que proporciona una selección de juegos laterales basados en tecnología *flash*); un *widget* de Back Bet (una aplicación lateral de apuestas que permite a los jugadores poner apuestas laterales en cartas por venir y hacer apuestas de seguro y cercado que juegos de apuestas); y un *widget* de News Feed.
Un *widget* es una mini-aplicación que puede ser personalizada que muestra información actualizada continuamente y permite realizar tareas sin abrir una nueva ventana. Por ejemplo, los *widgets* relacionados con juegos pueden permitir mostrar a un jugador una calculadora de póquer, información de seguimiento, titulares de póquer y otras aplicaciones que resultarán útiles para jugadores o la red con el tiempo. Ejemplos de aplicaciones que no son juegos pueden incluir aplicaciones de música, anuncios, aplicaciones sociales (por ejemplo, charla entre jugadores), aplicaciones de mantenimiento, aplicaciones de internet, aplicaciones no relacionadas con la visualización (es decir, aplicaciones que se ejecutan pero que no muestran contenido), etc.

La interfaz 12 de usuario del gestor de widgets proporciona una manera de organizar los widgets para permitir al jugador acceder rápidamente, sin, por ejemplo, abarrotar la mesa de póquer/casino. La interfaz del gestor de widgets puede situarse en cualquier lado de la mesa de póquer/casino.

Después de la activación un widget puede ser lanzado sobre el contenedor 13 de widgets y ser incorporado en una propia PESTAÑA dedicada, cada widget puede ejecutarse como un proceso independiente.

El entorno ilustrado del entorno de red del juego comprende un cliente 300 acoplado funcionalmente con una plataforma 301 de juego a través de un canal 302 bidireccional de comunicación de cliente. Cualquier dispositivo que tiene capacidad de entrada y visualización (por ejemplo un ordenador personal, estación de trabajo, PDA, teléfono móvil, dispositivo de WebTV, máquina de apuestas, máquina de juego adaptativa, etc.) y capaz de comunicarse con la plataforma 301 de juego directamente o a través de una red de comunicación puede servir como el cliente 300. Opcionalmente, tal dispositivo puede tener además un dispositivo con capacidad de captura de vídeo y/o capacidad de salida de audio. Un proceso que se ejecuta en el cliente en comunicación con las plataformas de juego o partes de las mismas se denomina en lo sucesivo como un proceso de cliente.

En determinadas realizaciones del tema de asunto descrito por la presente, el proceso de cliente comprende un proceso 310 de juego (ilustrado como ejecutándose en la mesa de póquer 11) capaz de ejecutar por lo menos parte de una aplicación de juego. En ciertas realizaciones del tema de asunto descrito por la presente, puede proporcionarse un intercambio de entradas en directo de vídeo/audio entre el proceso de juego que se ejecuta

como una parte del proceso de cliente y la plataforma de juego. Una parte de las respectivas capacidades de representación puede ser reservada para mostrar una imagen de vídeo de los otros jugadores y/o el crupier.
La plataforma 301 de juego comprende un servidor de juego (no se muestra) configurado para recibir datos de entrada de uno o más procesos de juego, para ejecutar la lógica de uno o más juegos en consecuencia, y para informar de los resultados a los procesos del juego según los principios y las reglas del juego. El servidor de juego puede estar configurado como una aplicación de juego del lado del servidor, en donde cada proceso de juego puede estar configurado para ejecutar el correspondiente lado de cliente del juego. La compartición de funciones de juego entre el servidor de juego y los procesos de juego puede variar dependiendo del juego y la implementación del mismo, por ejemplo el proceso de juego puede proporcionar sólo funciones de entrada/salida y/o ejecutar adicionalmente determinados programas relacionados para producir gráficos y/o ejecutar adicionalmente algunos o todos los programas relacionados con una lógica del juego e intercambiar los datos con el servidor de juego, etc. Se apreciará que los servidores y/o los clientes pueden ser implementados alternativamente como cualquier combinación adecuada de software, firmware y hardware.

La plataforma de juego comprende además una unidad 303 de servicios *back-end* acoplados funcionalmente, una unidad 304 de base de datos y de almacenamiento y una unidad 305 de administración de *back-office.* La unidad 304 de base de datos y de almacenamiento está configurada para dar cabida a toda la información necesaria relacionada con los juegos y los usuarios, incluyendo la configuración del servidor de juego (p. ej. juegos disponibles, límites de juego, etc.), los datos y perfiles de usuarios, los datos de gestión de suscripciones y gestión de suscriptores (por ejemplo datos relacionados con la apertura de una cuenta para un usuario, cierre de una cuenta, permitir a un usuario agregar o retirar fondos de una cuenta, cambiar la dirección o el número de identificación personal del usuario, etc.), historias de sesiones, resultados detallados de juegos, transacciones monetarias, datos estadísticos, etc. A modo de ejemplo no limitativo, los datos relacionados con el usuario pueden incluir el nombre del usuario, la dirección, la edad, el género, el estado civil, el número de niños, el salario, la ocupación, las aficiones y las preferencias o cualquier otro dato personal. Adicionalmente, los datos relacionados con el usuario pueden incluir datos relacionados con el juego de determinados usuarios, por ejemplo, número y total de apuestas durante la semana anterior, campos favoritos de juegos, dinero total ganado, bonificaciones, etc.

La unidad de *back-end* 303 está configurada para dar cabida y manejar cuentas de usuario, y para permitir realizar transacciones monetarias según los datos/órdenes recibidos.

La plataforma de juego puede comprender además otros servidores (no se muestran) (por ejemplo servidor de seguridad, servidor de juego en directo, etc.). Estos servidores pueden configurarse para intercambiar datos con la unidad de *back-end*, unidad de *back-office*, unidad de base de datos y de almacenamiento, etc.
Según determinadas realizaciones con el tema de asunto presentado actualmente, al activar un *widget*, el proceso 300 de cliente puede ejecutar además por lo menos un proceso 311 de *widget* (ilustrado como ejecutándose en el contenedor 13 de *widgets*). Los procesos de *widget* se ejecutan independientemente del proceso 310 de juego y son gestionados por un gestor 312 de *widgets*, con la interfaz 12 de usuario de gestor de *widgets* detallada anteriormente.

El gestor de *widgets* está conectado funcionalmente con el proceso 310 de juego a través de un enlace 307 de comunicación de juego y con los procesos 311 de *widgets* a través de un enlace 308 de comunicación de *widget*.
El proceso 310 de juego se comunica con la plataforma 301 de juego (y/o las unidades del mismo) a través del canal 302 de comunicación de cliente de cualquier manera apropiada de comunicación cliente-servidor conocida en la técnica. El gestor 312 de *widgets* se comunica con la plataforma 301 de juego (y/o las unidades del mismo) a través del mismo canal 302 de comunicación de cliente utilizando la conexión de *socket* del proceso de juego.
Opcionalmente, el gestor de *widgets* puede comunicarse con la plataforma de juego (y/o las unidades de la misma) a través de un canal dedicado (no se muestra). Los procesos 311 de *widget* se comunican con el proceso de juego y/o la plataforma de juego (y/o las unidades de la misma) a través del gestor de *widgets*.
Opcionalmente, la plataforma de juego puede comprender además una unidad de gestión de usuarios y/o unidad de servicios de *widget* y/u otras unidades dedicadas a los *widgets*. Como alternativa o adicionalmente, todas o algunas de las unidades comprendidas en la plataforma de juego pueden servir para el juego y para las aplicaciones de *widget*.

De este modo, el gestor de *widgets* permite la comunicación entre el proceso de juego y los procesos de *widget* en ejecución así como la comunicación entre los procesos en ejecución de *widget* y la infraestructura de juego existente (p. ej. unidad de *back-end*, unidad de administración de *back-office*, etc.). La comunicación entre los procesos en ejecución de *widget* y la infraestructura existente de juego puede proporcionarse utilizando la conexión de *socket* (a través del canal 302 de comunicación de cliente) establecido para el proceso de juego.
El gestor 312 de *widgets* proporciona además funciones de control y de seguridad relacionadas con los *widgets*.

A modo de ejemplo no limitativo, tales funciones incluyen:

- Almacenamiento intermedio que permite la verificación de petición de API de *widgets*;
- Proporcionar permisos para peticiones de API (dicho conjunto de API está permitido para cada *widget* específico);
- Impedir la instalación de *widgets* no permitidos;
- Reunir y transferir todas las actividades de *widgets* y de usuario;
- Mantener la descarga de *widgets* publicados en la tienda 306 de aplicaciones de *widgets*;
- Actualizar los *widgets* sin la necesidad de actualizar el proceso de juego;
- Controlar las notificaciones de usuario y otras funciones.
-

Opcionalmente, las funciones de comunicación (o parte de las mismas) del gestor de *widgets* pueden ser implementadas como Interfaces de Programación de Aplicaciones (API) utilizadas para la integración entre un nuevo *widget* y el juego.

Haciendo referencia a la Fig. 2 se ilustra el gestor de *widgets* según determinadas realizaciones del tema de asunto descrito por la presente. El gestor de *widgets* comprende un módulo de procesamiento 403 acoplado funcionalmente con una interfaz 401 de *widget* y una interfaz 402 de servicio. La interfaz 401 de *widget* está configurada para permitir todas las comunicaciones necesarias entre procesos de juego y de *widget* ejecutándose independientemente como partes del proceso de cliente. La interfaz de *widget* se puede conectar con el proceso de juego a través del enlace 307 y con el proceso de *widget* a través del enlace 308. La interfaz de *widget* puede configurarse además para permitir la comunicación necesaria entre los *widgets* y la interfaz de servicio. Opcionalmente, el proceso de *widget* puede comunicarse directamente con la interfaz de servicio, sin participación de la interfaz de *widget*.

La interfaz 401 de *widget* puede comprender un módulo de API de cliente (no se muestra) que soporta opciones de API *push/pull* y *get/set*. A modo de ejemplo no limitativo, el módulo puede incluir:

- API *pull* que funciona preguntando o llamando a determinada funcionalidad (por ejemplo una petición de tamaño actual de pila);
- API *push* que funciona registrando eventos o un oyente que envía regularmente información desde el proceso de juego a los *widgets*. (por ejemplo - escucha la charla del crupier);
- API *get* que funciona permitiendo a los *widgets* para pedir información relacionada con el proceso de juego (por ejemplo obtener el tamaño del bote, obtener el seudónimo de jugador, obtener los movimientos jugados en la última mano, etc.);
- API *set* que funciona permitiendo que los *widgets* transfieran órdenes a través de la interfaz 402 de servicio y el canal 302 de comunicación para actualizar los datos (por ejemplo cargar fondos de $1 o establecer el avatar).
-

La interfaz 402 de servicio está configurada para permitir la comunicación de los procesos de *widget* y/o el gestor de *widgets* con la plataforma de juego (y/o las partes de la misma), y opcionalmente con la tienda 306 de aplicaciones de *widgets* (Fig. 1). La comunicación con la tienda de aplicaciones puede proporcionarse a través de la plataforma de juego o directamente a través de un enlace dedicado (no se muestra).

Entre las funciones permitidas por la interfaz de servicio está:

- proporcionar y gestionar el acceso a la unidad 305 de administración de *back-office*;
- proporcionar y gestionar el acceso a la tienda 306 de aplicaciones de *widgets*;
- gestionar la descarga y actualización de *widgets*;
- permitir el mantenimiento de la información de *widgets* de cuentas de jugadores;
- proporcionar y gestionar el acceso a la unidad de *back-end* 303 y controlar las actividades de micro-transacciones para los *widgets*;
- proporcionar y gestionar el acceso a la unidad 304 de base de datos y de almacenamiento para el uso de *widgets*.
-

La interfaz 402 de servicio puede comprender un módulo de API de servicio (no se muestra). A modo de ejemplo no limitativo, el módulo de API de servicio puede incluir:

- API de órdenes que funciona enviando y recibiendo órdenes en tiempo real hacia y desde la unidad de servicios de *back-end*;
- asegurar el API de canal;
- API de canal en tiempo real y otros.

De este modo, según determinadas realizaciones del tema de asunto descrito por la presente, el gestor de *widgets* puede proporcionar a los desarrolladores de *widgets* de terceros un Juego de Desarrollo de Software (SDK) que incluye una Interfaz de Programación de Aplicación (API), permitiendo la integración con la

infraestructura de juego, incluyendo el uso de la unidad de *back-end* para la gestión de usuario y de transacciones financieras relacionadas con el *widget*.

Cabe señalar que el tema de asunto presentado no está limitado por la arquitectura específica descrita con referencia a las Figs. 1-2, y es igualmente aplicable a cualquier arquitectura de red que facilite el juego en línea por ordenador. El tema de asunto presentado no está limitado por la configuración ilustrada del gestor de *widgets* y/o la plataforma de juego; puede consolidarse una funcionalidad equivalente o puede dividirse de otra manera. En realizaciones diferentes, las unidades y/o las partes funcionales de las mismas pueden colocarse en una única o en múltiples ubicaciones geográficas (incluyendo la duplicación para una alta disponibilidad) conexiones operativas entre las unidades y dentro de las unidades pueden implementarse directa o indirectamente, incluyendo la conexión a distancia. El tema de asunto presentado también puede ser puesto en práctica en entornos distribuidos de ordenador.

El módulo de procesamiento 403 (denominado también en lo sucesivo como “procesador”) comprende una unidad 411 de registro/autorización, una unidad 412 de Administración de Eventos (EventAdmin), unidad 413 de Administración de Órdenes (CommandAdmin) y una unidad 414 de propiedades de sistema; las unidades están conectadas funcionalmente entre sí. El módulo de procesamiento puede comprender además unidades adicionales, como por ejemplo, una unidad de base de datos. Las unidades están configuradas para permitir el funcionamiento aún más detallado haciendo referencia a las Figs. 3-5.

La unidad de registro/autorización 411 se puede hacer funcionar para verificar las peticiones de registro/autorización relacionadas con los *widgets*. La unidad 414 de propiedades de sistema está configurada para permitir a un jugador personalizar las propiedades de aspecto y sensación del *widget* a través de la interfaz 12 de usuario de gestor de *widgets*.

La unidad 412 Administración de Eventos se puede hacer funcionar para permitir un modelo de Publicación/Suscripción de manejo de eventos ilustrada esquemáticamente en la Fig. 3a. La unidad 412 de Administración de Eventos despacha los eventos entre el Publicador 505 de Eventos (proceso de juego) y los Suscriptores 506 de eventos (procesos de *widgets*) interponiendo un canal de eventos. Los publicadores anuncian eventos para el canal y el canal de eventos define a qué manipulador debe notificarse. De este modo, los publicadores y los suscriptores no tienen conocimiento directo uno del otro, lo que simplifica la gestión de eventos.

La unidad 412 de Administración de Eventos maneja una lista de suscripciones que indica una lista de fuentes de datos y *widgets* específicos que se suscriben a eventos asociados con el proceso de juego en ejecución (el publicador de eventos) como una parte del proceso de cliente. Las listas de suscripciones pueden ser especificadas, o pueden ser clasificadas, de modo que algunas fuentes de datos y/o *widgets* soliciten los datos sólo para determinados tipos de eventos. Cada evento puede ser definido por lo menos por dos criterios: un tema de eventos y propiedades de evento. El tipo evento ayuda a emparejar el evento con el proceso apropiado de *widget* (el suscriptor de evento), mientras que las propiedades de evento pueden basarse en cualquiera de varios criterios, incluyendo, pero no limitado a, tipos de aplicaciones, tipos de tema de asunto, número de eventos, momentos del día, tipos de jugador, ajustes de jugador, etc.

Opcionalmente, la unidad 412 de Administración de Eventos puede analizar información asociada con los datos de eventos (por ejemplo, analizar etiquetas descriptivas incorporadas en los datos de eventos, analizar metadatos de eventos, analizar los datos asociados con cuentas de jugador que inician los eventos, etc.) determinar tipos de eventos. La unidad 412 de Administración de Eventos puede proporcionar entonces los datos de eventos a los *widgets* que pueden estar interesados en los tipos determinados de eventos.

La unidad 412 de Administración de Eventos determina qué Suscriptores 506 de Eventos han sido registrados para el tema especificado en el momento en el que el evento ha sido anunciado. Hay dos tipos principales de publicación de los eventos: sincrónico y asíncrono. La publicación asíncrona de eventos se utiliza en el caso de que sea irrelevante para el Publicador 505 de eventos el orden con el que los Suscriptores 506 de Eventos recibirán y procesarán un evento específico. Para eventos asíncronos, la unidad 412 de Administración de Eventos envía notificaciones de los próximos eventos relevantes para un Suscriptor dado 506 de Eventos sin esperar una notificación con éxito de todos los respectivos Suscriptores 506 de Eventos acerca de un evento anterior. Cuando se utiliza publicación sincrónica de eventos, la unidad 412 de Administración de Eventos encuentra a todos los Suscriptores 506 de Eventos abonados a un determinado evento, y a su vez notifica a cada uno. La unidad de Administración de Eventos envía la notificación de un próximo evento sólo después de que cada Suscriptor interesado 506 de Eventos sea notificado con éxito de este determinado evento.

Los diagramas de secuencias de eventos presentados en las Figs. 3b – 3c ilustran esquemáticamente ejemplos no limitativos del modelo de publicación-suscripción. En el proceso de registro ilustrado en la Fig. 3b, el proceso 311 de *widget* (el suscriptor) registra (501) en la unidad 412 de Administración de Eventos para su tema

solicitado (por ejemplo, en eventos relacionados con cartas); y de ahora en adelante escucha (502) los respectivos eventos próximos. En el proceso de publicación ilustrado en la Fig. 3c, el proceso 310 de juego (el publicador) utiliza la unidad de Administración de Eventos para anunciar/enviar (503) eventos. La unidad 412 de Administración de Eventos delega el evento al Suscriptor apropiado (Proceso 311 de *Widget*).

Haciendo referencia ahora a la Fig. 4, se proporciona un diagrama de flujo generalizado del proceso de gestión de eventos. Tras el registro y la autorización con éxito, el proceso 311 de *widget* es suscrito a determinados eventos asociados con el proceso 310 de juego que se ejecuta independientemente como una parte del mismo proceso 300 de cliente. Cuando se produce un evento suscrito, los correspondientes datos de evento son recibidos (601) por la interfaz 401 de *widget* del proceso 310 de juego y se reenvían (602) a la unidad de 412 de Administración de Eventos. La unidad 412 de Administración de Eventos define el tema y las propiedades del evento. Los temas definen el tipo del evento y son utilizados para emparejar (603) el evento con los correspondientes suscriptores de evento (proceso de *widget*) por consiguiente el atributo de propiedades proporciona los datos relacionados con el evento real. Los datos de evento son enviados a todos los procesos de *widget* que están suscritos al tema del evento. En caso de evento asíncrono, la unidad (604) de Administración de Eventos inicia la entrega asíncrona del evento, en donde la entrega de un próximo evento es iniciada independientemente del éxito de la entrega anterior de evento. En caso de evento sincrónico, la unidad (605) de Administración de Eventos inicia la entrega sincrónica del evento, en donde la entrega de un próximo evento es iniciada con la entrega exitosa del evento.

La unidad 413 de Administración de Eventos maneja las órdenes (por ejemplo orden de ejecución de *widget*, peticiones de información, órdenes relacionadas con transacciones financieras asociadas con los *widgets*, etc.) utilizando un modelo de petición-respuesta. La unidad 413 de Administración de Órdenes despacha las órdenes entre Ejecutores de Órdenes y Solicitantes de Órdenes, como se describirá con todo detalle más adelante, interponiendo un canal de órdenes. Los Ejecutores de Órdenes anuncian órdenes por el canal de órdenes, y el canal de órdenes define qué solicitudes se deben ejecutar. De este modo, los publicadores y los manipuladores no tienen conocimiento directo uno del otro.

La unidad 413 de Administración de Órdenes se puede hacer funcionar para permitir la transferencia de órdenes hacia y desde la plataforma de juego a través de la interfaz 402 de servicio (por ejemplo órdenes hacia y desde la unidad de *back-end* 303 para cargar y/o abonar en cada cuenta financiera de usuario).

Haciendo referencia ahora a la Fig. 5 se proporciona un diagrama de flujo generalizado del proceso de transacción según determinadas realizaciones del tema de asunto descrito por la presente. En caso de que una transacción financiera sea iniciada por un proceso de *widget* (solicitante de orden), la correspondiente petición financiera de transacción es recibida (701) por la interfaz 401 de *widget* y es reenviada (702) a la unidad 413 de Administración de Órdenes dentro del módulo de procesamiento 403. La petición de transacción financiera entonces es reenviada (703) a la unidad 303 de servicios de *back-end* (ejecutor de orden) a través de la interfaz 402 de servicio. La unidad 303 de servicios de *back-end* ejecuta (704) la orden de transacción y actualiza la cuenta de balance del cliente en consecuencia. La unidad 303 de servicios de *back-end* envía entonces (705), a través de la interfaz 402 de servicio, datos de confirmación a la unidad 413 de Administración de Órdenes, que reenvía (706) los datos de transacción al respectivo proceso de *widget* a través de la interfaz 401 de *widget*.

Opcionalmente, el gestor de *widgets* puede comunicarse además con un jugador a través de la UI 12, permitiendo de este modo la participación de jugador en el flujo de transacción (por ejemplo para la confirmación de las peticiones de transacción financiera y/o para recibir confirmación de las transacciones e información relacionada con la transacción).

Opcionalmente, el proceso de transacción financiera puede implicar además una unidad de servicios de *widget* (no se muestra). La unidad de servicio de *widget* puede ser una parte del gestor 312 de *widgets* la plataforma 301 de juego y/o una unidad autónoma. Por ejemplo, la unidad de servicios de *widget* puede recibir desde la unidad 303 de servicios de *back-end* y almacenar todos los datos relacionados con la transacción relevantes para los *widgets*.

También se entenderá que el sistema según la invención puede ser un ordenador programado de manera adecuada. Igualmente, la invención contempla un programa informático que es legible por un ordenador para ejecutar la invención. La invención contempla además una memoria legible por máquina que plasma palpablemente un programa de instrucciones ejecutable por la máquina para ejecutar la invención.

Los expertos en la técnica apreciarán fácilmente que pueden aplicarse varias modificaciones y cambios a las realizaciones de la invención como se han descrito anteriormente en esta memoria sin apartarse de su alcance, definido en y por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un gestor (312) de *widgets* para gestionar por lo menos un proceso (311) de *widget* que está configurado para ejecutarse como una parte de un proceso (300) de cliente, dicho proceso de cliente está configurado para comunicarse con una plataforma (301) de juego a través de un canal (302) de comunicación de cliente, el gestor (312) de *widgets* está caracterizado por comprender una interfaz (401) de *widget* para comunicar dicho por lo menos un proceso (311) de *widget* con un proceso (310) de juego que está configurado para ejecutarse como una parte de dicho proceso (300) de cliente independientemente del proceso (311) de *widget*, una interfaz (402) de servicio para comunicar dicho por lo menos un proceso (311) de *widget* con la plataforma (301) de juego y un módulo de procesamiento (403), el módulo de procesamiento (403), la interfaz (401) de *widget* y la interfaz (402) de servicio están acopladas funcionalmente entre sí.

2. El gestor (312) de *widgets* de la reivindicación 1, en donde la interfaz (402) de servicio está configura para comunicar dicho por lo menos un proceso (311) de *widget* con la plataforma (301) de juego a través de dicho canal (302) de comunicación de cliente.

3. El gestor (312) de *widgets* de las reivindicaciones 1 o 2, en donde la interfaz (401) de *widget* está configurada para comunicar dicho por lo menos un proceso (311) de *widget* con dicho proceso (310) de juego que utiliza eventos de publicación/suscripción, y en donde el proceso (310) de juego está configurado para actuar como un publicador (505) y el proceso de *widget* está configurado para actuar como un suscriptor (506).

4. El gestor (312) de *widgets* de cualquiera de las reivindicaciones 1 - 3, en donde la interfaz de servicio está configurada para comunicar dicho por lo menos un proceso (311) de *widget* y la plataforma (301) de juego y partes de la misma utilizando órdenes de petición/respuesta, y en donde el proceso de *widget* está configurado para actuar como un solicitante y la plataforma de juego y partes de la misma están configuradas para actuar como un ejecutor.

5. El gestor (312) de *widgets* de cualquiera de las reivindicaciones 1 – 4, en donde la interfaz (401) de *widget* comprende por lo menos una interfaz de programación de aplicación seleccionada de un grupo que comprende:

- interfaz de programación de aplicación tipo *pull* configurada para llamar a determinada funcionalidad;
- interfaz de programación de aplicación tipo *push* configurada para registrar uno o más eventos predefinidos;
- interfaz de programación de aplicación tipo *get* configurada para permitir al proceso de *widget* solicitar información relacionada con el proceso del juego; y
- interfaz de programación de aplicación tipo *set* configurada para permitir al proceso de *widget* transferir ordenes a través de la interfaz de servicio.

6. El gestor (312) de *widgets* de cualquiera de las reivindicaciones 1 – 5, en donde la interfaz (402) de servicio comprende por lo menos una interfaz de programación de aplicación seleccionada de un grupo que comprende:

- interfaz de programación de aplicación de órdenes configurada para enviar y recibir órdenes en tiempo real relacionadas a transacciones financieras asociadas con el proceso de *widget*;
- interfaz de programación de aplicación de canal asegurado configurada para permitir la comunicación con la plataforma de juego y/o partes de la misma a través de un canal seguro; y
- interfaz de programación de aplicación de canal en tiempo real configurada para permitir la comunicación con la plataforma de juego y/o partes de la misma a través de un canal en tiempo real.

7. Un programa informático que comprende medios de código de programa informático que implementa el gestor (312) de *widgets* según cualquiera de las reivindicaciones 1 – 6, en donde dicho programa está en un soporte no transitorio legible por ordenador y configurado para ejecutarse en un dispositivo electrónico programable que ejecuta un proceso (300) de cliente.

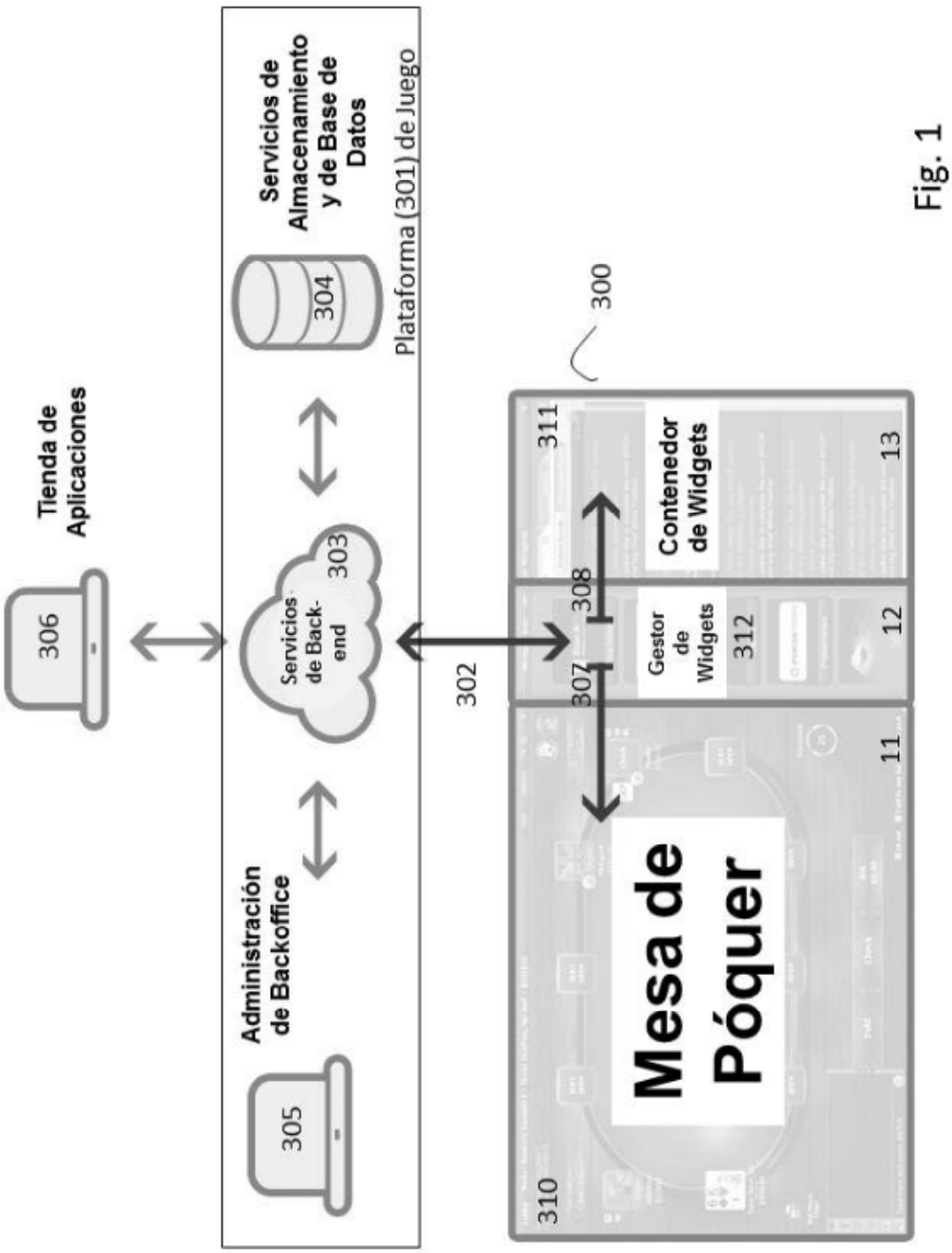
Tienda de Aplicaciones
306
Administración de Backoffice
305
Plataforma (301) de Juego
Servicios de Back-end
303
Servicios de Almacenamiento y de Base de Datos
304
302
300
310
Mesa de Póquer
11
307
308
Gestor de Widgets
312
12
311
Contenedor de Widgets
13

Fig. 1

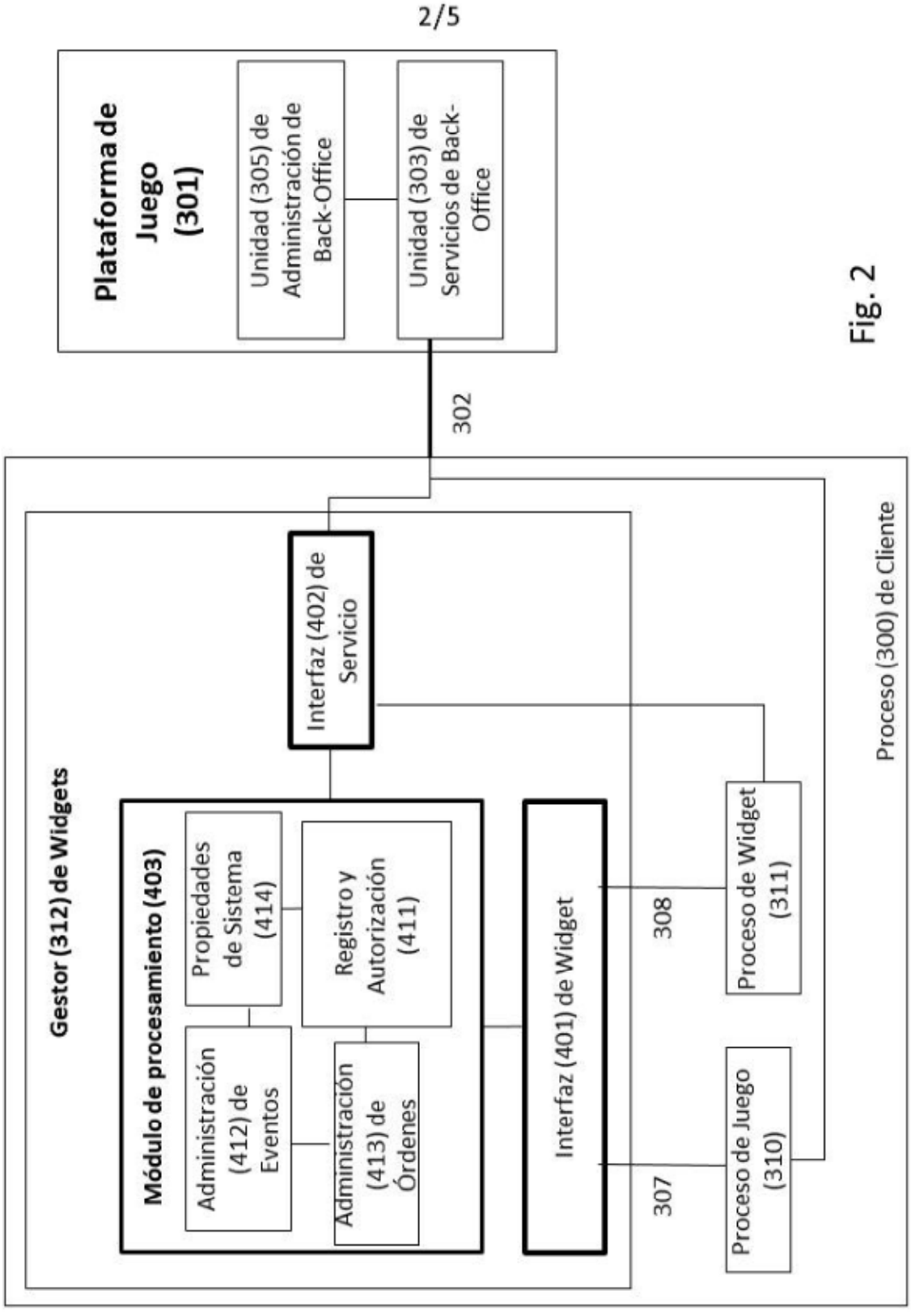
Plataforma de Juego (301)
Unidad (305) de Administración de Back-Office
Unidad (303) de Servicios de Back-Office
302
Proceso (300) de Cliente
Gestor (312) de Widgets
Interfaz (402) de Servicio
Módulo de procesamiento (403)
Propiedades de Sistema (414)
Registro y Autorización (411)
Administración (412) de Eventos
Administración (413) de Órdenes
Interfaz (401) de Widget
308
307
Proceso de Widget (311)
Proceso de Juego (310)

Fig. 2

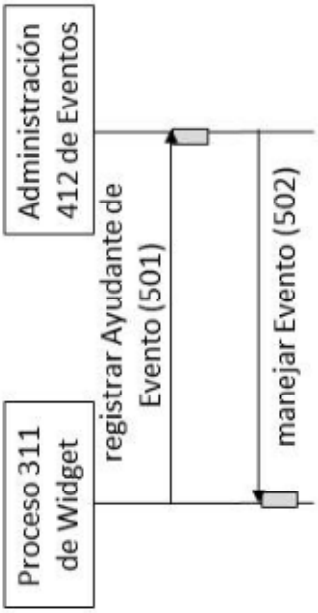

Fig. 3b

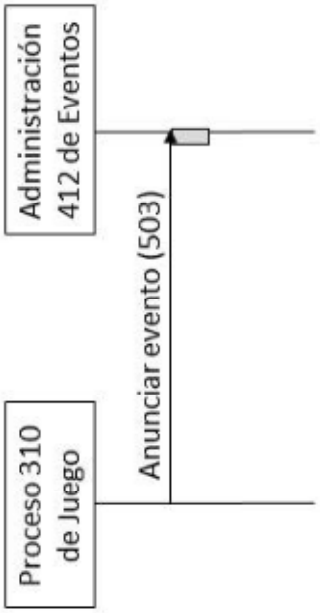

Fig. 3c

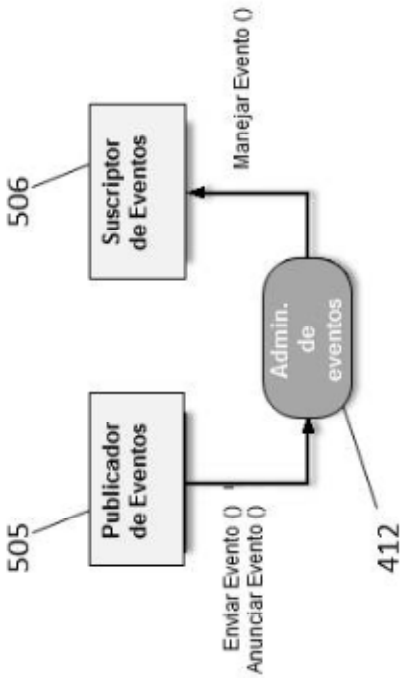

Fig. 3a

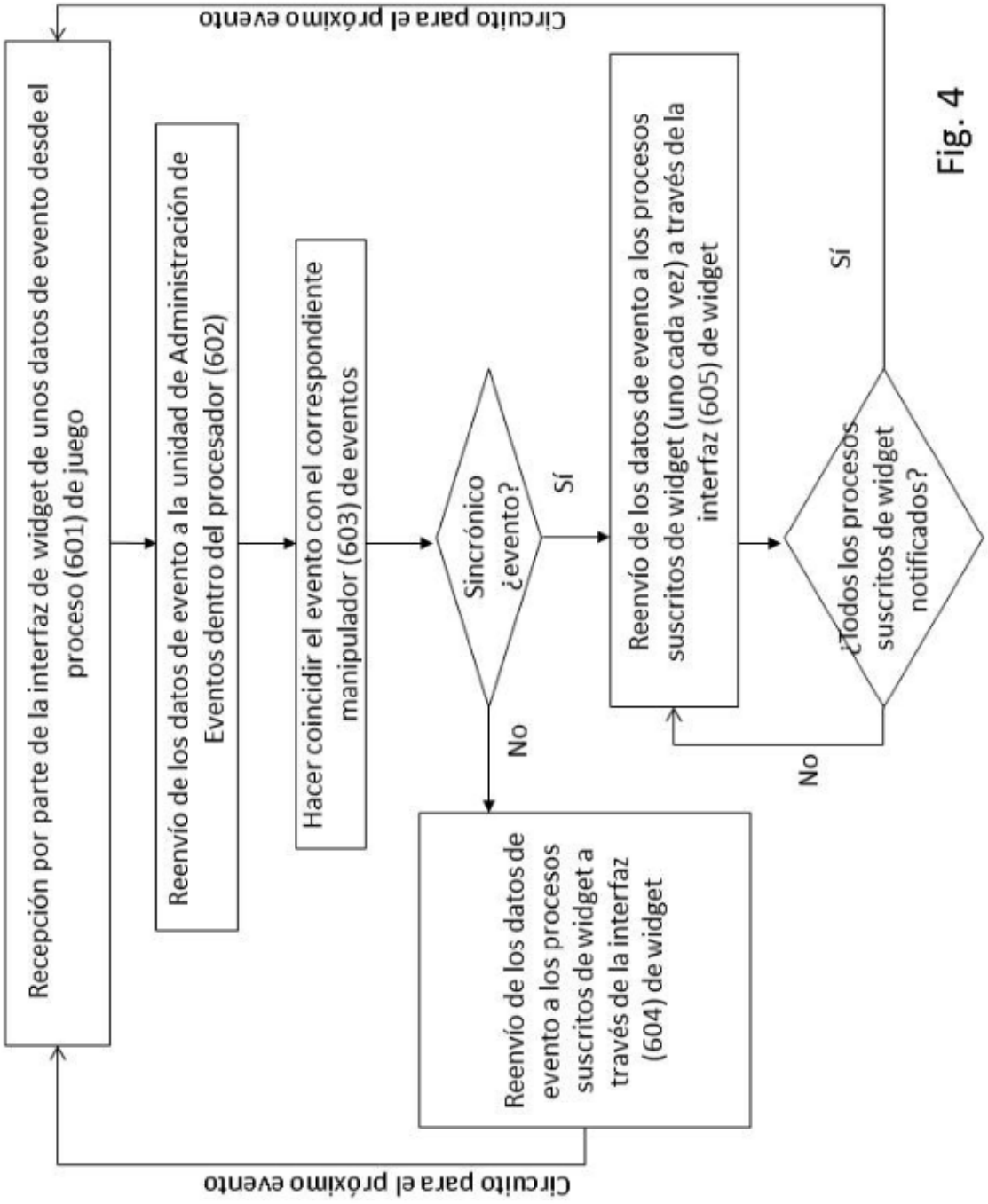
Recepción por parte de la interfaz de widget de unos datos de evento desde el proceso (601) de juego
Reenvío de los datos de evento a la unidad de Administración de Eventos dentro del procesador (602)
Hacer coincidir el evento con el correspondiente manipulador (603) de eventos
Sincrónico ¿evento?
Sí
No
Reenvío de los datos de evento a los procesos suscritos de widget (uno cada vez) a través de la interfaz (605) de widget
¿Todos los procesos suscritos de widget notificados?
Sí
No
Reenvío de los datos de evento a los procesos suscritos de widget a través de la interfaz (604) de widget
Circuito para el próximo evento
Circuito para el próximo evento

Fig. 4

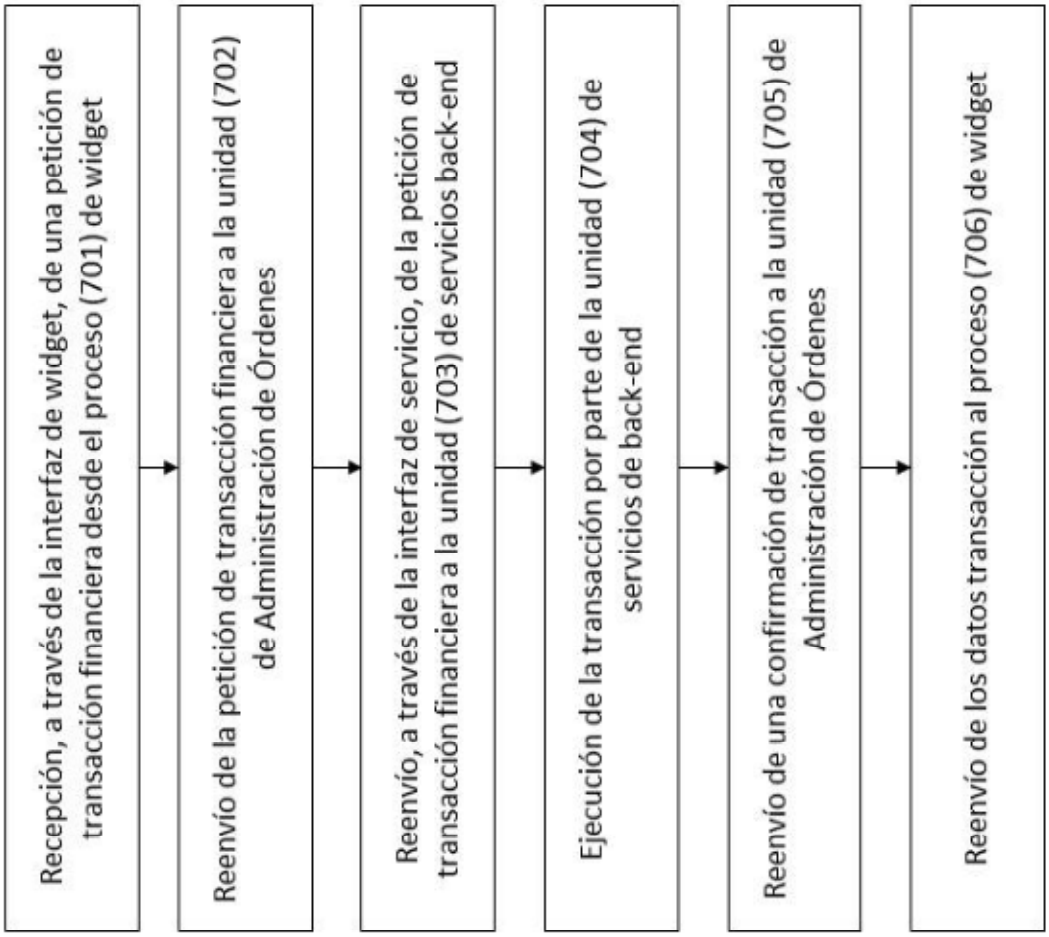
Recepción, a través de la interfaz de widget, de una petición de transacción financiera desde el proceso (701) de widget
Reenvío de la petición de transacción financiera a la unidad (702) de Administración de Órdenes
Reenvío, a través de la interfaz de servicio, de la petición de transacción financiera a la unidad (703) de servicios back-end
Ejecución de la transacción por parte de la unidad (704) de servicios de back-end
Reenvío de una confirmación de transacción a la unidad (705) de Administración de Órdenes
Reenvío de los datos transacción al proceso (706) de widget

Fig. 5