Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Componente de notificaciones integrable en aplicaciones WAINE

Autora: Ana Varela Rodríguez
Tutor: Antonio Luis Delgado González

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Departamento de Ingeniería Telemática
.

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Componente de notificaciones integrable en aplicaciones WAINE

Autora:

Ana Varela Rodríguez

Tutor:

Antonio Luis Delgado González

Profesor Asociado

Dpto. Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Componente de notificaciones integrable en aplicaciones WAINE

Autora: Ana Varela Rodríguez.

Tutor: Antonio Luis Delgado González.

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2013

El Secretario del Tribunal

A mi familia
A mi compañero
A mis amigos y telequitos
A mis profesores
Y a mis gatas.

Agradecimientos

Me gustaría agradecer en primer lugar al tutor de este trabajo Antonio Luis Delgado por su paciencia, compromiso, conocimiento, esfuerzo y tiempo en todo momento del desarrollo de este.

En segundo lugar a mi familia, sin ellos no habría podido cursar esta carrera que me apasionaba al mismo tiempo me hacía llorar (en sentido literal). Gracias a todos por el esfuerzo y paciencia que habéis puesto sobre mi.

Aunque forma parte de mi familia, quería hacer mención especial a Guille, que ha aguantado mis días malos y siempre con alguna frase o abrazo para animarme y hacerme continuar. Sin él y su apoyo no hubiera podido finalizar este trabajo.

A mis amigos de toda la vida por comprender y respaldarme en este camino.

A mis compañeros de facultad, los de la primera fase y a los de la especialidad. Ha sido un trayecto duro pero mucho menos duro a vuestro lado. Gracias por ayudarme, y por dejarme compartir con vosotros almuerzos, cafés, horas de biblioteca, horas de sótano, horas de clases, de laboratorio y de centro de cálculo (y algún que otro momento más).

Por último, quería agradecier su esfuerzo y dedicación a los profesores con los que he coincidido durante estos años. He aprendido muchísimo, todo gracias a vuestro profundo conomiciento y vuestra pasión por la enseñanza.

Estudiar este grado, me ha abierto puertas y alguna que todavía está sin abrir, me siento muy agradecida por haber elegido este camino, y por fin, poder finalizarlo.

Ana Varela
Sevilla, 2023

Resumen

En la actualidad, nos encontramos inmersos en un mundo interconectado donde la comunicación es un papel fundamental en nuestras vidas. La tecnología ha evolucionado a pasos de gigantes y nos ha brindado una gran variedad de aplicaciones para comunicarnos de manera instantánea y efectiva haciendo uso de notificaciones.

No sólo las personas realizan notificaciones. Constantemente recibimos notificaciones remitidas por aplicaciones y sistemas de información.

El objetivo de este trabajo de fin de grado es la implementación de un *componente de software reutilizable* para el envío de notificaciones. Este componente dotará a aplicaciones desarrolladas con el entorno WAINE de la capacidad para el envío de notificaciones.

Para su desarrollo, diferenciamos dos partes, el desarrollo del **servicio de notificaciones** que se ha desarrollado en lenguaje PHP y la construcción de **interfaces de usuarios en entornos WAINE**, que se ha empleado una técnica conocida como *Desarrollo de Interfaces de Usuario Basado en Modelos* (MBUID).

Como resultado se realizan pruebas en entornos WAINE del componente de notificaciones, se analiza el componente desarrollado, así como líneas de avance.

Abstract

Today, we are living in an interconnected world where communication plays a key role in our lives. Technology has evolved by leaps and bounds and has provided us with a wide variety of applications to communicate instantly and effectively using notifications.

It is not only people who make notifications. We constantly receive notifications from applications and information systems.

The objective of this project is the implementation of a reusable software component designed to send notifications. This component will provide applications developed with WAINE environments with the ability to send notifications.

For its development, we can identify two parts, the development of the software component, which has been developed in PHP language and the construction of user interfaces in WAINE environments, which has been used a technique known as Model-Based User Interface Development (MBUID).

As a result, the notification component is tested in WAINE environments, the developed component is analyzed, as well as lines of progress.

Sumario

Índice de tablas

Índice de figuras

1 Introducción

El presente documento recoge la memoria del trabajo de fin de grado titulada *Componente de notificaciones integrable en aplicaciones WAINE* para la obtención del título de Graduado en Ingeniería de las Tecnologías de Telecomunicación por la Universidad de Sevilla de la alumna Ana Varela Rodríguez.

En la actualidad, nos encontramos inmersos en un mundo interconectado donde la comunicación es un papel fundamental en nuestras vidas. La tecnología ha evolucionado a pasos de gigantes y nos ha brindado una gran variedad de aplicaciones para comunicarnos de manera instantánea y efectiva. Entre las más destacadas se encuentran el correo electrónico, WhatsApp y los mensajes de texto (SMS).

El correo electrónico, conocido comúnmente como email, ha cambiado la forma en la que nos comunicamos tanto en el ámbito personal como profesional. Con la posibilidad de acceder a nuestras bandejas de entrada desde cualquier dispositivo con conexión a internet, el correo electrónico se ha convertido en una herramienta indispensable para la comunicación global.

Recientemente, aplicaciones como WhatsApp o Telegram, aplicaciones de mensajería instantánea han ganado una notable popularidad en los últimos años. Ofrece una amplia gama de funciones que van más allá de simples mensajes de texto como mensajes de voz, video llamadas de alta calidad o incluso podemos compartir fotos y videos.

Sin olvidarnos del SMS, una forma de comunicación que ha perdurado a lo largo del tiempo. Su capacidad de compartir fotos o vídeos es limitada en comparación con las anteriores, pero siguen siendo una herramienta eficaz para enviar mensajes cortos y directos.

Todas estas opciones nos ofrecen una flexibilidad y la capacidad de elegir qué opción se adapta mejor a nuestras necesidades. La comunicación nunca ha sido tan accesible y versátil como lo es hoy en día.

Estos sistemas, nos permiten enviar notificaciones, lo cual resulta esencial para mantenernos informados, conectados con las diferentes plataformas y servicios que usamos en nuestro día a día.

Las notificaciones nos ayudan a optimizar nuestra productividad y eficiencia, así como mejorar la participación y compromiso del usuario. Al recibir recordatorios o actualizaciones, promociones o nuevas funciones en una aplicación, somos más propensos a participar activamente, mejorando así su experiencia en general.

En conclusión, la necesidad de que las aplicaciones envíen notificaciones a los usuarios nos mantienen conectados, mejoran nuestra productividad y fomentan la participación. También indicar que debemos establecer un equilibrio adecuado para que éstas notificaciones sean útiles y no se conviertan en una fuente de distracción o similar.

WAINE (Web Application INterface Engine), es un MB-UIDE motor de desarrollo de aplicaciones web que, basándose en modelos de interfaces de usuario, permite crear las aplicaciones cumpliendo unos objetivos básicos como; *Independencia, Seguridad; Personalización, Minimización de la programación y Eficiencia*. Tratando así de simplificar al máximo el modelo de una aplicación de gestión.

WAINE por si mismo no proporciona un módulo predeterminado para el envío de notificaciones,

luego la necesidad de incluir esta funcionalidad para las aplicaciones desarrolladas con **WAINE** resulta muy interesante.

Un servicio de notificaciones puede proporcionar varios beneficios importantes entre ellos se destacan:

- Comunicación con los usuarios. Las notificaciones permiten mantener una comunicación directa con los usuarios.
- Mejora de la retención de usuario. Al enviar notificaciones oportunas, el usuario se sentirá atendido y fomenta su participación.
- Recordatorios y alertas.
- Interacción y atención: notificar nuevos contenidos, eventos o funcionalidades.
- Acciones específicas: incentivar usuarios a realizar acciones, por ejemplo participar en una encuestas o leer un articulo.

En resumen, para aplicaciones desarrolladas con **WAINE** puede mejorar la comunicación con el usuario, enviar recordatorios y alertas importantes y en general mejorar la experiencia de los usuarios con la aplicación.

El servicio de notificaciones wnotif es un proyecto que pretende resolver el problema comentado anteriormente, e insertar un componente de notificaciones para aplicaciones desarrolladas con WAINE.

Los objetivos de este proyecto son:

- Elaborar un ***componente software reutilizable*** diseñado para facilitar el envío de notificaciones.
- Dotar de forma sencilla a futuras aplicaciones desarrolladas con WAINE la ***capacidad para enviar notificaciones.***
- ***Reducir el coste*** de futuros proyectos.
- Desarrollar un componente de cierta complejidad que sirva como ***ejemplo al desarrollo de nuevos componentes.***

A continuación se indican los pasos seguidos para el desarrollo del componente de notificaciones wnotif:

1. Fase de *análisis*, donde se detallan los requisitos, el estado inicial y los bocetos de las interfaces necesarias.
2. *Planificación* temporal del TFG, que incluye las tareas a realizar y los entregables generados en esta etapa.
3. Fase de *diseño*, que incluye el diseño de las interfaces y la arquitectura de la aplicación. Descripción detallada del proceso así como diagramas y unidades de interacción.
4. Implementación del servicio, donde se aborda la *construcción* del servicio de notificaciones, la API y las interfaces de usuario.
5. Finalmente se presentan las *conclusiones*, donde se analiza el trabajo realizado y se realiza una retrospectiva del mismo.

Esta memoria se organiza siguiendo las fases empleadas para el desarrollo de wnotif.

2 Documento de análisis de wnotif

Antonio Luis Delgado González, aldelgado@us.es 09/01/2017

2.1 Introducción

El proyecto wnotif pretende desarrollar un componente software reutilizable que permita a las aplicaciones y a sus usuarios enviar notificaciones de varios tipos.

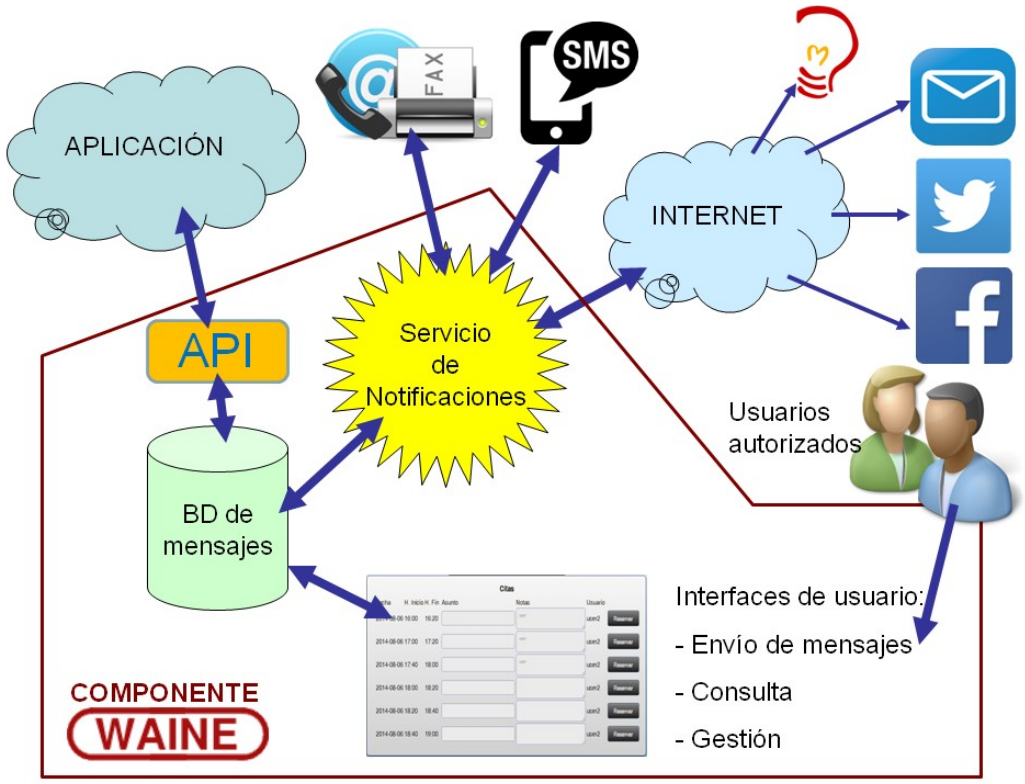

Ilustración 1: Diagrama informal del sistema

Este componente será desarrollado para su futuro empleo en aplicaciones creadas con **WAINE**[1][2].

El componente estará formado por tres subsistemas principales:

- API: que permitirá a las distintas aplicaciones (agentes) solicitar el envío de notificaciones y consultar sobre el estado de las mimas.
- Las interfaces de usuario: empleadas por los usuarios realizar notificaciones, obtener información de las mismas y administrar el sistema
- Servicio de notificaciones: el encargado de realizar los envíos de notificaciones.

2.1.1 Objetivos del proyecto

1. Elaborar un componente software reutilizable para el envío de notificaciones.
2. Dotar de forma sencilla a futuras aplicaciones desarrollados con WAINE de la capacidad de envío de notificaciones
3. Reducir el coste de futuros proyectos.
4. Desarrollar un componente de cierta complejidad que sirva como ejemplo al desarrollo de nuevos componentes.

2.2 Catálogo de requisitos

En esta sección se enumeran los requisitos identificados para el componente wnotif.

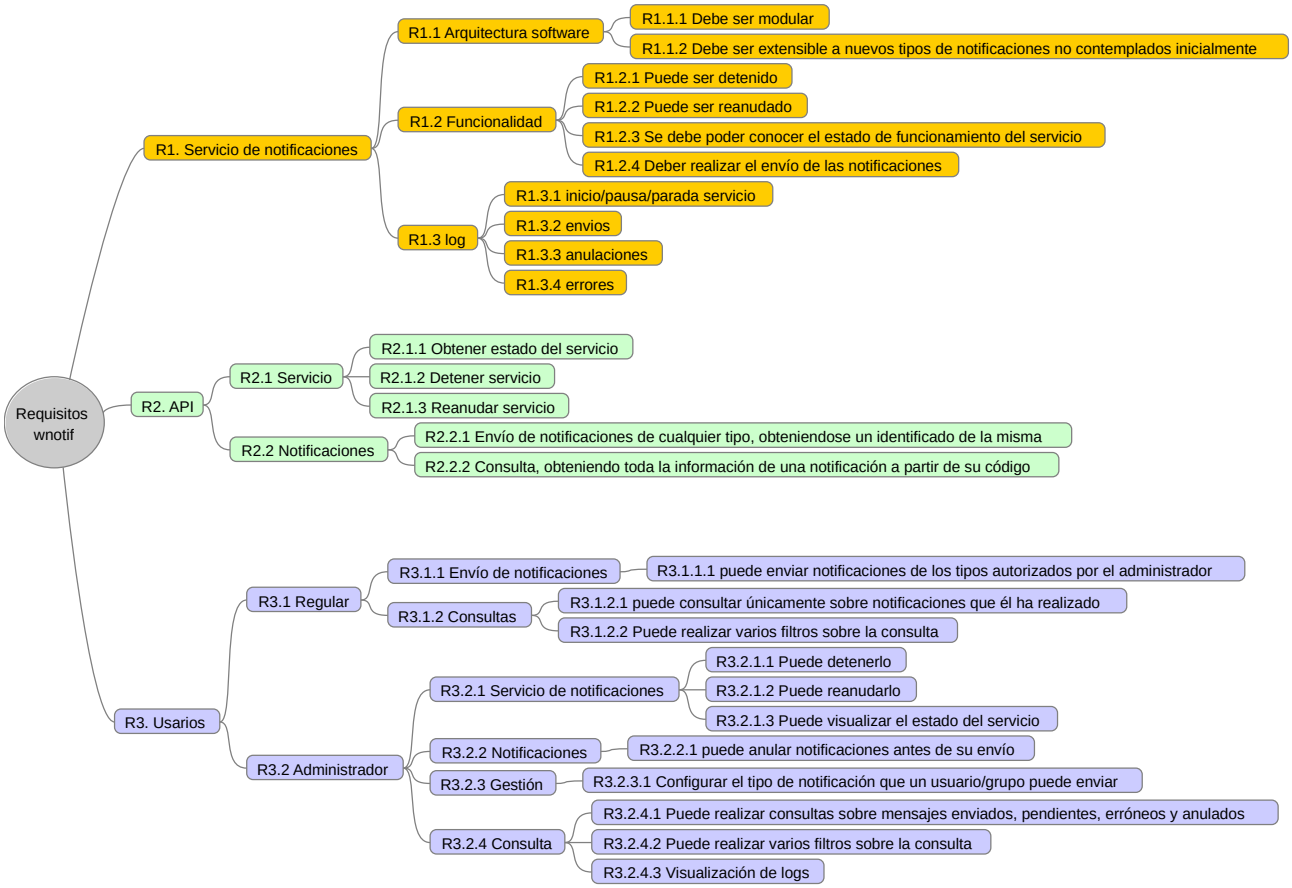

Ilustración 2: Requisitos de wnotif

2.2.1 Servicio de notificaciones

El servicio de notificaciones será el encargado de enviar las notificaciones a los notificados. También entre sus funciones estará el mantenimiento del log del sistema.

- **R1.1 Arquitectura software**
  - *R1.1.1 Debe ser modular*. El sistema de notificaciones contendrá una serie de módulos o subsistemas encargados del envío de cada tipo de notificación. Estos módulos deberán

soportar una interfaz común.

- *R1.1.2 Debe ser extensible a nuevos tipos de notificaciones no contemplados inicialmente*. Extender el sistema para soportar un nuevo tipo de notificación debería implicar:
  1. la escritura de un nuevo módulo para ese tipo de notificación
  2. registrar el nuevo módulo en el sistema de notificaciones

- **R1.2 Funcionalidad**
  - *R1.2.1 Puede ser detenido*. El servicio de notificaciones podrá detenerse si estaba en funcionamiento.
  - *R1.2.2 Puede ser reanudado*. El servicio de notificaciones podrá reanudarse si previamente ha sido detenido.
  - *R1.2.3 Se debe poder conocer el estado de funcionamiento del servicio*. Se deben proveer mecanismos para conocer el estado en el que se encuentra el servicio.
  - *R1.2.4 Deber realizar el envío de las notificaciones*. El servicio de notificaciones es el encargado de realizar el envío de todas ellas, y si no fuera posible realizarlo tras un número de reintentos establecido, marcarlas como erróneas.
- **R1.3 log**
  - *R1.3.1 inicio/parada servicio*. Se debe anotar en el log del servicio los inicios y paradas del mismo.
  - *R1.3.2 envíos.* Se deben anotar en el log del servicio los envíos realizados.
  - *R1.3.4 errores*. Se deben anotar en el log del servicio los envíos que no han podido ser realizados y que han sido marcados como errores.
  - *R1.3.3 anulaciones*. Se deben anotar en el log del servicio las anulaciones realizadas.

2.2.2 API

El API está destinado a que las aplicaciones (o agentes) puedan hacer uso y obtener información del servicio de notificaciones.

- **R2.1 Servicio**
  - R2.1.1 *Obtener estado del servicio*. Debe devolver a los agentes el estado del servicio: en ejecución o detenido.
  - R2.1.2 *Detener servicio*. Permite a los agentes detener el funcionamiento del servicio de notificaciones.
  - R2.1.3 *Reanudar servicio*. Permite a los agentes reanudar el funcionamiento del servicio de notificaciones.
- **R2.2 Notificaciones**
  - R2.2.1 *Envío de notificaciones de cualquier tipo, obteniéndose un identificado de la misma*. Los agentes podrán solicitar el envío de notificaciones de cualquier tipo. Para

ello se deberán aportarán todos los datos necesarios para poder realizar el envío y el API devolverá el identificador de solicitud de notificación.

- R2.2.2 *Consulta, obteniendo toda la información de una notificación a partir de su código*. Dado un identificador de solicitud de envío, el API devolverá toda su información asociada.

2.2.3 Usuarios

- **R3.1 Regular**
  - R3.1.1 Envío de notificaciones
    - R3.1.1.1 *Puede enviar notificaciones de los tipos autorizados por el administrador*. Un usuario podrá solicitar el envío de notificaciones dentro de las que el administrador haya autorizado para el grupo de usuarios al que pertenece.
  - R3.1.2 Consultas
    - R3.1.2.1 *Puede consultar únicamente sobre notificaciones que él ha realizad*o. Un usuario regular podrá realizar consultas sobre las notificaciones que él ha solicitado.
    - R3.1.2.2 *Puede realizar varios filtros sobre la consulta*. Para realizar estas consultas podrá filtrar la información por varios campos.
- **R3.2 Administrador**
  - R3.2.1 Servicio de notificaciones
    - R3.2.1.1 *Puede detenerlo*. El usuario administrador podrá detener el servicio de notificaciones.
    - R3.2.1.2 *Puede reanudarlo*. El usuario administrador podrá reanudar el servicio de notificaciones.
    - R3.2.1.3 *Puede visualizar el estado del servicio*. El usuario administrador podrá visualizar cuál es el estado del servicio de notificaciones.
  - R3.2.2 Notificaciones
    - R3.2.2.1 *Puede anular notificaciones antes de su envío*. El usuario administrador podrá anular cualquier tipo de notificación (de agente o usuario) obviamente antes de que se haya producido su envío.
  - R3.2.3 Gestión
    - R3.2.3.1 *Configurar el tipo de notificación que un usuario/grupo puede enviar*. El usuario administrador podrá asignar a cada grupo de usuarios de una aplicación el tipo de notificación que puede realizar.
  - R3.2.4 Consulta
    - R3.2.4.1 *Puede realizar consultas sobre mensajes enviados, pendientes, erróneos y anulados*. El usuario administrador podrá realizar consultas sobre cualquier tipo de notificación.
    - R3.2.4.2 *Puede realizar varios filtros sobre la consulta*. Para realizar estas consultas podrá filtrar la información por varios campos.

- R3.2.4.3 *Visualización de logs*. El usuario administrador podrá visualizar el log del servicio de notificaciones.

2.3 Diagrama de contexto

En esta sección se presenta el diagrama de contexto del sistema.

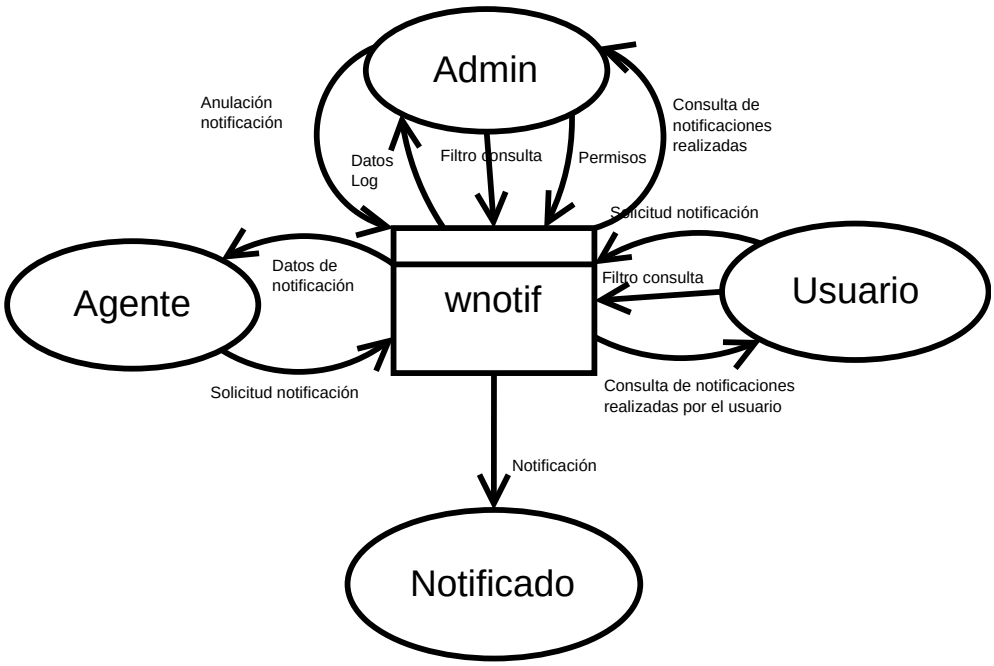

Ilustración 3: Diagrama de contexto

2.3.1 Entidades externas

- **Admin**
  - Usuario administrador del sistema
- **Usuario**
  - Usuario regular del sistema: usuario autorizado a realizar notificaciones
- **Agente**
  - Aplicación o subsistema que realiza notificaciones
- **Notificado**
  - Persona o grupo de personas que recibe la notificación

2.3.2 Flujos de datos

- Notificado

- *Notificación* (R1.2.4): Mensaje que recibe el notificado (email, SMS, jabber, etc.)

- Usuario
  - *Solicitud de notificación* (R3.1.1): Datos suministrados por el usuario para que el sistema envíe una notificación
    - Tipo de notificación que desea enviar (email, SMS, jabber, etc.)
    - Destinatario de la notificación
    - Asunto de la notificación (no siempre será necesario, dependerá del tipo de notificación)
    - Cuerpo de la notificación
  - *Filtro consulta* (R3.1.2.2): Información que suministra el usuario para filtrar la información que desea recibir en la consulta. Todos los componentes de este flujo de datos son opcionales
    - Estado de la notificación: puede tener los valores pendiente, enviada, errónea y anulada.
    - Tipo de notificación
    - Fecha de creación de notificación inicial para la búsqueda.
    - Fecha de creación de notificación final para la búsqueda.
  - *Consulta de notificaciones realizadas por el usuario* (R3.1.2.1): Información que recibe el usuario cuando realiza una consulta.
    - Identificador de la notificación
    - Tipo de notificación (email, SMS, jabber, etc.)
    - Instante de creación de la solicitud
    - Instante de envío de la solicitud
    - Destinatario de la notificación
    - Cuerpo del mensaje
- Admin
  - *Filtro consulta* (R3.2.4.2): Todos los componentes de este flujo de datos son opcionales
    - Estado de la notificación: puede tener los valores pendiente, enviada, errónea y anulada.
    - Agente que solicita el envío
    - Usuario que solicita el envío.
    - Tipo de notificación
    - Fecha de creación de notificación inicial para la búsqueda.
    - Fecha de creación de notificación final para la búsqueda.
  - *Consulta de notificaciones realizadas* (R3.2.4.1):

        - Identificador de la notificación
        - Tipo de notificación (email, SMS, jabber, etc.)
        - Agente que realizó la notificación
        - Usuario que realizó la notificación
        - Instante de creación de la solicitud
        - Instante de envío de la solicitud
        - Destinatario de la notificación
        - Cuerpo del mensaje
    - *Datos anulación notificación* (R3.2.2.1):
        - Identificador de la notificación a anular
    - *Datos permisos* (R3.2.3.1):
        - Par formado por un identificador de grupo y un identificador de tipo de notificación. Tiene el significado de que ese grupo queda autorizado para enviar ese tipo de notificación.
- Agente
    - *Solicitud de notificación* (R2.2.1):
        - Identificador de la notificación de la que se desea obtener su información
    - Datos de notificación (R2.2.2):
        - Identificador de la notificación
        - Tipo de notificación (email, SMS, jabber, etc.)
        - Remitente de la notificación
        - Destinatario de la notificación
        - Agente que realizó la notificación
        - Usuario que realizó la notificación
        - Instante de creación de la solicitud
        - Instante de envío de la solicitud
        - Instante de anulación de la solicitud
        - Asunto del mensaje
        - Cuerpo del mensaje

2.4 Modelo de procesos

A continuación se describen los principales procesos de wnotif

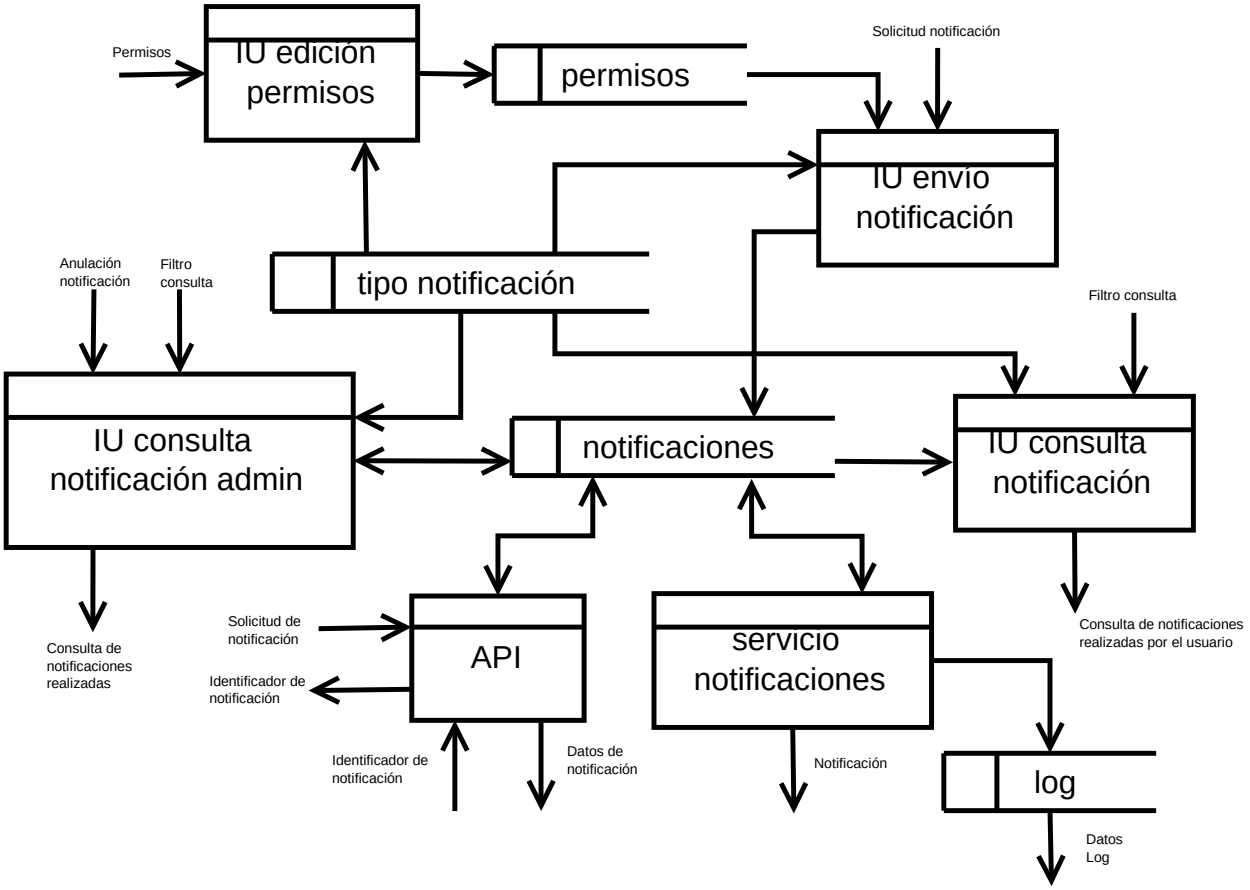

Ilustración 4: Diagrama de flujo de datos de nivel 1

2.4.1 Procesos

- **API** (R2): Debe permitir a los agentes el envío de notificaciones y la consulta de datos de una notificación.
  - Para el envío de una notificación se deben aportar los datos necesarios para su alta (*Solicitud de notificación*) y se obtendrá el identificador de la notificación creada si el proceso tuvo éxito. (R2.2.1)
  - Para la consulta de datos de una notificación se aportará el identificador de la notificación que se quiere consultar y se obtendrán todos los datos de la misma. (R2.2.2)
- **Servicio de Notificaciones** (R2): Encargado de enviar las notificaciones. Este proceso se ejecutará de forma periódica y tomara la información necesaria del almacén notificaciones y realizará las acciones necesarias para emitir las notificaciones solicitadas. Además realizará las siguientes acciones:
  - Si el envío tiene éxito actualizará la fecha de envío de la notificación. (R1.2.4)
  - En caso de no poder emitir la notificación (se podrá configurar el número de reintentos)

la marcará como errónea. (R1.2.4)

- Escribirá información de inicios y paradas del servicio, así como de envíos y errores en el log. (R1.3)

- **IU envío notificación**: Permite a un usuario regular cumplimentar la información necesaria para realizar una solicitud de notificación. Debe informar al usuario del identificador asignado a su solicitud. (R3.1.1)
- **IU consulta notificación**: Permite a un usuario regular realizar consultas sobre las notificaciones que ha solicitado. Para ello, el usuario puede aportar ciertos campos que conforman un filtro. (R3.1.2)
- **IU edición de permisos**: Permite al usuario administrador asociar a cada grupo de usuarios los tipos de notificaciones que puede solicitar. (R3.2.3.1)
- **IU consulta notificación Admin**: Permite al usuario administrador realizar consultas sobre cualquier notificación del sistema. El administrador puede aportar ciertos campos que conforman un filtro. Sobre el resultado obtenido el administrador puede anular notificaciones. (R3.2.4, R3.2.2)

2.4.2 Almacenes de datos

- **Tipo de notificación**: Contiene las descripción de todos los tipos de notificación soportados.
- **Notificación**: Contiene todas las notificaciones realizadas y por realizar, las erróneas y las anuladas.
- **Permisos**: Para cada grupo almacena los tipos de notificaciones que le están permitido realizar
- **Log**: Log del sistema de notificaciones conteniendo la siguiente información:
  - Instante temporal: formato ISO 8601 `YYYY-MM-DDThh:mm:ssTZD`
  - Acción: podrá ser cualquiera de los siguientes valores
    - service start
    - service stop
    - sent
    - error
    - cancel
  - Información extra
    - Identificador de notificación (para las acciones sent, error y cancel).

2.5 Modelo de datos

A continuación se presenta el modelo de datos del sistema expresado por un diagrama entidad-relación.

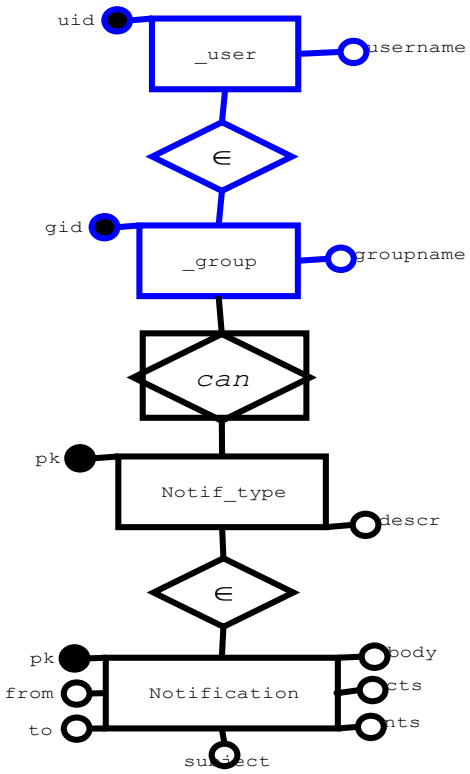

Ilustración 5: Diagrama entidad-relación

2.5.1 Entidades

- **Notif_type**
  - Tipo de notificación
  - *pk*: Clave primaria
  - *descr*: Descripción del tipo, p.e.: SMS, Email, Jabber, twitter, etc.
- **Notification**
  - Notificación
  - *pk*: Clave primaria
  - *from*: Remitente, p.e: root@cdc.us.es, 678901234, @Etsi, etc.
  - *to*: Destinatario, p.e: ana@gmail.com, 654321098, @PedroRus, etc.
  - *subject*: Asunto (opcional)
  - *body*: Cuerpo de la notificación
  - *cts*: Instante de creación
  - *sts*: Instante de envío
  - *ats*: Instante de anulación

2.5.2 Relaciones

- **Can**
  - Relación entre un grupo de usuarios (*_group*) y los tipos de notificaciones que puede enviar

2.6 Interfaces de usuario y acciones

Las principales interfaces y acciones de los usuarios del sistema son las siguientes:

	Detener el Servicio	Reanudar el servicio	Estado del servicio	Administración de permisos	Consulta y anulación	Envío de notificaciones	Consulta de notificaciones
Administrador	✔✔✓	✓	✔✔✓	✓	✓		
Usuario regular						✓	✓

A continuación, se describen en detalle

2.6.1 Usuario administrador

Las acciones que puede realizar el usuario administrador son:

2.6.2 Detener el servicio

Esta acción permite al usuario administrador detener el servicio si el mismo está en funcionamiento. (R3.2.1.1)

2.6.3 Reanudar el servicio

Esta acción permite al usuario administrador reanudar el servicio si previamente fue detenido. (R3.2.1.1)

2.6.4 Estado del servicio

Esta interfaz de usuario indica al administrador el estado de ejecución del servicio. Presentará dos posibles mensajes: "EN EJECUCIÓN" o "DETENIDO". (R3.2.1.3)

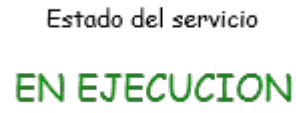

Ilustración 6: Formulario de estado del servicio

2.6.5 Administración de permisos

Este formulario permite al administrador asignar a cada grupo de usuario los distintos tipos de notificaciones que puede emitir. (R3.2.3.1)

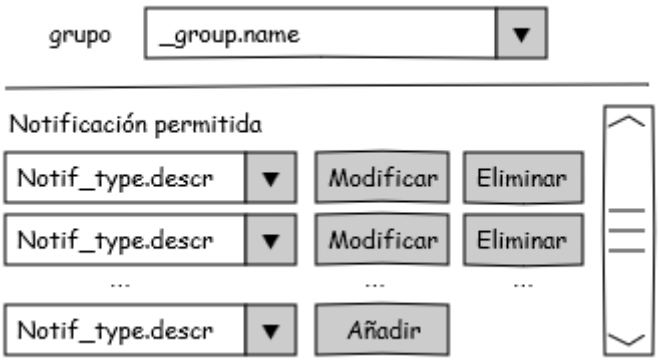

Ilustración 7: Formulario de administración de permisos

2.6.6 Consulta y anulación de notificaciones

Con esta interfaz de usuario un usuario administrador puede consultar notificaciones y dentro del resultado obtenido en la consulta anular las que considere oportunas. (R3.2.4, R3.2.2.1)

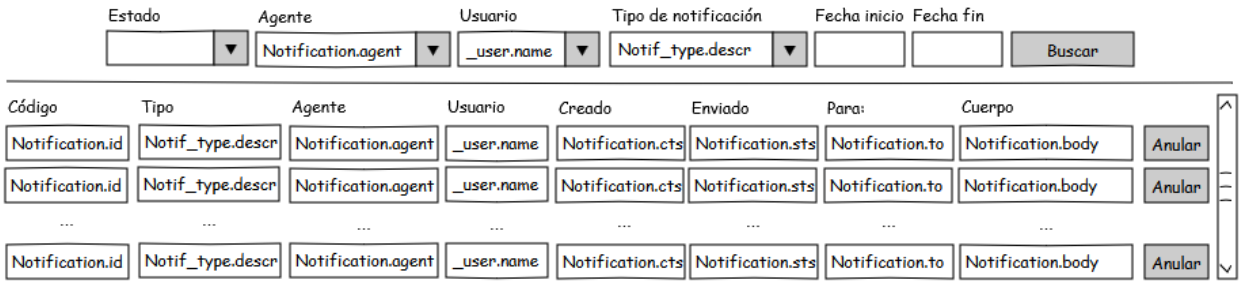

Ilustración 8: Formulario de consulta y anulación de notificaciones

Todos los campos para los filtros son opcionales y cuando son cumplimentados se comportan como un AND Como se puede apreciar se pueden emplear los siguientes filtros:

- Estado: puede tener los valores pendiente, enviada, errónea y anulada.
- Agente: agente que solicita el envío
- Usuario: usuario que solicita el envío.
- Tipo de notificación: tipo de notificación
- Fecha de inicio: fecha de creación de notificación inicial para la búsqueda.

- Fecha de fin: fecha de creación de notificación final para la búsqueda.

Sobre el resultado de la consulta se debe permitir al administrador anular notificaciones que no han sido enviadas aún (Botón "Anular").

2.7 Usuario regular

Los principales interfaces de usuario para el un usuario regular son las siguientes:

2.7.1 Envío de notificaciones

El formulario de envío de notificaciones permite a un usuario regular solicitar el envío de notificaciones siempre dentro de los tipos permitidos por el usuario administrador (ver sección 2.6.5). (R3.1.1.1)

Ilustración 9: Formulario de envío de notificación

Debe tenerse en cuenta que el campo Asunto no siempre es obligatorio (depende del tipo de notificación).

2.7.2 Consulta de notificaciones

Esta interfaz permite a los usuarios consultar las notificaciones que han emitido. (R3.1.2.1)

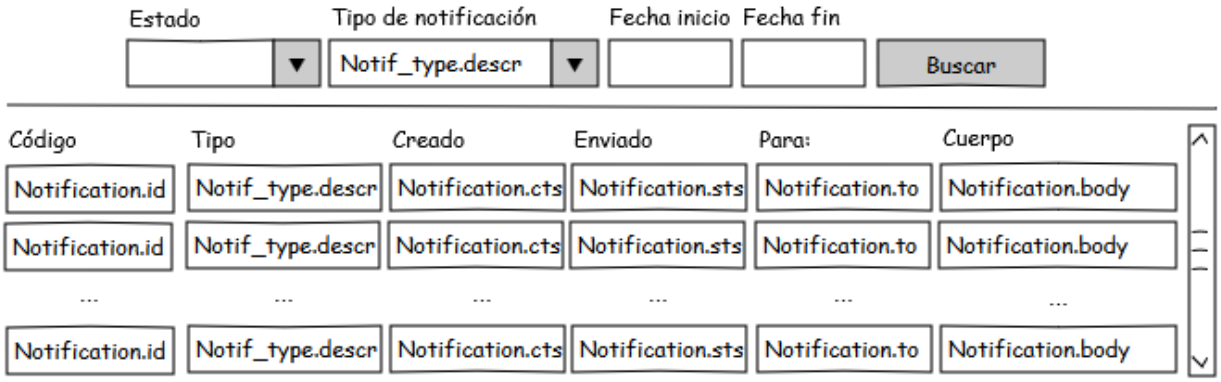

Ilustración 10: Formulario de consulta de notificaciones

Todos los campos para los filtros son opcionales. Como se puede apreciar se pueden emplear los siguientes filtros:

- Estado: puede tener los valores pendiente, enviada, errónea y anulada.
- Tipo de notificación: tipo de notificación
- Fecha de inicio: fecha de creación de notificación inicial para la búsqueda.
- Fecha de fin: fecha de creación de notificación final para la búsqueda.

3 Plan del servicio wnotif

Ana Varela Rodríguez, anavarelarod@gmail.com 30/06/2023

3.1 Introducción

Este documento presenta el plan de proyecto del trabajo de fin de grado wnotif. Este proyecto comprende el desarrollo de un componente software integrable en aplicaciones **WAINE** que realice el envío de notificaciones. Éstas pueden ser de varios tipos, vía SMS, fax o a través de internet mediante Facebook, Twitter, email, linkedin...

El trabajo abarca el diseño, construcción e implantación de un componente reutilizable que servirá como ejemplo al desarrollo de nuevos componentes, y que ayudará a reducir el coste de futuros proyectos.

El componente constará de tres subsistemas principales:

- *API*: Conjunto de funciones para solicitar el envío de notificaciones y consultar el estado de las mismas.
- *Servicio de notificaciones*: servicio que se encarga de realizar los envíos de notificaciones.
- *Interfaces de usuario:* interfaces para permitir al administrador y a los usuarios realizar notificaciones, obtener el estado de las mismas y administrar el sistema.

El sistema utiliza una base de datos que almacenará toda la información acerca de las notificaciones, como el tipo de notificación, los permisos que tendrán los diferentes usuarios a la hora de realizar envíos, estado de las mismas e información de log.

3.1.1 Objetivos

Los objetivos del proyecto son los siguientes:

- Elaborar un componente software reutilizable para el envío de notificaciones.
- Dotar de forma sencilla a futuras aplicaciones desarrolladas con **WAINE**, de la capacidad de envío de notificaciones.
- Reducir el coste de futuros proyectos.

En resumen, supondrá un importante avance a la hora de la comunicación entre aplicación y usuario. Éstos tendrán la opción de generar notificaciones, ya que el servicio será escalable, modular y fácil de usar gracias a las interfaces de usuario que se crearán para la total personalización de las solicitudes. Además, el servicio de notificaciones será totalmente adaptable a cualquier tipo de aplicación, no sólo a las desarrolladas con **WAINE**, ya que gracias a la API, estas aplicaciones (agentes), pueden comunicarse con el servicio muy fácilmente.

3.2 Alcance

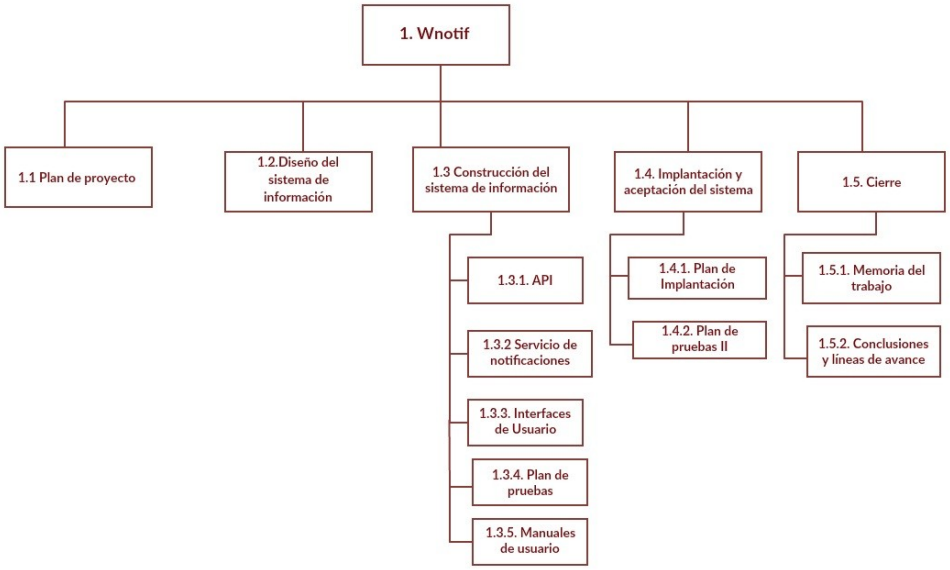

Ilustración 11: Diagrama WBS

En el diagrama de descomposición del trabajo (WBS) se indican los principales productos del proyecto.

Pasamos a describir los diferentes entregables que componen el trabajo:

- *Plan de proyecto:* documento que resume los objetivos del trabajo, muestra la planificación temporal, diagrama de estructura del trabajo, recursos empleados y participantes implicados en la realización del mismo. (*Punto 1.1 del Diagrama WBS*)
- *Diseño del sistema de información:* documento de diseño (P*unto 1.2 del Diagrama WBS*) que sigue las siguientes secciones:
    1. Introducción
    2. Arquitectura del sistema: diagrama de despliegue, comoponentes y elementos que forman parte del sistema.
    3. Modelo físico de datos: estructura física de datos que utilizará el sistema.
    4. Diseño de las interfaces de usuario: diseño de las unidades de interacción para los usuario.
    5. Casos de usos reales: que definen al servicio wnotif, incluyendo un diagrama de casos de uso.
    6. Clases. Diagrama de clases presentando la estructura que tiene el servicio de notificaciones y la API.
    7. Especificaciones de construcción.
    8. Migración y carga inicial de datos.Estos

9. Plan de pruebas técnico: plan definidos por niveles para las pruebas de implementación e implantación del sevicio.

- *Construcción del sistema de información:* generación y construcción de los diferentes componentes que forman parte del proyecto siguiendo y se realizan el correspondiente plan de pruebas, con pruebas unitarias, de integración, de sistema entre otras (*Punto 1.3.4 del Diagrama WBS*), así como los manuales de usuario correspondientes (*Punto 1.3.5. del Diagrama WBS*) . Está separado en tres productos principales que corresponden con el proceso de construcción del proyecto (*Punto 1.3 del Diagrama WBS)* :
  - *API:* API que permitirá a las distintas aplicaciones realizar consultas y solicitar envíos de notificaciones. Generación del código del mismo. (*Punto 1.3.1 del Diagrama WBS*)
  - *Servicio de notificaciones:* componente principal del trabajo, que será el encargado de realizar las notificaciones, obtener información de las mismas. Generación del código y construcción del mismo. Éste será desarrollado en el lenguaje PHP. (*Punto 1.3.2 del Diagrama WBS*)
  - *Interfaz de usuario:* construcción de una interfaz de usuario desarrollada con **WAINE** compuesta por varios formularios para realizar el envío de notificaciones al gusto del usuario. (*Punto 1.3.3 del Diagrama WBS*).
  - Además, de estos tres componentes, también se requiere la construcción de una base de datos, donde se guardara toda la información acerca de las notificaciones.
- *Implantación* y aceptación del sistema: constituye el resto de pruebas del sistema (Implantación y aceptación) y la incorporación del sistema creado al entorno de operación. (*Punto 1.4.1, 1.4.2 del Diagrama WBS)*.
- *Cierre:* este apartado implica la elaboración de la memoria del proyecto, que recogerá todos los puntos comentados anteriormente, posibles mejoras, aspectos abiertos y las conclusiones de este trabajo. (*Punto 1.5.1, 1.5.2 del Diagrama WBS*).

3.3 Planificación temporal

En esta sección se realiza una descripción de cada una de las fases del TFG y de las actividades que la componen. Para cada una de estas actividades se indicará su periodo de realización y en que han consistido. Además, se adjunta un diagrama de Gantt para sintetizar de forma clara la planificación temporal.

El proyecto se compone de seis fases:

1. Formación
2. Análisis.
3. Diseño.
4. Construcción
5. Implantación.
6. Cierre.

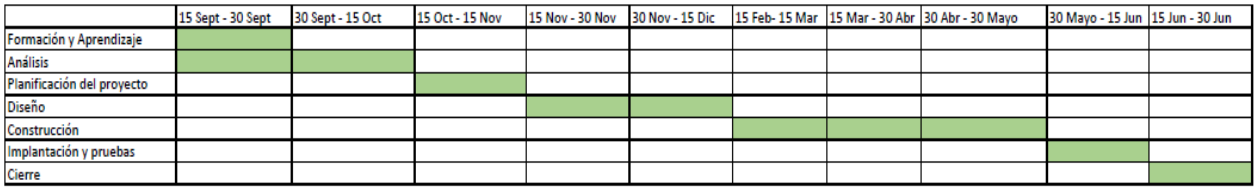

Ilustración 12: Planificación temporal

A continuación, se detallan en los siguientes apartados cada uno de las fases en los que se divide el trabajo.

3.3.1 Formación y aprendizaje

Esta fase es la dirigida a la formación previa necesaria para entender la plataforma de desarrollo WAINE, así como información necesaria para entender la comunicación entre las servicios de notificaciones y familiarizarnos con el lenguaje de programación:

- **Lectura de documentación:** documentación disponible para el entorno de desarrollo **WAINE**, y también sobre metodología V3
- **Formación básica** en PHP [3]

3.3.2 Análisis

Se detalla el servicio de notificaciones wnotif, diagramas y esquemas del servicio, requisitos que la aplicación debe de cumplir y especificaciones de las interfaces de usuarios, roles que se aplican y funcionalidades de cada rol.

El documento de análisis ha sido proporcionado por el tutor de este trabajo, en este documento además de todo lo mencionado anteriormente, se tratan la situación inicial y los pasos a seguir para el propio desarrollo de la aplicación.

Durante este periodo se **estudiaron los requisitos** y se empezó a planificar un diseño del servicio de notificaciones, entendiendo y analizando cada una de las partes definidas en el documento de análisis.

Así también, en este periodo se realiza la **redacción del plan de proyecto** de este servicio de notificaciones. Documento que consta de los recursos, planificación temporal, diagrama WSS así como los anexos y objetivos del mismo.

3.3.3 Diseño

En esta fase se realiza el diseño del modelo físico de datos, la arquitectura del sistema y de las interfaces de usuario del servicio de notificaciones.

Se realizan diferentes diagramas UML[4] para describir las diferentes artefactos que definen y componen este servicio.

Se realiza la redacción del **documento de diseño** donde se especifican; el modelo físico de datos a través del diagrama E/R proporcionado, la arquitectura del sistema, diagrama de clases, la definición de las interfaces de usuario utilizando tablas rol/funcionalidad y unidades de interacción, definición de casos de usos,y se detalla plan de pruebas técnico.

3.3.4 Construcción

En esta fase se realiza la codificación del servicio, desarrollando el código PHP, SQL y ASL.

Además de redactan manuales de usuario.

- **Implementación código SQL**: realización del código SQL que incluye las tablas y la vista utilizada. Además, se incluye la realización de un código con la carga inicial de datos
- **Implementación código PHP.** Construcción de las clases y servicios para la implementación de este servicio de notificaciones. La construcción de la API se realiza en este punto siendo el más complejo y denso del trabajo.
- **Implementación código ASL:** realización del código ASL del módulo en base a la tabla rol-funcionalidad y a los diagramas Entidad-Relación, anotados para implementar las distintas interfaces de usuario.
    - Para ellos se realiza una **formación previa y prácticas** sobre el lenguaje ASL y la construcción de interfaces de usuario [5].
- Redacción de **manuales de usuario**.
- Redacción del **documento de construcción**. Documento con el código desarrollado, tanto del servicio de notificaciones como de las interfaces de usuario y copia de los manuales de usuario.

3.3.5 Implantación

Esta fase se dedica al despliegue del servicio de notificaciones y la realización de pruebas de implantación.

Tiene como objetivo la aceptación del sistema en su totalidad, y la realización de las pruebas necesarias para el paso a producción del mismo.

Consiste en la preparación del entorno, la instalación de los componentes, la activación de los procedimientos tanto automáticos y manuales asociados y cuando proceda, la migración o la carga de datos. Se realizan las pruebas de implementación (tiempos de respuesta deseados, condiciones extremas, seguridad e implementaciones en otros sistemas) y aceptación (ajuste de las necesidades por parte de los usuarios) obteniendo como punto de partida las pruebas realizadas en el proceso de construcción del sistema.

Redacción del **documento de implantación** con toda la información de lo descrita anteriormente.

3.3.6 Cierre

En esta fase se incluyen las tareas finales para la conclusión del proyecto de TFG.

- **Redacción de la memoria.** Unificación de todos los capítulos/documentos mencionados anteriormente.
    - En ella recapitularemos un capitulo final de conclusiones y lineas de avance del proyecto para futuros trabajos relacionados.
- **Redacción de la presentación:** presentación utilizada para la defensa del TFG.

- **Defensa del trabajo.** Exposición del mismo y defensa del TFG.

3.4 Recursos

A continuación detallamos los recursos materiales (hardware y software) y recursos humanos utilizados durante la realización del trabajo.

3.4.1 Recursos hardware

- masai.us.es (193.147.162.156)
  - Dell PoweEdge 860
  - CPU
  - RAM
    - 2 x 1GB 2RX8 NANYA PC2-5300E (NT1GT72U8PB0BY-3C)
    - 2 x 1GB 2RX8 Kingston PC2-5300E (KD6502-ELG)
  - Almacenamiento
    - 2 x 2TB Western Digital Red (WD20EFRX)
    - WCC4M2FEJT74
    - WCC4M5KTJKS6
- *waine.us.es (193.147.162.180)*: Servidor utilizado para la realización de tareas relacionadas con aplicaciones Waine.
  - Máquina virtual
  - Servicios
    - SSH (22)
    - Web (80)

3.4.2 Recursos software

- *Dia 0.97.2, Creately (web), draw.io(web)*: realización de diagramas estructurados.
- *ArgoUML :* realización de diagramas UML
- *Pencil 2.0.5:* herramienta software para la creación de mockups de interfaces, diagramas de flujo y diagramas.
- *LibreOffice7,2,5,2*: procesador de texto, compatible con la mayoría de las herramientas ofimáticas.
- *Notepad++ v7.1*: editor de texto y de código fuente libre con soporte para varios lenguajes de programación.
- *Openproj 1.4*: software de administración de proyectos, incluye diagramas de Gantt, gráfico PERT, diagramas de estructura de descomposición del trabajo (WBS), informe de uso de tareas y muchas utilidades más.
- *TaskCoach 1.4.3*: gestor de tareas de código abierto simple para realizar un seguimiento de

tareas compuestas.

- *https://tohtml.com/:* syntax highlighting para la memoria.
- *WinSCP 5.9.4*: cliente SFTP de código abierto y libre. Cliente FTP para Windows. Realiza copias seguras de archivos entre una máquina remota y máquina local.
- *KiTTY 0.76.1.3: fork* de la versión 0.76 de PuTTY, que utiliza el protocolo telnet/ssh para conectarnos a una máquina remota para Windows. Utiliza el núcleo PuTTY y tiene como objetivo mejorar la experiencia añadiendo nuevas funciones.

3.4.3 Participantes

- Antonio Luis Delgado González
- Ana Varela Rodríguez

4 Diseño de wnotif

Ana Varela Rodríguez, anavarelarod@gmail.com 30/06/2023

4.1 Introducción

El presente documento expone una descripción detallada del sistema wnotif. Este diseño sigue la metodología de planificación, desarrollo y mantenimiento de sistemas de información ***métrica v3 [6].***

El sistema de notificaciones está compuesto por tres subsistemas principales: API, Servicios de notificaciones e interfaces de usuario, los cuales abordamos de manera independiente.

Nos centraremos en cada una de ellas, definiendo su arquitectura, entorno y especificación de componentes.

A partir de la información que se aborda en este documento, tendremos definido:

- La arquitectura del sistema, donde definimos el particionamiento físico del sistema de información, la ubicación de cada subsistema, así como la especificación detallada de infraestructura tecnológica necesaria.
- El modelo físico de datos, se expone la base de datos relacional, las tablas, atributos así como la relación entre las tablas definidas (claves).
- El diseño de las interfaces de usuario, se definen mediante formularios las diferentes interfaces de usuario que se van a utilizar para el desarrollo de este sistema. A través de ellas el usuario (regular o administrador) podrá efectuar las acciones definidas por este servicio de notificaciones.
- Casos de uso, los cuales determinan todas las actividades que pueden llevar a cabo cada entidad externa (administrador, usuario, agente o notificado) los procesos de este servicio.
- Clases, se especifican las clases a través de un diagrama de clases, detallando cada método y atributos de las mismas, así como la función de cada una.
- Especificaciones de construcción
- Migración de datos, los cuales serán necesarios para el funcionamiento inicial del sistema y se muestran los archivos *.sql utilizados para la carga de datos.
- Plan de pruebas. Se plantea un plan de pruebas técnico, que recoge los objetivos del sistema, establece y coordina la estrategia de trabajo, y provee del marco adecuado para planificar paso a paso las actividades de prueba.
- Por último, las especificaciones de implantación que son los requisitos previos que debe de tener un sistema para la futura instalación del servicio.

4.2 Arquitectura del sistema

En el diagrama de despliegue que se muestra, podemos observar los elementos físicos en los que se divide el sistema, y los demás componentes que forman parte de este proyecto.

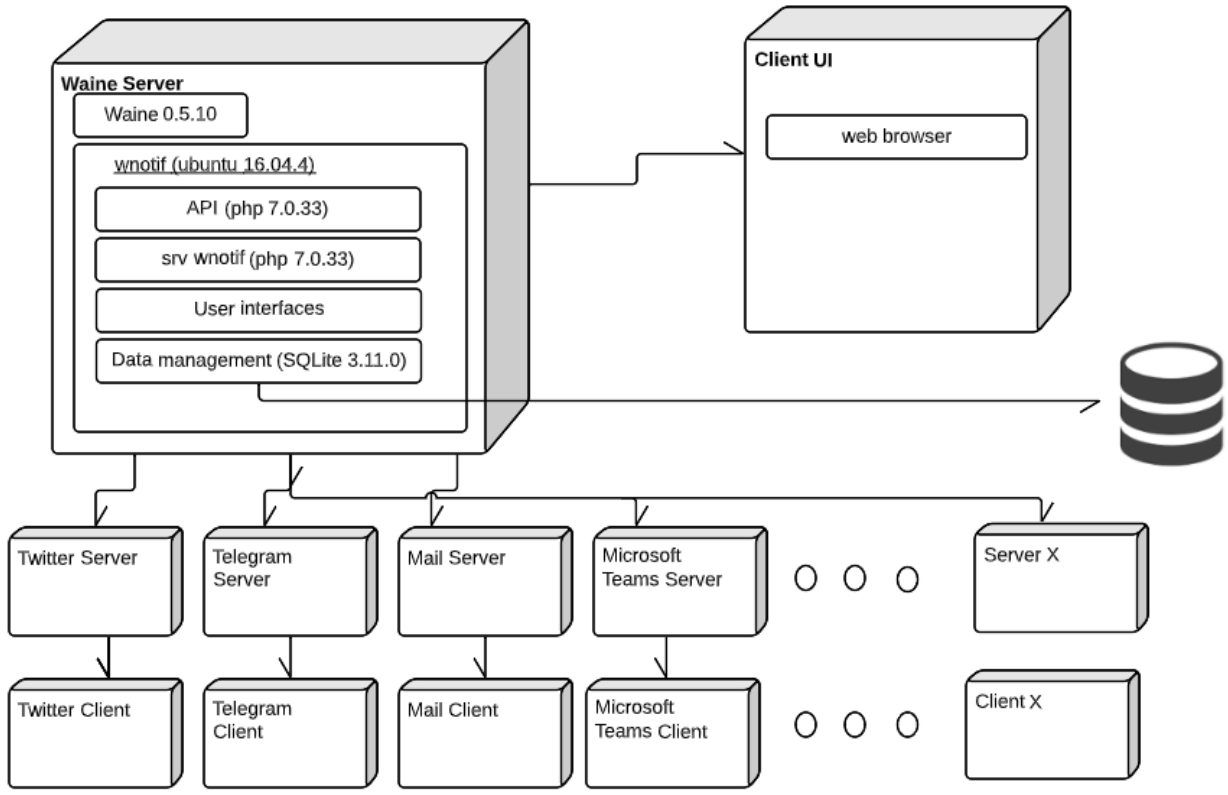

Ilustración 13: Diagrama de despliegue

En primer lugar tenemos un **servidor** **WAINE**, este componente de notificaciones ha sido creado con **WAINE**. La versión de este servidor es la versión 0.5.10.

En el nodo servidor se encuentran los siguientes componentes de estos servicios, los cuales vamos a definir de forma independiente:

- **API**, se desarrollará en una versión de PHP 7.0.33. Ésta permitirá **solicitar** el envío de notificaciones y consultar sobre el estado de las mismas, este programa interactúa únicamente con la aplicación (cliente) y con la base de datos para consultar y guardar la información correspondiente.
- **Servicio de notificaciones.** Su objetivo principal es el envío de las notificaciones, lo lleva a cabo comunicándose con los servidores de las aplicaciones correspondientes, el servidor ejecuta Apache, PHP 7.0.33 y el motor **WAINE** versión 0.5.10.
- **Servidor de base de datos**, se desarrolla en el lenguaje sql, que es un sistema de gestión de base de datos relacional. PHP incluye SQLite en su última versión, compatible con **WAINE**
- **Cliente**: navegador web,que representará interfaces de usuario generadas por el motor **WAINE**, por el cual el usuario detiene, reanuda, observa estado, edita permisos, consulta y anula notificaciones (usuario administrador) o envía/consulta notificaciones (usuarios regulares). Las aplicaciones desarrolladas con **WAINE** son compatibles con los principales navegadores como son Chrome, Firefox y Microsoft Edge

Conectados con el servidor, tendremos los diferentes servicios que hacen uso de este componente de notificaciones (Telegram, Twitter, Mail...) se detallarán más adelante en los siguientes apartados de este mismo documento.

4.3 Modelo físico de datos

En esta sección se define la estructura física de datos que utilizará el sistema, para una mayor eficiencia en el tratamiento de los datos.

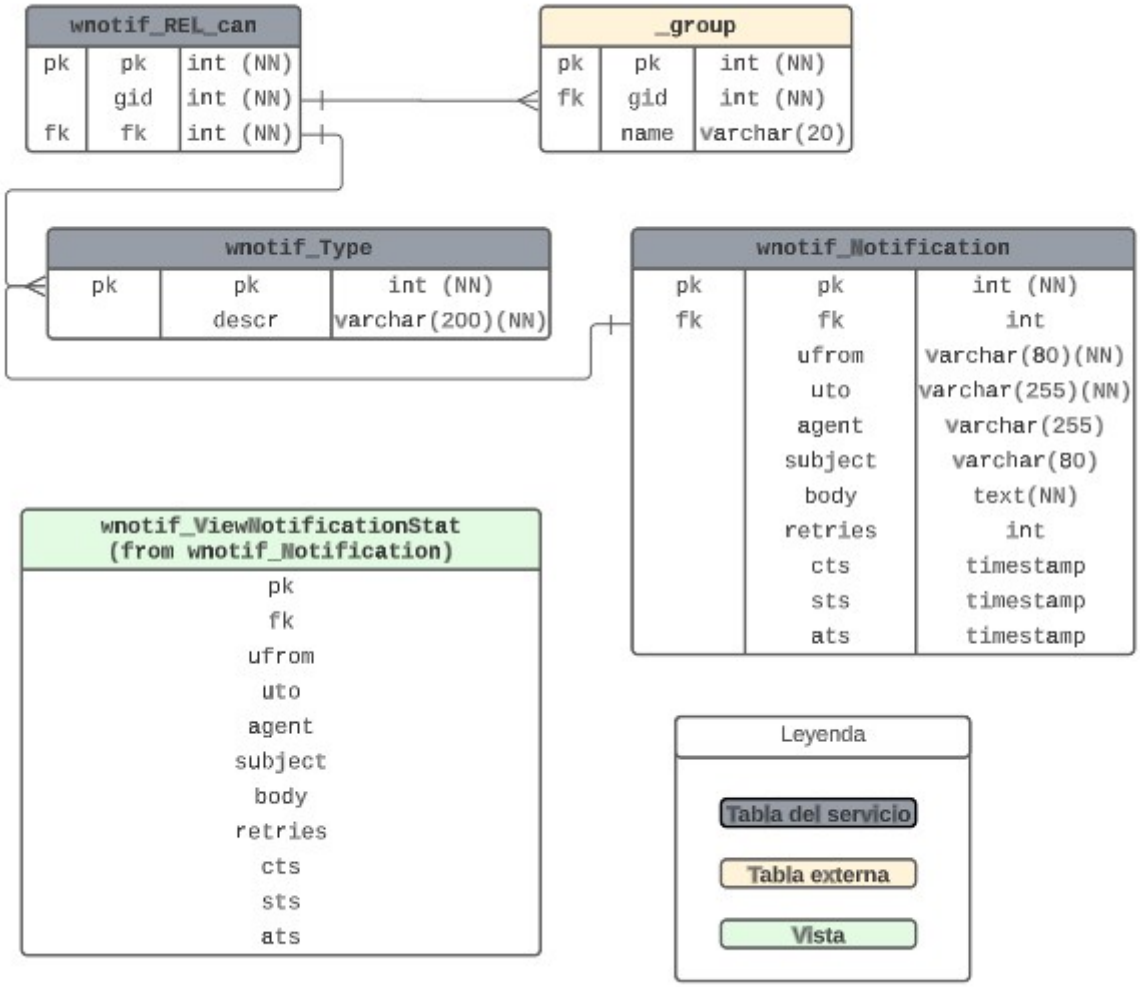

Ilustración 14: Modelo físico de datos

Identificamos tres tablas principales del sistema:

- **wnotif_Notification:** contiene todas las notificaciones realizadas y por realizar, las erróneas y las anuladas. Sus columnas son las siguientes:
  - *pk*: identificador único que enumera a cada notificación. Clave primaria. Tipo INTEGER.
  - *fk:* identificador que representa el tipo de notificación de la que se trata. Clave foránea (pk de la tabla wnotif_Type). Tipo INTEGER.
  - *from*: remitente de la notificación. Campo obligatorio. Tipo VARCHAR.

- *uto*: destinatario de la notificación. Campo obligatorio. Tipo VARCHAR .
- *agent*: agente de la notificación. Tipo VARCHAR.
- *body*: cuerpo de la notificación. Campo obligatorio. Tipo TEXT.
- *subject*: asunto (opcional). Tipo VARCHAR .
- *retries*: Intentos de envío de la notificación. Valor por defecto tres intentos. Tipo INTEGER
- *cts:* instante de la creación. Tipo TIMESTAMP.
- *sts:* instante de envío. Tipo TIMESTAMP.
- *ats*: instante de anulación. Tipo TIMESTAMP.

- **wnotif_Type:** contiene la descripción de todos los tipos de notificación soportados. Compuesta por las columnas:
  - *pk*: identificador único que representa a cada tipo de notificación. Clave primaria. Tipo INTEGER.
  - *description:* breve descripción del tipo, p.e: SMS, Email, Jabber, twitter, etc. Tipo VARCHAR.
- **wnotif_RELcan:** contiene la relación entre un grupo de usuarios (tabla propia de wnotif _group) y los tipos de notificaciones que puede enviar.
  - p*k:* identificador único que representa cada tipo de relación existente entre un determinado grupo y un tipo de notificación. Clave primaria. Tipo INTEGER.
  - *fk*: identificador que indica el tipo de notificación de la que se trata. Clave ajena. Tipo INTEGER
  - *gid*: entero que indica el identificador del grupo. Clave ajena. Tipo INTEGER.
- **wnotif_ViewNotificationStat:** Vista de la tabla *wnotif_Notification*, consulta predeterminada que se guarda como un objeto en la base de datos. Concretamente, selecciona los campos: pk, fk, ufrom, uto, agent, subject, body, retries, cts, sts y ats. Filtramos notificaciones dependiendo de las siguientes consultas (de manera secuencial) y asignamos un estado (seteamos un atributo *estado* en la vista) en cada una de ellas con los valores:
  - *Fallida*: El campo retries es igual a tres, es decir, se han agotado los intentos de envío.
  - *Anulada*: El valor ats (anullation timestamp) no es nulo, luego la notificación ha sido anulada.
  - *Enviada*: Si el valor sts (sent timestamp) no es nulo.
  - *Pendiente*: Si el valor de cts (creation timestamp) no es nulo.

4.4 Diseño de las interfaces de usuario

Para su uso en las aplicaciones **WAINE** se diseñan las siguientes unidades de interacción para los siguientes usuarios Administrador, y usuario regular

4.4.1 Administrador

Se definen a continuación las interfaces de usuario dirigidas al administrador del servicio.

Este usuario tiene el poder de configurar el tipo de notificación que un usuario/grupo puede enviar, realizar consultas de notificación, detener/reanudar el servicio, visualizar el estado de este mismo y anular notificaciones antes de su envío.

4.4.1.1 IU_state. Estado del servicio

Esta interfaz de usuario indica al administrador el estado de ejecución del servicio. Presenta dos posibles mensajes: "EN EJECUCIÓN" o "DETENIDO".

Rol	IU_state
Propósito principal	Mostrar al usuario el estado del servicio.
Tarea encomendada	Presentar un mensaje por pantalla dependiendo del estado del servicio, "EN EJECUCIÓN" o "DETENIDO".
Usuario implicado	Usuario administrador
Responsabilidad	• Los mensajes deberán actualizarse constantemente, comprobando periódicamente la existencia del fichero de lock, el cual existirá si el servicio está en ejecución y no existirá si se encuentra detenido.
Responsabilidad (sobre qué interfaz/tabla es responsable)	• Fichero de lock
Relación con otras interfaces/tablas	• Fichero de lock. La clase *wnotif_LockFile* será la encargado de crear este fichero una vez se active el servicio y eliminarlo una vez que se detenga.

Tabla 1: Rol/funcionalidad. IU_state

4.4.1.2 IU_admin. Administración de permisos

Esta unidad de interacción permite al administrador asignar a cada grupo de usuarios los distintos tipos de notificaciones que puede enviar.

Se trata de un contenedor de tipo compuesto (relation).

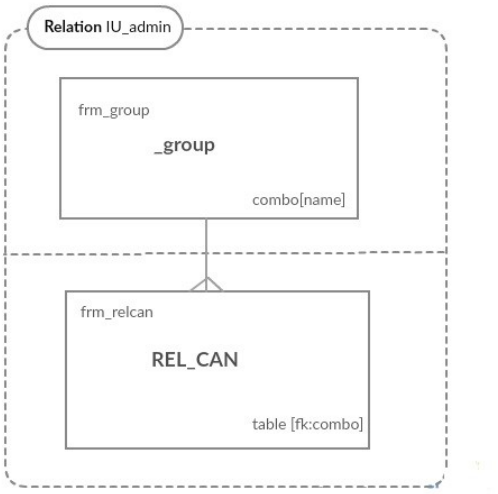

Ilustración 15: Unidad de Interacción. Administración de permisos.

Relaciona dos formularios simples *frm_group* y *frm_relcan*. En el modelo se indica la relación uno a muchos que relacionan ambos formularios.

- *frm_group:* formulario formado por el campo *name* de la tabla *_group*, el cual indica el nombre del grupo.
- *frm_relcan:* formulario compuesto por el campo *fk* de la tabla *REL_CAN* explicada en el apartado tres de este documento, indica el tipo de notificación al cual queremos modificar o eliminar, administrando así los permisos de cada grupo.

Rol	**IU_admin**
Propósito principal	Asignar tipo de notificaciones a grupos de usuarios.
Tarea encomendada	Permite al administrador asignar a cada grupo de usuario los distintos tipos de notificaciones que puede emitir.
Usuario implicado	Usuario administrador
Responsabilidad	• Mostrar la información correspondiente a cada grupo indicado en el combo principal, pudiendo así modificar, eliminar o añadir tipos de notificaciones a al grupo.

Responsabilidad (sobre qué interfaz/tabla es responsable)	• Tabla *wnotif_RELcan*
Relación con otras interfaces/tablas	• Tabla *wnotif_Type*

Tabla 2: Rol/funcionalidad. IU_admin

4.4.1.3 IU_qryremnotif. Consulta y anulación de notificaciones

En esta unidad de interacción un usuario con privilegios de administrador puede consultar notificaciones (enviadas, pendientes, erróneas y anuladas) y dentro del resultado obtenido en la consulta, anular las que considere oportunas antes de su envío.

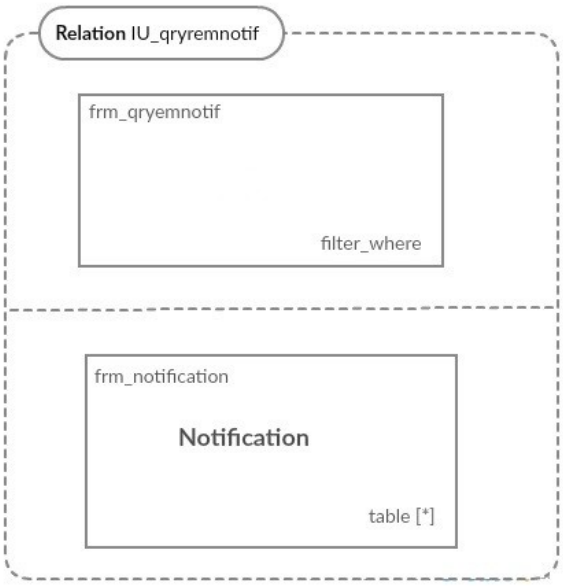

Ilustración 16: Unidad de Interacción. Consulta y anulación de notificaciones

Se trata de un formulario compuesto (relation) *IU_qryrenotif,* formado por dos formularios simples *frm_qryrenotif* y *frm_notification.*

- *frm_qryrenotif:* formulario de tipo filter_where el cual contiene varios combos personalizables para buscar la notificación.
  - Este filtro contiene los campos:
    - *Estado:* estado de la notificación, puede contener los valores pendiente, enviada, errónea y anulada.

- *Agente*: agente que solicita el envío.
- *Usuario*: usuario que solicita el envío.
- *Tipo de notificación*: tipo de notificación.
- *Fecha de inicio*: fecha de creación de notificación inicial para la búsqueda.
- *Fecha de fin:* fecha de creación de notificación final para la búsqueda.

Todos estos campos son opcionales.

- *frm_notification:* Contiene todos los campos de la tabla *wnotif_Notification* comentada en el apartado tres de este documento.

Rol	**IU_qryremnotif**
Propósito principal	Consultar y anular notificaciones.
Tarea encomendada	• Anular cualquier tipo de notificación (de agente o usuario), antes de que se haya producido el envío. • Realizar consultas sobre cualquier tipo de notificación (enviados, pendientes, erróneos y anulados) filtrando por varios campos.
Usuario implicado	Usuario administrador
Responsabilidad	• Mostrar información acerca de la notificación a consultar (id, tipo, agente, cts, sts, destinatario, cuerpo). • Al pulsar el botón “Anular” la interfaz deberá anular la notificación.
Responsabilidad (sobre qué interfaz/tabla es responsable)	• Tabla *wnotif_notification*
Relación con otras interfaces/tablas	• Tabla *wnotif_Type* y *wnotif_RELcan*

Tabla 3: Rol/funcionalidad. IU_qryemnotif

4.4.2Usuario regular

Se muestra la interfaz de usuario orientada para el usuario regular.

Este tipo de usuario tiene la opción de insertar y consultar información sobre las notificaciones.

4.4.2.1 IU_sendnotif. Envío de notificaciones

Formulario de envío de notificaciones que permite al *usuario regular* solicitar el envío de notificaciones siempre dentro de los tipos permitido por el usuario administrador.

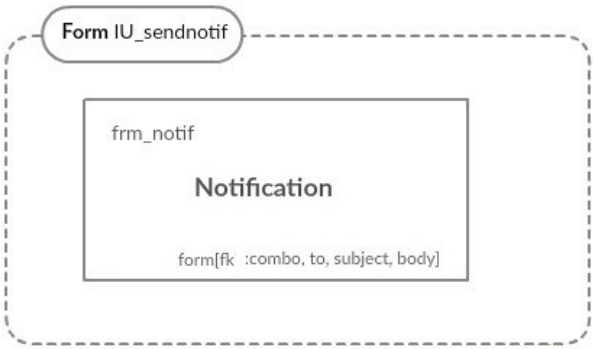

Ilustración 17: Unidad de Interacción. Envío de notificación.

Imagen 5: Unidad de Interacción. Envío de notificación.

Está compuesta por un formulario simple (form) *IU_sendnotif*, cuyo formulario *frm_notif* está formado por los siguientes atributos de la tabla **wnotif_Notification**:

- *fk (tipo de notificación):* este campo será un combo, en el cual el usuario puede elegir el tipo de notificación que desea enviar.
- *to:* destinatario de la notificación.
- *from*: origen de la notificación
- *subject:* Asunto del mensaje.
- *body:* Cuerpo del mensaje.

Rol	**IU_sendnotif**
Propósito principal	Solicitar envío de notificaciones (dentro de los tipos permitidos).
Tarea encomendada	A través de la inserción de datos suministrados por el usuario (tipo de notificación, destino, asunto y cuerpo), éste puede solicitar el envío de una notificación a un sistema externo.
Usuario implicado	Usuario regular
Responsabilidad	• Realizar un envío seguro. • Proporcionar un identificador al usuario, asignado a su notificación.
Responsabilidad (sobre qué	• Tabla *wnotif_notification*

interfaz/tabla es responsable)	
Relación con otras interfaces/tablas	• Tabla *wnotif_Type* y *wnotif_RELcan*

Tabla 4: Rol/funcionalidad. IU_sendnotif

4.4.2.2 IU_query. Consulta de notificaciones

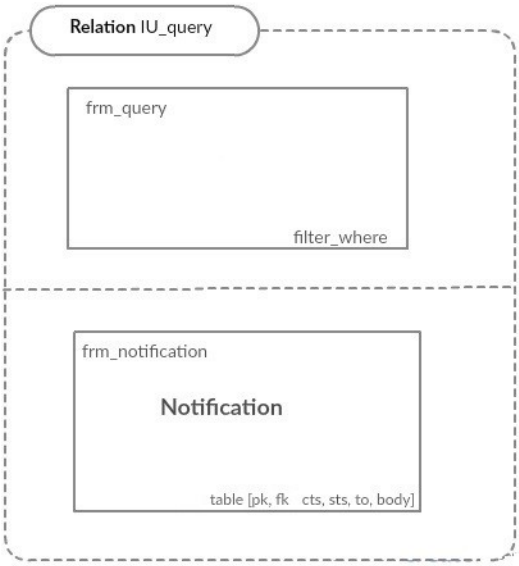

Ilustración 18: Unidad de Interacción. Consulta de notificación.

Interfaz que permite a los usuarios regulares consultar las notificaciones que se han emitido.

Está compuesto de dos formularios simples *frm_notification* y f*rm_query*.

- *frm_query*: todos los campos de este filtro son opcionales, donde tenemos el estado de la notificación (pendiente, enviada, errónea y anulada), el tipo de notificación, la fecha de inicio y la fecha de fin.
- *frm_notification*: este formulario contiene los campos pk, fk, cts, sts, to y body de la tabla Notification.

Rol	**IU_query**
Propósito principal	Consultar notificaciones.
Tarea encomendada	• Permite al usuario realizar consultar sobre las notificaciones que él ha realizado.

	• Puede realizar varios filtros sobre la consulta.
Usuario implicado	Usuario regular
Responsabilidad	• Retornar la información (id, tipo, cts, sts, destinatario, cuerpo) solicitada por el usuario.
Responsabilidad (sobre qué interfaz /tabla es responsable)	• *Tabla wnotif_Notification*
Relación con otras interfaces/tablas	• Tablas *wnotif_Notification* y *wnotif_Type*

Tabla 5: Rol/funcionalidad. IU_query

4.5 Casos de uso reales

En esta sección se representan los diferentes casos de uso que definen al servicio wnotif. Se han definido algunos de ellos en el siguiente diagrama de casos de uso.

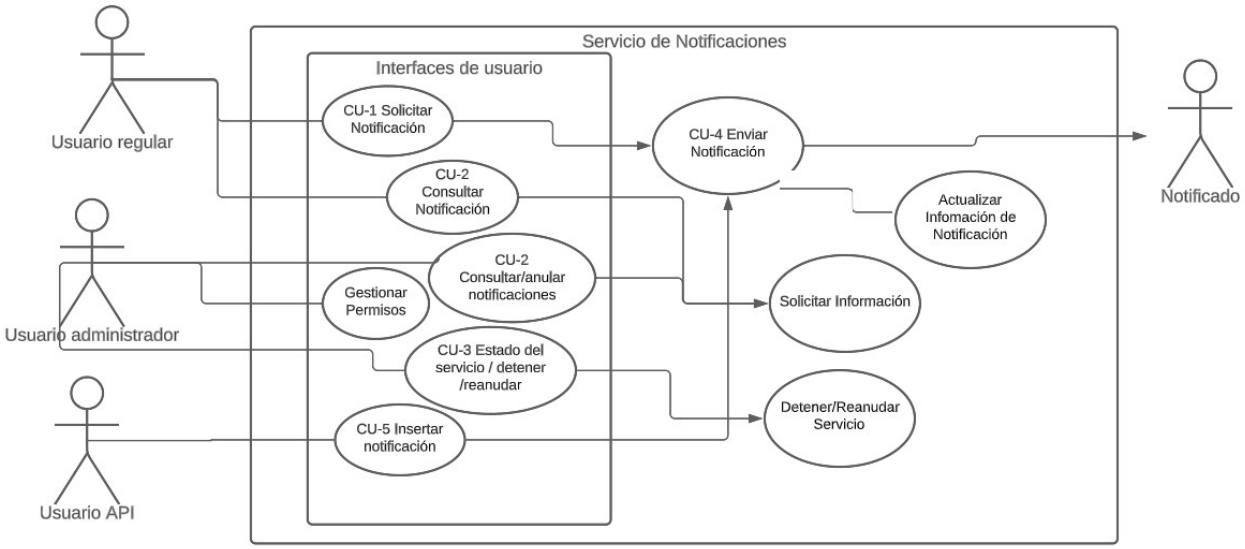

Ilustración 19: Diagrama de casos de uso de wnotif

4.5.1 Usuarios

4.5.1.1 Solicitar Notificación (Usuario regular)

<table>
<tr><th>CU-1</th><th>Solicitar notificación</th></tr>
<tr><td>Dependencias</td><td>R.3.1.1. Puede enviar notificaciones de los tipos autorizados por el administrador</td></tr>
<tr><td>Precondición</td><td>El agente/usuario debe tener previamente permisos para enviar la notificación del tipo indicado.</td></tr>
<tr><td>Descripción</td><td>El agente/usuario desea realizar el envío de una notificación.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><td></td><td>Acción</td></tr>
<tr><td></td><td>Usuario inicia sesión en la interfaz con su usuario/contraseña.</td></tr>
<tr><td></td><td>Hace clic en “Envío de notificaciones”</td></tr>
<tr><td></td><td>Selecciona el tipo de notificación y los campos obligatorios.</td></tr>
<tr><td></td><td>Usuario hace clic en el botón “Añadir”.</td></tr>
<tr><td></td><td>La notificación se ha añadido a la base de datos correctamente y devuelve un id al usuario.</td></tr>
<tr><td></td><td>Si el usuario/agente desea solicitar una notificación de nuevo vuelva al paso 2.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>La notificación queda registrada y con estado “pendiente de envío”.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>3</td><td>Si el usuario no rellena los campos obligatorios y hace clic en “Añadir”, aparecerá un mensaje indicando que los campos no pueden estar vacíos.</td></tr>
</table>
</td></tr>
<tr><td>Comentarios</td><td>No existen ninguna limitación en cuanto al número de solicitudes de servicio que un usuario puede realizar.</td></tr>
</table>

Tabla 6: CU-1 - Solicitar notificación

4.5.1.2 Consultar Notificación

CU-2	Consultar Notificación
Dependencias	R3.1.2.1 Puede consultar únicamente sobre notificaciones que el usuario ha realizado. R3.1.2.2 Puede realizar varios filtros sobre la consulta

<table>
<tr><td></td><td>R3.2.4.1 El administrador puede realizar consultar sobre mensajes enviados, pendientes erróneos y anulados.
R3.2.4.2 Puede realizar varios filtros sobre la consulta.</td></tr>
<tr><td>Precondición</td><td>La notificación debe existir previamente para poder consultarla.</td></tr>
<tr><td>Descripción</td><td>El agente/usuario/administrador desea realizar una consulta de una o varias notificaciones.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>Usuario inicia sesión en la interfaz con su usuario/contraseña.</td></tr>
<tr><td>2</td><td>
<table>
<tr><td>2.1</td><td>En caso del administrador, éste hace clic en “Consulta/anulación de notificaciones.</td></tr>
<tr><td>2.2</td><td>En caso de usuario regular, éste hace clic en “Consulta de notificaciones”</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>Indica dentro del combo los datos a consultar (Estado, Fecha, Usuario o Tipo de Notificación)</td></tr>
<tr><td>4</td><td>Usuario hace clic en el botón “Filtrar”.</td></tr>
<tr><td>5</td><td>Si el usuario/agente/administrador desea cambiar los datos de consulta, basta con cambiarlos y hacer clic de nuevo en “Filtrar”</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Una lista de notificaciones se despliega con la información correspondiente (Id, Tipo, Agente, Usuario, Creado, Enviado, Para, Asunto.</td></tr>
<tr><td>Excepciones</td><td></td></tr>
<tr><td>Comentarios</td><td>El usuario administrador podrá realizar consultas sobre cualquier tipo de notificación, mientras que el usuario regular solo puede consultar las que él ha emitido.</td></tr>
</table>

Tabla 7: CU-2 Consultar notificación

4.5.1.3 Detener/Reanudar el servicio (usuario administrador)

<table>
<tr><th>CU-3</th><th>Detener/Reanudar Servicio</th></tr>
<tr><td>Dependencias</td><td>R1.2.1 Puede ser detenido
R1.2.2 Puede ser reanudado</td></tr>
<tr><td>Precondición</td><td>Ninguno</td></tr>
<tr><td>Descripción</td><td>Envío de Notificaciones</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario Administrador Inicia Sesión</td></tr>
<tr><td>2</td><td>El usuario hace clic en el botón “Detener el servicio”
<table><tr><td>2.1</td><td>En caso de querer reanudar el servicio el usuario hace clic en “Reanudar el servicio”</td></tr></table></td></tr>
<tr><td>3</td><td>Un prompt aparece preguntando al usuario si desea detener/reanudar el servicio. El usuario hace clic en “Si”</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El servicio se reanuda/detiene.</td></tr>
<tr><td>Excepciones</td><td>Si el servicio ya se encuentra activo en caso de reanudarlo, aparecerá un mensaje indicando “El servicio ya se encuentra activo” y lo contrario si el servicio ya estaba previamente detenido.</td></tr>
<tr><td>Comentarios</td><td>Se puede comprobar el estado del servicio previamente en la sección “Estado del servicio”. Sólo el usuario administrador puede hacer uso de estas funciones.</td></tr>
</table>

Tabla 8: CU-3 Detener/reanudar el servicio

4.5.2API - Notificaciones

4.5.2.1 Enviar Notificación

CU-4	Enviar Notificación
Dependencias	R3.1.2.1 El servicio de notificaciones deberá realizar el envió de las notificaciones.
Precondición	Las notificaciones tienen que haber sido solicitadas previamente. El sistema debe de estar activado.

<table>
<tr><td>Descripción</td><td>Se realizan el envío de notificaciones pendientes.</td></tr>
<tr><td>Secuencia normal</td><td><table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El servicio lee de la base de datos las notificaciones con estado “Pendiente de enviar”</td></tr>
<tr><td>2</td><td>Se identifica el tipo de notificación y se envía a su correspondiente servicio para gestionar el envío.</td></tr>
<tr><td>3</td><td>El servidor indicado procede a enviar la notificación.</td></tr>
<tr><td>4</td><td>La notificación en caso de fallo, vuelve a enviar hasta un máximo de tres reintentos.</td></tr>
</table></td></tr>
<tr><td>Post-condición</td><td>Se escribe en el log toda la información de la notificación (envío, anulación, errores).
La API devuelve el identificador de la notificación tras el envío.
El estado de la notificación cambia a “Enviada” en caso de éxito o “Fallida” en caso de error.</td></tr>
<tr><td>Excepciones</td><td></td></tr>
<tr><td>Comentarios</td><td>Se deberán aportar todos los datos necesarios para poder realizar el envío.</td></tr>
</table>

Tabla 9: CU-4 - Enviar notificación

4.5.2.2 Insertar Notificación (Usuario API)

<table>
<tr><th>CU-5</th><th>Insertar Notificación</th></tr>
<tr><td>Dependencias</td><td>R2.2.2 Envío de notificaciones de cualquier tipo, obteniéndose el id de la misma</td></tr>
<tr><td>Precondición</td><td>Ninguno</td></tr>
<tr><td>Descripción</td><td>Envío de Notificaciones</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>Paso</td><td>Acción</td></tr><tr><td>1</td><td>El usuario API inicia sesión en la interfaz de usuario</td></tr><tr><td>2</td><td>Hace clic en el botón “Insertar notificaciones”</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>Un mensaje aparece por pantalla indicando el id/código de la notificación.</td></tr>
<tr><td>Excepciones</td><td></td></tr>
<tr><td>Comentarios</td><td>El tipo de la notificación insertada es aleatorio.</td></tr>
</table>

Tabla 10: CU-5 - Insertar notificación

4.5.3 Clases

El siguiente diagrama de clases presenta la estructura que tiene el servicio de notificaciones, consta de 4 clases principales (*lwnotif_LockFile, wnotif_LogFile, wnotif_Srv y wnotif_Notification*), e incluyendo las clases correspondientes a las aplicaciones que se encargan de enviar las notificaciones a los notificados. (email, Slack, Telegram, Write...). Se presenta el diagrama y se explica un poco más en detalle la función de cada clase, atributos y métodos (todas estas clases serán especificadas en el capítulo de Construcción):

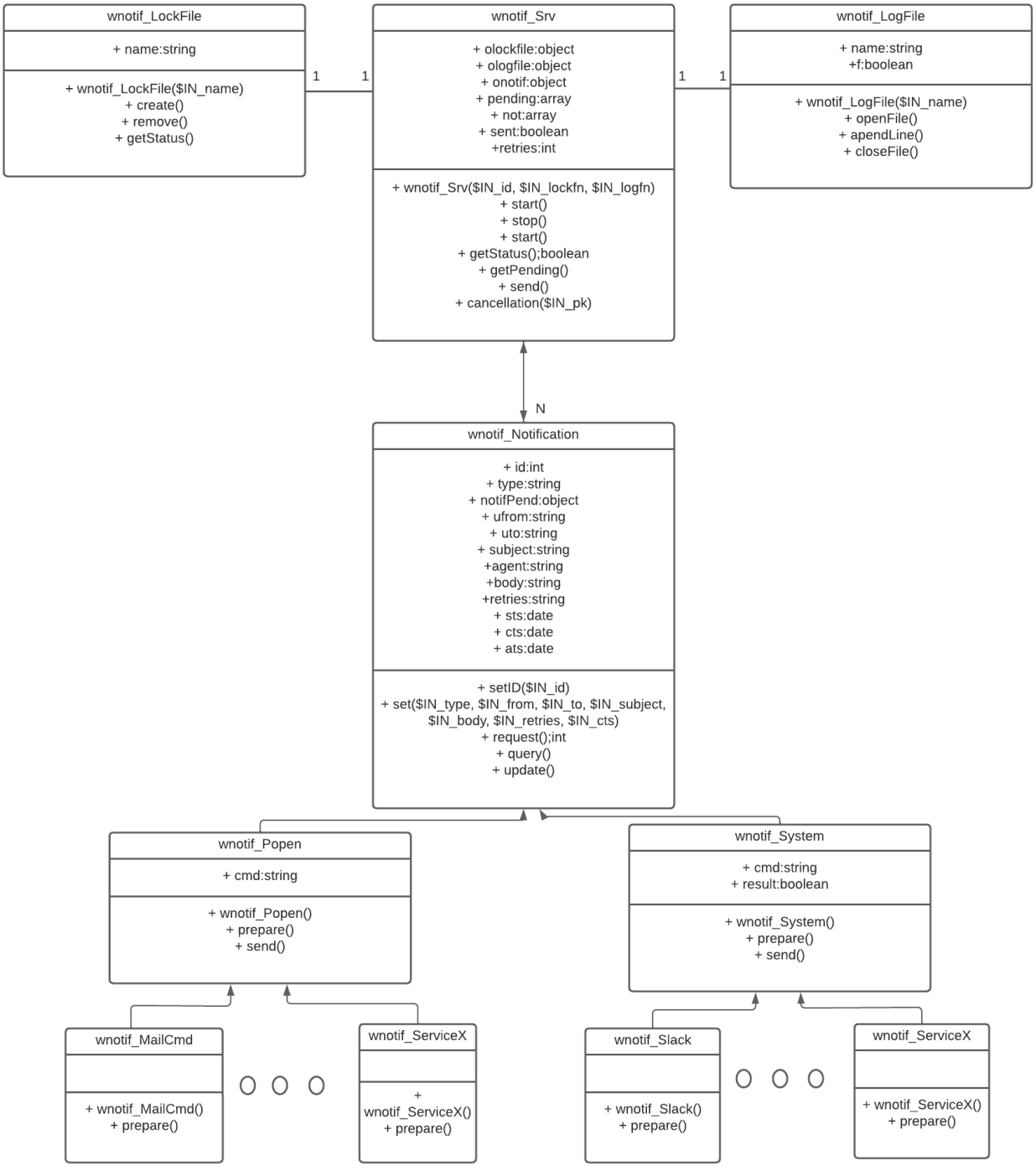

Ilustración 20: Diagrama de clases de wnotif

- **wnotif_LockFile:** *define la detención/reanudación del servicio.* Incluye un atributo "name" , que es el nombre o ruta del fichero que indica el estado del servicio, si se encuentra activo

o detenido Sus métodos son los siguientes:

- `create()`. Se encarga de crear el fichero bloqueo del servicio (si éste no existe) cuando es llamado, y devuelve "true" sí el fichero se ha creado correctamente y "false" en caso contrario.
- `remove()`. Elimina el fichero de bloqueo (si éste existe previamente) cuando es llamado, y devuelve "true" si ha sido borrado con éxito y "false" si lo contrario.
- `getStatus()`. Este método permite conocer el estado de funcionamiento del servicio, comprobando si existe o no el fichero de inicio del servicio. Devuelve "true" si el servicio está activo (existe el archivo) o "false" si el servicio está inactivo (el archivo de bloqueo no existe).

- **wnotif_LogFile: *gestiona la creación y edición de un fichero de log***, en el cual se anotan los envíos realizados, los que no han podido ser realizados, los inicios y paradas del servicio y las anulaciones. Tiene un atributo "name", que es el nombre del fichero y un único método:
  - `appendLine`(). Recibe como parámetro la línea a escribir en el fichero de log y la escribe en él. Devuelve "true" si se ha escrito correctamente y "false" si lo contrario.
- **wnotif_Srv: *se encarga de realizar el envío de las notificaciones a sus correspondientes aplicaciones.*** Define los siguientes atributos y métodos:
  - `onotif`. Objecto que identifica la notificación a gestionar y que utilizaremos para *enviar y definir la notificación* en los servicios/aplicaciones.
  - `not`:(array) notificación en concreto con la que trabajamos al filtrar en el array por tipo de notificación para posteriormente asignar los atributos de éste al objeto onotif.
  - `pending`. *array de notificaciones* obtenidas de la base de datos marcadas como "pendientes" para realizar el envío de éstas.
  - `sent`. Boleano que indica si la notificación ha sido enviada con éxito o no.
  - `retries`. *Número de intentos* para enviar la notificación (tres intentos por defecto).
  - `getPending`(). Obtiene las notificaciones pendientes de envío de la base de datos y las almacena en un *array* (`pending`), para posteriormente seleccionarlas por tipo de *notificación* (`not`), creando cada objeto correspondiente dependiendo del tipo del que se trate (`onotif`).
  - `send`(). Llama al método del mismo nombre de las clases correspondientes al tipo de notificación (*wnotif_Popen y wnotif_System*) indicándole el objeto previamente creado en el método `getPending`(), si no fuera posible realizarlo una vez, se incrementaría el atributo de número de re intentos en el atributo (`retries`), y se volvería a realizar el envío. Si el envío se realiza con éxito, se actualiza al fecha de envío de la notificación (`sts`) mediante el método `update`()
  - `getStatus`(). Llama al método del mismo nombre de la clase wnotif_LockFile, que devuelve el estado de funcionamiento del servicio, en ejecución o parado.
  - `start`(). Llama al método `create`() de la clase *wnotif_LockFile*, el cual crea el fichero de de bloqueo para indicar el servicio en ejecución.
  - stop(). Llama al método remove() de la clase wnotif_LockFile, el cual elimina el fichero

de bloqueo y así detener el servicio.

- **wnotif_Notification(API):** *solicita, consulta y actualiza notificaciones.*

  Contiene los métodos `query`(), `request`() y `update`() para la consulta e inserción de notificaciones.

  - El método `request`() inserta en la base de datos la información proporcionada por el usuario/sistema y devuelve el id de la notificación en caso de éxito.
  - El método `query`() hace consulta de la información de una notificación en concreto, recibiendo el id de ésta. Devuelve un array con la información (origen, destino, cuerpo, asunto…).
  - Método `update`() actualiza la información en la base de datos. La información está previamente actualizada en el objeto de la notificación con el método set().

- **wnotif_Popen:** *realiza el envío de la notificación*. Extiende de la clase *wnotif_Notification* y abre una tubería hacia un proceso ejecutado bifurcando el comando dado por comando ($cmd), siendo 'w' el modo. Envía la notificación con el método *send*().
- **wnotif_System:** *realiza el envío de la notificación* . Extiende de la clase *wnotif_System*. Con el método `send`() ejecuta el comando por consola recibido por las subclases de los servicios ($cmd).
- **wnotif_XXX(Servicio):** *prepara la notificación para el envió* en el servicio correspondiente. extienden de las clases *wnotif_System* o *wnotif_Popen*, dependiendo del tipo de proceso de envío. Con el método *`prepare`*(), preparan el comando a ejecutar que es en sí la sentencia o el comando a ejecutar para proceder con el envío de la notificación.

4.6 Especificaciones de Construcción

En este apartado se generan las especificaciones para la construcción del sistema de información.

Utilizaremos la versión del servidor **WAINE** 0.5.10.

En el desarrollo del sistema de notificaciones utilizaremos el lenguaje de PHP.

Este desarrollo contiene el proceso principal y todas las clases explicadas anteriormente. Todas estas funciones formarán el servicio wnotif y la API, las cuales se conectarán con la base de datos Sqlite3 (incluida en la versión de PHP) para el almacenamiento de los datos en la base de datos.

Para la base de datos trabajaremos con SQLite, el cual es un sistema de base de datos relacional, la ventaja de esta base de datos es que no es un proceso independiente, si no que se enlaza con el programa pasando a ser una parte integral del mismo. Se utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones.

Este servicio está construido sobre la base de **WAINE**.

Para las interfaces de usuario se construye en lenguaje ASL, lenguaje de descripción que utiliza **WAINE** como propio lenguaje de especificación para definir modelos de la interfaces de usuario.

La estructura de estas interfaces de usuario se basa en la utilización de formularios y structs

(simples o compuestos).

Para su implantación, se genera un paquete haciendo uso del sistema de gestión de paquetes de **WAINE**, haciendo uso de la herramienta wpkg que se detallará en el capítulo de implantación.

4.6.1 Migración y carga inicial de datos

En este trabajo no se abarcará ningún tipo de migración, debido a que estamos creando un modelo de datos desde el inicio, únicamente se llevará a cabo la carga inicial de datos de la tabla wnotif_*Type, que se* cargará con los tipos de notificaciones permitidas por nuestro sistema.

Esta tabla contiene los campos *pk* (índice único) y *descr* (descripción de la notificación).

Debido a que utilizaremos la base de datos SQLite, se realizará una inserción de datos a partir de la sentencia INSERT manualmente, o a través de un script que se detallará más adelante.

```
INSERT INTO  wnotif_Type  (pk, descr) VALUES (0, 'test');
```

4.7 Plan de pruebas técnico

En este apartado se plantea el plan de pruebas del servicio.

Se divide por niveles, que serán: pruebas unitarias, integración, de implementación, de sistema y aceptación.

4.7.1 Pruebas Unitarias

Estas constituyen la prueba inicial del sistema , una vez concluida la construcción del ***código***. Principalmente se comprobará que cada clase/métodos sigue el flujo normal de funcionamiento, sin observar que el resultado es el correcto (de momento). Es decir, se introducirá información por separado, para comprobar que cada clase/método por separado realiza su función.

Se han creado las siguientes pruebas para las clases *wnotif_LogFile* y *wnotif_LockFile.*

Todos estos archivos se detallan en el capítulo de construcción del servicio.

- ***wnotif_LogFileTest.php***: se crea un objeto wnotif_LogFile recibiendo como parámetro el nombre del archivo de log a crear al método *appendLine*() pasándole como parámetro la línea a escribir. Como resultado se espera que se cree el archivo y se escriba la/las correspondiente(s) lineas en el archivo de log.
- ***wnotif_LockFIleTest.php:*** se crea un objeto wnotif_LockFile indicando como parámetro el nombre del archivo de bloqueo a crear y se realizarán las siguiente pruebas:
    - Se llama al método *create*() y luego al método *getStatus*(). Como resultado se espera que se haya creado el archivo de bloqueo y que el método devuelva “true”, lo que indica que el archivo de bloqueo existe (y por ende que el sistema se encuentra en funcionamiento).
    - Llamada al método *remove*() y a continuación al método *getStatus*(). Como resultado se espera que se elimine el archivo de bloqueo y el método *getStatus*() devuelva “false”, indicando que el archivo no se encuentra en la ruta indicada (y el servicio se encuentra detenido).

4.7.2 Pruebas de integración

En esta etapa, comprobaremos la comunicación entre clases, así como los datos que se transfieren son los adecuados.

Se han generado los siguientes archivos de pruebas donde se prueban tanto independiente como conjuntamente las diferentes clases que componen el sistema.

Todos estos ficheros serán detallados en el apartado de construcción del servicio.

- ***wnotif_Notification.php:*** con este archivo se prueba la API del sistema. Inserta una notificación con el método *set*() y la solicita con el método *request*().
  - Así como se prueba la actualización de la notificación en base de datos con el método *update*() proporcionando nuevos datos.
  - Se ejecuta el método *query*() para la consulta de la notificación.
- ***wnotif_SrvTest.php:*** crea un objeto wnotif_LockFile para activar el servicio (si no se encuentra previamente activo) y llama al método send(). Este fichero arranca el servicio y lee de la base de datos las notificaciones pendientes de enviar y realiza el envío de éstas. Como resultado óptimo, el envió de las notificaciones pendientes es correcto y el sistema actualiza en la base de datos la información pertinente.

Debido a que el sistema está dividido en módulos, primero se comprueba el funcionamiento de cada componente por separado y posteriormente que se establece la comunicación correcta entre los mismos. (API y base de datos, Servicio de Notificaciones y base de datos, interfaces de usuario y base de datos).

4.7.3 Pruebas de sistemas

Estas pruebas componen el funcionamiento global del sistema de comunicación, luego se comprueban la comunicación entre componente **WAINE** con usuarios “ficticios” y con las aplicaciones, para probar el correcto flujo de notificaciones y que se cumple con las especificaciones del trabajo.

Para ello se realizan pruebas en la interfaz de usuario con los usuarios (user, admin y api) los cuales realizan la prueba. Ésta simula un agente el cual recibe notificaciones y se comunica con la API para el envío/consulta de notificaciones.

El usuario regular (user) prueba solicitar notificaciones y seguir el ciclo de vida de la notificación. Se prueba también la consulta de notificaciones.

Se comprueba el funcionamiento de detención y reanudación del servicio con el usuario admin y la gestión de permisos por el cual el usuario sólo puede solicitar el envío de las notificaciones en los grupos/tipos que el usuario admin le proporciona permisos.

Todas estas pruebas se realizan en un entorno de prueba.

4.7.4 Pruebas de implantación

Una vez realizadas las pruebas en el entorno de desarrollo con componentes auxiliares y usuarios de prueba, se llevan a cabo las pruebas de en el entorno de operación.

La generación de un paquete mediante la herramienta wpkg que contiene todo el servicio de notificaciones será el comienzo.

Este paquete deberá ser instalado en un sistema adaptado para la implantación (cumpliendo con los requisitos de implantación. Una vez realizada la instalación se ejecuta en la máquina y se realizan pruebas en el sistema para comprobar la correcta instalación.

4.7.5 Pruebas de aceptación

Por último serán los usuarios los que deben revisar los criterios de aceptación que se especificaron previamente en el plan de pruebas del sistema y, posteriormente, dirigir las pruebas de aceptación final.

4.8 Requisitos de implantación

En este apartado se señalaran los requisitos necesarios para la implementación del servicio de notificaciones wnotif, requisitos que necesitarán los usuarios para la implantación de este servicio:

- Versión Waine 0.5.10
- Versión PHP superior o igual a 4.3.10
- Versión Apache superior o igual a 1.3.33
- SQLite igual o superior a 3.11.0.

5 Construcción de wnotif

Ana Varela Rodríguez, anavarelarod@gmail.com

5.1.1 Introducción

En este documento se detalla el proceso de implementación seguido para el servicio de notificaciones wnotif. Se identifican los siguientes puntos en este documento:

- **Construcción del modelo físico de datos:** realización del código SQL que incluye las tablas, las vistas. Además, se incluye la realización de un código con la carga inicial de datos.
- **Construcción del servicio de notificaciones.** Definición de clases y ficheros a incluir desarrollados en PHP que realizan la función de envío, administración y configuración de servicios de notificaciones.
- **Construcción de API.** Definición de los archivos que realizan la función del API. Solicitud y consulta de notificaciones.
- **Construcción del código ASL e interfaces de usuario:** definición del entorno de desarrollo de interfaces de usuario basado en el modelo WAINE para generar nuestra aplicación web, definición de modelos.
- **Construcción de pruebas.** Definición de las pruebas unitarias y de sistemas realizados para las clases php.
- **Manuales de usuario:** Se incluyen manual de uso para cada uno de los roles de al aplicación desarrollada.
- **Anexos.** Información adicional

5.2 Construcción del modelo físico de datos

En este apartado detallamos como se ha realizado la construcción del modelo físico de datos del servicio. Esto se corresponde con la definición del modelo de dominio de **WAINE.**

De acuerdo con el apartado de Modelo físico de datos del Documento de Diseño, se han definido las siguientes tablas para el sistema de notificaciones.

Se ha utilizado el lenguaje SQL para trabajar con las tablas que definen el sistema utilizando como base de datos el motor SQLite.

Las tablas y vistas se han definido en un mismo fichero **wnotif_CREATE.sql** que contienen código escrito en lenguaje de consulta estructurada (SQL).

Definimos por partes las siguientes tablas:

5.2.1 Tabla wnotif_Type

Define los tipos de notificación definidos en el sistema. Su código SQL es:

```
CREATE TABLE wnotif_Type (
  pk integer PRIMARY KEY,
  descr varchar(200) NOT NULL
);
```

5.2.1.1 Carga inicial de datos

La carga inicial de datos es necesaria para esta tabla, ya que se indican los tipos de notificaciones que se pueden actualizar en el sistema.

Esto depende de las aplicaciones utilizadas, en este proyecto en concreto se han introducido las siguientes:

```
INSERT INTO  wnotif_Type  (pk, descr) VALUES (0, 'test'), (1, 'Twitter'),
(2, 'Slack'), (3, 'Mailphp'), (4, 'Mailcmd'), (5, 'Telegram'), (6,
'Write'), (7, 'Teams');
```

5.2.2 Tabla wnotif_Notification

Define todos los campos de una notificación. Su código SQL es:

```
CREATE TABLE wnotif_Notification (
  pk integer PRIMARY KEY,
  fk integer,
  ufrom varchar(80) NOT NULL,
  uto varchar(255) NOT NULL,
  agent varchar(255),
  subject varchar(80),
  body text NOT NULL,
  retries integer,
  cts timestamp DEFAULT current_timestamp,
  sts timestamp,
  ats timestamp,

  CONSTRAINT fk_rel_can FOREIGN KEY (fk) REFERENCES wnotif_Type(pk)
);
```

5.2.3 Vista ViewNotificationStat

Vista utilizada para obtener los atributos de la notificación de la tabla *wnotif_Notification* en base a su estado. Su código SQL es:

```
CREATE VIEW wnotif_ViewNotificationStat AS
  SELECT pk,fk,ufrom,uto,agent,subject,body,retries,cts,sts,ats,
         CASE
           WHEN retries=3 THEN "Fallida"
           WHEN ats IS NOT NULL THEN "Anulada"
```

```
            WHEN sts IS NOT NULL THEN "Enviada"
            WHEN cts IS NOT NULL THEN "Pendiente"
            ELSE "Error imposible"
          END AS estado
  FROM wnotif_Notification;
```

5.2.4 Tabla wnotif_RELcan

Define la relación entre los grupos y los permisos de usuarios para qué tipos de notificación. Su correspondiente código SQL es:

```
CREATE TABLE wnotif_RELcan (
  pk integer PRIMARY KEY,
  gid int NOT NULL,
  fk int NOT NULL,

  CONSTRAINT fk_rel_can FOREIGN KEY (fk) REFERENCES wnotif_Type(pk)
 );
```

5.3 Construcción del servicio de notificaciones

En este apartado se definen los diferentes módulos y clases que componen el sistema de notificaciones wnotif, el cual es el encargado de realizar los envíos de las notificaciones.

Se muestra el esquema de ficheros que definimos en este apartado. Todos el código de construcción se encuentran alojados bajo el directorio ***wnotif.bin***.

Se ha elaborado un componente software re utilizable para el envío de basados en distintos ficheros, cumpliendo con los requisitos R.1.1.1 y R1.1.2.

La carpeta ***/tests***, contiene los tests que se comentan en el apartado de pruebas de este documento.

En este apartado definimos todos los ficheros *.inc necesarios para el desarrollo y generación de este servicio de notificaciones.

```
+---etc/
| \---wnotif.bin/
| |   |        wnotif.cfg
| |   |        wnotif_services.cfg
| |   |        wnotif_LockFile.inc
| |   |        wnotif_LogFile.inc
| |   |        wnotif_Notification.inc
| |   |        wnotif_Srv.inc
| |   |        wnotif_System.inc
| |   |        wnotif_Popen.inc
| |   |        wnotif_MailCmd.inc
| |   |        wnotif_Mailphp.inc
| |   |        wnotif_Slack.inc
```

```
| |   |        wnotif_Teams.inc
| |   |        wnotif_Telegram.inc
| |   |        wnotif_Twidge.inc
| |   |        wnotif_Write.inc
| |   \-----tests/
| |          wnotif_NotificationTest.php
| |          wnotif_SrvTest.php
| |          wnotif_LockFileTest.php
| |          wnotif_LogFileTest.php
```

5.3.1 Ficheros de configuración

Definimos los archivos de configuración (**wnotif.cfg y wnotif_services.cfg)**:

- **wnotif.cfg**: contiene las variables globales utilizadas por el sistema. Como la ruta destino para los archivos de log y de lock, nombre del servidor, número de reintentos o ruta de origen de la base de datos.

```
<?php
global    $wnotif_LOGDIR,$wnotif_LOCKDIR,$wnotif_DATABASEDIR,
$wnotif_RETRIES,$wnotif_SERVER;

$wnotif_SERVER='S_WNOTIF';
$wnotif_LOGDIR='/var/www/html/wnotif/etc/wnotif.bin/';
$wnotif_LOCKDIR='/var/www/html/wnotif/tmp/';
$wnotif_DATABASEDIR='/var/www/html/wnotif/DB';
$wnotif_RETRIES=2;
?>
```

- **wnotif_services.cfg**: contiene las variables globales utilizadas por los servicios de cada tipo de notificación. Contiene rutas, nombres de usuarios y configuraciones de aplicación.

```
<?php
$WEBHOOKURLSLACK='https://hooks.slack.com/services/T05A4CP47KK/
B059PUKNY3X/Wa5CGGuCmnCZFn6mWDJ8HzRu';
$WEBHOOKURLTELEGRAM='https://api.telegram.org/
bot5703408886:AAFvXII5S5kf1FMPsasfVzM1CERpcChOgtM/sendMessage';
$ICONSLACK=':warning:';
$USERNAMESLACK='wnotifbot2';
$CHANNELSLACK='servicionotif';
$WEBHOOKURLTEAMS='https://everisgroup.webhook.office.com/
webhookb2/5d8d447e-ee92-4d11-be05-70fe1b0172b3@3048dc87-43f0-4100-9acb-
ae1971c79395/IncomingWebhook/e060b4d2de024bb5a68ffd49eee69a8e/b8066ace-
a348-45f5-8461-a4b33df38719';
?>
```

5.3.2 Ficheros a incluir

Definimos los archivos que contienen las clases, atributos y funciones, desarrolados en código PHP. Éstos archivos nos ayudan a hacer más eficiente la programación ya que un archivo puede ser referenciado por muchos otros archivos en lugar de tener que reescribir el código.

5.3.2.1 Configuración de los archivos Lock y Log

Definimos el archivo ***wnotif_LockFile.inc*** que gestiona el estado del servicio (detenido/en ejecucción).

Con este mecanismo crearemos un archivo de *Lock* para indicar que el recurso está "bloqueado", ya que varios usuarios o sistemas pueden acceder a nuestro servicio simultáneamente, este archivo indica a los usuarios/sistemas que esperen hasta que se libere dicho recurso.

Se define su código correspondiente:

```
<?php

 require_once 'wnotif.cfg';
 require_once 'wnotif_Srv.inc';

     class wnotif_LockFile {
     var $name;

      function wnotif_LockFile($IN_name){
          global $wnotif_LOCKDIR;
          $this->name=$wnotif_LOCKDIR.$IN_name;
      }

      function create (){
        $f=fopen ($this->name, 'w');
        fclose($f);
      }

      function remove (){
      if(file_exists($this->name))
        unlink($this->name);
      }

      function getStatus (){
        $v=file_exists($this->name);
        return $v;
      }
   }
?>
```

De la misma forma, definimos a continuación el archivo ***wnotif_LogFile()***. Éste es el responsable de crear un archivo de log para el sistema donde se reflejarán datos de relevancía como el estado del

servicio y notificaciones enviadas o fallidas.

```php
<?php

 require_once 'wnotif.cfg';
 require_once 'wnotif_Srv.inc';

 class wnotif_LogFile {
 var $name;
 var $f;

 function wnotif_LogFile($IN_name){
 global $wnotif_LOGDIR;
 $this->name=$wnotif_LOGDIR.$IN_name;
 }

 function openFile(){
 $this->f=fopen($this->name, 'a');
 if ($this->f == false){
 echo 'Failed to open Log file';
 }
 }

 function appendLine($IN_line){
 $v=fwrite($this->f, date('Y.m.d H:i:s')." ".$IN_line."\n");
 if ($v == false){
 echo 'Failed to write on Log file';
 }else
 return $v;
    }

         function closeFile(){
            fclose ($this->f);
         }
   }
?>
```

5.3.2.2 Servidor de Notificaciones

El fichero ***wnotif_Srv.inc***, es el corazón de este servicio, éste se encarga de recoger las notificaciones pendiente y enviarlas a su correspondiente servicio (dependiendo del tipo) para automáticamente enviarlas (o reenviarla en caso de fallo).

También es el encargado de cancelar notificaciones.

Este es su código fuente:

```php
<?php

    require_once 'wnotif_LockFile.inc';
    require_once 'wnotif_LogFile.inc';
    require_once 'wnotif_Notification.inc';
```

```
    require_once 'wnotif_services.cfg';
    require_once 'wnotif.cfg';
    require_once 'wnotif_Twidge.inc';
    require_once 'wnotif_Mailphp.inc';
    require_once 'wnotif_MailCmd.inc';
    require_once 'wnotif_Popen.inc';
    require_once 'wnotif_Slack.inc';
    require_once 'wnotif_Pushover.inc';
    require_once 'wnotif_Teams.inc';
    require_once 'wnotif_Write.inc';
    require_once 'wnotif_Telegram.inc';


    class wnotif_Srv {
      var $arraynotif;
      var $id;
      var $olockfile;
      var $ologfile;
      var $onotif;
      var $pending;
      var $not;
      var $sent;

      function wnotif_Srv($wnotif_SERVER,$IN_lockfn,$IN_logfn){
        $this->id=$wnotif_SERVER;
        $this->olockfile=new wnotif_LockFile($IN_lockfn);
        $this->ologfile=new wnotif_LogFile($IN_logfn);

        $this->pending=array();
        $this->not=array();
      }


 /*Start the service, create a lock file if not exists, write in the
logfile*/
 function start(){
 if($this->getStatus()==false){
 $this->olockfile->create();
 $this->ologfile->openFile();
 $this->ologfile->appendLine('Service started');
 $this->ologfile->closeFile();
 }else
 echo "Service already started. Exiting\n";
 }

 /*Stop the service, remove the lock file and write in the logfile*/
 function stop(){
 $this->olockfile->remove();
 $this->ologfile->openFile();
 $this->ologfile->appendLine('Service stopped');
 $this->ologfile->closeFile();
```

```
 }
 /*Get the current status*/
 function getStatus(){
 $v=$this->olockfile->getStatus();
 return $v;
 }

 /*Select a notification which needs to be send from database (sts=null
&& ats=null)*/
 function getPending(){
 global $wnotif_DATABASEDIR;
 $db=new PDO("sqlite:".$wnotif_DATABASEDIR);
 $result=$db->query('SELECT * FROM wnotif_Notification where sts is null
and ats is null');
 foreach($result as $row){
 $this->pending[]=$row;
 }
 }

 /*Send notification one by one from array, create an object for each
kind of application*/
 function send(){
 global $wnotif_RETRIES;
 for($i=0; $i<count($this->pending); $i++){
 $this->not=$this->pending[$i];
 switch($this->not['fk']){
 case 1: echo "Type 1 - Twitter Notification"."\n";
 $this->onotif=new wnotif_Twidge();
 break;

 case 2: echo "Type 2 - Slack notification"."\n";
 $this->onotif=new wnotif_Slack();
 break;

 case 3: echo "Type 3 - Mail by php notification"."\n";
 $this->onotif=new wnotif_Mailphp();
              break;

            case 4: echo "Type 4 - Mail by cmd notification"."\n";
              $this->onotif=new wnotif_MailCmd();
              break;

            case 5: echo "Type 5 - Telegram Notification"."\n";
              $this->onotif=new wnotif_Telegram();
              break;

            case 6: echo "Type 6 - Write command notification"."\n";
              $this->onotif=new wnotif_Write();
              break;

            case 7: echo "Type 7 - Teams notification"."\n";
              $this->onotif=new wnotif_Teams();
              break;
                }
```

```
            $this->onotif->set($this->not['fk'],$this->not['ufrom'],
$this->not['uto'],$this->not['subject'],$this->not['body'],$this-
>not['retries']);
            $this->onotif->setID($this->not['pk']);
            print_r($this->onotif);
            $this->onotif->prepare();
            $sent=$this->onotif->send();
            if($sent!=0){
              if(($this->onotif->retries)<$wnotif_RETRIES){
                echo("Sending again..."."\n");
                $this->ologfile->openFile();
                $this->ologfile->appendLine('Notification failed, retry
number "'.$this->onotif->retries.'",  notification with id: "'.$this-
>not['pk'].'"');
                $this->ologfile->closeFile();
                $this->onotif->send();
                $sent=$this->onotif->send();
                print_r($this->onotif);
                $this->onotif->retries++;
                $this->onotif->update();
                print_r($this->onotif);
              }else{
                  echo "All retries failed... proceed to cancel
notification"."\n";
                  $this->ologfile->appendLine('Notification has been
cancelled: "'.$this->not['pk'].'"');
                  $this->onotif->ats=date('Y-m-d H:i:s');
                  $this->onotif->update();
                  print_r($this->onotif);
              }
            }
            else{
            $this->ologfile->openFile();
            $this->ologfile->appendLine('Notification has been sent with
id: "'.$this->not['pk'].'"');
            $this->ologfile->closeFile();
            $this->onotif->sts=date('Y-m-d H:i:s');
            $this->onotif->update();
            echo("Sent & Updated"."\n");
            }
        }
    }

    /*Searching & cancel one or serveral notifications*/
    function cancellation($IN_PK){
        getPending();
        for($i=0; $i<count($this->pending); $i++){
          $this->not=$this->pending[$i];
          if($this->not['pk']==$IN_PK){
           $this->not->ats=date('Y-m-d H:i:s');
```

```
        }
    }
   }
}
?>
```

Métodos de envío

Las siguientes clases, recogen el argumento $cmd que han preparado previamente las clases de los diferentes tipos de notificaciones que se detallan en

- ***wnotif_System:*** la función system() utiliza *fork* prara crear un proceso hijo que ejecuta el comando shell especificado. El código es el siguiente:

```
<?php

require_once 'wnotif_Srv.inc';

 class wnotif_System extends wnotif_Notification {

 var $cmd;
 var $result;

 function wnotif_System() {
 }

 function prepare(){
 $this->cmd="EMPTYCMD";
 }

 function send(){
 if(system($this->cmd,$rv)===false){
              echo "ERROR $rv\n\n";
              $this->result=1;
             }
            else {
                $this->result=0;
            }
        return $this->result;
        }
    }
?>
```

- ***wnotif_Popen***: la función popen() abre un proceso creando una tubería, bifurcándose e invocando al shell. El tipo de argumento puede especificar sólo lectura (indicado co la letra “r” o escritura (indicado con la letra “w”, no ambos. El valor devuelto es un flujo E/S y debe de cerrarse con pclose(). Este es su código fuente:

```
<?php
```

```
require_once 'wnotif_Srv.inc';

class wnotif_Popen extends wnotif_Notification {

 var $cmd;

 function wnotif_Popen(){
 }
  function prepare(){
 $this->cmd="EMPTYCMD";
 }

 function send(){
 $pf=popen($this->cmd, 'w');
 $result=fwrite($pf, $this->body);
 pclose($pf);
 }
}
?>
```

5.3.2.3 Tipos de notificaciones

Con la posibilidad de aumentar las formas de envió de las notificaciones, en este proyecto se han definido las siguientes clases dependiendo del tipo de notificación enviada.

Estas clases ***preparan*** el comando (*$cmd*), que se ejecuta por consola para realizar el envío de la notificación.

- ***wnotif_MailCmd***: a traves del comando mail de linux se envia un correo electrónico al destinatario de la notificaión. Su código es el siguiente:

```
<?php

require_once 'wnotif_Popen.inc';

class wnotif_MailCmd extends wnotif_Popen {

        var $cmd;
        var $obody='';

        function wnotif_MailCmd(){
        }

        function prepare(){
             $this->cmd='mail '.$this->uto." -s '".$this->subject."' -r
'From ".$this->ufrom."'"."' <<< "."'$this->body'";
            }
   }
?>
```

- ***wnotif_Mailphp.inc:*** Hace uso de la función propia de PHP para envíar correos electrónicos. Luego esta clase no extiende de las clases wnotif_Popen o wnotif_System para realizar el envío ya que se realiza internamente y no utilizando el shell. El código fuente es el siguiente:

```
<?php

require_once 'wnotif_Notification.inc';

class wnotif_Mailphp extends wnotif_Notification {

        function wnotif_Mailphp(){
        }

        function prepare(){
        }

        function send(){
           $headers='From: '.$this->ufrom."\r\n" . 'Reply-To: '.$this-
>ufrom ."\r\n";
           echo "headers '$headers'"."\n";
           $bool = mail($this->uto, $this->subject, $this->body,
$headers);
           echo("result ---> '$bool'"."\n");
           return($bool);
        }
}
?>
```

Para las siguientes tres clases de notificaciones se ha hecho uso de ***"Incoming Webhooks"[7] [8] [9]***como vía para el envío de notificaciones.
Los Webhooks entrantes, son una forma sencilla de enviar mensajes desde aplicaciones/servicios hacía otras aplicaciones. Al crear un Incoming Webhook, se obtiene una URL única a la que se envía una carga JSON con el texto del mensaje y algunas opciones.

Hemos utilizado el comando *cURL* para transferir los datos desde el servidor realizando una petición HTTP POST:

Se han configurado para enviar mensajes en grupos ya preconfigurados desde la propia aplicación. Se indica en el apartado ANEXO.

- ***wnotif_Slack***: Envía una notificación a un canal de un espacio en Slack. El código fuente es el siguiente:

```
<?php

 require_once 'wnotif_System.inc';
 require_once 'wnotif.cfg';
```

```
 require_once 'wnotif_services.cfg';

class wnotif_Slack extends wnotif_System {

 function wnotif_Slack(){
    }

    function prepare(){

    global $WEBHOOKURLSLACK;
    global $USERNAMESLACK;
    global $CHANNELSLACK;
    global $ICONSLACK;

    $this->cmd="curl -k -X POST -H 'Content-type: application/json' --
data '{\"text\":\"$this->body\", \"channel\":\"$CHANNELSLACK\",
\"icon_emoji\": \"$ICONSLACK\"}' \"$WEBHOOKURLSLACK\"";
    }
}
?>
```

- ***wnotif_Teams***: Se envían notificaciones a un canal propio de un equipo en Teams[9].Se define el correspondiente código fuente:

```
<?php

 require_once 'wnotif_System.inc';
 require_once 'wnotif.cfg';
 require_once 'wnotif_services.cfg';


class wnotif_Teams extends wnotif_System {


 function wnotif_Teams(){
    }

    function prepare(){

    global $WEBHOOKURLTEAMS;

    $this->cmd="curl -X POST -H 'Content-type: application/json' --data
'{\"text\":\"$this->body\"}' \"$WEBHOOKURLTEAMS\" ";
    }
}
?>
```

- ***wnotif_Telegram:*** para la aplicación Telegram, se encontró un comando que al instalarse previamente en la máquina virtual utilizada, permitía (tras previa configuración) realizar envíos de mensajes vía telegram. Este comando "*telegram-send"[10]* desactualizado por el

momento (queda comentado en el código). Como alternativa utilizamos también una petición HTTP ya que para Telegram también es posible hacer uso de Webhooks entrantes[8].

Este es el código fuente:

```
<?php
 require_once 'wnotif_System.inc';
 require_once 'wnotif_services.cfg';

class wnotif_Telegram extends wnotif_System {


    function wnotif_Telegram(){

    }

    function prepare (){

    /*$this->cmd="telegram-send \"$this->body\"";*/
    global $WEBHOOKURLTELEGRAM;

    $this->cmd="curl -k -X POST -H 'Content-type: application/json' --
data '{\"text\":\"$this->body\", \"chat_id\":\"$CHATID\"}'
\"$WEBHOOKURLTELEGRAM\" ";

    }
}
?>
```

- ***wnotif_Twidge [11]:*** Se hace uso de *twidge* es un cliente de Twitter para el terminal. Es compatible con nuestra máquina virtual y se encuentra en los repositorios de ubuntu,

  Para su instalación/configuraicón se ejecuta el siguiente comando.

Tras el comando *setup*, te indica que acudas a la dirección de internet, para conceder acceso desde Twitter a esta aplicación, te indicará un número que tendrás que pegar a continuación.

```
avarela@u-1604-64:~$ sudo apt-get install twidge
avarela@u-1604-64:~$ twidge setup
Welcome to twidge.  We will now configure twidge for your
use with Twitter (or a similar service).  This will be quick and easy!
Please wait a moment while I query the server...
OK, next I need you to authorize Twidge to access your account.
Please cut and paste this URL and open it in a web browser:

Click Allow when prompted.  You will be given a numeric
key in your browser window.  Copy and paste it here.
(NOTE: some non-Twitter services supply no key; just leave this blank
if you don't get one.)
```

Se pueden hacer uso de muchos comandos, entre los que se destacan:

- *dmsend -> envía un mensaje directo*
- *block -> para bloquear a alguien*
- *follow -> para seguir a alguien*
- *lsarchive -> los últimos Tweets que has envíado*
- *lscommands -> una lista de todos los comandos*
- *lsdm -> lista los mensajes directos más recientes que te han enviado*
- *lsdmarchive -> Lista los mensajes directos más recientes que has enviado*
- *lsblocking -> Lista las personas que has bloqueado*
- *lsfollowers -> lista las personas que te siguen*
- *lsfollowing -> lista las personas a las que sigues*
- *lsrecent -> lista las tweets mas recientes de las personas a las que sigues*
- *lsreplies -> lista los últimos tweets que te mencionan*
- *lsrt -> lista los retweets mas recientes de aquellos a los que sigues*
- *lsrtarchive -> lista los retweets mas recientes que has hecho*
- *lsrtreplies -> lista los retweets de tus tweetsList others' retweets of your statuses*
- *setup -> para configurar Twidge*
- *unblock -> para dejar de bloquear a alguien*
- *unfollow -> para dejar de seguir a alguien*
- *update -> para enviar un Tweet*

El último comando *update* es el más interesante para nuestro servicio, luego hemos hecho uso de él en la clase ***wnotif_Twitdge***, cuyo código fuente es el siguiente:

```
<?php

 require_once 'wnotif_System.inc';

 class wnotif_Twidge extends wnotif_System {

    function wnotif_Twidge() {
    }

        function prepare(){
        $this->cmd="twidge update '".$this->body."'";
        }
}
?>
```

- ***wnotif_Write [12]:*** Se hace uso de la utilizad Write de Linux, comando de escritura que lee

las líneas de entrada estándar y las escribe en el terminal de usuario especificado. El código fuente es el siguiente:

```php
<?php

require_once 'wnotif_Srv.inc';

class wnotif_Write extends wnotif_Popen {

    var $cmd;

    function wnotif_Write(){
    }

        function prepare(){
        $this->cmd='write '.$this->uto;
        }
}
?>
```

5.4 Construcción de la API

En esta sección se incluye el código PHP desarrollado para implementar la API que permite a las distintas aplicaciones **WAINE** *solicitar el envió de notificaciones y consultar sobre el estado de las mismas*. El código se encuentra definido en el fichero ***wnotif_Notification.inc*** y es el siguiente:

```php
<?php

   require_once 'wnotif.cfg';
   require_once 'wnotif_services.cfg';
   require_once 'wnotif_Srv.inc' ;

  class wnotif_Notification {
        var $id;
        var $type;
        var $notifPend;
        var $ufrom;
        var $uto;
        var $sts;
        var $cts;
        var $ats;
        var $subject;
        var $body;
        var $retries=0;

        function wnotif_Notification() {}

        function setID($IN_id){
          if (func_num_args() != 1) {
              echo 'Invalid arguments, please introduce an id'."\n";
```

```
          }else
          $this->id=$IN_id;
          }

        /*Set the notification values*/
        function set($IN_type,$IN_from,$IN_to,$IN_subject,$IN_body,
$IN_retries){
          if (func_num_args() != 6){
            echo 'Invalid arguments, please check arguments of
Notification'."\n";
          }
          else {
            $this->type=$IN_type;
            $this->ufrom=$IN_from;
            $this->uto=$IN_to;
            $this->subject=$IN_subject;
            $this->body=$IN_body;
            $this->retries=$IN_retries;
             }
        }


        /*Insert a notification and return the id if is ok*/

        function request(){
          global $wnotif_DATABASEDIR;
          $db=new PDO("sqlite:".$wnotif_DATABASEDIR);
          if (empty($db)){
            echo 'Failed to establish a connection to database';
          }
          $insert='INSERT INTO wnotif_Notification(fk, ufrom, uto,
subject, body, retries) VALUES
(:fk, :ufrom, :uto, :subject, :body, :retries)';
          $stmt=$db->prepare($insert);
          $stmt->bindParam(':fk', $this->type);
          $stmt->bindParam(':ufrom', $this->ufrom);
          $stmt->bindParam(':uto', $this->uto);
          $stmt->bindParam(':subject', $this->subject);
          $stmt->bindParam(':body', $this->body);
          $stmt->bindParam(':retries', $this->retries);
          $stmt->execute();
          $idd=$db->lastInsertId();
          if (empty($idd)){
            echo 'Failed to insert notification, please check the
params';
           }
          else
            echo "Notification inserted in db, ready to be send with id:
$idd"."\n";
        }
```

```
          /*Receive notification id and return data notification*/

          function query(){
            global $wnotif_DATABASEDIR;
            $db=new PDO("sqlite:".$wnotif_DATABASEDIR);

            if (empty($db)){
                  echo 'Failed to establish a connection to database';
            }

            $result=$db->query('SELECT * FROM wnotif_Notification where
pk='.$this->id.';');
            if (empty($result)){
                echo ('Failed to receive data from database'."\n");
               }else{
                $row=$result->fetch();
                $this->type=$row['fk'];
                $this->ufrom=$row['ufrom'];
                $this->uto=$row['uto'];
                $this->subject=$row['subject'];
                $this->body=$row['body'];
                $this->cts=$row['cts'];
                $this->sts=$row['sts'];
                $this->ats=$row['ats'];
             }
            }

          /*Update notification info*/
          function update(){
            global $wnotif_DATABASEDIR;
            $db=new PDO("sqlite:".$wnotif_DATABASEDIR);

            $update='UPDATE wnotif_Notification SET
sts=:sts,subject=:subject, body=:body, uto=:uto, ufrom=:ufrom, fk=:fk,
retries=:retries, ats=:ats WHERE pk=:pk';
            $stmt=$db->prepare($update);
            $stmt->bindParam(':pk', $this->id);
            $stmt->bindParam(':fk', $this->type);
            $stmt->bindParam(':ufrom', $this->ufrom);
            $stmt->bindParam(':uto', $this->uto);
            $stmt->bindParam(':subject', $this->subject);
            $stmt->bindParam(':body', $this->body);
            $stmt->bindParam(':sts', $this->sts);
            $stmt->bindParam(':retries', $this->retries);
            $stmt->bindParam(':ats', $this->ats);
            echo($this->sts."\n");
            $stmt->execute();
          }
  }
?>
```

5.5 Construcción de las interfaces de usuario

En esta sección se procede a detallar el proceso de construcción de las interfaces de usuario para cada una de las funciones y usuarios definidos en el sistema. El diseño de estas interfaces se ha hecho en base a los diagramas entidad-relación y tablas de rol/funcionalidad recogidos en el apartado de Modelo de Datos y Interfaces de usuario yacciones del docuemnto de análisis.

Al utilizarse el entorno de desarrollo de interfaces de usuario basado en modelo WAINE, para generar la aplicación web, es necesario definir los cuatro modelos que utiliza [2]:

- ***Modelo de dominio***: define los objectos accesibles para los uuarios a través de la interfaz de usuario. Se implementa usando el lenguaje SQL
- ***Modelo de presentación:*** recoge los aspectos de la presentación de la interfaz de usuario. Posee dos elementos principales, los contenedores y los formularios. Este modelo se implementa haciendo uso del lenguage ASL.
- ***Modelo de usuario:*** proporciona una descomposición jerárquica d ellos usuarios que se encuentra en un grupo de acuerdo a su rol. Se implementa haciendo uso del lenguaje ASL.
- ***Modelo de dialogo:*** describe las acciones que un usuario puede hacer en el sistema. Este modelo se implementa usando el lenguaje ASL.

La implementación de las interfaces se ha realizado utilizando el lenguaje de descripción ASL propio del MB-UIDE **WAINE**.

En los siguientes apartados definimos el Modelo de Dialogo asociado a cada uno de los usuarios de la aplicación, el Modelo de Presentación de cada una de las interfaces y varias capturas del aspecto final de las interfaces implementadas.

5.5.1 Usuario Administrador

Se muestra a continuación las diferentes acciones e interfaces que puede utilizar el usuario administrador.

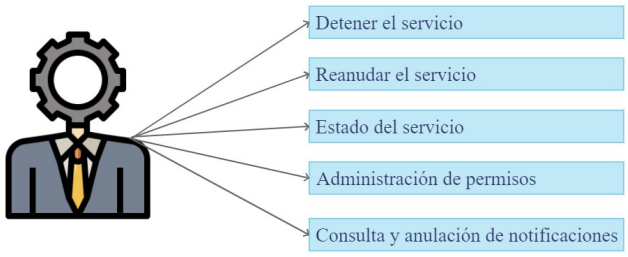

Ilustración 21: Esquema Menú Administrador

El código ASL correspondiente al Modelo de Dialogo del administrador es:

```
    <main id="main_admin"
          caption="en=Administrator menu|es=Menu del administrador">
```

```
        <menu caption="en=Notification Management|es=Gestion de
notificaciones">
            <option caption="en=Permission administration|
es=Administración de permisos" call="wnotif.cgroupAdmin"/>
            <option caption="en=Query/cancellation of notifications|
es=Consulta/anulación de notificaciones" call="wnotif.cqryrem"/>
            <option caption="en=Stop Service|es=Deterner el servicio"
action="wnotif.astop"/>
            <option caption="en=Run service|es=Reanudar el servicio"
action="wnotif.astart"/>
            <option caption="en=Service status|es=Estado del servicio"
action="wnotif.agetStatus"/>
        </menu>
        <menu caption="en=Misc|es=Otros">
            <option caption="en=Waine page|es=Web de Waine"
                    url="http://waine.us.es"/>
            <option caption="en=Author|es=Autor"
call="meta.struct.appinfo"/>
            <option caption="en=Logout|es=Salir" url="logout.php"/>
        </menu>
    </main>
```

La interfaz de usuario obtenida del menú administrador:

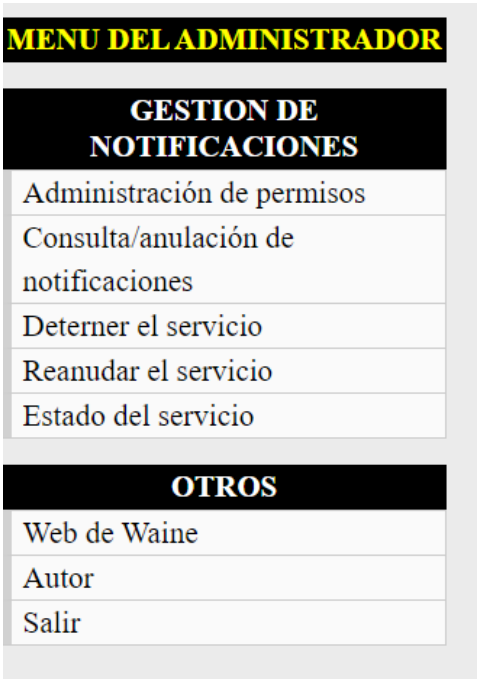

Ilustración 22: Interfaz de usuario final Menú Administrador

5.5.1.1 Interfaz. Administración de Permisos

El código ASL desarrollado para el Modelo de Presentación de la interfaz *Administración de permisos* es:

```
    <struct id="wnotif.cgroupAdmin" type="relation">
        <param name="form_split" value="rows=100,*"/>

        <param name="formid" value="wnotif.group"/>
        <param name="form_type" value="combo"/>
        <param name="source_filter_where" value="gid != 1"/>


        <param ord="2" name="formid" value="wnotif.fRelCan"/>
        <param ord="2" name="form_type" value="table"/>
        <param ord="2" name="button_data" value="1"/>
        <param ord="2" name="source_filter_field" value="gid"/>
    </struct>
```

Luego la interfaz de usuario obtenida es la siguiente:

Ilustración 23: Interfaz de usuario final Administración permisos

5.5.1.2 Interfaz. Consulta y anulación de notificaciones

El código ASL desarrollado para el Modelo de Presentación de la interfaz *Consulta y Anulación de notificacines* es el siguiete:

```
    <struct id="wnotif.cqryrem" type="relation">
        <param name="form_split" value="rows=140,*"/>

        <param name="formid" value="wnotif.ffilteradmin"/>
```

```
        <param name="form_type" value="filter-where"/>
        <param name="form_subtype" value="wide"/>
        <param name="form_wide" value="300"/>
        <param name="form_filterw_oper" value="=;=;=;=;LIKE;"/>
        <param name="form_filterw_nexus" value="and"/>
        <param name="button_misc" value="0"/>

        <param ord="2" name="formid" value="wnotif.fqryrem"/>
        <param ord="2" name="form_type" value="table"/>
        <param ord="2" name="button_misc" value="0"/>
        <param ord="2" name="button_action" value="1"/>
        <param ord="2" name="fields_readonly" value="1-9"/>

        <param ord="2" name="__nofill" value="1"/>
    </struct>
```

La interfaz de usuario obtenida:

Ilustración 24: Interfaz de usuario final Consulta y anulación de notificaciones

5.5.1.3 Acciones. Detener el servicio

Para esta acción no se ha implementado Modelo de Presentación como en las dos anteriores haciendo uso de la etiqueta <struct> en ASL, al tratarse de un acción que no requiere parámetros previos, se ha utilizado la etiqueta <action>.

El código ASL es el siguiente:

```
<action id="wnotif.astop" type="php">
        <code>
        set_include_path("etc/wnotif.bin");
        require_once("wnotif_LockFile.inc");
        $lf=new wnotif_LockFile($wnotif_SERVER.".stop.lock");
        $lf->create();
        </code>
        <msg>es=Desea detener las notificaciones ?|en=Do you want to
pause notifications ?</msg>
   </action>
```

La interfaz de usuario final muestra el mensaje por pantalla para confirmar la acción:

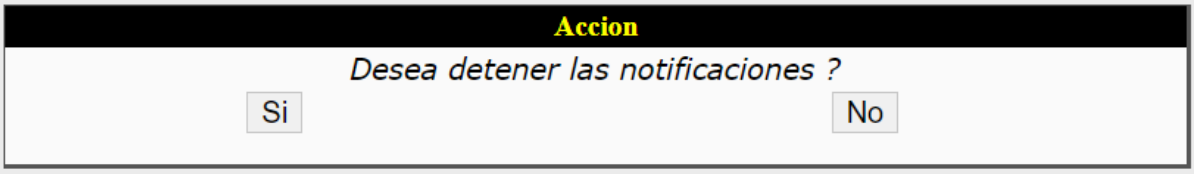

Ilustración 25: Interfaz de usuario final Mensaje detección del servicio

5.5.1.4 Acciones. Reanudar el servicio

De la misma forma que en el apartado anterior, para reanudar el servicio se ha desarrollado el siguiente código ASL:

```
   <action id="wnotif.astart" type="php">
        <code>
        set_include_path("etc/wnotif.bin");
        require_once("wnotif_LockFile.inc");
        $lf=new wnotif_LockFile($wnotif_SERVER.".stop.lock");
        $lf->remove();
        </code>
        <msg>es=Desea reanudar las notificaciones ?|en=Do you want to
start notifications service ?</msg>
   </action>
```

El mismo mensaje aparecerá en la interfaz de usuario justo al hacer clic en la opción del menú *Reanudar el servicio*:

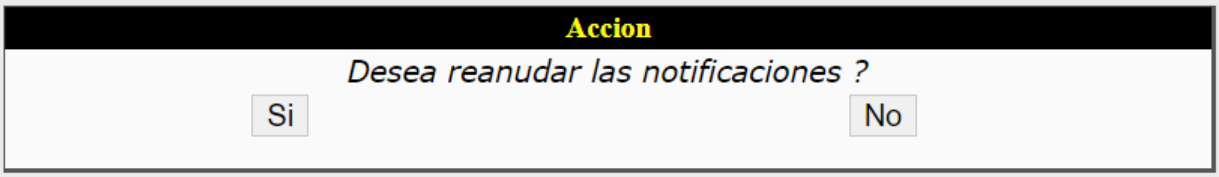

Ilustración 26: Interfaz de usuario final Mensaje reanudación del servicio

5.5.1.5 Acciones. Estado del servicio

La acción para consultar el estado actual del servicio tiene el siguiente código ASL:

```
<action id="wnotif.agetStatus" type="php">
        <code>
        set_include_path("etc/wnotif.bin");
```

```
        require_once("wnotif_LockFile.inc");
        $lf=new wnotif_LockFile($wnotif_SERVER.".stop.lock");
        if(!$lf->getStatus())
            echo "wnotif en ejecución";
        else
            echo "wnotif detenido";
        </code>
    </action>
```

Aparecerán los siguientes mensajes por pantalla dependiendo del estado del servicio (en este caso el servicio está detenido):

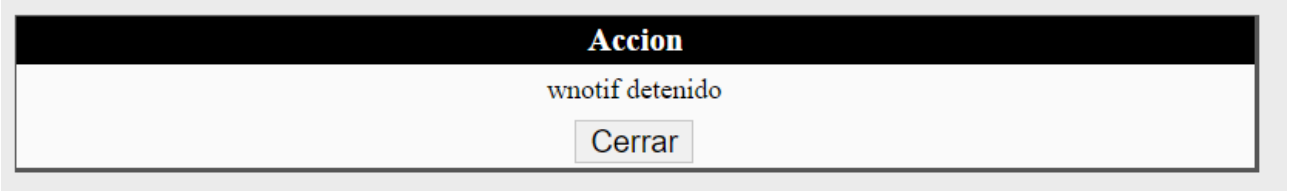

Ilustración 27: Interfaz de usuario final Mensaje servicio detenido

5.5.2 Usuario Regular

Las interfaces de usuario al alcance de un usuario regular son las siguientes:

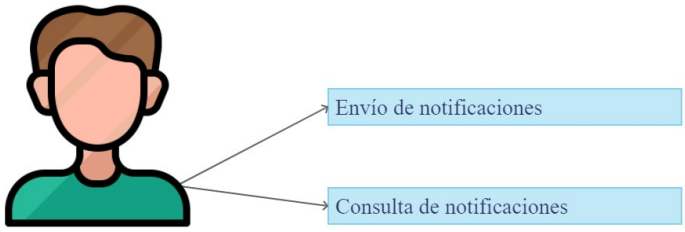

Ilustración 28: Esquema Menú Usuario R

El código ASL definido para el Modelo de Dialogo del usuario regular es:

```
<main id="main_ruser"
          caption="en=Default user menu|es=Menu del usuario por defecto">
       <menu caption="en=Management|es=Gestion">
           <option caption="en=Send notification|es=Envío de
notificaciones" call="wnotif.csend"/>
           <option caption="en=Notification query|es=Consulta de
notificaciones" call="wnotif.cquery"/>
       </menu>
       <menu caption="en=Misc|es=Otros">
           <option caption="en=Waine page|es=Web de Waine"
                   url="http://waine.us.es"/>
           <option caption="en=Author|es=Autor"
call="meta.struct.appinfo"/>
```

```
            <option caption="en=Logout|es=Salir" url="logout.php"/>
        </menu>
    </main>
```

La menú de usuario regular resultante es:

Ilustración 29: Interfaz de usuario final Menú del Usuario Regular

5.5.2.1 Interfaz. Envío de notificaciones

El código ASL para el Modelo de Presentación del envió de notificaciones es el siguiente:

```
    <struct id="wnotif.csend" type="form">
        <param name="formid" value="wnotif.fsend"/>
        <param name="form_type" value="form"/>
        <param name="button_misc" value="0"/>
        <param name="button_insert" value="1"/>
    </struct>
```

Permite al usuario regular solicitar el envío de notificaciones siempre dentro de los tipos permitidos, la interfaz de usuario final resultante es se trata de un formulario simple:

Ilustración 30: Interfaz de usuario final Envío de notificaciones

5.5.2.2 Interfaz. Consulta de notificaciones

El código ASL para el Modelo de Presentación es el siguiente:

```
<struct id="wnotif.cquery" type="relation">
     <param name="form_split" value="rows=150,*"/>

     <param name="formid" value="wnotif.ffilter"/>
     <param name="form_type" value="filter-where"/>
     <param name="form_subtype" value="wide"/>
     <param name="form_filterw_oper" value="=;=;LIKE;=;"/>
     <param name="form_filterw_nexus" value="and"/>
     <param name="button_misc" value="0"/>

     <param ord="2" name="formid" value="wnotif.fquery"/>
     <param ord="2" name="form_type" value="list"/>
     <param ord="2" name="button_misc" value="0"/>

     <param ord="2" name="__nofill" value="1"/>
 </struct>
```

Esta interfaz permite a los usuarios consultar las notificaciones que ha emitido. La interfaz está compuesta por un formulario compuesto, cuyos elementos del modelo de dominio están relacionados entre sí.

Ilustración 31: Interfaz de usuario final Consulta de notificaciones

5.5.3 Usuario API

A modo de demostración, se ha creado un usuario extra para introducir notificaciones de forma aleatoria en el sistema.

El Modelo de Dialogo sería el siguiente en código ASL:

```
<main id="main_apiuser"
          caption="en=API user menu|es=Menu del usuario API">
       <menu caption="en=Notif Generator|es=Generador de notificaciones">
            <option caption="en=Notification Insert|es=Insertar
notificaciones" action="wnotif.arequest"/>
       </menu>
       <menu caption="en=Misc|es=Otros">
           <option caption="en=Waine page|es=Web de Waine"
                   url="http://waine.us.es"/>
           <option caption="en=Author|es=Autor"
call="meta.struct.appinfo"/>
           <option caption="en=Logout|es=Salir" url="logout.php"/>
       </menu>
   </main>
```

El menú consta de una única opción "*Generar notificaciones*" y se obtiene como se muestra a continuación:

Ilustración 32: Interfaz de usuario final Menú de usuario API

5.5.3.1 Acción. Insertar notificaciones

El código ASL para la inserción de notificaciones es el siguiente:

```
      <action id="wnotif.arequest" type="php">
       <code>
           set_include_path("etc/wnotif.bin");
           require_once "wnotif.cfg";
           require_once "wnotif_services.cfg";
           require_once "wnotif_Notification.inc";

           $randomNot=rand(1, 7);
           $from="anavarelarod@gmail.com";
           $to="anavarelarod@gmail.com";
           $subject="Prueba test 1";
           $body="Hello";
           $retries=0;

           $n=new wnotif_Notification();
           $n->set($randomNot,$from,$to,$subject,$body,$retries);
           $n->request();
       </code>
  </action>
```

Esta acción inserta una notificación aleatoria de cualquier tipo en la base de datos lista para ser enviada.

Como resultado por pantalla aparecerá un mensaje indicando el id de la notificación insertada.

Se puede consultar que la notificación ha sido insertada correctamente abriendo el menú administrador en la pestaña "*Consultar/anular notificaciones"*, ésta tendrá el estado *"Pendiente"*.

También se puede hacer una consulta directa en la base de datos para comprobar que la notificación se ha añadido correctamente.

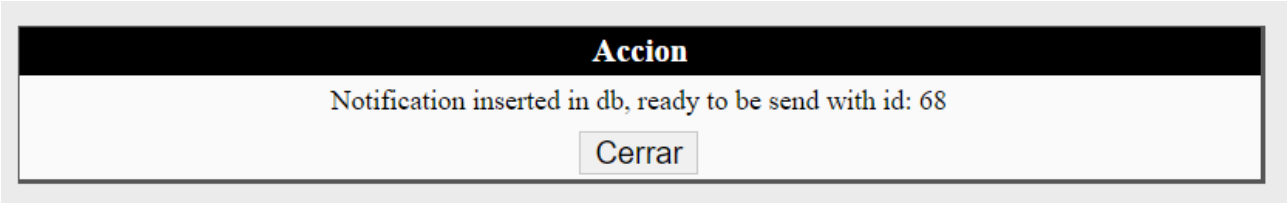

Ilustración 33: Interfaz de usuario final, Solicitar notificación aleatoria (API)

Ilustración 34: Interfaz de usuario final. Notificación aleatoria pendiente a ser enviada

5.6 Construcción de pruebas técnicas

El diseño de las pruebas se realizó en el documento de Diseño, capítulo de Plan de Pruebas técnicas, y en este apartado se procede a definir la construcción de las mismas para las distintas áreas y funcionalidades del sistema.

5.6.1 Pruebas Unitarias

Principalmente se comprobará que cada clase/métodos sigue el flujo normal de funcionamiento.

Para ello se han creado los siguientes archivos de prueba:

- ***wnotif_LogFileTest.php***

```php
<?php

require 'wnotif_LogFile.inc';

$l=new wnotif_LogFile('wnotif.log');

$l->openFile();
$l->appendLine('Linea1');
$l->appendLine('Linea2');
$l->appendLine('Linea3');
$l->closeFile();
```

```
?>
```

Trás la ejecución de este archivo de text, se espera que se haya creado el archivo correspondiente de log, se abra el archivo listo para la escritura, se escriban las líneas correspondientes y posteriormente se cierre el archivo.

```
root@u-1604-64:/var/www/html/wnotif/etc/wnotif.bin# php
wnotif_LogFileTest.php
root@u-1604-64:/var/www/html/wnotif/etc/wnotif.bin#
root@u-1604-64:/var/www/html/wnotif/etc/wnotif.bin# cat wnotif.log
2023.06.13 18:54:10 Linea1
2023.06.13 18:54:10 Linea2
2023.06.13 18:54:10 Linea3
```

- ***wnotif_LockFile.php***

```
<?php

require_once 'wnotif_LockFile.inc';

$l=new wnotif_LockFile("prueba");
$l->create();

echo 'name: '.$l->name."\n";
echo 'getStatus(): '.(($l->getStatus())?'true':'false')."\n";
$l->remove();
echo 'getStatus(): '.(($l->getStatus())?'true':'false')."\n";

?>
```

Como resultado se espera que, al ejecutar este test, se haya creado un archivo de lock llamado *"prueba"*, y que devuelva el nombre del archivo (con su ruta correspondiente) y el estado del sistema antes y después de eliminarlo.

```
root@u-1604-64:/var/www/html/wnotif/etc/wnotif.bin# php
wnotif_LockFileTest.php


name: /var/www/html/wnotif/tmp/prueba
getStatus(): true
getStatus(): false
```

5.6.2 Pruebas de integración

En este apartado se comprueba la comunicación entre clases, así como los datos que se transfieren son los adecuados.

Hemos creado los siguientes archivos:

- ***wnotif_Notification.php.*** Se prueba la API del sistema.

```
<?php
```

```
   require_once 'wnotif.cfg';
   require_once 'wnotif_services.cfg';
   require_once 'wnotif_Notification.inc';

$n=new wnotif_Notification();

//id, from, to, subject, body, sts, cts)
$n->set('1','avarela','avarela','Subject Prueba test I','Cuerpo Prueba
test I','0');

//Request the notification into db
$n->request();

//Query
$n->query("73");

//Updating info
$n->subject='Subject Prueba test II';
$n->body='Cuerpo Prueba test I';
$n->uto='test'.date('s').'@gmail.com';
$n->ufrom='test@u.com';
$n->sts=date('Y-d-m H:i:s');
$n->update();

//Query to check the update
$n->query("73");

?>
```

Con esta prueba, estamos:

- Introduciendo una nueva notificación en el sistema son *set*()
- Solicitando la notificación para su envío con *request*()
- Consultando los datos de la notificación con *query*(), en este caso en concreto consultado la notificación con id = 73.
- Actualizando algunos campos de la notificación con *update*(), esta función devuelve la fecha de actualización si todo ha sido actualizado correctamente.
- Haciendo de nuevo una consulta para comprobar que se han actualizado correctamente.

El resultado será el siguiente:

```
root@u-1604-64:/var/www/html/wnotif/etc/wnotif.bin# php
wnotif_NotificationTest.php

Notification inserted in db, ready to be send with id: 82
Array
(
    [pk] => 73
    [0] => 73
```

```
    [fk] => 1
    [1] => 1
    [ufrom] => avarela
    [2] => avarela
    [uto] => avarela
    [3] => avarela
    [agent] =>
    [4] =>
    [subject] => Subject Prueba I
    [5] => Subject Prueba I
    [body] => Cuerpo Prueba test I
    [6] => Cuerpo Prueba test I
    [retries] => 0
    [7] => 0
    [cts] => 2023-06-16 16:17:58
    [8] => 2023-06-16 16:17:58
    [sts] =>
    [9] =>
    [ats] =>
    [10] =>
)
2023-16-06 18:36:10 //Fecha de actualización
Array
(
    [pk] => 73
    [0] => 73
    [fk] => 1
    [1] => 1
    [ufrom] => test@u.com
    [2] => test@u.com
    [uto] => test10@gmail.com
    [3] => test10@gmail.com
    [agent] =>
    [4] =>
    [subject] => Subject Prueba test II
    [5] => Subject Prueba test II
    [body] => Cuerpo Prueba test II
    [6] => Cuerpo Prueba test II
    [retries] => 0
    [7] => 0
    [cts] => 2023-06-16 16:17:58
    [8] => 2023-06-16 16:17:58
    [sts] => 2023-16-06 18:36:10
    [9] => 2023-16-06 18:36:10
    [ats] =>
    [10] =>
)
```

- wnotif_SrvTest.php

Con estse archivo se ha simulado un envío de notificaciones mediante el servidor de notificaciones.

Este es el código fuente de la prueba:

```php
<?php

require_once 'wnotif_Srv.inc';
require_once 'wnotif_Notification.inc';
require_once 'wnotif.cfg';

$uno=new wnotif_Srv($wnotif_SERVER,"Prueba.lock","Prueba.log");

$uno->start();
echo 'getStatus(): '.(($uno->getStatus())?'true': 'false')."\n";

$uno->getPending();
$uno->send();

$uno->stop();
?>
```

Con esta prueba estamos realizando lo siguiente:

- Iniciando el servicio con *start*()
- Recoger las notificaciones con estado “Pendiente a enviar” con getPending()
- Enviando todas ellas a sus respectivos servicios mediante la función send()
- Parar el servicio con la función stop()

Se espera que todas las notificaciones se procesen correctamente:

```
root@u-1604-64:/var/www/html/wnotif/etc/wnotif.bin# php
wnotif_SrvTest.php


getStatus(): true
Type 1 - Twitter Notification
wnotif_Twidge Object
(
    [cmd] =>
    [result] =>
    [id] => 84
    [type] => 1
    [notifPend] =>
    [ufrom] => avarela
    [uto] => avarela
    [sts] =>
    [cts] =>
    [ats] =>
    [subject] => Subject Prueba I
    [body] => Cuerpo Prueba test I
    [retries] => 0
)
2023-06-17 00:10:08
Sent & Updated
```

Las pruebas de sistema y de implantación se describen y especifican en el siguiente capitulo de implantación.

5.7 Implementación del servicio en entornos WAINE

5.7.1 Herramientas y conceptos

En esta sección se presentan las herramientas de WAINE empleadas para la creación de una instancia de aplicación (***mkapp***) y para la elaboración de paquetes (***wpkg***). Al mismo tiempo se introducen los conceptos asociados a las mismas.

5.7.1.1 Instancia de aplicación WAINE y herramienta mkapp.

Una instancia de aplicación WAINE es un directorio que contiene una serie de archivos, directorios y enlaces. Aunque esta instancia de aplicación podría ser creada de forma manual, debe emplearse la herramienta *mkapp* para su creación [13]

5.7.1.2 Herramienta mkapp

La herramienta *mkapp* crea una instancia de aplicación ligada a una determinada versión del motor WAINE. Esta herramienta también genera un repositorio de configuraciones por defecto que permiten un funcionamiento básico de la aplicación.

La versión de WAINE empleada es la 0.5.10.

5.7.1.3 Sistema de gestión de paquetes y herramienta wpkg

WAINE cuenta con su propio sistema de gestión de paquetes, el cual se apoya sobre tres elementos principales[14]:

1. Los ficheros de paquetes wpk
2. La herramienta wpkg
3. El directorio packages

Desarrollamos las dos primeras teniendo en cuenta que será útil conocimiento previo de ellas antes de realizar la construcción del paquete wnotif.

Ficheros de paquetes wpk

Un fichero wpk puede contener descripciones ASL, código PHP, código SQL o cualquier otro tipo de fichero empleado en una instancia de aplicación.
Además, los paquetes contienen un archivo llamado ***meta.xml***, el cual aporta metainformación sobre el paquete.
También cuentan con un par de scripts, ***preins.sh*** y ***postins.sh*** que son ejecutados antes y después de la instalación del paquete respectivamente. Los paquetes también pueden aportar documentación, instrucciones, ejemplos, diagramas, etc.
Finalmente, todo el contenido se comprime en un archivo con formato tgz (tar+gzip) [18] .
La estructura de un archivo wpk es la siguiente:

- **Directorio ASL**: Contiene fragmentos de código ASL para procesar y agregar al repositorio de interfaces de usuario de la instancia de aplicación.

- **Directorio files:** Contiene los archivos necesarios y requeridos para el correcto funcionamiento de las funcionalidades software distribuidas en el paquete. Estos archivos pueden ser de cualquier tipo: archivos de configuración (.cfg), temas (.themes), imágenes, scripts, php…etc. Es imprescindible que estos archivos mantengan la estructura u organización de directorios que deberán tener en la instancia de aplicación, de esta manera serán adecuadamente procesados y agregados al repositorio de interfaces de usuario de instancia.
- **Directorio meta:** Contiene los archivos ***meta.xml, preins.sh y postins.sh.*** Esta información es la siguiente: nombre del paquete, versión, autor, descripción del paquete, funcionalidad proporcionada y dependencias. Esta información es utilizada para alimentar la base de datos de paquetes instalados, resolver dependencias y detectar posibles conflictos
- **Directorio doc:** Contiene el código SQL del paquete, archivos de documentación, ejemplos de uso, etc.

La herramienta wpkg

Es la herramienta para la gestión de paquetes **WAINE**. Permite listar los componentes instalados, añadir, modificar y eliminar paquetes en una instancia de aplicación.[14]

Las diferentes opciones que podemos utilizar son:

- **create**: crea un directorio con la estructura de un paquete vacío para que el desarrollador pueda añadir contenido.
- **package**: crea un archivo wpk (paquete) a partir un directorio con el contenido del paquete.
- **install**: instala un paquete wpk sobre una instancia de aplicación
- **remove**: elimina un paquete de una instancia de aplicación.
- **list**: lista los paquetes wpk instalados sobre una instancia de aplicación.
- **test**: verifica que un directorio de paquetes en desarrollo es correcto.

5.7.2 Construcción del paquete wnotif

En la presente sección se construye el paquete *.wpk para el servicio wnotif.

Una vez aclarados los conceptos previos, la creación de la estructura del paquete se realiza con el siguiente comando (comentado en el apartado anterior):

```
root@u-1604-64:/home/avarela# /usr/local/lib/waine-0.5.10/bin/wpkg create
wnotif_1.0


root@u-1604-64:/home/avarela/wnotif_1.0# ls
ASL  doc  files  meta
```

Con ello tenemos la estructura de directorios vacíos para indicar el contenido de nuestro servicio.

La estructura de ficheros y organización del paquete se detalla en el siguiente diagrama de despliegue:

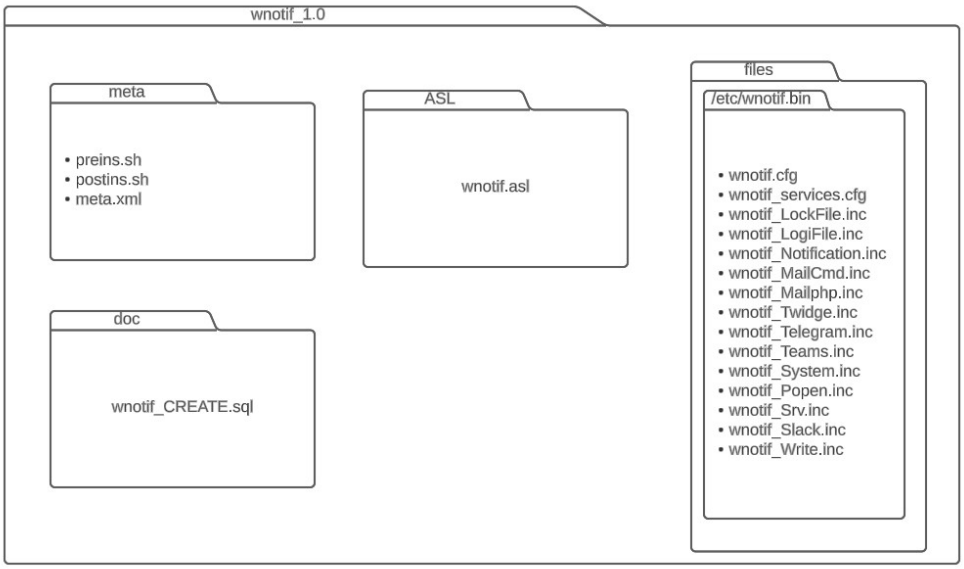

Ilustración 35: Diagrama de paquetes wnotif_1.0

Procedemos a detallar el contenido de los diferentes paquetes/directorios:

5.7.2.1 Directorio meta

Como se ha indicado en el apartado anterior contiene los archivos:

Archivo meta.xml

```
<wpkg>
<package>
    <name>wnotif</name>
    <ver>1.0</ver>
    <author>Ana Varela</author>
    <date>2023-05-31</date>
    <description>
    Paquete de instalacion del Servicio de notificaciones "wnotif"
en su forma basica. Permite enviar/gestionar/administrar notificaciones
hacia el exterior.
Contiene los archivos relativos a la gestion, administracion, envio y
modificacion de notificaciones.
    </description>
</package>
<provides>
    <func>/etc/wnotif.bin</func>
    <func>/etc/wnotif.bin</func>
    <func>wnotif.csend</func>
    <func>wnotif.fsend</func>
    <func>wnotif.cquery</func>
    <func>wnotif.ffilter</func>
    <func>wnotif.cqryrem</func>
```

```
    <func>wnotif.fquery</func>
    <func>wnotif.cgroupAdmin</func>
    <func>wnotif.ffilteradmin</func>
    <func>wnotif.fqryrem</func>
    <func>wnotif.group</func>
    <func>wnotif.fRelCan</func>
    <func>wnotif_Notification</func>
    <func>wnotif_RELcan</func>
    <func>wnotif_Type</func>
</provides>
<depends>
    <package>WAINE-0.4.8</package>
</depends>
</wpkg>
```

La versión de **WAINE** indicada es la soportada por el sistema y la adecuada para la instalación y configuración de paquetes, independientemente que estemos usando la versión 5.10 para la creación de la instancia.

Archivo preins.sh

```
echo "This package requires functional $WPKGDIR/bin/mdbupd and
$WPKGDIR/bin/dbupd"
echo -n "Are these scripts ready ? (y/N): "
read RESP
if [ "$RESP" = "y" ]
then
 chkfile "! -s" $WPKGDIR/bin/mdbupd 502 "Installation aborted. Unready
$WPKGDIR/bin/mdbupd"
 chkfile "! -x" $WPKGDIR/bin/mdbupd 503 "Installation aborted. Can't exec
$WPKGDIR/bin/mdbupd"
 chkfile "! -s" $WPKGDIR/bin/dbupd 504 "Installation aborted. Unready
$WPKGDIR/bin/dbupd"
 chkfile "! -x" $WPKGDIR/bin/dbupd 505 "Installation aborted. Can't exec
$WPKGDIR/bin/dbupd"
else
 abort 501 "Installation aborted"
fi
```

Archivo postins.sh

```
for FILE in $SRCDIR/doc/*.sql
do
 $WPKGDIR/bin/dbupd $FILE
done
```

5.7.3 Directorio ASL

Contiene el código ASL del paquete: Cuyo contenido es el código ASL del fichero ***wnotif.asl***:

```
<?xml version='1.0' ?>
<!DOCTYPE asl PUBLIC "-//ITI//DTD XWF 0.6 //EN"
   "/usr/local/lib/waine-0.5.10/lib/asl.dtd">
<asl>
      <head>
        <meta class="ModuleInfo" name="appname" value="wnotif"/>
        <meta class="ModuleInfo" name="ver" value="1.0"/>
        <meta class="ModuleInfo" name="date" value="2023-05-31"/>
        <meta class="ModuleInfo" name="author" value="Ana Varela"/>
        <meta class="ModuleInfo" name="email"
value="anavarelarod@gmail.com"/>
    </head>

  <action id="wnotif.astart" type="php">
        <code>
        set_include_path("etc/wnotif.bin");
        require_once("wnotif_LockFile.inc");
        $lf=new wnotif_LockFile($wnotif_SERVER.".stop.lock");
        if(!$lf->getStatus())
            echo "El servicio ya se encuentra en ejecución";
        else
            echo "El servicio ha sido reanudado";
        $lf->remove();
        </code>
        <msg>es=Desea reanudar las notificaciones ?|en=Do you want to
start notifications service ?</msg>
   </action>

   <action id="wnotif.astop" type="php">
        <code>
        set_include_path("etc/wnotif.bin");
        require_once("wnotif_LockFile.inc");
        $lf=new wnotif_LockFile($wnotif_SERVER.".stop.lock");
        if(!$lf->getStatus())
            echo "El servicio ha sido detenido";
        else
            echo "El servicio ya se encuentra detenido";
        $lf->create();
        </code>
        <msg>es=Desea detener las notificaciones ?|en=Do you want to
pause notifications ?</msg>
   </action>

    <action id="wnotif.agetStatus" type="php">
        <code>
        set_include_path("etc/wnotif.bin");
        require_once("wnotif_LockFile.inc");
        $lf=new wnotif_LockFile($wnotif_SERVER.".stop.lock");
        if(!$lf->getStatus())
            echo "wnotif en ejecución";
```

```
        else
            echo "wnotif detenido";
        </code>
    </action>

        <action id="wnotif.arequest" type="php">
        <code>
            set_include_path("etc/wnotif.bin");
            require_once "wnotif.cfg";
            require_once "wnotif_services.cfg";
            require_once "wnotif_Notification.inc";

            $randomNot=rand(1, 7);
            $from="anavarelarod@gmail.com";
            $to="anavarelarod@gmail.com";
            $subject="Prueba test 1";
            $body="Hello";
            $retries=0;

            $n=new wnotif_Notification();
            $n->set($randomNot,$from,$to,$subject,$body,$retries);
            $n->request();
        </code>
    </action>


    <form id="wnotif.fsend" source="wnotif_Notification" caption="en=Send
Notification|es=Envío de notificaciones">
      <fields>
        <key source="pk"/>
        <int source="fk" caption="en=Notification Type:|es=Tipo de
notificación:" len="40" maxlen="80">
            <search>DATA:wnotif_Type;pk IN (SELECT fk FROM wnotif_RELcan
WHERE gid=%usergid);descr;</search>
            <searchfield>pk,descr</searchfield>
        </int>
        <string source="ufrom" caption="en=From:|es=De:" len="40"
maxlen="80">
            <defvalue>%username</defvalue>
        </string>
        <string source="uto" caption="en=To:|es=Para:" len="40"
maxlen="80"/>
        <string source="subject" caption="en=Subject:|es=Asunto:"
len="40" maxlen="80"/>
        <text source="body" caption="en=Body:|es=Cuerpo:" canbenull="Y">
            <width>40</width>
            <height>3</height>
        </text>
      </fields>

     <events>
```

```
      <afterinsert>
       <action type="php">
        <code>echo html_jscode("alert('Notificacion creada
correctamente');");return true;</code>
       </action>
      </afterinsert>
   </events>

   </form>


   <struct id="wnotif.csend" type="form">
      <param name="formid" value="wnotif.fsend"/>
      <param name="form_type" value="form"/>
      <param name="button_misc" value="0"/>
      <param name="button_insert" value="1"/>
   </struct>


    <form id="wnotif.ffilter" source="" caption="en=Notifications Query|
es=Consulta de notificaciones">
         <fields>
           <key source="pk"/>
           <string source="estado" caption="en=State|es=Estado" len="40"
maxlen="20" canbenull="Y">
               <search>Pendiente;Enviada;Fallida;Anulada;</search>
           </string>
           <int source="fk" caption="en=Notification Type|es=Tipo de
notificación" len="20" maxlen="20" canbenull="Y">
               <search>DATA:wnotif_Type;pk IN (SELECT fk FROM
wnotif_RELcan WHERE gid=%usergid);descr;</search>
               <searchfield>pk,descr</searchfield>
               <widget>combonull</widget>
           </int>
           <date source="cts" caption="en=Created date|es=Fecha de
creación" canbenull="Y"/>
           <string source="ufrom" caption="en=User:|es=Usuario:"
len="15" maxlen="80" canbenull="Y" attr="H">
               <defvalue>%username</defvalue>
           </string>
         </fields>
    </form>

    <form id="wnotif.fquery" source="wnotif_ViewNotificationStat"
caption="en=Notifications|es=Notificaciones">
      <fields >
         <key source="pk"/>
         <string source="pk" caption="en=Code|es=Código" len="40"
maxlen="80"/>
         <string source="fk" caption="en=Type|es=Tipo" len="40"
maxlen="80">
           <search>DATA:wnotif_Type;;;</search>
           <searchfield>pk,descr</searchfield>
         </string>
```

```
        <string source="cts" caption="en=Created|es=Creado" len="40"
maxlen="80"/>
        <string source="sts" caption="en=Sent|es=Enviado" len="40"
maxlen="80"/>
        <string source="uto" caption="en=To|es=Para" len="40"
maxlen="80"/>
        <string source="subject" caption="en=Subject|es=Asunto" len="40"
maxlen="80" />
      </fields>
   </form>

    <struct id="wnotif.cquery" type="relation">
        <param name="form_split" value="rows=150,*"/>

        <param name="formid" value="wnotif.ffilter"/>
        <param name="form_type" value="filter-where"/>
        <param name="form_subtype" value="wide"/>
        <param name="form_filterw_oper" value="=;=;LIKE;=;"/>
        <param name="form_filterw_nexus" value="and"/>
        <param name="button_misc" value="0"/>

        <param ord="2" name="formid" value="wnotif.fquery"/>
        <param ord="2" name="form_type" value="list"/>
        <param ord="2" name="button_misc" value="0"/>

        <param ord="2" name="__nofill" value="1"/>
    </struct>

    <form id="wnotif.ffilteradmin" source=""
caption="en=Notifications/Cancel Query|es=Consulta y anulación de
notificaciones">
          <fields>
            <key source="pk"/>
            <string source="estado" caption="en=State|es=Estado" len="30"
maxlen="20" canbenull="Y">
                <search>Pendiente;Enviada;Fallida;Anulada;</search>
            </string>
            <string source="agent" caption="en=Agent|es=Agente" len="10"
maxlen="20" canbenull="Y"/>
            <string source="ufrom" caption="en=User:|es=Usuario" len="15"
maxlen="80" canbenull="Y" />
            <int source="fk" caption="en=Notification Type|es=Tipo de
notificación" len="20" maxlen="20" canbenull="Y">
                <search>DATA:wnotif_Type;;;</search>
                <searchfield>pk,descr</searchfield>
                <widget>combonull</widget>
            </int>
            <date source="cts" caption="en=Created date|es=Fecha de
creación" len="10" maxlen="20" canbenull="Y"/>
          </fields>
    </form>
```

```
    <form id="wnotif.fqryrem" source="wnotif_ViewNotificationStat"
caption="en=Notifications|es=Notificaciones">
      <fields>
         <key source="pk"/>
         <string source="pk" caption="en=Code|es=Código" len="40"
maxlen="80"/>
         <string source="fk" caption="en=Type|es=Tipo" len="40"
maxlen="80">
                <search>DATA:wnotif_Type;;;</search>
                <searchfield>pk,descr</searchfield>
        </string>
         <string source="agent" caption="en=Agent|es=Agente" len="40"
maxlen="80"/>
         <string source="ufrom" caption="en=User|es=Usuario" len="40"
maxlen="80"/>
         <string source="cts" caption="en=Created|es=Creado" len="40"
maxlen="80"/>
         <string source="sts" caption="en=Sent|es=Enviado" len="40"
maxlen="80"/>
         <string source="uto" caption="en=To|es=Para" len="40"
maxlen="80"/>
         <string source="subject" caption="en=Subject|es=Asunto" len="40"
maxlen="80"/>
      </fields>

         <buttons>
          <action type="source"  caption="en=Cancel|es=Anular">
              <tooltip>Pulsa para anular una notificación</tooltip>
              <code>UPDATE wnotif_Notification SET ats=current_timestamp
WHERE pk=%pk</code>
              <msg>Notificación Anulada</msg>
          </action>
        </buttons>

    </form>

    <struct id="wnotif.cqryrem" type="relation">
        <param name="form_split" value="rows=140,*"/>

        <param name="formid" value="wnotif.ffilteradmin"/>
        <param name="form_type" value="filter-where"/>
        <param name="form_subtype" value="wide"/>
        <param name="form_wide" value="300"/>
        <param name="form_filterw_oper" value="=;=;=;=;LIKE;"/>
        <param name="form_filterw_nexus" value="and"/>
        <param name="button_misc" value="0"/>

        <param ord="2" name="formid" value="wnotif.fqryrem"/>
        <param ord="2" name="form_type" value="table"/>
        <param ord="2" name="button_misc" value="0"/>
        <param ord="2" name="button_action" value="1"/>
        <param ord="2" name="fields_readonly" value="1-9"/>
```

```
        <param ord="2" name="__nofill" value="1"/>
    </struct>


    <form id="wnotif.group" source="_group" caption="en=Group|es=Grupo">
        <datasource>_CONF/DOM/mdb.cfg</datasource>
        <orderby>name</orderby>
        <fields>
           <key source="gid"/>
           <string source="name" caption="en=Groupname|es=Grupo" len="12"
maxlen="40" msg="en=Invalid groupname|es=Nombre de grupo invalido"
                   tooltip="en=The groupname (40 characters max)|
es=Nombre de grupo (max. 40 caracteres)"/>
        </fields>
   </form>

   <form id="wnotif.fRelCan" source="wnotif_RELcan" caption="en=Allowed
Notifications|es=Notificaciones permitidas">
      <fields>
         <key source="pk"/>
         <string source="fk" caption="en=Type|es=Tipo" len="40"
maxlen="80">
                <search>DATA:wnotif_Type;;;</search>
                <searchfield>pk,descr</searchfield>
        </string>
        <fkey source="gid"/>
      </fields>

      <events>
           <beforeinsert>
           <action type="php">
                <code>return !
waine_DsGetNumRows("wnotif_RELcan","gid='%values[2]' and
fk='%values[1]'");</code>
                <msg>es=No se permiten duplicados|en=Duplicated values
not allowed</msg>
            </action>
           </beforeinsert>
       </events>

    </form>

    <struct id="wnotif.cgroupAdmin" type="relation">
        <param name="form_split" value="rows=100,*"/>

        <param name="formid" value="wnotif.group"/>
        <param name="form_type" value="combo"/>
        <param name="source_filter_where" value="gid != 1"/>
```

```
        <param ord="2" name="formid" value="wnotif.fRelCan"/>
        <param ord="2" name="form_type" value="table"/>
        <param ord="2" name="button_data" value="1"/>
        <param ord="2" name="source_filter_field" value="gid"/>

    </struct>

</asl>
```

5.7.4 Directorio files

Contiene las clases php y ficheros de configuración incluidos para el desarrollo del servicio de notificaciones, así como el desarrollo de la API.

El contenido de estos se indicó en los apartados Construcción del servicio de notificaciones y Construcción de la API.

5.7.5 Directorio doc

Contiene el código SQL de este paquete. En concreto hablamos del fichero ***wnotif_CREATE.sql***, definidas las tablas anteriormente en el apartado Construcción del modelo físico de datos.

Contiene la carga incial de datos para la tabla ***wnotif_Type.***

Indicamos el contenido completo de este fichero:

```
CREATE TABLE wnotif_Type (
  pk integer PRIMARY KEY,
  descr varchar(200) NOT NULL
);
CREATE TABLE wnotif_RELcan (
  pk integer PRIMARY KEY,
  gid int NOT NULL,
  fk int NOT NULL,

  CONSTRAINT fk_rel_can FOREIGN KEY (fk) REFERENCES wnotif_Type(pk)
 );
CREATE VIEW wnotif_ViewNotificationStat AS
  SELECT pk,fk,ufrom,uto,agent,subject,body,retries,cts,sts,ats,
         CASE
           WHEN retries=3 THEN "Fallida"
           WHEN ats IS NOT NULL THEN "Anulada"
           WHEN sts IS NOT NULL THEN "Enviada"
           WHEN cts IS NOT NULL THEN "Pendiente"
           ELSE "Error imposible"
         END AS estado
  FROM wnotif_Notification;
CREATE TABLE wnotif_Notification (
  pk integer PRIMARY KEY,
  fk integer,
  ufrom varchar(80) NOT NULL,
  uto varchar(255) NOT NULL,
```

```
  agent varchar(255),
  subject varchar(80),
  body text NOT NULL,
  retries integer,
  cts timestamp DEFAULT current_timestamp,
  sts timestamp,
  ats timestamp,

  CONSTRAINT fk_rel_can FOREIGN KEY (fk) REFERENCES wnotif_Type(pk)
);

INSERT INTO  wnotif_Type  (pk, descr) VALUES (0, 'test'), (1, 'Twitter'),
(2, 'Slack'), (3, 'Mailphp'), (4, 'Mailcmd'), (5, 'Telegram'), (6,
'Write'), (7, 'Teams');
```

5.7.6 Archivo wpk

Una vez desarrolado el contenido de cada directorio, se procede a crear el paquete haciendo uso del siguiente comando:

```
root@u-1604-64:/home/avarela# /usr/local/lib/waine-0.5.10/bin/wpkg
package wnotif_1.0
wnotif_1.0/
wnotif_1.0/meta/
wnotif_1.0/meta/preins.sh
wnotif_1.0/meta/meta.xml
wnotif_1.0/meta/postins.sh
wnotif_1.0/doc/
wnotif_1.0/doc/wnotif_CREATE.sql
wnotif_1.0/files/
wnotif_1.0/files/etc/
wnotif_1.0/files/etc/wnotif.bin/
wnotif_1.0/files/etc/wnotif.bin/wnotif_services.cfg
wnotif_1.0/files/etc/wnotif.bin/wnotif_Telegram.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif.cfg
wnotif_1.0/files/etc/wnotif.bin/wnotif_Popen.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Write.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Notification.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Teams.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_MailCmd.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Srv.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Mailphp.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_System.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_LogFile.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Slack.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_Twidge.inc
wnotif_1.0/files/etc/wnotif.bin/wnotif_LockFile.inc
wnotif_1.0/ASL/
wnotif_1.0/ASL/wnotif.asl
root@u-1604-64:/home/avarela#
```

Tras ejecutar el comando, ya tenemos el paquete en nuestro directorio de trabajo:

```
root@u-1604-64:/home/avarela# ls
DB  _PRACTICAS_ASL  public_html  waine-0.4.6.tgz  waine-0.5.10.tgz
wnotif_1.0  wnotif_1.0.wpk
root@u-1604-64:/home/avarela#
```

Este paquete seraá instalado en una instancía de aplicación para sus respectivas pruebas y comprobación del funcionamiento del servicio en el siguiente capítulo Implantación del servicio.

5.8 Manuales de usuario

En esta sección se recogen los manuales de usuario del servicio de notificaciones de wnotif. En ellos se describen las distintas funcionalidades que tiene disponible cada uno de los roles de usuario y las interfaces a través de las cuales se puede hacer uso de éstas.

5.8.1 Manual de usuario de wnotif

5.8.1.1 Introducción

Servicio de notificaciones para aplicaciones creadas con **WAINE** – wnotif.

5.8.1.2 Roles

Los diferentes roles existentes dentro de este servicio son:

- Usuario Administrador
- Usuario Regular

5.8.1.3 Manual de usuario del Usuario Administrador

En esta sección encontraremos el manual de usuario para el administrador del servicio wnotif.

Acerca de este manual

Este manual incluye una descripción general de las funciones de wnotif, que usted como administrador del servicio utilizará para realizar labores de configuración, consulta, edición o administración de este servicio.

Dado que este manual está escrito específicamente para el rol de administrador del servicio, el objetivo es ayudar a utilizar la interfaz de usuario de wnotif, sin necesidad de conocimientos técnicos.

Quién debe usar el manual

Este manual está dirigido al usuario que necesite gestionar y administrar el servicio de notificaciones de wnotif.

Cómo está organizado el manual

Para facilitar la navegación, este manual se divide en varias secciones. Cada parte con una funcionalidad concreta de wnotif. Las secciones son las siguientes:

1. Introducción

Introducción

Wnotif es un servicio de notificaciones utilizado en aplicaciones diseñadas con **WAINE**, que permite realizar envío de notificaciones a otras aplicaciones.

A través de la interfaz de usuario se permite realizar notificaciones, obtener información de las mismas y administrar el sistema.

Particularmente para el usuario administrador se definen en el siguiente apartado las funcionalidades disponibles.

Funcionalidades

Las funcionalidades a las que tiene acceso el administrador del servicio son:

Configuración	• *Detener* el servicio • *Reanudar* el Servicio • *Consultar* el estado del servicio
Administración	• Administración de *permisos* para usuarios regulares
Edición	• *Anulación* de notificaciones (no enviadas)
Consulta	• *Consulta de notificaciones* de cualquier tipo/usuario/estado

Tabla 11: Manual de usuario. Funcionalidades de administrador

Configuración. Detener el servicio

Descripción

En el Menú de Administrador, a la izquierda de la interfaz general, el administrador tiene la opción de detener el servio de notificaciones haciendo clic sobre la sección "*Detener el servicio"*.

Ilustración 36: Manual de usuario. Administrador. Detener el servicio

Tras hacer clic en ella, aparecerá un mensaje de confirmación en la interfaz general, indicando si el usuario desea detener el servicio.

El usuario deberá hacer clic en *"Si"* si desea detener el servicio de notificaciones o "*No*", en caso contrario.

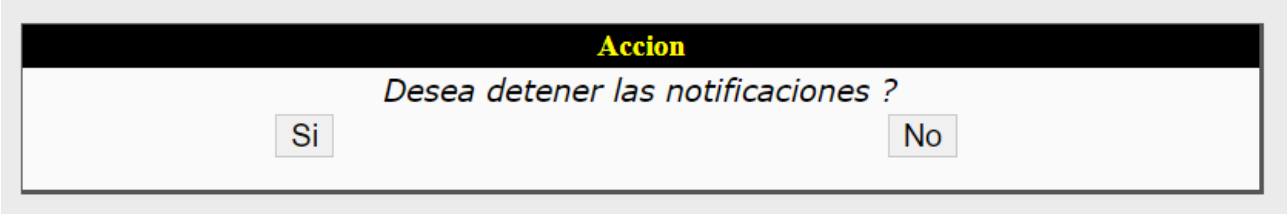

Ilustración 37: Manual de usuario. Administrador. Detener el servicio II

Después de hacer clic "Si", el usuario de administrador obtendrá dos tipos de mensajes por pantalla dependiendo del estado del servicio.

1. "El servicio ha sido detenido". Este mensaje aparece cuando el servicio se encontraba anteriormente en ejecución.

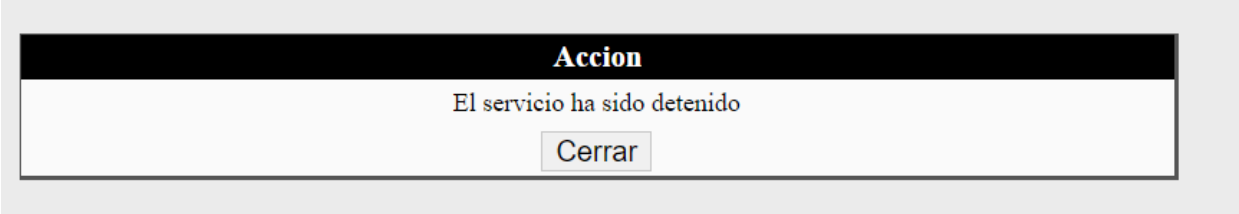

Ilustración 38: Manual de usuario. Administrador. Servicio Detenido

2. "El servicio ya se encuentra detenido". El servicio se encuentra parado actualmente.

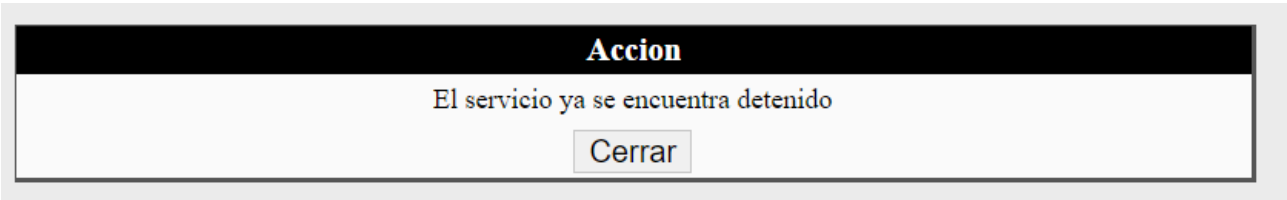

Ilustración 39: Manual de usuario. Administrador. Servicio Detenido II

Notas

Para confirmar la detención del servicio, el usuario administrador puede hacer clic sobre la sección *"Estado del Servicio"*, así mismo también puede hacer clic sobre esta sección para consultar el estado del servicio previamente.

Configuración. Reanudar el servicio

Descripción

En el Menú de Administrador, a la izquierda de la interfaz general, el administrador tiene la opción de reanudar el servio de notificaciones haciendo clic sobre la sección "Reanudar *el servicio"*.

Ilustración 40: Manual de usuraio. Administrador. Reanudar el servicio

Tras hacer clic en ella, aparecerá un mensaje de confirmación en la interfaz general, indicando si el usuario desea reanudar el servicio.

El usuario deberá hacer clic en "Si" si desea reanudar el servicio de notificaciones o "No", en caso contrario.

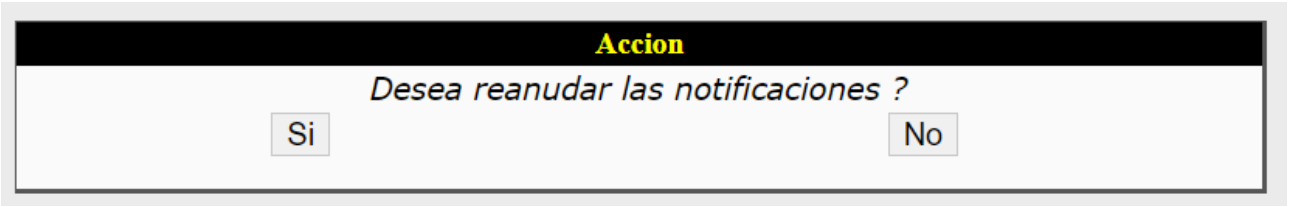

Ilustración 41: Manual de Usuario. Administrador. Reanudar notificaciones

Después de hace clic "Si", el usuario de administrador obtendrá dos tipos de mensajes por pantalla dependiendo del estado del servicio.

1. "El servicio ha sido reanudado". Este mensaje aparece cuando el servicio se encontraba anteriormente detenido.

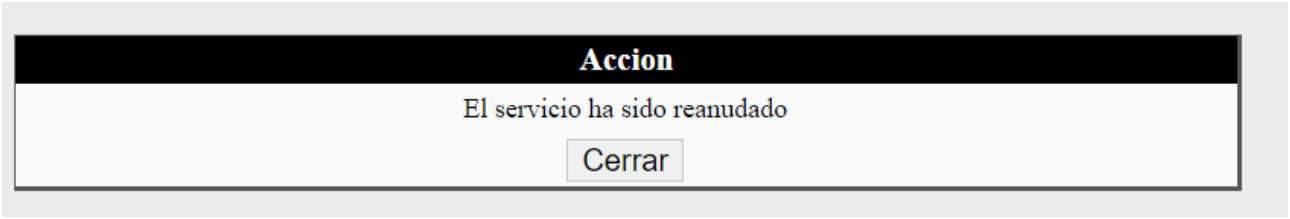

Ilustración 42: Manual de usuario. Administrador. Reanudar servicio I

2. "El servicio ya se encuentra en ejecución". El servicio se encuentra activado actualmente.

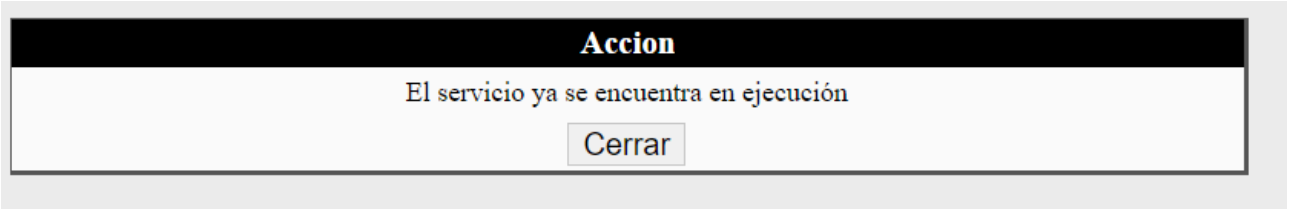

Ilustración 43: Manual de usuario. Administrador. Reanudar servicio II

Notas

Para confirmar la reanudación del servicio, el usuario administrador puede hacer clic sobre la sección *"Estado del Servicio"*, así mismo también puede hacer clic sobre esta sección para consultar el estado del servicio previamente.

Configuración. Estado del servicio

Descripción

En el Menú de Administrador, a la izquierda de la interfaz general, el administrador tiene la opción de reanudar el servio de notificaciones haciendo clic sobre la sección "**Estado del servicio".**

Ilustración 44: Manual de usuario. Administrador. Estado del servicio

Una vez hecho clic, un mensaje aparecerá por pantalla indicando el estado del servicio.

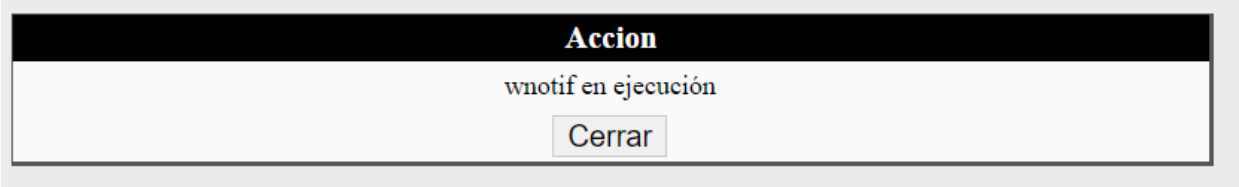

Ilustración 45: Manual de usuario. Administrador. Estado del servicio I

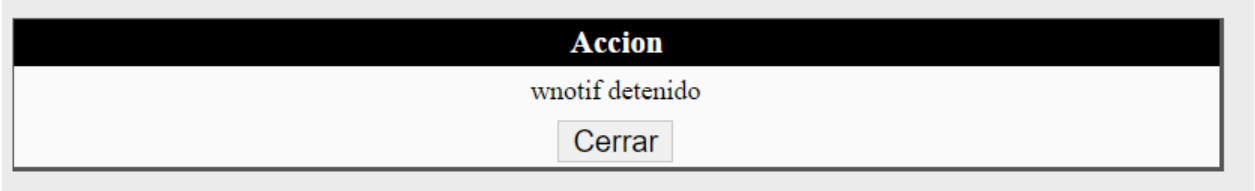

Ilustración 46: Manual de usuario. Administrador. Estado del servicio II

Administración de permisos

Como usuario Administrador, posee la funcionalidad de conceder permisos a los usuarios regulares en relación a qué tipo de notificación pueden solicitar/enviar.

Descripción

Para acceder a la interfaz de administración de permisos, el usuario administrador deberá hacer clic sobre la pestaña *"Administración de permisos"*

Ilustración 47: Manual de Usuario. Administrador. Administración de permisos

Una vez dentro observamos los siguientes interfaces :

- **Grupo**
  - Contiene el combo del mismo nombre, el usuario administrador puede seleccionar un grupo dentro de la lista a la que desea proporcionar permisos.
- **Notificaciones permitidas**
  - Tabla dinámica para añadir o eliminar tipos de notificaciones permitidas. El usuario Administrador puede añadir, modificar o eliminar el tipo de notificación

Para añadir un nuevo tipo de notificación a un grupo, el usuario administrador deberá seleccionar en el combo el tipo de notificación que desea insertar. Una vez elegida haga clic en *"Añadir"*.

Para eliminar un tipo de notificación de la tabla, haga clic en el botón "*Eliminar"* a la derecha de la fila.

IU final

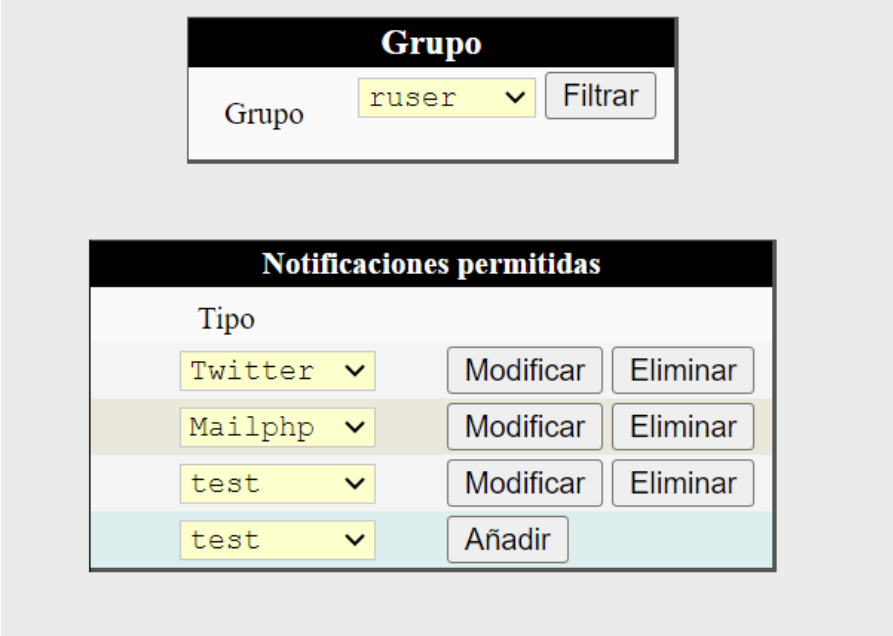

Ilustración 48: Manual de Usuario. Administrador. Administración de permisos IU

Consulta/anulación de notificaciones

El usuario administrador puede consultar el estado y la información de todas las notificaciones del sistema.

Otra de las funcionalidades que realiza el usuario administrador es el poder de anular notificaciones antes de su envío.

Descripción

En el Menú de Administrador, a la izquierda de la interfaz general, el administrador tiene la opción de reanudar el servio de notificaciones haciendo clic sobre la sección "*Consulta/anulación de notificaciones*".

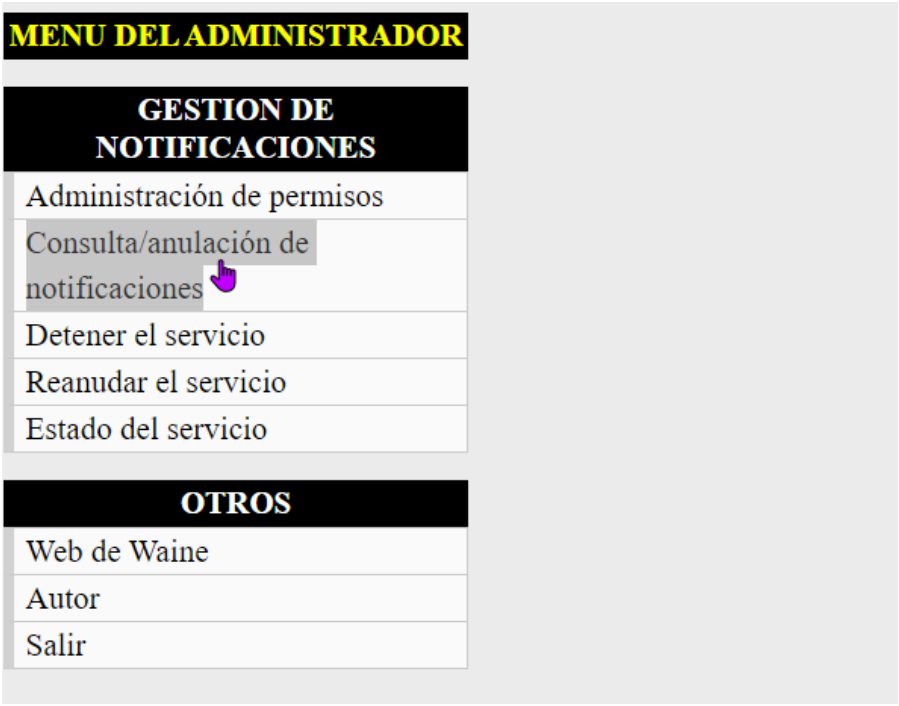

Ilustración 49: Manual de Usuario. Administrador. Menú consulta/anulación notificaciones

Una vez dentro de la pantalla vemos la primera parte en relación a la consulta:

- **Consulta y anulación de notificaciones:** Filtro donde el usuario indica qué tipo de información dentro de cada notificación desea consultar.
  - **Estado**: la notificación puede estar *Pendiente, Enviada, Fallida o Anulada.*
  - **Agente**: aplicación o subsistema que realiza la notificación. Campo opcional.
  - **Usuario:** usuario autorizado a realizar notificaciones. Campo opcional.

- **Tipo de notificación:** combo, el usuario administrador puede elegir una opción o ninguna (espacio en blanco). Si no selecciona ningún tipo se filtraran notificaciones de todos los tipos en la tabla.
- **Fecha de creación:** fecha de creación. Campo opcional. Si no se selecciona se filtrarán todas las fechas de creación registradas.

Cuando el usuario haya introducido los datos necesarios para la búsqueda, deberá hacer clic sobre el botón "*Filtrar"*.

Notas:

- Si el usuario administrador necesita deshacer los cambios o cambiar la información para la consulta ha de hacer clic en el botón "*Deshacer*".
- Para actualizar la tabla con la nueva información haga clic de nuevo en el botón "*Filtrar*".

En la tabla que se encuentra abajo de consultas, **Notificaciones**, aparecerán las notificaciones que coincidan con la información proporcionada, en caso de no encontrar ninguna la tabla se mostrará vacía.

Datos recibidos

- Identificador de la notificación o código
- Tipo de notificación
- Instante de creación de la solicitud
- Instante de envío de la solicitud (en caso de filtrar por estado "Enviada"
- Destinatario de la notificación
- Asunto

Para anular una notificación, lo único que debe hacer es hacer clic sobre el botón a la derecha de cada notificación en la tabla que indica "*Anular*". Haga clic en él y tras confirmar la anulación haciendo clic en "*Aceptar*", la notificación se marcará como anulada.

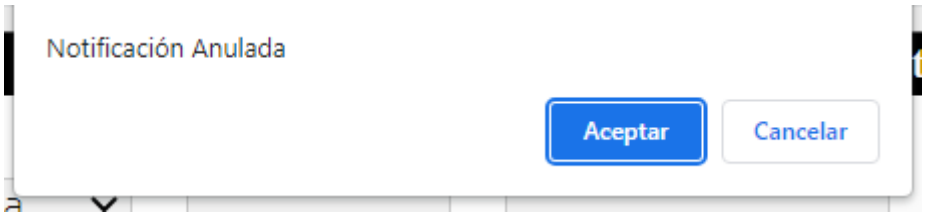

Ilustración 50: Manual de Usuario. Administrador. Anular notificación

IU Final

La interfaz de usuario final para esta consulta/anulación de notificaciones es la siguiente:

- Búsqueda efectiva:

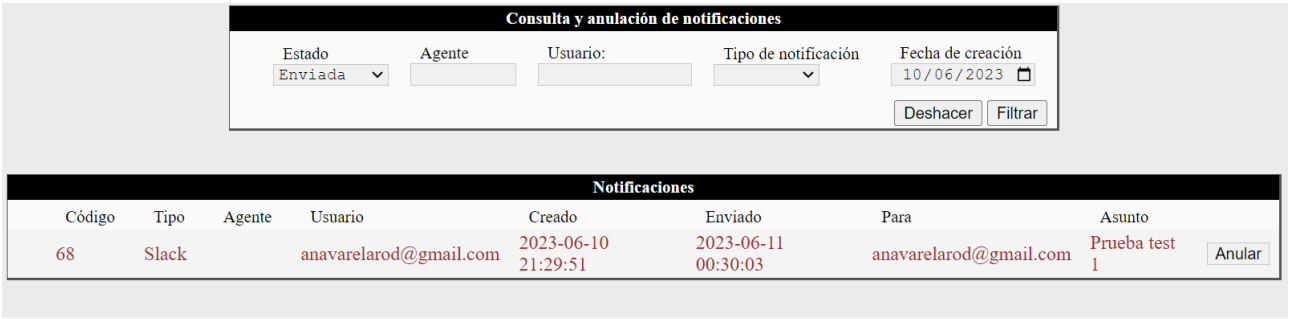

Ilustración 51: Manual de Usuario. Administrador. Consulta de notificaciones IU

- Búsqueda sin resultados:

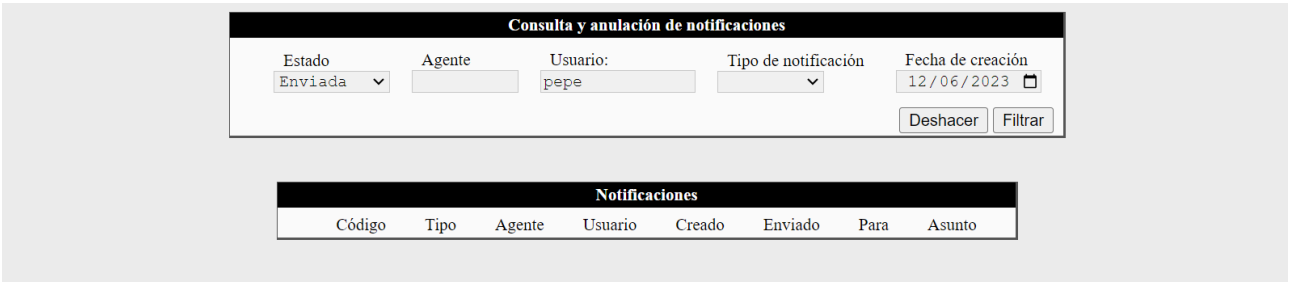

Ilustración 52: Manual de Usuario. Administrador. Consulta de notificaciones IU II

5.8.1.4 Manual de usuario del Usuario Regular

En esta sección encontraremos el manual de usuario para el usuario regular del servicio wnotif.

Acerca de este manual

Este manual incluye una descripción general de las funciones de wnotif, que usted como usuario utilizará para realizar labores de envío y consulta en este servicio.

Dado que este manual está escrito específicamente para el rol de usuario regular, el objetivo es ayudar a utilizar la interfaz de usuario de wnotif, sin necesidad de conocimientos técnicos.

Quién debe usar el manual

Este manual está dirigido al usuario que necesite realizar el envío de notificaciones utilizando el servicio de notificaciones de wnotif.

También para aquel que deseé consultar el estado de las notificaciones previamente solicitadas por el propio usuario.

Cómo está organizado el manual

Para facilitar la navegación, este manual se divide en varias secciones. Cada parte con una funcionalidad concreta de wnotif para el usuario regular. Las secciones son las siguientes:

- Introducción
- Funcionalidades

- Solicitud de notificación
- Consulta de notificaciones realizadas por el usuario
- Figuras y señalizaciones

Introducción

El sistema wnotif es un servicio de notificaciones utilizado en aplicaciones diseñadas con **WAINE**, que permite realizar envío de notificaciones a otras aplicaciones.

A través de la interfaz de usuario se permite realizar notificaciones, obtener información de las mismas y administrar el sistema.

Particularmente para el usuario regular se definen en el siguiente apartado las funcionalidades disponibles.

Funcionalidades

Las funciones que puede realizar el usuario regular son las siguientes:

Solicitud	El usuario puede enviar notificaciones de los tipos sobre los cuales esté autorizado.
Consulta	Puede consultar únicamente sobre notificaciones que el propio usuario ha realizado.
	Puede realizar varios filtros sobre la consulta.

Tabla 12: Manual de usuario. Funcionalidades de usuario regular

Solicitud de notificaciones

El usuario tiene la función de suministrar los datos para que el sistema envíe una notificación.

Descripción

Trás hacer login en la aplicación, el usuario podrá ver el menú correspondiente en la izquierda de la pantalla principal, igual al que se indica en la siguiente figura.

En dicho menú, deberá seleccionar la pestaña *"Envío de notificaciones"*.

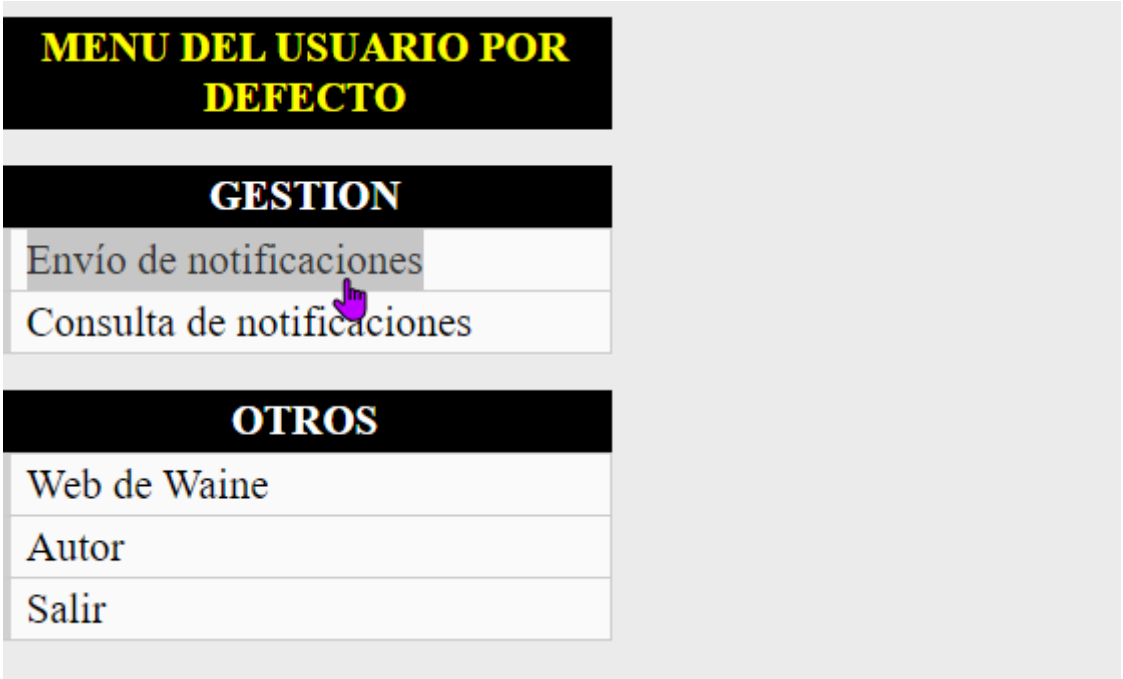

Ilustración 53: Manual de Usuario. Regular. Menú Envío de notificaciones

Al hacer clic, el usuario deberá introducir los campos correspondientes para solicitar la notificación.

Campos

- **Tipo de notificación:** Combo con el listado de tipos de notificaciones, las cuales el usuario tiene autorización para enviar. Deberá elegir una opción del combo.
- **De:** Remitente de la notificación. Campo obligatorio. Por defecto el campo de texto está predefinido con el usuario actual.
- **Para:** Destinatario de la notificación. Campo obligatorio
- **Asunto:** Asunto de la notificación, Campo opcional (obligatorio dependiendo del tipo de notificación).
- **Cuerpo:** Descripción o cuerpo de la notificación. Campo opcional, pero recomendable.

A continuación de haber rellenado los campos, el usuario deberá hacer clic en el botón de "*Añadir*" para realizar la solicitud.

Si ésta se ha insertado correctamente el siguiente mensaje aparecerá por pantalla:

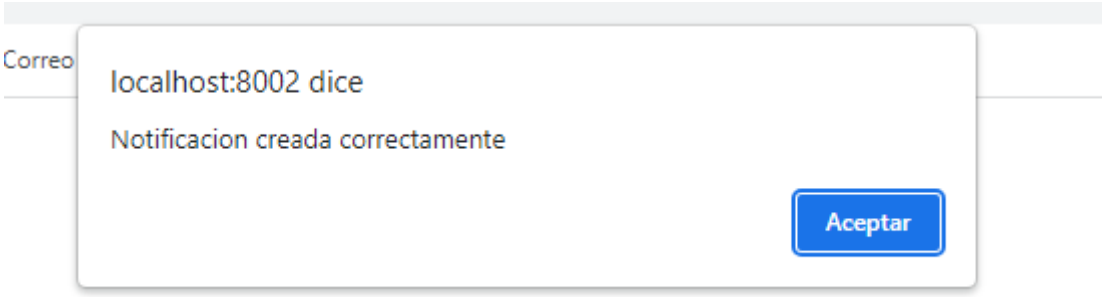

Ilustración 54: Manual de Usuario. Regular. Envío de notificaciones

En cambio si alguno de los campos obligatorios no ha sido rellenado por el usuario, aparecerá el siguiente mensaje trás hacer clic en "*Añadir*".

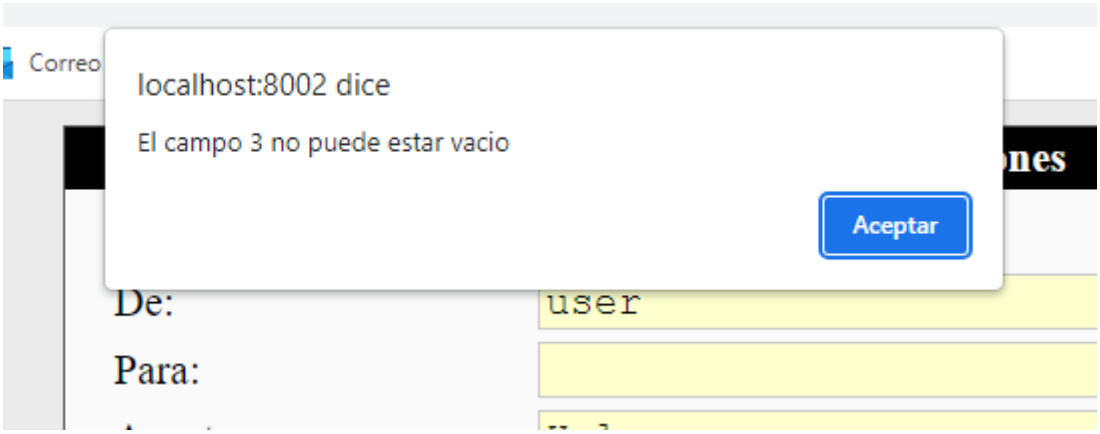

Ilustración 55: Manual de Usuario. Regular. Envío de notificaciones II

Notas

- Para volver a realizar la solicitud de una nueva notificación, el usuario deberá hacer clic de nuevo en la pestaña del menú de la izquierda "*Envío de notificaciones*".
- Se pueden realizar tantos envíos como el usuario requiera.

IU Final

La interfaz de usuario final de la consulta de notificaciones se muestra en la siguiente figura:

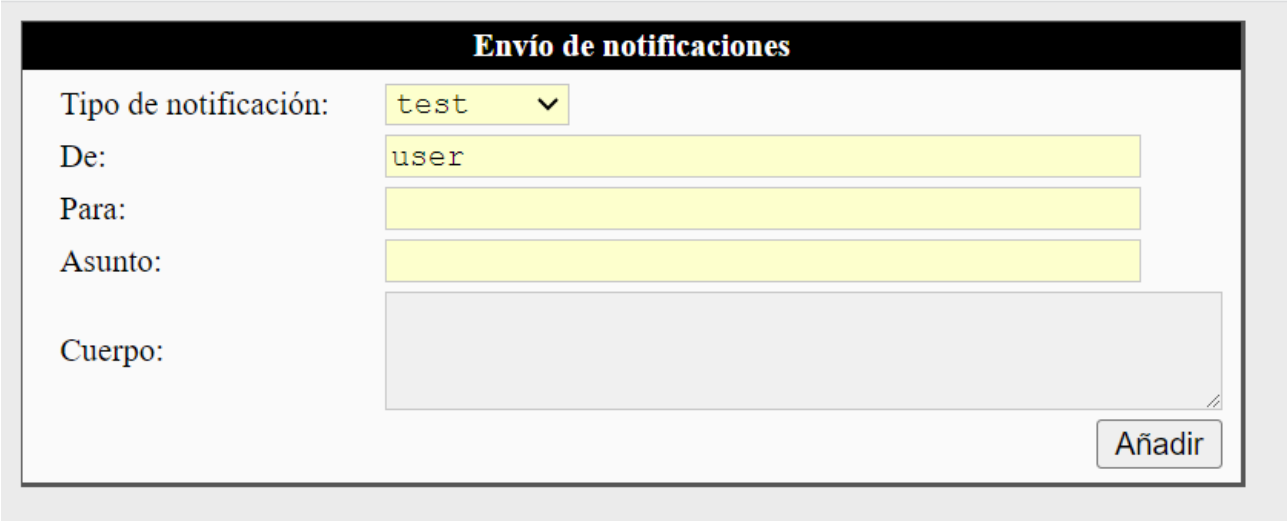

Ilustración 56: Manual de Usuario. Regular. Envío de notificaciones IU

Consulta de notificaciones

El usuario regular puede consultar información sobre las notificaciones realizadas por él mismo.

Descripción

En el menu de la izquierda de la pantalla principal el usuario podrá hacer uso de la funcionalidad de consulta haciendo clic en la pestaña *"Consulta de notificaciones"*

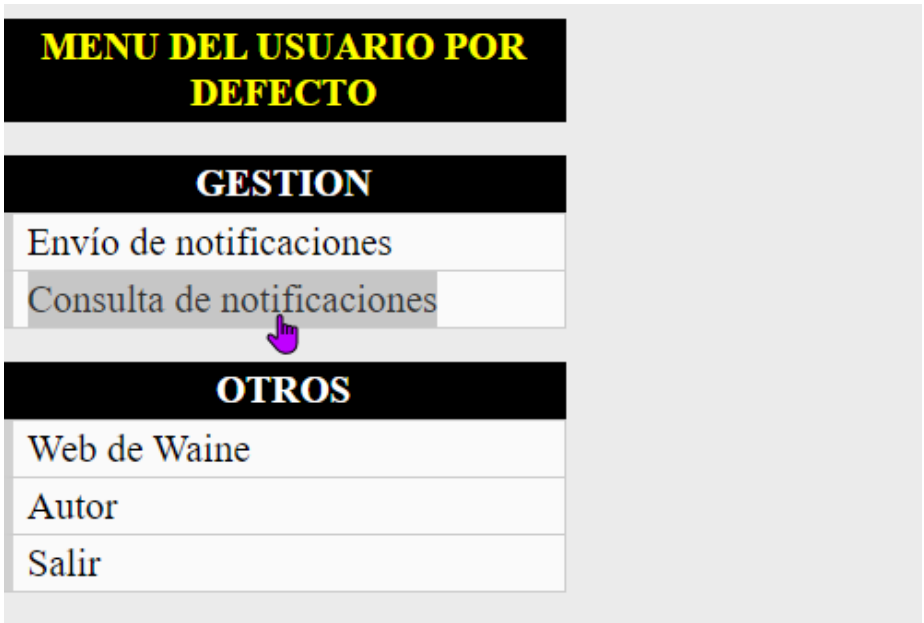

Ilustración 57: Manual de Usuario. Regular. Menú Consulta de notificaciones

Una vez dentro de la opción, el usuario podrá suministrar datos para filtrar la información que desea recibir en la consulta.

Datos a suministrar

- **Estado:** Combo a elegir entre los estados Pendiente; Fallida; Enviada o Anulada.
- **Tipo de notificación:** combo a elegir entre los tipos de notificación que se ha enviado. Se puede seleccionar dentro de los tipos permitidos por el usuario. Puede estar vacío, en este caso se filtrarán por todos los tipos de notificaciones.
- **Fecha de creación.** Campo opcional. Si no se selecciona ninguno se filtraran independientemente de la fecha.

Una vez los campos estén seleccionados, el usuario deberá hacer clic en el botón "*Filtrar*", para ejecutar la búsqueda.

Datos recibidos

- Identificador de la notificación o código
- Tipo de notificación
- Instante de creación de la solicitud
- Instante de envío de la solicitud (en caso de filtrar por estado "Enviada")
- Destinatario de la notificación
- Asunto

Notas

- Si la búsqueda ha sido correcta, aparecerán los campos en la tabla correspondientes
- En caso contrario, la tabla aparecerá vacía.
- En el caso de querer deshacer la búsqueda, el usuario deberá hacer clic en el botón "*Deshacer*"

IU Final

La interfaz de usuario final para esta funcionalidad:

Ilustración 58: Manual de Usuario. Regular. Consulta de notificaciones

5.9 Anexos

5.9.1 Configuración Twidge

Para configurar la aplicación de twidge en nuestro linux, es necesario realizar una pequeña pre configuración.

Tras hacer la instalación de twidge en nuestra máquina, debemos configurar nuestra cuenta de twitter para poder hacer uso de los servicios directamente en nuestra cuenta.

Bastará con escribir el comando *twitter setup* en la línea de comandos y tendremos que acceder a la URL que se indica.

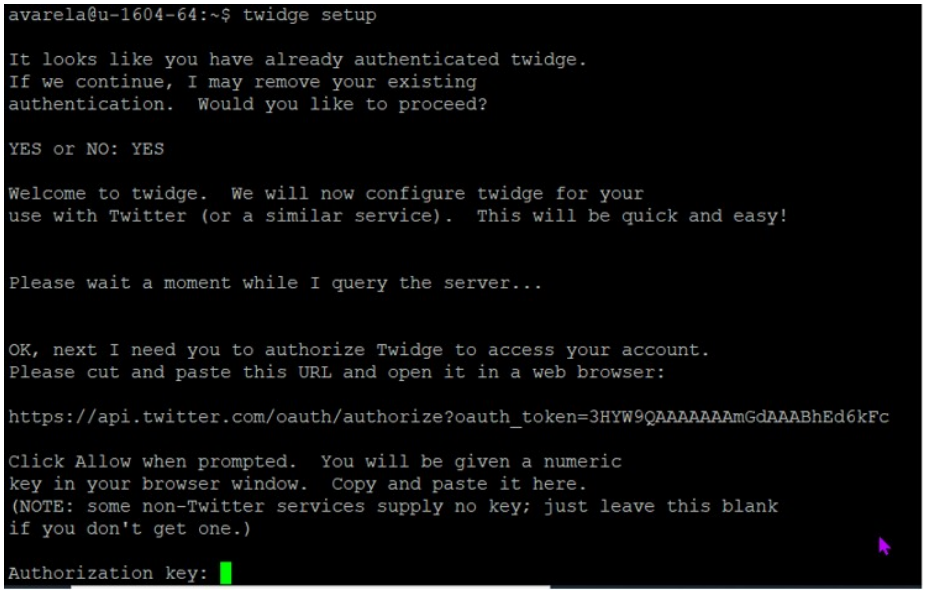

Ilustración 59: Twidge Setup

Una vez dentro, debemos aceptar las condiciones y copiar el correspondiente clave de autenticación para poder vincular nuestra cuenta.

Una vez vinculada podemos hacer uso de los comandos de twidge y se harán efectivos directamente en nuestra cuenta.

5.9.2 Configuración de Teams

Dado que hemos hecho uso de WebHook[9] entrantes para el envío de notificaciones a estas aplicaciones, tenemos que configurar esta aplicación *"Incoming Webhook"* en nuestra aplicación de escritorio (o web).

Para ello debemos de ir al canal o equipo donde queramos enviar las notificaciones y hacer clic en el menú --> conectores

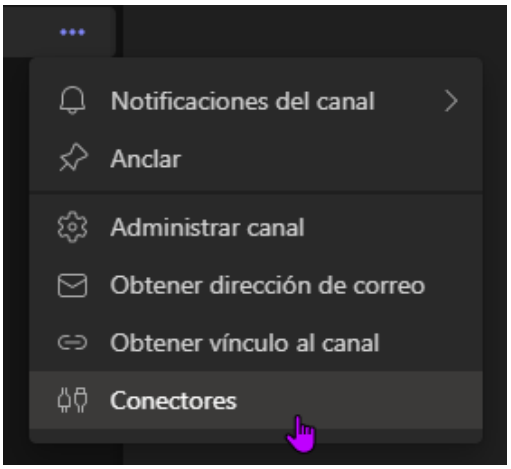

Ilustración 60: Configuración Teams - Conectores

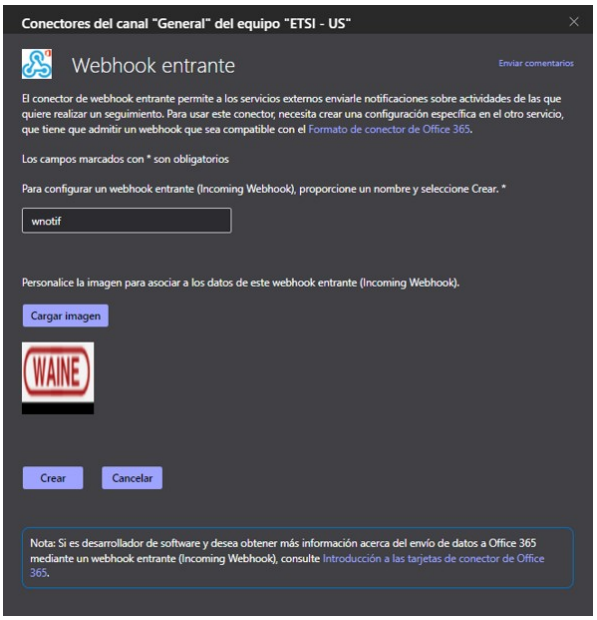

Ilustración 61: Configuración Teams - Conectores II

Una vez configurado nuestro conector, obtenemos la URL que insertaremos en nuestro servicio de notificaciones (concretamente en el archivo ***wnotif_services.cfg,*** definido anteriormente).

Así mismo comprobamos que funciona correctamente enviando una notficación de prueba.

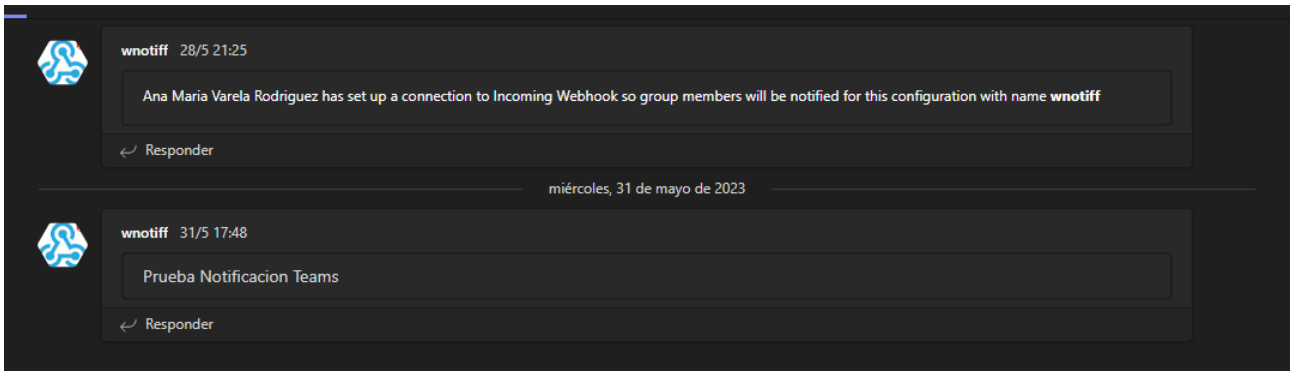

Ilustración 62: Configuración Teams - Notificación entrante

5.9.3 Configuración Slack

Para la configuración de Slack, repetimos un proceso similar al anterior,.

En nuestra aplicación de escritorio, debemos crear la aplicación de WebHook entrantes[7]en nuestra aplicación de escritorio (o web) y posterormente seleccionar qué canal querémos que reciba las notificaciones enviadas por nuestro servicio.

Hemos creado la aplicación wnotifSlack que será la encargada de gestionar la notificación y el canal #servicionotiff. para recibirlo.

Añadimos la aplicación creada y copiamos la URL entrante a nuestro archivo de configuración para los envíos.

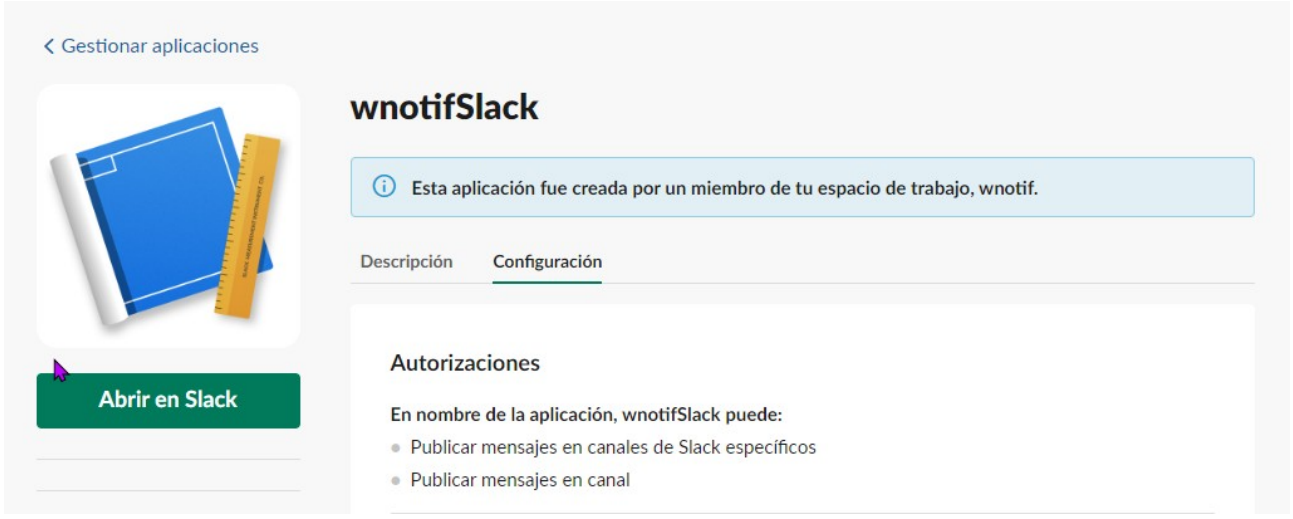

Ilustración 63: Configuración Slack

5.9.4 Código ASL

En el presente anexo se incluye el código ASL complementario de la instancia para la inclusión de los menús y de los usuarios, desarrollado para el servicio de notificación de wnotif.

- ***wnotif_menu_users.asl:***

```
<?xml version='1.0' ?>
<!DOCTYPE asl PUBLIC "-//ITI//DTD XWF 0.6 //EN"
   "/usr/local/lib/waine-0.5.10/lib/asl.dtd">
<asl>
    <head>
        <meta class="ModuleInfo" name="appname"
value="wnotif_menu_users"/>
        <meta class="ModuleInfo" name="ver" value="1.0"/>
        <meta class="ModuleInfo" name="date" value="2023-05-31"/>
        <meta class="ModuleInfo" name="author" value="Ana Varela"/>
        <meta class="ModuleInfo" name="email"
value="anavarelarod@gmail.com"/>
    </head>

   <!--<xi:include href="/usr/local/lib/waine-0.5.10/include/meta.asl"/>
-->
    <group gid="1" name="admin">
        <user  uid="1" name="admin" passwd="admin"
               mainid="main_admin" descr="Admin User"/>
    </group>

    <group gid="2" name="ruser">
        <user  uid="2" name="user" passwd="user"
               mainid="main_ruser" descr="Regular User"/>
    </group>

    <group gid="3" name="apiuser">
        <user  uid="3" name="api" passwd="api"
               mainid="main_apiuser" descr="API user"/>
    </group>

    <main id="main_admin"
          caption="en=Administrator menu|es=Menu del administrador">
       <menu caption="en=Notification Management|es=Gestion de
notificaciones">
           <option caption="en=Permission administration|
es=Administración de permisos" call="wnotif.cgroupAdmin"/>
           <option caption="en=Query/cancellation of notifications|
es=Consulta/anulación de notificaciones" call="wnotif.cqryrem"/>
           <option caption="en=Stop Service|es=Detener el servicio"
action="wnotif.astop"/>
           <option caption="en=Run service|es=Reanudar el servicio"
action="wnotif.astart"/>
           <option caption="en=Service status|es=Estado del servicio"
action="wnotif.agetStatus"/>
       </menu>
       <menu caption="en=Misc|es=Otros">
```

```
            <option caption="en=Waine page|es=Web de Waine"
                    url="http://waine.us.es"/>
            <option caption="en=Author|es=Autor"
call="meta.struct.appinfo"/>
            <option caption="en=Logout|es=Salir" url="logout.php"/>
        </menu>
    </main>

     <main id="main_ruser"
           caption="en=Default user menu|es=Menu del usuario por defecto">
        <menu caption="en=Management|es=Gestion">
            <option caption="en=Send notification|es=Envío de
notificaciones" call="wnotif.csend"/>
            <option caption="en=Notification query|es=Consulta de
notificaciones" call="wnotif.cquery"/>
        </menu>
        <menu caption="en=Misc|es=Otros">
            <option caption="en=Waine page|es=Web de Waine"
                    url="http://waine.us.es"/>
            <option caption="en=Author|es=Autor"
call="meta.struct.appinfo"/>
            <option caption="en=Logout|es=Salir" url="logout.php"/>
        </menu>
    </main>

     <main id="main_apiuser"
           caption="en=API user menu|es=Menu del usuario API">
        <menu caption="en=Notif Generator|es=Generador de notificaciones">
             <option caption="en=Notification Insert|es=Insertar
notificaciones" action="wnotif.arequest"/>
        </menu>
        <menu caption="en=Misc|es=Otros">
            <option caption="en=Waine page|es=Web de Waine"
                    url="http://waine.us.es"/>
            <option caption="en=Author|es=Autor"
call="meta.struct.appinfo"/>
            <option caption="en=Logout|es=Salir" url="logout.php"/>
        </menu>
    </main>

<
```

6 Implantación de wnotif

Ana Varela Rodríguez, anavarelarod@gmail.com 30/06/2023

6.1 Introducción

El presente documento tiene como objetivo mostrar el procedimiento de implantación del servicio de notificaciones wnotif.

6.2 Entorno de implantación

El entorno de implantación ha sido realizado en una máquina virtual basada en la imagen WDE (**WAINE** Development Ecosystem). Esta imagen cuenta con todo el software necesario para el desarrollo de aplicaciones **WAINE**. Esta basada en GNU/Linux.

- Usuario de importación: avarela
- Directorios de trabajo:
  - */var/www/html/ :* directorio en el cual se crea y desarrolla la instancia de aplicación
  - /wnotif_0 : contiene los paquetes asociados a la aplicación wnotif preparados para la instalación en la instancia de aplicación.

Los pasos a seguir para el despliegue/implantación de la aplicación wnotif son los siguientes:

1. Creación de la instancia de la aplicación
2. Configuración de los orígenes de datos
3. Configuracion de los archivos ***mdbupd*** y ***dbupd***
4. Generar la aplicación en la instancia a partir del código ASL.
5. Instalación del paquete **wnotif_1.0.pkg**

6.3 Crear una instancia de aplicación WAINE

En el capítulo anterior, definimos conceptos teóricos sobre una instancia de aplicación basada en **WAINE**.

Ponemos en practica los conocimientos para crear una instancia de aplicación de nuestro servicio de notificaciones usando la herramienta **mkapp.[13]**

```
avarela@u-1604-64:/var/www/html# /usr/local/lib/waine-0.5.10/bin/mkapp
wnotif_0
```

Tras ejecutar este comando, tendremos una instancia de aplicación llamada **wnotif_0** creada con la versión 0.5.10 del motor **WAINE**. Este comando crea toda la estructura de directorios y ficheros bajo la ruta */var/www/html/wnotif_0* necesarios para que la aplicación funcione correctamente con la mencionada versión de **WAINE.**

6.4 Configuración de los origénes de datos

Dentro de la instancia **wnotif_0** tenemos varios archivos de configuración donde se indican la ruta y el nombre de los drivers y base de datos utilizados.

El motor de base de datos utilizado es 3.11.

```
avarela@u-1604-64:/home/avarela# sqlite3 -version
3.11.0 2016-02-15 17:29:24 3d862f207e3adc00f78066799ac5a8c282430a5f
```

- Comprobamos que el fichero *db.cfg* está asociado a SQLite3 y usando base de datos DB

```
avarela@u-1604-64:/var/www/html/wnotif_0/_CONF/DOM# ls
db.cfg  dsource.cfg  mdb.cfg  txtfile.cfg  txtlines.cfg  userfunc.inc
root@u-1604-64:/var/www/html/wnotif_0/_CONF/DOM# cat db.cfg
<?php

   $DBDRIVER="dssqlite3.inc";

   $DBFILE="./DB";

?>
```

- Lo mismo para el fichero *mdb.cfg*

```
root@u-1604-64:/var/www/html/wnotif_0/_CONF/DOM# cat mdb.cfg
<?php

   $DBDRIVER="dssqlite3.inc";

   $DBFILE="./MDB3";

?>
```

6.5 Configuración de los archivos dbup y mdbup

Comprobamos si es necesario configurar los ficheros ***dbupd*** y ***mdbup*** del sistema de paquetes de la instancia para que estén enlazados a nuestro motor de base de datos.Estos scripts son los encargados de actuar sobre la base de datos y el repositorio de la itnerfaz de usuario (MDB) cuando se instala un paquete.

En nuestro caso comprobamos que ambos están configurados para las bases de datos SQLite3 en los ficheros ***dbup*** y ***mdupd***.

```
<?php
avarelar@u-1604-64:/var/www/html/wnotif_0/packages/bin# ls -ls
total 32
0 lrwxrwxrwx 1 www-data www-data  13 jun 14 18:30 dbupd -> dbupd.sqlite3
4 -rwxr-xr-x 1 www-data www-data  86 jun 14 18:30 dbupd.mysql
4 -rwxr-xr-x 1 www-data www-data  85 jun 14 18:30 dbupd.pgsql
4 -rwxr-xr-x 1 www-data www-data  93 jun 14 18:30 dbupd.sqlite2
4 -rwxr-xr-x 1 www-data www-data  94 jun 14 18:39 dbupd.sqlite3
0 lrwxrwxrwx 1 www-data www-data  14 jun 14 18:30 mdbupd ->
mdbupd.sqlite3
```

```
4 -rwxr-xr-x 1 www-data www-data 267 jun 14 18:30 mdbupd.sqlite2
4 -rwxr-xr-x 1 www-data www-data 270 jun 14 18:39 mdbupd.sqlite3
4 -rwxr-xr-x 1 www-data www-data 926 jun 14 18:30 wmcodeinst
4 -rwxr-xr-x 1 www-data www-data 606 jun 14 18:30 wmcoderpl
avarelar@u-1604-64:/var/www/html/wnotif_0/packages/bin#

?>
```

Ambos están correctamente configurados.

6.6 Agregación del código ASL a la instancia

Como hemos indicado en los capitulos anteriores, la aplicación tiene diferentes roles y funcionalidades asociados a ellos, esto es configurable luego hemos separado la parte del código ASL que define estos menús y usuarios para que la aplicación sea modular.

El código fuente del fichero **wnotif_users_menu.asl** está definido en el capitulo anterior.

Con la herramienta ***asl2mdb[15]***, gestionamos repositorios de la interfaz de usuario, creamos repositorios a partir de las especificaciones ASL, aunque también se pueden añadir elementos a un repositorio exsitente o eliminarlos.

Para agregarlo a la aplicación final ejecutamos el siguiente comando:

```
avarelar@u-1604-64:/var/www/html/wnotif_0#
/usr/local/lib/waine-0.5.10/bin/asl2mdb --create --sqlite3
/home/avarela/wnotif/wnotif_menu_users.asl
```

6.7 Instalación del paquete wnotif_1.0.wpk

En el capítulo anterior, concretamente en el apartado de construcción del paquete se ha creado el paquete **wnotif_1.0.pkg**, listo para ser instalado en una instancia basada en **WAINE**.

Ahora que tenemos preparada nuestra instanacia de aplicación, procedemos a instalar, haciendo uso de la herramienta **wpkg**, utilizando la opción *install*.

```
avarelar@u-1604-64:/var/www/html# /usr/local/lib/waine-0.5.10/bin/wpkg
install /home/avarela/wnotif_1.0.wpk wnotif_0

Installing /home/avarela/wnotif_1.0.wpk
Verfiying dependences
        WAINE-0.4.8 ok
Dependences ok
Checking conflicts
No conflicts
This package requires functional wnotif_2/packages/bin/mdbupd and
wnotif_0/packages/bin/dbupd
working with ASL in /tmp/tmp.QaPhPEdQDU/wnotif_1.0/ASL/wnotif.asl
Updating wnotif_0/MDB3 with /tmp/tmp.QaPhPEdQDU/wnotif_1.0/ASL/wnotif.asl
asl2mdb.gensqlite Init
```

```
asl2mdb.Message: wnotif_0/MDB3 already exists. Appending.
asl2mdb.Message: Generating sql /tmp/tmp.TwM13o4h6a.
asl2mdb.Message: Generating sqlite file wnotif_0/MDB3.
Updating wnotif_0/DB with
/tmp/tmp.QaPhPEdQDU/wnotif_1.0/doc/wnotif_CREATE.sql
done !


        Paquete de instalacion del Servicio de notificaciones "wnotif"
en su forma basica. Permite enviar/gestionar/administrar notificaciones
hacia el exterior.
Contiene los archivos relativos a la gestion, administracion, envio y
modificacion de notificaciones.
```

Una vez ejecutado, comprobamos el funcionamiento en nuestro navegador.

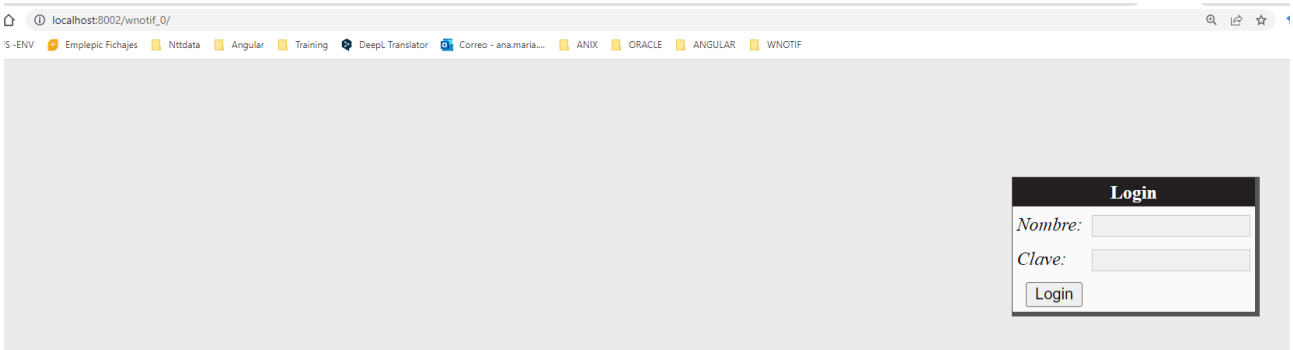

Ilustración 64: Instancia en navegador wnotif_0

6.7.1 Puesta en marcha del servicio wnotif

En esta sección se presenta la puesta en marcha del servicio, ya que mediante la interfaz de usuario para arrancar y parar el servicio se hace uso de un fichero de lock, que detiene o reanuda la ejecución del mismo.

El fichero de ejecución se encuentra en un archivo **crontab [16]**, que es ejecutado cada cinco minutos en el servidor.

La tarea que se ejecucta es la siguiente:

```
avarela@u-1604-64:/etc$ cat crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
```

```
*/5 *   * * *   root    php
/var/www/html/wnotif_2/etc/wnotif.bin/wnotif_Srv.php
```

El archivo **wnotif_Srv.php** es el responable de ejecutar el sistema. Este llama al servidor wnotif y ejecuta el servicio de notificaciones.

Activa el servicio, pero antes comprueba que no existe ningún fichero *$wnotif_SERVER.stop.lock*, que indica que el servicio ha sido parado por el usuario en la interfaz de usuario.

Si este no existe, se procede a leer notificaciones y ejecutar el servicio cada 5 minutos como se ha indicado anteriormente.

El código fuente es el siguiente:

```
<?php
require_once 'wnotif.cfg';
require_once 'wnotif_Srv.inc';
require_once 'wnotif_Notification.inc';

$l=new wnotif_LockFile($wnotif_SERVER.".stop.lock");

if(!$l->getStatus()){

        $s=new wnotif_Srv($wnotif_SERVER,$wnotif_SERVER.".lock",
$wnotif_SERVER.".log");

        echo "$wnotif_LOCKDIR/$wnotif_SERVER.lock -
$wnotif_LOGDIR/$wnotif_SERVER.log - getStatus(): ".(($s-
>getStatus())?'true': 'false')."\n";

        if(!$s->getStatus()){
                $s->start();
                $s->getPending();
                $s->send();
                $s->stop();
        }
}else{
        $l=new wnotif_LogFile($wnotif_SERVER.".log");
        $l->openFile();
        $l->appendLine('Service locked');
        $l->closeFile();
}

?>
```

6.8 Pruebas de implantación

EL objetivo de este apartado es comprobar el funcionamiento correcto del sistema integrado en el entorno de operación y permitir al usuario que, desde el punto de vista de operación, realice la aceptación del sistema una vez instalado.

Para ello hemos creado varias instancias para poder trabajar con ellas y realizar los tests.

Seguiremos el siguiente plan de pruebas para los diferentes usuarios:

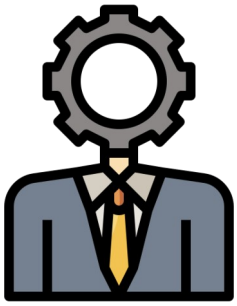 *Usuario Administrador*	*PAdmin-1.* Gestionar Permisos para Usuario Regular.
	PAdmin-2. Parar el servicio.
	PAdmin-3. Activar el servicio.
	PAdmin-4. Consultar el estado del servicio.
	PAdmin-5.Consultar notificaciones enviadas.
	PAdmin-6. Anular una notificación.

Tabla 13: Plan de pruebas usuario Administrador

 *Usuario Regular*	*PUser-1.* Solicitar una notificación
	PUser-2. Consultar notificación previamente enviada.
	PUser-3. Solicitar una notificación - Datos incorrectos

Tabla 14: Plan de pruebas usuario Regular

6.8.1 *PAdmin-1.* Gestionar Permisos para Usuario Regular

Se ejecutan los siguientes pasos para la realización de la prueba.

1. Se inicia sesión con el usuario Administrador.
2. Hacemos clic en la pestaña "*Administración de permisos*"
3. Seleccionamos el grupo al que queremos proporcionar permisos, en este caso Usuario Regular "*ruser*"
4. Se añade el tipo de notificación que se desea en el combo del formulario "Notificaciones permitidas", en nuestro caso elegimos "Slack".
5. Se intenta añadir de nuevo el tipo "Slack" para comprobar que no se permiten duplicados.

Resultado esperado:

- El usuario Administrador puede añadir el tipo de notificacion seleccionado al usuario indicado.
- No se permiten duplicados.

Resultado actual:

- El usuario Administrador puede añadir correctamente el tipo de notificacion seleccionado al usuario indicado.
- No se permiten duplicados.

A continuación se presentan las evidencias:

Ilustración 65: PAdmin-1. Administrar permisos a ruser

Ilustración 66: PAdmin-1. Tipo de notificación duplicada

6.8.2 *PAdmin-2*. Detener el servicio

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión con el usuario administrador (admin/admin)
2. Hacemos clic en la pestaña "Detener el servicio"
3. Se realiza una confirmación para detener el servicio haciendo clic en "Si"
4. El servicio se ha detenido. Aparece un mensaje en la pantalla indicando que el servicio ha sido detenido.

Resultado esperado:

- El usuario Administrador puede detener el servicio correctamente siguiendo los pasos.

Resultado actual:

- El usuario Administrador puede detener el servicio correctamente siguiendo los pasos.

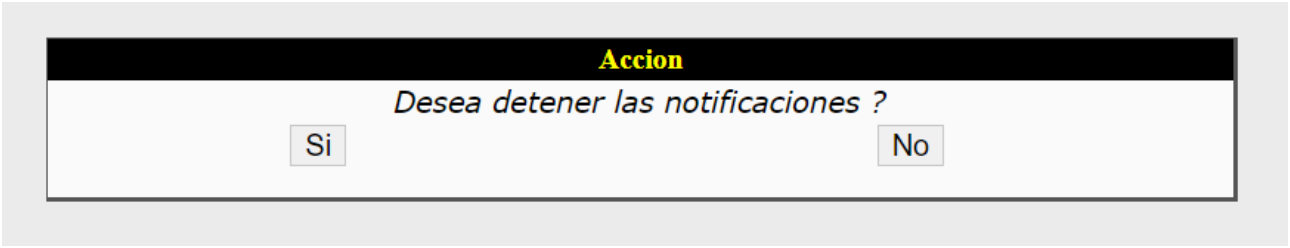

Ilustración 67: PAdmin-2. Detener el servicio. Mensaje de confirmación

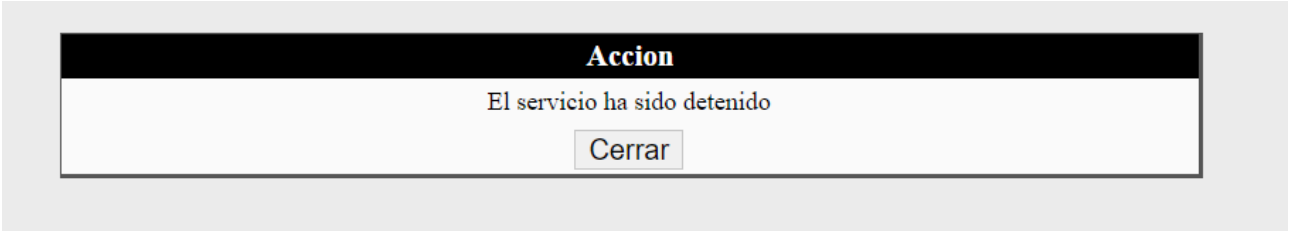

Ilustración 68: PAdmin-2. Detener el servicio. Mensaje de servicio detenido

Nota. Puede darse el caso de que el servicio se encuentre detenido actulmente, luego encontraremos el siguiente mensaje:

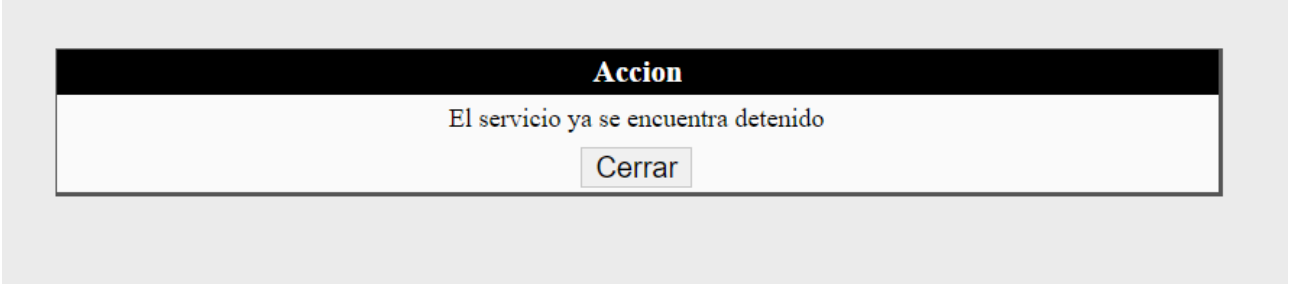

Ilustración 69: PAdmin-2. Detener el servicio. Mensaje de servicio detenido II

6.8.3 *PAdmin-3*. Reanudar el servicio

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión con el usuario administrador (admin/admin)
2. Hacemos clic en la pestaña "Reanudar el servicio"
3. Se realiza una confirmación para detener el servicio haciendo clic en "Si"
4. El servicio se ha reanudado. Aparece un mensaje en la pantalla indicando que el servicio ha sido iniciado.

Resultado esperado:

- El usuario Administrador puede reanudar el servicio correctamente siguiendo los pasos.

Resultado actual:

- El usuario Administrador puede reanudar el servicio correctamente siguiendo los pasos.

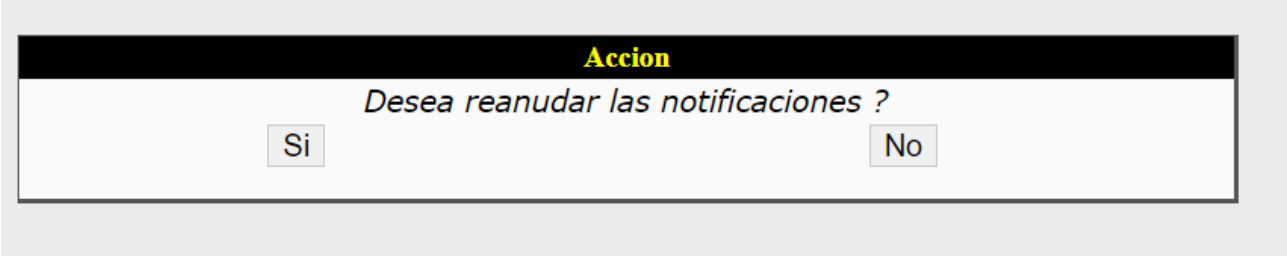

Ilustración 70: PAdmin-3. Detener el servicio. Mensaje de confirmación

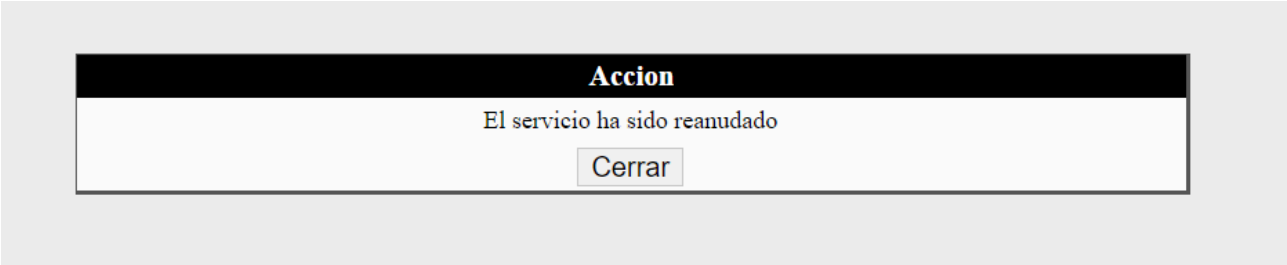

Ilustración 71: PAdmin-3. Reanudar el servicio. Mensaje Servicio Reanudado

Nota. Puede darse el caso de que el servicio se encuentre detenido actulmente, luego encontraremos el siguiente mensaje:

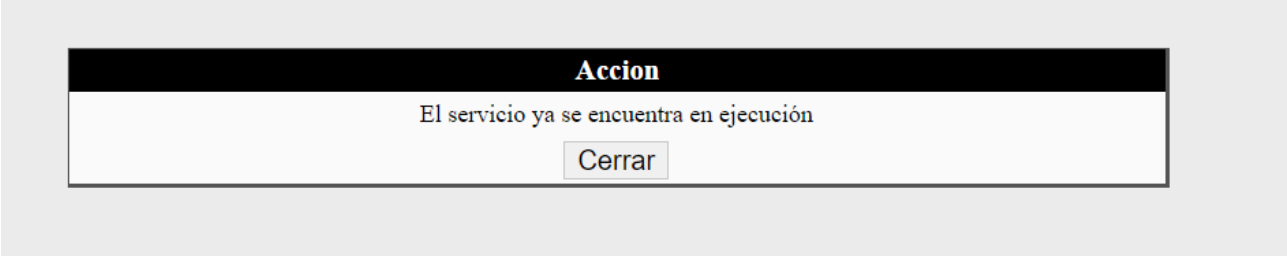

Ilustración 72: PAdmin-3. Reanudar el servicio. Mensaje Servicio Reanudado II

6.8.4 *PAdmin-4*. Consultar el estado el servicio

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión como usuario administrador (admin/admin)
2. Hacemos clic en la pestaña "Estado del servicio".
3. Aparece un mensaje por pantalla con el estado del servicio.

Resultado esperado:

• El usuario Administrador puede consultar el estado del servicio

Resultado actual:

• El usuario Administrador puede consultar el estado del servicio

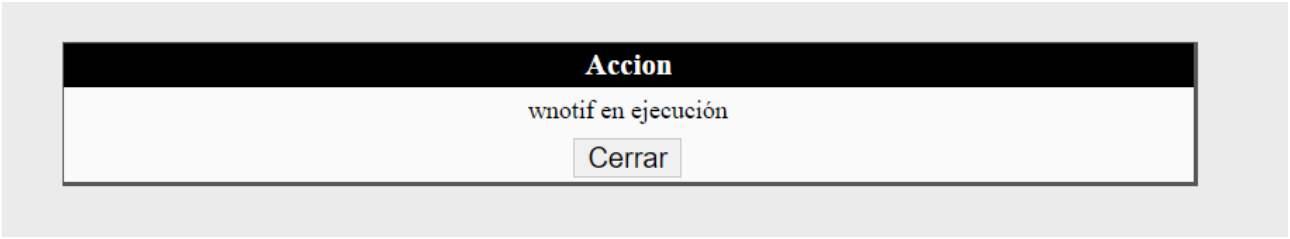

Ilustración 73: PAdmin-4. Estado del servicio

6.8.5 *PAdmin-5*. Consultar notificaciones enviadas

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión con el usuario administrador (admin/admin).
2. Hacemos clic en la pestaña "Consulta/anulación de notificaciones.
3. Se rellenan los campos correspondientes para filtrar notificaiones, en este caso filtramos por "Enviadas", luego se selecciona la opción "Enviada" en el combo "Estado".
    1. El tipo de notificación por defecto es test, pero podemos elegir el tipo de notificación haciendo el despliegue de opciones del combo. Si seleccionamos la opción vacía del combo filtraremos por todos los tipos de notificación.
    2. El resto de campos son opcionales. Si no se rellena ninguno se filtrarán todas las notificaciones que han sido enviadas, independientemente de la fecha usuario o agente.
4. Hacemos clic en el botón "Filtrar".

Resultado esperado:

- Se despliega una tabla con las notificacinoes que han sido enviadas.

Resultado actual:

- Se despliega una tabla con las notificacinoes que han sido enviadas.

Ilustración 74: PAdmin-5. Consultar notificaciones enviadas

6.8.6 *PAdmin-6*. Anular Notificación

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión con el usuario administrador (admin/admin).
2. Hacemos clic en la pestaña "Consulta/anulación de notificaciones.
3. Se rellenan los campos correspondientes para filtrar notificaiones, en este caso filtramos por "Enviadas", luego se selecciona la opción "Pendientes" (También pueden anularse las notificaciones "Fallidas") en el combo "Estado".
4. Una vez seleccionada la información se hace clic en el botón "Filtrar".
5. Obtenemos la tabla con las notificaciones pendientes, y hacemos clic en el botón de la derecha "Anular".

Resultado esperado:

- El usuario Administrador puede anular una notificación antes de su envío.

Resultado actual:

- El usuario Administrador puede anular una notificación antes de su envío.

Ilustración 75: PAdmin 6 - Anular Notificación

Ilustración 76: PAdmin 6 - Anular Notificación II

6.8.7 *PUser-1.* Solicitar una notificación

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión con el usuario regular (user/user).
2. Se hace clic en la pestaña "*Envío de notificaciones*".
3. Una vez dentro de la pantalla de envío, se rellenan los campos
    1. Tipo de notificación (sólo apareceran las aplicaciones en las cuales el usuario tiene permisos, previamente asignados por el Administrador, para enviar).
    2. Origen de la notificación
    3. Destinatario de la notificación
    4. Asunto (obligatorio solo algunos casos)
    5. Cuerpo (opcional)

Una vez indicados, se procede a hacer clic en el botón "*Añadir*".

Resultado esperado:

- La notificación ha sido enviada correctamente

Resultado actual:

- La notificación ha sido enviada correctamente.

Ilustración 77: PUser-1. Solicitar una notificación I

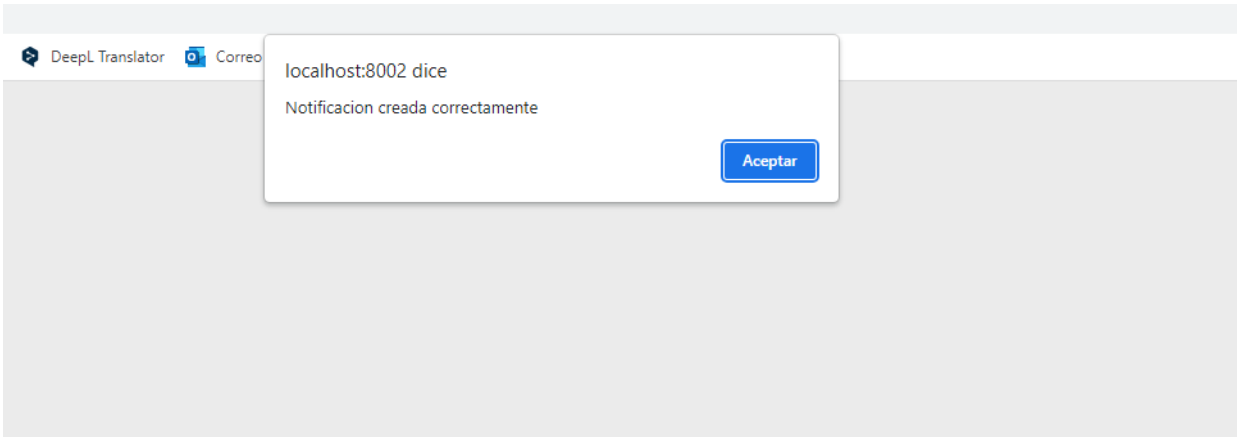

Ilustración 78: PUser-1. Solicitar una notificación II

Si todo ha ido correcto y el sistema se encuentra en ejecución. Esta notificacion será enviada.

Se recibe notificación por parte de la aplicación Slack.

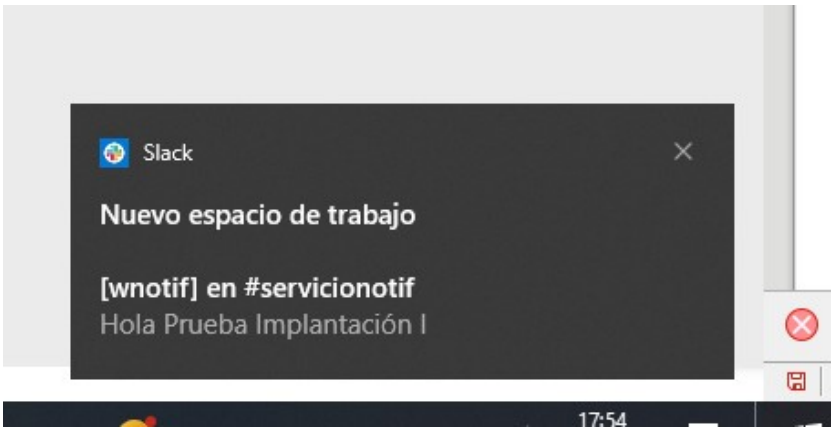

Ilustración 79: PUser-1. Solicitar una notificación III

6.8.8 *PUser-2*. Consultar una notificación

Se detallan los pasos a seguir y los resultados esperados para esta situación.

1. Se inicia sesión con el usuario regular (user/user).
2. Se hace clic en la pestaña "Consulta *de notificaciones*".
3. Una vez dentro de la pantalla de envío, se rellenan los campos:
    1. Tipo de notificación (el usuario sólo puede consultar las notificaciones que han sido enviado desde su usuario.
    2. Fecha (opcional). Si no se selecciona ninguna, aparecerán todas las notificaciones hasta la fecha.
    3. Estado (Pendiente, Enviada, Fallida, Errónea). En este caso seleccionamos "*Enviada*" para comprobar la notificación de la prueba anterior.

Una vez indicados, se procede a hacer clic en el botón "*Filtrar"*.

Resultado esperado:

- Aparece una tabla con la notificación esperada.

Resultado actual:

- Aparece una tabla con la notificación esperada.

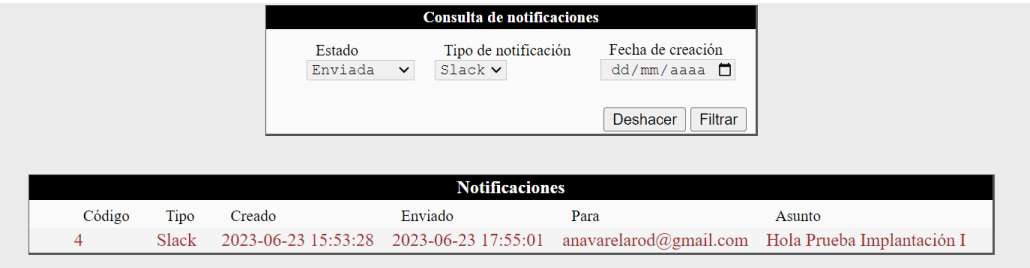

Ilustración 80: PUser-2. Consultar una notificación

7 Conclusiones

En esta sección se presentan las conclusiones de este trabajo de fin de grado. Comenzaremos con un análisis del proyecto realizado, valoración y elección de WAINE para la realización de este servicio. Seguidamente se plantea una retrospectiva con lecciones aprendidas, puntos positivos y negativos y las lineas de avance para el proyecto.

7.1 Análisis del Servicio de notificaciones wnotif

Resulta interesante realizar un análisis temporal del proceso de desarrollo del servicio.

El desarrollo del proyecto se ha divido en diferentes fases, cada una de ella con su correspondiente esfuerzo, dedicación y dificultades.

La fase de *Formación y aprendizaje* ha tenido una duración aproximadamente de una semana. Mucha de la información estaba disponible Online y a través de unas prácticas de auto formación. No se encontraron grandes dificultades, quizás se echa en falta algo más de documentación sobre todo a la hora de la implementación de las interfaces de usuario. Aún así no es trabajoso ya que tuve mucho soporte por parte del tutor Antonio Luis Delgado.

La fase de *Análisis* fue realizada por el tutor. Resultado de la misma fue un documento de Análisis. Mi tarea en esta fase consistió en leer y analizar la documentación para comprender el alcance y las restricciones del TFG.

La *Planificación del proyecto* ha tenido una duración de aproximadamente una semana. Se realiza la planificación temporal y se realiza el entregable de planificación, que aporta una descripción detallada de las fases y recursos necesarios.

En la fase de *Diseño*, se han invertido alrededor de 3 semanas. Aquí se realiza la definición del modelo físico de datos, diseño de las interfaces de usuario, casos de usos reales, diagrama de clases, especificaciones de construcción, migración y carga inicial de datos, plan de pruebas técnico y requisitos de implantación.

La fase más compleja y en la cual se ha dedicado más tiempo es la fase de *Construcción* del proyecto.

Se definen y construyen las diferentes clases en PHP del proyecto que constituyen el servidor de notificaciones y la API. Se requiere un esfuerzo aproximado de 2 meses. Aquí se encontraron varios problemas:

- La integración con aplicaciones de terceros fue algo más complicado de lo esperado.
- Búsqueda de alternativas para la implementación de otros servicios para el envío de las notificaciones.

Éstos se fueron resolviendo a medida del avance del proyecto. En esta misma fase se realizó la implementación de las interfaces de usuario en entornos WAINE, en código ASL.

Por último la fase de *Implantación* que duró aproximadamente 2 días. Se hicieron uso de las herramientas de WAINE para la gestión de paquetes resultó bastante sencillo la implementación del servicio en otras instancias.

Para la fase final de *Cierre* se invirtió aproximadamente 3 semanas en realizar la memoria del trabajo.

7.2 Ahorro en futuros proyectos

Uno de los objetivos del proyecto es Reducir el esfuerzo de desarrollo de futuros proyectos

A continuación se indican algunos aspectos por los que puede ser interesante el desarrollo de un servicio de notificaciones modular.

- ***Reutilización de código:*** Al tener un componente reutilizable, se pueden utilizar en múltiples proyectos sin necesidad de desarrollar la funcionalidad desde cero. Esto ahorra tiempo y esfuerzo de desarrollo.
- ***Mejorar la experiencia del usuario***. Al proporcionarse notificaciones relevantes, oportunas y personalizadas puede aumentar la participación del usuario.
- ***Simplificar el mantenimiento.*** Si utilizamos este componente en varios proyectos WAINE, el mantenimiento de este puede ser centralizado y así se reduce los costos asociados con la gestión de múltiples implementaciones de notificaciones.

7.3 Lecciones aprendidas

Para el desarrollo del servicio de notificaciones se han adquirido conocimientos muy interesantes, tanto de gestión como técnicos para el desarrollo de software

El proyecto trata diferentes áreas, comenzando con el aprendizaje teórico del ciclo de vida de un proyecto de desarrollo, realización de diagramas y obteniendo una experiencia real de la gestión de un proyecto de principio a fin.

Se han aprendido conceptos de programación en PHP en la fase de construcción. El servicio se ha desarrollado en este lenguaje y se ha invertido tiempo en aprender nociones básicas sobre él.

Se han empleado conocimientos adquiridos durante el grado como el lenguaje SQL, protocolos como SSH para la conexión remota, SFTP para transferencia de archivos, administración básica de Linux.

Finalmente, siendo una parte muy importante de este desarrollo, las interfaces de usuario basado en modelos. La implementación de interfaces en el entorno de desarrollo WAINE ha sido un aprendizaje nuevo e interesante. Los modelos facilitan la comprensión de la especificación final en ASL.

7.4 Puntos positivos y negativos

Como puntos positivos remarcaría:

- Contribución al conocimiento. He podido conocer y aprender de los entornos de desarrollo de interfaces de usuario basado en modelos.
- La implementación de un formulario se simplifica considerablemente con el desarrollo de interfaces en el lenguaje ASL en comparación con los lenguajes más comúnmente utilizados como HTML o Java Script.
- Durante el proceso de estudio, se han adquirido conocimientos sobre la integración de

servicios de terceros en el contexto del servicio de notificaciones, además de mejorar las habilidades de búsqueda e investigación.

- Cumplimiento de un objetivo académico.

Y en los puntos negativos:

- Poca documentación para mejorar la interfaz de usuario. Se ha requerido en varias ocasiones buscar en el dtd para encontrar la etiqueta correcta.
- La integración de algunas aplicaciones como WhatsApp , SMS o Facebook resultaron bastante complicados de configurar lo cual retrasó bastante y bloquearon la construcción. Finalmente se tuvo que prescindir de la integración de éstas.

7.5 Líneas de avance

Para finalizar, se indican las siguientes lineas de avance para seguir ampliando este servicio de notificaciones.

- ***Personalización avanzada.*** Mejora de la interfaz de usuario más personalizada.
- ***Programación de notificaciones.*** Podría indicarse la hora y fecha concreta de envío de la notificación.
- ***Integración con otras aplicaciones*** de terceros que se adapten mejor al servicio y a la aplicación en cada momento, por ejemplo Whatsapp o SMS.
- ***Estadísticas de notificaciones.*** Implementar herramientas de análisis para medir la tendencia o tasas de envío. Por ejemplo, tiempo medio de envío, porcentaje de notificaciones fallidas.
- ***Envío de notificaciones a múltiples usuarios.*** La implementación actual sólo permite el envío de una notificación por usuario con un único destinatario.
- ***Mejorar la entrega y la fiabilidad.*** En el modelo actual, se realiza el reenvío tres veces hasta que resulta fallida. Se busca la fiabilidad y confirmación del envío.

Bibliografía

1: , , , http://www.waine.org/
2: A. Delgado, A. Estepa, J.A. Troyano, R. Estepa, Reusing UI elements with Model-Based User Interface Development, 2016
3: , , , https://www.php.net/
4: , , , https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/
5: , , , http://waine.us.es/dokuwiki/doku.php?id=start&idx=waine:asl
6: , , , https://manuel.cillero.es/doc/metodologia/metrica-3/procesos-principales/psi/actividad-6/
7: , , , https://api.slack.com/messaging/webhooks
8: , , , https://core.telegram.org/bots/webhooks
9: , , , https://learn.microsoft.com/en-us/microsoftteams/platform/webhooks-and-connectors/what-are-webhooks-and-connectors
10: , , , https://pypi.org/project/telegram-send/
11: , , , https://jgoerzen.github.io/twidge/twidge-html/twidge.html
12: , , , https://manpages.ubuntu.com/manpages/trusty/es/man1/write.1.html
13: , , , http://waine.us.es/dokuwiki/doku.php?id=waine:tool:mkapp
14: , , , http://waine.us.es/dokuwiki/doku.php?id=waine:doc:sistema_de_gestion_de_paquetes
15: , , , http://waine.us.es/dokuwiki/doku.php?id=waine:tool:asl2mdb
16: , , , https://man7.org/linux/man-pages/man5/crontab.5.html