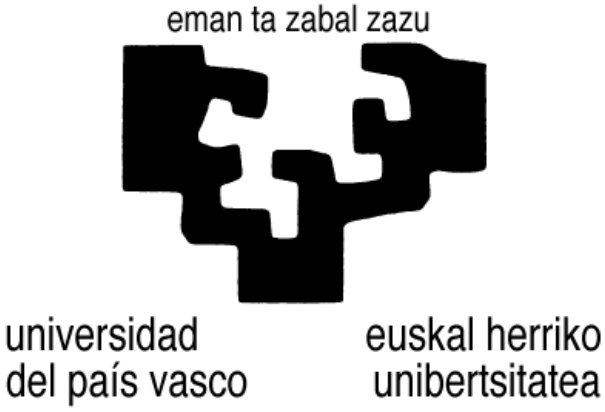


Facultad de Informática / Informatika Fakultatea

Centro de gestión de notificaciones Push para dispositivos móviles basados en IOS y Android

Alumno/a: D. Txomin Vila Ruiz

Director/a: D. Jesús Bermudez De Andrés

Proyecto Fin de Carrera, junio 2015

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En esta memoria se presenta el proyecto de fin de carrera "Centro de gestión de notificaciones Push para dispositivos móviles basados en IOS y Android", cuyo objetivo es crear una herramienta web, de libre distribución, que permita gestionar, programar y realizar envíos de notificaciones Push para aplicaciones móviles que funcionen bajo entornos IOS (Apple) y Android.

Las notificaciones Push es un sistema que permite, mediante mensajes cortos, la comunicación entre el desarrollador de una aplicación para dispositivos móviles y los usuarios que tengan dicha aplicación instalada en su dispositivo.
Es un sistema en el cual es el emisor quien inicia la comunicación, evitando así la consulta periódica al servidor por parte del cliente de los sistemas tradicionales basados en cliente servidor.

La notificación del mensaje en el dispositivo móvil es instantánea y no requiere que la aplicación destinataria esté ejecutándose ni tampoco de ningún servicio adicional, es el propio sistema operativo del dispositivo móvil quien provee de los mecanismos necesarios para que la comunicación pueda realizarse. Los únicos requisitos para la comunicación son: que el dispositivo este encendido, que esté conectado a internet y no tengan bloqueadas, desde la configuración del sistema, la entrada de notificaciones Push.

El centro de gestión de notificaciones Push es un back end web, que permitirá a sus usuarios, gestionar el envío de notificaciones Push a los dispositivos que tengan su aplicación instalada. Se podrán realizar envíos inmediatos de notificaciones, programar envíos para que se realicen en un momento determinado y obtener un feedback estadístico del impacto que la notificación ha tenido. Además mediante la inclusión de un servicio web será posible interactuar con la herramienta desde aplicaciones de terceros.

La herramienta se desarrollará cumpliendo los últimos estándares web HTML5 y CSS3, y siguiendo el patrón de diseño web adaptable (Responsive Web Design), para que la herramienta se adapte al entorno del usuario, ya sea un ordenador de sobremesa, una Tablet o un Smartphone.

Como infraestructura se utilizara la tecnología open source LAMP, Linux + Apache + MySQL + PHP, y a su vez el desarrollo se distribuirá en modo open source. Podrá ser libremente descargado, utilizado y modificado por parte de cualquier usuario.

Cabe destacar que la herramienta está preparada para trabajar tanto en un modelo clásico y un modelo SAAS. El modelo clásico corresponde a una instalación que será utilizada por un único usuario. El modo SAAS, software como servicio, corresponde a

una instalación en un servidor accesible desde internet (la nube), en el cual múltiples usuarios podrán hacer uso la misma instancia para hacer envíos de notificaciones a sus dispositivos.

A lo largo de esta memoria se presenta la planificación y el estudio de la viabilidad del proyecto como son el Análisis de Requisitos, el Análisis de Factibilidad y el Documento de Objetivos del Proyecto. Al igual se presenta el modo de llevarlo a cabo en Diseño, Implementación y Resultados y correcciones.

2. ANÁLISIS DE ANTECEDENTES

En este apartado se repasan las herramientas y tecnologías que se estima que serán necesarias para la correcta realización del proyecto. En primer lugar se describen los diferentes sistemas de notificaciones a dispositivos móviles centrándonos en el funcionamiento de los servidores APNS de Apple y GCM de Google, que son los servidores encargados de intermediar en el envío de notificaciones Push en las plataformas IOS y Android respectivamente. Seguidamente introducimos los servicios web y las arquitecturas disponibles para implementarlos SOAP (Simple Object Access Protocol) y REST (Representational State Transfer).

También comentaremos el estándar HTML5, CSS3, jQuery y el diseño responsivo, utilizados en la construcción de la herramienta web. Para concluir explicaremos el software de terceros que será necesario emplear y la infraestructura LAMP, que usa las siguientes herramientas: Linux como sistema operativo, Apache como servidor web, MySQL como sistema de gestión de bases de datos y PHP como lenguaje de programación.

2.1. Notificaciones a dispositivos móviles

Las notificaciones a dispositivos móviles son un sistema de alertas que los dispositivos son capaces de emitir para advertir a los usuarios de los diferentes eventos o acciones que se están produciendo. Pueden ser utilizadas para numerosos fines siendo los más comunes los de: alertar de la recepción de un nuevo email, envío de eventos de la agenda, noticias acerca de futuros lanzamientos de productos, descuentos, promociones, etc.

Con el crecimiento del uso de dispositivos móviles de los últimos años, las notificaciones ofrecen un canal de comunicación directo con el usuario final muy útil en sistemas de marketing y de fidelización de usuarios.

Para poder realizar notificaciones a dispositivos móviles existen varias alternativas: los clásicos mensajes de texto SMS/MMS y notificaciones mediante el envío de emails, y el sistema de notificaciones PUSH que ofrecen los Smartphone y Tablet.

Los **SMS** o servicio de mensajes cortos (Short Message Service) son un servicio que permite el envío de mensajes de texto entre teléfonos móviles. Forma parte del servicio estándar de telefonía móvil digital siendo el número de teléfono el identificador del dispositivo. La principal ventaja de los SMS es que cualquier dispositivo móvil tiene la capacidad de enviarlos y recibirlos. Como desventajas hay que indicar que se trata de un servicio de pago, en el cual el envío de cada mensaje tiene un coste asociado y además requiere que el usuario del dispositivo receptor del mensaje previamente haya comunicado al emisor el número de teléfono de su dispositivo.

Los **MMS** son una actualización de los SMS en la que además de mensajes de texto se puede enviar contenido multimedia como imágenes, videos y audio.

Las notificaciones vía **email,** ofrecen otra alternativa para el envío de notificaciones entre dispositivos móviles. Se basa en el envío de un email desde el emisor al receptor en cuyo contenido este el mensaje que se quiere notificar. Es un sistema gratuito y la mayoría de dispositivos móviles disponen de gestores de correo. Como contrapartida no todos los usuarios tienen configurado el correo en sus dispositivos móviles y eso provoca que no sea un sistema de notificaciones inmediato, siendo posible que el receptor nunca llegue a ver la notificación.

Las **notificaciones Push** o tecnología Push son un sistema implementado en los denominados Smartphones y Tablets que permiten la comunicación directa mediante mensajes de texto, entre el desarrollador de una aplicación para dichos dispositivos y los usuarios que tengan la aplicación instalada en su dispositivo. Esta comunicación se realiza mediante Internet.

Las notificaciones Push son un sistema en el que el emisor o servidor inicia la comunicación, ahorrando los recursos utilizados en los sistemas tradicionales en donde es el cliente quien consulta periódicamente al servidor. La recepción de este tipo de notificaciones es instantánea y no requiere que la aplicación destino a la que está dirigida la notificación, esté abierta o ejecutando servicios en segundo plano. Es el propio software del sistema operativo del dispositivo móvil quien despierta a la aplicación destinataria cuando se recibe la comunicación.

Las ventajas de este tipo de notificaciones son las siguientes:

- Son gratuitas.
- Son instantáneas.
- Crean una comunicación directa entre el desarrollador de una aplicación móvil y los usuarios que se han instalado la aplicación sin la necesidad de conocer el número de teléfono del móvil para iniciar la comunicación como en el sistema de los SMS.
- Permiten la interacción directa con la aplicación destinataria de la notificación, pudiendo usarse también para actualizar contenidos de la aplicación.
- Selectividad de usuarios, las notificaciones llegan únicamente a los usuarios que tienen la aplicación instalada y en términos de marketing el seguimiento de los mensajes es mayor.
- Permiten la fidelización de los usuarios de la aplicación móvil dado que permite una interacción más personal con los usuarios.
- Todos los sistemas operativos de dispositivos móviles Smartphone y Tablet proveen de sistemas para la realización de notificaciones PUSH.

Como desventajas cabe destacar que no todos los dispositivos móviles pueden instalar aplicaciones (apps) siendo instalables únicamente en los llamados Smartphones y Tablets. También es necesario que el dispositivo esté conectado a Internet, aunque es habitual que esto sea así en los Smartphone.

Nuestra elección para realizar el gestor de notificaciones móviles es el sistema de notificaciones PUSH. Dado que el objetivo del proyecto es el de realizar un gestor que permita a los desarrolladores de aplicaciones para dispositivos móviles basados en sistemas IOS y Android, realizar notificaciones a los usuarios de sus aplicaciones, las notificaciones PUSH son el único sistema que nos permiten está comunicación directa entre el desarrollador y sus usuarios. Además con la creciente demanda por parte de los usuarios de los dispositivos Smartphone y Tablet podemos decir que son un sistema de futuro y con amplias opciones de crecimiento.

2.1.1. Notificaciones Push en sistemas basados en IOS

Los dispositivos basados en el sistema operativo IOS, son aquellos dispositivos móviles comercializados por la compañía Apple y entre los cuales se encuentran los iPhone, iPad y los iPod Touch, que son un producto similar a los iPhone pero sin la capacidad de realizar llamadas telefónicas.

Para poder enviar notificaciones Push a estos dispositivos Apple facilita el sistema APNS, Apple Push Notification Services, el cual es un servicio que actúa de intermediario en el proceso de comunicación y envió de notificaciones Push. Ofrecen un servicio robusto y eficiente estableciendo conexiones seguras, encriptadas y persistentes. El servicio provee de los mecanismos necesarios para el envío de notificaciones Push, encargándose del transporte y enrutamiento de la notificación al dispositivo destinatario.

En los sistemas basados en IOS una notificación Push es, un mensaje corto que se compone de dos bloques principales de los datos: el token del dispositivo y la carga útil de envío. El token dispositivo es análogo a un número de teléfono, contiene información que permite al servicio APNS localizar el dispositivo en el que está instalada la aplicación cliente y también se utiliza para autenticar la notificación. La carga útil es un listado de propiedades, definidas en una estructura JSON, que contiene la información que será notificada al usuario de la aplicación.

A la hora de recibir la comunicación en el dispositivo destinatario, las notificaciones Push pueden adoptar dos estilos diferentes: tiras y alertas. Las tiras son un tipo de notificación no intrusivo que aparece en la parte superior de la pantalla, que en el caso de pulsar encima de ellas se abrirá la aplicación destinataria y que desaparecerá, a los segundos, si el usuario no interactúa con ellas. Las alertas, son mensajes modales que requieren de interacción por parte del usuario. Presentan dos botones, el primero con la

opción de “ver” la cual nos abrirá la aplicación destino de la notificación, y el segundo con la de cancelar para omitir la notificación.

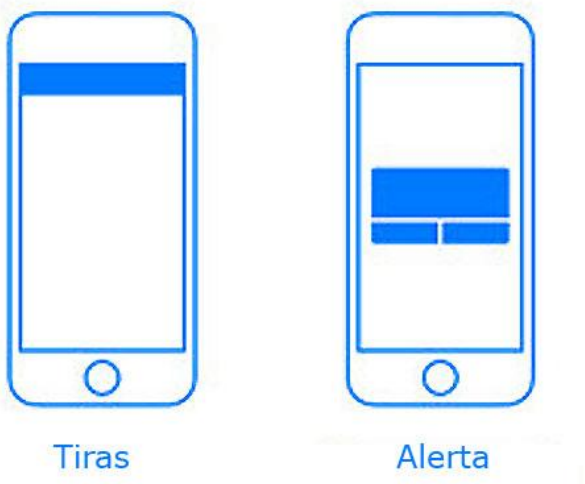


Figura 1. Estilos de las notificaciones Push en dispositivos IOS

Así mismo, el sistema operativo IOS provee al usuario del dispositivo de un centro de notificaciones en el cual se puede acceder al listado de las últimas notificaciones recibidas.

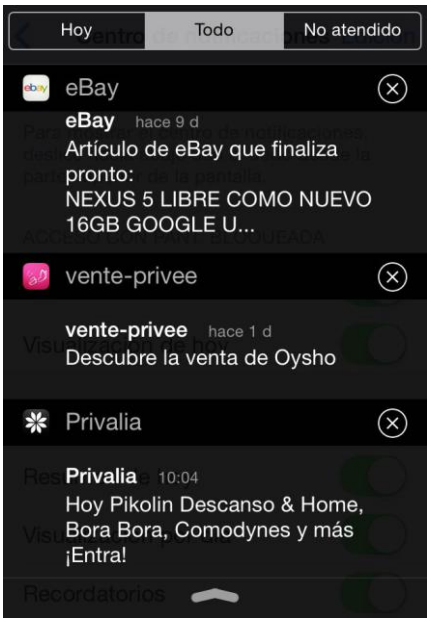


Figura 2. Centro de notificaciones Push en dispositivos IOS

2.1.1.1. Token del dispositivo

Para poder recibir notificaciones Push es necesario obtener el identificador obtener el identificador, denominado token, de los dispositivos móviles destinatarios de las notificaciones. Para ello los dispositivos deben registrarse en el servicio APNS. Este proceso se inicia automáticamente y de modo transparente al usuario cuando una aplicación móvil, con la capacidad de recibir notificaciones Push, se instala en el dispositivo.

El proceso de generación y comunicación del token es el siguiente: el sistema operativo del dispositivo recibe la solicitud de registro de la aplicación, se conecta al servicio APNS y envía la solicitud. El servicio APNS genera un token único que identifica inequívocamente al dispositivo y a la instancia de la aplicación instalada, esto es, cada instalación de una aplicación en un dispositivo genera un token diferente evitando así problemas de seguridad y envíos de notificaciones fraudulentos de terceros. Una vez generado el token, este se transmite al dispositivo que a su vez lo transmite al desarrollador de la aplicación o proveedor, que deberá guardar el token para poder utilizarlo como destinatario de las notificaciones.
La siguiente figura ilustra el proceso de trasmisión del token del dispositivo:

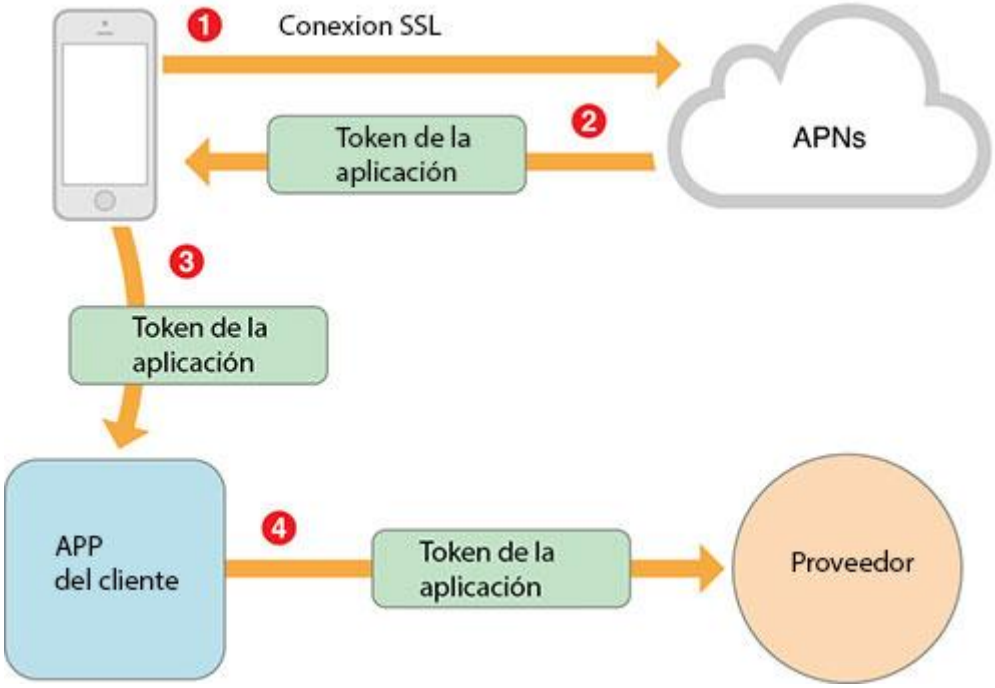


Figura 3. Proceso de transmisión del token de un dispositivo móvil basado en IOS

2.1.1.2. Carga útil de la notificación Push

Cada notificación Push incluye una carga útil, que contiene la información acerca de cómo el sistema debe alertar al usuario y el mensaje a notificar. El tamaño máximo

permitido para una carga útil es de 256 bytes. En caso de superarse este límite, el servicio APNS denegara la entrega de la notificación.

Para componer un objeto de carga útil es necesario utilizar una estructura JSON (según la definición de RFC 4627). Esta estructura debe contener una raíz principal denominada "aps", cuya estructura básica contendrá las diferentes propiedades que especifican las diferentes acciones:

- Un mensaje de texto plano (atributo alert) que se mostrará al usuario.
- Una insignia numérica (atributo badge), que se mostrara en el icono de la aplicación.
- Un sonido (atributo sound) a reproducir cuando se entrega la notificación.
- Un aviso opcional (atributo content-available) que sirve para indicar a la aplicación si existe nuevo contenido disponible a descargar o sincronizar.

El atributo "alert", además de texto plano, puede descomponerse en otra estructura JSON más compleja, con la cual se permite hacer referencia a cadenas de texto predefinidas en la aplicación destinataria y que sustituirán al texto mostrado en la notificación. Esta característica es útil cuando la aplicación destinataria está localizada en varios idiomas dado que permite que una notificación pueda traducirse al idioma que el usuario tenga configurada la aplicación.

Las posibles duplas atributo – valor de la estructura JSON son las siguientes:

Atributo	Tipo de valor	Definición
alert	String o estructura JSON	El sistema operativo muestra en la pantalla del dispositivo una alerta con el valor dado al atributo. Si se utiliza una estructura JSON para definir el valor de este atributo consultar las descripciones de la tabla 2.
badge	Integer	El número que se muestra como insignia en el icono de la aplicación en el escritorio del sistema operativo del dispositivo móvil.
sound	String	El nombre del archivo de sonido que se reproduce cuando la notificación es entrega en el dispositivo. En caso de ir vacío o con el valor "default", se reproducirá el sonido por defecto del sistema operativo.
content-available	Integer	Si el valor de este atributo es 1 indica que la aplicación a la que va dirigida la notificación dispone de nuevo contenido para descargar o sincronizar. Se utilizan para permitir a las aplicaciones la descarga de contenido en segundo plano.

Tabla 1. Atributos y valores de la estructura "aps"

Atributo	Tipo de valor	Definición
body	String	Es el texto del mensaje que se muestra.
action-loc-key	String o nulo	Si se especifica un valor, hace referencia a una cadena del archivo de recursos de localización de la aplicación, y se utilizara como título del botón de acción de la notificación
loc-key	String	Si se especifica un valor, hace referencia a una cadena del archivo de recursos de localización de la aplicación, esta sustituye al atributo body y se muestra como texto de mensaje. Además en la cadena puede ir formateada utilizando los patrones %@ o %n$@ para poder así, utilizar contenido variable especificado en el atributo "loc-args".
loc-args	Array de strings	Cadenas de texto que sustituyen a los patrones especificados en el atributo "loc-key"
launch-image	String	Nombre del archivo de alguna de las imágenes de la aplicación destino. Cuando se abra la aplicación a través del botón de acción de la notificación, se mostrara la imagen especificada como imagen de carga de la aplicación.

Tabla 2. Atributos y valores de la estructura "alert"

```
{
    "aps" : {
        "alert" : "Nueva actualización disponible.",
        "badge" : 9,
        "sound" : "default"
    }
}
```

Figura 4. Estructura JSON con el contenido de una notificación Push simple

```
{
    "aps" : {
        "alert" : {
            "loc-key" : "NEW_MESSAGE",
            "loc-args" : [ "Weezer", "Rock Star"]
        },
        "sound" : "default"
    }
}
Valor de la cadena NEW_MESSAGE = "Esta noche actuación de %@ en la sala %@"
```

Figura 5. Estructura JSON con el contenido de una notificación Push utilizando los atributos loc-key y loc-args

2.1.1.3. Envío de notificaciones Push

En el envío de notificaciones Push en sistemas IOS, los datos fluyen en una dirección. El proveedor, creador de la aplicación, compone la notificación incluyendo: el token del dispositivo y la carga útil. Una vez compuesta, se autentica en el servidor APNS y realiza el envío. Una vez la petición ha sido aceptada, el servidor APNS que se encarga de retrasmitirla al dispositivo destino.

Para autenticarse en el servicio APNS y poder realizar envíos de notificaciones Push, se establece un canal seguro mediante una conexión TLS (Transport Layer Security). Para abrir este canal seguro el proveedor tiene que autenticarse mediante su certificado digital, obtenible desde su cuenta desarrollador de aplicaciones para dispositivos móviles de Apple. Si un mismo proveedor desarrolla diferentes aplicaciones, dispondrá de un certificado por cada una de las aplicaciones.

El proceso de autenticación se ilustra en la siguiente figura:

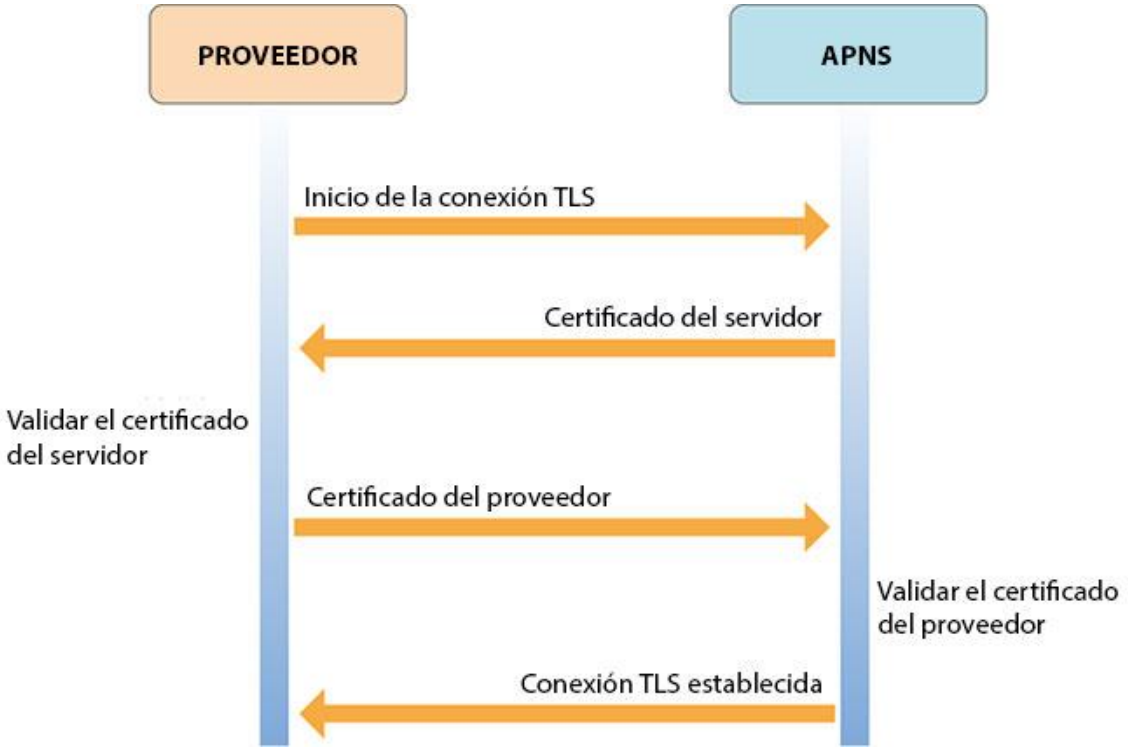


Figura 6. Apertura de canal seguro TLS para el envío de notificaciones Push

Una vez el canal está abierto, el servicio APNS atenderá las peticiones de envió de notificaciones Push y se encargará de entregarlas a su destinatario.

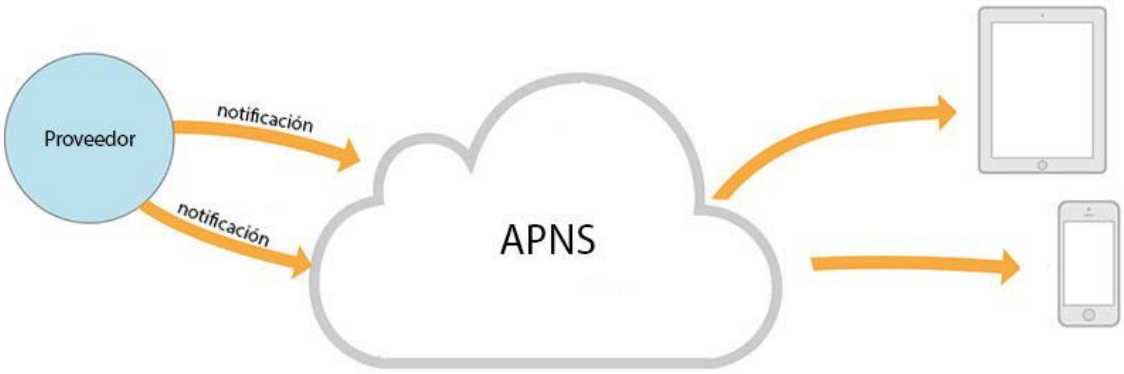


Figura 7. Flujo de datos en el envío de una notificación Push

El servicio APNS, incluye por defecto un componente de calidad de servicio (QoS). Este componente se encarga de realizar automáticamente las funciones de guardado y redistribución de las notificaciones. Si el servicio APNS encuentra dificultades a la hora de enviar una notificación, bien porque el dispositivo destino no está disponible o debido a errores puntuales en el servicio, la notificación se almacena en los servidores de Apple y se entrega al destinatario cuando este vuelva a estar disponible. Si hay varias notificaciones pendientes de entrega mientras el dispositivo no está disponible, cada una de las nuevas notificaciones anulara la anterior, solo enviando al destinatario la última

de ellas. Así mismo si un dispositivo permanece mucho tiempo en estado no disponible las notificaciones pendientes se descartan.

2.1.2. Notificaciones PUSH en sistemas basados en Android

Los dispositivos basados en el sistema operativo Android, son aquellos dispositivos móviles comercializados por diferentes compañías que utilizan el sistema operativo Android, provisto de forma gratuita por la compañía Google. Es un sistema operativo multiplataforma basado en Linux y actualmente es el sistema con mayor cuota de dispositivos en el mercado de Smartphones.

Para poder enviar notificaciones Push a dispositivos basados en Android, Google provee del servicio GCM, Google Cloud Messaging. Es el servicio que permite enviar y recibir información desde el proveedor de una aplicación móvil a un dispositivo basado en Android que tenga instalada dicha aplicación. El servicio provee de todos los mecanismos necesarios para encolar, enviar y recibir las notificaciones Push.

Como característica diferenciadora del sistema de envió de notificaciones en sistemas basados en IOS, en Android una vez la notificación Push es recibida por el dispositivo, es la propia aplicación destinataria la que se encarga de procesarla en lugar del sistema operativo. Esta característica tiene sus ventajas e inconvenientes. La ventaja principal es que al ser la propia aplicación la que gestiona las notificaciones, el desarrollador de la aplicación tiene libertad para decidir el tratamiento o respuesta a dar a cada notificación. El inconveniente es que este proceso, automático en sistemas IOS, es la necesidad de realizar la implementación que se encargue de dar respuesta a las notificaciones.
Dependiendo el tipo de respuesta que se dé a la notificación se clasifican dentro de estos tres tipos:

Envíos de mensajes de texto
Es el tipo de respuesta más habitual en el uso de las notificaciones Push. Consiste en el envío de un mensaje que se mostrara en el dispositivo destinatario. Este mensaje se compone de tres bloques principales de datos: el token del dispositivo del destinatario, las opciones de envío y su carga útil.
El token es el identificador único del dispositivo destinatario que contiene la información necesaria para que el servicio GCM pueda procesar el envío y entrega de la notificación. Las opciones de envío representan las opciones configurables de la comunicación y la carga útil es el listado de propiedades que contienen la información que se va a mostrar al destinatario. En todos los bloques la información se representa usando el formato JSON.

Al igual que en los sistemas basados en IOS, a hora de notificarse el mensaje en el dispositivo, estos pueden adoptar dos estilos diferentes: tiras y alertas. Las tiras son un tipo de notificación no intrusivo que aparece en la parte superior de la pantalla y

desaparecen a los segundos si el usuario no interactúa con ellas, y las alertas son mensajes modales que presentan dos botones, el primero con la opción de "acción" la cual nos abrirá la aplicación destino de la notificación, y el segundo con la opción de cancelar para omitir la notificación. Este tipo de notificación requiere interacción por parte del usuario.

Figura 8. Estilos de las notificaciones Push en dispositivos Android

Así mismo, el sistema operativo Android provee al usuario del dispositivo de un centro de notificaciones en el cual se puede acceder al listado de las últimas notificaciones recibidas.

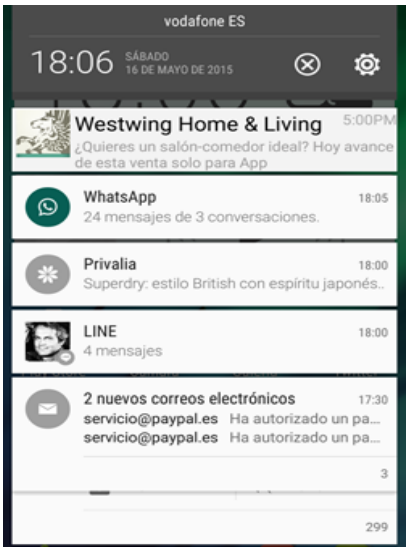


Figura 9. Centro de notificaciones Push en dispositivos Android

Envíos de mensajes mudos:

El objetivo de estas notificaciones no es la de enviar un mensaje al usuario, si no la de enviar el mensaje a la propia aplicación. El proceso de envió de estas notificaciones es

idéntico al anterior y se utilizan para: realizar modificaciones en la configuración interna en la aplicación, actualizaciones de contenidos, bloquear/desbloquear el acceso a ciertas secciones, etc…

Para el usuario de la aplicación esta comunicación es trasparente, dado que no será consciente de que esta se ha producido.

Envíos de mensajes para sincronizar datos:
Al igual que los mensajes mudos, este tipo de notificaciones también son trasparentes para el usuario. Se utilizan exclusivamente para indicar a la aplicación la necesidad de sincronizarse con el servidor para obtener o actualizar datos. Este tipo de comunicación es más ligera que las anteriores, dado que no se envía el bloque de carga útil, solo los bloques de identificación del dispositivo destinatario y el de opciones. Por ejemplo en una aplicación de localización vía GPS se pueden utilizar este tipo de notificaciones para avisar a la aplicación de que tiene que conectarse al servidor para obtener las coordenadas de las nuevas localizaciones.

2.1.2.1. Token del dispositivo

Para poder recibir notificaciones Push, los dispositivos Android deben registrarse en el servicio GCM. Este proceso se realiza automáticamente una vez la instalación de la aplicación en un dispositivo a finalizado. El proceso de generación del token del dispositivo es el siguiente:

- Una vez instalada la aplicación, el sistema operativo Android recibe la solicitud de registro del dispositivo en el servicio GCM, enviando al servicio el "Sender ID" y el "Application ID". El "Sender ID" es el identificador del desarrollador de la aplicación y el "Application ID" es el identificador de la propia aplicación. Estos datos van implícitos en la propia instalación de la aplicación.

- Si el proceso de registro se ha realizado correctamente el servicio GCM responderá con el token del dispositivo, también denominado "Registration ID", a la aplicación móvil. Este token identifica inequívocamente a la aplicación y al dispositivo en el que se ha realizado la instalación.

- Desde la aplicación Android se reenvía el token del dispositivo hacia el servidor del proveedor de la aplicación, que guardará el identificador para poder usarlo en el envío de notificaciones Push.

La siguiente figura ilustra el proceso de creación y trasmisión del token del dispositivo:

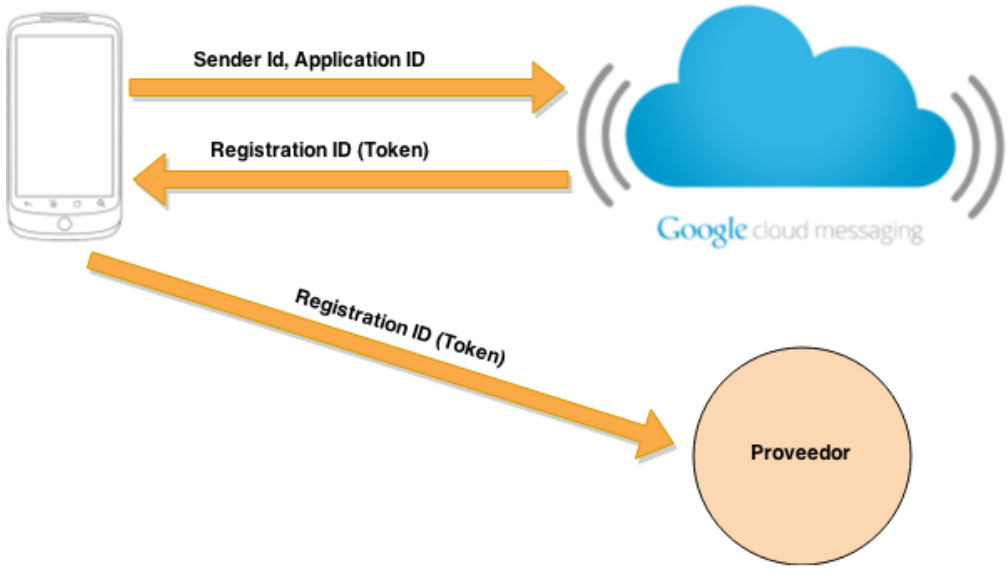


Figura 10. Proceso de transmisión del token de un dispositivo móvil basado en Android

2.1.2.2. Opciones de envío

Las opciones del envió en las notificaciones Push a sistemas Android especifican los atributos del mensaje que se va a trasmitir. En la siguiente tabla se ilustran todas las opciones disponibles:

Atributo	Tipo de valor	Definición
message_id	String	Parámetro obligatorio que actúa como identificador único del mensaje enviado.
collapse_key	String o nulo	Si se especifica un valor, identifica a un grupo de notificaciones que pueden ser descartadas si todavía no han sido entregados al dispositivo. Se utiliza para evitar enviar demasiadas notificaciones similares de una misma aplicación cuando el dispositivo vuelva a estar disponible. Solo se permite el uso de cuatro collapse_key diferentes a la vez, si se excede esta cantidad, el servicio GCM descartara una de ellas sin la opción de poder especificar cuál de ellas se descarta.
priority	String	Especifica la prioridad del mensaje en un rango de valores de 0 a 10, siendo 0 la máxima prioridad y 10 la menor.
content_avalilable	Boolean	Este parametro sirve para indicar a la aplicación que hay nuevo contenido descargable que tiene

		que sincronizar.
delay_while_idle	Boolean	Si el parámetro está asignado a true, indica que la notificación no debe procesarse en el servicio GCM hasta que el dispositivo destinatario esté disponible.
time_to_live	Integer	Indica el tiempo máximo, en segundos, de retrasmisión de la notificación en el caso que el dispositivo destino no esté disponible. El tiempo máximo configurable es de 4 semanas, correspondiendo también con el valor por defecto.
delivery_receipt_ requested	Boolean	Este parámetro permite al desarrollador de la aplicación solicitar el acuse de recibo del envío de la notificación. Este acuse de recibo se envía cuando el dispositivo destinatario confirma que ha recibido la notificación. Por defecto esta opción esta desactivada.
restricted_package_ name	String	Mediante este parámetro se especifica el nombre de del paquete de instalación de la aplicación. Si se indica la notificación solo llegará a aquellos destinatarios cuya versión de la aplicación coincida con la establecida.
dry_run	Boolean	Si este parámetro se establece a true, permite al emisor de la notificación testear el envío sin que este se realice.

Tabla 3. Opciones de envío en las notificaciones Push a sistemas Android

2.1.2.3. Carga útil de la notificación Push

La carga útil de una notificación Push en un sistema Android es una estructura JSON, cuya raíz es la palabra reservada “data”, en la que se especifica un conjunto de relaciones atributo valor con las cuales se compondrá el mensaje a notificar. Los atributos son preestablecidos libremente por el desarrollador de la aplicación con la restricción de no poder utilizar como nombre del atributo: ninguna palabra reservada del servicio GCM (“from” o cualquier palabra que comience por “google” o “gcm”), ni tampoco el nombre de ningún atributo de las opciones de envío. Por su parte los valores deberán ser obligatoriamente cadenas de texto, el sistema no reconoce otro tipo de datos como enteros o booleanos.

Una vez enviada la notificación, la aplicación destinataria se encargara de tratar la carga útil recibida y mostrar el mensaje al usuario del dispositivo móvil.

```
{
    "data":
    {
        "mensaje": "El partido a finalizado con el resultado",
        "resultado": "3-2"
    }
}
```

Figura 11. Ejemplo de carga útil de una notificación Push para sistemas basados en Android

A fecha de Junio de 2015 Google ha actualizado el servicio de notificaciones ampliando la funcionalidad y añadiendo nuevos atributos configurables en la carga útil de las notificaciones. Mediante estas nuevas modificaciones se da la opción a que el procesamiento de la carga útil de la notificación pueda recaer sobre el sistema operativo y no sobre el desarrollador de la aplicación, que era la única opción hasta el momento. Aunque no entra dentro del objetivo del proyecto la actualización a estas nuevas funcionalidades vamos a listar en la siguiente tabla las nuevas opciones para que se tengan en cuenta en futuras actualizaciones de la herramienta. Hay que tener en cuenta que estas características nuevas solo funcionan en las nuevas versiones del sistema operativo Android.

Se ha añadido un nuevo elemento raíz opcional denominado “notificación”, en el cual se pueden definir los siguientes atributos:

Atributo	Tipo de valor	Definición
title	String	Especifica el título de la notificación
body	String	Especifica el cuerpo o mensaje de la notificación
icon	String	Especifica el icono de la notificación. Debe ser el identificador de una imagen de la aplicación. Por defecto se muestra el icono de la aplicación.
sound	String	Especifica el sonido a reproducir cuando se recibe la notificación.
badge	String	El número que se muestra como insignia en el icono de la aplicación en el escritorio del sistema operativo del dispositivo móvil.
tag	String	Si se especifica valor, agrupa a los mensajes que se notifican con el mismo tag, de tal modo que en el centro de notificación solo se muestra la última notificación recibida.
color	String	Especifica en formato #rrggbb, el color del icono de la notificación.
click_action	String	Mediante este parámetro se puede especificar la acción o pantalla de la aplicación que se mostrara cuando el usuario clica en la notificación.

Tabla 4. Nuevos atributos de carga útil en las notificaciones Push a sistemas Android

2.1.2.4. Envío de notificaciones Push

En el envío de notificaciones Push en sistemas basados en Android, los datos fluyen en una dirección, desde el servidor del proveedor hacia los dispositivos con la aplicación instalada. El proveedor de la aplicación compone la notificación incluyendo: los datos de autenticación en el servicio CGM, el token del dispositivo a notificar, las opciones de envío y opcionalmente la carga útil. Si la petición ha sido aceptada, el servicio GCM que se encarga de retrasmitirla al dispositivo destinatario.

El envío de una notificación se realiza mediante una solicitud POST al servicio CGM en la siguiente url: https://gcm-http.googleapis.com/gcm/send. La solicitud se compone de dos partes: el HTTP header o cabecera y el HTTP body o cuerpo.

La cabecera debe contener los siguientes parámetros:

- Authorization:key. Es la clave de registro de la aplicación en el servicio CGM obtenible desde la cuenta de desarrollador de aplicaciones Android. Esta clave de registro es única para cada aplicación por lo que si un desarrollador tiene varias aplicaciones, tendrá que crear una clave de registro para cada una de ellas.
- Content-Type: application/json. Sirve para indicar que el formato en el que se van a retrasmitir los datos es JSON.

```
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{
  "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
  "data" : {
    ...
  },
}
```

Figura 12. Ejemplo de solicitud POST de una notificación Push Android

El cuerpo, dependiendo del tipo de notificación a enviar, se compondrá de:

- Token del dispositivo del destinatario mediante el uso del atributo “to”
- Opciones de envío
- En el caso de no ser un mensaje de sincronización, con la carga útil.

En las siguientes se pueden ver ejemplo de notificaciones de sincronización y envío de mensajes con y sin opciones:

```
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA
{
  "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx..."
}
```

Figura 13. Ejemplo de notificación Android de sincronización

```
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{
  "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
  "data" : {
    "resultado": "5-1",
    "tiempo-de-juego": "22:19"
  },
}
```

Figura 14. Ejemplo de notificación Android sin opciones de envío

```
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{
  "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
  {
    "collapse_key": "tiempo-y-resultado",
    "time_to_live": 108,
    "delay_while_idle": true
  },
  "data" : {
    "resultado": "5-1",
    "tiempo-de-juego": "22:19"
  },
}
```

Figura 15. Ejemplo de notificación Android con opciones de envío

2.2. Servicios Web

Un servicio web o web services es una tecnología que permite la interoperabilidad máquina a máquina entre distintas aplicaciones. Se utilizan para que aplicaciones desarrolladas en diferentes lenguajes de programación y ejecutándose en distintas plataformas puedan intercambiar datos entre sí. El uso más habitual es en el que una aplicación, denominada publicadora del servicio, expone parte de su lógica de negocio o acceso a datos mediante un API, a otras aplicaciones consumidoras del servicio publicado.

Para que el proceso de comunicación sea factible, un servicio web utiliza diferentes protocolos web y a su vez debe ser accesible desde la web. Aunque no es necesario, se

aconseja que disponga de una interfaz pública que permita a los consumidores del servicio consultar las especificaciones del servicio.

Para implementar un servicio web se pueden utilizar diferentes modelos de arquitectura y protocolos. Entre estos modelos destacan dos: SOAP (Single Object Access Protocol) y REST (Representational State Transfer).

2.2.1. SOAP

SOAP o Single Object Access Protocol, es una arquitectura de construcción de servicios web que define un protocolo basado en XML como marco de envío de mensajes fuertemente tipados.

Cada operación que proporciona el servicio se define explícitamente en una estructura XML, tanto la solicitud como la respuesta de la operación. Así mismo cada parámetro de cada operación se define de igual modo especificando su tipo: entero, string o cadena de texto, estructuras de datos, etc...

Como resultado de este proceso de modelado del servicio generalmente se genera un fichero WSDL (Web Service Description Language) que actúa como descriptor del servicio web. El fichero WSDL es la interfaz que describe los métodos del servicio junto con sus parámetros de entrada y respuesta. También puede definirse como el contrato entre el proveedor del servicio y sus consumidores.

Aunque esta interfaz no es obligatoria para la comunicación con un servicio SOAP, el no hacer uso de ella dificulta sobremanera la interactuación con el servicio, dado que habría que realizar un programación ad hoc para la comunicación. En el caso que si se haga uso de la interfaz, existen numerosas herramientas que convierten estas interfaces WSDL en objetos o clases nativas de la plataforma desde la que se va a consumir el servicio evitando así la programación de un sistema de comunicación con el servicio.

La comunicación con los servicios SOAP se realiza mediante el intercambio de mensajes XML, llamados SOAP Envelope, que se componen de una cabecera y un cuerpo.

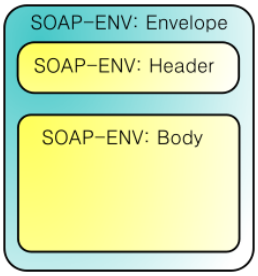


Figura 16. Estructura de un mensaje SOAP

Estos mensajes pueden ser enviados diferentes protocolos de transporte, aunque en la práctica se utiliza el protocolo HTTP por las ventajas que ofrecen al usarlos sobre cortafuegos (firewalls) en contraposición a otros protocolos como GIOP/IIOP o DCOM que suelen ser repelidos por estos.

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetails xmlns="http://warehouse.example.com/ws">
       <productId>827635</productId>
     </getProductDetails>
   </soap:Body>
</soap:Envelope>
```

Figura 17. Ejemplo de llamada a un servicio SOAP

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
       <getProductDetailsResult>
         <productName>Toptimate 3-Piece Set</productName>
         <productId>827635</productId>
         <description>3-Piece luggage set.  Black Polyester.</description>
         <price>96.50</price>
         <inStock>true</inStock>
       </getProductDetailsResult>
     </getProductDetailsResponse>
   </soap:Body>
 </soap:Envelope>
```

Figura 18. Ejemplo de respuesta de un servicio SOAP

Los servicios web SOAP se caracterizan por: su **interoperabilidad**, debido a estar basados en mensajes XML, estos se pueden invocar desde múltiples lenguajes de programación. **Escalabilidad**, al realizar la comunicación mediante el protocolo HTTP es fácilmente escalable además de raramente ser bloqueado por los cortafuegos. **Independencia,** se pueden implementar servicios web basados en SOAP desde cualquier lenguaje y plataforma. **Seguridad**, soportan el protocolo WS-Security desarrollado por IBM, Microsoft y VeriSign que contiene la especificación de cómo debe garantizarse la seguridad e integridad de los mensajes enviados mediantes servicios web.

2.2.2. REST

REST o Representational State Transfer es un modelo de construcción de servicios web basado en la manipulación de recursos a través del protocolo de transporte HTTP. A diferencia de los servicios SOAP donde se define un protocolo estandarizado para el uso de los mismos, REST hace uso de la interfaz web evitando así la creación de nuevas normas o estándares. Los datos a intercomunicar se pueden estructurar en cualquier formato existente, aunque por lo general se utiliza el formato JSON. Los métodos y parámetros de los servicios no están fuerte tipados y siguen el paradigma de la programación orientada a objetos.

Los servicios REST ofrecen una interfaz web simple que permite interactuar con el servicio web de manera fácil y eficaz. Tienen una gran escalabilidad debido a que es un protocolo cliente servidor sin estado, esto es, cada mensaje enviado por protocolo HTTP contiene la información necesaria para conformar la petición. Utiliza una sintaxis universal para identificar los recursos, cada recurso solo se puede obtener a través de su URI única. Las URI actúan como identificador del recurso y son la clave de la

arquitectura REST, dado que permiten el uso de URIs que pueden ser comprendidas tanto por humanos como maquinas.

```
/equipos
/equipos/real-sociedad
/equipos/eibar
/equipos/eibar/jugadores
```

Figura 19. Ejemplo de URIs REST

Al ser únicas, existe la posibilidad de cachear las respuestas de las URI para mejorar el rendimiento de estos servicios.

Para la comunicación con los servicios se emplean el subconjunto de operaciones definidas por el protocolo HTTP: GET, POST, PUT y DETELE que se asocian al tipo de operación que se va a ejecutar, comúnmente llamado operaciones CRUD (Create, Read, Update y Delete)

La operación GET (leer), operación por defecto en las peticiones web, se utiliza para realizar exclusivamente peticiones de consulta o de solo lectura. Aunque no es obligatorio, las respuestas a las operaciones GET siempre deben devolver la misma estructura de datos.

Utilizamos la operación POST (crear), para crear un nuevo subconjunto de un recurso o también para la actualización de varios subconjuntos de un recurso.

La operación PUT (actualizar) se utiliza para actualizar recursos específicos.

Utilizamos la operación DELETE (borrar) para eliminar recursos.

```
GET /equipos - Consulta los equipos
POST /equipos/real-sociedad - Crea el equipo Real Sociedad
PUT /equipos/real-sociedad/Anoeta – Actualiza el campo del equipo Real Sociedad
DELETE /equipos/real-sociedad - Borra el equipo Real Sociedad
```

Figura 20. Llamadas GET, POST, PUT y DELETE a un servicio REST

El tipo de URIs utilizadas en los servicios REST son virtuales, esto es, no existe una página física que corresponda con la dirección de cada una de ellas, si no que es una única página física, el servicio web, la que se encarga de responder a todas las peticiones entrantes. Para poder crear la estructura de las URIs será necesario el uso de mecanismos que permitan la virtualización y redirección de las URIs virtuales al servicio. La mayoría de servidores web proveen de estos mecanismos siendo el ".htaccess" de Apache y "rewrite_module" de IIS los más conocidos.

Como respuesta a las peticiones, los servicios REST utilizan comúnmente respuestas en formato JSON, devueltas en el cuerpo de la petición HTTP. Los consumidores del servicio interpretaran estas respuestas y realizaran las operaciones oportunas dentro de sus aplicaciones.

```
HTTP/1.1 200 OK
Date: Fri, 22 Feb 2015 22:32:22 GMT
Server: Apache
X-Powered-By: Phusion Passenger (mod_rails/mod_rack) 3.0.17
ETag: "b8a7ef8b4b282a70d1b64ea5e79072df"
X-Runtime: 13
Cache-Control: private, max-age=0, must-revalidate
Content-Length: 209
Status: 200
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Content-Type: js; charset=utf-8
{
   "equipos":
   {
     "Eibar",
     "Real Sociedad",
     "Real Madrid",
     "Barcelona",
     "Athletic de Bilbao"
   }
}
```

Figura 21. Respuesta de una llamada GET /equipos a un servicio REST

Los servicios REST se caracterizan por no definir protocolos, aprovechando la infraestructura web existente, su facilidad y extensión de uso, un simple navegador web es un cliente potencial de un servicio web REST. El desarrollo de estos servicios es sencillo, no se requiere del uso herramientas de terceros para su construcción, ya que pueden implementarse mediante archivos de texto interpretados. Tienen mayor escalabilidad y rendimiento que los servicios basados en SOAP y no necesitan de una compleja especificación para su uso. En cuanto a seguridad los servicios REST soportan el estándar de seguridad web SSL.

2.2.3. Conclusiones

Después de realizar el estudio entre los modelos SOAP y REST hemos tomado la decisión de utilizar los servicios web basados en el modelo REST para implementar nuestra aplicación. Las razones fundamentales por las cuales nos hemos decidido:

- Facilidad de uso y potenciales consumidores. Los servicios REST pueden ser usados fácilmente desde cualquier plataforma o dispositivo que pueda realizar peticiones HTTP, siendo posible el su consumo desde un simple navegador web, lo cual incluye a cualquier dispositivo móvil. Para usar servicios web basados en SOAP de un modo sencillo es necesario que la plataforma de desarrollo sea capaz de interpretar las interfaces WSDL, si no se disponen de estas herramientas su uso se complica sobremanera.

- Rendimiento. Al no ser necesario el uso de XML las llamadas y respuestas de los servicios REST tienen mejor rendimiento y consumen menos recursos. Además las peticiones de consulta se pueden cachear mientras que en SOAP no.
- No usa interfaces. En los servicios SOAP las modificaciones en la interfaz WSDL obliga a realizar modificaciones en los clientes que los consumen.

2.3. HTML5

Además del lenguaje en sí, se denomina HTML5 al conjunto de tecnologías que permiten la creación de sitios y aplicaciones web. Este conjunto de tecnologías de componen del lenguaje HTML5, que permite estructura y presentar el contenido en la web, CSS3 que es el lenguaje de estilos que permite describir la presentación de un documento escrito en HTML y la librería escrita en el lenguaje de scripting javascript, jQuery, mediante la cual se puede acceder y manipular en tiempo real a los elementos creados mediante HTML. Mediante estos nuevos estándares y tecnologías es posible crear aplicaciones siguiendo el patrón de diseño responsivo de modo que la aplicación creada se adapte al dispositivo que se esté usando para visualizarla.

2.3.1. El lenguaje HTML5

Hyper Text Markup Language 5 o HTML5 es la última revisión del lenguaje utilizado para la elaboración de páginas y aplicaciones web. Esta quinta versión paso a ser el estándar en octubre del 2014 sustituyendo al estándar anterior HTML4/XHTML y está regulada por el consorcio W3C, World Wide Web Consortium, que es la organización internacional e independiente que se encarga de regular los estándares y recomendaciones web. Se considera que es un estándar vivo dado que esta en continua evolución.

El HTML, es el lenguaje interpretado por los navegadores web para mostrar las páginas web. Es un lenguaje de marcas de hipertexto, es decir, un lenguaje que permite escribir texto de modo estructurado y semántico. Se utiliza para la construcción de la estructura del contenido de un documento web, incluyendo: textos, imágenes, audio, video, etc.
También es un lenguaje de referencias externas, es decir, los elementos externos que componen cada documento: páginas, imágenes, scripts, etc. no se incrustan en el mismo si no que se añaden mediante texto, referencias a la ubicación en el que se encuentran cada uno de los elementos externos. De este modo, un documento HTML solo se componte de texto plano y son los navegadores quienes se encargan de interpretar su contenido.

Para ello utiliza un conjunto de elementos, definidos mediante etiquetas, con los cuales se estructuran los diferentes tipos de contenidos. Los elementos se componen de dos propiedades básicas: atributos y contenido, y van encapsulados entre las etiquetas de

inicio <> y las etiquetas de cierre </>, que indican el final de elemento. También existen elementos sin contenido, denominados elementos vacíos, que no tienen etiqueta de cierre. Los atributos del elemento están contenidos en la etiqueta de inicio y el contenido se ubica entre la etiqueta de inicio y final.

```
<p class="foo">contenido del elemento</p>
```

Figura 22. Párrafo en HTML

Todos los elementos no vacíos pueden a su vez contener más elementos, formando así estructuras más complejas. Para crear estructuras complejas hay que cumplir la regla de anidamiento de elementos: *las etiquetas de cierre deben escribirse en orden inverso a las etiquetas de inicio*. Si no se cumple la regla el código resultante no cumplirá el estándar y quedara expuesto a que los navegadores no sean capaces de interpretarlo correctamente.

Como la mayoría de lenguajes de marcado, para la correcta construcción de documentos, el HTML se define a través de un DTD (Definición del tipo de documento). El DTD establece la estructura de etiquetas del documento, los atributos que pueden tener cada una de las etiquetas y los posibles valores que los atributos pueden tomar. Para declarar el tipo de documento que se utiliza en un documento HTML, se utiliza la instrucción "doctype". Esta instrucción debe estar situada al principio del documento y debe estar acompañado de la estructura minina básica de todo documento HTML, una etiqueta raíz "html", una etiqueta de cabecera "head", en donde se definen la información técnica para el navegador y una etiqueta de cuerpo "body" en la que se define el contenido del documento, esto es lo que se muestra en el navegador.

```
<!DOCTYPE html>
<html>
    <head>
    </head>

    <body>
    </body>
</html>
```

Figura 23. Estructura mínima de un documento HTML

Si un documento cumple con todas las reglas definidas en el DTD declarado, se denomina documento válido o documento estándar y teóricamente debería ser interpretado del mismo modo en todos los navegadores web existentes, aunque hasta la aparición de HTML 5, como veremos más adelante, no siempre ocurría así.

El conjunto de elementos HTML existentes se pueden clasificar bien por su modo de presentación o por su interpretación semántica. En la clasificación por modo de presentación existen dos tipos de elementos: los elementos de bloque y los elementos de línea. Los elementos de bloque son aquellos que empiezan en una nueva línea y ocupan el total del espacio disponible en la misma. Por contra los elementos de línea solo ocupan el espacio que necesitan para mostrar su contenido.

En la clasificación por interpretación semántica los elementos se clasifican en función del contenido que dichos elementos contienen. Se clasifican en elementos: estructurales, de presentación, de texto, formularios, tablas, listas, imágenes, videos, hipertexto, listas, meta-información, scripts y hojas de estilos.

Los elementos HTML más comunes, definidos por su etiqueta son:

Estructurales:

- <html>: Etiqueta que representa la raíz de un documento HTML. Indica a los navegadores que el documento a interpretar se trata de un documento HTML. Este elemento contiene al resto de elementos y siempre va precedido de la instrucción "doctype" que declara el DTD con el que está construido el documento.
- <head>: Etiqueta que representa la cabecera del documento HTML. Contiene información técnica del documento que por lo general no se muestra al usuario pero resulta muy útil para informar acerca del contenido del documento a los buscadores web como por ejemplo Google.
- <body>: Representa el contenido del documento que los navegadores muestran al usuario. Es la parte visible del documento HTML.
- <title>: Etiqueta que sirve para definir el título del contenido web. Generalmente los navegadores suelen mostrar este título en el marco superior de su interfaz.
- : Etiqueta neutro contenedor de otros elementos HTML. Comúnmente denominado capa sirve para estructurar los bloques de contenido del documento.

Presentación:

- : Se utiliza para dar importancia al texto incluido en este elemento. Los navegadores muestran el contenido de esta etiqueta en negrita.
- : Se emplea para enfatizar texto y los navegadores lo muestran mediante texto en cursiva.
- <i>: Utilizado para mostrar texto en cursiva. Actualmente, aunque todavía no es el uso estándar, se utiliza como etiqueta neutro para definir iconos. En un claro

ejemplo de que el HTML es un estándar viviente dado que se espera que en futuras revisiones se modifique su definición en el estándar web.

-
: Salto de línea. El siguiente elemento se presentara en una nueva línea.

Texto:

- <h1>-<h6>: Encabezados de las diferentes secciones del documento con diferentes prioridades, siendo h1 el más relevante y h6 el menos relevante. Se denomina el índice del documento donde el h1 es el título y el resto de encabezados h2, h3, etc. las secciones y subsecciones.
- <p>: Etiqueta de bloque que se utilizada para incluir párrafos de texto.
- : Etiqueta de línea que se utiliza para incluir frases textuales. Sustituye al antiguo elemento font.

Formulario:

- <form>: Etiqueta raíz que representa un formulario con el que el usuario puede interactuar. Debe definir los atributos “action” y “method”. El atributo “action” indica cual es la URI que va a recoger los datos enviados por el formulario y el atributo “method” indica el modo de envío: GET o POST.
- <label>: Define el texto asociado a un elemento input del formulario.
- <input>: Especifica un tipo de campo de entrada de datos con el cual el usuario puede interactuar. La presentación de estos elementos varía dependiendo el tipo de input declarado mediante el atributo “type”. Estos pueden ser cajas de texto, selectores de archivos locales para subir al servidor en el que se encuentra alojada la web, selectores, checkboxes, etc.
- <button>: Etiqueta que representa un elemento de tipo botón.

Tablas y listas:

- <table>: Define una tabla de datos dentro del documento. Hasta hace unos años este tipo de contenido se utilizaba erróneamente para dar formato a la presentación del documento.
- <tr>: Define una línea dentro de la tabla.
- <td>: Define una celda dentro de la línea. Tiene que ir contenida dentro de una etiqueta tr.
- : Sirve para definir una lista no ordenada, cada elemento de la lista va precedido mediante el símbolo de un círculo.
- : Sirve para definir listas ordenadas, cada elemento de la lista va precedido mediante una numeración.

- : Define cada elemento de una lista, tanto ordenada como no ordenada.

Contenido multimedia:

- <img>: Representa una imagen. Requiere del atributo “src”, en el cual se indica la ubicación en la que se encuentra la imagen física y del atributo “alt” en el cual se indica una descripción textual de la imagen que el navegador nuestra si encuentra dificultades a la hora de mostrar la imagen. Este último atributo es utilizado por navegadores accesibles puedan leer el contenido de la imagen.
- <video>: Permite referenciar un archivo de video en el documento HTML.

Hipertexto:

- <a>: Etiqueta ancla, comúnmente llamado enlace o hipervinculo, que hace referencia a otro documento HTML interno o externo al sitio web. Para referenciar a otro documento se utiliza el atributo “scr” el cual contienen la URI del contenido referenciado.

Scripts, hojas de estilos y meta-información:

- <script>: Se utiliza para referenciar scripts de cliente como por ejemplo JavaScript, que permiten la manipulación desde el navegador web de los elementos definidos en el documento.
- <style>: Sirve para definir reglas de estilos utilizadas para definir la presentación de los elementos del documento. Es muy común la separación de estas reglas en archivos independientes que se pueden referenciar mediante la etiqueta <link>
- <meta>: Definen meta-información acerca del documento, como su autor, palabras clave, descripción del contenido, etc.

Aunque la mayoría de las etiquetas son comunes entre revisiones anteriores a HTML5, este nuevo estándar supone un punto de inflexión en como los distintos navegadores interpretan el estándar web. Hasta la aparición de HTML5 era muy habitual tener que desarrollar varias versiones alternativas de parte del documento web para que este se interpretara de igual manera en los distintos navegadores. Desde que HTML5 fue establecido como estándar los diferentes desarrolladores de navegadores web han centrado sus esfuerzos en realizar navegadores que cumplan con este estándar, facilitando así la codificación de documentos HTML a los desarrolladores web.

Las principales características diferenciadoras de HTML5 sobre los estándares anteriores son:

- Incorporación de nuevas etiquetas, canvas, audio y video con sus respectivos codecs, para visualizar contenido multimedia. Hasta la llegada de HTML5 era necesario recurrir a tecnologías de terceros para mostrar ciertos contenidos multimedia, no siendo soportados en muchos navegadores, como por ejemplo los navegadores web de los dispositivos móviles.
- Formularios inteligentes. HTML5 incorpora mecanismos para definir los tipos de datos de los input (email, números, fechas) y permite la validación del contenido incluso mediante el uso de expresiones regulares.
- Incorporación de visores de contenido XML como MathML y SVG, utilizados para interpretar fórmulas matemáticas y gráficos vectoriales respectivamente.
- Nuevas etiquetas semánticas que permiten describir el significado del contenido. Aunque de cada al usuario no tienen un impacto visual son especialmente útiles a la hora de posicionar la web en los motores de búsqueda.
- Nuevas APIs que permiten realizar Drag & Drop, geolocalización, almacenamiento local persistente, WebSockets para realizar comunicación bidireccional, etc. Aunque todavía muchas de ellas no son relevantes dado que no todos los navegadores las soportan.

2.3.2. CSS3

CSS 3 (Cascading Style Sheets) u hojas de estilos en cascada es la tercera revisión del lenguaje usado para definir la presentación de los documentos HTML. Al igual que el HTML es el World Wide Web Consortium el encargado de la especificación del estándar de las hojas de estilo que tanto desarrolladores como navegadores web deberán cumplir para crear y visualizar correctamente los documentos web.

La principal motivación para el desarrollo de las hojas de estilos en cascada es la de separar la estructura de un documento HTML de su presentación, consiguiendo así una notable mejora de la comprensión de los documentos y una mayor facilidad para su desarrollo.

Para poder modificar la presentación de los elementos HTML las hojas de estilo se componen mediante reglas. Cada regla se construye mediante uno o varios selectores, que a su vez pueden contener pseudo-clases y pseudo-elementos, y un bloque de declaración de estilos formado por una o varias declaraciones propiedad - valor. La sintaxis se puede ver reflejada en la siguiente figura:

```
selector [, selector2, ...] [:pseudo-class][::pseudo-element] {
 propiedad: valor;
 [propiedad2: valor2;
 ...]
}
```

Figura 24. Sintaxis CSS

En CSS los selectores sirven para indicar a cual o cuales elementos HTML se ven afectados por el bloque de propiedades definido para el selector. Como es posible definir distintas reglas que afecten a un mismo elemento HTML pueden producirse colisiones de estilos. Para solucionar estas colisiones y decidir cuál es la regla a aplicar, CSS se rige por el siguiente sistema de prioridades:

1. **Origen de las reglas:** Las reglas definidas por el autor del documento prevalecen sobre las reglas por defecto que los navegadores preestablecen a cada elemento.
2. **!important:** Si el valor de un estilo se marca como importante, este prevalecerá sobre el resto de estilos. En el supuesto que existan diferentes estilos marcados como importantes se continuará con la comprobación de las siguientes reglas de prioridades.
3. **Peso del selector:** Una regla con un selector de mayor peso prevalece sobre una con uno de menor peso, siendo el orden de prioridad:
    a. Selectores de clave: Son los selectores que acceden al atributo ID de un elemento HTML mediante el carácter reservado '#'.
    b. Selectores de clase: Son los selectores que acceden al atributo class de un elemento HTML mediante el carácter reservado '.'.
    c. Selectores HTML: Son los selectores que acceden directamente a la etiqueta HTML.
4. **Orden dentro de la lista de reglas:** Si la colisión no se ha resuelto con las reglas anteriores, prevalecerá la última de ellas.

Los selectores pueden ir acompañados de pseudo-clases y pseudo-selectores. Las pseudo-clases sirven para aplicar las reglas de estilos a estados especiales en los que se pueda encontrar un elemento HTML. Los estados más comunes son "link", que corresponde con los enlaces del documento que no han sido visitados, "visited", que corresponde con los enlaces visitados y "hover" que se activa cuando se pasa, con el puntero del ratón, por encima del elemento. Su sintaxis es selector:pseudo-clase. Los pseudo-elementos sirven para aplicar reglas a partes específicas del elemento seleccionado como por ejemplo: dar estilos a la primera letra, línea o subelemento del elemento seleccionado. Su sintaxis es selected::pseudo-elemento.

Las propiedades de las reglas CSS definen los diferentes atributos de presentación modificables de los elementos HTML como pueden ser: color de texto, color de fondo,

posición, tamaños de texto, tipografías, ancho y alto de los elementos, bordes, márgenes, etc. La propiedad indica el tipo de atributo y el valor define como se mostrará. La sintaxis es *propiedad1:valor; propiedad2:valor*.

```
#wrapper {
    color: black;
    font-family: Arial;
}

body {
    background-color: Red;
    font-size: 1em
}

.counter {
   position:absolute;
   top:0;
}
```

Figura 25. Reglas CSS con selector de ID, etiqueta y clase

Para dar formato CSS a un documento, existen tres maneras diferentes. La primera de ellas es el uso de los denominados estilos en línea. Consiste en insertar las reglas directamente en las etiquetas de apertura de los elementos HTML. Es una práctica bastante extendida heredada de revisiones anteriores, aunque muy poco recomendable. Incrustar las reglas supone duplicar cada una de ellas en cada elementos que compone el documento HTML, con su consiguiente dificultad a la hora de realizar actualizaciones. La segunda opción es el uso de hojas de estilo internas que consiste en incrustar la hoja de estilos dentro de la etiqueta <head> del documento HTML mediante el uso de la etiqueta <style>. Mediante esta opción se consigue separar los estilos del contenido, aun así no es la mejor opción para sitios web con numerosos documentos HTML, dado que habría que duplicar las reglas en cada uno de ellos. La tercera y mejor opción es la de utilizar hojas de estilo externas al documento, referenciadas mediante la etiqueta <link>. Es el modo más versátil ya que separa físicamente las reglas de estilos de la estructura. Además es muy común que sitios web con diferentes documentos exista una única hoja de estilos para todos ellos. Esta opción, también favorece el rendimiento de interpretación de las reglas en los navegadores que tienen la capacidad de cachear las hojas de estilos externas.

La especificación de la tercera versión de CSS incluye numerosas novedades, entre las que destacan: la capacidad de redondear bordes, realizar gradientes de colores, animaciones de elementos HTML con diferentes tipos de transiciones, uso de transparencias, múltiples imágenes de fondo, carga de fuentes externas, etc. Sin embargo lo que dota de gran potencia a la nueva revisión CSS3 es la capacidad de realizar consultas sobre el dispositivo en el que se está visualizando el documento web. Estas reglas se denominan “media queries” y permiten crear grupos de reglas que solo se aplicaran si el dispositivo cliente cumple con los requisitos marcados en la hoja de estilos. Son la base fundamental del diseño web responsivo.

Las media queries permiten consultar las siguientes propiedades:

- **Width y height**: permite conocer las dimensiones del área de visualización del navegador. Estas consultas pueden expresarse tanto en valores absolutos como mínimos y máximos.
- **Resolution**: permite conocer la densidad de pixeles del dispositivo.
- **Orientation:** permite comprobar si el dispositivo está en posición panorámica, mayor ancho que alto, o en posición vertical, mayor alto que ancho. Esta propiedad es especialmente útil en dispositivos móviles cuyas pantallas pueden girar.
- **Aspect-ratio:** permite conocer la proporción ancho: alto de la pantalla del dispositivo.
- **Color:** permite conocer la capacidad de reproducir colores que posee el dispositivo.

Las media queries se pueden aplicar a la hora de referenciar la hoja de estilos o bien insertadas directamente en las reglas. La sintaxis queda reflejada en la siguiente figura:

```
@media not|only mediatype and (media feature) {
    CSS-Code;
}
<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="mystylesheet.css">
```

Figura 26. Sintaxis de las media queries CSS

El uso de hojas de estilos para modificar la presentación a los documentos web proporciona importantes ventajas tanto en el desarrollo como a nivel técnico. Al hacer uso de ellas se obtiene un código fácilmente modificable y al separar el contenido y estructura de la presentación es extensible a diferentes dispositivos, evitando así la necesidad de crear distintas versiones de un mismo documento web para diferentes dispositivos. A nivel técnico el uso de hojas de estilos reduce el tamaño de los documentos y el tiempo que necesitan los navegadores para interpretarlas consiguiendo así páginas web más rápidas y eficientes.

2.3.3. JavaScript y JQuery

JavaScript, también denominado JS, es un lenguaje de programación interpretado, orientado a objetos, imperativo, débilmente tipado e imperativo que en el entorno web principalmente se utilizada como un lenguaje del lado de cliente, esto es, no se ejecuta en el servidor en donde está alojada la web sino que se ejecuta directamente en el navegador. Su sintaxis es un superconjunto de la especificación ECMAScript (ECMA-262) cuya estructura de programación es idéntica a la del lenguaje C salvo en la

definición del ámbito de las variables. En C el ámbito de las variables es el bloque en el que han sido declaradas y en JavaScript es el de la función en el que han sido declaradas.

El JavaScript se utiliza para escribir funciones que interactúan con los elementos HTML del documento a través del denominado DOM (Document Object Model), que es el conjunto estándar de objetos que los navegadores utilizan para representar los documentos HTML.

Los usos más habituales del JavaScript en las páginas web son:

- Solicitar contenido al servidor, a través de peticiones asíncronas (AJAX) sin la necesidad de recargar la página. Esto permite la actualización de distintas secciones del documento web sin la necesidad de recargar el resto de contenido que permanece estático.
- Animación de los distintos elementos de la página, cambiar su tamaño, modificar su posición dentro del DOM, eliminación de elementos existentes y creación de nuevos elementos.
- Validación de los valores de entrada de formularios web. Con la entrada de HTML5 esta práctica caerá en desuso.
- Recopilación de información sobre los hábitos de uso que los usuarios en el sitio web. Esta información está relacionada con aplicaciones de estadísticas de uso y aplicaciones de marketing que utilizan estos datos para enfocar las campañas y los anuncios.
- Captura de eventos que el usuario realiza en la página web para realizar acciones cuando estos ocurran.
- Dotar a los elementos de nueva funcionalidad que el HTML no contempla como por ejemplo la creación de galerías de imágenes dinámicas a través de listas HTML.

Como el JavaScript se ejecuta localmente, se ejecuta en el navegador del usuario, el tiempo de respuesta a las acciones programadas es rápido además tiene la capacidad de detectar acciones de los usuarios imposibles de detectar mediante HTML como las pulsaciones del teclado y el movimiento del ratón.

Todos los navegadores web actuales son capaces de interpretar JavaScript, de hecho es el único lenguaje de lado de cliente por el que los distintos desarrolladores de navegadores comparten su apoyo. Los navegadores incorporan un intérprete, denominado motor JavaScript, que interpreta el código y ejecuta los comandos codificados. Existen numerosos motores JavaScript y generalmente cada compañía desarrolladora de navegadores utiliza el suyo. Cada motor tiene sus ventajas e inconvenientes y no todos tienen en el mismo rendimiento ni implementan del mismo

modo el acceso a sus características. Los motores más destacados son: V8 utilizado por Google Chrome, Chakra utilizado por Internet Explorer a partir de su novena versión, SpiderMonkey utilizado por Mozilla Firefox y Nitro utilizado por los navegadores Safari de Apple.

Con el propósito de paliar las diferencias entre los diferentes motores JavaScript nace jQuery. JQuery es una librería, framework o API de funciones de código abierto que ofrece funcionalidades JavaScript. Dentro de estas funcionalidades se encarga de lidiar con las diferencias de los motores JavaScript para que el desarrollador no tenga que preocuparse por ello. Está construido sobre la filosofía "write less do more", esto es, utilizando esta librería con menos código, se implementa más funcionalidad que usando código JavaScript básico, reduciendo así el coste de desarrollo.

Para usar jQuery es necesario referenciar la librería dentro del documento web mediante el uso de la etiqueta <script>. Una vez referenciado dispondremos de acceso a toda la funcionalidad provista mediante la función "$()", también denominado selector jQuery. Mediante esta función podremos:

- Tener acceso completo a toda la estructura DOM del documento pudiendo seleccionar los diferentes elementos usando una sintaxis similar a la que se utiliza en CSS para realizar la selección de elementos.
- Manipular las hojas de estilos definidas en el documento.
- Realizar efectos y animaciones.
- Uso de AJAX simplificado.
- Soporta extensión. Existe una gran comunidad que realiza extensiones para JQuery que amplían aún más las funcionalidades de la librería.
- Es compatible con todos los navegadores principales.

JQuery no define un nuevo lenguaje de programación, solo define nuevas funciones. Para su uso es necesario programar con JavaScript y aprender a incorporar en el desarrollo las nuevas funciones que aporta.

2.3.4. Diseño responsivo

El diseño responsivo o adaptable es una filosofía emergente en el desarrollo del diseño web que consiste en adaptar la apariencia de la página web a las características del dispositivo que se está usando para su visualización. Esta filosofía nace ante la necesidad de dar respuesta a los cambios de hábitos de uso de las páginas web por parte de los usuarios, siendo en los últimos años los dispositivos móviles quienes más tráfico y uso generan en el mundo web. Esta técnica se fundamenta en las capacidades de CSS3 mediante el cual podemos obtener información acerca del dispositivo que se está utilizando y optimizar así las dimensiones de visualización de los textos, imágenes y contenido de tal manera que se legible en todos los dispositivos.

Esta técnica está recomendada como buena práctica por el consorcio W3C y con su uso conseguiremos mejorar la experiencia de navegación del usuario. Un usuario que accede a una web a través de un dispositivo mediante el cual no puede percibir correctamente el contenido, probablemente no vuelva a utilizar la web. Además al ser solo necesaria la implementación de una única plantilla de diseño, se reduce el coste de desarrollo comparándolo con webs que implementan diferentes plantillas para cada tipo de dispositivo, como por ejemplo los portales móviles "m.midominio.com".

Figura 27. Diseño responsivo

2.4. LAMP

Para poder dar soporte a la aplicación que vamos a implementar vamos a utilizar la infraestructura LAMP, que es una combinación de diferente software mediante los cuales se define una infraestructura open source y gratuita de un servidor web. La infraestructura LAMP se compone de: Linux como sistema operativo, Apache como servidor web, MySQL como sistema de base de datos relacional y PHP como lenguaje de programación.

Siendo la mencionada la agrupación más extendida, es posible la instalación de la infraestructura en diferentes sistemas operativos como WAMP y MAMP, en donde el sistema operativo utilizado es Windows y OSX respectivamente. También existen alternativas en el uso del sistema de bases de datos pudiendo usar MariaBD y MongoBD, y en el uso del lenguaje de programación, pudiendo usar Perl y Phyton.

Como nuestro objetivo es crear una aplicación open source que pueda ser utilizada y modificada por terceros, vamos a basarla en la agrupación más extendida: Linux, Apache, MySql y PHP.

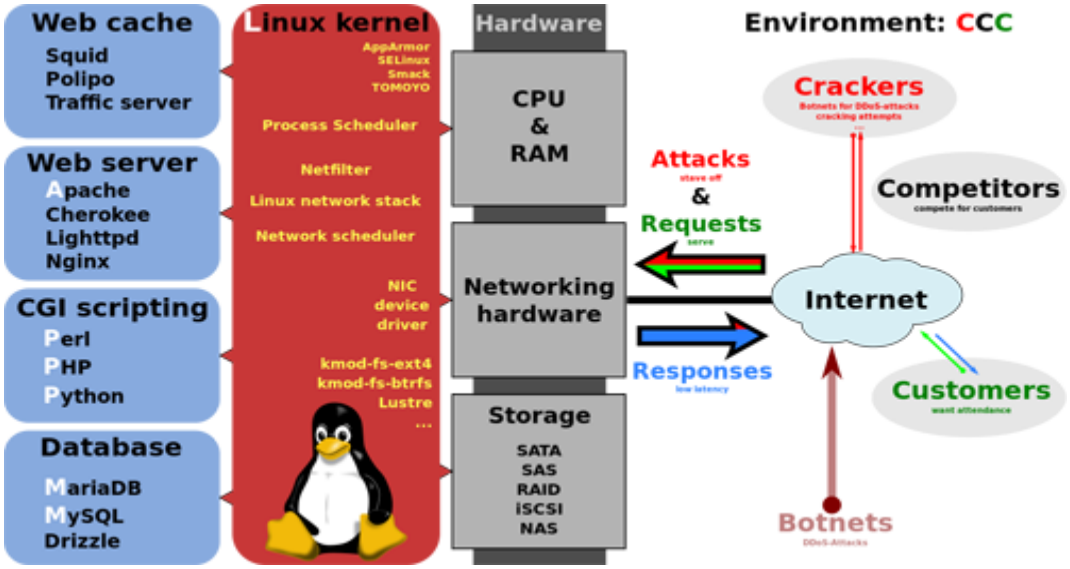


Figura 28. Arquitectura de la infraestructura LAMP

A continuación vamos a ver las características de cada componente de la infraestructura elegida.

2.4.1. Apache

Apache es un servidor web HTTP que aunque en un principio fue escrito para Unix, posteriormente fue implementado como servidor web multiplataforma, pudiendo ser utilizado en plataformas Linux, Windows, OS, etc. Es el servidor web más extendido a nivel mundial desde 1996 debido en gran parte a su estabilidad, seguridad y facilidad de uso. Además es un software open source que puede ser utilizado y modificado libremente por los usuarios. Sigue en evolución debido al continuo desarrollo por la comunidad Apache Software Foundation, que se encargan de dotar al servidor de nuevas funcionalidades y mejoras.

Apache, como todo servidor web, se encarga de procesar las peticiones de contenido web que recibe y servirlas a través de la red al navegador web solicitante que a su vez se encargara de interpretar el contenido de la respuesta recibida. Para realizar esta comunicación utiliza el protocolo HTTP.

El protocolo HTTP (HyperText Transfer Protocol) es el protocolo estándar que permite la comunicación en los entornos web a través del puerto 80 TCP. Es un protocolo orientado a transacciones que sigue el esquema de petición-respuesta entre el cliente y el servidor. La información que se transmite desde el servidor se denomina recurso, se identifica mediante su URI y puede ser el resultado del proceso de texto estático, de la ejecución de algún programa, consultas sobre la base de datos, etc. Se trata de un protocolo sin estado, no guarda información acerca de las peticiones realizadas, lo cual choca con la frecuente necesidad de las web de almacenar datos de las peticiones. Los servidores web se encargan de paliar esta necesidad mediante la inclusión de sistemas que permiten guardar datos tanto en el cliente, mediante las cookies, como en el servidor mediante las variables de sesión.

Cada petición HTTP está formada por encabezados en los que se especifican la dirección del contenido solicitado y opcionalmente una serie de metadatos acerca del solicitante y datos enviados desde formularios web.

```
GET /index.php HTTP/1.1
Host: cgnp.sandbox.com.es
User-Agent: Mozilla Firefox
[Línea en blanco]
```

Figura 29. Petición HTTP

Como respuesta de la petición, el servidor web envía al cliente el código de estado de la petición, en el cual se indica si la petición se procesado correctamente o no, el contenido de la petición y metadatos del contenido. Los códigos de respuesta más comunes son: 1xx conexión rechazada, 200 petición correcta, 3xx redirecciones de contenido, 4xx error en la parte cliente al realizar la petición y 5xx error de servidor al procesar la solicitud.

```
HTTP/1.1 200 OK
Date: Fri, 05 Jun 2015 16:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html>
<body>
<h1>Notificaciones PUSH</h1>
</body>
</html>
```

Figura 30. Respuesta HTTP

Para poder realizar las transacciones, el protocolo HTTP define ocho métodos diferentes:

- **GET**: Solicita una respuesta con el contenido del recurso solicitado. La información de esta petición se transmite a través de la URI agregando los parámetros correspondientes. Ej: GET /peticion.php?idioma=es. No es un buen método para trasmitir información que pueda ser sensible dado que quedara expuesta.
- **POST**: Envía datos para que sean procesados por el recurso al que van dirigidos. Los datos se incluyen en el cuerpo de la petición HTTP.
- **PUT**: Se utiliza para subir archivos desde el cliente al servidor mediante una conexión vía socket. Generalmente no se utiliza debido a que puede ocasionar problemas de seguridad, además los servidores hosting no suelen tener esta opción habilitada.

- **DELETE**: Borra el recurso solicitado.
- **HEAD**: Solicita una respuesta GET pero sin el contenido de la solicitud. Sirve para obtener la información acerca de la respuesta pero no la respuesta en sí.
- **TRACE**: Sirve para solicitar al servidor que añada a la respuesta, los datos de como se ha realizado la comunicación. Se utilizada para realizar diagnósticos y comprobar el rendimiento de las peticiones.
- **OPTIONS**: Devuelve la información acerca de los diferentes métodos HTTP que soporta el servidor web mediante el cual se está realizando la petición.
- **CONNECT**: Sirve para establecer la conexión con el servidor web de conexiones encriptadas.

El servidor Apache, como la mayoría de servidores, también implementa el protocolo HTTPS, que amplia al protocolo HTTP añadiendo una capa de seguridad. En este protocolo toda la información que se transmite entre el cliente y el servidor va encriptada. Esto dota de una seguridad extra en la comunicación a costa de una comunicación más lenta. Se utiliza en páginas cuya información es sensible, como por ejemplo las páginas de las entidades bancarias.

Otra ventaja del servidor Apache es su modularidad. Dispone de un núcleo central que implementa la comunicación básica entre el cliente y servidor a partir del cual se puede ampliar su funcionalidad a través de la instalación de módulos. A través de estos módulos el servidor Apache puede realizar conexiones a numerosos sistemas de gestión de bases de datos, da soporte a numerosos lenguajes de programación, como PHP, Python, Perl, JAVA e incluso es capaz de ejecutar lenguajes de programación web propietarios, como son ASP y C#.NET de Microsoft. Dentro de estos módulos cabe destacar los módulos “mod_rewrite“, utilizado para transformar URI dinámicas en estáticas y “mod_deflate”, utilizado para comprimir la respuesta con el fin de agilizar la comunicación. Estos módulos han sido posteriormente copiados por el resto de servidores web.

En definitiva, Apache es un servidor fácil de usar, extensible a través de sus módulos, multiplataforma, popular, gratuito y eficiente que se ajusta a los objetivos de nuestro proyecto.

2.4.2. MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado en ANSI C y aunque está desarrollado por una empresa privada, Sun Microsystems que pertenece a Oracle, se ofrece bajo licencia GNU GPL que permite la libertad de uso, compartición y modificación del producto. Esta licencia a contribuido a que MySQL sea uno de los sistemas de gestión de bases de datos más utilizado en entornos web, siendo usado por muchas de las grandes web existentes como Wikipedia, Facebook y YouTube.

MySQL permite la administración de bases de datos relacionales, que los las bases de datos que almacenan los datos en diferentes tablas conectadas por relaciones que permiten el acceso rápido y con la posibilidad de realizar búsquedas y combinaciones de datos. Las principales características de MySQL son:

- Uso del lenguaje SQL (Structured Query Language) para la interacción con los datos almacenados.
- Es multiplataforma pudiendo ser instalado en diferentes sistemas como Linux, Windows, OSX, Solaris, etc.
- Diferentes mecanismos de almacenamiento de los datos.
- Capacidad de distribuir los datos de forma geográfica.
- Control de transacciones.
- Soporte para claves extranjeras.
- Conexión segura.
- Sistemas de replicación de las bases de datos.
- Búsquedas e indexación de campos de texto.

Como hemos visto para la interacción con los datos MySQL utiliza SQL o Lenguaje Estructurado de Consultas del inglés Structured Query Language. Es un lenguaje declarativo de acceso a bases de datos relacionales que define una gran cantidad de operaciones. Podemos entender SQL como el conjunto de dos sublenguajes el DDL (Data Definition Language) y el DML (Data Manipulation Language)

El lenguaje de definición de datos DDL, se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye las funciones necesarias para la manipulación de las tablas en donde se almacenan los datos. DDL define cuatro operaciones básicas: Create, Alter, Drop y Truncate.

CREATE
Permite la creación de nuevas tablas, vistas sobre las tablas y procedimientos almacenados.

ALTER
Comando para modificar la estructura de una tabla. Permite la creación y borrado de campos, modificar los tipos de cada campo, añadir o quitar índices, etc.

DROP
Permite eliminar cualquier objeto de la base de datos.

TRUNCATE
Comando para truncar el contenido de una tabla, esto es, permite el borrado completo del contenido de una tabla. Es el sistema más rápido para borrar contenido pero no permite hacer borrados parciales.

El lenguaje de manipulación de datos DML, permite realizar operaciones de consulta y manipulación de los datos guardados dentro de una base de datos relacional a través de diferentes sentencias.

SELECT

Permite realizar consultas simples y complejas sobre los datos almacenados en la base de datos. Define diferentes palabras reservadas mediante las cuales se pueden realizar combinaciones, agrupaciones, ordenaciones y agregar condiciones que los datos deben cumplir para ser añadidos en el resultado de la consulta.

INSERT

Mediante esta sentencia se pueden agregar registros a una tabla. Los valores que se van a insertar deben cumplir las restricciones y deben ajustarse a la estructura de la tabla en la que se van a insertar. También es posible combinar esta sentencia con la sentencia SELECT para realizar copias de datos desde otras tablas.

UPDATE

Esta sentencia se utiliza para modificar el valor de uno o varios campos de un conjunto de registros. Como en la sentencia INSERT los datos deben cumplir las restricciones asociadas a la tabla.

DELETE

Se utiliza esta sentencia para realizar borrados de uno o más registros de una tabla.

2.4.3. PHP

PHP (Hypertext Preprocessor) es un lenguaje de programación de código abierto que se utiliza para el desarrollo de aplicaciones web. Como particularidad a otros lenguajes también utilizados para el desarrollo web, PHP tiene la capacidad de poder ser incrustado directamente en el código HTML. Se trata de un lenguaje que se ejecuta en el servidor de manera transparente para los usuarios, los cuales no pueden acceder al código que se está ejecutando. Mediante este código se genera una respuesta HTML que es la que se envía al cliente para que sea interpretada en su navegador web.

PHP está enfocado para la programación de scripts y por su naturaleza de estar preparado para ejecutarse en servidores HTTP, es capaz de recopilar los datos que se generan en este tipo de comunicaciones, como recuperar los datos enviados por formularios, recibir y crear cookies, etc.

Para que se pueda ejecutar el servidor web debe tener instalado el analizador PHP, que se provee a modo de un módulo instalable. Es multiplataforma pudiendo emplearse en los principales sistemas operativos siendo el sistema Linux y servidor Apache la configuración más habitual.

Aunque está enfocado a la programación de scripts, también se puede utilizar para realizar programación por procedimientos o programación orientada a objetos, característica que dota de gran versatilidad al lenguaje. Además que para generar

contenido HTML, PHP se extiende a través de módulos que amplían la capacidad del lenguaje permitiendo: la creación de imágenes, ficheros PDF, interactuar con el sistema de ficheros local del servidor en donde se esté ejecutando, acceso a bases de datos, procesamiento de textos por medio de expresiones regulares, etc. PHP también tiene la capacidad de comunicarse otros servicios usando protocolos como LDAP, IMAP, POP3, SNMP, HTTP, etc.

Para poder ejecutar código PHP este debe estar contenido entre sus limitadores. Los limitadores de apertura pueden ser <?php o <? y el de cierre es ?>. Si se incluye código PHP fuera de estos limitadores será procesado como texto plano y no como código. Para definir variables estas se tiene que prefijar con el símbolo $ y como se trata de un lenguaje débilmente tipado, no es necesario indicar el tipo de las variables. Las variables pueden recibir cualquier nombre y distinguen entre mayúsculas y minúsculas. Para definir cadenas de texto se puede utilizar indistintamente las comillas simples y las dobles. En cuanto a las palabras reservadas y escritura de sentencias, PHP comparte la sintaxis de C y cada sentencia debe finalizar con punto y coma (;).

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <title> Ejemplo básico PHP</title>
    </head>
    <body>
        <?php
            echo 'Hola mundo';
        ?>
    </body>
</html>
```

Figura 31. Código PHP incrustado en HTML

A diferencia de otros lenguajes PHP es interpretado y aunque en general este tipo de lenguajes son considerablemente más lentos a la hora de procesar el código que otros lenguajes de bajo nivel o compilados, para paliar este inconveniente PHP provee de mecanismos para utilizar técnicas de cache en archivos físicos y en memoria.

Debido a la flexibilidad, potencia y alto rendimiento, PHP ha tenido una gran acogida como lenguaje base para el desarrollo de aplicaciones web siendo unos de los lenguajes más utilizados para el desarrollo de portales web dinámicos. Incluso portales con gran demanda de tráfico lo utilizan como por ejemplo Facebook y Wikipedia.

2.5. Marco de trabajo

Para poder llevar a cabo el proyecto prevemos que será necesario utilizar aplicaciones que nos permitan realizar su desarrollo. El conjunto de todas las aplicaciones utilizadas compondrán nuestro marco de trabajo.

2.5.1. Sublime Text

El Sublime Text es un editor de texto plano especializado en edición de lenguajes de marcado como es el HTML. Además dispone de múltiples módulos adicionales que facilitan el desarrollo de aplicaciones con lenguajes de programación como PHP mediante un entorno de desarrollo integrado que realiza comprobaciones de sintaxis en tiempo real del código escrito. Estas características facilitan el desarrollo de las aplicaciones.

2.5.2. Adobe Photoshop

Adobe Photoshop es un programa de edición de imágenes que utilizaremos para los ajustes necesarios en la creación de la interfaz gráfica del desarrollo.

2.5.3. PhpMyAdmin

PhpMyAdmin es una interfaz gráfica que permite el acceso y manipulado de bases de datos realizadas con MySQL. La utilizaremos para la creación de la base de datos y depuración en el desarrollo.

2.5.4. Firezilla

Firezilla es un cliente FTP para realizar transferencias de archivos. Lo utilizaremos para poder transferir los archivos locales del proyecto al servidor web en el cual se alojara el portal web.

2.5.5. LAMP

Como hemos comentado anteriormente usaremos LAMP como infraestructura de nuestra aplicación web.

2.5.6. Android Studio

Android Studio es un entorno de desarrollo gratuito provisto por Google que permite crear aplicaciones móviles para dispositivos móviles basados en Android. En nuestro proyecto lo utilizaremos para crear la app con la que testear la aplicación creada.

2.5.7. Xcode

Xcode es el entorno de desarrollo provisto por Apple que permite crear aplicaciones móviles para dispositivos móviles basados en IOS. En nuestro proyecto lo utilizaremos para crear la app con la que testear la aplicación creada.

3. ANÁLISIS DE FACTIBILIDAD

Antes de realizar el proyecto se ha completado un análisis previo para determinar si es un proyecto realizable o por el contrario es inviable.

El presente proyecto debe desarrollar una herramienta que permita la comunicación mediante notificaciones Push entre los desarrolladores de aplicaciones para dispositivos móviles y los clientes de sus aplicaciones. Además debe ofrecer una interfaz de usuario amigable y fácil de utilizar, cualquier usuario con conocimientos básicos en herramientas de gestión web debería poder utilizarla.

Tras estudiar los requisitos del proyecto y analizando la documentación ofrecida por Apple y Google del funcionamiento de las notificaciones Push en sus plataformas, podemos concluir con que existen métodos que permiten realizar el proyecto por lo tanto se trata de un proyecto realizable.

Sabiendo que se trata de un proyecto realizable y habiéndose tenido en cuenta los factores de riesgo, recogidos en el *Documento de Objetivos del Proyecto (DOP)* podemos considerar que el proyecto es factible.

4. DOCUMENTO DE OBJETIVOS DEL PROYECTO

En este capítulo se presenta el Documento de Objetivos del Proyecto (DOP). En él se van a recopilar las pautas de control y gestión del proyecto. Se explicará el alcance y los objetivos generales del proyecto. Igualmente se incluirá una descripción del proyecto, de cada una de las fases en las que se descompone, como van a estar planificadas, la metodología de trabajo, los entregables, el seguimiento, los riegos que presenta el proyecto y las fuentes de información que se prevén utilizar.

4.1. Descripción

Centro de gestión de notificaciones Push es el nombre del proyecto propuesto por el alumno Txomin Vila Ruiz como Proyecto de Fin de Carrera.
Se trata de una plataforma web de libre distribución, que permitirá la gestión y envío de notificaciones Push a aplicaciones móviles desarrolladas tanto para IOS (Apple) como para Android (Google).

El proyecto se desarrollará utilizando la infraestructura LAMP: sistema operativo Linux, servidor web Apache, sistema de gestión de bases de datos MySQL y lenguaje de programación PHP. La elección responde a que es la combinación más utilizada y popular en entornos web, es flexible y además es gratuita. De este modo, el centro de gestión de notificaciones Push podrá ser libremente instalado tanto en entornos privados, servidores locales, en servidores publicados en internet o bien ser utilizarlo en modo SAAS (Software como servicio).

El proyecto constara de 2 módulos:

- Un Back-end (o módulo de administración), que será un portal web, accesible bajo petición de credenciales, con un menú de usuario que permitirá: la configuración de las aplicaciones móviles destinatarias de las notificaciones; la planificación y envío de las notificaciones; la visualización del histórico de envíos y estadísticas de los mismos. Cumplirá los estándares HTML5 y CSS3 por lo que será accesible y funcional desde cualquier dispositivo que disponga de un navegador web que interprete los estándares web. Además se seguirá la filosofía del diseño web adaptable (Responsive Design) para que el portal web se adapte al entorno que el usuario este utilizando, sea una ordenador de sobremesa, una Tablet o un Smartphone.

- Un servicio web, que actuara como interfaz intermedia y que se encargara de la comunicación entre el Back-end y los servidores APSN (Apple / IOS) y GCM (Google / Android) que son los que permiten la comunicación con los dispositivos móviles. Además también será posible acceder, con su debida

autenticación, al servicio web de modo externo, pudiendo ser usado a modo de API desde aplicaciones de terceros.

4.2. Motivaciones y objetivos

El desarrollo de este proyecto viene motivado por la experiencia personal en el desarrollo de aplicaciones móviles, en las que una vez realizadas se solicitaba la realización de un sistema ad hoc para el envío de notificaciones Push que en la mayoría de ocasiones debido a su coste no se llevaban a cabo.

También se barajaba la opción de usar otras herramientas de autor tipo SAAS, pero debido a sus costes y poca versatilidad tampoco eran opción.

El objetivo de este proyecto es realizar una herramienta web totalmente funcional y de libre distribución que permita el envío y la gestión de notificaciones Push a aplicaciones de dispositivos móviles basados en plataformas IOS y Android. De este modo los programadores de aplicaciones móviles dispondrán de una herramienta versátil que podrán utilizar en sus proyectos sin coste añadido.

Para poder llegar al objetivo final será necesario cumplir los siguientes hitos:

- Definir el modelo de base de datos para persistir la información de la herramienta

- Construir el Back-end (módulo de administración) que permitirá a los usuarios interactuar con la aplicación.

- Realizar el servicio web que permita la comunicación entre la aplicación y los dispositivos móviles.

- Desarrollar una aplicación móvil, tanto para IOS como Android, para poder enviar las notificaciones Push.

4.3. Alcance del proyecto (Fases y actividades de cada fase)

Este proyecto se ha dividido en las siguientes fases, describiendo a continuación cada una de ellas con las actividades que lo componen.

4.3.1. Gestión del proyecto

Esta fase se llevará a cabo a lo largo de todo el proyecto. En ella se incluyen las tareas de tomas de decisiones, reuniones y comunicación con el director del proyecto.

4.3.2. Análisis y obtención de requerimientos

El análisis de requerimientos es la fase que traerá consigo un estudio de las tecnologías implicadas en el proyecto. Para ello será necesario:

- El estudio del funcionamiento de los servidores APSN de Apple que son los encargados de permitir la comunicación con dispositivos IOS.

- El estudio del funcionamiento de los servidores GCM de Google que son los encargados de permitir la comunicación con dispositivos Android.

- El estudio de los estándares HTML5 y CSS3 necesario para el desarrollo del Back-end.

- El estudio del protocolo REST para el desarrollo del servicio web.

- El análisis de la infraestructura LAMP y la información necesaria sobre Linux, Apache, MySQL y PHP para la implementación del proyecto.

- Instalación del software necesario para la realización del proyecto.

- Planear un sistema de backups para evitar cualquier tipo de pérdida de información o avance del proyecto.

- Al finalizar esta fase dispondremos de las herramientas y conocimiento necesario para el desarrollo completo del proyecto.

4.3.3. Diseño del sistema

El objetivo de esta fase es la definición de la arquitectura del sistema y la especificación detallada de todos los componentes que van a componer el sistema.

Comenzaremos con el diseño del modelo de la base de datos. Se realizara un modelo UML definiendo en él la estructura de la base de datos, los tipos de datos que serán utilizados y las relaciones entre los datos.

La siguiente tareas será el diseño del Back-end que comprenderá tanto los diseños de las interfaces con las que el usuario final podrá interactuar con la plataforma, como el diseño estructural a nivel de programación y la toma de decisiones del tipo programación a utilizar: programación orientada a objetos o programación estructurada.
Otra de las tareas en este punto es la definición del servicio web encargado de la comunicación entre el Back-end y los servidores de notificaciones Push. Se definirán los métodos públicos y la funcionalidad de cada uno de ellos.

4.3.4. Implementación

En esta fase se implementara la plataforma web completa, que engloba al Back-end de administración y el servicio web de comunicación. Así mismo se implementara una aplicación móvil para el testeo de la plataforma tanto en versión IOS como en versión Android.

4.3.5. Pruebas y correcciones

Se elaborarán un conjunto de pruebas, de tal manera que se pueda realizar un testeo completo de la aplicación. Esto ayudara a encontrar los posibles fallos de la aplicación con los que realizaremos un informe de errores que se irán corrigiendo a medida que se vayan encontrando.

4.3.6. Elaboración de manuales

Para facilitar el uso de la aplicación a los usuarios, se desarrollaran dos manuales de ayuda, el primero denominado "Manual de usuario", en el que se explicara el funcionamiento del programa y el modo de uso del servicio web, el segundo denominado "Manual de instalación" expondrá los pasos s a seguir para la correcta instalación de la plataforma en web compatibles.

4.3.7. Documentación

La documentación se redactará a medida que vaya avanzando el proyecto. Se recopilará la información adquirida durante la ejecución de cada una de las fases en las que se divide el proyecto. Esto abarca desde la toma de decisiones y diseño del sistema hasta los detalles de implementación del mismo.

4.4. Metodología de trabajo

El trabajo se realizará individualmente por el alumno dedicando las horas necesarias para alcanzar el objetivo del proyecto. Dividiremos la metodología en tres tipos de procesos: métodos tácticos, métodos formativos, y métodos operativos.

4.4.1. Métodos tácticos

Los métodos tácticos son aquellos referidos a la toma de decisión y control de proyecto. Se establecerá una comunicación vía email o reuniones con el director del proyecto para valorar las decisiones de las herramientas a utilizar y para realizar un control de situación. Al ser un proyecto de naturaliza web se pondrá a disposición un dominio web http://cgnp.sandbox.com.es en el cual se podrán ver los progresos del proyecto.

4.4.2. Métodos formativos

Los métodos formativos abarcan las tareas de aprendizaje de uso de las herramientas y tecnologías necesarias. Una vez elegidas las tecnologías a utilizar será necesario adquirir los conocimientos necesarios consultando libros, manuales o tutoriales.

4.4.3. Métodos operativos

Una vez definido el alcance total y objetivos del proyecto habrá que realizar el desarrollo de los mismos. Las tareas que el desarrollador deberá realizar son las siguientes:

- Documento de objetivos de proyecto
- Análisis y captura de requerimientos
- Diseño del sistema
- Implementación
- Pruebas y correcciones
- Elaboración de manuales
- Documentación y memoria

4.5. Planificación

La fecha de comienzo del proyecto está fijada para Octubre del 2014, siendo en un principio la finalización del mismo en Junio del 2015. En el supuesto de tener que retrasar la finalización del mismo deberá realizarse en a plazo para entregarlo en el siguiente curso lectivo. A continuación se describen las fases con sus tareas y las horas que se esperan invertir en cada una de ellas.

4.5.1. Gestión del proyecto

La estimación total de la fase es de 35 horas y se desglosan de la siguiente manera:

TAREA	SESIONES	HORAS	TOTAL
Reuniones con el director del proyecto	8	1,5 hora	12 horas
Actas de las reuniones	8	1 hora	8 horas
Backups diarios del sistema	60	0,25 hora	15 horas
		TOTAL:	35 horas

Tabla 5. Planificación de la "Gestión del proyecto"

4.5.2. Análisis y obtención de requerimientos

La estimación total de la fase es de 61 horas y se desglosan de la siguiente manera:

TAREA	TOTAL
Estudio de los servidores APNS	10 horas
Estudio de los servidores GCM	10 horas
Estudio de HTML5, CSS3 y diseño responsivo	16 horas
Estudio del protocolo REST	10 horas
Estudio de la infraestructura LAMP	10 horas
Instalación del software en el equipo	5 horas
TOTAL:	61 horas

Tabla 6. Planificación del "Análisis y obtención de requerimientos"

4.5.3. Diseño del sistema

La estimación total de la fase es de 34 horas y se desglosan de la siguiente manera:

TAREA	TOTAL
Modelo de la base de datos	8 horas
Diseño del back end	16 horas
Redacción de los casos de uso	5 horas
Diseño del servicio web	5 horas
TOTAL:	34 horas

Tabla 7. Planificación del "Diseño del sistema"

4.5.4. Implementación

La estimación total de la fase es de 132 horas y se desglosan de la siguiente manera:

TAREA	TOTAL
Implementación del modelo de la base de datos	4 horas
Implementación del back end	100 horas
Implementación del servicio web	20 horas
Implementación de las aplicaciones móviles para realizar las pruebas	8 horas
TOTAL:	132 horas

Tabla 8. Planificación de la "Implementación"

4.5.5. Pruebas y correcciones

La estimación total de la fase es de 15 horas y se desglosan de la siguiente manera:

TAREA	TOTAL
Pruebas de la aplicación	10 horas
Correcciones	5 horas
TOTAL:	15 horas

Tabla 9. Planificación de las "Pruebas y correcciones"

4.5.6. Elaboración de manuales

La estimación total de la fase es de 10 horas y se desglosan de la siguiente manera:

TAREA	TOTAL
Manual de instalación	10 horas
TOTAL:	10 horas

Tabla 10. Planificación de la "Elaboración de manuales"

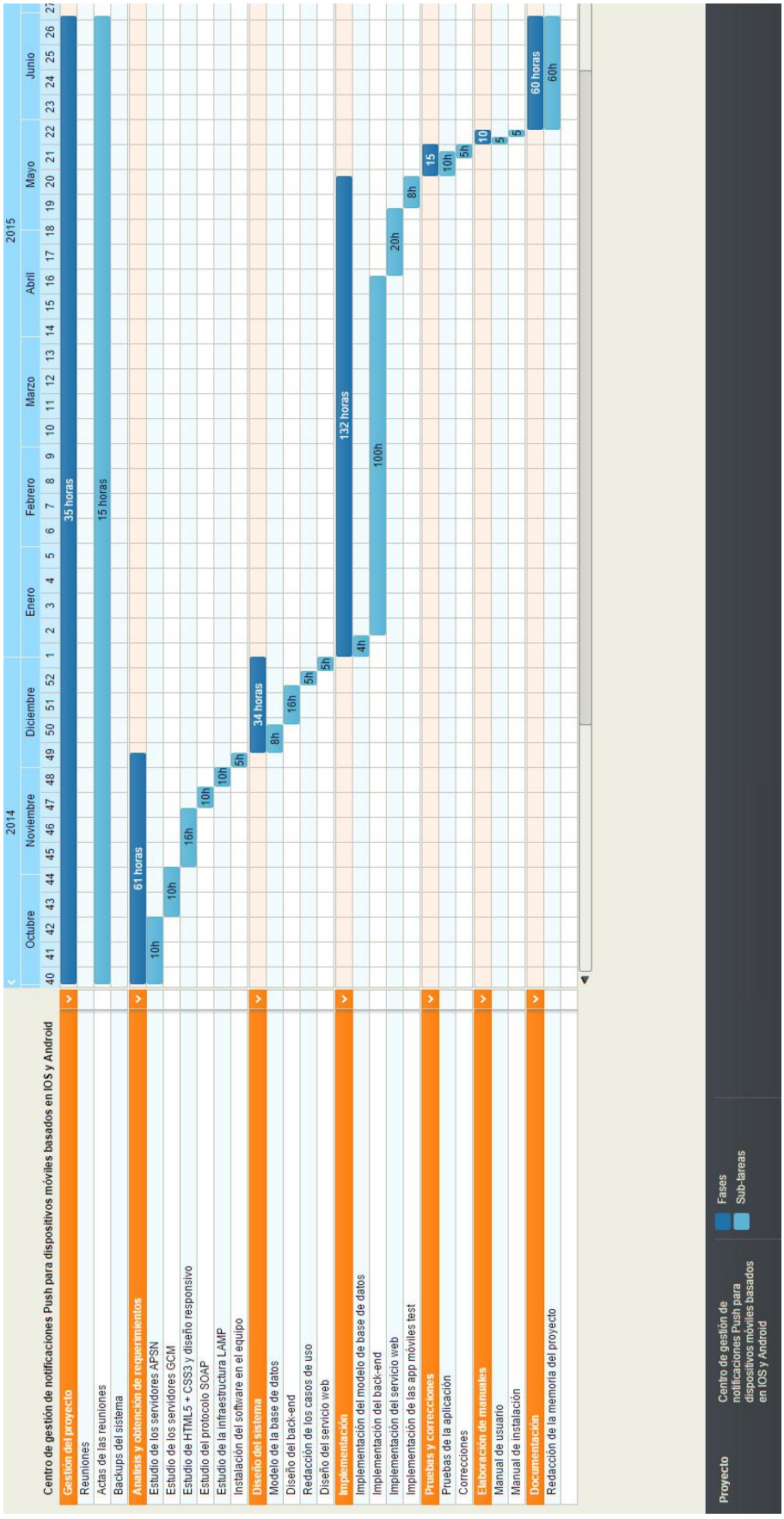
4.5.7. Documentación

La estimación total de la fase es de 60 horas y se desglosan de la siguiente manera:

TAREA	TOTAL
Redacción de la memoria del proyecto	60 horas
TOTAL:	60 horas

Tabla 11. Planificación de la "Documentación"

4.6. Diagrama de GANTT



4.7. Factores de riesgo

Este proyecto presenta unos factores de riesgo que hay que considerar. A continuación se presenta una lista de todos ellos con la solución propuesta si estos se producen.

4.7.1. Proximidad del plazo de entrega

El plazo de entrega para el presente curso finaliza el día 1 de Julio, lo cual nos deja dos meses escasos para la realización del proyecto. Debido a que pueden surgir complicaciones en el desarrollo del mismo puede acarrear el incumplimiento del plazo. Si esto sucediese se volverían a planificar los plazos para pasarlo al año siguiente, desarrollando el proyecto en los plazos marcados en el curso lectivo.

4.7.2. Enfermedad

Una enfermedad puede afectar a la planificación del proyecto. Si se trata de una enfermedad corta, que no supere la semana, lo resolveremos añadiendo más carga de trabajo en los siguientes días para evitar que el desarrollo se vea afectado. Si por el contrario es una enfermedad de largo periodo habrá que realizar una nueva planificación.

4.7.3. Vacaciones

Durante el periodo de vacaciones es muy posible que las personas implicadas en el proyecto no estén accesibles, no se pueda mantener una comunicación ni realizar reuniones. Para ello se preverán estas situaciones dejando previamente zanjadas todas las decisiones que puedan suponer un retraso en la planificación durante el periodo vacacional, de tal modo que el desarrollo del proyecto no se detenga.

4.7.4. Recursos

Los recursos son una parte fundamental de este proyecto y aunque en principio solo será necesario un ordenador para el desarrollo, siempre puede ocurrir una avería del equipo. Para ello el desarrollador va a disponer de dos equipos, un ordenador de sobremesa y un ordenador portátil. De este modo minimizados el impacto de una avería en uno de ellos disponiendo del otro para continuar el trabajo. Además la existencia de backups periódicos evitara la perdida de información y en caso que ambos equipos fallen, se podrá realizar una instalación rápida en cualquier otro equipo.

4.7.5. Dificultad del proyecto

Aunque el desarrollado cuenta con experiencia suficiente como para abordar desarrollo del proyecto, puede ocurrir que la dificultad sea mayor de la esperada en un primer momento. Si esto ocurriese habría que hacer una nueva planificación e invertir horas adicionales para completar el desarrollo.

4.7.6. Modificaciones de los requisitos

La modificación de los requisitos afectara a la planificación inicial del proyecto. Si esto ocurre habrá que realizar un estudio de los cambios y una nueva planificación.

4.8. Entregables del proyecto

Los entregables del proyecto son los siguientes y se entregaran en modo impreso y/o digital:

- Actas de reuniones.
- Informes del estudio de los servidores APSN.
- Informe del estudio de los servidores GCM.
- Informe del estudio HTML5 + CSS3 y diseño responsivo.
- Informe del protocolo SOAP.
- Informe de la infraestructura LAMP.
- Informe de los casos de uso de la aplicación.
- Informe sobre la implementación del proyecto, diagramas UML del sistema de gestión de bases de datos y de funcionamiento / documentación del sistema.
- Informe de las pruebas realizadas.
- Manual de instalación.
- Memoria.

4.9. Fuentes de información y recursos necesarios

Las fuentes de información serán los manuales de los portales web de las tecnologías a utilizar para el desarrollo del proyecto, siendo esta una información más actualizada de la que podamos encontrar en libros o publicaciones en papel. Las principales fuentes serán las siguientes webs:

- https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html [Servidores APSN]
- http://developer.android.com/google/gcm/index.html [Servidores GCM]
- http://www.w3schools.com/ [HTML5 + CSS3]
- http://www.w3.org/TR/soap/ [Servicios web]
- http://lamphowto.com/ [LAMP]

5. RESULTADOS EXPERIMENTALES

En este apartado se exponen el diseño y la implementación de la aplicación desarrollada, que incluye el backend, el servicio web y las aplicaciones de dispositivos móviles para la recepción de las notificaciones Push.

5.1. Diseño

5.1.1. Centro de gestión de notificaciones Push (back-end)

El dentro de gestión de notificaciones Push es un back-end web en los que se definen los siguientes casos de uso

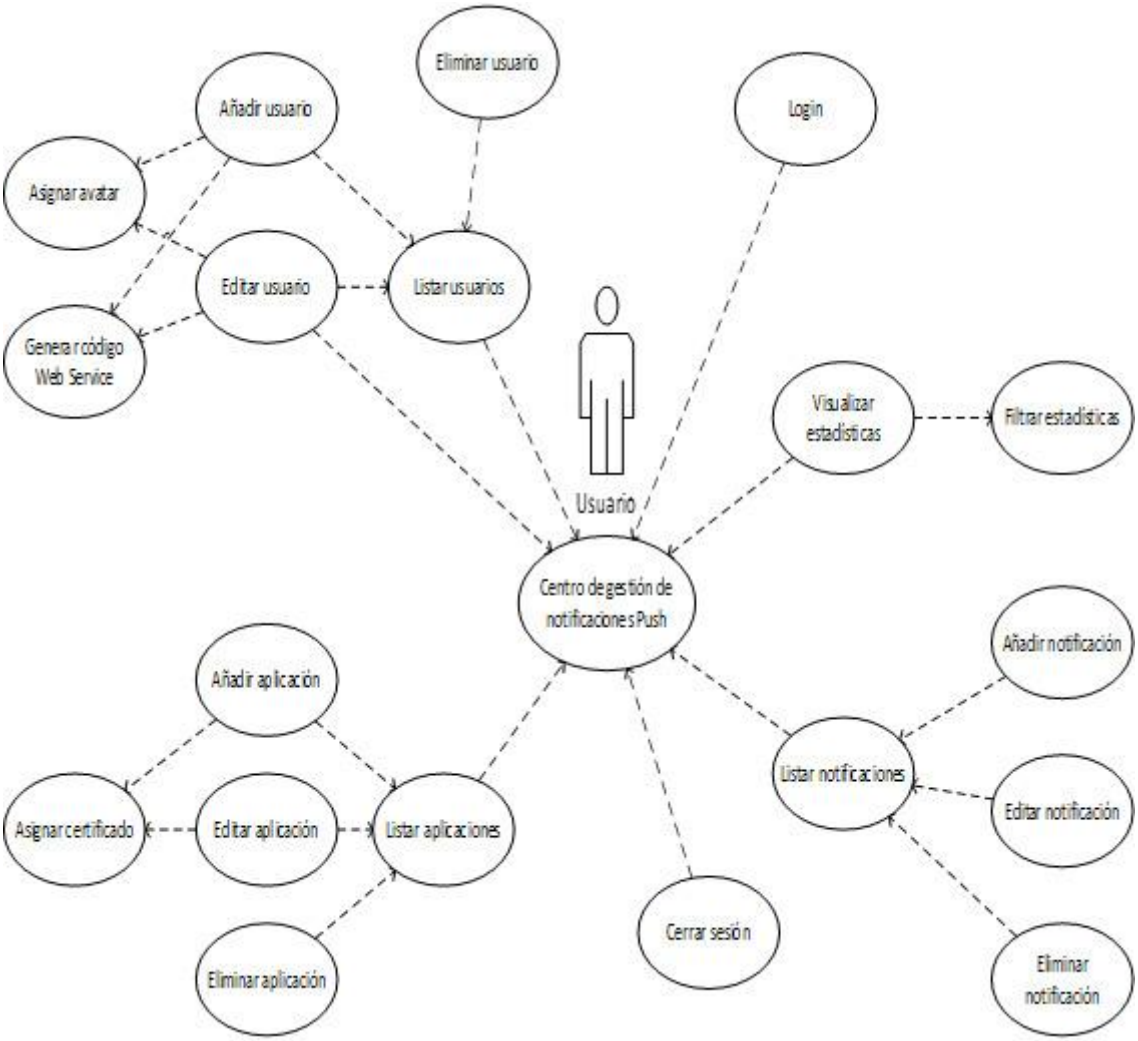


Figura 32. Casos de uso del centro de gestión de notificaciones Push

En los siguientes subapartados se muestran las descripciones de cada caso de uso.

5.1.1.1. Login

El acceso a la plataforma está restringido a usuarios dados de alta en el sistema. Para poder acceder la plataforma el usuario deberá autenticarse mediante su nombre de usuario y contraseña.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario no está autenticado en el sistema.
- Flujo de eventos:
    1. El usuario accede a la plataforma mediante un navegador web.
    2. La plataforma muestra un formulario web solicitando los datos de acceso.
    3. El usuario introduce los datos.
    4. Si los datos son correctos la plataforma concede el acceso en caso contrario nuestra un mensaje de error y vuelve al paso 2.
- Postcondiciones: El usuario se mantiene autenticado hasta que cierre la sesión o el navegador.

5.1.1.2. Visualizar estadísticas

Interfaz en la que se visualizan datos del uso de la aplicación y estadísticas de envío de las notificaciones Push.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. El usuario selecciona la opción “dashboard” en el menú de usuario.
    2. La plataforma muestra la siguiente información:
        - Información acerca del estado del sistema, numero de aplicaciones configuradas, notificaciones enviadas dispositivos notificados y notificaciones pendientes de enviar.
        - Gráfico vectorial en el que se visualiza la relación entre las notificaciones enviadas y las notificaciones leídas por los destinatarios.

5.1.1.3. Filtrar estadísticas

Permite filtrar el gráfico en el que se visualiza la información entre las notificaciones enviadas y las notificaciones leídas por los destinatarios.

- Actores: Usuario de la plataforma.

- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “dashboard”.
- Flujo de eventos:
    1. El usuario hace clic sobre el botón filtrar del gráfico vectorial.
    2. El usuario selecciona las aplicaciones, la fecha inicial y la fecha final.
    3. Se actualiza el gráfico mostrando la información relativa a los filtros seleccionados por el usuario.

5.1.1.4. Listar notificaciones

Interfaz en la que se visualizan el listado de notificaciones y el estado de las mismas. El estado de la notificación puede ser enviada o pendiente de envío.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. El usuario selecciona la opción “Planificador Push” en el menú de usuario.
    2. La plataforma muestra el listado de notificaciones.

5.1.1.5. Añadir notificación

Permite programar un nuevo envío de una notificación Push a una aplicación.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “Planificador Push”
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Nueva notificación Push”.
    2. La plataforma muestra el formulario de alta de notificaciones Push.
    3. El usuario selecciona la aplicación destinataria, la fecha y hora de envío, la insignia y sonido de la notificación, si se trata de una notificación para actualizar contenido y el mensaje de la notificación. Una vez realizado pulsa sobre el botón “Crear notificación”.
    4. La plataforma añade la notificación para que se proceda a su envío.

5.1.1.6. Editar notificación

Permite editar una notificación Push.

- Actores: Usuario de la plataforma.

- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “Planificador Push”.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Editar notificación” de una de las notificaciones del listado.
    2. La plataforma muestra el formulario con los valores de la notificación seleccionada.
    3. El usuario edita los valores correspondientes y pulsa el botón “Modificar notificación”
    4. La plataforma actualiza los datos de la notificación.

5.1.1.7. Eliminar notificación

Elimina una notificación Push

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “Planificador Push”.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Eliminar notificación” de una de las notificaciones del listado.
    2. La plataforma elimina la notificación.
- Postcondiciones: La plataforma mantiene los datos resultantes del envío de la notificación para evitar que se pierdan los datos estadísticos.

5.1.1.8. Listar aplicaciones

Interfaz en la que se visualiza el listado de aplicaciones para dispositivos móviles configuradas. Estas aplicaciones son las destinatarias de las notificaciones Push.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. El usuario selecciona la opción “APPs” en el menú de usuario.
    2. La plataforma muestra el listado de aplicaciones.

5.1.1.9. Añadir aplicación

Permite añadir una nueva aplicación destinataria de las notificaciones Push.

- Actores: Usuario de la plataforma.

- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “Apps”
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Añadir APP”.
    2. La plataforma muestra el formulario de alta de una nueva aplicación.
    3. El usuario introduce el nombre y descripción de la aplicación, si la aplicación es para dispositivos basados en IOS y/o dispositivos basados en Android
        - Si se trata de una aplicación basada en IOS el usuario tiene que subir el certificado, indicar si la aplicación está en desarrollo o producción y la contraseña del certificado.
        - Si se trata de una aplicación para dispositivos basados en Android el usuario tiene que introducir el “Api Key” de su aplicación.
    4. La plataforma añade la aplicación para que esté disponible como destinataria de nuevas notificaciones Push.

5.1.1.10. Editar aplicación

Permite editar los parámetros de una aplicación.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “APPs”
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Editar APP” de una de las aplicaciones del listado.
    2. La plataforma muestra el formulario de edición con los valores de la aplicación seleccionada.
    3. El usuario edita los valores los valores correspondientes y pulsa el botón “Modificar APP”
    4. La plataforma actualiza los datos de la aplicación.

5.1.1.11. Eliminar aplicación

Elimina una aplicación del sistema

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “APPs”.
- Flujo de eventos:

1. El usuario hace clic sobre la opción “Eliminar APP” de una de las aplicaciones del listado.
2. La plataforma elimina la aplicación.

- Postcondiciones: Se mantienen los datos resultantes del envío de las notificaciones a la aplicación borrada para evitar que se pierdan los datos estadísticos.

5.1.1.12. Asignar certificado

Permite asignar el certificado de una aplicación móvil para dispositivos basados en IOS. Este certificado lo obtiene el creador de la aplicación desde su cuenta de desarrollador para dispositivos IOS.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección de alta o edición de una aplicación para dispositivos basados en IOS.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Subir certificado PEM”.
    2. La plataforma abre el sistema de ficheros local.
    3. El usuario selecciona el certificado de su disco local y pulsa el botón aceptar.
    4. La plataforma hace una copia del certificado y la almacena en el alojamiento web en donde está instalada.

5.1.1.13. Listar usuarios

Interfaz en la que se visualizan el listado de usuarios con acceso a la plataforma. Al instalar la plataforma, por defecto existe un usuario root o administrador.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y es el usuario root.
- Flujo de eventos:
    1. El usuario selecciona la opción “Usuarios” en el menú de usuario.
    2. La plataforma muestra el listado de usuarios activos.

5.1.1.14. Añadir usuarios

Permite añadir un nuevo usuario

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema, se encuentra en la sección “Usuarios” y es el usuario root.

- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Añadir Usuario”.
    2. La plataforma muestra el formulario de alta de un nuevo usuario.
    3. El usuario introduce el nombre, email, nombre de usuario, contraseña, idioma y opcionalmente genera su código Push.
    4. La plataforma añade al usuario.

5.1.1.15. Editar usuarios

Permite editar los datos asociados a un usuario. El usuario root será capaz de poder modificar los datos de cualquier usuario. Un usuario estándar solo podrá modificar sus datos.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección “Usuarios” si es un usuario root o “Mi perfil” si es un usuario estándar.
- Flujo de eventos:
    1. El usuario root hace clic sobre la opción “Editar Usuario” de uno de los usuarios del listado. Si es un usuario estándar hace clic en la opción “Mi Perfil” del menú.
    2. La plataforma muestra el formulario de edición con los valores del usuario seleccionado.
    3. El usuario edita los valores los valores correspondientes y pulsa el botón “Modificar Usuario”
    4. El sistema actualiza los datos del usuario.

5.1.1.16. Eliminar usuario

Elimina a un usuario de la plataforma.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema, se encuentra en la sección “Usuarios” y es el usuario root.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Eliminar Usuario” de uno de los usuarios del listado.
    2. La plataforma elimina al usuario y sus datos relacionados del sistema.
- Postcondiciones: La plataforma mantiene los datos resultantes del envío de las notificaciones a las aplicaciones de este usuario para evitar que se pierdan los datos estadísticos.

5.1.1.17. Asignar avatar

Permite asignar un avatar a un usuario para mostrarlo en la interfaz de la plataforma.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección de alta o edición de usuarios.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Subir avatar”.
    2. La plataforma abre el sistema de ficheros local.
    3. El usuario selecciona la imagen de su disco local y pulsa el botón aceptar.
    4. La plataforma hace una copia de la imagen y la almacena en el alojamiento web en donde está instalada.

5.1.1.18. Generar código Web Service

Permite generar un código Push que permitirá el acceso al servicio web (API) de la plataforma a través de aplicaciones de terceros.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema y se encuentra en la sección de alta o edición de usuarios.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Generar código Web Service”.
    2. La plataforma genera un código aleatorio de 16 dígitos y rellena el campo del formulario con el valor generado.
    3. El usuario pulsa el botón modificar usuario.
    4. La plataforma actualiza los datos introducidos.

5.1.1.19. Cerrar sesión

Cierra la sesión del usuario autenticado en el sistema.

- Actores: Usuario de la plataforma.
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. El usuario hace clic sobre la opción “Cerrar sesión”
    2. La plataforma muestra una interfaz solicitando la confirmación de cierre de sesión.
    3. El usuario confirma la solicitud.

4. La plataforma cierra la sesión del usuario redireccionando a la interfaz de acceso.

5.1.2. Servicio web

Mediante el desarrollo de un servicio web vamos a ampliar y exponer parte de la lógica de negocio de nuestro desarrollo para permitir que diferentes aplicaciones externas puedan interactuar con algunas de las características de la plataforma. El servicio web define los siguientes casos de uso:

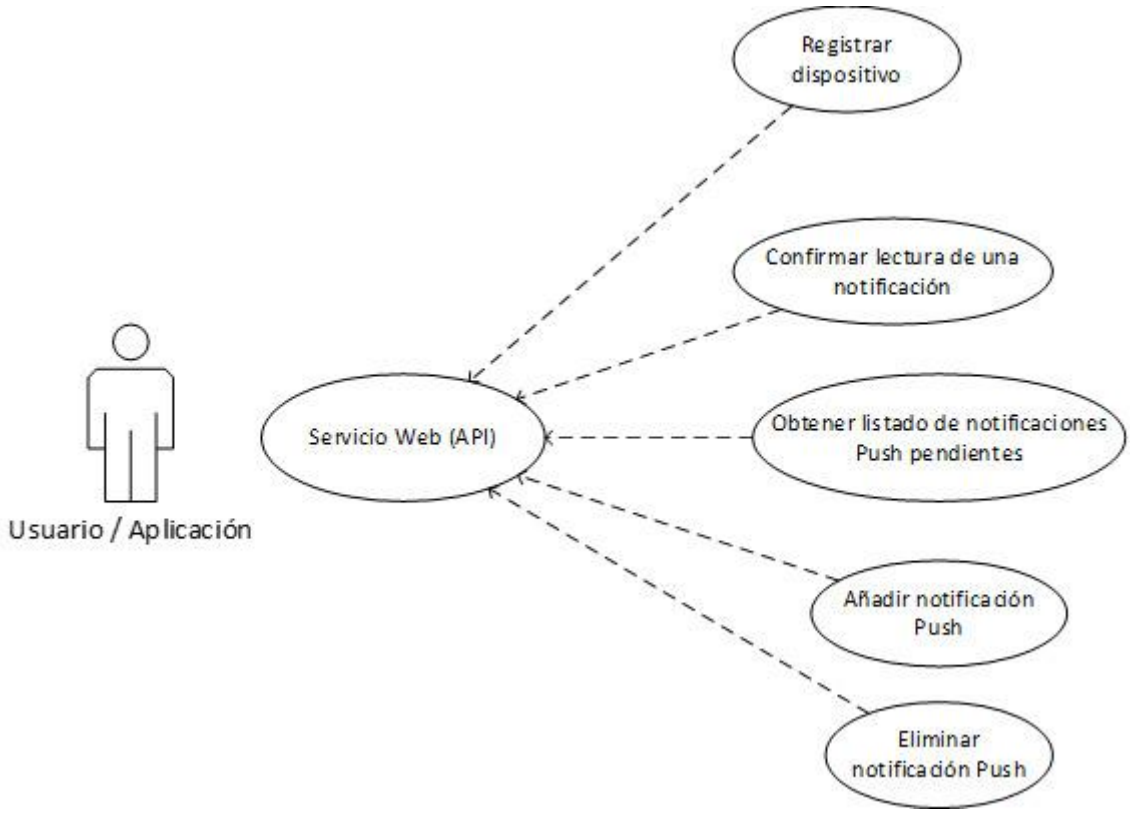


Figura 33. Casos de uso del servicio web (API)

En los siguientes subapartados se muestran las descripciones de cada caso de uso.

5.1.2.1. Registrar dispositivo

Permite registrar un dispositivo a la lista de dispositivos destinatarios de una aplicación móvil dada de alta en la plataforma. Por lo general este servicio se invocará automáticamente cuando un dispositivo instala una aplicación móvil, de un desarrollador que utiliza la plataforma de gestión de notificaciones Push.

- Actores: Dispositivo móvil o usuario de la plataforma.

- Precondiciones: Deben existir en la plataforma el usuario y aplicación referenciados en la petición.
- Flujo de eventos:
    1. El usuario o aplicación genera la petición al servicio, incluyendo el nombre de usuario, el código de acceso al servicio web, el identificador de la aplicación, el identificador del dispositivo y la plataforma móvil (IOS o Android).
    2. Si la petición es aceptada la plataforma registra el dispositivo. lo asocia a la aplicación a la que pertenece.
    3. La plataforma genera un mensaje JSON con el resultado de la operación.

5.1.2.2. Confirmar lectura de una notificación

Permite registrar la confirmación de lectura de una notificación Push enviada a un dispositivo concreto. El desarrollador de la aplicación móvil deberá programar la llamada a este servido desde el evento de abrir notificación que proveen las plataformas de desarrollo de aplicaciones moviles.

- Actores: Dispositivo móvil.
- Flujo de eventos:
    1. El dispositivo móvil recibe una notificación Push enviada a través del centro de gestión de notificaciones Push.
    2. El usuario del dispositivo móvil recibe la notificación e interactúa con la misma accediendo a la aplicación móvil.
    3. La aplicación genera una petición al servicio incluyendo, el identificador del usuario, el código de acceso al servicio web y el identificador de la notificación recibida.
    4. La plataforma actualiza el estado de la notificación clasificándola como leída
    5. La plataforma devuelve un mensaje JSON con el resultado de la operación.

5.1.2.3. Obtener listado de notificaciones Push pendientes

Permite obtener un listado, en formato JSON, de las notificaciones Push que están pendientes de envío.

- Actores: Usuario de la plataforma o aplicación externa.
- Flujo de eventos:
    1. El usuario o aplicación generan una petición al servicio incluyendo el identificador del usuario, el código de acceso al servicio web y

opcionalmente el identificador de una aplicación configurada en la plataforma.

2. La plataforma responde con un mensaje JSON que incluye el listado de notificaciones pendientes solicitadas.

5.1.2.4. Añadir notificación Push

Permite programar un nuevo envío de una notificación Push a una aplicación. La motivación de este servicio es la de que una aplicación externa a la plataforma pueda programar notificaciones Push cuando se produzca algún evento.

- Actores: Usuario de la plataforma o aplicación externa.
- Flujo de eventos:
    1. El usuario o aplicación externa generan una petición al servicio incluyendo el identificador del usuario, el código de acceso al servicio web, la fecha y hora de envío, la insignia y sonido de la notificación, si se trata de una notificación para actualizar contenido y el mensaje de la notificación.
    2. La plataforma añade la notificación para que se proceda a su envío.
    3. La plataforma responde con un mensaje JSON con el resultado de la operación.

5.1.2.5. Eliminar notificación Push

Permite eliminar una notificación Push.

- Actores: Usuario de la plataforma o aplicación externa.
- Precondiciones: La notificación a eliminar todavía no se ha enviado.
- Flujo de eventos:
    1. El usuario o aplicación generan una petición al servicio incluyendo el identificador del usuario, el código de acceso al servicio web y opcionalmente el identificador de la notificación.
    2. La plataforma elimina la notificación del sistema.
    3. La plataforma responde con un mensaje JSON con el resultado de la operación.

5.1.3. Interfaz gráfica

El diseño de la interfaz gráfica del back end se ha seguido el patrón de diseño responsivo, cuyo objetivo es que la interfaz se adapte al dispositivo usado, con el fin de mejorar la experiencia de uso a los usuarios. Para ello se ha diseñado una interfaz sencilla e intuitiva que cualquier usuario con conocimientos básicos sobre el entorno web será capaz de interactuar con ella satisfactoriamente.

El diseño de la interfaz es a pantalla completa y se han definido tres adaptaciones globales a tipos de dispositivos: ordenadores de escritorio, tablets y dispositivos móviles con unas resoluciones medias mayores a 1024 pixeles, entre 800 y 1024 pixeles y menores de 800 pixeles respectivamente.

Mediante la utilización de CSS3 se consultan las propiedades del dispositivo y la interfaz se adapta a las características de la pantalla de visualización.

La interfaz y su adaptabilidad a diferentes dispositivos se pueden ver en las siguientes figuras:

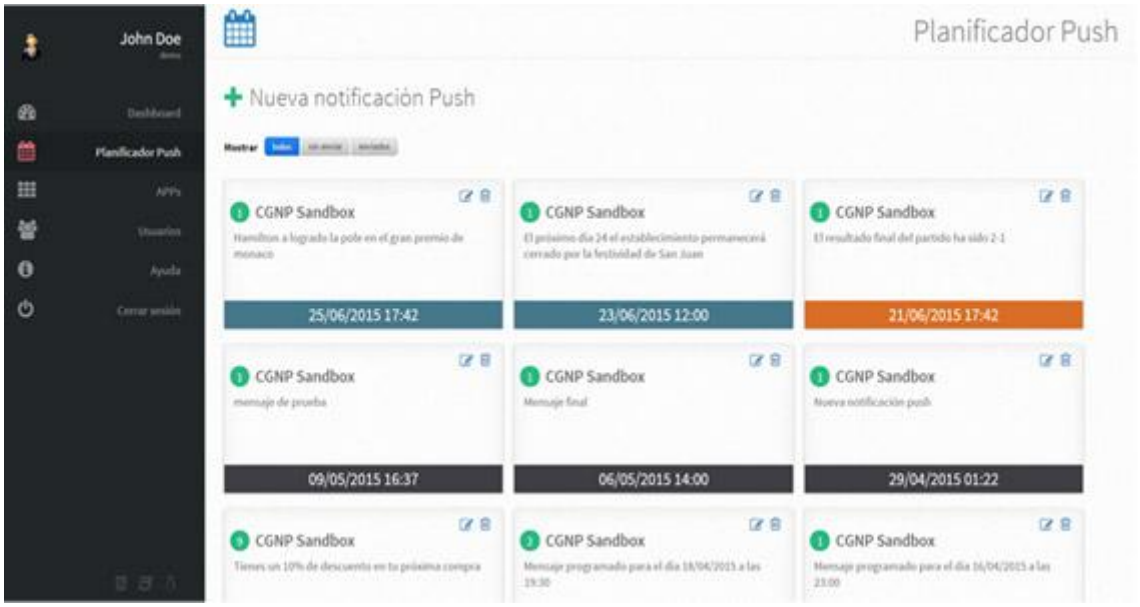


Figura 34. Interfaz gráfica adaptada a ordenadores de escritorio

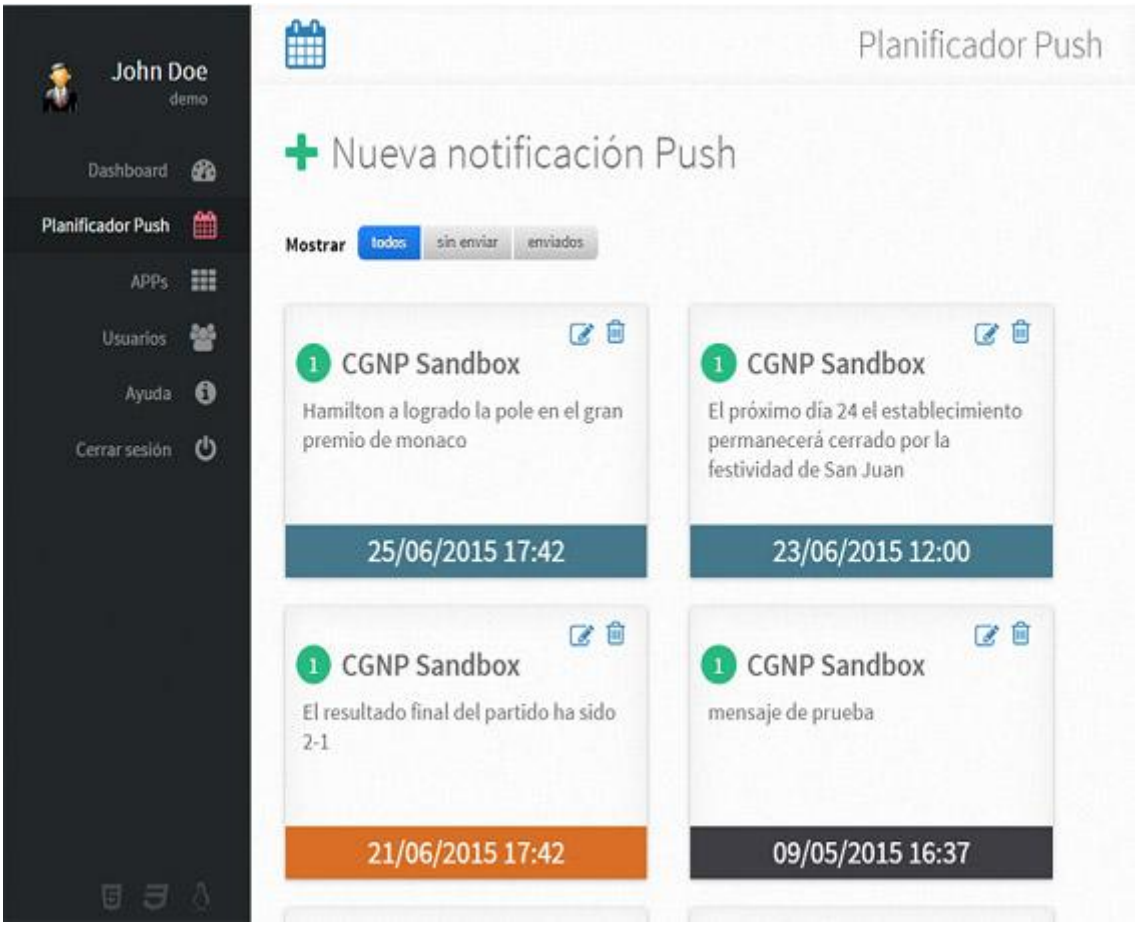


Figura 35. Interfaz gráfica adaptada a tablets

Figura 36. Interfaz gráfica adaptada a dispositivos móviles con el menú de usuario plegado

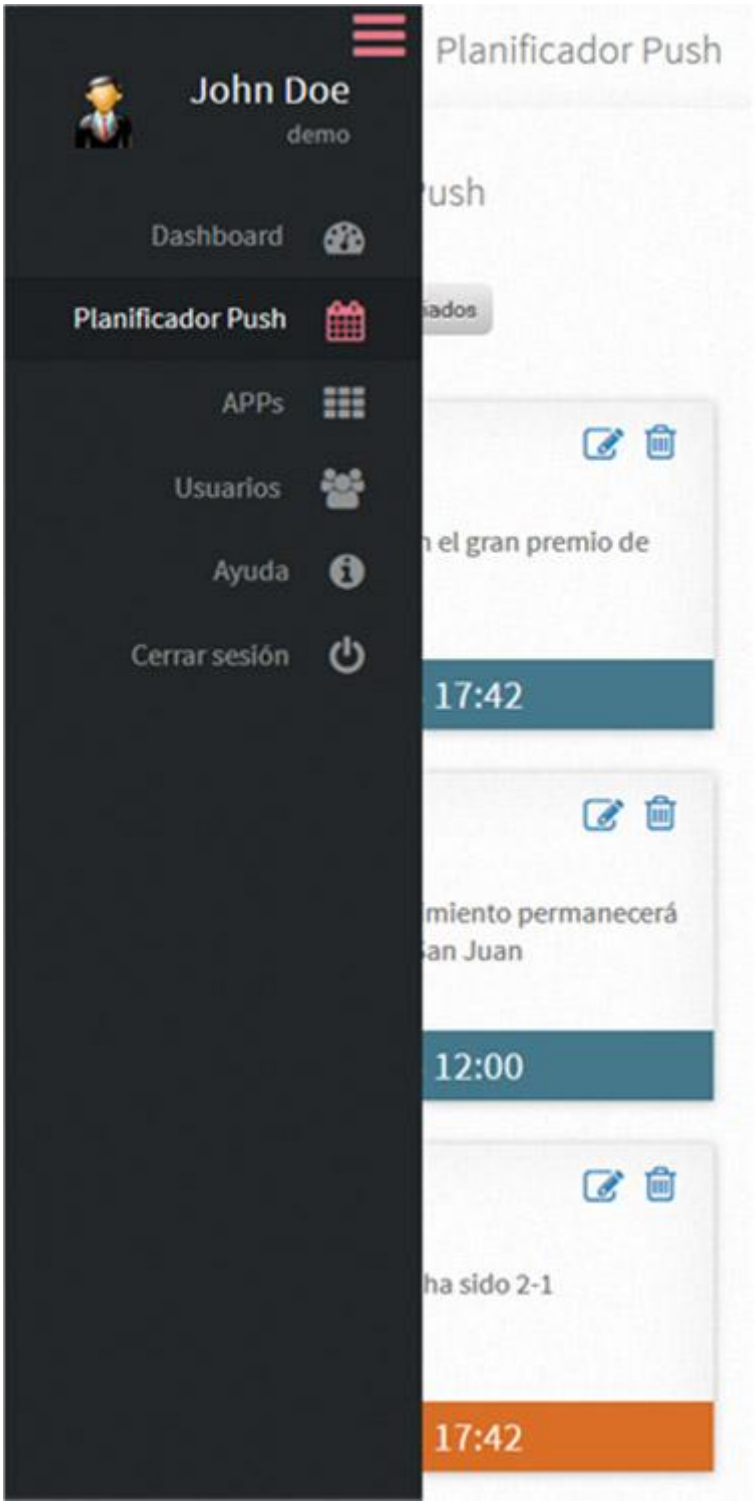


Figura 37. Interfaz gráfica adaptada a dispositivos móviles con el menú de usuario desplegado

5.1.4. Aplicaciones móviles

Para poder realizar las pruebas de funcionamiento de la plataforma vamos a desarrollar 2 aplicaciones móviles (Apps), una para entorno IOS y otra para entorno Android.

Estas aplicaciones no definen casos de uso dado que la única finalidad de ellas es la de recibir las notificaciones enviadas desde el centro de gestión de notificaciones Push desarrollado que se definen en el caso de uso "añadir notificación".

5.2. Implementación

5.2.1. Tecnologías

Para el desarrollo del proyecto se han usado gran variedad de tecnologías que incluyen las empleadas para implementar las capas de presentación, la lógica de negocio, el acceso a datos y la infraestructura del servidor web.

5.2.1.1. Presentación

Para la presentación de los casos de uso, formularios y controles, se empleado la tecnología HTML5, que agrupa las tecnologías HTML5, CSS3 y JavaScript. En nuestra implementación hemos usado JavaScript a través de la librería jQuery.

Figura 38. Tecnología HTML5 + CSS3 + jQuery

5.2.1.2. Lógica de negocio

La lógica de negocio de las funcionalidades del backend de ha implementado con PHP como lenguaje de programación. Para la creación de las aplicaciones móviles con fines de test se ha empleado Xcode para la versión IOS y Java para la versión Android.

Figura 39. Tecnología php, Java y Objective-C

5.2.1.3. Acceso a datos

Para almacenar los datos de la aplicación se ha utilizado el sistema de gestión de bases de datos relacional MySQL.

Figura 40. Tecnología MySQL

5.2.1.4. Infraestructura web

Para que el desarrollo sea funcional y pueda ser accesible a través de la web se ha empleado el servidor HTTP Apache, que será el encargado de intermediar en la comunicación entre el navegador utilizado por el cliente y la lógica de negocio de la aplicación.

Figura 41. Tecnología Apache

5.2.2. Software empleado

El software utilizado para el desarrollo del proyecto ha sido obtenido desde los sitios web de los proveedores del mismo. Cabe destacar a excepción del software necesario para la implementación de la aplicación de test para dispositivos móviles basados en IOS, todo el software utilizado de uso libre y gratuito.

5.2.2.1. Sublime Text

Se ha utilizado el editor de texto avanzado Sublime Text para el desarrollo de las capas de presentación y la lógica de negocio del back end, dado que su flexibilidad permite su uso para editar tanto HTML, CSS, JavaScript y código PHP.

5.2.2.2. Adobe Photoshop

Se ha utilizado Adobe Photoshop para realizar retoques fotográficos a diversos elementos de la interfaz gráfica del back end.

5.2.2.3. PhpMyAdmin

Se ha empleado PhpMyAdmin para la creación y manipulación de la estructura de la base de datos MySQL que emplea la plataforma.

5.2.2.4. Firezilla

Se ha utiliza este cliente FTP para transferir la implementación realizada al servidor web en donde se aloja la plataforma.

5.2.2.5. LAMP

Como infraestructura web se ha empleado LAMP, que incluye el sistema operativo Linux como sistema operativo, el servidor Apache como servidor HTTP o web, MySQL como motor de bases de datos y PHP como lenguaje de programación.

5.2.2.6. Android Studio

Para la implementación de la aplicación móvil para el testeo de las notificaciones Push en sistemas basados en Android, se ha empleado el entorno de desarrollo Android Studio provisto por Google.

5.2.2.7. Xcode

Para la implementación de la aplicación móvil para el testeo de las notificaciones Push en sistemas basados en IOS, se ha empleado el entorno de desarrollo Xcode provisto por Apple.

5.2.3. Centro de gestión de notificaciones Push – Arquitectura del sistema

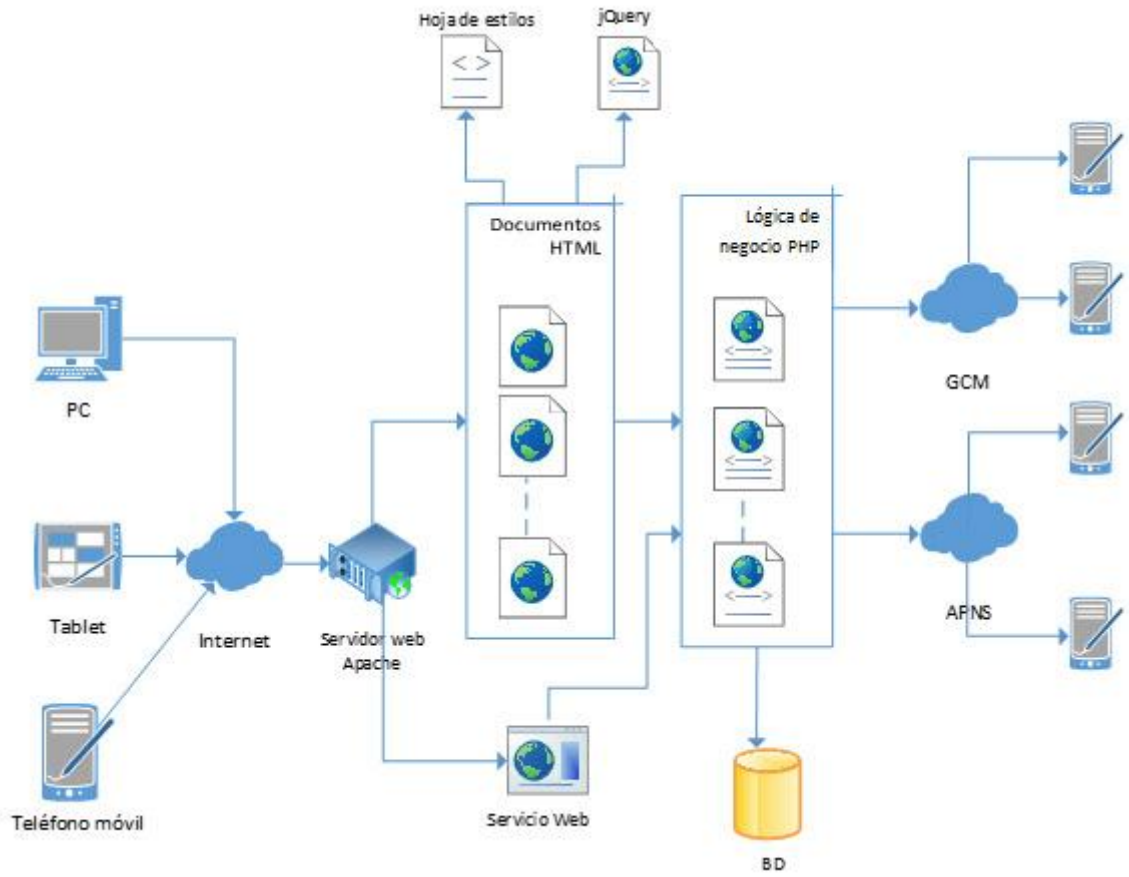


Figura 42. Arquitectura del sistema

Para la implementación de la plataforma se han utilizado 2 capas: la capa de presentación y la capa de lógica de negocio.

La capa de presentación se compone de un conjunto de documentos HTML, estructurados utilizando la última versión del estándar HTML5, que enriquecidos mediante la creación de una hoja de estilos CSS y scripts de lado de cliente, usando la librería jQuery, expone al usuario una interfaz web homogénea con la cual va a poder interactuar con la funcionalidad de la plataforma.

La capa de lógica de negocio agrupa al conjunto de clases creadas, siguiendo el paradigma de programación orientado a objetos (POO), que se encargan de implementar la funcionalidad del sistema, la creación de contenido HTML dinámico, el acceso a la base de datos y el envío de las notificaciones Push.

Así mismo se ha implementado un servicio web siguiendo la arquitectura REST que expone parte de la lógica de negocio de la aplicación para que pueda ser utilizada desde aplicaciones de terceros, sin tener que acceder al back end implementado.

5.2.4. Descripción de las clases

En este apartado vamos a realizar la especificación funcional del sistema detallando las clases implementadas que componen la lógica del negocio y su interfaz gráfica.

5.2.4.1. Clase Login

Figura 43. Formulario de acceso a la plataforma

El formulario de acceso a la plataforma solicita las credenciales del usuario, nombre de usuario y contraseña. Una vez introducidas el sistema las valida permitiendo el acceso en caso que están sean correctas y denegándolo en caso contrario.

Si se permite el acceso se crea una variable a de sesión que contiene el identificador del usuario para que permanezca logeado en el sistema. Esta variable se elimina bien si el usuario cierra el navegador, cierra la sesión o si no se produce interacción alguna con la plataforma durante 20 minutos.

Para dotar a la plataforma de un extra de seguridad, una vez el usuario accede al sistema se genera un token aleatorio para la sesión con el fin de evitar ataques Cross Scripting. Este tipo de ataques se aprovechan de vulnerabilidades de las web para intentar obtener accesos ilegítimos o manipular el contenido de la web.

La clase Login se encarga de la lógica de negocio que gestiona el acceso a la plataforma. Cuenta con los siguientes atributos:

- `Private` PDO: Instancia a la clase de acceso a la base de datos.
- `Private` ID: Código identificador del usuario.
- `Private` Admin: Valor booleano que indica si el usuario es administrador o no.
- `Private` Name: Nombre del usuario
- `Private` Username: Nombre de usuario para el acceso a la plataforma.
- `Private` Avatar: Ruta de la imagen que actúa como avatar del usuario
- `Private` Lang: Identifica el idioma del usuario
- `Private` Token: Cadena de texto aleatoria para evitar ataques Cross Scripting

Y define los siguientes métodos:

- Public __construct(): Método constructor que crea la instancia de la clase.
- Public loginUser($user, $pswx): Este método se encarga de validar el acceso a la plataforma desde el formulario de acceso y guardar la sesión.

```
public function loginUser($user, $pswx){
    $stmt = $this->pdo->prepare('SELECT ID FROM user WHERE
active = 1 AND username = ? AND password = ?');
    $stmt->execute(array($user, $pswx));
    $fUser = $stmt->fetch(PDO::FETCH_OBJ);
    if ($fUser){
        $_SESSION['user'] = $fUser->ID;
        $_SESSION['token'] = $this->randomToken(20);
        echo '/modules/dashboard/dashboard.php';
    }
    unset($fUser); unset($stmt);
}
```

- Public loginUserByPushCode ($user, $pushCode): Este método se encarga de validar el acceso a la plataforma desde el servicio web.
- Public checkLogin(): Este método se encarga comprobar si la sesión en la plataforma esta activa o no.
- Public logout(): Método para cerrar la sesión en la plataforma.
- Public randomToken($long): Método para obtener el token aleatorio de sesión con una longitud de caracteres establecida mediante el parámetro de entrada $long.

Así mismo la clase define los métodos públicos para el acceso a sus atributos privados.

5.2.4.2. Clase Dashboard

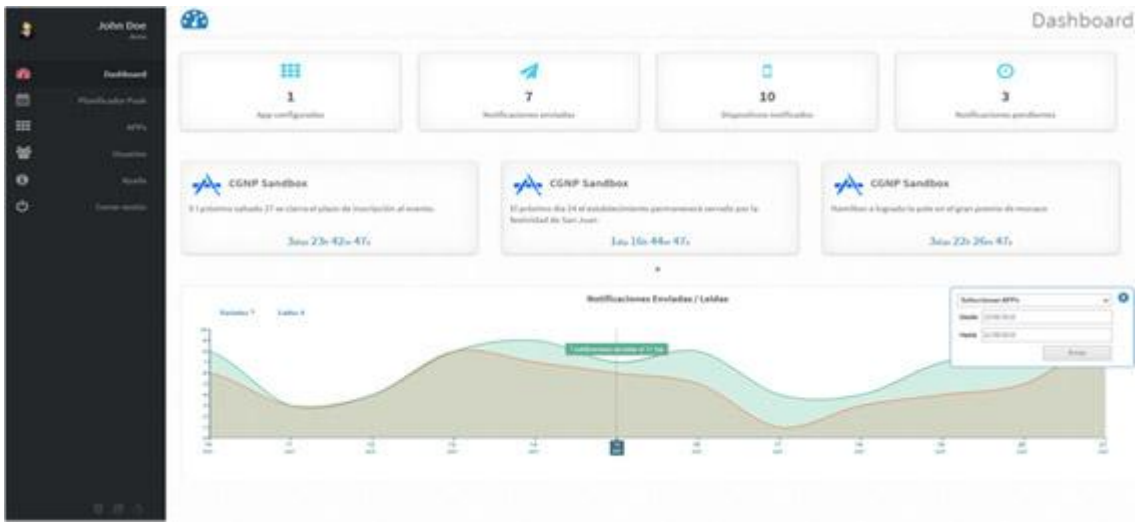


Figura 44. Interfaz del Dashboard

El Dashboard es interfaz que se muestra una vez el usuario accede a la plataforma. En esta interfaz se puede visualizar la información relativa al el estado actual de la plataforma. Existen 3 bloques diferenciados:

El en primer bloque se muestra información básica del uso de la plataforma que es el número de aplicaciones configuradas, el número total de notificaciones enviadas, el número total de dispositivos notificados y el número de notificaciones pendientes de envío. Las notificaciones pendientes de envío son aquellas que están programadas para una fecha posterior a la actual.

En el segundo bloque se muestra la información de las notificaciones pendientes de envío. Se muestra la aplicación a la que pertenecen, el mensaje a notificar y una cuenta atrás que indica el tiempo que resta para que se produzca el envío de la notificación.

El tercer bloque es un gráfico, creado con zingchart que es una librería jQuery para la creación de gráficos vectoriales. En el gráfico se muestra información relativa a la relación notificaciones enviadas / notificaciones leídas para poder obtener un feedback del impacto que las notificaciones están teniendo. La línea verde representa las notificaciones enviadas y la línea naranja las notificaciones leídas. El usuario podrá modificar las fechas y seleccionar la/s aplicación/es de las que desea visualizar los datos estadísticos utilizando el botón habilitado para ello en la parte superior derecha del gráfico.

Para visualizar el grafico se utiliza una representación en formato JSON en la cual se establecen los colores, los textos, los valores de caja eje y las animaciones de carga del gráfico.

La representación en formato JSON es la siguiente:

```
{
    "-background-color":"white",
    "border-top":"3px solid #7e7e7e",
      "border-bottom":"3px solid #7e7e7e",
      "border-right":"3px solid #7e7e7e",
      "border-left":"3px solid #7e7e7e",
      "border-color":"black",
      "border-size":"5",
      "font-family":"Roboto",
      "graphset":[{
            "type":"area",
            "background-color":"#fff",
            "utc":true,
            "title":{
                  "y":"15px",
                  "text":"",
                  "background-color":"none",
                  "font-color":"#666",
                  "font-size":"24px",
                  "height":"25px",
```

```
            "font-family":"Roboto"
        },
        "plotarea":{
            "margin":"100 60 100 60",
            "-y":"125px",
            "-alpha-area":"0.1"
        },
        "labels":[{
            "text":"Enviadas: %plot-1-value",
            "default-value":"",
            "color":"#2980B9",
            "x":80,
            "y":50,
            "width":120,
            "text-align":"left",
            "bold":0,
            "font-family":"Roboto",
            "font-size":"14px",
            "font-weight":"bold"
        },
        {
            "text":"Leidas: %plot-0-value",
            "default-value":"",
            "color":"#2980B9",
            "x":200,
            "y":50,
            "width":120,
            "text-align":"left",
            "bold":0,
            "font-family":"Roboto",
            "font-size":"14px",
            "font-weight":"bold"
        }],
        "scale-x":{
            "label":{
                "-text":"Quantity",
                "font-size":"14px",
                "font-family":"helvetica",
                "font-weight":"normal",
                "offset-x":"10%",
                "font-angle":360,
                "offset-y":"-0px"
            },
            "item":{
                "font-color":"#05636c",
                "font-weight":"normal",
                "font-family":"helvetica",
                "font-size":"12px",
                "offset-x":"-5%"
            },
            "zooming":1,
            "max-labels":12,
            "-min-value":1,
            "labels":["10<br />Jun","11<br />Jun","12<br
/>Jun","13<br />Jun","14<br />Jun","15<br />Jun","16<br />Jun","17<br
/>Jun","18<br />Jun","19<br />Jun","20<br />Jun","21<br />Jun"],
            "max-items":12,
            "-line-color":"#000",
            "-line-style":"solid",
            "line-width":"1px",
            "font-family":"Roboto",
```

```
        "guide":{
                "-line-color":"#000",
                "line-width":"0px",
                "alpha":0.2,
                "line-style":"dashed",
                "font-family":"Roboto",
                "font-size":"14px"
        },
        "tick":{
                "line-width":"2px",
                "-line-color":"#000"
        },
        "minor-ticks":0,
        "minor-tick":{
                "alpha":1,
                "placement":"outer",
                "-line-color":"#000"
        },
        "minor-guide":{
                "visible":false
        },
        "-item":{
                "font-size":"12px",
                "-font-color":"#000",
                "font-family":"Roboto"
        },
        "transform":{
                "type":"date",
                "all":"%M %y",
                "guide":{
                        "visible":false
                },
                "item":{
                        "visible":false
                }
        }
},
"crosshair-x":{
        "plot-label":{
                "visible":false
        }
},
"scale-y":{
        "values":"0:10:1",
        "item":{
                "font-color":"#05636c",
                "font-weight":"normal",
                "font-family":"roboto"
        },
        "font-family":"Roboto",
        "font-size":"12px",
        "guide":{
                "-line-color":"#000",
                "line-width":"0px",
                "alpha":0.2,
                "line-style":"dashed",
                "font-family":"Roboto",
                "font-size":"14px"
        }
},
"plot":{
```

```
                    "line-width":2,
                    "marker":{
                          "size":1,
                          "visible":false
                    },
                    "tooltip":{
                          "-visible":false,
                          "font-family":"Roboto",
                          "font-size":"14px",
                          "text":"%v%t el %data-days",
                          "text-align":"left"
                    }
              },
              "series":[
                    {
                          "values":[6,3,4,8,7,6,5,1,3,4,5,9],
                          "data-days":["10 Jun","11 Jun","12 Jun","13
Jun","14 Jun","15 Jun","16 Jun","17 Jun","18 Jun","19 Jun","20
Jun","21 Jun"],
                          "line-color":"#fc8d62",
                          "aspect":"spline",
                          "background-color":"#fc8d62",
                          "alpha-area":".3",
                          "font-family":"Roboto",
                          "font-size":"14px",
                          "text":" notificaciones leidas",
                          "animation":{
                                "effect":"ANIMATION_SLIDE_BOTTOM"
                          }
                    },
                    {
                          "values":[8,3,4,8,9,7,8,4,4,7,8,9],
                          "data-days":["10 Jun","11 Jun","12 Jun","13
Jun","14 Jun","15 Jun","16 Jun","17 Jun","18 Jun","19 Jun","20
Jun","21 Jun"],
                          "line-color":"#66c2a5",
                          "background-color":"#66c2a5",
                          "alpha-area":".3",
                          "text":" notificaciones enviadas",
                          "aspect":"spline",
                          "font-family":"Roboto",
                          "font-size":"14px",
                          "animation":{
                                "effect":"ANIMATION_SLIDE_BOTTOM"
                          }
                    }]
              }]
}
```

La clase Dashboard se encarga de la lógica de negocio que genera el contenido para la interfaz. Cuenta con los siguientes atributos:

- `Private` PDO: Instancia a la clase de acceso a la base de datos.
- `Private` Login: Instancia de la clase Login para comprobar si el acceso es legítimo.

- `Private` LastNDays: Indica la longitud del eje x del gráfico vectorial que representa a los días.
- `Private` DayFrom: Representa al día inicial desde el cual se quiere visualizar el gráfico vectorial.
- `Private` DayTo: Representa al día final desde el cual se quiere visualizar el gráfico vectorial.

Y define los siguientes métodos:

- `Public` __construct(): Método constructor que crea la instancia de la clase.
- `Public` loadInfoBlock(): Este método general el contenido HTML para el bloque de información básica del sistema de la interfaz.
- `Public` loadUndeliveryNotificationList(): Este método genera el contenido HTML que representa al listado de notificaciones Push que se encuentran pendientes de envío.
- `Public` loadSendChart(): Genera la representación JSON del grafico vectorial que muestra la relación de notificaciones enviadas y leídas filtrados por los valores de los atributos de la clase.

Así mismo la clase implementa los métodos para el acceso a sus atributos privados.

5.2.4.3. Clase User

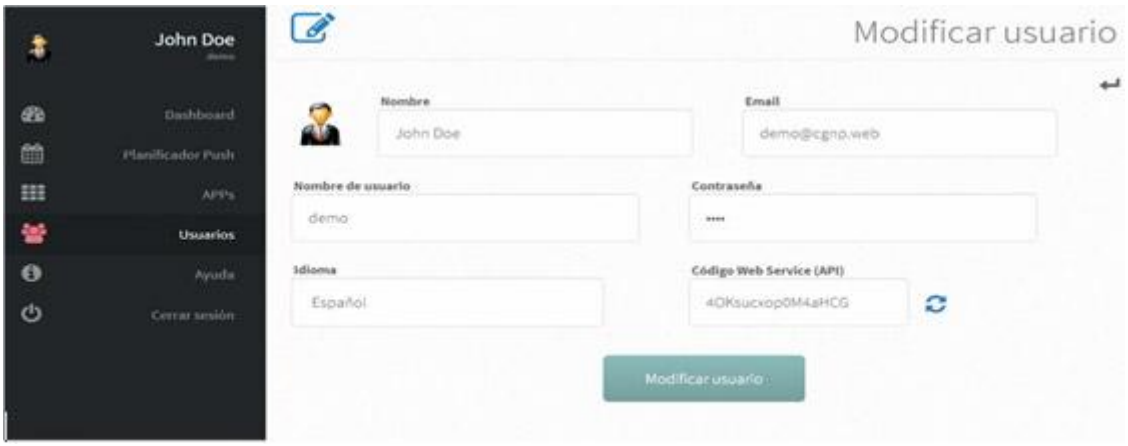


Figura 45. Interfaz de edición de usuarios

La clase User se encarga de la lógica de negocio de la gestión de los usuarios con acceso a la plataforma. Permite gestionar el alta, la edición y eliminación de los usuarios así como generar el HTML para presentar el listado de usuarios existentes.

Cuenta con los siguientes atributos:

- `Private` PDO: Instancia a la clase de acceso a la base de datos.
- `Private` ID: Identificador del usuario.
- `Private` Admin: Valor que indica si se trata de un usuario administrador o de un usuario estándar.
- `Private` Name: Nombre real del usuario.
- `Private` Email: Cuenta de correo electrónico del usuario.
- `Private` UserName: Nombre de acceso a la plataforma.
- `Private` Password: Contraseña de acceso a la plataforma.
- `Private` Avatar: Imagen asociada al usuario.
- `Private` Lang: Idioma del usuario.
- `Private` pushID: Token de acceso del usuario para la utilización del servicio web.

Y define los siguientes métodos:

- `Public` __construct(): Método constructor que crea la instancia de la clase.
- `Public` loadUserList (): Este método genera el contenido HTML que representa al listado de usuarios que están dados de alta en la plataforma.
- `Public` loadUser(): Carga los datos del usuario en el formulario de edición.
- `Public` create(): Crea un nuevo usuario con los valores introducidos en el formulario de alta.
- `Public` update(): Actualiza los datos de un usuario con los valores introducidos en el formulario de edición.
- `Public` delete(): Borra de la plataforma al usuario seleccionado.

Así mismo la clase implementa los métodos para el acceso a sus atributos privados.

Desde el formulario de alta y edición de usuarios se permite la asignación de un avatar, que es la representación gráfica que se asocia al usuario. Para implementar esta funcionalidad y poder transferir la imagen, del sistema de ficheros del dispositivo usado al sistema de ficheros de la plataforma, se emplea evento jQuery. Mediante este evento se abre el selector de archivos local para que el usuario seleccione la imagen y una vez seleccionada se transfiere al servidor mediante AJAX. Este sistema de envío permite realizar la transferencia sin la necesidad de tener que recargar el formulario HTML.

El siguiente fragmento de código representa el código jQuery que permite realizar el evento:

```
$('#frmManageUser').on('click', '#avatarUpload', function(e){
        e.preventDefault();
```

```
        $('#uploadifive-hAvatarUpload
input').not('#hAvatarUpload').filter( ":last"
).attr('accept','image/*');
        $('#uploadifive-hAvatarUpload
input').not('#hAvatarUpload').filter( ":last" ).trigger("click");
    });

    $('#hAvatarUpload').uploadifive({
        'uploadScript'      : '/modules/users/avatar-upload.php',
        'multi'             : false,
        'auto'              : true,
        'fileType'          : 'image',
        'fileSizeLimit'     : 256,
        'formData'          : {'uid' : $('#avatarUpload').attr('data-
uid'), 'token' : $('#avatarUpload').attr('data-token')},
        'onUploadComplete'  : function(file, data) {
            var response = jQuery.parseJSON(data);
            if (response.error == 0){
                $('#avatarUpload img').attr('src', response.bImg);
                if (response.sImg != '' ) { $('#logo img').attr('src',
response.sImg); }
            } else {
                alert(response.errorMsg);
            }
        },
        'onError'           : function(errorType) {
            alert(errorType);
        }
    });
```

5.2.4.4. Clase MySQL

La clase MySQL se encarga de la comunicación de la plataforma con el sistema de gestión de bases de datos. Para ellos se ha empleado la extensión Objetos de Datos PHP (PDO por sus siglas inglesas). Es la interfaz de acceso a bases de datos más novedosa y ligera disponible para acceder a MySQL desde PHP.

Debido que el acceso a datos es uno de los puntos críticos en el rendimiento de las aplicaciones web, para su implementación se ha seguido un patrón de diseño singleton o de instancia única. De esto modo evitamos el uso de múltiples conexiones con el consiguiente ahorro de recursos del sistema.

Como la plataforma no cuenta con programación multihilo no ha sido necesaria la implementación de un sistema de semáforos de exclusión mutua, para evitar la creación de múltiples instancias por diferentes procesos que se estén ejecutando a la vez.

El patrón de conexión se implementa del siguiente modo:

```
private function __construct() {
    $this->pdo = new PDO();
```

```
    $this->pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $this->pdo->query('SET NAMES utf8');
}

public function getDBInstance(){
    if (is_null($this->pdo )){
        $this->__construct();
    } else {
        return $this->pdo;
    }
}
```

5.2.4.5. Clase Lang

La clase Lang se encarga de gestionar la localización del contenido HTML estático para que la plataforma sea multiidioma. Aunque no entra dentro del objetivo del proyecto el realizar las traducciones del texto, siendo el idioma principal de la aplicación el español, se ha provisto de un mecanismo mediante el cual se puede localizar fácilmente la plataforma a cualquier idioma.

La implementación se ha basado en la creación de ficheros XML que se encuentran en ruta "*/modules/lang/xx"* de la plataforma. Donde *xx* representa al código identificador universal de cada idioma, también denominado código ISO. Para la traducción de la plataforma a un nuevo idioma basta con duplicar el contenido de un idioma previamente disponible a la nueva carpeta nombrada con el código ISO del idioma en cuestión y traducir el contenido de las etiquetas de los archivos XML.

La estructura XML de los archivos de localización es la siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<LANG>
    <TEXT ID="identificador 1">contenido 1</TEXT>
    <TEXT ID="identificador 2">contenido 1</TEXT>
    <TEXT ID="identificador 3">contenido 1</TEXT>

    <TEXT ID="identificador n">contenido n</TEXT>
</LANG>
```

5.2.4.6. Clase APPs

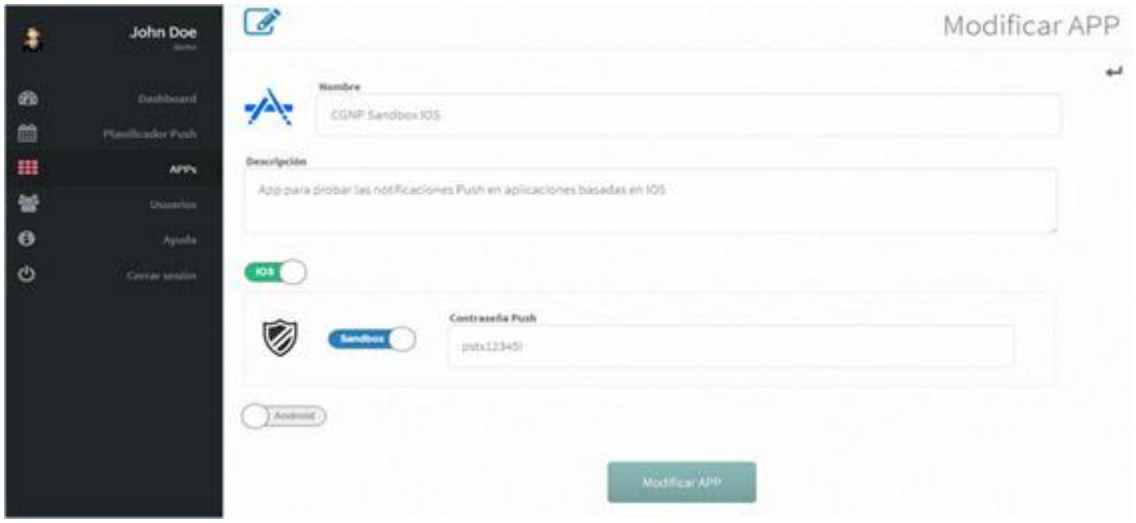


Figura 46. Interfaz de edición de APPs

La clase APPs se encarga de la lógica de negocio de la gestión de las aplicaciones que serán las destinatarias de las notificaciones Push enviadas desde la plataforma. Permite gestionar el alta, la edición y eliminación de las aplicaciones así como generar el HTML para presentar el listado de aplicaciones configuradas.

La configuración de las aplicaciones varía en función de si se trata de una aplicación IOS o Android. Además de los datos comunes que son el nombre y descripción de la aplicación será necesario configurar los datos de acceso a los servicios APSN de Apple y GCM de Google dependiente de si la aplicación es IOS o Android respectivamente.

Para poder configurar una aplicación IOS, el usuario deberá indica, mediante el selector provisto en la interfaz gráfica, el entorno en el que se encuentra la aplicación, pudiendo estar en desarrollo (sandbox) o producción. Además será necesario transferir el certificado de envío de notificaciones Push y su contraseña. El certificado es un archivo que el desarrollado de la aplicación debe generar desde su panel de desarrollado de aplicaciones para dispositivos IOS accesible a través del portal de internet https://developer.apple.com. Una vez rellenados estos campos la aplicación será candidata para realizar envíos de notificaciones Push.

La configuración de aplicaciones Android es más sencillo, siendo únicamente necesario introducir el "API Key", que es el código de acceso al servicio GCM para la aplicación Android. Este código lo obtiene el desarrollado de la aplicación en el portal de desarrolladores de Google accesible a través del portal de internet https://developers.google.com. Una vez rellenado este campo la aplicación será candidata para el envío de notificaciones Push.

Para dar más versatilidad a la plataforma, una aplicación que tenga versiones tanto para IOS como para Android podrá configurarse de modo conjunto en un solo registro o por separado.

La clase cuenta con los siguientes atributos:

- `Private` PDO: Instancia a la clase de acceso a la base de datos.
- `Private` ID: Identificador de la aplicación.
- `Private` Owner: Identificador del usuario propietario de la aplicación.
- `Private` Name: Nombre de la aplicación.
- `Private` Description: Descripción de la aplicación.
- `Private` IOS: Indica si la aplicación está disponible para IOS
- `Private` IOSSandbox: Indica si la aplicación está en modo de desarrollo o producción.
- `Private` IOSPem: Nombre del certificado de la aplicación IOS.
- `Private` IOSPass: Contraseña del certificado.
- `Private` Android: Indica si la aplicación está disponible para Android.
- `Private` AndroidApiKey: Token de acceso a la plataforma GCM para aplicaciones Android.

Y define los siguientes métodos:

- `Public` __construct(): Método constructor que crea la instancia de la clase.
- `Public` loadAppList (): Este método genera el contenido HTML que representa al listado de aplicaciones que están configuradas en la plataforma.
- `Public` loadApp(): Carga los datos de configuración de una aplicación en el formulario de edición.
- `Public` create(): Crea una nueva aplicación con los valores introducidos en el formulario de alta.
- `Public` update(): Actualiza los datos de una aplicación con los valores introducidos en el formulario de edición.
- `Public` delete(): Borra de la plataforma la aplicación seleccionada.
- `Public` checkAppOwner(): Comprueba que la aplicación pertenece al usuario que esta logeado en la plataforma.
- `Public` registerDevice(): Registra un dispositivo destinatario de las notificaciones Push dirigidas a la aplicación. Este método se invoca desde el servicio web.

Así mismo la clase implementa los métodos para el acceso a sus atributos privados.

5.2.4.7. Clase Push

Figura 47. Listado de notificaciones Push

La clase Push se encarga de la lógica de negocio del envío de notificaciones Push a las aplicaciones configuradas en la plataforma. Permite gestionar el alta, la edición, eliminación y envío de las notificaciones así como generar el HTML para presentar el listado de notificaciones o cola de envío.

En el listado de notificaciones Push se representa el estado de cada una de ellas mediante el uso de un código de colores. Los 3 posibles estados son:

- Color **gris oscuro**: La notificación ha sido enviada.
- Color **naranja**: La notificación esta programa para enviarse en un plazo máximo de 24 horas.
- Color **azul**: La notificación está programada para enviarse en un plazo mayor a 24 horas.

Este listado representa la cola de envíos que se ordena respecto a la fecha en la que se ha programado él envió.

El proceso de envío de una notificación consta de dos partes: alta y envío. El alta corresponde a rellenar el formulario de alta de una nueva notificación indicando la aplicación destinataria, la fecha y hora del envío, la insignia numérica de la notificación, su sonido, el mensaje a notificar y si se trata de una notificación para actualizar contenido de la aplicación destinataria. Una vez rellenados los campos y pulsado el botón de aceptar del formulario, la notificación se añade a la cola de envíos.

El envío se realiza mediante proceso automático que cada minuto selecciona de la cola de envíos aquellas notificaciones cuya fecha de envío sea igual o anterior a la fecha del proceso. Si hay notificaciones que cumplan el requisito, genera la petición de comunicación con los servicios APNS y/o GCM dependiendo la plataforma de la

aplicación a la que va dirigida la notificación y envía la petición para que esta se procese. Si el proceso es correcto los dispositivos destinatarios recibirán la notificación.

La estructura que representa a una petición al servicio APNS de Apple se genera mediante el siguiente código:

```
$msg = chr(0)
       . pack('n', 32)
       . pack('H*', $deviceToken)
       . pack('n', strlen($payload))
       . $payload;
```

Siendo el identificador del dispositivo (deviceToken) un código alfanumérico de 64 caracteres y la carga útil de la notificación (payload) la siguiente estructura JSON:

```
{
  "aps":
  {
      "alert": "El proceso de selección comienza el día 27",
      "badge": 1,
      "sound": "default",
      "content-available": 0
  }
}
```

Donde el atributo “alert” contiene el mensaje a notificar, el atributo “badge” la insignia número a mostrar en el icono de la aplicación destinataria, “sound” el sonido que el dispositivo notificado reproduce cuando se entrega la notificación y el flag “content-available” que indica la existencia o no de contenido para sincronizar.

La estructura que representa la solicitud de envío una notificación con mensaje en un dispositivo Android es la siguiente:

```
{
  "to": "04c8edfb758f55a49734f349342...6a37c93",
  "data":
  {
      "alert": "El proceso de selección comienza el día 27",
  }
}
```

Donde el atributo “to” indica el identificador del dispositivo a notificar y el atributo “alert” el mensaje.

Cuando se trata de una notificación para sincronizar contenido la estructura es:

```
{
  "to": "04c8edfb758f55a49734f349342..6a37c93"
}
```

La clase Push cuenta con los siguientes atributos:

- `Private` PDO: Instancia a la clase de acceso a la base de datos.
- `Private` ID: Identificador de la notificación.
- `Private` App: Identificador de la aplicación a la que va dirigida la notificación.
- `Private` Messagge: Mensaje a notificar.
- `Private` Badge: Insignia a mostrar en la notificación.
- `Private` Sound: Identificador del audio que emite la aplicación al ser notificada.
- `Private` ContentAvailable: Indica si se trata de una notificación de nuevo contenido disposible.
- `Private` SendDate: Fecha de envío de la notificación.

Y define los siguientes métodos:

- `Public` __construct(): Método constructor que crea la instancia de la clase.
- `Public` loadPushList (): Este método genera el contenido HTML que representa la lista de notificaciones registradas en la plataforma.
- `Public` loadPush(): Carga los datos de una notificación en el formulario de edición.
- `Public` create(): Crea una nueva notificación con los valores introducidos en el formulario de alta.
- `Public` update(): Actualiza los datos de una notificación con los valores introducidos en el formulario de edición. Solo se permite actualizar una notificación cuando su estado sea "no enviada".
- `Public` delete(): Borra de la plataforma la notificación seleccionada.
- `Public` setPushAsReceived($id): Función que establece como leída a la notificación cuya identificación corresponda con el parámetro $id. Este método se invoca desde el servicio web.

Así mismo la clase implementa los métodos para el acceso a sus atributos privados.

5.2.5. Servicio web

El servicio web implementado expone parte de la lógica de negocio de la plataforma implementada para que sea accesible desde otros aplicativos. El servicio se ha implementado siguiendo la arquitectura REST, Representational State Transfer en la cual para invocar los métodos del servicio se realiza a través de peticiones HTTP.

Para realizar estas peticiones se utilizan URIs virtuales que representan el estado del método del servicio al que se quiere acceder que a su vez representa al recurso con el que se interactúa. La estructura de las peticiones es la siguiente:

/push-api/identificador-del-recurso/usuario/codigo-web-sevice/parámetro1/parámetro2/…/parámetro n

Donde:

- */push-api/* es la raíz en la cual se encuentra publicado el servicio.
- El identificador del recurso, es el identificador del método que se desea ejecutar.
- Usuario es el nombre de usuario solicitante de la petición.
- El código web service es el token de acceso al servicio del usuario solicitante de la petición.
- Los parámetros son los valores asignables al recurso.

5.2.6. Base de datos

La base de datos se emplea para almacenar todos los valores necesarios para el correcto funcionamiento de las clases descriptas anteriormente. Está gestionada por el motor MySQL y se accede a la misma utilizando el protocolo PDO mediante las credenciales de acceso.

El modelo de la base de datos es el siguiente:

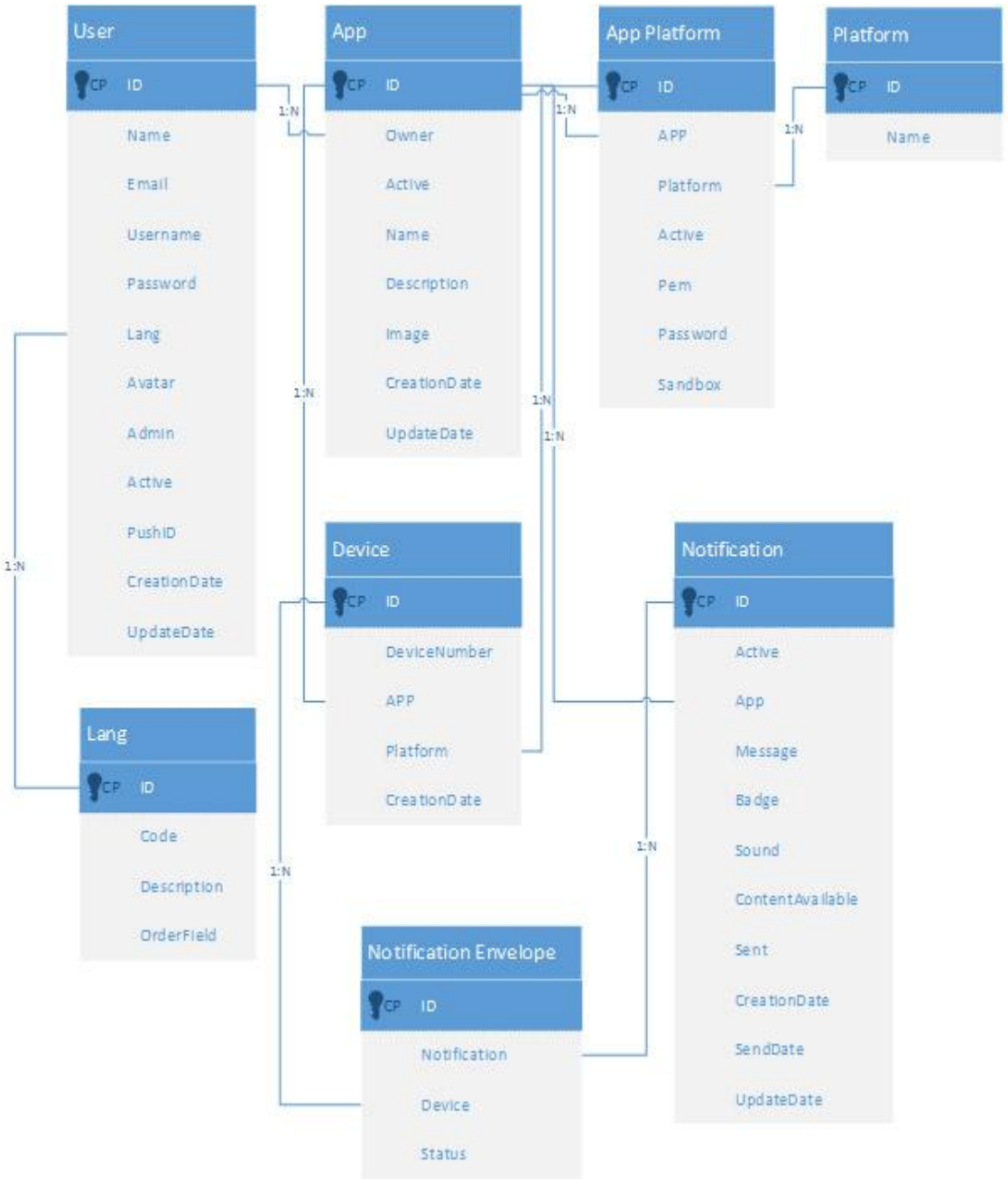


Figura 48. Modelo de la base de datos

User: Almacena los datos de los usuarios dados de alta en la plataforma y alimenta a la sección de usuarios.

Lang: Almacena los idiomas en los que está disponible la plataforma.

App: Almacena los datos básicos de las aplicaciones configuradas en la plataforma. Alimenta la sección de gestión de aplicaciones y selector de aplicación en el alta y edición de las notificaciones Push.

Platform: Almacena los sistemas operativos cuyas aplicaciones pueden ser configuradas en la plataforma. En nuestro caso IOS y Android.

App Platform: Almacena los datos específicos relacionados con la aplicación y la plataforma a la que pertenece.

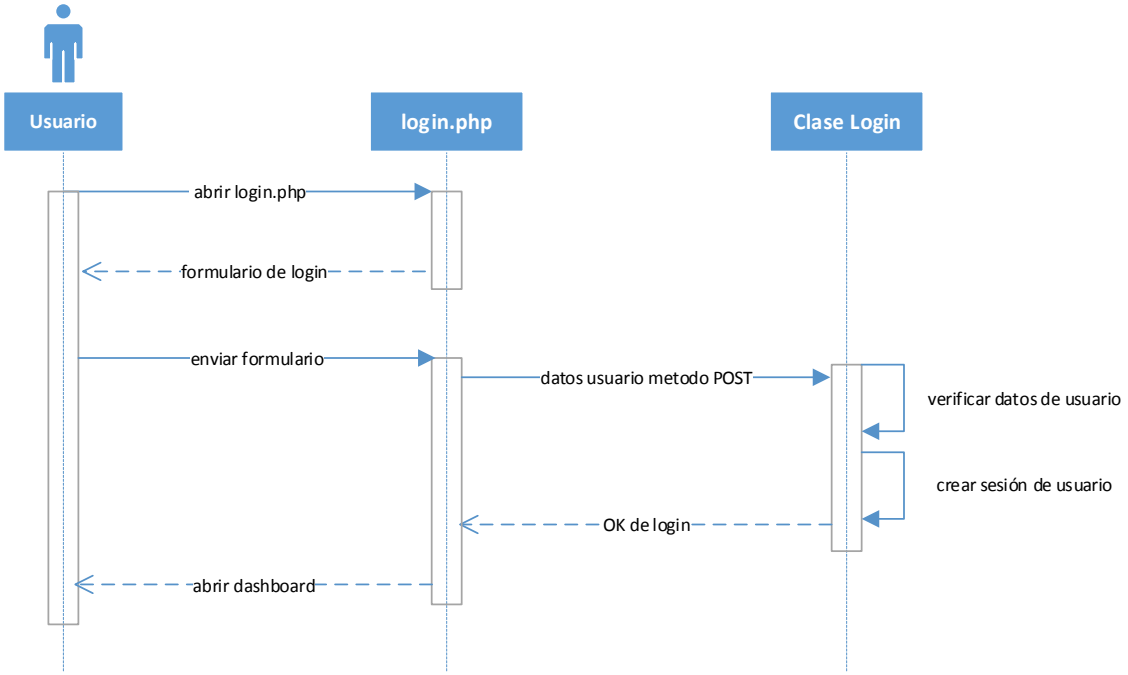
Device: Almacena los identificadores de los dispositivos destinatarios y los relaciona con su aplicación.

Notification: Almacena los datos relativos a las notificaciones Push registradas en la plataforma. Alimentar la sección del "Planificador Push" y el proceso de envío de notificaciones.

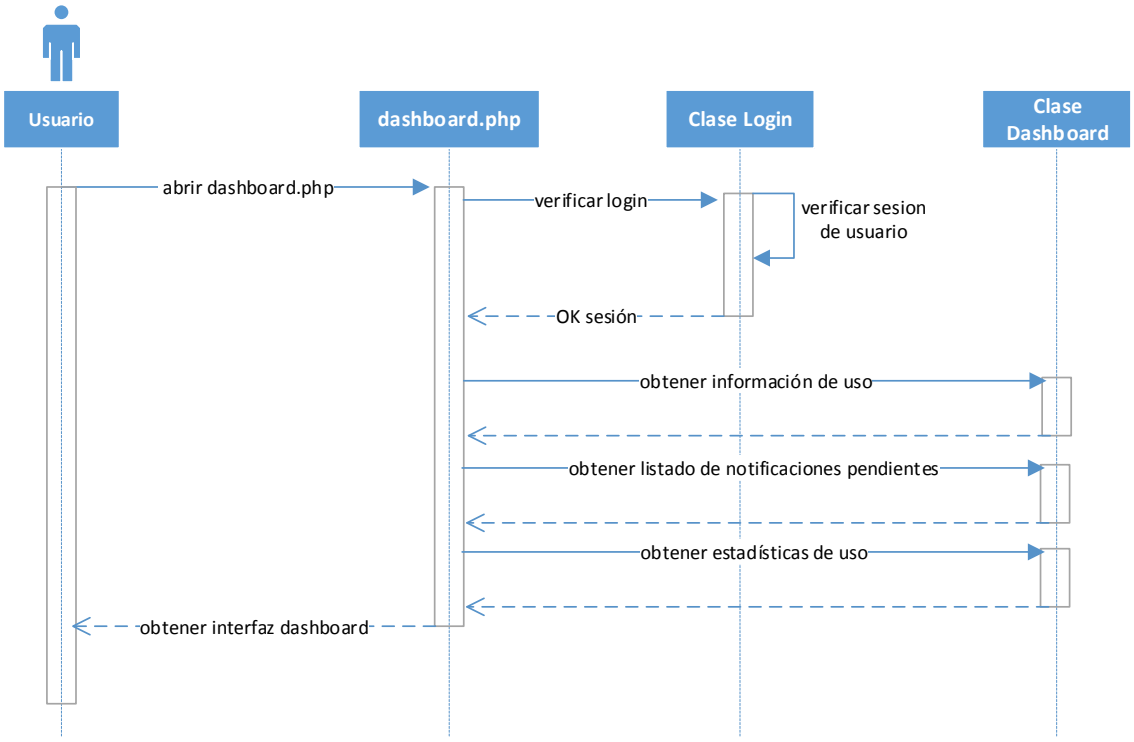
Notification envelope: Almacena el estado de cada notificación a cada dispositivo.

5.2.7. Diagramas de secuencia

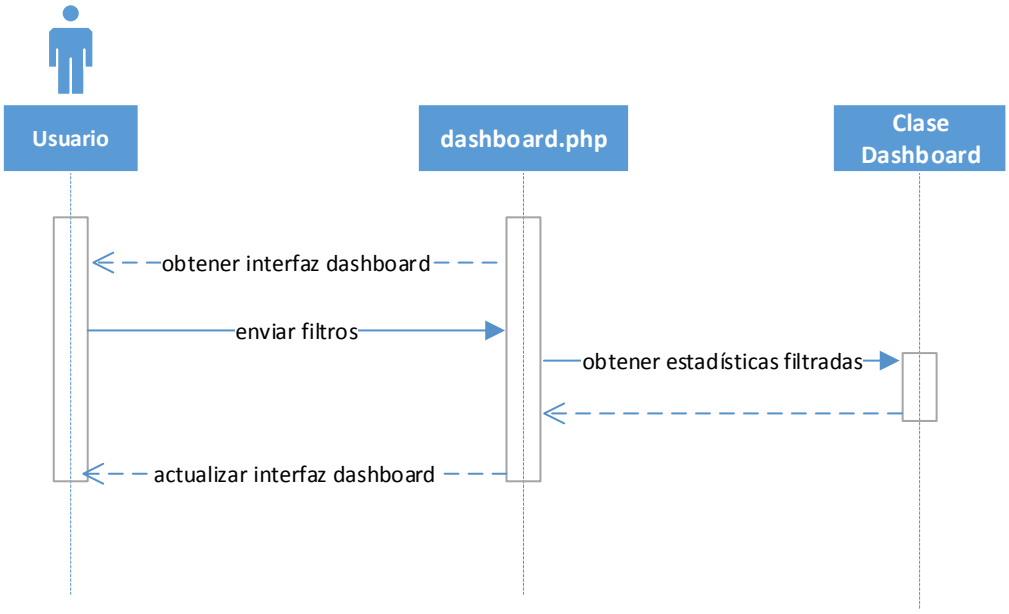
5.2.7.1. Caso de uso Login



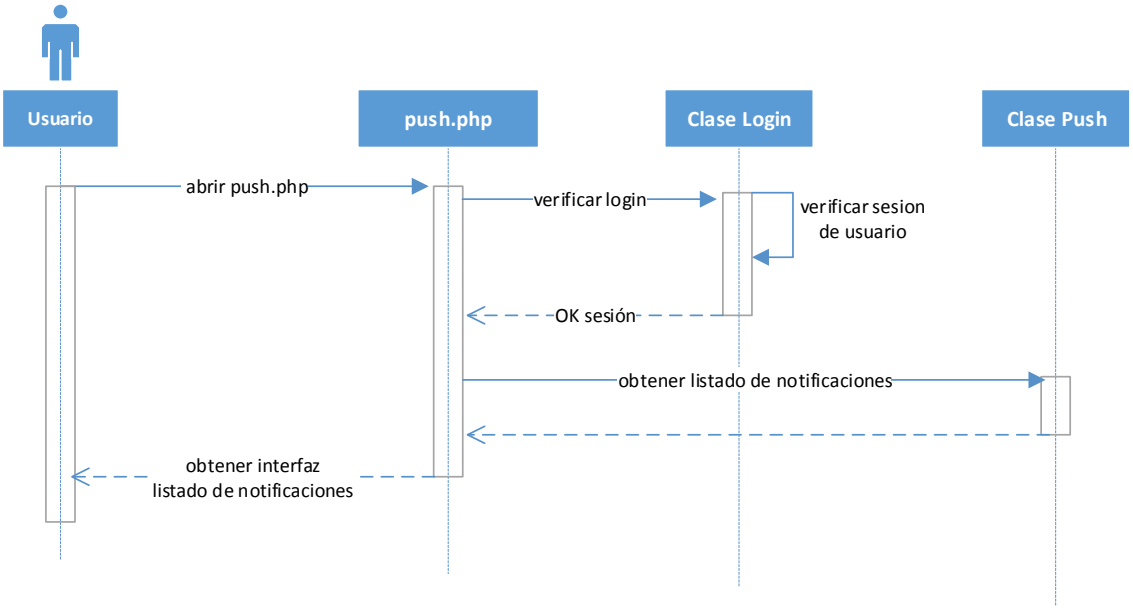
5.2.7.2. Caso de uso Visualizar estadísticas



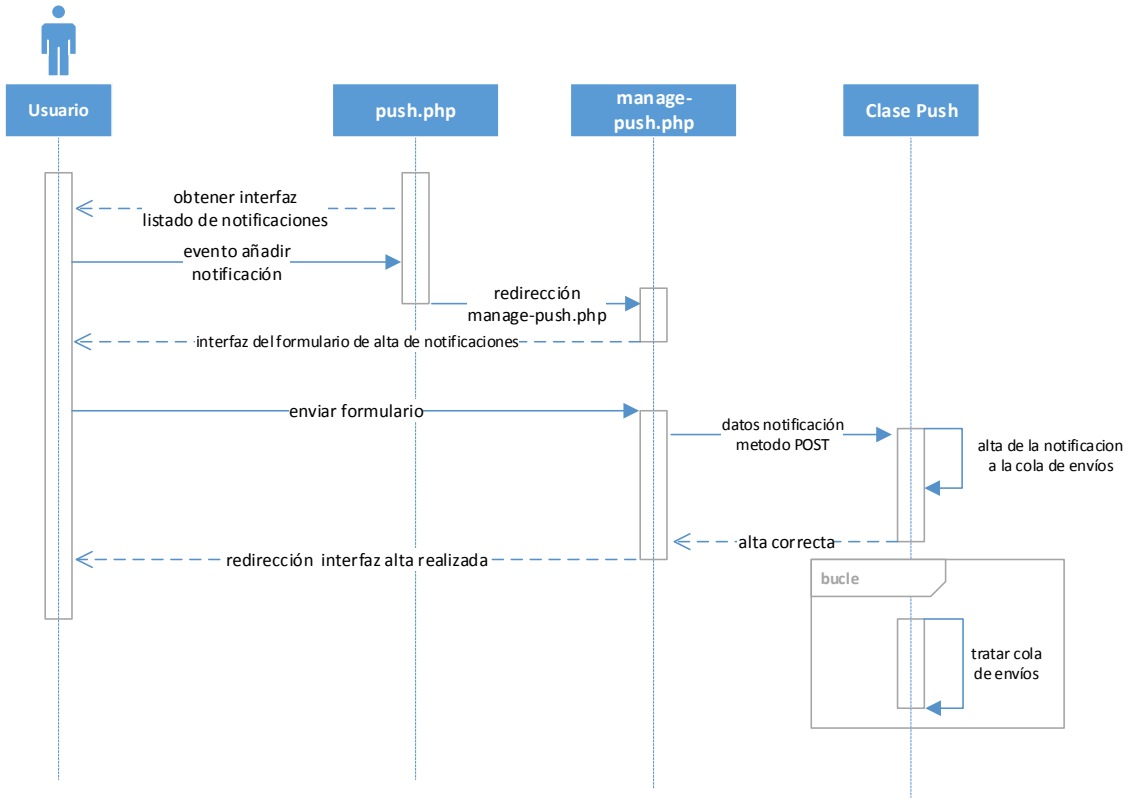
5.2.7.3. Caso de uso Filtrar estadísticas



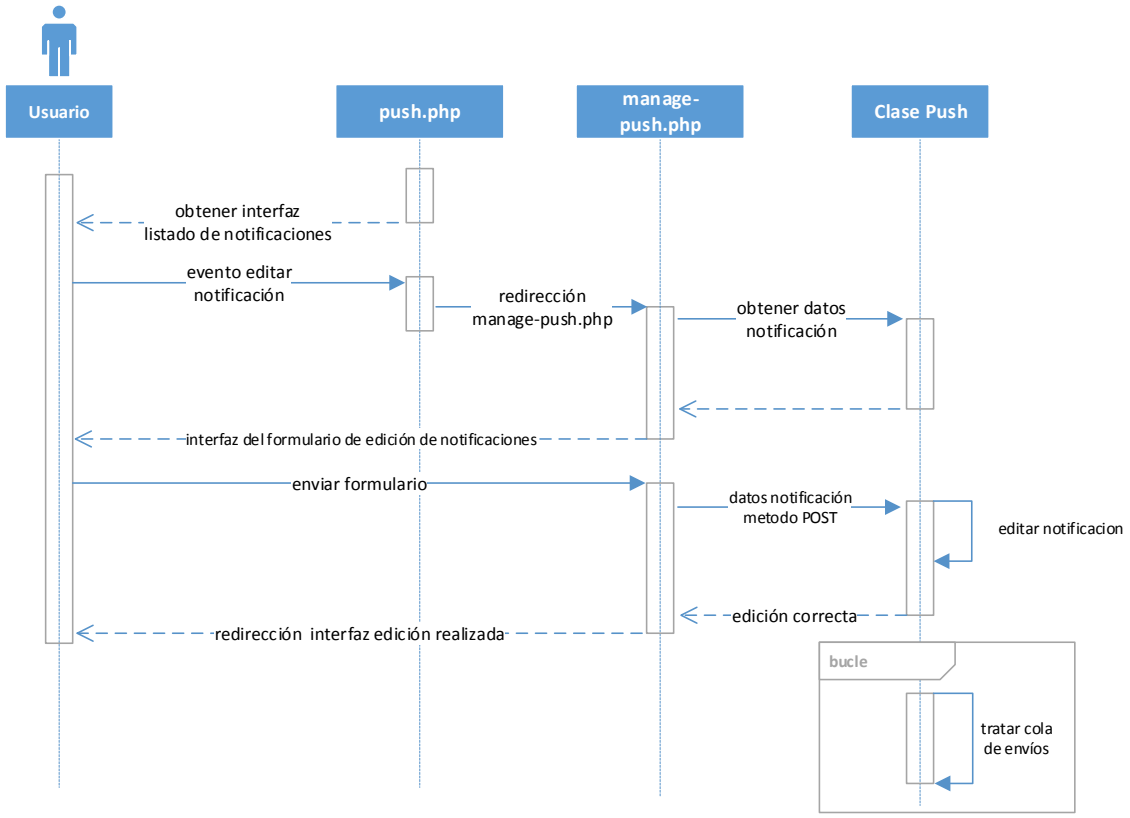
5.2.7.4. Caso de uso Listar notificaciones



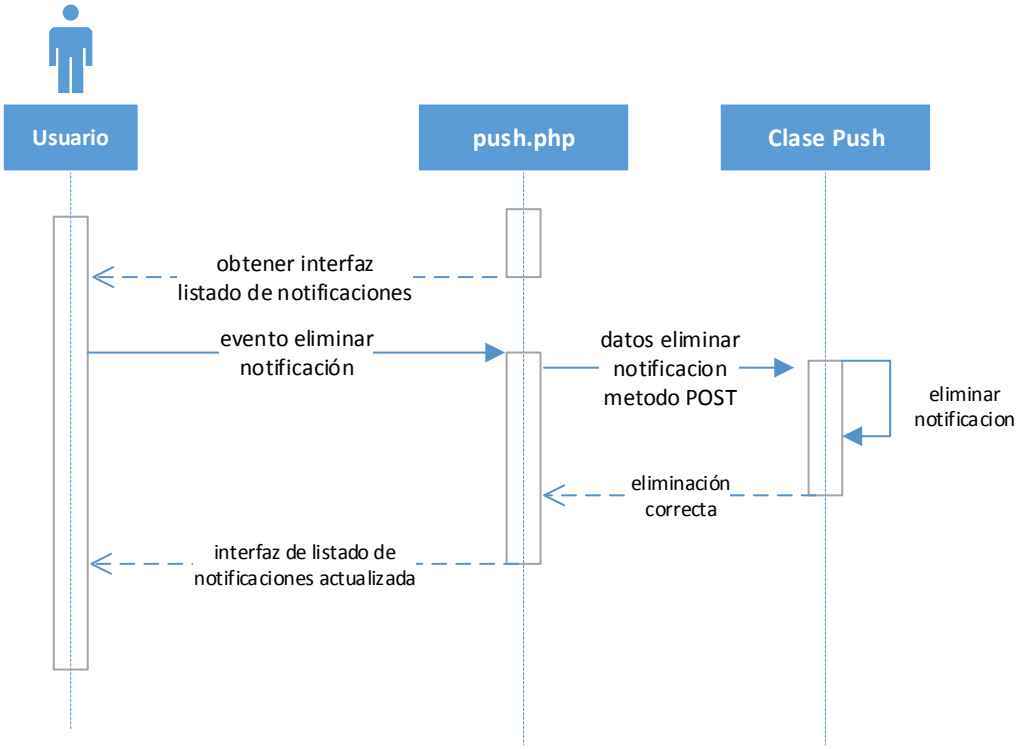
5.2.7.5. Caso de uso Añadir notificación



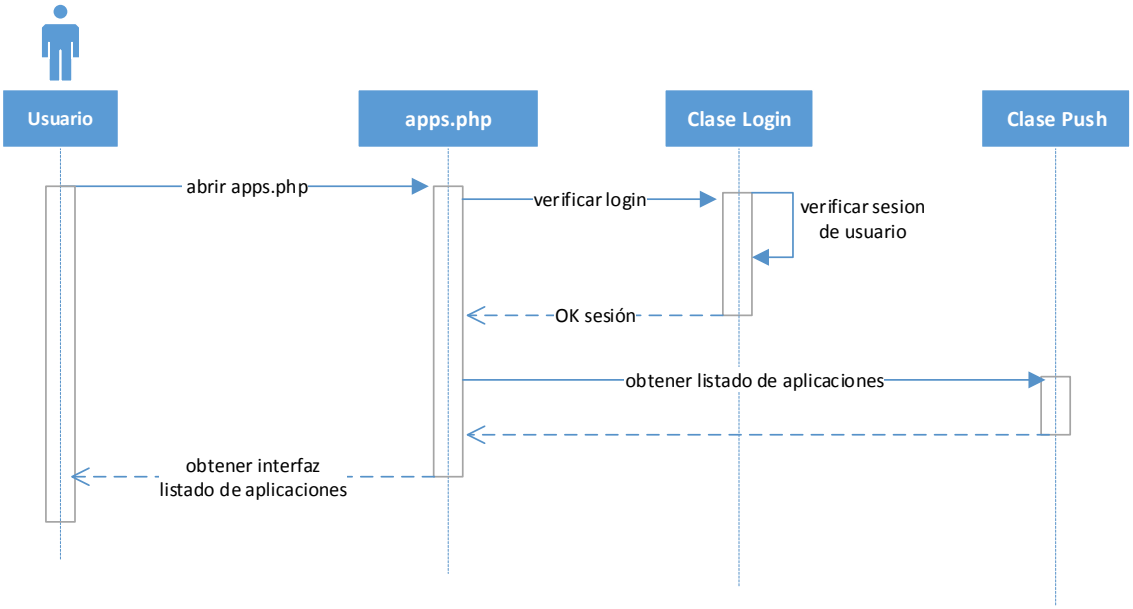
5.2.7.6. Caso de uso Editar notificación



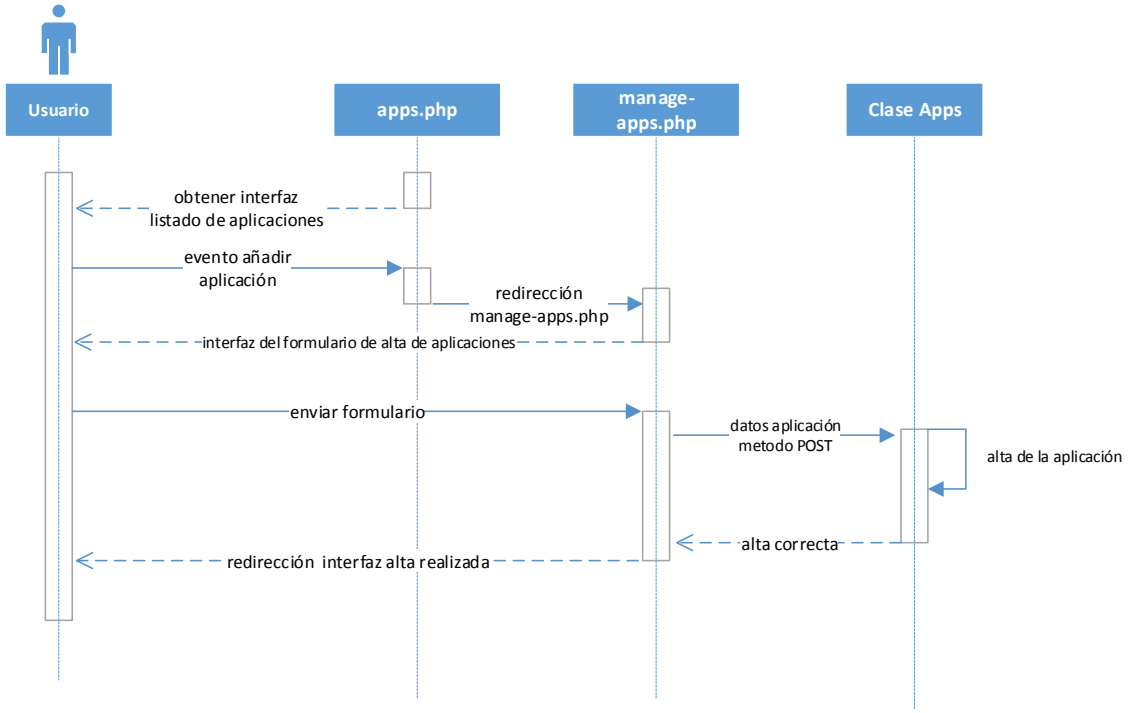
5.2.7.7. Caso de uso Eliminar notificación



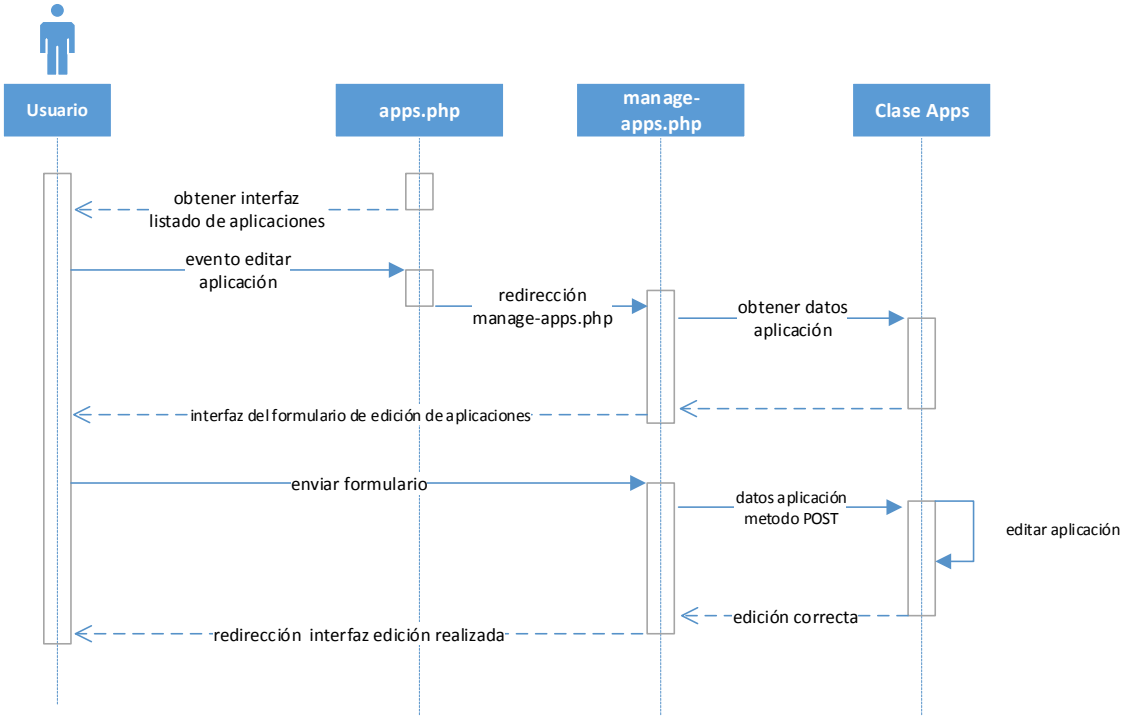
5.2.7.8. Caso de uso Listar aplicaciones



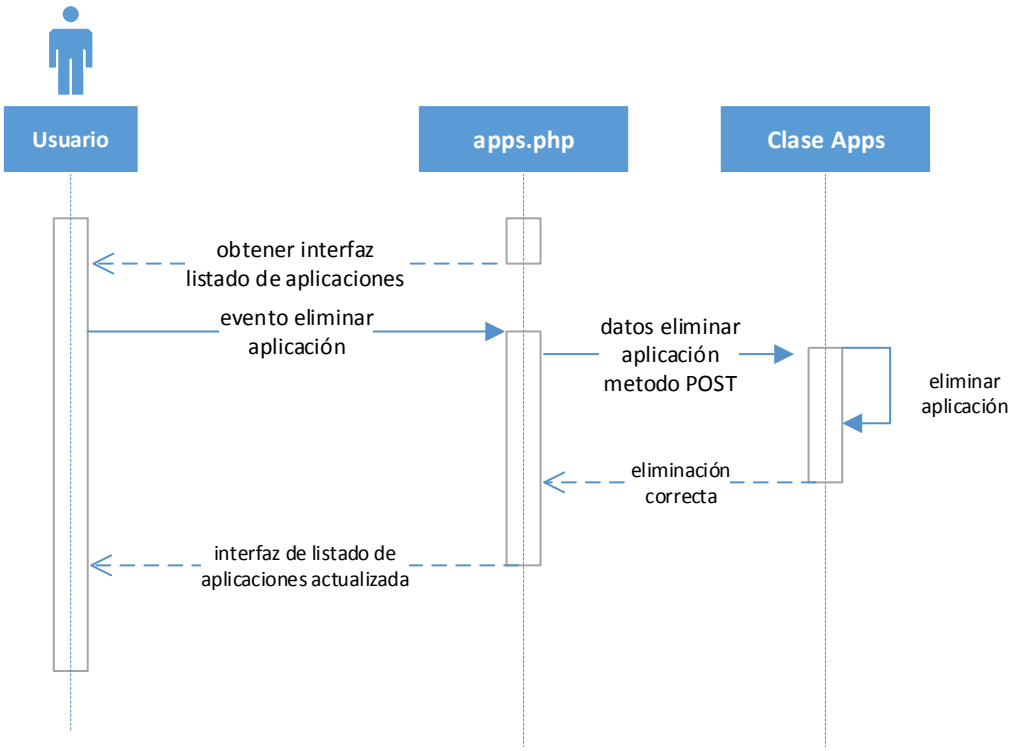
5.2.7.9. Caso de uso Añadir aplicación



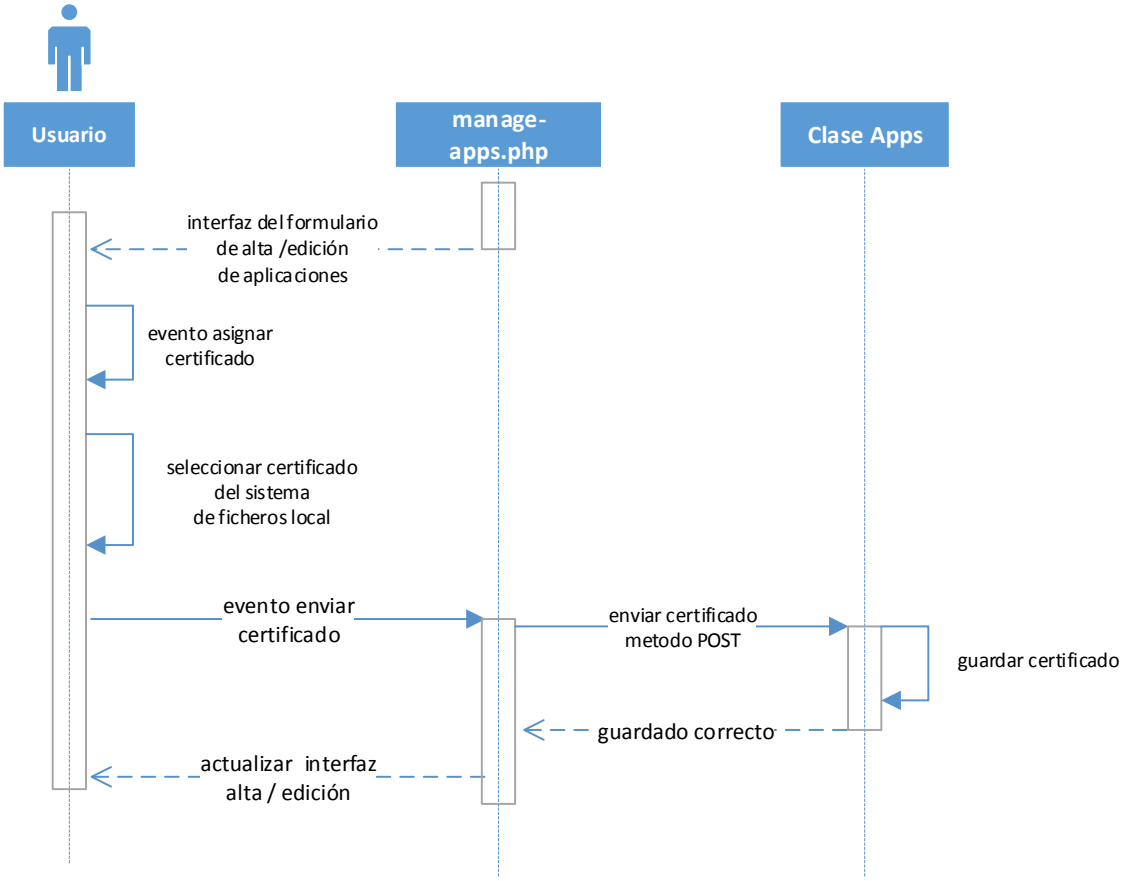
5.2.7.10. Caso de uso Editar aplicación



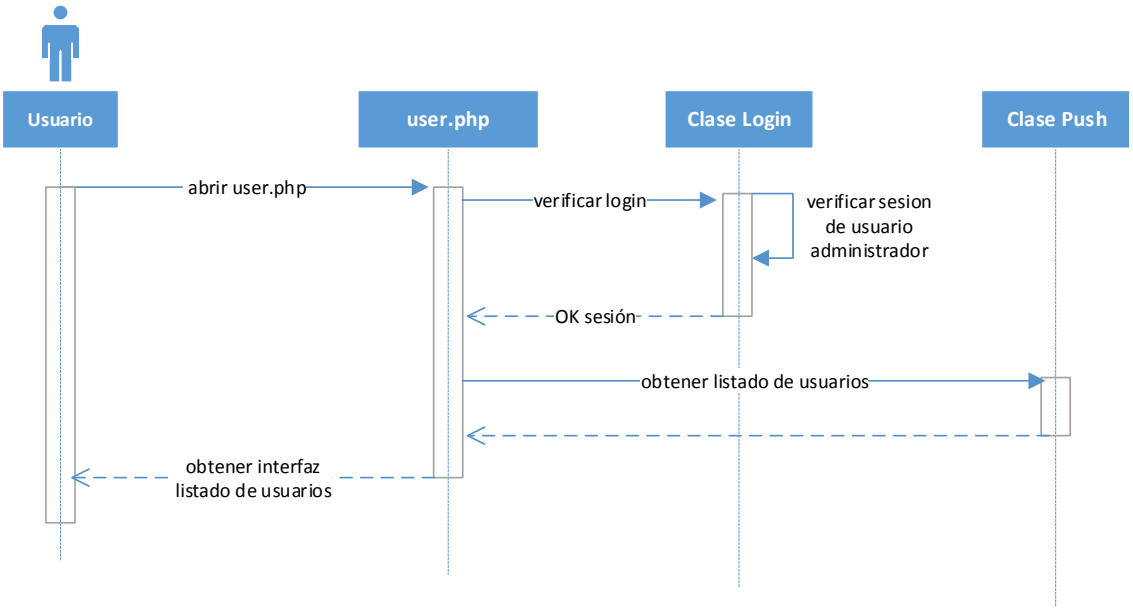
5.2.7.11. Caso de uso Eliminar aplicación



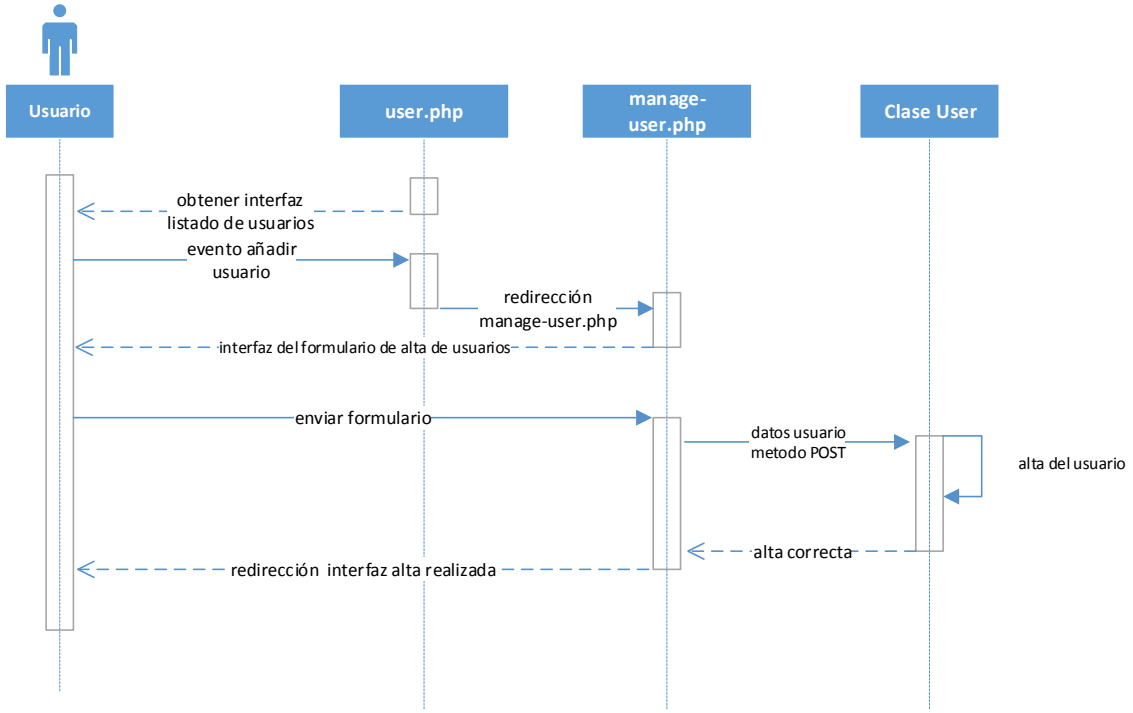
5.2.7.12. Caso de uso Asignar certificado



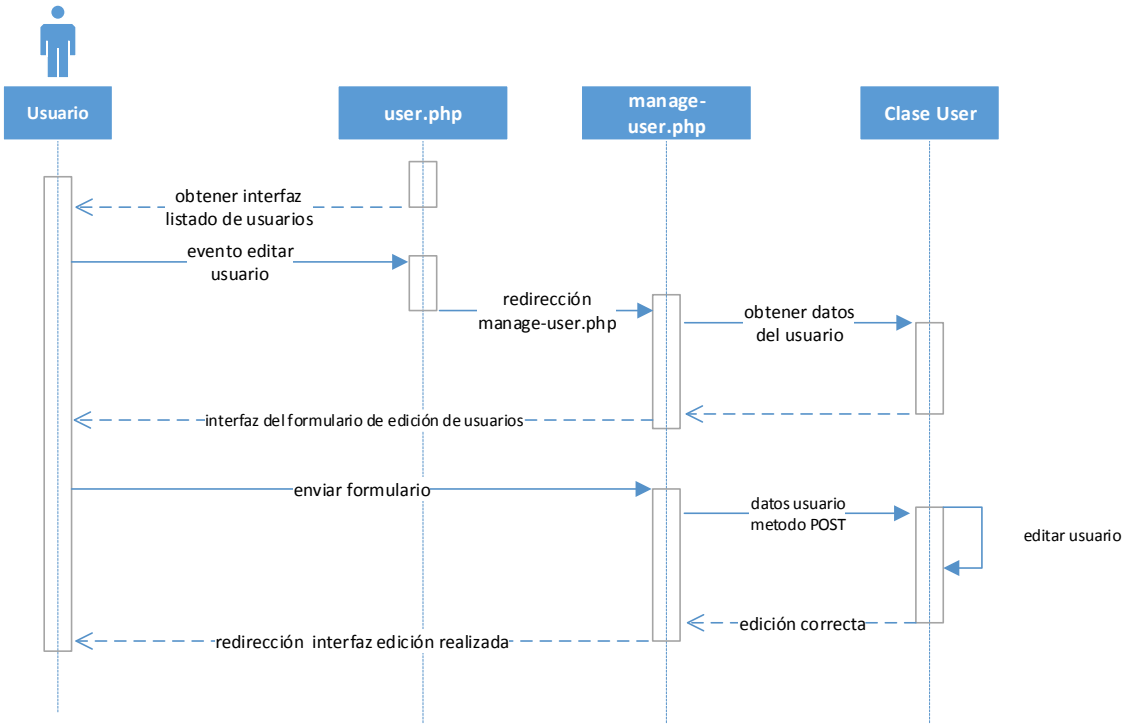
5.2.7.13. Caso de uso Listar usuarios



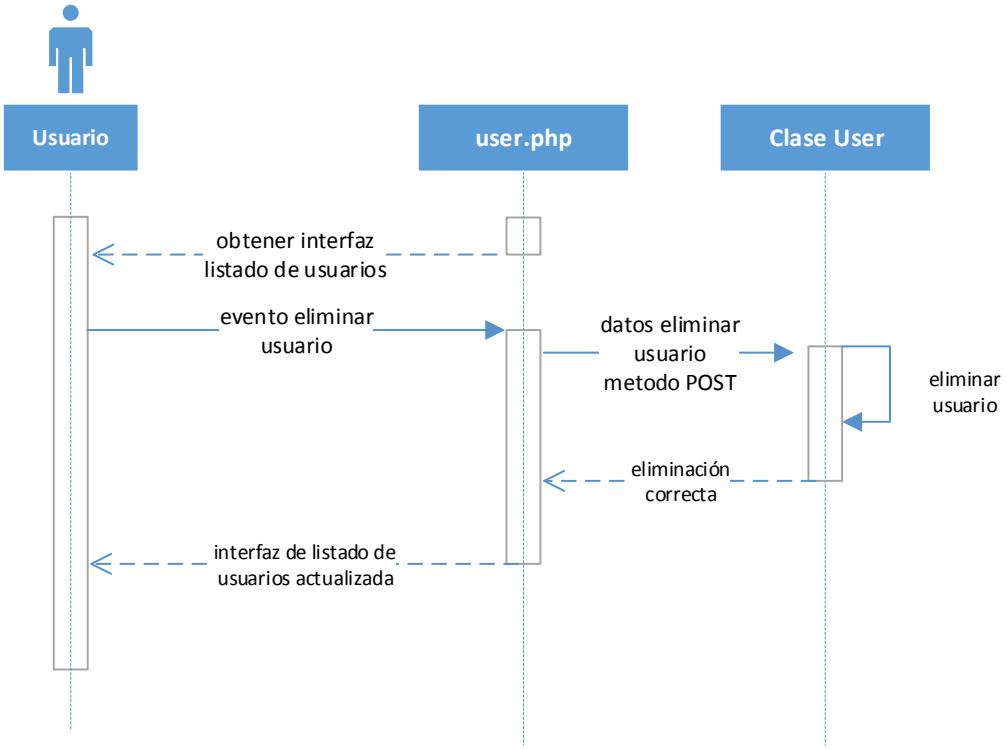
5.2.7.14. Caso de uso Añadir usuario



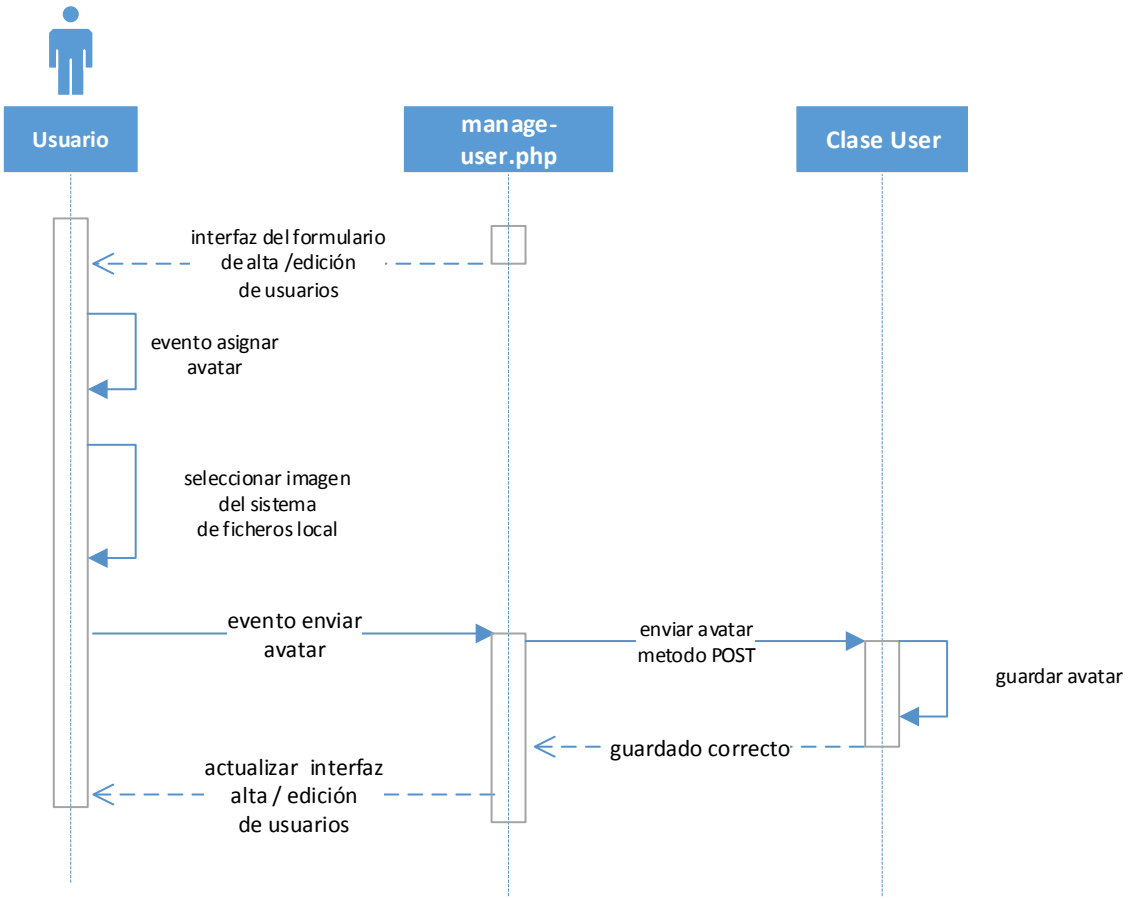
5.2.7.15. Caso de uso Editar usuario



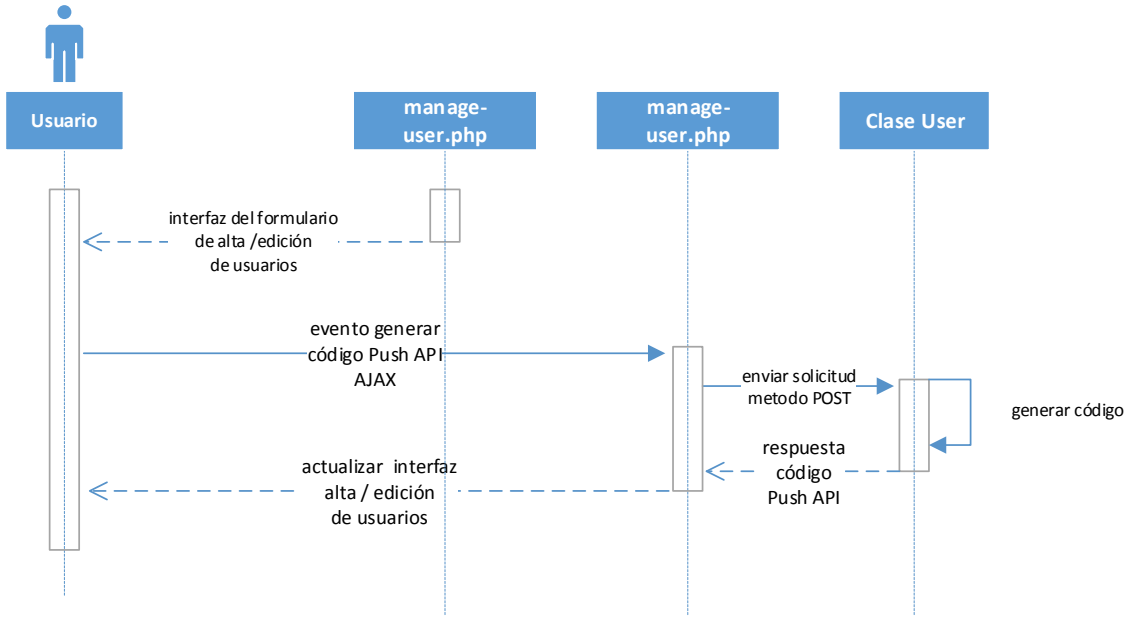
5.2.7.16. Caso de uso Eliminar usuario



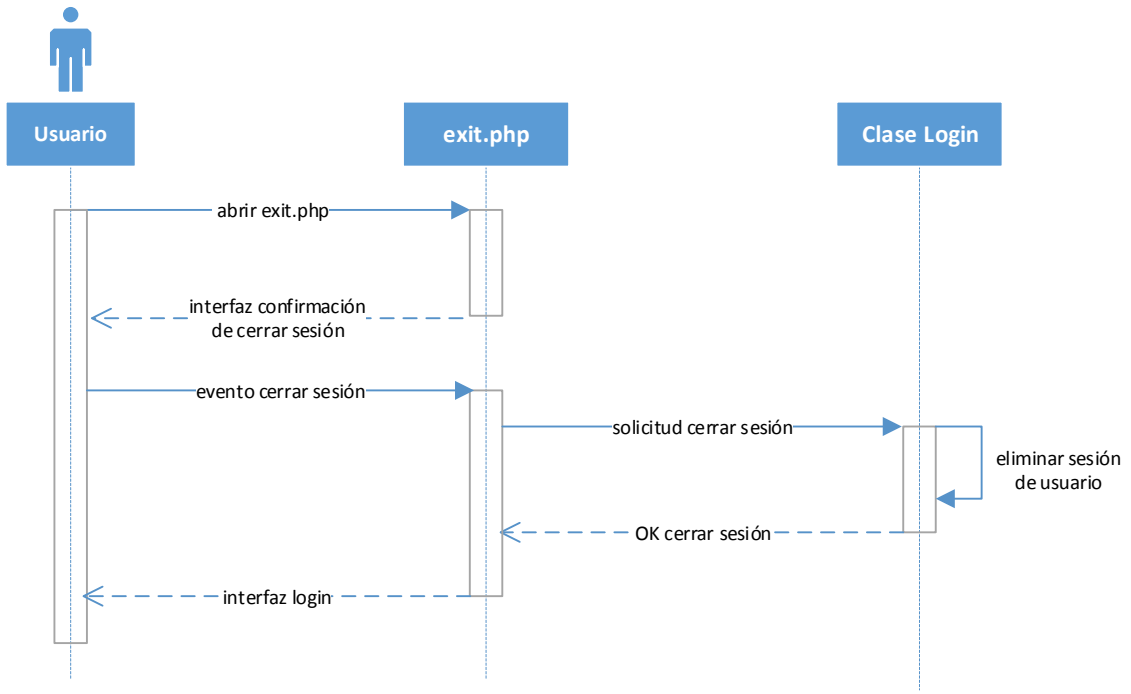
5.2.7.17. Caso de uso Asignar avatar



5.2.7.18. Caso de uso Generar código web Service



5.2.7.19. Caso de uso Cerrar sesión



5.3. Pruebas y correcciones

Para testear las aplicaciones desarrolladas hemos diseñado una serie de pruebas que permiten verificar su correcto funcionamiento y compatibilidad con diferentes sistemas. Los resultados de las pruebas realizadas se resumen a continuación.

5.3.1. Acceso al sistema

Desarrollada para evaluar el sistema de acceso a la plataforma. Se ha testado que solo es posible acceder a la misma mediante el uso de las credenciales y que no es posible acceder a través de ataques malintencionados o mediante agujeros de seguridad.

Como todos los sistemas de autenticación mediante contraseña, no está exento a ataques por fuerza bruta, aunque son fácilmente evitables mediante el uso de una política de asignación de contraseñas fuertes, aunque claro está, depende del usuario de la plataforma.

5.3.2. Alta, edición y borrado de usuarios

Desarrollada para evaluar la funcionalidad completa del menú de gestión de usuarios. En las pruebas realizadas se ha probado:

- El alta usuarios.
- La asignación de avatares.
- La generación del código de acceso al servicio web.
- La edición de usuarios previamente creados.
- Que el acceso a la lista de usuarios solo es posible a través de un usuario administrador.
- Eliminación de usuarios.

Hemos tenido éxito a la primera en 5 de las pruebas realizadas. Se detectó una implementación errónea que permitía ver el listado de usuarios a usuarios no administradores. Se corrigió la incidencia y se volvieron a realizar las pruebas esta vez con resultados positivos.

5.3.3. Alta, edición y borrado de aplicaciones

Desarrollada para evaluar la funcionalidad del menú de gestión de aplicaciones. En las pruebas realizadas se ha probado:

- El alta de aplicaciones.
- La asignación de certificados a las aplicaciones.
- Configuración de aplicaciones para diferentes plataformas.
- La edición de aplicaciones previamente creadas.

- Eliminación de aplicaciones.

En todos los casos se obtuvieron resultados satisfactorios y la herramienta se comporta de modo correcto.

5.3.4. Alta, edición y borrado de notificaciones Push

Desarrollada para evaluar la funcionalidad del menú de gestión de notificaciones. En las pruebas realizadas se ha probado:

- El alta notificaciones.
- Programación temporal de notificaciones para que se envíen en un momento concreto.
- La edición de notificaciones previamente creadas.
- Comprobación que no se permite editar notificaciones enviadas.
- Eliminación de notificaciones.

En las pruebas realizadas se obtuvieron resultados satisfactorios a excepción de un comportamiento erróneo en el guardado de las fechas de envió de las notificaciones. Se detectó que las fechas introducidas en el formulario no correspondían con las fechas guardadas en la base de datos. El error estaba provocado por el formato de fecha utilizado por el servidor web. El servidor estaba configurado para utilizar el formato de fecha ingles *mm/dd/aaaa*. Para evitar futuros problemas con diferentes servidores en donde puede estar alojada la plataforma se optó por utilizar un formato de fecha estándar internacional que soportan tanto los servidores Apache y las bases de datos MySQL cuyo formato es *aaaa-mm-dd*. Realizadas las modificaciones oportunas se solvento la incidencia.

5.3.5. Recepción de notificaciones en los dispositivos móviles

Para comprobar la recepción de las notificaciones en dispositivos móviles se han creado dos sencillas aplicaciones de test, una para dispositivos móviles IOS y otra para dispositivos móviles Android, cuya única función es la de estar configuradas para poder recibir notificaciones Push.

Una vez configuradas las aplicaciones en la plataforma se procedió al alta de varias notificaciones Push programadas para enviarse en diferentes momentos. La totalidad de notificaciones programadas llegaron a los dispositivos obteniendo así un resultado satisfactorio.

5.3.6. Compatibilidad del backend con diferentes navegadores web y plataformas

Prueba desarrollada para comprobar que la plataforma es accesible y usable a través de los diferentes navegadores web existentes y plataformas.

Se ha probado el funcionamiento de la plataforma desde los siguientes navegadores:

- Internet Explorer 9+ en sistemas Windows
- Mozilla Firefox en sistemas Windows y Mac
- Google Chrome en sistemas Windows, Mac, IOS y Android
- Safari en sistemas Mac y IOS

En todos los navegadores testados la plataforma se ha comportado correctamente, adaptándose en cada caso a las características del dispositivo utilizado. Aunque previamente lo conocíamos, hay problemas de compatibilidad con navegadores Internet Explorer en sus versiones 8 y anteriores debido a que no cumplen el estándar HTML5, aunque no tiene mayor repercusión al estar considerados de navegadores obsoletos.

5.3.7. Comprobación que el código HTML generado cumple el estándar

Prueba desarrollada para comprobar que el código HTML generado para la presentación de las interfaces de la plataforma cumple el estándar HTML5.

Para ellos se ha utilizado la herramienta online https://validator.w3.org/ provista por el consorcio W3C. Mediante la inserción del código HTML generado en el formulario de validación de la herramienta, se desencadena un proceso cuyo resultado indica si el documento HTML es válido o no, indicando en caso que no sea válido los errores de validación.

Se han realizado test de validación las distintas interfaces de la aplicación obtenido resultados positivos en todas ellas.

Figura 49. Resultado de test de validación de un documento HTML5

5.3.8. Ejecutar peticiones al servicio web

Desarrollada para evaluar la funcionalidad del servicio web.
En las pruebas realizadas se ha probado:

- Que el acceso al servicio solo es posible mediante el cumplimento del formato de solicitudes.
- La ejecución de los métodos implementados.
- Acceso a través de diferentes plataformas y dispositivos.

Se ha detectado que algunos servidores web de hosting “lowcost” donde el rendimiento de los mismos es bajo y sus recursos están limitados el funcionamiento del servicio web no es el esperado debido a restricciones del sistema. En las pruebas realizadas en servidores que podemos denominar estándar el servicio se comporta correctamente obteniendo resultados satisfactorios.

6. CONCORDANCIA ENTRE LOS RESULTADOS Y OBJETIVOS

El objetivo del proyecto era la creación de una plataforma web que permita a los desarrolladores de aplicaciones móviles gestionar el envío de notificaciones Push de una manera intuitiva, sencilla y a coste cero, dado que la aplicación será distribuida en forma de software libre. Este objetivo se ha cumplido con creces dado que la plataforma implementada cumple con todos los requisitos deseados.

Además para dotar de más versatilidad al producto, la plataforma permite la configuración standalone de modo que sea una implantación autónoma en donde una instalación será utilizada por un solo desarrollador, o de modo SAAS, en donde múltiples usuarios pueden compartir la misma implantación instalada en la nube.

Así mismo se ha provisto de un servicio web para que desarrolladores de aplicaciones que dispongan de múltiples plataformas de trabajo puedan intercomunicar todas ellas. Por ejemplo un desarrollador que tiene una aplicación móvil y a su vez un blog, podría invocar el proceso de envió de una notificación, a través del servicio web, cuando se produzca algún evento en el blog.

Cabe destacar que el resultado es una plataforma que cumple al 100% con los estándares web y por lo tanto puede ser utilizada desde cualquier plataforma usando cualquier navegador actual que cumpla con el estándar HTML5 y no corre riesgo de quedar obsoleta ya que el HTML5 es una apuesta con gran futuro para la realización de aplicaciones. Además la programación orientada a objetos y en dos capas con la que se ha implementado ha permitido construir un sistema eficiente el cual apenas necesita recursos para funcionar ágilmente.

7. COMPARACIÓN CON OTRAS ALTERNATIVAS

En este apartado tratamos las distintas alternativas para implementar la plataforma. Así presentaremos alternativas al desarrollo del backend.

La primera alternativa propuesta, sería la implementación de la plataforma mediante herramientas y lenguajes de programación propietarios. Este planteamiento fue desechado dado que la plataforma estaría sujeta a las licencias del software propietario empleado para su creación y el planteamiento primordial era el de crear software libre mediante el uso de software libre, de tal manera que pueda ser distribuido libremente y modificado por cualquier desarrollador sin tener que pagar licencias de software de desarrollo.

Otra posibilidad hubiera sido la de implementar la plataforma como una aplicación de escritorio para Windows. Al igual que en la alternativa anterior supondría estar sujetos a licencias de software y además limitaría su público objetivo, dado que solo podría ser utilizada por desarrolladores que utilicen entornos Windows. Este escenario estaría muy alejado de la realidad en el desarrollo de aplicaciones móviles, donde las plataformas de desarrollo más utilizadas son OSX en máquinas Mac de Apple para el desarrollo de aplicaciones móviles para dispositivos IOS y Linux para el desarrollo de aplicaciones móviles para dispositivos Android.

En ambos casos, la adaptación del sistema actual a una de las alternativas planteadas es relativamente sencillo, puesto que la lógica de negocio e interfaces están creadas y únicamente habría que traducirlas y adaptarlas al lenguaje escogido y plataforma escogido, aunque no sería una estrategia muy acertada.

8. CONCLUSIONES Y LINEAS FUTURAS

El desarrollo realizado cumple con los objetivos marcados al comienzo del proyecto. La aplicación está preparada y puede servir como base para cubrir objetivos más ambiciosos. Para ellos se pueden destacar dos líneas de trabajo: extender la aplicación para soportar el envío de notificaciones a otras plataformas y crear un sistema de segmentación de los destinatarios de las notificaciones.

En la primera línea cabe destacar que muchas compañías se están sumando y están apostando por la integración de notificaciones Push en sus productos. Aunque a día de hoy IOS y Android copan un 95% de la cuota de mercado de dispositivos móviles, será interesante ver cómo evoluciona el mercado y extender la aplicación para enviar notificaciones a más plataformas, como por ejemplo puede ser Windows Phone. Así mismo el futuro de las notificaciones Push parece que no va a ser exclusivo de los móviles, dado que recientemente Google ha implantado un sistema de notificaciones Push para su navegador Chrome. Si esta propuesta evoluciona será interesante también extender la aplicación para que envíe notificaciones a navegadores web.

La segunda línea estaría enfocada a extender la funcionalidad de la plataforma para soportar segmentación de destinatarios de las notificaciones Push. Mediante la implantación de un sistema para recabar información acerca del uso que los usuarios hacen de las aplicaciones móviles configuradas en la plataforma, el sistema sería capaz de interpretar esa información y clasificar a cada uno de los destinatarios en un sistema de clasificación predefinido. Esta característica sería realmente útil en aplicaciones de comercio electrónico para poder definir campañas de marketing dirigidas a los usuarios según sus preferencias obtenidas a través del uso que hacen de la aplicación.

9. BIBLIOGRAFÍA

Sitios web oficiales:

- Apple Developer Center (documentación sobre notificaciones Push para IOS): https://developer.apple.com
- Android Developer Center (documentación sobre notificaciones Push para Android): http://developer.android.com
- Sitio oficial del lenguaje PHP: http://php.net/
- Documentación acerca de protocolos y estándares de la web http://www.w3.org/

Foros de desarrolladores:

- Code Project (documentación sobre desarrollos web basados en LAMP): http://www.codeproject.com/
- Stack Overflow (documentación sobre desarrollos web basados en LAMP): http://www.stackoverflow.com