(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 922 476**

(51) Int. CI.:

G06F 9/44 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **31.08.2015 PCT/IB2015/056615**

(87) Fecha y número de publicación internacional: **17.03.2016 WO16038503**

(96) Fecha de presentación y número de la solicitud europea: **31.08.2015 E 15781145 (6)**

(97) Fecha y número de publicación de la concesión europea: **18.05.2022 EP 3191942**

(54) Título: **Editor de aplicaciones web interactivo**

(30) Prioridad:

10.09.2014 US 201462048774 P
22.12.2014 US 201414580172

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
15.09.2022

(73) Titular/es:

MK SYSTEMS USA INC. (100.0%)
251 Little Falls Drive
Wilmington, DE 19808, US

(72) Inventor/es:

UNTER ECKER, OLIVER y
WESTMAN, NICKOLAS

(74) Agente/Representante:

DURAN-CORRETJER, S.L.P

DESCRIPCIÓN

Editor de aplicaciones web interactivo

SECTOR TÉCNICO

Las realizaciones de la invención se refieren al sector técnico de la edición de aplicaciones; y, más específicamente, a un editor de aplicaciones web interactivo.

ESTADO DE LA TÉCNICA ANTERIOR

Una aplicación web es software que habitualmente se ejecuta en un navegador o en otra aplicación de red y se crea utilizando tecnologías que entiende el navegador, tales como lenguaje de marcado de hipertexto (HyperText MarkUp Language, HTML), hojas de estilo en cascada (Cascading Style Sheets, CSS) y lenguajes de secuencia de comandos del lado del cliente (por ejemplo, JavaScript). Estas tecnologías se utilizan comúnmente para crear directamente la interfaz de usuario (UI) de la aplicación web y proporcionar la funcionalidad de la aplicación web.

Las tecnologías de aplicaciones web del lado del cliente no tienen "conjuntos de miniaplicaciones de UI" definidos, tales como los que se encuentran comúnmente en plataformas de aplicaciones nativas, tales como la plataforma Windows, la plataforma Macintosh, etc. En consecuencia, habitualmente es necesario crear/definir los elementos de UI en una aplicación web. Además, no hay ningún estándar para crear/definir elementos de UI para aplicaciones web. Aunque existen marcos de trabajo populares como jQuery y permiten fácilmente el acceso a, y la manipulación del modelo de objetos de documento (Document Object Model, DOM) y de las CSS, dichos marcos de trabajo no proporcionan abstracciones de UI. Por lo tanto, las sutiles diferencias entre navegadores y plataformas (por ejemplo, una plataforma móvil frente a una plataforma de escritorio) requieren un conocimiento y tratamiento específicos con el fin de proporcionar experiencias de usuario satisfactorias. Como resultado de estos inconvenientes, la creación de UI para una aplicación web habitualmente es *ad hoc*, incompleta, engorrosa y, a menudo, imperfecta o carente de funcionalidad, comparada con el desarrollo de UI para entornos de aplicaciones nativas. Por ejemplo, algo tan sencillo conceptualmente como incluir un botón "eficiente" (un botón que responda de forma fiable y con una latencia similar a la nativa cuando es seleccionado por el usuario) es complejo en una aplicación web frente a una aplicación nativa.

Existen editores interactivos para determinadas tecnologías web, tales como HTML, CSS y JavaScript, y se utilizan regularmente en el desarrollo de aplicaciones web. Sin embargo, dichas tecnologías se basan en la comprensión del usuario de la sintaxis de las tecnologías y, por tanto, en general solo son accesibles para o utilizadas por desarrolladores de software expertos en dichas tecnologías web.

IPTV es una plataforma de suministro de multimedia que utiliza una infraestructura de red basada en paquetes (por ejemplo, redes de acceso a internet de banda ancha) para ofrecer contenido de vídeo a los clientes, habitualmente como una alternativa al suministro mediante televisión por el aire tradicional, televisión por satélite y televisión por cable. Una solución de IPTV habitual incluye equipos de adquisición de vídeo de cabecera para recibir contenido de televisión, instalaciones de IPTV intermedias (por ejemplo, Mediaroom™ de Ericsson) que incluyen plataformas de servidor y software intermedio de IPTV asociado, dispositivos de conexión en red (por ejemplo, enrutadores, conmutadores) para la distribución de contenido entre diversos nodos del sistema de IPTV, nodos de acceso (por ejemplo, equipos de línea de abonado digital de muy alta velocidad binaria (VDSL o VHDSL) o de red óptica pasiva (PON)) para permitir el transporte de alto ancho de banda hasta, y desde el emplazamiento de cliente, y aplicaciones proporcionadas por el operador que gestionan el sistema de IPTV y/o proporcionan aplicaciones de IPTV de usuario final.

Los consumidores (también denominados usuarios, usuarios finales, espectadores, clientes o abonados) de servicios de IPTV utilizan interfaces de usuario de aplicaciones proporcionadas por el operador en estaciones finales (tales como descodificadores (set-top boxes, STB), tabletas, teléfonos inteligentes, ordenadores portátiles, ordenadores personales, etc.) para acceder al contenido de IPTV. Sin embargo, estas aplicaciones son difíciles de crear, ya que, a menudo, son sistemas tremendamente complejos que incluyen muchas capas de abstracción y se basan en códigos base personalizados. Además, los operadores también desean proporcionar interfaces de usuario (UI) de aplicación personalizadas para diferentes usuarios o dispositivos, y se ha demostrado que es muy difícil tanto generar como mantener dichas UI personalizadas.

El aspecto y el comportamiento de las UI para aplicaciones web habitualmente se ha predefinido. El código que los define (CSS, para la mayor parte) es tradicionalmente fijo pasado el tiempo de compilación y habitualmente es referenciado estáticamente por una aplicación web. Aunque algunas aplicaciones pueden permitir el "skinning" (cambio de apariencia) con fines de personalización de apps, muchas aplicaciones (por ejemplo, las de grandes compañías) requieren una gran cantidad de personalización, mucho más allá de lo que el "skinning" puede proporcionar, ya que es necesario poder controlar cómo deberían mostrarse y/o comportarse los elementos de interfaz de usuario, quizás de forma radicalmente diferente. Para satisfacer dichos requisitos, algunos desarrolladores de aplicaciones han recurrido a excavar en los códigos base y aplicar (estáticamente) dichas

personalizaciones. Esto normalmente requiere un conocimiento experto del código base, y puede conducir a la creación de muchos errores de programación no intencionados. En consecuencia, existe la necesidad de una solución para modificar aplicaciones web complejas que permita crear experiencias de UI ampliamente diferentes, dependiendo de la preferencia del desarrollador, sin requerir cambios de código base.

Los preprocesadores de hojas de estilo en cascada (CSS) han sido un enfoque utilizado recientemente para construir más fácilmente aplicaciones basadas en tecnología web enriquecidas que se pueden personalizar. Sin embargo, los preprocesadores de CSS habitualmente se diseñan para ejecutarse como parte de una etapa de construcción para producir archivos de CSS estáticos que pueden utilizar entonces directamente los navegadores sin personalización adicional. Adicionalmente, algunos preprocesadores de CSS llevan a cabo todo el análisis sintáctico, la transformación y la sustitución de variables en una etapa computacionalmente costosa en el lado de cliente, degradando, por tanto, el rendimiento y la experiencia de usuario.

La Patente US 2012167041 A1 describe un procedimiento que incluye recibir credenciales durante la ejecución de una aplicación en un dispositivo informático. Las credenciales se evalúan para determinar si un usuario asociado con las credenciales está autorizado para editar la aplicación mientras la aplicación se está ejecutando. Tras determinar que el usuario está autorizado, se permite el control de edición durante la ejecución de la aplicación. El control de edición está asociado con un elemento de interfaz gráfica de usuario (GUI) de la aplicación y se puede manejar para actualizar el elemento de GUI durante la ejecución de la aplicación.

CARACTERÍSTICAS

Un objetivo de la invención es superar las desventajas de la técnica anterior. Este objetivo de la invención se resuelve mediante las reivindicaciones independientes. En las reivindicaciones dependientes se definen realizaciones específicas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La invención se puede comprender mejor haciendo referencia a la siguiente descripción y a los dibujos adjuntos que se utilizan para ilustrar realizaciones de la invención. En los dibujos:

la figura 1 ilustra las etapas en un sistema para la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según realizaciones de la invención;
la figura 2 es un diagrama de flujo que ilustra operaciones de ejemplo para la edición interactiva de una aplicación web, según una realización;
la figura 3 ilustra una interfaz de usuario de ejemplo que muestra la simulación de UI simultáneas de diferentes plataformas, según una realización;
la figura 4 ilustra operaciones de ejemplo realizadas en una estación final de usuario para ejecutar una aplicación web parametrizada que incluye un editor interactivo y para editar la aplicación web, según una realización;
la figura 5 ilustra operaciones de ejemplo para llevar a cabo la recarga de UI en una realización;
la figura 6 ilustra operaciones de ejemplo para llevar a cabo la recarga de UI en otra realización;
la figura 7 ilustra operaciones de ejemplo para la manipulación directa, según una realización;
la figura 8 ilustra una interfaz de usuario de ejemplo que incluye una funcionalidad para que un usuario seleccione un componente disponible predefinido y lo añada a una aplicación web, según una realización;
la figura 9 ilustra una realización de ejemplo para distribuir las ediciones a una aplicación web, según algunas realizaciones;
la figura 10 ilustra una realización de ejemplo para distribuir las ediciones a una aplicación web, según otras realizaciones;
la figura 11 ilustra un flujo en una estación final para generar dinámicamente una aplicación en tiempo de ejecución basándose en un archivo de definiciones de UI, según realizaciones de la invención;
la figura 12 ilustra un diagrama de bloques de un sistema que incluye un sistema de IPTV que utiliza generación dinámica de aplicaciones en tiempo de ejecución basándose en una parametrización de reglas de estilo, según una realización de la invención;
la figura 13 ilustra código de estilo aumentado y código de generación de estilo, según una realización de la invención;
la figura 14A ilustra un código de invocación de generación de estilo que, cuando se utiliza para invocar el código de generación de estilo de la figura 13, da lugar al conjunto ilustrado de reglas de estilo válidas y a la generación de la interfaz de usuario personalizada, según una realización de la invención;
la figura 14B ilustra un código de invocación de generación de estilo adicional que, cuando se utiliza para invocar el código de generación de estilo de la figura 13, da lugar al conjunto adicional ilustrado de reglas de estilo válidas y a la generación de la interfaz de usuario personalizada adicional, según una realización de la invención;
la figura 15 ilustra un flujo en una estación final de servidor para utilizar reglas de estilo parametrizadas para permitir la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según realizaciones de la invención;
la figura 16 ilustra un flujo en una estación final de servidor para utilizar reglas de estilo parametrizadas para permitir la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según algunas

realizaciones de la invención; y
la figura 17 ilustra un flujo en una estación final para utilizar código de generación de estilo generado por una estación final de servidor para la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según realizaciones de la invención.

DESCRIPCIÓN DE LAS REALIZACIONES

En la siguiente descripción, se exponen numerosos detalles específicos. Sin embargo, se comprende que las realizaciones de la invención se pueden llevar a la práctica sin estos detalles específicos. En otros casos, circuitos, estructuras y técnicas bien conocidas no se muestran en detalle con el fin de no oscurecer la comprensión de esta descripción. Los expertos en la materia, con las descripciones incluidas, podrán implementar la funcionalidad apropiada sin demasiada experimentación.

En la presente memoria se utiliza texto entre paréntesis y bloques con bordes de trazos (por ejemplo, trazos grandes, trazos pequeños, punto-raya y puntos) para ilustrar operaciones opcionales que añaden características adicionales a las realizaciones de la invención. Sin embargo, no se debe entender que dicha notación significa que estas son las únicas opciones u operaciones opcionales, y/o que los bloques con bordes continuos no son opcionales en determinadas realizaciones de la invención.

Las referencias en la memoria descriptiva a "una realización", "una realización de ejemplo", etc. indican que la realización descrita puede incluir una función, estructura o característica particular, pero no necesariamente todas las realizaciones incluyen la función, estructura o característica particular. Además, dichas expresiones no se refieren necesariamente a la misma realización. Además, cuando se describe una función, estructura o característica particular en relación con una realización, se entiende que está dentro del conocimiento de un experto en la materia efectuar dicha función, estructura o característica en relación con otras realizaciones, tanto si se describe explícitamente como si no.

En las siguientes descripción y reivindicaciones, se pueden utilizar los términos "acoplado" y "conectado", junto con sus variantes. Se debe comprender que no se pretende que estos términos sean sinónimos entre sí. "Acoplado" se utiliza para indicar que dos o más elementos, que pueden estar o no en contacto físico o eléctrico directo entre sí, colaboran o interactúan entre sí. "Conectado" se utiliza para indicar el establecimiento de comunicación entre dos o más elementos que están acoplados entre sí.

En la presente memoria se describe un editor de aplicaciones web interactivo. Un componente de software (por ejemplo, JavaScript) que representa el editor interactivo se carga dinámicamente en la instancia en ejecución de una aplicación web que se construye a partir de uno o varios componentes de UI. En respuesta a recibir una selección de usuario de una representación visual de un componente en la instancia en ejecución de la aplicación web, se determina el elemento de modelo de objetos de documento (DOM) correspondiente a la selección y se determina el componente de UI correspondiente al elemento de DOM seleccionado, junto con los parámetros para el componente de UI determinado. Los parámetros del componente de UI determinado pueden incluir sus tipos e intervalos, que el editor permite visualizar por medio del editor interactivo sin conocimiento previo de los parámetros del componente de UI. Se muestra al usuario un editor de valores que permite al usuario inspeccionar y/o modificar los valores del o de los parámetros. Después de modificar un valor, suprimir un valor y/o añadir un valor, se activa una actualización de UI y se refleja en la instancia actualmente en ejecución de la aplicación web. Los valores editados de los parámetros de los componentes se pueden conservar. En algunas realizaciones, la actualización de la UI se lleva a cabo sin recargar toda la aplicación web.

Según la invención, el editor interactivo se utiliza en un entorno donde la aplicación web se genera dinámicamente en tiempo de ejecución bajo el control de un archivo de definiciones de UI. Este marco de trabajo de componentes mapea conjuntos de parámetros de nivel superior al de implementación a la estructura, el comportamiento y el aspecto de la aplicación web. Este archivo de definiciones de UI puede ser utilizado flexiblemente por estaciones finales de usuario para, en tiempo de ejecución, generar dinámicamente la aplicación web, o puede ser utilizado por una estación final de servidor para construir código de aplicación listo para su utilización (por ejemplo, código HTML, CSS y/o JavaScript). Según la invención, los archivos de definiciones de UI se utilizan junto con un conjunto de archivos de definiciones de componentes que proporcionan definiciones de componentes que se pueden instanciar en tiempo de ejecución. Cada una de las definiciones de componentes puede incluir un conjunto de estilos visuales y/o comportamientos predeterminados, que se pueden anular/cambiar por medio de las definiciones en el archivo de definiciones de UI respectivo. Los desarrolladores de aplicaciones pueden construir rápida y fácilmente variantes de la aplicación construyendo también archivos de definiciones de UI alternativos que incluyen diferentes definiciones que producirán diferentes aspectos, estructuras y comportamientos de las aplicaciones generadas subsiguientemente. Por ejemplo, se pueden generar diferentes archivos de definiciones de UI para diferentes tipos de acceso a los que diversos usuarios tienen permitido el acceso (por ejemplo, algunos usuarios tienen permitido el acceso a un determinado tipo de contenido, y, por tanto, el archivo de definiciones de UI incluirá definiciones para elementos de UI personalizados para acceder a ese contenido), para diferentes tipos de estaciones finales (por ejemplo, para pantallas mayores o menores), para diferentes tipos de sistemas operativos o aplicaciones de soporte utilizadas por los usuarios, etc. El archivo de definiciones de UI se puede editar por medio de la utilización del editor

interactivo, tal como se describe en la presente memoria.

En algunas realizaciones, los procedimientos, aparatos y sistemas descritos en la presente memoria se pueden utilizar para generar diversas aplicaciones basadas en tecnología web, tales como aplicaciones utilizadas por usuarios de sistemas de IPTV. En algunos sistemas de IPTV, el código de aplicación con el que interactúa un usuario (por ejemplo, la aplicación que se ejecuta en un STB, tableta, teléfono inteligente, ordenador personal, etc.) es código para acceder a contenido de IPTV. Estas aplicaciones, aunque habitualmente fueron creadas originalmente por un proveedor de tecnología de IPTV, a menudo están altamente personalizadas por los operadores de IPTV. Utilizando realizaciones de la invención, un operador de IPTV solo tiene que interactuar con uno o varios archivos de definiciones de UI para generar y/o editar una o varias aplicaciones personalizadas para sus usuarios. En algunas realizaciones, el editor de aplicaciones web interactivo permite la personalización de la aplicación web por personal no técnico o incluso por usuarios finales; permite que la aplicación web se construya desde cero; permite que los diseñadores realicen cambios directos en la aplicación web sin, o con asistencia y soporte limitados al desarrollador, permite el diseño "directo del producto" que elimina etapas intermedias propensas a errores desde el diseño del producto hasta la implementación técnica, y permite el desarrollo multipantalla o multiplataforma simultaneo.

La figura 1 ilustra etapas en un sistema 100 para la generación dinámica en tiempo de ejecución de una aplicación web en tiempo de ejecución bajo el control de un archivo de definiciones de UI, según realizaciones de la invención. El sistema 100 incluye tres etapas independientes, pero interrelacionadas: una etapa 102 de creación y ejecución de aplicación, una etapa 104 de edición de aplicación y una etapa 108 de gestión de ciclo de vida de componentes.

La etapa 104 de edición de aplicación permite que un usuario entre en un modo de edición (o modo de diseño) 124 para crear o editar un archivo de definiciones de UI. Esto puede implicar que el usuario lance el editor interactivo para editar el archivo de definiciones de UI. El usuario puede comenzar la edición como un proceso semicontinuo 126, en el que se produce una edición interactiva 106 donde los resultados/el efecto de una edición se pueden presentar al usuario, tal como se describirá con más detalle más adelante en la presente memoria. Esta edición interactiva 106 puede incluir uno o varios de selección de componentes 128 (por ejemplo, seleccionar los componentes a editar), posicionamiento 130 (por ejemplo, definir dónde en la UI se va a colocar el componente), edición de parámetros (por ejemplo, cambiar o establecer un valor para el componente de UI seleccionado que afecta a su estructura, comportamiento y/o aspecto) y/o instanciación 134 de componentes (por ejemplo, cargar un componente definido para que el operador lo observe y/o interactúe con la versión definida actualmente de un componente). Simultáneamente o después de modificar el archivo de definiciones de UI, el archivo de definiciones de UI se puede guardar en la estación final de usuario que lleva a cabo la edición o en una estación final de servidor.

La etapa 102 de creación y ejecución de aplicación incluye la adquisición 114 de un archivo 122 de definiciones de UI desde una estación final de servidor o la propia estación final de usuario 120. En realizaciones de la invención, el archivo de definiciones de UI incluye instrucciones de un lenguaje de definición de aplicaciones web (Webapp Definition Language, "WADL"). WADL es un lenguaje que permite la descripción de conjuntos de componentes de aplicaciones web, sus conexiones (por ejemplo, comportamientos e interacciones) y conjuntos de parámetros específicos del personalizador para la generación automatizada de aplicaciones web. En una realización, WADL sirve como la parte de "definición externa" de un sistema mayor para la creación programática de aplicaciones web. En una realización, WADL es un alias de JSON y, por tanto, se transporta fácilmente como una carga útil de XMLHttpRequest (XHR), se escribe por máquina y se lee por máquina fácilmente y es puede ser escrita por un humano y leída por un humano. La utilización de WADL de definiciones de tipo permite, además, comprobaciones de coherencia interna y un mapeo eficiente a tipos internos calculables. Además, la utilización de WADL de variables permite referencias compartidas a una definición, y una comodidad añadida para las personas que escriben definiciones, y las fases de JavaScript permiten un gran grado de expresividad con un "coste" mínimo (para el implementador o estudiante del lenguaje).

El archivo de definiciones de UI puede incluir muchas secciones que detallan componentes a incluir en la aplicación (es decir, la estructura de la aplicación web, el aspecto de la aplicación y el comportamiento de los componentes dentro de la aplicación). En una realización, el archivo de definiciones de UI también incluye uno o varios de un identificador de versión, un conjunto de valores de atributo predeterminados "a nivel de aplicación" y un conjunto de declaraciones de componentes. Una parte de un archivo de definiciones de UI de ejemplo se proporciona a continuación como la Tabla A:

Tabla A - Archivo de definición de UI de ejemplo

```
{
  "version": "0,8",
  "defaults": {
    "container": {
      "backgroundColor": "$dark_backgroundColor"
    },
    "actionButtons": {
      "button": {
        "type": "Size",
        "width": "70px",
        "height": "60px"
      },
      "button_backgroundColor": "$brand_color",
      "in_animation": "none",
      "out_animation": "none"
    }
  },
  "theme": {
    "dark_backgroundColor": "#111111",
    "brand_color": "rgba(100, 50, 30, 0,7)",
    "content_minSize": {
      "type": "Size",
      "width": 1024,
      "height": 786
    },
    "body_font": "\"Segoe UI\", sans-serif"
  },
  "components": [
    {
      "class": "views.Details",
      "root": ".details",
```

```
      "components": [
        {
          "class": "actionButtons",
          "layout": {
            "type": "horizontal",
            "contains": "actionButton",
            "controls": "actionBar"
          }
        }
      ]
    },
    {
      "class": "views.YourStuff",
      "type": "container",
      "params": {
        "feeds": [subscriber:Continue, subscriber:Pins,
subscriber:Rentals],
        "resetToFirstTabAndItemOnReEnter": true
      },
      "components": [
        {
          "class": "views.DetailsPane",
          "type": "details_pane",
          "root": "#detailsPanel",
          "params": {
            "metadataPos": {
              "type": "Position",
              "top": "72px",
              "right": "140px"
            }
          }
        },
        {
          "class": "actionButtons",
          "root": "#actionButtons>.actions",
          "params": {
            "barPos": {
              "type": "Position",
              "top": "60px",
              "right": "10px"
            },
            "button": {
              "type": "Size",
              "width": "100px",
              "height": "72px"
            },
            "button_spacing": "4px",
            "in_animation": "slideInFromRight 0,5s both
0.25s",
            "out_animation": "slideOutToRight 0,5s both"
          },
          "layout": {
            "type": "vertical",
            "contains": "actionButton",
            "controls": "actionBar"
```

```
            }
          },
          {
            "class": "views.Filmstrip",
            "params": {
              "item_gap": "4px",
              "backgroundColor": "rgba(0,0,0,0.40)"
            }
          }
        ]
      }
    ]
}
```

Volviendo a la figura 1, la etapa 102 de creación y ejecución de aplicación, además de utilizar el archivo 122 de definiciones de UI, también incluye el análisis sintáctico del archivo de definiciones de UI para identificar los componentes necesarios, y el registro de estos componentes en un marco de trabajo 116. El marco de trabajo tiene provisiones para la instanciación de componentes en tiempo de ejecución, manteniendo la visibilidad de determinados componentes de contenedor de nivel superior (por ejemplo, apilamiento de diálogos), y la búsqueda de componentes y de parámetros de componentes. Por ejemplo, y como se describirá con más detalle más adelante en la presente memoria, el marco de trabajo proporciona una función para realizar una búsqueda desde un nodo de DOM asociado a un componente de la aplicación web en ejecución y una función para encontrar/enumerar los parámetros del componente. En este punto, comienza 118 la etapa 108 de gestión de ciclo de vida de componentes.

Para cada componente y cada subcomponente que se puede definir mediante el archivo de definiciones de UI 136, la etapa 108 de gestión de ciclo de vida de componentes incluye una subetapa de instanciación de componente 112 que puede comenzar, en una realización, cuando la aplicación comienza a lanzarse (por ejemplo, cuando un navegador carga por primera vez una página de la aplicación). En esta subetapa de instanciación de componente 112, los valores de atributo predeterminados (proporcionados por los archivos de definiciones de componentes definidos, que también pueden incluir código (HTML y/o JavaScript) para proporcionar la funcionalidad de componente) se reconciliarán (o "combinarán") con los valores de atributo definidos del archivo de definiciones de UI basándose en un conjunto de reglas. En algunas realizaciones, los valores de atributo específicos del componente del archivo de definiciones de UI tendrán prioridad sobre los valores de atributo en conflicto de la parte de "valores predeterminados" del archivo de definiciones de UI, y los valores de atributo de la parte de "valores predeterminados" del archivo de definiciones de UI tendrá prioridad sobre los valores de atributo en conflicto establecidos por los archivos de definiciones de componentes; sin embargo, se pueden definir otras reglas de prioridad dependiendo de los deseos de la instancia de configuración particular.

Con los valores predeterminados de componente "mezclados" determinados, a continuación, la subetapa de instanciación de componente 112 lleva a cabo el mapeo de parámetros de CSS 140 para aplicar las reglas de estilo a los componentes a colocar en la página sin crear conflictos. A continuación, se inyectan en la página las reglas de estilo (por ejemplo, CSS). El mapeo de parámetros de CSS se describirá con más detalle más adelante en la presente memoria.

La subetapa de instanciación de componente 112 también incluye mezclar 144 los parámetros de definición de UI (por ejemplo, relacionados con el comportamiento de componente) con los parámetros de componente predeterminados. Esto también se puede producir según una regla de prioridad; en una realización, los parámetros de definición de UI tienen prioridad sobre los parámetros de componente predeterminados de los archivos de definiciones de componentes.

A continuación, se obtiene 146 el código de visualización parametrizado necesario (por ejemplo, HTML) para los componentes necesarios (basándose en código de los archivos de definiciones de componentes) y el HTML necesario se inyecta en la aplicación. De manera similar, se recopila la definición de componentes de código de secuencia de comandos parametrizado 150 y se asigna 152 todo el código de secuencia de comandos necesario (por ejemplo, JavaScript) para soportar los componentes y la UI. Por tanto, cada componente generado participa como un componente activo en el marco de trabajo 154.

La etapa 108 de gestión de ciclo de vida de componentes también puede incluir, en una realización, una subetapa de destrucción de componentes 110 que funciona para eliminar componentes de aplicación si es necesario. Por ejemplo, si un comportamiento de un componente es eliminar otro componente, ese otro componente se puede eliminar eliminando de la utilización activa 156 cualesquiera reglas de estilo específicas del componente (por ejemplo, CSS), eliminando de la utilización activa 158 cualquier código de visualización (por ejemplo, HTML) del componente y eliminando 160 cualesquiera objetos de código de secuencia de comandos (por ejemplo, JavaScript)

creado para el componente.

La figura 2 es un diagrama de flujo que ilustra operaciones de ejemplo para la edición interactiva de una aplicación web, según una realización. Las operaciones de este y otros diagramas de flujo se describirán haciendo referencia a las realizaciones de ejemplo de los otros diagramas. Sin embargo, se debe entender que las operaciones de los diagramas de flujo se pueden llevar a cabo por realizaciones de la invención distintas de las explicadas haciendo referencia a estos otros diagramas, y las realizaciones de la invención explicadas haciendo referencia a estos otros diagramas pueden llevar a cabo operaciones distintas de las analizadas haciendo referencia a los diagramas de flujo. Las operaciones de la figura 2 se pueden llevar a cabo por una estación final, tal como una estación final de usuario o por medio de otro dispositivo informático. En una realización, antes de las operaciones ilustradas en la figura 2, se genera dinámicamente una aplicación web en tiempo de ejecución bajo el control de un archivo de definiciones de UI. Una realización de ejemplo de generación de la aplicación web se describe con respecto a la figura 11.

En la operación 210, un componente de software que representa un editor interactivo de la aplicación web se carga dinámicamente en una instancia en ejecución de la aplicación web. El editor interactivo permite la edición de uno o varios componentes de UI de esa instancia en ejecución de la aplicación web. El editor interactivo puede ser una secuencia de comandos del lado del cliente, tal como escrito en JavaScript. El editor interactivo se puede acoplar por sí mismo en el tiempo de ejecución de la aplicación web por medio de manipulación de DOM. En una realización, el editor interactivo se puede acoplar en el tiempo de ejecución por medio de un gancho de software en la aplicación web. En otra realización, el editor interactivo se puede acoplar en el tiempo de ejecución de la aplicación web por medio de una extensión de navegador, sin el conocimiento de la aplicación web. El flujo pasa de la operación 210 a la operación 215.

En la operación 215, se recibe una selección de un elemento de DOM de la aplicación web. La selección puede tomar diferentes formas, incluyendo una interacción de usuario definida con la representación visual de un componente de UI en la aplicación web en ejecución (por ejemplo, un clic con el botón derecho sobre la representación visual, etc.) o por medio de una entrada de texto directa de un ID o nombre de componente. A continuación, el flujo pasa a la operación 220. Aunque la selección de usuario del elemento de DOM parece ser relativamente sencilla, está bastante involucrada a nivel de implementación para asociar el elemento de DOM seleccionado con el componente que lo creó.

En la operación 220, se determina el componente de UI correspondiente al elemento de DOM seleccionado. En una realización, la determinación incluye inspeccionar el DOM para encontrar el elemento de DOM asociado con un componente, comenzando en el elemento de DOM seleccionado y recorriendo hasta su o sus padres hasta que encuentra el componente asociado con el elemento de DOM seleccionado, lo que se puede determinar por medio de la presencia de atributos de datos específicos registrados en el elemento de DOM que indican su asociación de estado de componente y clase de componente.

A continuación, el flujo pasa a la operación 225, donde se determina un conjunto de uno o varios parámetros asociados con el componente de UI determinado. El conjunto de parámetros define el aspecto, la estructura y/o el comportamiento del componente. Por ejemplo, parámetros de ejemplo incluyen tamaño (por ejemplo, anchura, altura), color, fuente, posición, intervalo, conmutador de características booleano, valores de selección de características, URL de recursos, orientación de la presentación, curva de animación, especificación de temporización de animación, fuente de datos, objetos compuestos que contienen otros tipos sencillos o complejos, acción a llevar a cabo, etc.

El conjunto de parámetros se puede almacenar en el archivo de definiciones de UI. Además, en algunas realizaciones, también se determinan metadatos de parámetros, tales como descripciones, intervalos de parámetros y selecciones de valores de parámetros. El editor puede utilizar los metadatos de parámetros para mostrar sugerencias al usuario y para configurar el editor de valores (por ejemplo, para presentar un intervalo de deslizador deseado, para presentar posibles valores en un menú desplegable, etc.). Los metadatos de parámetros se pueden descargar con la aplicación web y pueden incluir un manifiesto de metadatos para componentes y parámetros conocidos.

El flujo pasa de la operación 225 a la operación 230, donde se muestra un editor de valores que está configurado para mostrar valores de, por lo menos, parte del conjunto de parámetros, de modo que se permite al usuario modificar esos valores utilizando el editor de valores. Cada parámetro que se muestra se presenta por el editor de valores como un par de una etiqueta que indica el nombre de parámetro y un editor de valores apropiado para el tipo de parámetro. Si también se determinan metadatos de parámetros, el editor de valores también puede mostrar los metadatos si es apropiado. El usuario puede interactuar con el editor de valores para inspeccionar y/o modificar los valores de los parámetros. En algunas realizaciones, se puede mostrar un subeditor para determinados tipos de parámetros. Por ejemplo, se puede mostrar un editor de selector de color cuando se edita un valor de color o se puede mostrar un editor de curva cuando se edita una curva de animación. A continuación, el flujo pasa a la operación 235, donde se recibe una modificación de por lo menos uno de los valores mostrados en el editor de valores.

A continuación, el flujo pasa de la operación 235 a la operación 240, donde la instancia en ejecución de la aplicación web se actualiza para reflejar el por lo menos un valor modificado.

Según la invención, el archivo de definiciones de UI se edita para reflejar el por lo menos un valor modificado y la aplicación web se regenera dinámicamente bajo el control del archivo de definiciones de UI editado. En una realización, cuando el usuario enfoca los conmutadores fuera del editor de valores (por ejemplo, el usuario selecciona una parte diferente de la aplicación web), se registra el valor de parámetro modificado y se activa una actualización de UI. La UI se puede actualizar de forma diferente en diferentes realizaciones, lo que se describirá con más detalle más adelante en la presente memoria.

Aunque la actualización de UI puede parecer un cambio sencillo para el usuario (un nuevo valor del parámetro se refleja en la aplicación web en ejecución actualmente), se produce un proceso complejo para conseguir la edición de UI. Se debe comprender que el nuevo valor (el valor de parámetro cambiado) puede producir un único cambio en la UI o puede producir múltiples cambios en la UI. Además, el nuevo valor puede cambiar el aspecto, el comportamiento y/o la estructura de la aplicación (en pequeña o en gran medida). El editor interactivo no tiene conocimiento previo de que el cambio de UI se activa por un cambio de valor de parámetro (esto permite muchos tipos diferentes de resultados de edición de parámetros, incluso para componentes y parámetros que actualmente no están en existencia). En algunas realizaciones, la recarga de la aplicación web se hace de tal manera que al usuario le parece que solo se está recargando la parte de la aplicación web que se está editando (de forma directa y selectiva). Por ejemplo, en algunas realizaciones, en lugar de recargar toda la aplicación web, solo se recarga la propia UI.

El flujo pasa de la operación 240 a la operación 245, donde la o las modificaciones se conservan. Existen diferentes formas de conservar las modificaciones en diferentes realizaciones. En una realización, el archivo de definiciones de UI editado se almacena en almacenamiento local (por ejemplo, almacenamiento local de HTML5) disponible para el dispositivo cliente. En una realización de este tipo, si se pulsa un botón de recarga de navegador que hace que la aplicación web se recargue completamente, la aplicación web está configurada para detectar la presencia del archivo de definiciones de UI almacenado en la sesión editado y para utilizarlo en lugar de su archivo de definiciones de UI usual. En otra realización, el archivo de definiciones de UI editado se carga en un servidor para descarga para futuras solicitudes del archivo de definiciones de UI.

La aplicación web se puede implementar en diferentes plataformas que tienen diferentes perfiles de visualización. Por ejemplo, la aplicación web se puede implementar en un teléfono inteligente (que tiene una pantalla relativamente pequeña), en una tableta (que habitualmente tiene una pantalla más grande que un teléfono inteligente) y/o en un dispositivo informático que se visualiza en una pantalla habitualmente más grande (por ejemplo, se implementa por medio de un descodificador que está acoplado a una televisión). En una realización, una simulación de diferentes plataformas se puede visualizar en la misma interfaz de usuario de modo que cualesquiera ediciones realizadas se reflejarán en la totalidad de las diferentes plataformas seleccionadas. Las plataformas no nativas se simulan de tal forma que los elementos visuales son casi precisos y las interacciones serán parcialmente completas. Cada plataforma simulada puede tener su propio archivo de definiciones de UI, de modo que los componentes y/o parámetros definidos en una plataforma pueden no darse en otra. Se lleva a cabo gestión de excepciones si se producen dichas desadaptaciones, tal como ignorar parámetros que no se producen en la plataforma de recepción o enviar un mensaje de advertencia al emisor de que se ha producido una desadaptación. Cada plataforma simulada lleva a cabo su implementación de una edición y, por lo tanto, una edición puede producir resultados sustancialmente diferentes entre las diferentes plataformas.

La figura 3 ilustra una interfaz de usuario de ejemplo que muestra la simulación de UI simultánea de diferentes plataformas, según una realización. Tal como se ilustra en la figura 3, la interfaz de usuario 305 incluye tres plataformas diferentes, la plataforma 315 (representada como una experiencia de plataforma de 1 pie, que, en general, se considera que es una plataforma de teléfono inteligente), la plataforma 320 (representada como una experiencia de plataforma de 2 pies, que, en general, se considera que es una plataforma de tableta o web) y la plataforma 325 (representada como una experiencia de plataforma de 10 pies, que, en general, se considera que es una plataforma de televisión). Se debe comprender que el número de plataformas y el tamaño de las plataformas ilustradas en la figura 3 es ejemplar y se pueden simular diferentes número y tamaños de las plataformas según la invención. La interfaz de usuario incluye el selector de simulación 310, que permite al usuario seleccionar diferentes plataformas predefinidas. Con respecto a la figura 2, si se están simulando múltiples plataformas diferentes, la modificación de un parámetro se aplicará a cada una de las diferentes plataformas simuladas. Por ejemplo, de manera similar a la operación 240, la aplicación web implementada en cada una de las diferentes plataformas se actualiza según la modificación del parámetro. Por tanto, un único usuario en una única máquina puede simular múltiples dispositivos de destino. Esto puede ayudar al usuario a evaluar los resultados de las ediciones en los diferentes dispositivos de destino.

La figura 4 ilustra operaciones de ejemplo realizadas en una estación final para ejecutar una aplicación web parametrizada que incluye un editor interactivo y editar la aplicación web, según una realización. La figura 4 ilustra un ejemplo de recarga de la aplicación web de tal forma que solo se recarga la propia UI para mejorar el rendimiento

de la aplicación web.

En la operación 410, la estación final comienza a ejecutar la aplicación web. Por ejemplo, la estación final comienza a descargar la aplicación web y comienza a analizar sintácticamente y cargar la aplicación web. A continuación, el flujo pasa a la operación 415, donde la estación final adquiere recursos requeridos para la aplicación web (por ejemplo, vinculados en la aplicación web), tales como código de generación de secuencia de comandos del lado del cliente (por ejemplo, JavaScript), uno o varios archivos de CSS, una o varias imágenes, etc., descargando estos recursos. Según la invención, la estación final adquiere el archivo de definiciones de UI (desde una lectura local o desde un servidor) y un conjunto de uno o varios archivos de definiciones de componentes que incluyen código para implementar componentes (incluyendo el conjunto de componentes indicado por el archivo de definiciones de UI y, opcionalmente, un conjunto de componentes adicionales no definidos en el archivo de definiciones de UI).

A continuación, el flujo pasa a la operación 420, done la estación final inicializa el marco de trabajo incluyendo el entorno, la localización, el marco de trabajo de componentes, el singleton, etc. Por ejemplo, el conjunto de componentes identificado se registra en el marco de trabajo de componentes de la aplicación web como un registro. Inicializar el entorno puede incluir determinar conmutadores que configuran la aplicación web a partir de un archivo de configuración cargado dinámicamente. Inicializar la localización puede incluir cargar cadenas de texto aplicables al lugar actual (país e idioma). Inicializar uno o varios singleton puede incluir construir instancias únicas y fijas de objetos que proporcionan servicios a otros objetos a nivel de aplicación. Inicializar el marco de trabajo de componentes puede incluir inicializar tecnologías para implementar la aplicación web, tales como tecnologías de representación de gráficos.

A continuación, el flujo pasa a la operación 425, donde la estación final inicializa la UI de la aplicación web utilizando el archivo de definiciones de UI. El archivo de definiciones de UI se puede leer de un directorio local (por ejemplo, en el caso de un cliente huésped) o se puede descargar desde un recurso de red (por ejemplo, en un cliente web) si no se ha descargado todavía. El archivo de definiciones de UI almacena la representación parametrizada externalizada de la aplicación web. La estructura, el comportamiento y el aspecto de la aplicación web se pueden recrear a partir del archivo de definiciones de UI y, por tanto, define completamente la aplicación web. Los componentes basados en el archivo de definiciones de UI y el conjunto de archivos de definiciones de componentes se instancian dinámicamente, lo que incluye generar dinámicamente objetos HTML, CSS y/o de secuencias de comandos del lado del cliente, que representan el conjunto de componentes basándose en la identificación de parámetros definida en el archivo de definiciones de UI. En una realización, esta instanciación dinámica se lleva a cabo por un módulo de generación de UI basándose en el archivo de definiciones de UI analizado sintácticamente (que define qué componentes instanciar, cuándo instanciarlos y con qué parámetros) y el registro. En una realización, los componentes pueden tener parámetros predeterminados inherentes, parámetros predeterminados por componente, tal como se define en el archivo de definiciones de UI (en su sección "valores predeterminados", véase la Tabla A anterior) y parámetros definidos sobre el nivel de instanciación concreto según el archivo de definiciones de UI (en su sección "componentes"). Los parámetros de componente (tal como se definen en el archivo de definiciones de UI y como valores predeterminados asociados con clases de componentes de UI) informan de la generación dinámica de HTML u CSS, que son una parte de la instanciación de componente. También informan de la configuración de objetos de JavaScript que representan componentes dentro del marco de trabajo de componentes. En una realización, este marco de trabajo de componentes de UI proporciona componentes en bruto (todavía sin configurar) y gestiona el ciclo de vida (y la visibilidad implícita) de los componentes. En algunas realizaciones, una plataforma de implementación es cualquier aplicación de representación gráfica basada en web y, en una realización, una plataforma de implementación es HTML5 y, por tanto, soporta cualquier dispositivo que pueda ejecutar HTML5, incluyendo de forma no limitativa teléfonos inteligentes, tabletas y ordenadores tradicionales.

A continuación, el flujo pasa a la operación 430, donde se carga un editor interactivo en la aplicación web. El editor interactivo se puede cargar en la aplicación web de una forma similar a la descrita anteriormente con respecto a la figura 2. Después de que se cargue el editor interactivo, en la operación 435 la estación final recibe una edición para la UI (por ejemplo, un cambio en un parámetro). La edición se puede recibir de una forma similar a la descrita anteriormente con respecto a la figura 2. A continuación, el flujo pasa a la operación 440, donde el archivo de definiciones de UI se actualiza para reflejar la edición (por ejemplo, se cambia un parámetro en el archivo de definiciones de UI). A continuación, el flujo pasa a la operación 445, donde la UI se reinicializa utilizando la UI actualizada para reflejar la UI modificada. La reinicialización de la UI se puede llevar a cabo de una forma similar a la inicialización de la UI con el archivo de definiciones de UI anterior. A continuación, el flujo pasa a la operación 450, donde si hay otra edición para la UI recibida, entonces el flujo vuelve a la operación 440, donde el archivo de definiciones de UI se actualiza con la edición; en caso contrario, las operaciones finalizan. Por tanto, en lugar de recargar toda la aplicación web y volver a adquirir los recursos, solo se recarga la propia UI.

Además, aunque no se ilustra en la figura 4, si los datos de un servidor se representan en la UI (como es habitual), se utiliza una capa de caché entre las solicitudes de servidor y la red, de tal forma que los componentes de UI recreados (aquellos componentes que no cambiaron desde la edición) pueden tener acceso inmediato a los datos. Además, el marco de trabajo tiene la capacidad de volver al mismo estado de UI cuando tiene lugar una recarga de UI.

La recarga de UI se puede hacer de forma diferente en diferentes realizaciones. La figura 5 ilustra operaciones de ejemplo para llevar a cabo la recarga de UI en una realización. En la operación 510, se suprime el árbol de componentes y el DOM. A continuación, en la operación 515, se lee y se analiza sintácticamente el archivo de definiciones de UI editado. A continuación, en la operación 520, se crea un árbol de componentes actualizado y un DOM actualizado. La figura 6 ilustra operaciones de ejemplo para llevar a cabo la recarga de UI en otra realización. A diferencia de la realización de la figura 5, la realización de la figura 6 no destruye el árbol de componentes existente y todos los elementos de DOM asociados. En lugar de destruir y recrear el árbol de componentes y el DOM, se crean versiones ocultas inactivas y no acopladas del árbol de componentes y del DOM, y se comparan con las versiones sin modificar (o anteriores). Solo se aplican las diferencias transfiriendo el estado (atributos o elementos u objetos activos) desde las versiones ocultas a las versiones activas/acopladas. El resultado es un árbol de componentes actualizado y un DOM actualizado en el que no es necesario que el DOM se desacople y se vuelva a acoplar. Esto permite que una actualización parezca estar libre de inestabilidades al usuario.

En la operación 610, se lee y se analiza sintácticamente el archivo de definiciones de UI editado. A continuación, en la operación 615, se crea una versión oculta no acoplada e inactiva del árbol de componentes y del DOM utilizando el archivo de definiciones de UI editado. A continuación, en la operación 620, la versión oculta del árbol de componentes se compara con la versión activa del árbol de componentes y la versión oculta del DOM se compara con la versión activa del DOM. A continuación, en la operación 625, la o las diferencias se aplican entre las versiones ocultas del árbol de componentes y del DOM comparadas con las versiones activas del árbol de componentes y del DOM. Por ejemplo, solo los componentes y los elementos de DOM nuevos/cambiados, o solo los atributos de elementos cambiados, se transfieren al estado de la versión activa del árbol de componentes y del DOM. La versión oculta del árbol de componentes y del DOM se puede eliminar, de tal forma que hay una jerarquía de componentes mezclados activos y una jerarquía de DOM mezclados activos.

En algunas realizaciones, se especifican determinadas interacciones de usuario que el usuario puede llevar a cabo directamente sobre componentes específicos y parámetros específicos. Por ejemplo, arrastrar con un ratón u otra entrada para mover un componente dentro de su componente padre se puede llevar a cabo directamente. Como otro ejemplo, cambiar el padre de un componente para que esté dentro de un nuevo componente padre se puede llevar a cabo directamente. Dichas manipulaciones directas requieren un conocimiento previo del editor del parámetro y del efecto que tendrá la edición. La manipulación directa permite una mayor interactividad para ediciones realizadas habitualmente. La manipulación directa también permite que el editor lleve a cabo la edición sin hacer una actualización de UI regular, lo que, por lo tanto, permite un rendimiento de rápida velocidad de actualización de pantalla. En una realización, la traslación (desplazamiento en x, y) de elementos de UI puede ser una operación de edición de manipulación directa deseada.

La figura 7 ilustra operaciones de ejemplo para la manipulación directa, según una realización. En la operación 710, se recibe una selección de un elemento de DOM, tal como se describe de manera similar con respecto a la operación 215. A continuación, en la operación 715, se determina un componente de UI correspondiente al elemento de DOM seleccionado, tal como se describe de manera similar con respecto a la operación 220. A continuación, el flujo pasa a la operación 720, donde se determina que el componente seleccionado es elegible para la manipulación directa que permite aplicar la edición sin actualizar toda la UI. Esta determinación se puede llevar a cabo, por ejemplo, identificando el componente o parámetro como uno que soporta manipulación directa. A continuación, el flujo pasa a la operación 725, donde se recibe una edición del elemento de DOM. A modo de ejemplo, un usuario mueve el elemento de DOM. A continuación, el flujo pasa a la operación 730, donde la UI se actualiza para reflejar la edición.

En algunas realizaciones, se pueden añadir nuevos componentes a la UI por medio de una interfaz de usuario que permite al usuario elegir entre un conjunto de componentes disponibles predefinidos y, por medio de un gesto u otra selección (por ejemplo, arrastrar y soltar) de uno de los componentes disponibles predefinidos, un nuevo componente se puede instanciar dentro de la aplicación web. Una aplicación web completa se puede construir desde cero utilizando una interfaz de este tipo. Además, después de la selección, también se pueden suprimir componentes. La figura 8 ilustra una interfaz de usuario de ejemplo 810 que incluye una funcionalidad para que un usuario seleccione un componente disponible predefinido y lo añada a una aplicación web, según una realización. Un conjunto de componentes de UI predefinidos 820 está disponible para seleccionarse e incluirse en la aplicación web. Por ejemplo, un usuario puede seleccionar uno de los componentes de UI 820 y arrastrar y soltar el nuevo componente en la UI para instanciar el componente de UI 830 en la aplicación web. El componente de UI recién instanciado 830 se puede editar exactamente igual que cualquier otro componente en la aplicación web. Por ejemplo, el editor de parámetros 815 permite al usuario editar los parámetros de los componentes de UI. El conjunto de componentes de UI predefinidos 820 puede incluir elementos de UI comunes, tales como botones, deslizadores, pestañas, etc. y/o puede incluir componentes específicos del dominio.

Las realizaciones descritas en la presente memoria permiten que las ediciones se visualicen simultáneamente en una simulación de múltiples dispositivos de destino en un único dispositivo. En algunas realizaciones, se utiliza una arquitectura basada en cliente/servidor para distribuir las ediciones de la aplicación web a uno o varios dispositivos de destino. Esto extiende las ventajas del editor de aplicaciones web interactivo a dispositivos físicos conectados. La arquitectura basada en cliente/servidor incluye por lo menos una estación final de recepción de ediciones (en

adelante, "receptor"), una estación final de envío de ediciones (en adelante, "emisor") y puede incluir un servidor que actúa como un conducto para pasar mensajes entre emisores y receptores. Puede haber uno o varios receptores y emisores que interactúan por medio de un protocolo de mensajes con una sesión introducida y un servidor de retransmisión de mensajes. Por ejemplo, las comunicaciones entre receptores y emisores pueden ser bidireccionales y producirse sobre el protocolo WebSocket (un protocolo para comunicaciones bidireccionales ("full-duplex") sobre TCP/IP). Se pueden utilizar protocolos de conexión de red bidireccionales alternativos.

La figura 9 ilustra una realización de ejemplo para distribuir las ediciones a una aplicación web, según algunas realizaciones. El emisor 920 se conecta con el servidor 915 y solicita el inicio de una nueva sesión. El servidor 915 lleva a cabo la gestión de sesión además de pasar mensajes de edición entre el emisor y los receptores 910A-N. El servidor 915 responde al emisor 920 con un ID de sesión y añade el emisor 920 a una lista de emisores que gestiona para el ID de sesión determinado. Posteriormente, uno o varios receptores 910A-N se conectan al servidor 915 y solicitan su intención de escuchar una sesión. El servidor 915 responde con una lista de ID de sesión disponibles. Los receptores 910A-N responden con su intención de unirse a una sesión con un ID seleccionado. El servidor 915 acusa recibo de la solicitud y añade cada receptor (en este ejemplo, los receptores 910A-N) a una lista de receptores que gestiona para el ID de sesión determinado. Algún tiempo después, el emisor 920 realiza una edición de la UI. Cuando se confirma la edición, se transmite un mensaje que describe la edición de UI 930 del emisor 920 al servidor 915. El servidor 915 recibe el mensaje de edición de UI 930 y pasa el mensaje de edición de UI a los receptores registrados en el ID de sesión determinado. Por ejemplo, tal como se ilustra en la figura 9, el servidor 915 transmite los mensajes de edición de UI 932A-N a los receptores 910A-N, respectivamente. Cada uno de los receptores 910A-N recibe los mensajes de edición y ejecuta la edición localmente. Por ejemplo, cada uno de los receptores 910A-N actualiza la instancia en ejecución de la aplicación web según la edición de una forma similar a la descrita con respecto a la operación 240 de la figura 2. El emisor 920 y los receptores 910A-N pueden dejar la sesión en cualquier momento. La sesión terminará cuando el emisor 920 deje la sesión. El emisor 920 y los receptores 910A-N pueden ser el mismo tipo de dispositivo (por ejemplo, con las mismas dimensiones de pantalla) o pueden ser tipos de dispositivos diferentes (por ejemplo, con diferentes dimensiones de pantalla y/o capacidades).

La figura 10 ilustra una realización de ejemplo para distribuir las ediciones a una aplicación web, según otras realizaciones. A diferencia de la realización ilustrada en la figura 9, la realización ilustrada en la figura 10 permite que múltiples emisores se registren para un ID de sesión determinado. En dicha realización de "múltiples emisores", los emisores también actúan como receptores. Por tanto, tal como se ilustra en la figura 10, los emisores/receptores 1020A-N están acoplados al servidor 1015, que también está acoplado a los receptores 1010A-N. Una estación final que desea ser un emisor de una sesión existente asume la función de receptor uniéndose a la sesión existente (por ejemplo, enviando una solicitud y un ID de sesión al servidor 1015). El servidor 1015 acusa recibo de, o deniega la solicitud. El servidor 1015 puede transmitir una notificación a todos los emisores/receptores 1020A-N cuando se une un nuevo emisor/receptor que incluye el recuento de emisores actual. Como un emisor también es un receptor, cuando se confirma una edición, no se lleva a cabo automáticamente. La edición se transmite desde el emisor/receptor al servidor 1015 y solo se llevará a cabo esa edición si se recibe subsiguientemente el mensaje de edición desde el servidor 1015 en su función de receptor. Por ejemplo, el emisor/receptor realiza una edición de UI y envía un mensaje 1030 que describe la edición de UI al servidor 1015. El servidor 1015 recibe el mensaje de edición de UI 1030 y transmite los mensajes de edición de UI 1032A-N a los receptores 1010A-N, respectivamente, y los mensajes de edición de UI 1034A-N a los emisores/receptores 1020A-N, respectivamente. Solo después de recibir el mensaje de edición de UI 1034A, el emisor/receptor 1020A ejecutará la edición de UI localmente.

Como en una realización de múltiples emisores un emisor también es un receptor y las ediciones se llevan a cabo solo cuando los mensajes de edición se reciben como receptores, todas las estaciones finales ven la misma secuencia de mensajes de edición. Los conflictos de edición potenciales se resuelven a través de la serialización por el servidor 1015 en algunas realizaciones. Por ejemplo, considérese que el emisor/receptor 1020A y el emisor/receptor 1020N han editado el mismo parámetro a diferentes valores y enviado sus mensajes de edición aproximadamente al mismo tiempo. El servidor 1015 procesará un mensaje antes que el otro mensaje y pasará ese mensaje a los receptores y, a continuación, procesará el otro mensaje y pasará ese mensaje a los receptores. Esto garantiza que ambos emisores/receptores reciben los mensajes de edición en el mismo orden. Aunque las ediciones en una realización de múltiples emisores requieren un recorrido adicional de ida y vuelta hasta el servidor adicional en comparación con una realización de un único emisor, se trata de una solución intermedia para garantizar que no existan conflictos.

De manera similar a la figura 9, los emisores/receptores 1020A-N y los receptores 1010A-N pueden dejar la sesión en cualquier momento. La sesión terminará cuando todos los emisores dejen la sesión. En cualquier momento, un receptor y/o un emisor/receptor puede solicitar el número de emisores o receptores registrados actualmente.

En algunas realizaciones, los receptores 1010A-N y los emisores/receptores 1020A-N pueden unirse a la sesión en cualquier momento. El estado de la aplicación web que se está editando puede no estar sincronizado con el estado inicial de la aplicación web y/o las ediciones que se han confirmado. En algunas realizaciones, después de que un nuevo receptor se una a la sesión, se envía una notificación a todos los emisores que les informa del nuevo receptor e incluye información que describe el estado del receptor. Esta información de estado puede incluir qué definición de UI está en uso (un identificador que identifica la definición de UI en uso en el receptor puede ser enviado por el

receptor cuando solicita unirse a la sesión), cuántas etapas de edición se han realizado desde la última carga de la definición de UI en uso actualmente en el receptor, etc. Basándose en este mensaje, el emisor puede, por medio de una solicitud al servidor, insertar una lista de mensajes de edición que están diseñados para sincronizar el receptor que se va a unir con el estado actual de la sesión de edición. El receptor que se va a unir puede aplicar la lista de mensajes de edición (por ejemplo, por medio de la utilización iterativa de su mecanismo de realización de ediciones a partir de mensajes de edición).

En algunas realizaciones, los mensajes de edición incluyen tanto valores antiguos como nuevos para un parámetro objetivo y se lleva a cabo un procedimiento de resolución de conflictos. El receptor inspecciona el mensaje de edición recién recibido y compara el valor "antiguo" con su valor actual para el parámetro objetivo. La edición se llevará a cabo solo si estos valores coinciden. Si estos valores no coinciden, el receptor, a través de su canal de comunicación dúplex con el servidor, envía un mensaje de conflicto de vuelta al o a los emisores, y el o los emisores muestran este conflicto. Esto indica el conflicto al usuario que realiza la edición y proporciona una forma de inspeccionar y resolver la edición manualmente. Por ejemplo, una realización puede mostrar un icono de alarma al lado del parámetro editado, decorado con la plataforma que informó del conflicto y el valor preexistente en el receptor. El usuario puede ignorar el conflicto y aceptar que la edición no se llevará a cabo en la estación final en conflicto. Alternativamente, el usuario puede, por medio de alguna funcionalidad, por ejemplo, un clic en dicho icono de advertencia, enviar un mensaje de edición de vuelta a los receptores de la plataforma en conflicto con un indicador de anulación establecido. Los receptores establecerán el nuevo valor objetivo, independientemente de la falta de coincidencia entre el valor antiguo y el existente, si el indicador de anulación está establecido. Por tanto, el sistema se protege contra anulaciones accidentales de valores de parámetros diferentes de forma intencionada por plataforma, pero sigue permitiendo al usuario forzar una actualización de valores.

La figura 11 ilustra un flujo 1100 en una estación final para generar dinámicamente una aplicación en tiempo de ejecución basándose en un archivo de definiciones de UI, según realizaciones de la invención. Las operaciones de este y otros diagramas de flujo se describirán haciendo referencia a las realizaciones de ejemplo de los otros diagramas. Sin embargo, se debe entender que las operaciones de los diagramas de flujo se pueden llevar a cabo por realizaciones de la invención distintas de las analizadas haciendo referencia a estos otros diagramas, y las realizaciones de la invención analizadas haciendo referencia a estos otros diagramas pueden llevar a cabo operaciones distintas de las analizadas haciendo referencia a los diagramas de flujo.

El flujo 1100 incluye, en el bloque 1105, la recepción por la estación final de un archivo de definiciones de interfaz de usuario (UI). El archivo de definiciones de UI incluye una pluralidad de definiciones que indican atributos de aspecto visual de partes de una aplicación, un conjunto de componentes que deben aparecer dentro de la aplicación y un conjunto de comportamientos que los componentes pueden llevar a cabo. Opcionalmente, esta recepción del archivo de definiciones de UI se produce en el bloque 1110 desde una estación final de servidor a través de una red 1110, y, opcionalmente, esta recepción del archivo de definiciones de UI se produce como parte del paquete de aplicación en el bloque 1115.

En algunas realizaciones, el operador genera el archivo de definiciones de UI por medio de una interfaz basada en texto sencilla, una interfaz de arrastrar y soltar (para añadir componentes a una UI) o una combinación de las dos. El archivo de definiciones de UI también se puede generar utilizando la interfaz ilustrada en la figura 8.

El flujo 1100 también incluye, en el bloque 1120, recibir un conjunto de uno o varios archivos de definiciones de componentes que incluyen código para implementar una pluralidad de componentes. La pluralidad de componentes incluye el conjunto de componentes indicado por el archivo de definiciones de UI (es decir, la totalidad del conjunto de componentes y, posiblemente, componentes adicionales, tienen definiciones en el conjunto de archivos de definiciones de componentes).

Después de un lanzamiento de inicio de la aplicación, en el bloque 1125, se lee/analiza sintácticamente el archivo de definiciones de UI para identificar los atributos de aspecto visual, el conjunto de componentes y el conjunto de comportamientos en el bloque 1130. Opcionalmente, el análisis sintáctico en el bloque 1130 incluye registrar como un registro el conjunto identificado de componentes con un marco de trabajo de componentes de la aplicación, en el bloque 1135.

El flujo 1100 también incluye, en el bloque 1140, instanciar dinámicamente el conjunto de componentes basándose en el archivo de definiciones de UI y en el conjunto de archivos de definiciones de componentes.

Según la invención, esto incluye, en el bloque 1145, generar dinámicamente objetos HTML, CSS y JavaScript que representan el conjunto de componentes basándose en los parámetros de identificación definidos en el archivo de definiciones de UI.

En una realización, tras la actualización de la aplicación web para reflejar una edición de la UI, se llevan a cabo las etapas de las operaciones 1130-1145 según el archivo de definiciones de UI actualizado.

En una realización, esta instanciación dinámica se lleva a cabo por un módulo de generación de UI basándose en el

archivo de definiciones de UI analizado sintácticamente (que define qué componentes se deben instanciar, cuándo se deben instanciar y con qué parámetros) y el registro. En una realización, los componentes pueden tener parámetros predeterminados inherentes, parámetros predeterminados por componente, tal como se define en el archivo de definiciones de UI (en su sección "valores predeterminados", véase la Tabla A anterior) y parámetros definidos sobre el nivel de instanciación concreto según el archivo de definiciones de UI (en su sección "componentes"). Los parámetros de componente (tal como se definen en el archivo de definiciones de UI y como valores predeterminados asociados con clases de componentes de UI) informan de la generación dinámica de HTML u CSS, que son una parte de la instanciación de componente. También informan de la configuración de objetos de JavaScript que representan componentes dentro del marco de trabajo de componentes. En una realización, este marco de trabajo de componentes de UI proporciona componentes en bruto (todavía sin configurar) y gestiona el ciclo de vida (y la visibilidad implícita) de los componentes. En algunas realizaciones, una plataforma de implementación es cualquier aplicación de representación gráfica basada en web y, en una realización, una plataforma de implementación es HTML5 y, por tanto, soporta cualquier dispositivo que pueda ejecutar HTML5, incluyendo de forma no limitativa teléfonos inteligentes, tabletas y ordenadores tradicionales.

En consecuencia, las realizaciones de la invención permiten la construcción en tiempo de ejecución de múltiples aplicaciones web dentro de los límites de un conjunto finito de bloques de construcción de UI (es decir, componentes) y su conjunto finito asociado de parámetros de configuración, bajo el control de un archivo de definiciones de UI.

La figura 12 ilustra un diagrama de bloques de un sistema que 1200 incluye un sistema de IPTV 1206 que utiliza la generación dinámica de aplicaciones en tiempo de ejecución basándose en código parametrizado, según una realización de la invención. El sistema 1200 incluye uno o varios proveedores de contenido 1210A-1210M que proporcionan recursos de vídeo al sistema de IPTV 1206 (o directamente a las estaciones finales 1228), que en última instancia se distribuirán/comunicarán a las estaciones finales 1228 (opcionalmente en el emplazamiento de usuario 1208) a través de una red de comunicación 1204. La red de comunicación 1204 puede incluir cualquier tipo de red de datos, red de voz, red de difusión, red basada en IP y/o red inalámbrica que facilite la comunicación de datos y contenido multimedia en cualquier formato. La red de comunicación 1204 se puede implementar utilizando cualquier tipo bien conocido de topología de red y/o protocolo de comunicación, y puede representarse o sino implementarse como una combinación de dos o más redes.

Las estaciones finales de usuario 1228 (o sistemas de visualización, dispositivos de consumidor, dispositivos cliente, etc.) son dispositivos electrónicos utilizados por los usuarios 1202A-1202N para acceder a recursos de vídeo por medio de una aplicación 1226A-1226E que proporciona acceso al sistema de IPTV 1206. En la presente memoria se ilustra un conjunto no exhaustivo de estaciones finales 1228 y se incluye un descodificador (STB) 1222 que se conecta a una pantalla 1224A (habitualmente un televisor, pero también puede ser otro tipo de monitor, proyector, etc.). Otras estaciones finales 1228 incluyen un teléfono inteligente 1230, una tableta 1232 y un ordenador portátil o un ordenador de sobremesa 1234, cada uno de los cuales puede incluir un procesador, almacenamiento legible por ordenador, una pantalla 1224B-1224D y, opcionalmente, una aplicación 1226A-1226E que se ejecuta para permitir la conectividad/interacción con el sistema de IPTV 1206. Sin embargo, otras estaciones finales se pueden implementar como cualquiera o una combinación de un dispositivo cableado y/o inalámbrico, como cualquier forma de dispositivo cliente de televisión (por ejemplo, STB 1222, grabadora de vídeo digital (DVR)), dispositivo de juegos, dispositivo informático, dispositivo informático portátil, dispositivo de consumidor, dispositivo multimedia, dispositivo de comunicación, dispositivo de procesamiento y/o representación gráfica de vídeo, dispositivo de aparato, teléfono móvil (por ejemplo, celular, voz sobre IP (VoIP), Wi-Fi, etc.), un dispositivo multimedia portátil (por ejemplo, un reproductor multimedia personal/portátil, un reproductor multimedia de mano, etc.), dispositivo ponible y/o como cualquier otro tipo de dispositivo que se implemente para recibir contenido multimedia en cualquier forma de datos de audio, vídeo y/o imagen. Una estación final (por ejemplo, STB 1233) también puede estar asociada con un usuario 1202A (es decir, una persona) y/o una entidad que maneja el dispositivo.

Las diversas estaciones finales (1222, 1230, 1232, 1234) mostradas en el sistema 1200 pueden incluir o no un dispositivo de visualización respectivo (por ejemplo, 1224A-1224D). Una estación final y un dispositivo de visualización en conjunto representan gráficamente y reproducen datos de audio, vídeo y/o imagen. Un dispositivo de visualización (por ejemplo, la pantalla 1224A) se puede implementar como cualquier tipo de una televisión, una televisión de alta definición (HDTV), una pantalla de cristal líquido (LCD), una pantalla de diodos emisores de luz (LED) o un sistema de visualización similar. Los diversos dispositivos cliente (por ejemplo, dispositivos de televisión, de juegos o informáticos) también se pueden asociar con uno o varios dispositivos de entrada, tales como un dispositivo de control remoto para aceptar entradas y selecciones seleccionables por el usuario en el dispositivo cliente de televisión, un controlador de juegos para entradas seleccionables por el usuario en el dispositivo de juegos y/o dispositivos de entrada de teclado y/o ratón para entradas seleccionables por el usuario en la estación final. Las estaciones finales descritas en la presente memoria también se pueden implementar con cualquier combinación diferente de otros componentes, tales como uno o varios procesadores, componentes de comunicación (por ejemplo, interfaces de red), componentes de memoria y/o circuitos de procesamiento y control. Por ejemplo, una estación final puede incluir interfaces de red para recibir recursos de vídeo desde el sistema de IPTV 1206 y/o los proveedores de contenidos 1210A-1210M, interfaces para recibir entradas de difusión y/o por el aire, mediante las cuales se pueden recibir recursos de vídeo por el aire. Las estaciones finales también pueden incluir uno o varios

sintonizadores para sintonizar canales de televisión y/o flujos de datos para su presentación y visualización.

Las estaciones finales y/o pantallas pueden incluir, opcionalmente, las aplicaciones (o "apps") de IPTV 1226A-1226E para ayudar a proporcionar conectividad al sistema de IPTV 1206. Estas apps de IPTV 1226, cuando se ejecutan por procesadores de los respectivos dispositivos, se pueden configurar para hacer que los respectivos dispositivos se conecten al sistema de IPTV 1206 (por ejemplo, utilizando un conjunto de interfaces de red), envíen solicitudes al sistema de IPTV 1206 (por ejemplo, de listas de recursos de vídeo, de los propios recursos de vídeo), reciban respuestas del sistema de IPTV 1206 (por ejemplo, elementos de interfaz de usuario (UI) del sistema de IPTV 1206, recursos de vídeo), presenten las interfaces de usuario del sistema de IPTV 1206 en las pantallas a los usuarios y/o muestren los recursos de vídeo y cualesquiera interfaces de usuario correspondientes (opcionales) (por ejemplo, controles de reproducción, recursos de vídeo adicionales, anuncios, etc.). En algunas realizaciones de la invención, las aplicaciones 1226A-1226E se construyen utilizando tecnologías basadas en web, incluyendo uno o varios de código de HTML, reglas de CSS, código de JavaScript, etc.

En la realización representada de la figura 12, el sistema de IPTV 1206 incluye uno o varios dispositivos informáticos 1230 que incluyen uno o varios procesadores 1234, interfaces de red 1236 (para conectarse a los proveedores de contenidos 1210A-1210M y/o al sistema de redes sociales 1220 y/o a las estaciones finales 1228) y medios de almacenamiento legible por ordenador 1232. En algunas realizaciones, los medios de almacenamiento legible por ordenador 1232 pueden almacenar copias de recursos de vídeo, que pueden ser proporcionadas por los proveedores de contenidos 1210A-1210M. El término "recurso de vídeo" se utiliza, en general, para referirse a vídeo o a una colección de imágenes que pueden incluir audio o no; sin embargo, el término "recurso de vídeo" también se puede utilizar genéricamente para referirse a un fragmento de contenido multimedia, incluyendo de forma no limitativa cualquier tipo de datos de audio, vídeo y/o imagen recibidos desde cualquier fuente de contenido y/o datos multimedia. Tal como se describe en la presente memoria, un recurso de vídeo puede incluir contenido de vídeo grabado, contenido de vídeo bajo demanda (VOD), contenido de vídeo OTT, contenido de televisión (por ejemplo, televisión "en vivo", televisión "de difusión"), anuncios, anuncios publicitarios, vídeos de música, películas, videoclips y otro contenido multimedia. Dependiendo de la configuración, el sistema de IPTV 1206 puede proporcionar los recursos de vídeo a las estaciones finales 1228 a través de la red 1204, pero en algunas configuraciones las estaciones finales 1228 utilizan la red 1204 para acceder directamente a los recursos de vídeo de los proveedores de contenido 1210A-1210M.

El medio de almacenamiento legible por ordenador 1232 también puede almacenar otro contenido multimedia, metadatos, juegos interactivos, aplicaciones basadas en red y/o cualquier otro contenido o datos (por ejemplo, datos de aplicación de guía de programas, datos de interfaz de usuario, contenido de anuncios, datos de subtítulos ocultos, metadatos de contenido, resultados de búsqueda y/o recomendaciones, etc.) para su utilización por el sistema de IPTV 1206 y/o las estaciones finales 1228 al interactuar con el sistema de IPTV 1206 y/o los recursos de vídeo.

En la realización representada, el conjunto de procesadores 1234 de los uno o varios dispositivos informáticos 1230 ejecuta una instancia de módulo de estilo personalizado 1240B, que se puede lanzar utilizando código de módulo de estilo personalizado 1240A almacenado por el medio de almacenamiento legible por ordenador 1232. La instancia de módulo de estilo personalizado 1240B se utiliza como parte del sistema para generar dinámicamente aplicaciones por medio de la utilización del código de estilo aumentado 1242 y del código de generación de estilo 1244 almacenados por el medio de almacenamiento legible por ordenador 1232. En una realización, el código de estilo aumentado 1242 incluye partes de reglas de estilo que siguen un estándar de estilo (por ejemplo, CSS) que se han modificado, o aumentado, para incluir expresiones que incluyen parámetros. Por tanto, el código de estilo aumentado 1242, en su totalidad, no seguirá estrictamente el estándar de estilo y, por tanto, se considerará inválido según ese estándar de estilo. La instancia de módulo de estilo personalizado 1240B puede traducir el código de estilo aumentado 1242 al código de generación de estilo 1244. En una realización, el código de generación de estilo 1244 es código ejecutable (por el o los dispositivos informáticos 1230 o por las aplicaciones 1226 que se ejecutan en la o las estaciones finales 1224) que puede generar reglas de estilo válidas 1248 a utilizar por las aplicaciones 1226. En algunas realizaciones, este código de generación de estilo 1244 comprende código de JavaScript, pero en otras realizaciones puede incluir cualquier otro código ejecutable por ordenador (por ejemplo, código escrito en Python, Lua, C++, C, ML, Fortran, PHP, Ruby, VBScript, Scheme, secuencias de comandos Shell, XSLT, Tcl, Java, Smalltalk, Objective C, C#, Visual Basic, etc.).

En una realización, el código de generación de estilo 1244 se ejecuta por medio de un conjunto de invocaciones de generación de estilo 1246, que pueden existir opcionalmente en (y ejecutarse por) el o los dispositivos informáticos 1230 o directamente por una estación final 1224 en tiempo de ejecución. En una realización, el conjunto de invocaciones de generación de estilo 1246 hace que el código de generación de estilo 1244 se ejecute utilizando un conjunto de variables de entrada, lo que hace que se genere el conjunto generado personalizado de reglas de estilo 1248. A continuación, en la figura 13, la figura 14A y la figura 14B se presentan con más detalle ejemplos de código de estilo aumentado 1242, código de generación de estilo 1244, código de invocación de generación de estilo 1246 y conjuntos generados de código de estilo válidos 1248.

La figura 13 ilustra código de estilo aumentado 1242 y código de generación de estilo 1244, según una realización

de la invención. El código de estilo aumentado 1242 y el código de generación de estilo 1244 se utilizan como parte de la solución global para proporcionar aplicaciones generadas dinámicamente; en particular, estas partes de la solución permiten estilos dinámicos de las interfaces de usuario de la aplicación.

En consecuencia, en una realización, el código de estilo aumentado 1242 comprende archivos de CSS parametrizados que se pueden utilizar en un sistema de transformación evaluable diferido para una finalización parametrizada, en tiempo de ejecución, dinámica y de alto rendimiento. En una realización, el código de estilo aumentado 1242 se proporciona como uno o varios archivos de entrada que incluyen reglas de CSS modificadas que se han aumentado con sintaxis basada en expresiones adicionales. En algunas realizaciones, el código de estilo aumentado 1242 se genera por un proveedor de tecnología y se construye expresamente para "exponer" determinados aspectos de una UI que se va a modificar por medio de la utilización de expresiones 1310 que incluyen variables 1315.

Notablemente, en una realización, los archivos de entrada del código de estilo aumentado 1242 pueden ser en sí mismos la salida de otro software de preprocesamiento de CSS, tal como el lenguaje de hojas de estilo dinámicas Less.js.

Tal como se representa en la figura 13, el código de estilo aumentado ilustrado 1242A incluye dos reglas, o declaraciones de estilo, incluyendo cada una de ellas sintaxis de estilo 1320 y expresiones 1310 (que utilizan variables 1315) demarcadas por delimitadores de expresión 1305. En este ejemplo, los delimitadores de expresión 1305 comprenden un delimitador de apertura (que comprende un signo de dólar seguido de una llave de apertura) y un delimitador de cierre (que comprende una llave de cierre), aunque en otras realizaciones se pueden utilizar otros delimitadores de apertura y de cierre, siempre que se puedan identificar de forma no ambigua por un analizador sintáctico. En este ejemplo, una primera regla (para objetos que tienen una clase que coincide con el selector de ".rule1") incluye tres declaraciones (o subreglas), donde un atributo "width" tiene un valor asociado que incluye una expresión 1310 que define el valor como la suma de una variable "itemWidth" con el número treinta. Una segunda declaración también incluye una expresión 1310 que indica que el atributo "border-style" debe tener un valor de atributo asociado representado por el valor de una variable denominada "borderPattern". Finalmente, la primera regla también incluye una tercera declaración que indica que un atributo "color" debe tener un valor de atributo asociado representado por el valor de una variable denominada "defaultColor". Una segunda regla (para objetos que tienen una clase que coincide con el selector de ".rule2") es similar a la primera regla, ya que también incluye las tres declaraciones para los atributos "width", "border-style" y "color". Sin embargo, las expresiones 1310 para "width" y "color" son diferentes; en este caso, el valor de atributo para "width" se configura para representar 1,5 veces el valor de la variable itemWidth, y el valor de atributo para "color" se configura para ser el resultado de una aplicación de una función "lighten()" que utiliza una variable "defaultColor" como su argumento. Esta función lighten() puede ser una función que es parte de un lenguaje del código de generación de estilo generado posteriormente 1244A, definido por el proveedor de tecnología, o puesto a disposición de alguna otra manera.

El formato de las expresiones 1310 dentro del código aumentado se puede configurar de varias formas; en una realización, el proveedor de tecnología simplemente define una gramática/sintaxis para estas expresiones 1310 que serán reconocidas por un analizador sintáctico utilizado para generar el código de generación de estilo 1244A. Sin embargo, en otras realizaciones, el formato de las expresiones 1310 sigue una gramática/sintaxis de lenguaje definida de un lenguaje de programación común.

Con los uno o varios archivos que representan el código de estilo aumentado 1242A, la instancia de módulo de estilo personalizado 1240B analizará sintácticamente y compilará el código de estilo aumentado 1242A en el código de generación de estilo 1244A. En una realización, el código de generación de estilo 1244A es código de JavaScript que, cuando se ejecuta, inyecta reglas de CSS en una carga de aplicación (por ejemplo, en un navegador). En otras realizaciones, el código de generación de estilo 1244A es código de otro lenguaje que está configurado para generar CSS válidas cuando se ejecuta con argumentos de entrada apropiados para las expresiones traducidas 1355. En la realización representada de la figura 13, el código de generación de estilo 1244A comprende un conjunto de funciones que aceptan un conjunto de variables/argumentos de entrada para itemWidth, borderPattern y defaultColor, y que devuelve una cadena de CSS válidas basándose en los valores de esos argumentos. En una realización, el código de generación de estilo 1244A está diseñado específicamente para ser rápido. Como un ejemplo, en algunas realizaciones este no incluye ningún análisis sintáctico explícito. En algunas realizaciones, el código de generación de estilo 1244A está diseñado para ejecutarse dentro de un tipo de aplicación particular 1226 (por ejemplo, un navegador web), diseñado para ejecutarse en un servidor, o es flexible para ser ejecutado por cualquiera de los dos.

Por ejemplo, en la realización representada de la figura 13, el código de generación de estilo 1244A generado es código de JavaScript válido, donde cada archivo de CSS parametrizado (es decir, el código de estilo aumentado 1242A) se transforma en código de generación de estilo 1244A que comprende una función de JavaScript que toma varios parámetros como su entrada, uno de los cuales es un mapa de valores puesto a disposición para su utilización dentro de esa función, tal como se define en el archivo de CSS parametrizado de entrada (por ejemplo, código de estilo aumentado 1242A). El código de generación de estilo generado 1244A está diseñado para un alto rendimiento, e incluye principalmente operaciones simples de concatenación de cadenas, junto con cualquier

operación adicional expresada en las CSS parametrizadas. En algunas implementaciones, se prevé que esta primera etapa ocurra infrecuentemente, como una etapa de construcción inicial para la aplicación a implementar.

Este enfoque, que incluye utilizar código de estilo aumentado 1242A para generar código de generación de estilo 1244A, proporciona varias ventajas diferentes que se centran en torno a la flexibilidad y el rendimiento. Este enfoque es flexible porque permite que las reglas de estilo parametrizadas (por ejemplo, reglas de CSS) "finalicen" fácilmente como reglas estándar para múltiples entornos/configuraciones. Adicionalmente, este enfoque es eficiente porque este enfoque en "dos etapas" lleva a cabo los aspectos computacionalmente costosos en su primera etapa de "traducir el código de estilo aumentado 1242A para generar el código de generación de estilo 1244A", lo que deja una cantidad menor de costes computacionales (por ejemplo, en tiempo de ejecución por una estación final relativamente menos potente) en su segunda etapa de invocar el código de generación de estilo 1244A al mismo tiempo que preserva la potencia y expresividad de la parametrización de estilo.

La segunda etapa de la solución incluye hacer que el código de generación de estilo 1244A se implemente en un entorno y se ejecute según sea necesario para obtener el nivel deseado de personalización. Una implementación de esa etapa puede ser ejecutarlo inmediatamente después de que se genere, como parte del mismo proceso de construcción. Esto sería útil principalmente en una situación donde un número finito de parámetros es conocido en tiempo de construcción, ya que los archivos de estilo coincidentes (por ejemplo, archivos de CSS) se pueden generar entonces eficientemente para cada conjunto de parámetros. Por tanto, en algunas realizaciones, la estación final de servidor (es decir, el o los dispositivos informáticos 1230) puede utilizar directamente el código de generación de estilo 1244A para llevar a cabo la "segunda etapa" para generar conjuntos de reglas de estilo válidas 1248 llamando al código de generación de estilo 1244A utilizando un conjunto de invocaciones de generación de estilo 1246. Por ejemplo, el o los dispositivos informáticos 1230, en los medios de almacenamiento legible por ordenador 1232, pueden almacenar un conjunto de invocaciones de generación de estilo para cada usuario y/o interfaz de usuario y/o escenario de implementación que hay que generar para la aplicación. Otra implementación de esta segunda etapa puede ser implementar el código de generación de estilo generado 1244A en servidores web que proporcionan aspectos de la aplicación, y hacer que esos servidores web "finalicen" las reglas de estilo (por ejemplo, CSS) basándose en un conjunto de parámetros que puede variar con cada solicitud.

Otra implementación más de la segunda etapa puede ser implementar el código de generación de estilo generado 1244A como parte de los archivos de JavaScript del lado del cliente que se sirven a los navegadores, y hacer que este código de finalización de estilo se ejecute dentro de cada navegador de usuario utilizando parámetros adquiridos por la aplicación del lado del cliente. Por tanto, en algunas realizaciones, la estación final de servidor (es decir, el o los dispositivos informáticos 1230) puede, en cambio, transmitir el código de generación de estilo 1244A (directa o indirectamente a través de otros dispositivos y/o procesos informáticos) a una estación final 1228, que generará por sí misma conjuntos de reglas de estilo válidas 1248 llamando al código de generación de estilo 1244A utilizando un conjunto de invocaciones de generación de estilo 1246. El conjunto de invocaciones de generación de estilo 1246 se puede almacenar localmente en la estación final 1228, recuperarse de la estación final de servidor o recuperarse de otro dispositivo informático. En una realización, aunque la misma versión del código de generación de estilo 1244A se puede transmitir a muchas estaciones finales diferentes de usuarios potencialmente diferentes de acuerdos de servicio potencialmente diferentes (estipulando qué contenido debe presentar la aplicación), el conjunto de invocaciones de generación de estilo 1246 puede ser único para cada usuario, grupo de usuarios, grupo de dispositivos que comparten una característica común, etc., para permitir que se generen aplicaciones completamente diferentes.

En estas realizaciones de la invención, la separación del análisis sintáctico del código de estilo aumentado 1242 (por ejemplo, CSS parametrizadas) a partir de la generación de las reglas de estilo finales (por ejemplo, invocando el código de generación de estilo 1244A), se consigue de tal manera que permite que la generación de estilo sea rápida, y que pueda realizarse en múltiples entornos, incluyendo en un navegador donde las herramientas de preprocesador de CSS existentes no soportan su utilización de tal forma, o no son prácticas debido a la penalización de rendimiento que se produciría al utilizarlas de tal forma.

La figura 14A ilustra código de invocación de generación de estilo 1246A que, cuando se utiliza para invocar el código de generación de estilo 1244A de la figura 13, da lugar al conjunto ilustrado de reglas de estilo válidas 1248A y a la generación de la interfaz de usuario personalizada 1425A, según una realización de la invención. En esta realización representada, el código de invocación de generación de estilo 1246A comprende código (por ejemplo, código JavaScript, tal como código jQuery) con una serie de variables de entrada 1425 correspondientes a las variables/parámetros del código de generación de estilo 1244A. El código de invocación de generación de estilo 1246A incluye estas variables de entrada 1425 que comprenden un atributo 1405 que coincide con las variables de las expresiones traducidas 1355 del código de generación de estilo 1244A, y valores 1410 a utilizar como argumentos para invocar el código de generación de estilo 1244A para generar el conjunto de reglas de estilo válidas 1248A. En este ejemplo representado, los argumentos de entrada incluyen valores 1410 de "50" para el atributo itemWidth, el valor "solid" para el atributo border-style y el valor "#567A3C" para el atributo defaultColor. Cuando se invoca utilizando estos valores, el código de generación de estilo 1244A generará el conjunto ilustrado de reglas de estilo válidas 1248A, que son válidas según el estándar de estilo. Como un ejemplo, el itemWidth de "50" se pasa al código de generación de estilo 1244A, lo que hace que el primer selector de clase ".rule1" tenga un valor

de "80px" (basándose en la expresión traducida 1355 de "30 + itemWidth" concatenada con "px"), y el segundo selector de clase ".rule2" tenga un valor de "75px" (basándose en la expresión traducida 1355 de "1,5 * itemWidth" concatenada con "px"). De manera similar, los valores de argumentos de entrada de border-style "solid" y defaultColor "#567A3C" hacen que el primer selector de clase tenga valores de "solid" y "#567A3C" (tal como se pasan) y hacen que el segundo selector de clase tenga valores de "solid dashed" y "6A8E50".

En consecuencia, la invocación del código de generación de estilo 1244A, en esta realización, hace que las reglas de estilo se apliquen a la UI personalizada 1425A, lo que incluye un primer conjunto de identificadores de usuario 1445A (por ejemplo, usuarios "que han iniciado sesión en" o "para los que se ha detectado utilización de" la aplicación), un conjunto de elementos de UI de selección de concentrador 1430 que permiten al o a los usuarios llevar a cabo funciones en la aplicación, y un primer elemento de UI 1435A y un segundo elemento de UI 1440A. Para los fines de esta ilustración, el primer elemento de UI 1435A (por ejemplo, , , etc.) tiene una clase de "rule1" y se presentará según la primera regla del conjunto generado de reglas de estilo 1248A, y el segundo elemento de UI 1440A tiene una clase de "rule2" y se presentará según la segunda regla del conjunto generado de reglas de estilo 1248A. En este ejemplo, el primer elemento de UI 1435A tiene una longitud horizontal mayor (80px) comparada con la longitud del segundo elemento de UI 1440A (75px), tal como estipula el conjunto generado de reglas de estilo 1248A. De manera similar, el border-style de "rule1" del conjunto generado de reglas de estilo 1248A hace que el primer elemento de UI 1435A tenga cuatro bordes sólidos, y el border-style de "rule2" del conjunto generado de reglas de estilo 1248A hace que el segundo elemento de UI 1435A tenga unos bordes superior e inferior "solid" y unos bordes izquierdo y derecho "dashed". Adicionalmente, el color de "rule1" del conjunto generado de reglas de estilo 1248A hace que el primer elemento de UI 1435A tenga un borde verde oscuro, y el color de "rule2" del conjunto generado de reglas de estilo 1248A hace que el segundo elemento de UI 1435A tenga un borde verde comparativamente más claro.

Del mismo modo, la figura 14B ilustra código de invocación de generación de estilo adicional 1246B que, cuando se utiliza para invocar el código de generación de estilo 1244A de la figura 13, da lugar a que el conjunto adicional ilustrado de reglas de estilo válidas 1248B y la interfaz de usuario adicional personalizada adicional 1425B, se generen según una realización de la invención. Este código de invocación de generación de estilo 1246B puede, por ejemplo, ser utilizado por otro usuario (por ejemplo, representado por el segundo identificador de usuario 1445B) de la misma aplicación o por otro dispositivo del mismo (o de diferente) usuario para generar una UI personalizada (diferente) 1425B. En este ejemplo representado, el código de invocación de generación de estilo 1246B incluye, en cambio, un itemWidth de "70", un border-style de "dotted" y un defaultColor de "#333333". Invocando el código de generación de estilo 1244A con estos parámetros, el conjunto generado de reglas de estilo válidas 1248B será diferente del conjunto generado de reglas de estilo válidas 1248A, a pesar de que ambas son el resultado de la utilización del mismo código de generación de estilo 1244A. Por tanto, en este ejemplo, el primer elemento de UI 1435B ahora será más corto que el segundo elemento de UI 1440B, tiene cuatro bordes punteados gris oscuro. De manera similar, el segundo elemento de UI 1440B ahora será más largo que el primer elemento de UI 1435B, tiene unos bordes superior e inferior punteados gris claro y unos bordes derecho e izquierdo de trazos gris claro.

La figura 6 ilustra un flujo 1500 en una estación final de servidor (por ejemplo, el o los dispositivos informáticos 1230) para utilizar las reglas de estilo parametrizadas (es decir, el código de estilo aumentado 1242A) para permitir la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según realizaciones de la invención.

El flujo 1500 incluye, en el bloque 1505, transformar un conjunto de una o varias reglas de estilo aumentadas en código de generación de estilo. El conjunto de reglas de código de estilo aumentado incluye sintaxis de estilo (por ejemplo, partes de reglas de CSS) y un conjunto de una o varias expresiones que incluyen un conjunto de una o varias variables. Sin embargo, el conjunto de reglas de código de estilo aumentado no son válidas según un estándar de estilo (por ejemplo, CSS) de la sintaxis de estilo. Este código de generación de estilo, cuando es ejecutado por un conjunto de una o varias invocaciones utilizando un conjunto de una o varias variables de entrada correspondientes al conjunto de variables, genera un conjunto de una o varias reglas de estilo válidas según el estándar de estilo.

El flujo también incluye, en el bloque 1510, ejecutar el código de generación de estilo utilizando el conjunto de variables de entrada para generar el conjunto de reglas de estilo válidas. En una realización, el conjunto de variables de entrada son parte del conjunto de invocaciones, que puede ser código escrito en un lenguaje común con el código de generación de estilo. En una realización, el lenguaje común es JavaScript, y en una realización el conjunto de reglas de estilo válidas son reglas de CSS.

En el bloque 1515, el flujo incluye, además, transmitir el conjunto de reglas de estilo válidas a una estación final de un usuario, haciendo que se presente al usuario una interfaz de usuario personalizada. En una realización, el conjunto de reglas de estilo válidas son parte de un archivo de CSS, que es representado gráficamente mediante una aplicación de navegador que se ejecuta en la estación final.

Opcionalmente, el flujo continúa una o varias veces ejecutando el código de generación de estilo utilizando otro conjunto de variables de entrada para generar otro conjunto de reglas de estilo válidas (en el bloque 1520) y

transmitiendo el otro conjunto de reglas de estilo válidas a otra estación final de otro usuario, lo que hace que se presente al otro usuario otra interfaz de usuario personalizada (en el bloque 1525). Los bloques 1520 y 1525 se pueden ejecutar opcionalmente una o varias veces, para proporcionar de forma sencilla y eficiente interfaces de usuario personalizadas para diferentes usuarios de la aplicación/del sistema. En una realización, las interfaces de usuario son de una aplicación de IPTV para permitir a los usuarios acceder al contenido proporcionado por un sistema de IPTV.

Tal como se ha descrito anteriormente, el código de generación de estilo se puede ejecutar en diversas ubicaciones por diversos dispositivos diferentes. Por ejemplo, el código de generación de estilo se puede ejecutar en una estación final.

La figura 16 ilustra un flujo 1600 en una estación final de servidor (por ejemplo, el o los dispositivos informáticos 1230) para utilizar reglas de estilo parametrizadas (es decir, código de estilo aumentado 1242A) para permitir la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según realizaciones de la invención. Las operaciones de la figura 16 son similares a las operaciones de la figura 15 con la excepción de que la estación final de servidor transmite el código de generación de estilo a una estación final que se configura entonces para ejecutar el código de generación de estilo.

El flujo 1600 incluye, en el bloque 1605, transformar un conjunto de una o varias reglas de estilo aumentadas en código de generación de estilo. El conjunto de reglas de código de estilo aumentado incluye sintaxis de estilo (por ejemplo, partes de reglas de CSS) y un conjunto de una o varias expresiones que incluyen un conjunto de una o varias variables. Sin embargo, el conjunto de reglas de código de estilo aumentado no son válidas según un estándar de estilo (por ejemplo, CSS) de la sintaxis de estilo. Este código de generación de estilo, cuando es ejecutado por un conjunto de una o varias invocaciones utilizando un conjunto de una o varias variables de entrada correspondientes al conjunto de variables, genera un conjunto de una o varias reglas de estilo válidas según el estándar de estilo. El flujo también incluye, en el bloque 1610, transmitir el código de generación de estilo a una estación final de un usuario.

La figura 17 ilustra un flujo 1700 en una estación final para utilizar el código de generación de estilo generado por una estación final de servidor para la generación dinámica en tiempo de ejecución de interfaces de usuario de una aplicación, según realizaciones de la invención. El flujo 1700 incluye, en el bloque 1705, recibir, en un conjunto de interfaces de red de la estación final desde una estación final de servidor, código de generación de estilo. El código de generación de estilo, cuando es ejecutado por un conjunto de una o varias invocaciones utilizando un conjunto de una o varias variables de entrada correspondientes al conjunto de variables, genera un conjunto de una o varias reglas de estilo válidas de un estándar de estilo e inyecta dinámicamente las reglas de estilo válidas en una aplicación. En una realización, las reglas de estilo son reglas de CSS y, en varias realizaciones, la aplicación puede ser una aplicación web que se ejecuta en un navegador o en una aplicación de propósito especial.

Opcionalmente, el flujo 1700 incluye, en el bloque 1710, recibir, desde la estación final de servidor, el conjunto de invocaciones que utilizan el conjunto de variables de entrada. Sin embargo, en otras realizaciones, el conjunto de invocaciones se crea en la estación final (por ejemplo, proporcionando una interfaz de usuario para solicitar los argumentos/valores para las invocaciones) o se recupera de una estación final de servidor diferente o de una estación final diferente.

En el bloque 1715, el flujo incluye, además, hacer que la interfaz de usuario personalizada se presente al usuario como resultado de ejecutar el código de generación de estilo según el conjunto de invocaciones. La interfaz de usuario incluye un conjunto de uno o varios elementos de interfaz de usuario que se diseñan según el conjunto de reglas de estilo válidas.

Aunque las figuras 13 a 17 se enfocan en la generación dinámica de reglas de estilo, las realizaciones también pueden crear dinámicamente otras partes de una aplicación, incluyendo de forma no limitativa componentes y comportamientos, utilizando sistemas similares. Aunque las realizaciones se han descrito en relación con un sistema de IPTV, en su lugar podrían utilizarse realizaciones alternativas para otros sistemas que utilicen aplicaciones personalizables. Por ejemplo, las realizaciones descritas en la presente memoria funcionan en prácticamente cualquier tipo de aplicación/sitio web que pueda beneficiarse de proporcionar "vistas" personalizadas en una aplicación que muestren diferentes interfaces, estilos, funciones, componentes y/o comportamientos.

Aunque las realizaciones dadas a conocer en la presente memoria describen CSS utilizándose como un estándar de estilo y un lenguaje de hojas de estilo, el alcance de la invención no se debe limitar a la utilización de CSS como el único lenguaje de estilo de UI, ya que se pueden utilizar otros lenguajes de estilo de UI.

Adicionalmente, aunque muchas realizaciones dadas a conocer en la presente memoria se enfocan en la parametrización de reglas de estilo (por ejemplo, CSS), el alcance de la invención no está limitado de ese modo. En cambio, en varias realizaciones de la invención, otros tipos de código informático se pueden parametrizar de esta manera, incluyendo de forma no limitativa código de componentes estructurales y/o de comportamiento (por ejemplo, JavaScript), código estructural (por ejemplo, HTML) y prácticamente cualquier otro tipo de código que

pueda formar partes de una aplicación, tanto si es una aplicación web como otro tipo de aplicación.

Un dispositivo electrónico almacena y transmite (internamente y/o con otros dispositivos electrónicos a través de una red) código (que está compuesto por instrucciones de software y que en ocasiones se denomina código de programa informático o un programa informático) y/o datos utilizando medios legibles por máquina (también denominados medios legibles por ordenador), tales como medios de almacenamiento legible por máquina (por ejemplo, discos magnéticos, discos ópticos, memoria de solo lectura (ROM), dispositivos de memoria flash, memoria de cambio de fase) y medios de transmisión legibles por máquina (también denominados una portadora) (por ejemplo, eléctricos, ópticos, de radio, acústicos u otras formas de señales propagadas, tales como ondas portadoras, señales de infrarrojos). Por tanto, un dispositivo electrónico (por ejemplo, un ordenador) incluye hardware y software, tal como un conjunto de uno o varios procesadores acoplados a uno o varios medios de almacenamiento legible por máquina para almacenar código para su ejecución en el conjunto de procesadores y/o para almacenar datos. Por ejemplo, un dispositivo electrónico puede incluir memoria no volátil que contiene el código, dado que la memoria no volátil puede conservar el código incluso cuando el dispositivo electrónico se apaga, y, aunque el dispositivo electrónico se encienda, esa parte del código que se va a ejecutar por el o los procesadores de ese dispositivo electrónico se copia desde la memoria no volátil, más lenta, a la memoria volátil (por ejemplo, memoria de acceso aleatorio dinámica (DRAM), memoria de acceso aleatorio estática (SRAM)) de ese dispositivo electrónico. Los dispositivos electrónicos habituales también incluyen un conjunto de una o varias interfaces de red físicas para establecer conexiones de red (para transmitir y/o recibir código y/o datos utilizando señales de propagación) con otros dispositivos electrónicos. Una o varias partes de una realización de la invención se pueden implementar utilizando diferentes combinaciones de software, software inalterable y/o hardware.

Aunque los diagramas de flujo en las figuras muestran un orden particular de operaciones llevadas a cabo por determinadas realizaciones de la invención, se debe entender que dicho orden es a modo de ejemplo (por ejemplo, realizaciones alternativas pueden llevar a cabo las operaciones en un orden diferente, combinar determinadas operaciones, solapar determinadas operaciones, etc.).

Adicionalmente, aunque la invención se ha descrito en términos de varias realizaciones, los expertos en la materia reconocerán que la invención no se limita a las realizaciones descritas, se puede llevar a la práctica con modificaciones y alteraciones dentro del alcance de las reivindicaciones adjuntas. Por tanto, la descripción se debe considerar como ilustrativa en lugar de limitativa.

REIVINDICACIONES

1. Procedimiento en una estación final (1228) de usuario para la edición interactiva de uno o varios componentes de interfaz de usuario, UI, de una instancia en ejecución de una aplicación web, que comprende:

adquirir un archivo de definiciones de UI que define la aplicación web incluyendo su estructura, comportamiento y aspecto, en el que el archivo (122) de definiciones de UI está configurado para la generación dinámica de código de aplicación que representa un conjunto de uno o varios componentes de UI a presentar dentro de la aplicación web, comprendiendo el código de aplicación por lo menos uno de lenguaje de marcado de hipertexto, HTML, y hojas de estilo en cascada, CSS;
inicializar una UI de la aplicación web utilizando el archivo (122) de definiciones de UI y un conjunto de archivos de definiciones de componentes, incluyendo el conjunto de archivos de definiciones de componentes código para implementar el conjunto de uno o varios componentes de UI indicados en el archivo de definiciones de UI, en el que la aplicación web se genera dinámicamente en tiempo de ejecución bajo el control del archivo (122) de definiciones de UI, y en el que inicializar la UI de la aplicación web utilizando el archivo (122) de definiciones de UI y el conjunto de archivos de definiciones de componentes incluye generar dinámicamente el código de aplicación que representa el conjunto de uno o varios componentes de UI;
cargar dinámicamente (210) en la instancia en ejecución de la aplicación web en la estación final (1228) de usuario un editor interactivo que permite la edición de uno o varios del conjunto de los uno o varios componentes de UI de la instancia en ejecución de la aplicación web;
en respuesta a recibir (215) una selección (128) de usuario de una representación visual de un componente en la instancia en ejecución de la aplicación web, determinar un primer elemento de modelo de objetos de documento, DOM, de la aplicación web;
determinar (220) un primer componente (820) de UI que corresponde al primer elemento de DOM seleccionado;
determinar (225), a partir del archivo (122) de definiciones de UI, un primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado (820);
mostrar (230) un editor de valores (815) que está configurado para mostrar para, por lo menos, uno del primer conjunto de uno o varios parámetros un primer valor y permite que ese primer valor se modifique para cambiar un comportamiento del primer componente de UI;
recibir (235) una modificación de por lo menos el primer valor del por lo menos uno del primer conjunto de uno o varios parámetros para cambiar el comportamiento del primer componente de UI; y
actualizar (240) la instancia en ejecución de la aplicación web para reflejar el primer valor modificado, modificando el archivo (122) de definiciones de UI con el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado (820), y regenerar dinámicamente la instancia en ejecución de la aplicación web bajo el control del archivo de definiciones de UI modificado (122) y el conjunto de archivos de definiciones de componentes para regenerar el primer componente (820) de UI para cambiar el comportamiento del primer componente (820) de UI basándose en el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI.

2. Procedimiento, según la reivindicación 1, que comprende, además:
conservar (245) el archivo de definiciones de UI modificado (122) en almacenamiento de la estación final (1228) de usuario, de tal forma que la aplicación web modificada bajo el control del archivo de definiciones de UI modificado (122) se utilice en futuras sesiones.

3. Procedimiento, según la reivindicación 1,
en el que el archivo (122) de definiciones de UI incluye, además, una pluralidad de definiciones que indican atributos de aspecto visual de partes de la aplicación web, el conjunto de uno o varios componentes que son bloques de construcción de UI a presentar dentro de la aplicación web y un conjunto de comportamientos que se pueden llevar a cabo por el conjunto de uno o varios componentes de UI, en el que el conjunto de comportamientos incluye el comportamiento del primer componente de UI, y en el que la pluralidad de definiciones incluye una pluralidad de pares atributo-valor;
en el que actualizar la instancia en ejecución de la aplicación web para reflejar el primer valor modificado incluye llevar a cabo lo siguiente:
analizar sintácticamente el archivo de definiciones de UI modificado (122) para identificar los atributos de aspecto visual, el conjunto de uno o varios componentes de UI y el conjunto de comportamientos; e
instanciar dinámicamente el conjunto de uno o varios componentes de UI basándose en el archivo de definiciones de UI modificado analizado sintácticamente (122) y el conjunto de uno o varios archivos de definiciones de componentes que incluye código para implementar el conjunto de uno o varios componentes de UI indicados por el archivo (122) de definiciones de UI.

4. Procedimiento, según la reivindicación 1, en el que actualizar la instancia en ejecución de la aplicación web para reflejar el primer valor modificado incluye solo reinicializar la UI de la aplicación web.

5. Procedimiento, según la reivindicación 1, que comprende, además:

en el que la instancia en ejecución de la aplicación web es de una primera plataforma;

simular, en la estación final (1228) de usuario, una UI de una segunda plataforma, en el que la segunda plataforma tiene un archivo (122) de definiciones de UI que es diferente del archivo (122) de definiciones de UI de la primera plataforma; y
actualizar la UI simulada de la segunda plataforma para reflejar el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado.

6. Procedimiento, según la reivindicación 1, que comprende, además:
transmitir un mensaje (930) que describe el primer valor modificado a un servidor (915) que está configurado para distribuir el mensaje a una o varias estaciones finales de usuario diferentes (1228) para hacer que cada una de esas una o varias estaciones finales de usuario diferentes (1228) modifique una instancia en ejecución de una aplicación web en su estación final de usuario respectiva (1228) para reflejar el primer valor modificado.

7. Procedimiento, según la reivindicación 1, que comprende, además:

recibir, desde un servidor (915), un mensaje (1034) que describe un valor modificado para un segundo componente (820) de UI de la aplicación web; y
actualizar la instancia en ejecución de la aplicación web para reflejar el valor modificado para el segundo componente (820) de UI.

8. Medio de almacenamiento no transitorio legible por ordenador que almacena instrucciones para la edición interactiva de uno o varios componentes de interfaz de usuario, UI, de una instancia en ejecución de una aplicación web que, cuando es ejecutado por un conjunto de uno o varios procesadores de una estación final (1228) de usuario, hace que la estación final (1228) de usuario lleve a cabo operaciones que comprenden:

adquirir un archivo de definiciones de UI que define la aplicación web incluyendo su estructura, comportamiento y aspecto, en el que el archivo (122) de definiciones de UI está configurado para la generación dinámica de código de aplicación que representa un conjunto de uno o varios componentes de UI a presentar dentro de la aplicación web, comprendiendo el código de aplicación por lo menos uno de lenguaje de marcado de hipertexto, HTML, y hojas de estilo en cascada, CSS;
inicializar una UI de la aplicación web utilizando el archivo (122) de definiciones de UI y un conjunto de archivos de definiciones de componentes, incluyendo el conjunto de archivos de definiciones de componentes código para implementar el conjunto de uno o varios componentes de UI indicados en el archivo de definiciones de UI, en el que la aplicación web se genera dinámicamente en tiempo de ejecución bajo el control del archivo (122) de definiciones de UI, y en el que inicializar la UI de la aplicación web utilizando el archivo (122) de definiciones de UI y el conjunto de archivos de definiciones de componentes incluye generar dinámicamente el código de aplicación que representa el conjunto de uno o varios componentes de UI;
cargar dinámicamente (210) en la instancia en ejecución de la aplicación web en la estación final (1228) de usuario un editor interactivo que permite la edición de uno o varios del conjunto de los uno o varios componentes de UI de la instancia en ejecución de la aplicación web;
en respuesta a recibir (215) una selección (128) de usuario de una representación visual de un componente en la instancia en ejecución de la aplicación web, determinar un primer elemento de modelo de objetos de documento, DOM, de la aplicación web;
determinar (220) un primer componente (820) de UI que corresponde al primer elemento de DOM seleccionado;
determinar (225), a partir del archivo (122) de definiciones de UI, un primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado (820);
mostrar (230) un editor de valores (815) que está configurado para mostrar para, por lo menos, uno del primer conjunto de uno o varios parámetros un primer valor y permite que ese primer valor se modifique para cambiar un comportamiento del primer componente de UI;
recibir (235) una modificación de por lo menos el primer valor del por lo menos uno del primer conjunto de uno o varios parámetros para cambiar el comportamiento del primer componente de UI; y
actualizar (240) la instancia en ejecución de la aplicación web para reflejar el primer valor modificado, modificando el archivo de definiciones de UI con el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado (820), y regenerar dinámicamente la instancia en ejecución de la aplicación web bajo el control del archivo de definiciones de UI modificado (122) y el conjunto de archivos de definiciones de componentes para regenerar el primer componente (820) de UI para cambiar el comportamiento del primer componente (820) de UI basándose en el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI.

9. Medio de almacenamiento no transitorio legible por ordenador, según la reivindicación 8, en el que el medio de almacenamiento no transitorio legible por ordenador almacena, además, instrucciones que, cuando son ejecutadas por el conjunto de procesadores de la estación final (1228) de usuario, hacen que la estación final (1228) de usuario lleve a cabo, además, el procedimiento según cualquiera de las reivindicaciones 2 a 7.

10. Estación final (1228) de usuario para la edición interactiva de uno o varios componentes de interfaz de usuario, UI, de una instancia en ejecución de una aplicación web, que comprende:

un procesador; y
un medio de almacenamiento no transitorio legible por ordenador acoplado al procesador y que contiene instrucciones ejecutables por dicho procesador mediante las cuales dicha estación final (1228) de usuario es operativa para:
adquirir un archivo de definiciones de UI que define la aplicación web incluyendo su estructura, comportamiento y aspecto, en el que el archivo (122) de definiciones de UI está configurado para la generación dinámica de código de aplicación que representa un conjunto de uno o varios componentes de UI a presentar dentro de la aplicación web, comprendiendo el código de aplicación por lo menos uno de lenguaje de marcado de hipertexto, HTML, y hojas de estilo en cascada, CSS;
inicializar una UI de la aplicación web utilizando el archivo (122) de definiciones de UI y un conjunto de archivos de definiciones de componentes, incluyendo el conjunto de archivos de definiciones de componentes código para implementar el conjunto de uno o varios componentes de UI indicados en el archivo de definiciones de UI, en el que la aplicación web se genera dinámicamente en tiempo de ejecución bajo el control del archivo (122) de definiciones de UI, y en el que la inicialización de la UI de la aplicación web utilizando el archivo (122) de definiciones de UI y el conjunto de archivos de definiciones de componentes incluye una generación dinámica del código de aplicación que representa en conjunto de uno o varios componentes de UI;
cargar dinámicamente (210) en la instancia en ejecución de la aplicación web en la estación final (1228) de usuario un editor interactivo que permite la edición de uno o varios del conjunto de los uno o varios componentes de UI de la instancia en ejecución de la aplicación web;
en respuesta a recibir (215) una selección (128) de usuario de una representación visual de un componente en la instancia en ejecución de la aplicación web, determinar un primer elemento de modelo de objetos de documento, DOM, de la aplicación web;
determinar (220) un primer componente (820) de UI que corresponde al primer elemento de DOM seleccionado;
determinar (225), a partir del archivo (122) de definiciones de UI, un primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado (820);
mostrar (230) un editor de valores (815) que está configurado para mostrar para, por lo menos, uno del primer conjunto de uno o varios parámetros un primer valor y permite que ese primer valor se modifique para cambiar un comportamiento del primer componente de UI;
recibir (235) una modificación de por lo menos el primer valor del por lo menos uno del primer conjunto de uno o varios parámetros para cambiar un comportamiento del primer componente de UI; y
actualizar (240) la instancia en ejecución de la aplicación web para reflejar el primer valor modificado, modificando el archivo (122) de definiciones de UI con el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI determinado (820), y regenerar dinámicamente la instancia en ejecución de la aplicación web bajo el control del archivo de definiciones de UI modificado (122) y el conjunto de archivos de definiciones de componentes para regenerar el primer componente (820) de UI para cambiar el comportamiento del primer componente (820) de UI basándose en el primer valor modificado del por lo menos uno del primer conjunto de uno o varios parámetros que definen el primer componente de UI.

11. Estación final (1228) de usuario, según la reivindicación 10, en la que el medio de almacenamiento no transitorio legible por ordenador contiene, además, instrucciones ejecutables por dicho procesador mediante las cuales dicha estación final (1228) de usuario está operativa, además, para: llevar a cabo el procedimiento según cualquiera de las reivindicaciones 2 a 7.

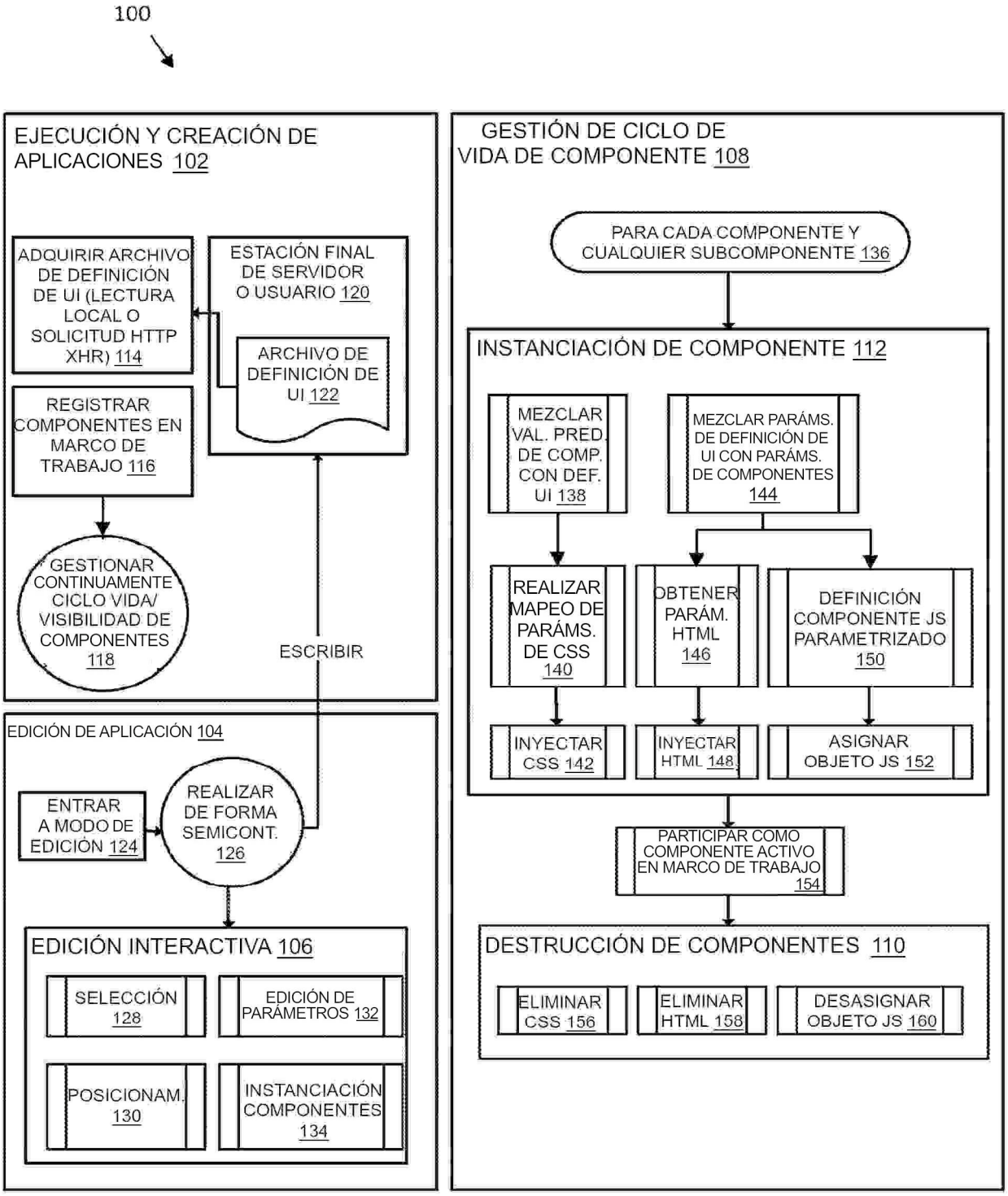

100
EJECUCIÓN Y CREACIÓN DE APLICACIONES 102
ADQUIRIR ARCHIVO DE DEFINICIÓN DE UI (LECTURA LOCAL O SOLICITUD HTTP XHR) 114
ESTACIÓN FINAL DE SERVIDOR O USUARIO 120
ARCHIVO DE DEFINICIÓN DE UI 122
REGISTRAR COMPONENTES EN MARCO DE TRABAJO 116
GESTIONAR CONTINUAMENTE CICLO VIDA/ VISIBILIDAD DE COMPONENTES 118
ESCRIBIR
EDICIÓN DE APLICACIÓN 104
ENTRAR A MODO DE EDICIÓN 124
REALIZAR DE FORMA SEMICONT. 126
EDICIÓN INTERACTIVA 106
SELECCIÓN 128
EDICIÓN DE PARÁMETROS 132
POSICIONAM. 130
INSTANCIACIÓN COMPONENTES 134
GESTIÓN DE CICLO DE VIDA DE COMPONENTE 108
PARA CADA COMPONENTE Y CUALQUIER SUBCOMPONENTE 136
INSTANCIACIÓN DE COMPONENTE 112
MEZCLAR VAL. PRED. DE COMP. CON DEF. UI 138
MEZCLAR PARÁMS. DE DEFINICIÓN DE UI CON PARÁMS. DE COMPONENTES 144
REALIZAR MAPEO DE PARÁMS. DE CSS 140
OBTENER PARÁM. HTML 146
DEFINICIÓN COMPONENTE JS PARAMETRIZADO 150
INYECTAR CSS 142
INYECTAR HTML 148
ASIGNAR OBJETO JS 152
PARTICIPAR COMO COMPONENTE ACTIVO EN MARCO DE TRABAJO 154
DESTRUCCIÓN DE COMPONENTES 110
ELIMINAR CSS 156
ELIMINAR HTML 158
DESASIGNAR OBJETO JS 160

FIG. 1

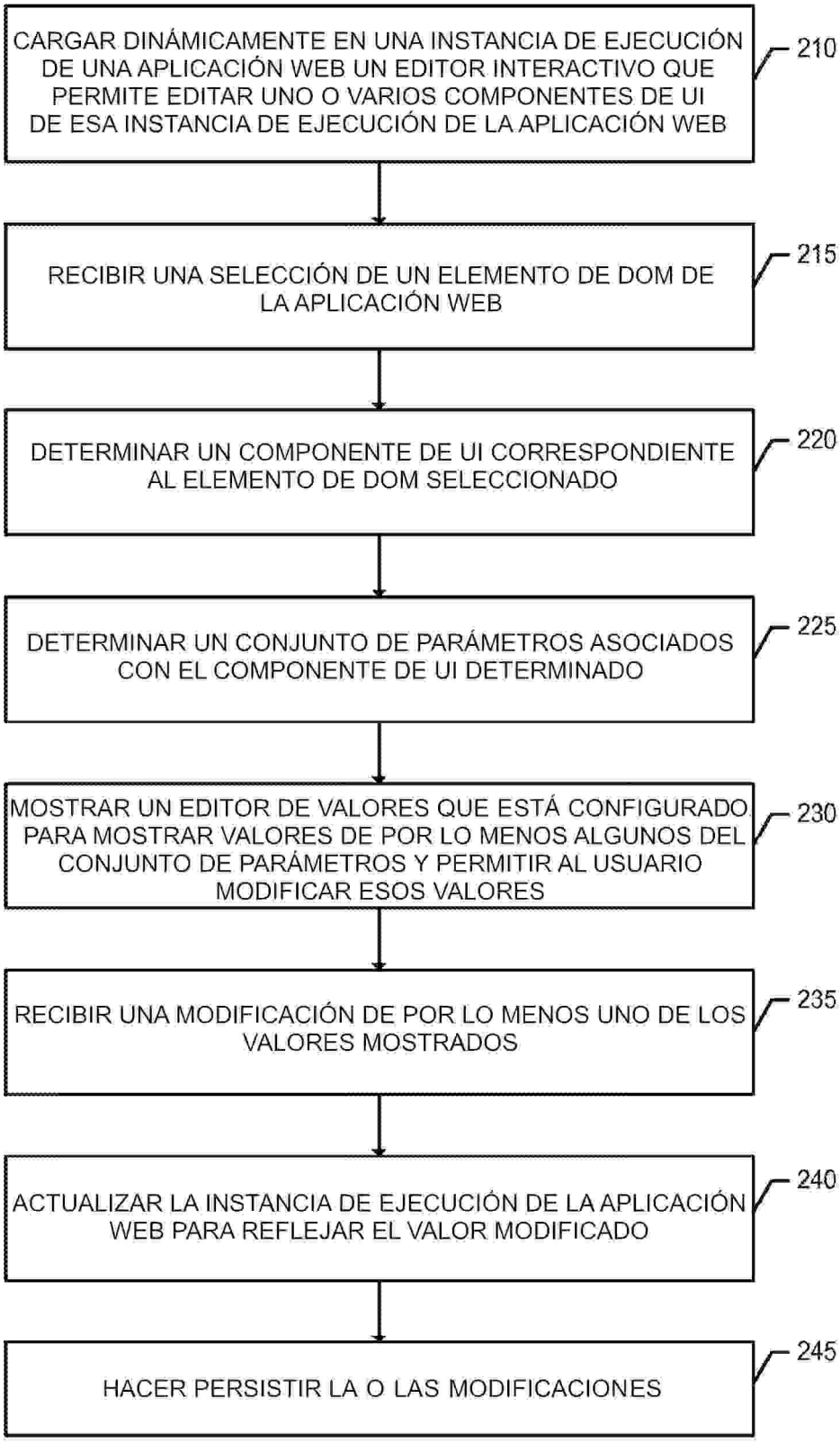

CARGAR DINÁMICAMENTE EN UNA INSTANCIA DE EJECUCIÓN DE UNA APLICACIÓN WEB UN EDITOR INTERACTIVO QUE PERMITE EDITAR UNO O VARIOS COMPONENTES DE UI DE ESA INSTANCIA DE EJECUCIÓN DE LA APLICACIÓN WEB
210
RECIBIR UNA SELECCIÓN DE UN ELEMENTO DE DOM DE LA APLICACIÓN WEB
215
DETERMINAR UN COMPONENTE DE UI CORRESPONDIENTE AL ELEMENTO DE DOM SELECCIONADO
220
DETERMINAR UN CONJUNTO DE PARÁMETROS ASOCIADOS CON EL COMPONENTE DE UI DETERMINADO
225
MOSTRAR UN EDITOR DE VALORES QUE ESTÁ CONFIGURADO PARA MOSTRAR VALORES DE POR LO MENOS ALGUNOS DEL CONJUNTO DE PARÁMETROS Y PERMITIR AL USUARIO MODIFICAR ESOS VALORES
230
RECIBIR UNA MODIFICACIÓN DE POR LO MENOS UNO DE LOS VALORES MOSTRADOS
235
ACTUALIZAR LA INSTANCIA DE EJECUCIÓN DE LA APLICACIÓN WEB PARA REFLEJAR EL VALOR MODIFICADO
240
HACER PERSISTIR LA O LAS MODIFICACIONES
245

FIG. 2

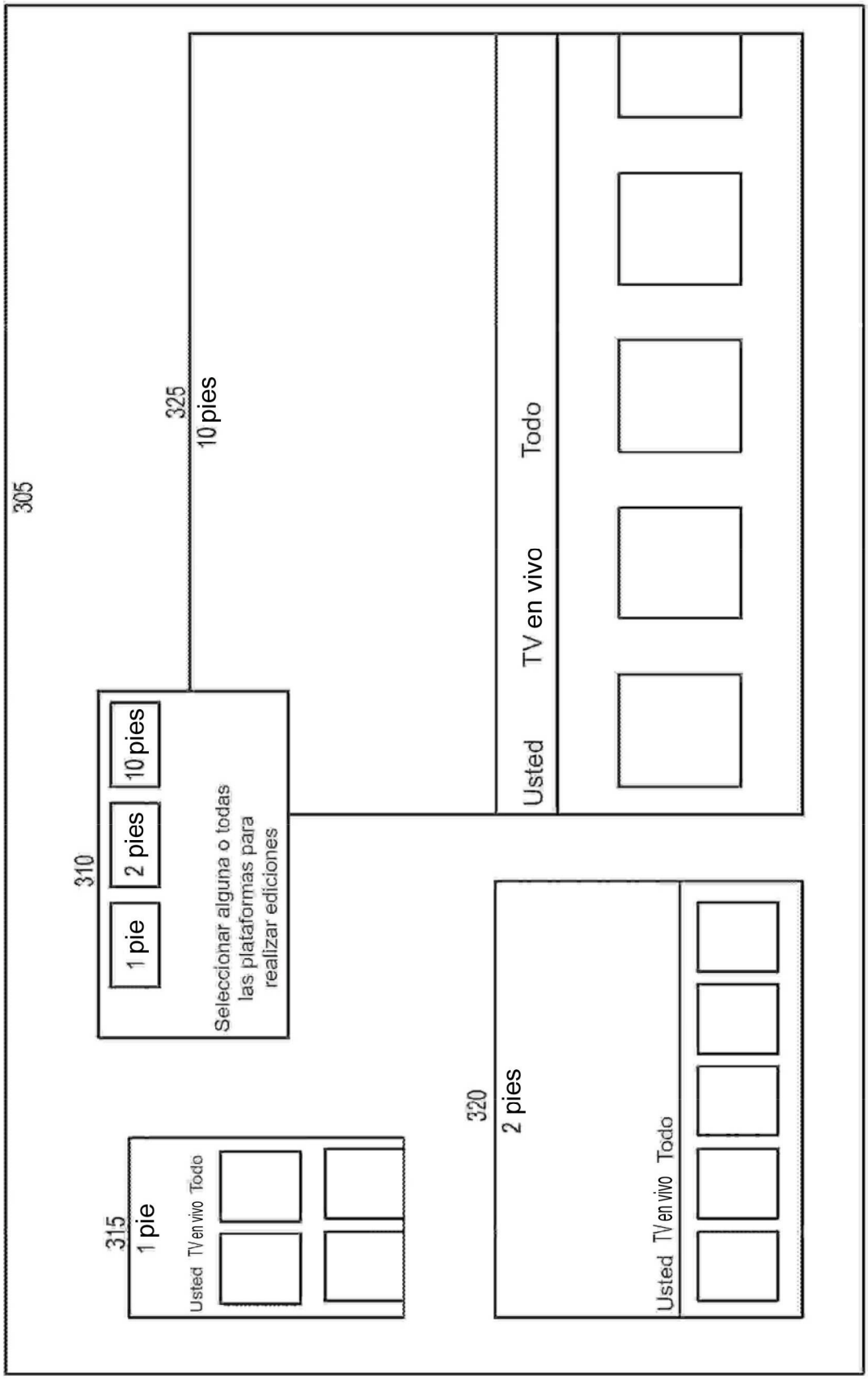

305
325
10 pies
310
1 pie
2 pies
10 pies
Seleccionar alguna o todas las plataformas para realizar ediciones
Usted
TV en vivo
Todo
315
1 pie
Usted TV en vivo Todo
320
2 pies
Usted TV en vivo Todo

FIG. 3

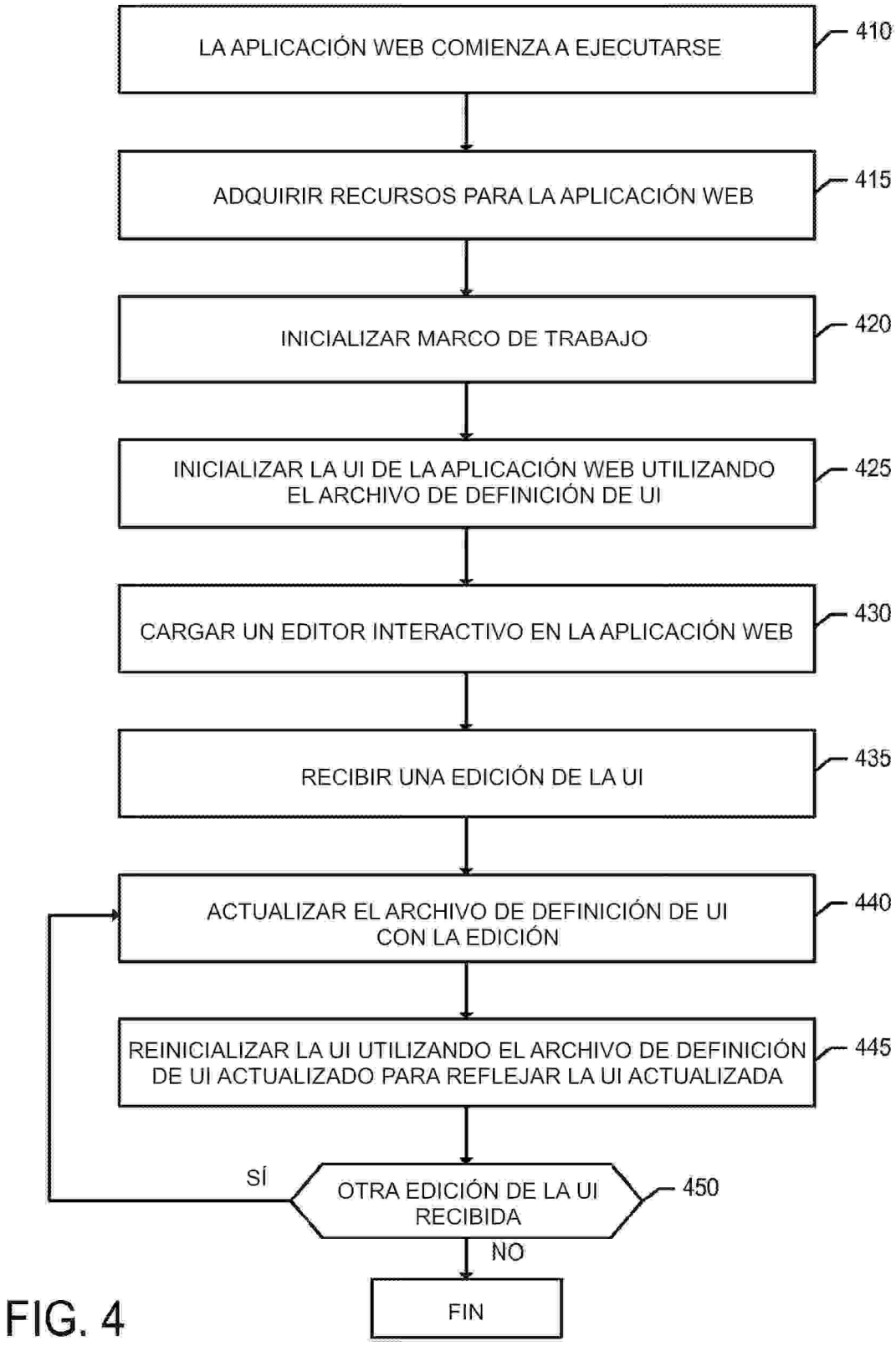

LA APLICACIÓN WEB COMIENZA A EJECUTARSE
410
ADQUIRIR RECURSOS PARA LA APLICACIÓN WEB
415
INICIALIZAR MARCO DE TRABAJO
420
INICIALIZAR LA UI DE LA APLICACIÓN WEB UTILIZANDO EL ARCHIVO DE DEFINICIÓN DE UI
425
CARGAR UN EDITOR INTERACTIVO EN LA APLICACIÓN WEB
430
RECIBIR UNA EDICIÓN DE LA UI
435
ACTUALIZAR EL ARCHIVO DE DEFINICIÓN DE UI CON LA EDICIÓN
440
REINICIALIZAR LA UI UTILIZANDO EL ARCHIVO DE DEFINICIÓN DE UI ACTUALIZADO PARA REFLEJAR LA UI ACTUALIZADA
445
SÍ
OTRA EDICIÓN DE LA UI RECIBIDA
450
NO
FIN

FIG. 4

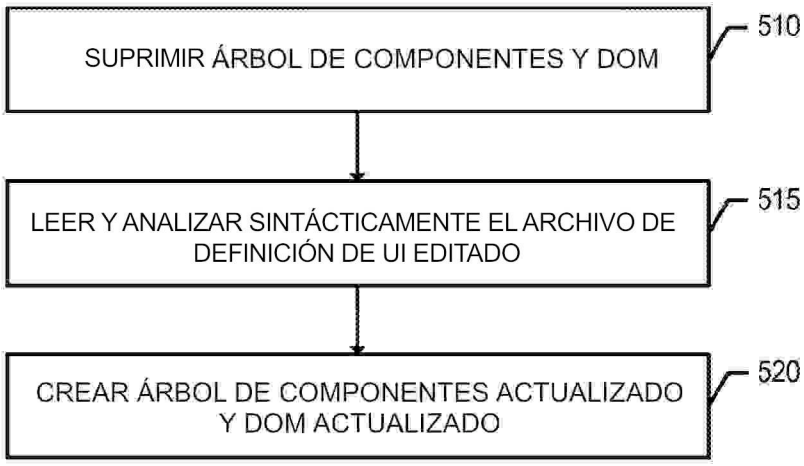

SUPRIMIR ÁRBOL DE COMPONENTES Y DOM
510
LEER Y ANALIZAR SINTÁCTICAMENTE EL ARCHIVO DE DEFINICIÓN DE UI EDITADO
515
CREAR ÁRBOL DE COMPONENTES ACTUALIZADO Y DOM ACTUALIZADO
520

FIG. 5

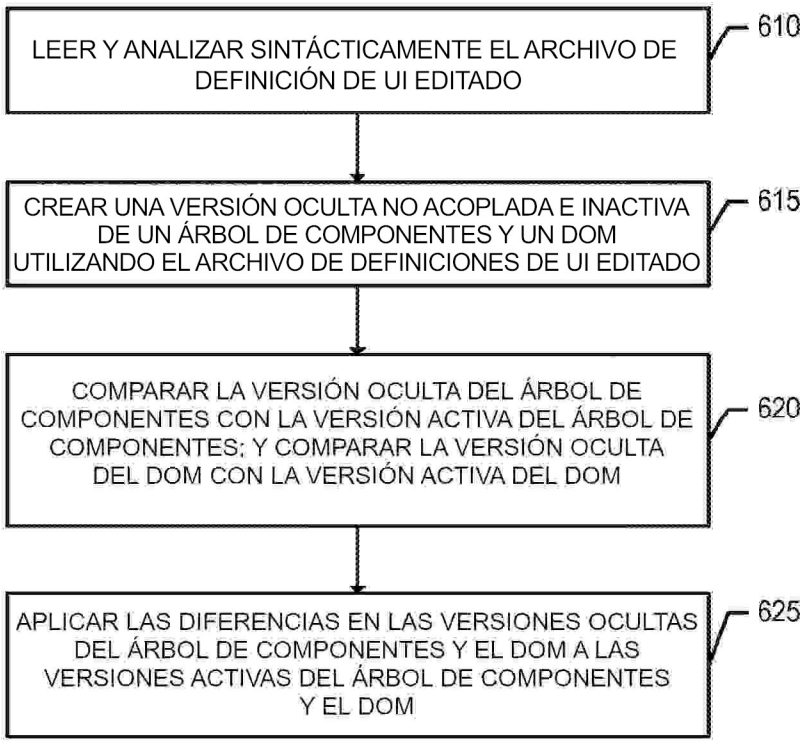

LEER Y ANALIZAR SINTÁCTICAMENTE EL ARCHIVO DE DEFINICIÓN DE UI EDITADO
610
CREAR UNA VERSIÓN OCULTA NO ACOPLADA E INACTIVA DE UN ÁRBOL DE COMPONENTES Y UN DOM UTILIZANDO EL ARCHIVO DE DEFINICIONES DE UI EDITADO
615
COMPARAR LA VERSIÓN OCULTA DEL ÁRBOL DE COMPONENTES CON LA VERSIÓN ACTIVA DEL ÁRBOL DE COMPONENTES; Y COMPARAR LA VERSIÓN OCULTA DEL DOM CON LA VERSIÓN ACTIVA DEL DOM
620
APLICAR LAS DIFERENCIAS EN LAS VERSIONES OCULTAS DEL ÁRBOL DE COMPONENTES Y EL DOM A LAS VERSIONES ACTIVAS DEL ÁRBOL DE COMPONENTES Y EL DOM
625

FIG. 6

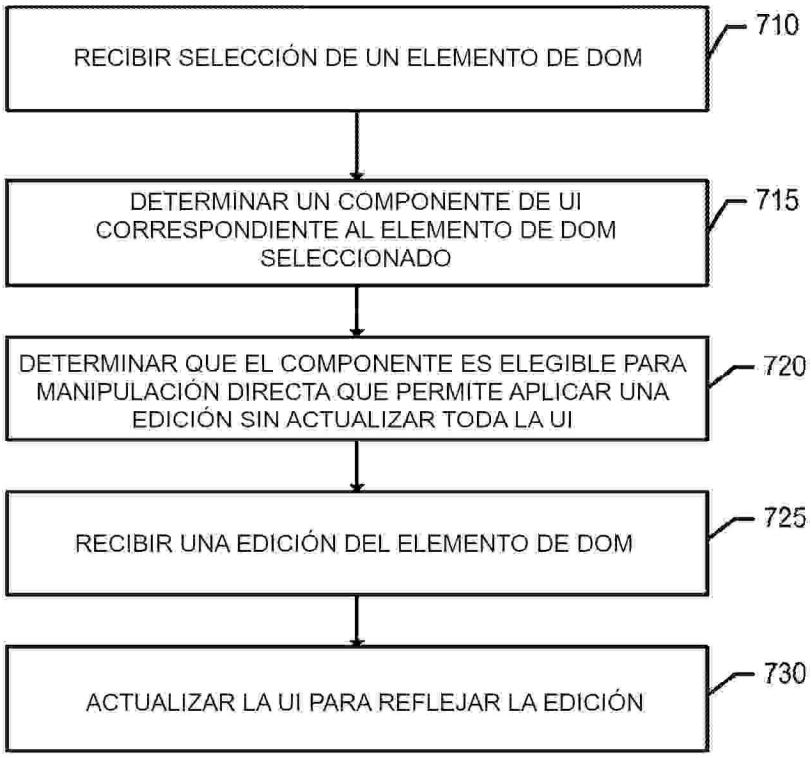

RECIBIR SELECCIÓN DE UN ELEMENTO DE DOM
710
DETERMINAR UN COMPONENTE DE UI CORRESPONDIENTE AL ELEMENTO DE DOM SELECCIONADO
715
DETERMINAR QUE EL COMPONENTE ES ELEGIBLE PARA MANIPULACIÓN DIRECTA QUE PERMITE APLICAR UNA EDICIÓN SIN ACTUALIZAR TODA LA UI
720
RECIBIR UNA EDICIÓN DEL ELEMENTO DE DOM
725
ACTUALIZAR LA UI PARA REFLEJAR LA EDICIÓN
730

FIG. 7

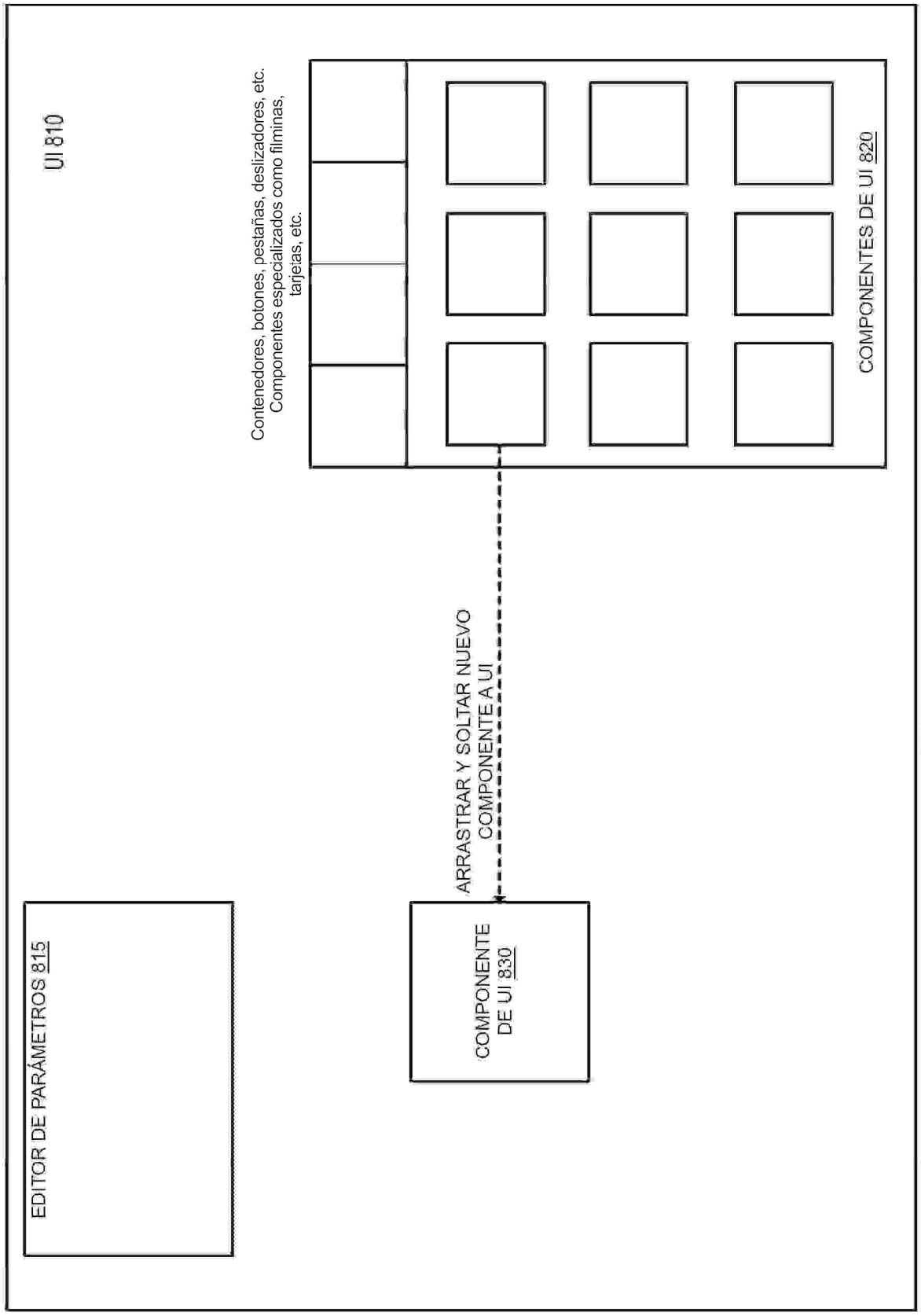

UI 810
Contenedores, botones, pestañas, deslizadores, etc.
Componentes especializados como filminas, tarjetas, etc.
COMPONENTES DE UI 820
ARRASTRAR Y SOLTAR NUEVO COMPONENTE A UI
COMPONENTE DE UI 830
EDITOR DE PARÁMETROS 815

FIG. 8

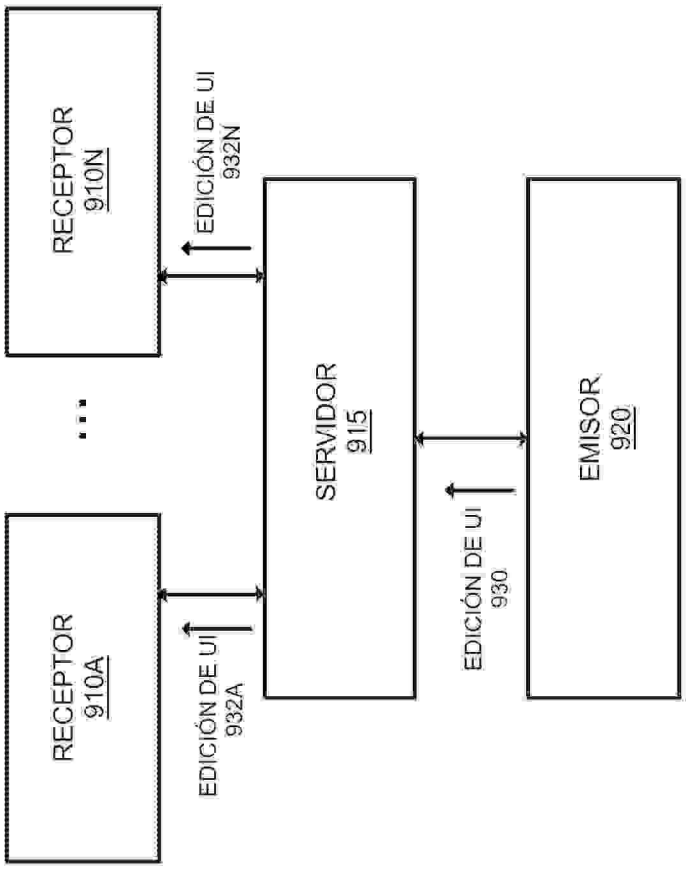

RECEPTOR
910N
EDICIÓN DE UI
932N
...
SERVIDOR
915
EMISOR
920
RECEPTOR
910A
EDICIÓN DE UI
932A
EDICIÓN DE UI
930

FIG. 9

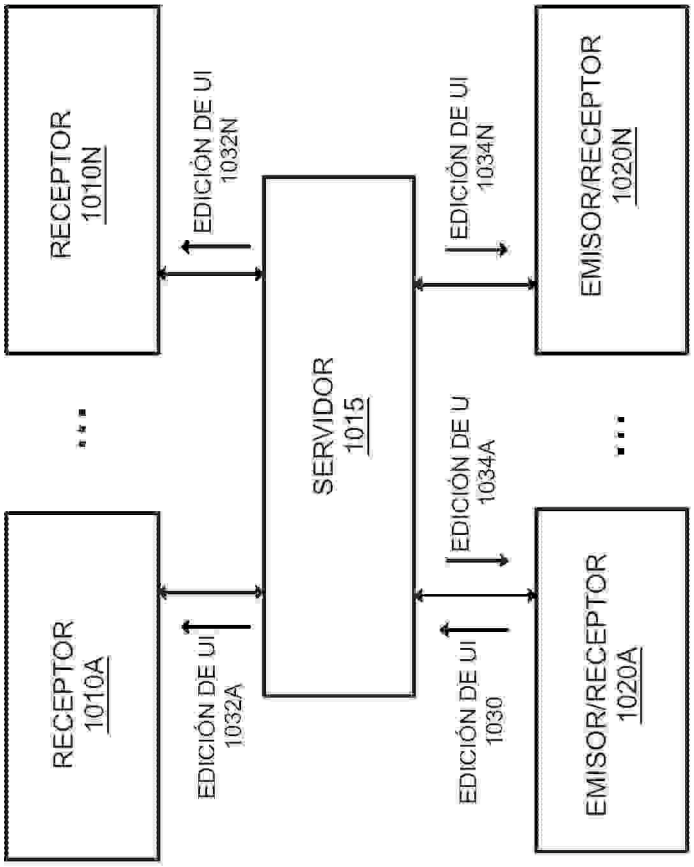

RECEPTOR
1010N
EDICIÓN DE UI
1032N
SERVIDOR
1015
EDICIÓN DE UI
1034N
EMISOR/RECEPTOR
1020N
RECEPTOR
1010A
EDICIÓN DE UI
1032A
EDICIÓN DE U
1034A
EDICIÓN DE UI
1030
EMISOR/RECEPTOR
1020A

FIG. 10

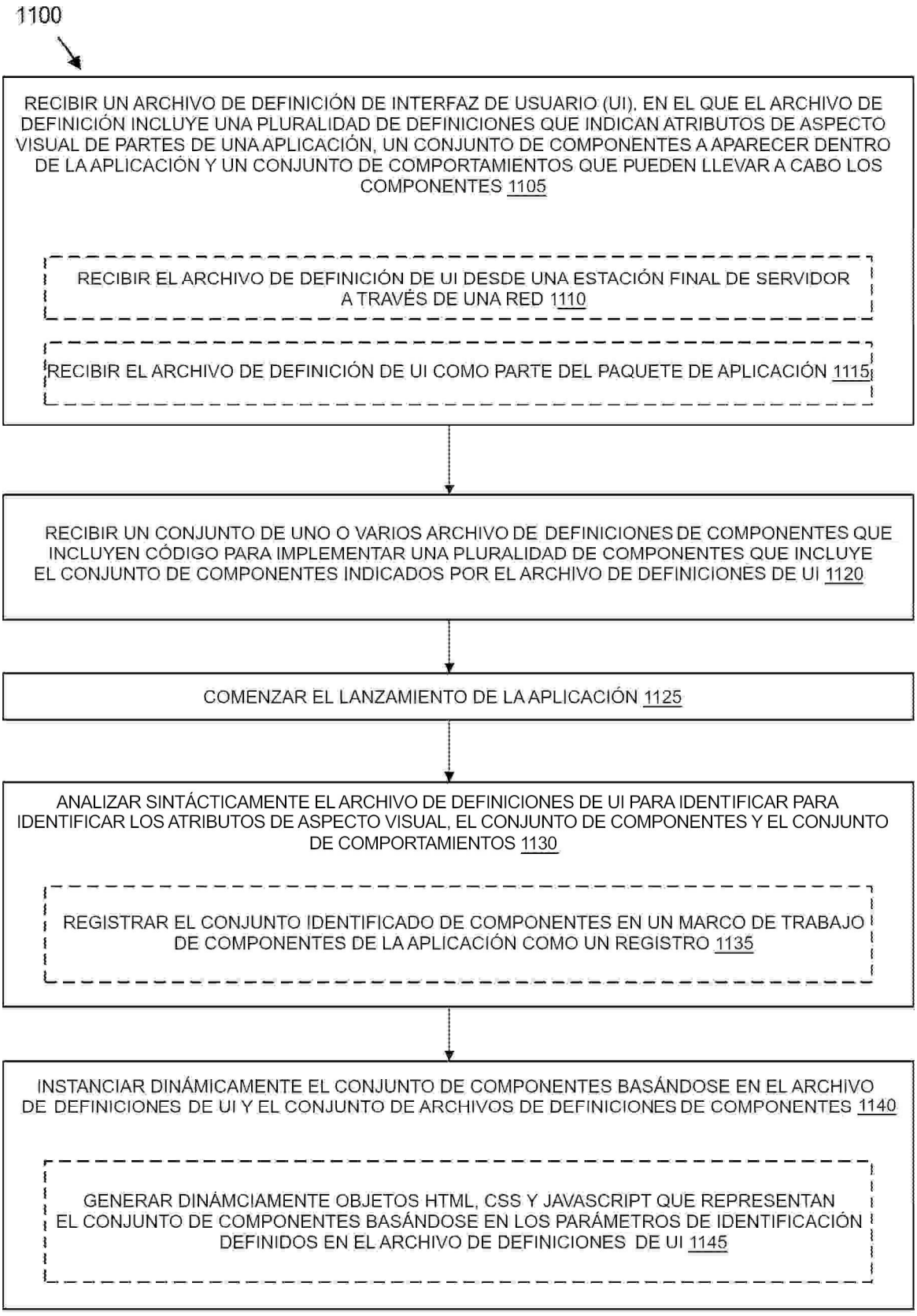

1100
RECIBIR UN ARCHIVO DE DEFINICIÓN DE INTERFAZ DE USUARIO (UI), EN EL QUE EL ARCHIVO DE DEFINICIÓN INCLUYE UNA PLURALIDAD DE DEFINICIONES QUE INDICAN ATRIBUTOS DE ASPECTO VISUAL DE PARTES DE UNA APLICACIÓN, UN CONJUNTO DE COMPONENTES A APARECER DENTRO DE LA APLICACIÓN Y UN CONJUNTO DE COMPORTAMIENTOS QUE PUEDEN LLEVAR A CABO LOS COMPONENTES 1105
RECIBIR EL ARCHIVO DE DEFINICIÓN DE UI DESDE UNA ESTACIÓN FINAL DE SERVIDOR A TRAVÉS DE UNA RED 1110
RECIBIR EL ARCHIVO DE DEFINICIÓN DE UI COMO PARTE DEL PAQUETE DE APLICACIÓN 1115
RECIBIR UN CONJUNTO DE UNO O VARIOS ARCHIVO DE DEFINICIONES DE COMPONENTES QUE INCLUYEN CÓDIGO PARA IMPLEMENTAR UNA PLURALIDAD DE COMPONENTES QUE INCLUYE EL CONJUNTO DE COMPONENTES INDICADOS POR EL ARCHIVO DE DEFINICIONES DE UI 1120
COMENZAR EL LANZAMIENTO DE LA APLICACIÓN 1125
ANALIZAR SINTÁCTICAMENTE EL ARCHIVO DE DEFINICIONES DE UI PARA IDENTIFICAR PARA IDENTIFICAR LOS ATRIBUTOS DE ASPECTO VISUAL, EL CONJUNTO DE COMPONENTES Y EL CONJUNTO DE COMPORTAMIENTOS 1130
REGISTRAR EL CONJUNTO IDENTIFICADO DE COMPONENTES EN UN MARCO DE TRABAJO DE COMPONENTES DE LA APLICACIÓN COMO UN REGISTRO 1135
INSTANCIAR DINÁMICAMENTE EL CONJUNTO DE COMPONENTES BASÁNDOSE EN EL ARCHIVO DE DEFINICIONES DE UI Y EL CONJUNTO DE ARCHIVOS DE DEFINICIONES DE COMPONENTES 1140
GENERAR DINÁMCIAMENTE OBJETOS HTML, CSS Y JAVASCRIPT QUE REPRESENTAN EL CONJUNTO DE COMPONENTES BASÁNDOSE EN LOS PARÁMETROS DE IDENTIFICACIÓN DEFINIDOS EN EL ARCHIVO DE DEFINICIONES DE UI 1145

FIG. 11

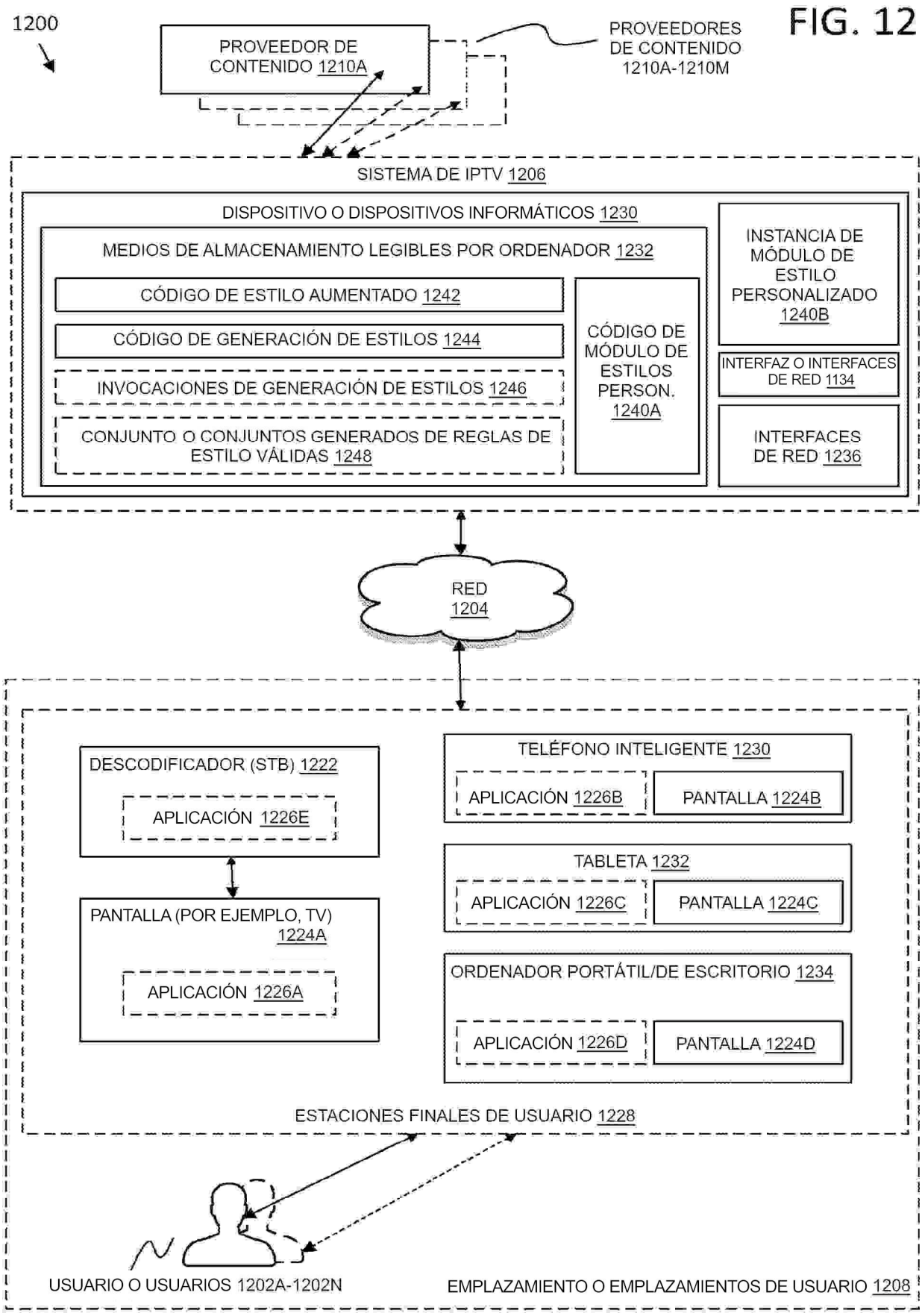

1200
PROVEEDOR DE CONTENIDO 1210A
PROVEEDORES DE CONTENIDO 1210A-1210M
SISTEMA DE IPTV 1206
DISPOSITIVO O DISPOSITIVOS INFORMÁTICOS 1230
MEDIOS DE ALMACENAMIENTO LEGIBLES POR ORDENADOR 1232
CÓDIGO DE ESTILO AUMENTADO 1242
CÓDIGO DE GENERACIÓN DE ESTILOS 1244
INVOCACIONES DE GENERACIÓN DE ESTILOS 1246
CONJUNTO O CONJUNTOS GENERADOS DE REGLAS DE ESTILO VÁLIDAS 1248
CÓDIGO DE MÓDULO DE ESTILOS PERSON. 1240A
INSTANCIA DE MÓDULO DE ESTILO PERSONALIZADO 1240B
INTERFAZ O INTERFACES DE RED 1134
INTERFACES DE RED 1236
RED 1204
DESCODIFICADOR (STB) 1222
APLICACIÓN 1226E
PANTALLA (POR EJEMPLO, TV) 1224A
APLICACIÓN 1226A
TELÉFONO INTELIGENTE 1230
APLICACIÓN 1226B
PANTALLA 1224B
TABLETA 1232
APLICACIÓN 1226C
PANTALLA 1224C
ORDENADOR PORTÁTIL/DE ESCRITORIO 1234
APLICACIÓN 1226D
PANTALLA 1224D
ESTACIONES FINALES DE USUARIO 1228
USUARIO O USUARIOS 1202A-1202N
EMPLAZAMIENTO O EMPLAZAMIENTOS DE USUARIO 1208

FIG. 12

FIG. 13

DELIMITADOR DE EXPRESIÓN 1305
EXPRESIÓN 1310
VARIABLE 1315
SINTAXIS DE ESTILO 1320
SINTAXIS DE ESTILO 1320
CÓDIGO DE ESTILO AUMENTADO 1242A
CÓDIGO DE GENERACIÓN DE ESTILO 1244A
EXPRESIÓN TRADUCIDA 1355
.rule1 {
width: ${ 30 + itemWidth }px;
border-style: ${ borderPattern };
color: ${ defaultColor };
}
.rule2 {
width: ${ 1.5 * itemWidth + "px"};
border-style: ${ borderPattern } dashed;
color: ${ lighten(defaultColor) };
}
(function (css) {
var map = {
sample: function (config) {
with(config) return ".rule1 {\r\n width: " + (30 + itemWidth) + "px;\r\n"
+ "border-style: " + borderPattern + ";\r\n"
+ "color: " + (defaultColor) + ";\r\n}" + "\r\n\r\n"
+ ".rule2 {\r\n width: " + (1.5 * itemWidth) + "px;\r\n"
+ "border-style: " + borderPattern + " dashed;\r\n"
+ "color: " + (lighten(defaultColor)) + ";\r\n}"
}
};
function getCSS(id, config) {
return map[id](config)
}
css.getCSS = getCSS;
})(window.css || (css = {}));

FIG. 14A

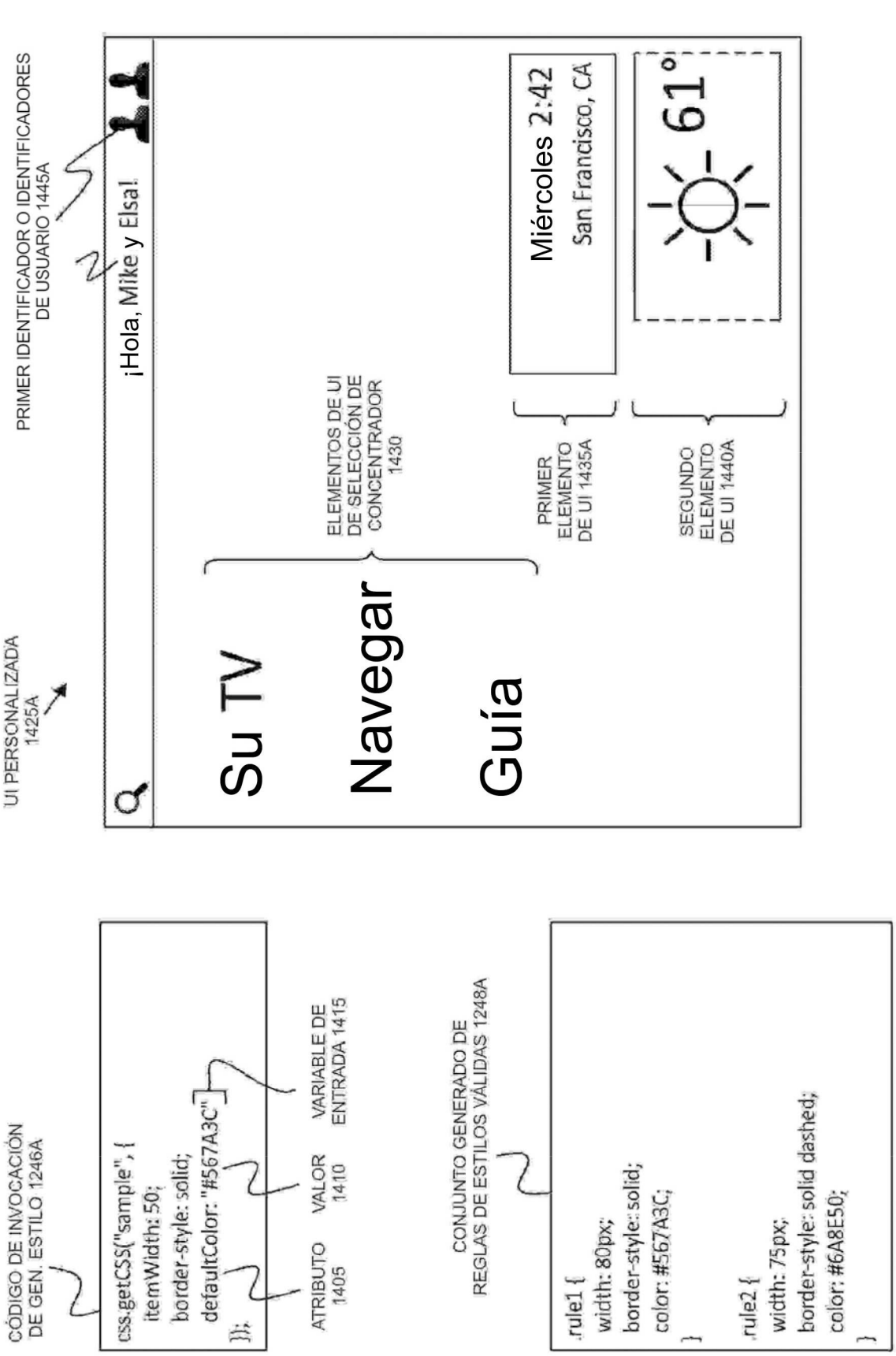
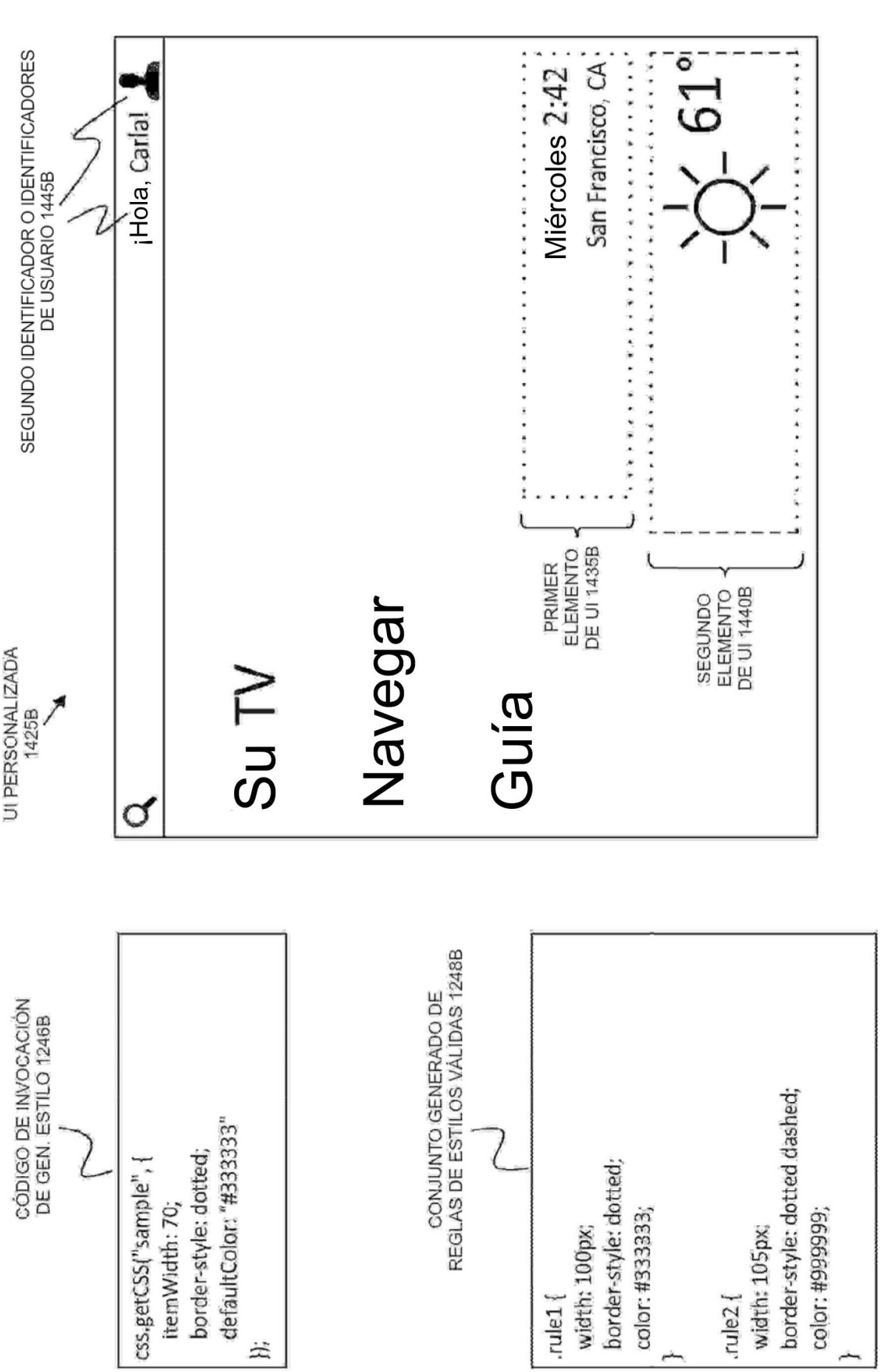

UI PERSONALIZADA
1425A
PRIMER IDENTIFICADOR O IDENTIFICADORES DE USUARIO 1445A
¡Hola, Mike y Elsa!
Su TV
Navegar
Guía
ELEMENTOS DE UI DE SELECCIÓN DE CONCENTRADOR 1430
PRIMER ELEMENTO DE UI 1435A
SEGUNDO ELEMENTO DE UI 1440A
Miércoles 2:42
San Francisco, CA
61°
CÓDIGO DE INVOCACIÓN DE GEN. ESTILO 1246A
css.getCSS("sample", {
itemWidth: 50;
border-style: solid;
defaultColor: "#567A3C"
});
ATRIBUTO 1405
VALOR 1410
VARIABLE DE ENTRADA 1415
CONJUNTO GENERADO DE REGLAS DE ESTILOS VÁLIDAS 1248A
.rule1 {
width: 80px;
border-style: solid;
color: #567A3C;
}
.rule2 {
width: 75px;
border-style: solid dashed;
color: #6A8E50;
}

FIG. 14B


UI PERSONALIZADA 1425B
SEGUNDO IDENTIFICADOR O IDENTIFICADORES DE USUARIO 1445B
¡Hola, Carla!
Q
Su TV
Navegar
Guía
PRIMER ELEMENTO DE UI 1435B
Miércoles 2:42
San Francisco, CA
61°
SEGUNDO ELEMENTO DE UI 1440B
CÓDIGO DE INVOCACIÓN DE GEN. ESTILO 1246B
css.getCSS("sample",{
itemWidth: 70;
border-style: dotted;
defaultColor: "#333333"
});
CONJUNTO GENERADO DE REGLAS DE ESTILOS VÁLIDAS 1248B
.rule1 {
width: 100px;
border-style: dotted;
color: #333333;
}
.rule2 {
width: 105px;
border-style: dotted dashed;
color: #999999;
}

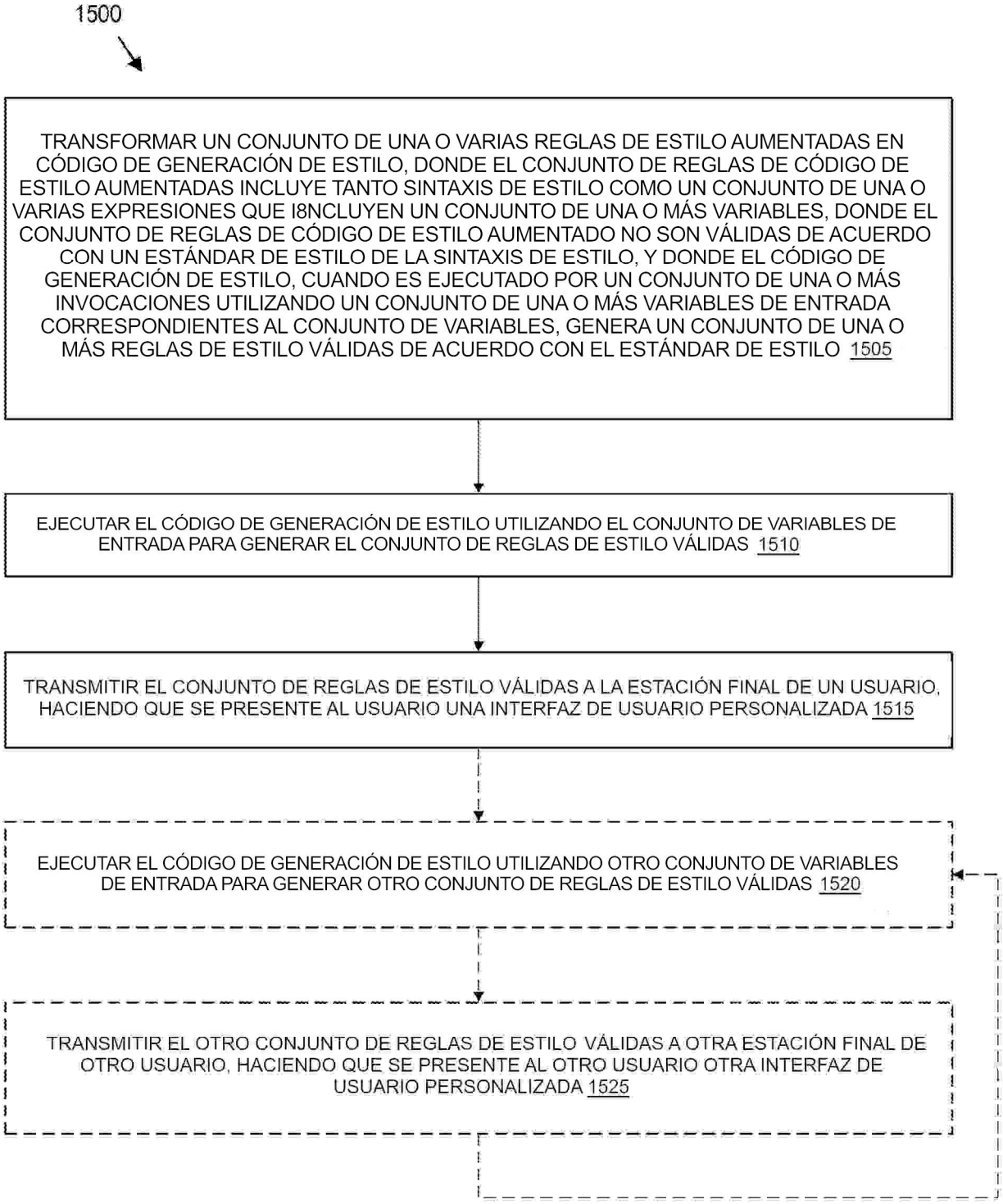

1500
TRANSFORMAR UN CONJUNTO DE UNA O VARIAS REGLAS DE ESTILO AUMENTADAS EN CÓDIGO DE GENERACIÓN DE ESTILO, DONDE EL CONJUNTO DE REGLAS DE CÓDIGO DE ESTILO AUMENTADAS INCLUYE TANTO SINTAXIS DE ESTILO COMO UN CONJUNTO DE UNA O VARIAS EXPRESIONES QUE I8NCLUYEN UN CONJUNTO DE UNA O MÁS VARIABLES, DONDE EL CONJUNTO DE REGLAS DE CÓDIGO DE ESTILO AUMENTADO NO SON VÁLIDAS DE ACUERDO CON UN ESTÁNDAR DE ESTILO DE LA SINTAXIS DE ESTILO, Y DONDE EL CÓDIGO DE GENERACIÓN DE ESTILO, CUANDO ES EJECUTADO POR UN CONJUNTO DE UNA O MÁS INVOCACIONES UTILIZANDO UN CONJUNTO DE UNA O MÁS VARIABLES DE ENTRADA CORRESPONDIENTES AL CONJUNTO DE VARIABLES, GENERA UN CONJUNTO DE UNA O MÁS REGLAS DE ESTILO VÁLIDAS DE ACUERDO CON EL ESTÁNDAR DE ESTILO 1505
EJECUTAR EL CÓDIGO DE GENERACIÓN DE ESTILO UTILIZANDO EL CONJUNTO DE VARIABLES DE ENTRADA PARA GENERAR EL CONJUNTO DE REGLAS DE ESTILO VÁLIDAS 1510
TRANSMITIR EL CONJUNTO DE REGLAS DE ESTILO VÁLIDAS A LA ESTACIÓN FINAL DE UN USUARIO, HACIENDO QUE SE PRESENTE AL USUARIO UNA INTERFAZ DE USUARIO PERSONALIZADA 1515
EJECUTAR EL CÓDIGO DE GENERACIÓN DE ESTILO UTILIZANDO OTRO CONJUNTO DE VARIABLES DE ENTRADA PARA GENERAR OTRO CONJUNTO DE REGLAS DE ESTILO VÁLIDAS 1520
TRANSMITIR EL OTRO CONJUNTO DE REGLAS DE ESTILO VÁLIDAS A OTRA ESTACIÓN FINAL DE OTRO USUARIO, HACIENDO QUE SE PRESENTE AL OTRO USUARIO OTRA INTERFAZ DE USUARIO PERSONALIZADA 1525

FIG. 15

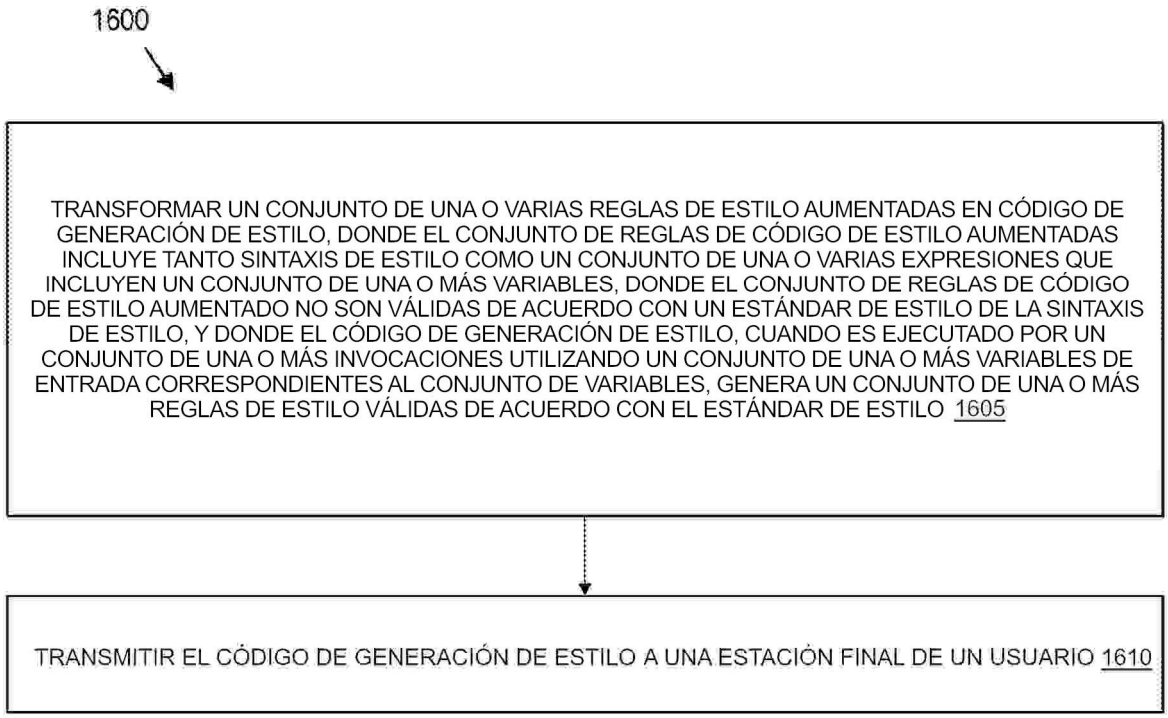

1600
TRANSFORMAR UN CONJUNTO DE UNA O VARIAS REGLAS DE ESTILO AUMENTADAS EN CÓDIGO DE GENERACIÓN DE ESTILO, DONDE EL CONJUNTO DE REGLAS DE CÓDIGO DE ESTILO AUMENTADAS INCLUYE TANTO SINTAXIS DE ESTILO COMO UN CONJUNTO DE UNA O VARIAS EXPRESIONES QUE INCLUYEN UN CONJUNTO DE UNA O MÁS VARIABLES, DONDE EL CONJUNTO DE REGLAS DE CÓDIGO DE ESTILO AUMENTADO NO SON VÁLIDAS DE ACUERDO CON UN ESTÁNDAR DE ESTILO DE LA SINTAXIS DE ESTILO, Y DONDE EL CÓDIGO DE GENERACIÓN DE ESTILO, CUANDO ES EJECUTADO POR UN CONJUNTO DE UNA O MÁS INVOCACIONES UTILIZANDO UN CONJUNTO DE UNA O MÁS VARIABLES DE ENTRADA CORRESPONDIENTES AL CONJUNTO DE VARIABLES, GENERA UN CONJUNTO DE UNA O MÁS REGLAS DE ESTILO VÁLIDAS DE ACUERDO CON EL ESTÁNDAR DE ESTILO 1605
TRANSMITIR EL CÓDIGO DE GENERACIÓN DE ESTILO A UNA ESTACIÓN FINAL DE UN USUARIO 1610

FIG. 16

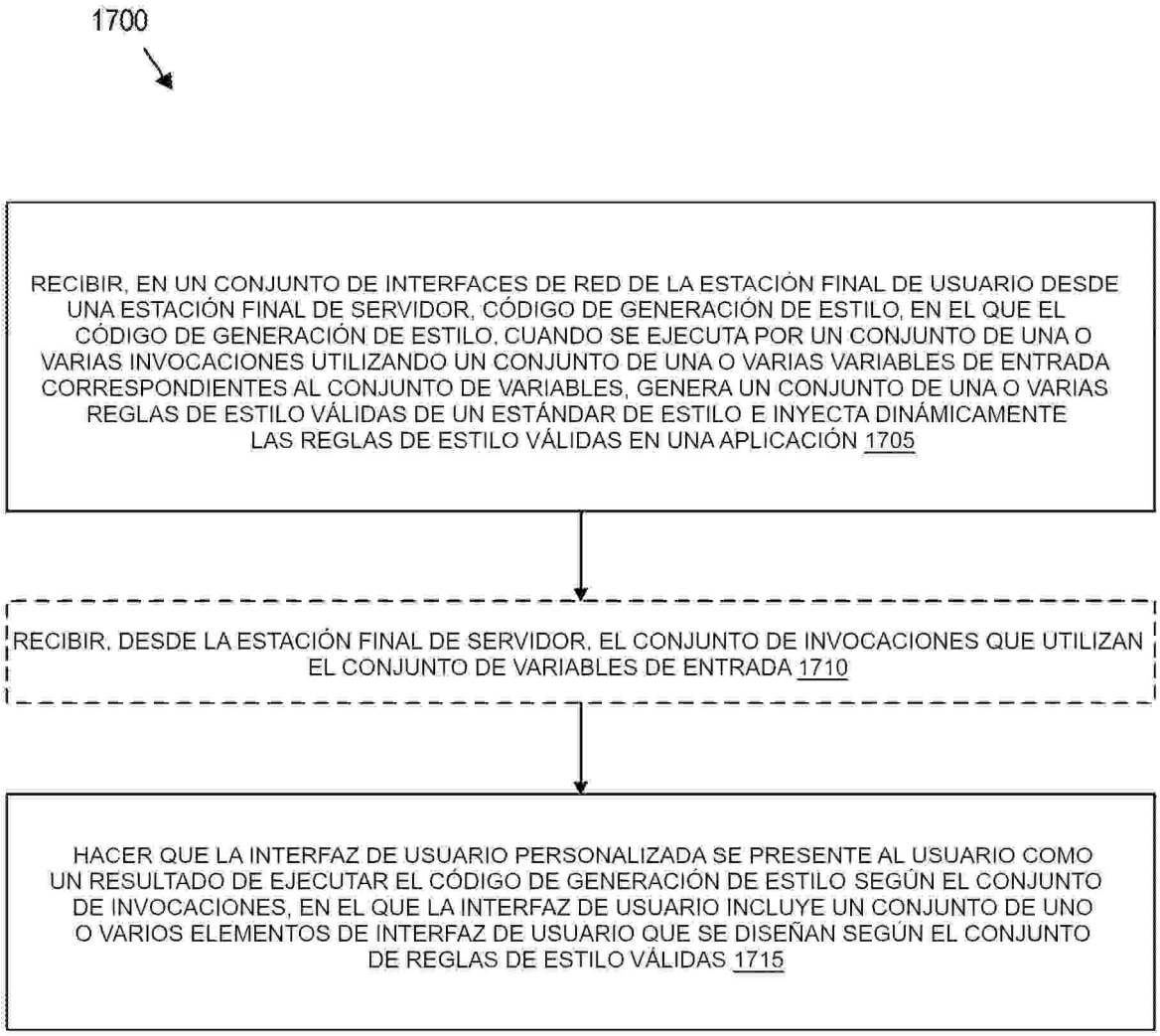

1700
RECIBIR, EN UN CONJUNTO DE INTERFACES DE RED DE LA ESTACIÓN FINAL DE USUARIO DESDE UNA ESTACIÓN FINAL DE SERVIDOR, CÓDIGO DE GENERACIÓN DE ESTILO, EN EL QUE EL CÓDIGO DE GENERACIÓN DE ESTILO, CUANDO SE EJECUTA POR UN CONJUNTO DE UNA O VARIAS INVOCACIONES UTILIZANDO UN CONJUNTO DE UNA O VARIAS VARIABLES DE ENTRADA CORRESPONDIENTES AL CONJUNTO DE VARIABLES, GENERA UN CONJUNTO DE UNA O VARIAS REGLAS DE ESTILO VÁLIDAS DE UN ESTÁNDAR DE ESTILO E INYECTA DINÁMICAMENTE LAS REGLAS DE ESTILO VÁLIDAS EN UNA APLICACIÓN 1705
RECIBIR, DESDE LA ESTACIÓN FINAL DE SERVIDOR, EL CONJUNTO DE INVOCACIONES QUE UTILIZAN EL CONJUNTO DE VARIABLES DE ENTRADA 1710
HACER QUE LA INTERFAZ DE USUARIO PERSONALIZADA SE PRESENTE AL USUARIO COMO UN RESULTADO DE EJECUTAR EL CÓDIGO DE GENERACIÓN DE ESTILO SEGÚN EL CONJUNTO DE INVOCACIONES, EN EL QUE LA INTERFAZ DE USUARIO INCLUYE UN CONJUNTO DE UNO O VARIOS ELEMENTOS DE INTERFAZ DE USUARIO QUE SE DISEÑAN SEGÚN EL CONJUNTO DE REGLAS DE ESTILO VÁLIDAS 1715

FIG. 17

REFERENCIAS CITADAS EN LA DESCRIPCIÓN

Esta lista de referencias citada por el solicitante es únicamente para mayor comodidad del lector. No forman parte del documento de la Patente Europea. Incluso teniendo en cuenta que la compilación de las referencias se ha efectuado con gran cuidado, los errores u omisiones no pueden descartarse; la EPO se exime de toda responsabilidad al respecto.

Documentos de patentes citados en la descripción

• US 2012167041 A1