Mensajería

MENSAJERÍA

I.E.S. LEOPOLDO QUEIPO

Daniel Ramírez Sánchez

Contents

[1. Introducción y Justificación. 2](#_Toc452403699)

[2. Objetivos 3](#_Toc452403700)

[3. Analisis del contexto 4](#_Toc452403701)

[4. Desarrollo del contenido 5](#_Toc452403702)

[5. Conclusion y valoracion personal 17](#_Toc452403703)

### Introducción y Justificación.

Mi proyecto sirve para enviar un mensaje a través de la aplicación de mensajería instantánea Telegram a todos los usuarios que queramos. Está diseñado para que funcione según las necesidades de la UNED de Melilla. La aplicación solo sirve para poder enviar mensajes, no recibirlos, es decir, el usuario no podrá leer los mensajes que envien como respuesta. No es una aplicación con fines interactivos, sino una herramienta con fines informativos, para comunicar algo a los usuarios que deseemos.

La aplicación ofrece distintas posibilidades a las tres herramientas que ofrece Telegram, para comunicarnos con más de un usuario:

1. Los **grupos**🡪 se suelen utilizar para compartir cosas con amigos, familiares o pequeños grupos de trabajo. Soporta hasta 200 miembros, y se puede cambiar el nombre y la foto de perfil.
2. Los **supergrupos**🡪 si el grupo crece, se convierte en una comunidad más grande, pueden soportar hasta 5.000 usuarios con un historial único y donde los mensajes borrados desaparecen para todos los miembros del grupo.
3. Los **canales**🡪 son herramientas para difundir mensajes a grandes grupos de usuarios, tienen una cantidad ilimitada de miembros y no permite la interacción con sus miembros.

De todas estas herramientas la más parecida son los **canales**. Se podría crear un canal de la UNED Melilla en el que estarían todos los tutores como miembros, pero esto obligaría a enviar un mismo mensaje a todos los tutores a la vez, cuando ese mismo mensaje estaba destinado a algunos en concreto.

Mi aplicación resuelve este problema, permitiendo enviar un único mensaje a contactos específicos, de esta forma se puede enviar avisos sobre cambios de horario, de exámenes, reuniones, o de cualquier otro asunto. Por defecto, la aplicación envía el mensaje después de treinta minutos, pero se puede modificar para enviarlo cuando queramos.

### Objetivos

Los objetivos a alcanzar son:

1. Fomentar el uso de aplicaciones de mensajería instantánea respecto al correo electrónico, como fuente para informar a contactos de una institución pública sobre cualquier cambio. Normalmente la información en las empresas, tanto privadas como públicas, se envían a través de correo electrónico. Con esta aplicación se puede enviar la información que se desee al contacto en cualquier momento.
2. Favorecer la información inmediata en cualquier lugar. La mayoría de la población dispone de un dispositivo móvil, que tiene Telegram para poder descargar, ya sea Android, IOS o Windows Phone. Cualquier usuario de un Smartphone podrá descargarse Telegram y recibir los mensajes.
3. Una forma sencilla e intuitiva para el usuario de poder enviar mensajes de forma instantánea gracias a la interfaz. La interfaz está diseñada para que sea agradable a la vista, es sencilla e intuitiva, cualquiera sería capaz de utilizarla para enviar un mensaje.
4. Se puede aplicar a cualquier ámbito, ya sea personal o profesional. Por ejemplo, e el ámbito personal, si queremos ir al cine, podemos indicar a nuestros contactos a qué hora iremos y qué película veremos, sin necesidad de montar un grupo de Telegram.
5. Proteger la privacidad de los tutores. Podemos enviar los mensajes sin necesidad de tener el número de móvil del tutor, solamente necesitamos su alias de Telegram para enviar el mensaje que queramos. Los mensajes además estarán cifrados, y cuenta con todas las medidas de seguridad que proporciona Telegram.

### Analisis del contexto

Como he mencionado anteriormente, esta aplicación permite usar una característica que no está implementada en Telegram, para que cualquiera pueda usarla a través de su interfaz intuitiva.

Los **grupos** se utilizan para enviar un mismo mensaje a todos los usuarios que están dentro de ese grupo. Podría utilizarse, el problema es que habría usuarios que no necesitan conocer el contenido del mensaje que se va a enviar, y mi aplicación logra eludir ese inconveniente.

Los **supergrupos** son como los grupos, pero almacenan muchísima más cantidad de contactos, así que tampoco podría cumplir el cometido de mi aplicación.

Los **canales** son la única herramienta que pueden cumplir la función de mi aplicación en el sentido de que es sólo un medio informativo. Los usuarios del canal no pueden responder, pero vuelve a caer en el problema de los anteriores, el mensaje se envía a todos los usuarios pertenecientes a este grupo.

Con mi aplicación solo conociendo el alias del usuario ya se puede enviar un mismo mensaje a una cantidad ilimitada de gente cuando se quiera, salvando las distancias de un segundo entre cada mensaje enviado, así además se protege el número de teléfono del usuario que vaya a recibir el mensaje.

### Desarrollo del contenido

#### 4.1. Requisitos

Los requisitos para que funcione la aplicación son:

1. Instalar los paquetes que vienen más abajo, instalar Telegram-cli y descargarse mi aplicación.
2. Tener un Smartphone o tablet con conexión a internet y tener instalado la aplicación de Telegram (solo es requisito si se quiere llevar un mejor control de los usuarios de nuestra lista de contactos).
3. Tener un Ubuntu server configurado con acceso a internet.
4. Disponer de un número de móvil.

#### 4.2. Paquetes necesarios e instalación

Para el correcto funcionamiento de la aplicación necesitaremos instalar en nuestro servidor los siguientes repositorios:

* install apache2 instala Apache.
* install libapache2-mod-php5 php5 php5-mcrypt instala PHP5.
* install python-pip instala Python.
* install python-pexpect instala pexpect para Python.

Además de instalar Telegram-cli que es el repositorio con el que podemos enviar mensajes, la instalación es sencilla:

1. Tendremos que instalar el repositorio git: install git.
2. Luego clonamos el paquete en la ruta que deseemos: git clone –recursive https://github/vydheng/tg.git && cd tg
3. Tras clonarlo debemos instalar los siguientes paquetes para que funcione Telegram-cli: install libreadline-dev libconfig-dev libssl-dev lua5.2 liblua5.2-dev libevent-dev libjansson-dev libpython-dev make
4. Nos vamos a la carpeta de instalación e introducimos./configure
5. Compilamos el programa con make
6. Para terminar vamos a la carpeta bin dentro de la instalación de Telegram-cli y ejecutamos ./telegram-cli -k tg-server.pub para iniciar el programa, nos pedirá introducir un número de teléfono, es obligatorio poner el prefijo, en el caso de España sería +34 XXXXXXXXX.
7. Recibiremos una notificación con un código, lo introducimos en la consola del servidor y ya tendremos instalado Telegram-cli en el servidor.

#### 4.3. Instalación de la aplicación web

Lo único que debemos hacer es introducir el siguiente código en la terminal de Ubuntu:

git clone –recursive https://github.com/sirdan93/Mensajeria.git

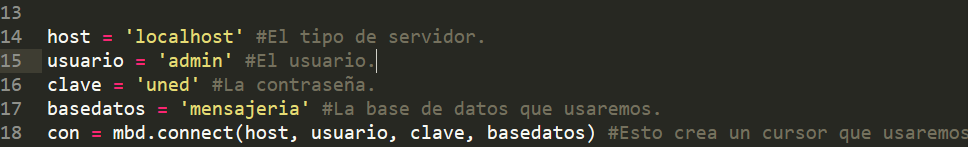
Lo copiamos en la carpeta que queramos, solo tendremos que hacer un par de modificaciones, en la carpeta llamada PHP. Dentro del proyecto se encuentra un archivo llamado script.py, que es el que se encargará de enviar el mensaje al destinatario, solo debemos modificar la ruta de instalación donde instalamos Telegram-cli

La línea 22



La ruta que viene por defecto es donde se encuentra la del localhost de Apache instalado.

La otra parte que hay que modificar pertenece a la conexión con la base de datos, y habría que hacer los cambios en dos archivos que se encuentran en la misma carpeta llamada PHP, uno es el script.py en la línea 13 para abajo tenemos conexiones a la base de datos, esta habría que cambiarla introduciendo la contraseña que tengamos en mysql para poder hacer conexión.



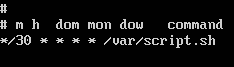
La última modificación debemos hacerla al archivo conectar.php, y tendremos que introducir los mismos cambios anteriores para que todos se guarden en la base de datos.

Una vez hecho esto solo nos queda, como medida de seguridad, guardar el archivo script.py en /var, para que esté más protegido, ahora creamos otro archivo. Como recomendación podemos guardarlo en /var para que esté más seguro y llamarlo por ejemplo ejecutar.sh, dentro escribiremos lo siguiente:



Con esto ejecutaremos el archivo script.py, solo debemos cambiar la ruta donde tengamos el archivo.

Por último escribir crontab –e en la terminal de Ubuntu y escribir la siguiente línea:



Y listo, ahora se ejecutará el script cada 30 minutos de cada hora, solo habría que añadir la base de datos, y empezar a agregar tutores y mensajes desde la aplicación web y a los 30 minutos los mensajes se enviarán si has instalado Telegram-cli y la aplicación correctamente.

#### 4.4. Desarrollo

En el mercado hay varias aplicaciones de mensajería instantánea, las más conocidas son Whatsapp y Telegram. Optamos por Telegram porque whatsapp no tiene cliente para consola de forma legal, es decir proporcionada por Facebook, hay repositorios ilegales con los que se puede “trastear”, pero tras usarlo varias veces te pueden banear el acceso del teléfono.

Telegram en cambio tiene cliente para consola llamado Telegram-cli, con la que se puede interactuar desde cualquier maquina Ubuntu, no hay peligro de baneo, las conversaciones están cifradas, es gratuito, y es libre.

La idea al principio era que la parte del código php interactuara con la base de datos y con el servidor a la vez, pero eso traía una serie de problemas, ya que era obligatorio usar el comando shell\_exec, este se usaba dentro de un archivo php, y permitía ejecutar comandos que se usarían en la terminal de Ubuntu. El truco consistía en usar el comando para abrir Telegram-cli y enviar un mensaje a cualquier usuario agregado a la lista de contactos, el problema era que el nombre del usuario y el mensaje debía introducirse manualmente porque no se podían almacenar los datos en variables y luego usar variables dentro del comando shell\_exec para usa en Telegram-cli.

No era posible desarrollar la aplicación de esa forma porque necesitaba crear variables para almacenar el alias del usuario y el mensaje, así que optamos por que el php y el servidor no se comunicaran de ninguna forma, todos los datos tienen que pasar por la base de datos. De esta forma a través del php se envían los datos obtenidos de los tutores y los mensajes a la base de datos, y de esta a través de un script de Python se hace las consultas necesarias para obtener los datos que necesitamos, en nuestro caso el mensaje que queremos enviar y el alias, ejecutar Telegram-cli y realizar el envío.

La base de datos consta de 5 tablas de datos:

-La tabla **facultad**🡪 guarda las facultades disponibles en la UNED de Melilla, no se puede agregar ninguna a través de la interfaz de la aplicación, solo se pueden insertar agregándolas a través del fichero donde se encuentra la base de datos. Cada tutor pertenece a una facultad.

-La tabla **tutor**🡪 guardará los datos personales del tutor que queramos almacenar, su nombre, apellido, su alias y la facultad a la que pertenece. La forma de enviar los mensajes a través de Telegram-cli será a través del alias, así que será obligatorio que todos se creen uno en Telegram. Un tutor puede recibir uno o muchos mensajes, y solo pueden pertenecer a una facultad.

-La tabla **usuario**🡪 almacena el número de usuarios que harán uso de la aplicación, no se pueden agregar nuevos usuarios a través de la aplicación ya que la UNED tiene su propia forma de administrarlos, así que está hecho de forma sencilla, y los nuevos usuarios deberán agregarse a la base de datos de forma manual.

-La tabla **mensaje**🡪 aquí se registran todos los mensajes enviados junto con el autor que lo ha escrito. El autor del mensaje será el usuario que ha iniciado sesión. Un mensaje puede ser enviado a uno o muchos tutores, además se pueden reutilizar los mensajes para enviárselo a otros tutores.

-La tabla **enviar**🡪 cuenta con el registro del mensaje que se desea enviar, la fecha en la que se ha creado y en la que se ha enviado para llevar un log, el tutor al que se desea enviar y el campo enviado es booleano para saber si se ha enviado el mensaje o no. Cada tutor tiene su propio envío, incluso si varios tutores van a recibir el mismo mensaje este envío está dividido en el número de tutores que vayan a recibir dicho mensaje.

Es una base de datos sencilla y creo que no necesita nuevas tablas, todo está en su justa medida ya que la información que vamos a manejar es poca.

Desde el comienzo del proyecto la base de datos ha sufrido muchas modificaciones. Al principio usaba el motor de almacenamiento Innodb, con el paso de tiempo lo cambié a Mariadb ya que el anterior se está quedando anticuado y Mariadb está pensado para cambiar un servidor por otro directamente. Además la tabla facultad formaba parte del campo tutor, pero tuve que cambiarla para almacenarlos en su propia tabla.

Llegué a eliminar la **tabla** **mensaje** para insertar los mensajes y el autor directamente en la **tabla enviar**, pero no podía acceder a los datos y tuve que crearla de nuevo, también había repetición de datos en el alias y lo solucioné, y varios fallos más de diseño que con el tiempo se ha ido resolviendo.

La aplicación cuenta con un código de php bastante sencillo, creado de forma modular y a través de variables, básicamente la mayoría son datos que se obtienen por método POST y métodos GET, algunas variables de SESSION para usar el nombre del usuario que ha iniciado sesión y poco más.

Al principio pensé en diseñarlo por clases y objetos pero al ser tan poco código no creo que hiciera falta, hecho de esta forma es muy sencillo, y cada línea de código está explicada y comentada para que cualquier usuario pueda modificarlo si quiere. Se puede mejorar y pasar el código a clases y objetos, pero no le veo la necesidad

Se pueden consultar los usuarios disponibles en la página sin visualizar sus contraseñas, solo pueden verse sus nombres y apellidos. Podemos agregar nuevos tutores, consultarlos a través de un buscador, ver los mensajes que se han enviado o están pendientes de enviar, tienen su propio buscador para que consultarlos sea más sencillo. Además de agregar nuevos mensajes y elegir el número de tutores que queremos que los reciban. Los tutores pueden modificarse completamente y eliminarse, podemos reutilizar un mensaje que hayamos enviado anteriormente y enviárselo a cualquier tutor que queramos.

El CSS de la aplicación tiene un estilo sencillo pero elegante, con marcadores de colores en las opciones más importantes siendo el verde el de agregar tutores o enviar información, el amarillo para modificar tutores o reutilizar un mensaje, y el rojo para volver atrás o eliminar un tutor. Estos ayudan a diferenciarlos rápidamente y que sea más sencillos de reconocer.

El CSS es una modificación propia de greyscale.css de la página www.startbootstrap.com, he quitado todo el Jquery, los desplazamientos por secciones dentro de la propia página, la barra de navegación la he modificado para que no aparezca cada vez que hay movimientos, la fuente de las letras las he cambiado a unas más legibles, y la imagen de fondo es de tono minimalista, para darle simplicidad al conjunto. He modificado los botones de información que venían por defecto agregando colores nuevos y efectos para mejorar la relación entre usuario e interfaz.

La página usa Bootstrap para el estilo sencillo que ofrece, además que es muy personalizable, ofrece un estilo minimalista que está de moda y lo llevan todas las aplicaciones web, y además ofrece un diseño responsable, permitiendo que se amolde a multitud de pantallas, por si la aplicación desea llevarse en un futuro a dispositivos móviles.

Al principio iba a utilizar CSS limpio, pero luego pensé que añadiendo Bootstrap quedaría un diseño más elegante. Además de que ofrece más posibilidades a nivel de diseño.

El servidor se comunica con la base de datos a través del script de python, como dije anteriormente la idea al principio era que el servidor interactuara con la parte de php, eso obligaba a usar un comando que daba varios problemas, es un buen método si se va a introducir de forma manual el usuario y el mensaje, pero no para hacer que se introduzca de forma automática, así que la idea desembarco en crear un script de python o de Linux para poder hacer la tarea que necesitaba.

Al principio lo iba a hacer con un script de Linux, pero no podía guardar las variables para usarlas en Telegram-cli y eso era un problema, además de que entonces solo serviría para servidores con Linux.

Al final opte por usar python, no solo había más documentación, sino que además era más sencillo de programar, y tenía ya una base del año anterior. Lo mejor es que además de ser un código sencillo y de fácil lectura, se puede amoldar a cualquier plataforma, facilitando su adaptación dependiendo del servidor que queramos.

##### 4.3.1. ¿COMO funciona el script y Telegram-cli?

Todo el código está explicado en el fichero script.py, pero aquí explicaré mejor como funciona. Repito que el objetivo de esta aplicación no es crear un chat, sino una forma de enviar mensajes a los contactos que queramos, son mensajes de difusión nada más.

Para que este script funcione tendremos que importar 3 módulos:

* Pexpect 🡪 que se usa para ejecutar programas externos, en este caso python.
* Time 🡪 para usar el tiempo a la hora de insertar el registro de envío.
* MySQLdb as mbd 🡪 es un módulo con un método que sirve para hacer consultas y ejecutar acciones con SQL.

Es necesario recoger los datos de conexión a la base de datos, seleccionando el servidor, el usuario, la contraseña y la base de datos que vamos a utilizar, todo ellos lo guardaremos en variables para poder utilizarlos mejor, ya que a continuación con el método mdb.connect vamos a guardar en una variable la conexión.

Luego iniciamos Telegram, es necesario esperar dos segundos para cada carga que vayamos a hacer, por ejemplo, al principio ejecutamos Telegram y esperamos dos segundos, esto lo hago porque el programa necesita cargarse, si no esperamos, es posible que no se inicie a tiempo o no lo haga correctamente.

Telegram-cli tiene un bug que es raro de ver, pero es posible que aparezca durante su ejecución y es que el sistema no hace nada, se queda bloqueado y ni envía mensajes ni recibe, la única forma conocida para solucionarlo es cerrar y abrir el programa de nuevo.

La segunda carga de dos segundos se realiza cuando cargamos la lista de contactos, Telegram-cli tiene un problema, y es que al iniciarlo necesita refrescar la lista de contactos, si no, no podemos enviar el mensaje a ese usuario. Se puede evitar si, casualmente, al iniciar Telegram-cli el usuario al que quieres enviar el mensaje te habla, o le hablas a él por la aplicación de escritorio o Smartphone, pero es más sencillo refrescar la lista de contactos y esperar esos dos segundos. Tengo que decir que he puesto dos segundos porque es lo máximo que he podido bajar con el número de contactos que tengo agregados, tengo unos cincuenta más o menos, no creo que con más usuarios este tiempo necesite aumentar porque lo hace muy rápido, pero a lo mejor después de un gran número de contactos si haga falta aumentar a un segundo más, pero no estoy seguro.

La lista de contactos tiene un problema, no entiendo muy bien el por qué, pero al refrescar la lista de contactos se muestran números de personas que hace tiempo que eliminaste o no tienes registrado en el móvil, es posible que aparezcan algunos contactos de tus contactos y esto puede comprometer su privacidad.

A mí por ejemplo me aparecían contactos con los que hace años que no tengo ninguna relación y ni siquiera los tengo guardados en el móvil, y estaban en un móvil antiguo, y a otros conocidos que los he usado para hacer pruebas le aparecen contactos que llevaban eliminados mucho tiempo y no le aparecían en la propia aplicación de Smartphone de Telegram, es más a algunos les ha aparecido contactos míos, o de otro compañero con solo tenerlo agregado a su lista de contactos.

Es importante saber esto porque puede comprometer la privacidad y seguridad de los usuarios que tengamos agregados.

Después creamos un cursor donde guardaremos un método de mbd que sirve para crear cursores y nos vendrá de perlas para hacer que funcione la aplicación, se hace la primera consulta que va a seleccionar los mensajes que no han sido enviados, se ejecutan con cur.execute que básicamente usa el cursor y execute para ejecutar la variable y obtener un resultado, se guarda en una variable y con cur.fetchall() muestra cada uno de los resultados obtenidos en la consulta.

Una vez hecho esto creo un bucle for para recorrer los resultados obtenidos en la consulta y que se ejecute para cada uno de los resultados todo lo que esté dentro del bucle. Luego se seleccionan los mensajes a través de su id y se sigue el mismo proceso anterior, solo que esta vez en vez de recorrer todos los campos de la tabla enviar, solo obtenemos el campo de texto en la tabla mensaje.

Lo mismo con la tabla tutor, pero esta vez cogemos el campo de alias, como he mencionado en otras ocasiones utilizamos el alias para no tener que usar el número de teléfono del usuario o su nombre, es mucho más cómodo ya que los nombres se pueden repetir y puede que haya conflictos, en cambio cada alias es único e irrepetible.

A la hora de enviar un mensaje por Telegram-cli nos encontramos con un problema, y es que para que esta aplicación web funcione se necesita de un número de teléfono, esto puede ser complicado a la hora de usarlo para instituciones públicas, ya que sería necesario que algún administrador dejara su número de teléfono para usarlo, e implicaría que todos sus contactos estuvieran disponibles al usar contact\_list utilizando Telegram-cli, puede adquirirse un número que sea único y exclusivamente para el uso de esta aplicación web, es la única solución viable para no poner en peligro la privacidad del número de teléfono.

Mi consejo, además, es que si se va a utilizar Telegram exclusivamente para esto solamente se instale Telegram-cli, no la aplicación de Telegram de esta forma no llega ningún mensaje de los usuarios, ya que sólo se va a iniciar única y exclusivamente para enviar el mensaje y luego se cierra. Se puede añadir un mensaje en cada envía que diga que no respondas al mensaje porque nos vas a recibir ninguna respuesta, ya que como dije anteriormente esta aplicación no es para interactuar entre el emisor y el receptor, es para que el emisor envíe un mensaje y el receptor lo lea y ya está.

Después se envía el mensaje al tutor deseado, es obligatorio poner como mínimo un segundo entre mensaje y mensaje, ya que si se envían varios mensajes a la vez es posible que algunos se envíen y otros no se envíen, con el segundo de espera entre cada mensaje permitimos que el programa tenga ese tiempo para enviar el mensaje, he comprobado y metido mensajes más largos y se envían sin problemas, aunque he descubierto que tiene un límite de caracteres que se pueden enviar y tiene que haber un mínimo de caracteres.

La aplicación tiene una serie de problemas que son difícil de solucionar en lo referente al envío de mensajes:

* No hay forma de averiguar si el mensaje ha llegado a su destinatario, es difícil porque el usuario tendría que perder la conexión a internet, nadie hoy en día suele estar más de 30 minutos sin internet, el mensaje le llegaría, pero con retraso.
* Esto es más raro aún, pero puede ocurrir, el mensaje se envía y milésimas de segundo después se hace la consulta a la base de datos se actualiza la tabla enviar con la fecha de envío e indicando que el mensaje ha sido enviado, pero puede ser que a la hora de enviar el mensaje desaparezca internet justo se ha iniciado Telegram y entrando en el bucle marque los mensajes como enviados a pesar de no estar enviados, esto es muy muy raro que ocurra ya que no depende del móvil, sino de la red del servidor y su conexión.

Se actualiza el mensaje de la tabla **enviar** a **enviado** y se introduce la fecha de envío para llevar un seguimiento del registro de envíos. Luego se cierra Telegram, la conexión con la base de datos y el cursor.

Como dije anteriormente de enviar el mensaje se encarga el servidor, la idea es que el sistema ejecute el script de Python cada 30 minutos para que no se sobrecargue el mensaje cuando tenga que enviar una gran cantidad de mensajes.

Así es cómo funciona el script de Telegram y Telegram-cli

#### 4.5. Mejoras futuras para la aplicación

Tal y como se encuentra la aplicación ahora mismo cumple su cometido perfectamente sin ningún fallo, pero se pueden añadir diversas mejoras para ampliar le experiencias haciendo uso de las herramientas que nos ofrece Telegram:

* Envío de contenido multimedia🡪 Telegram-cli permite el envío de contenido multimedia, y a pesar de que la aplicación web permite enviar mensajes podría también permitirse enviar imágenes o documentos .pdf.
* Uso de Bots🡪 Telegram-cli permite crear bots, cualquiera puede crearlos y añadir las preguntas y respuesta que quiera, sería una buena idea crear un bot para que los alumnos o tutores hicieran las preguntas que necesiten respuestas sin necesidad de preguntar a algún tutor o al centro, es una buena forma de acercarse al público y sería tan sencillo como mantenerlo al día. Además, con el bot se pueden crear encuestas que pueden ayudar a mejorar o a sacar conclusiones sobre lo que queramos.
* La aplicación puede complementarse con el uso de canales de Telegram, ya que Telegram-cli permite enviar mensaje a través de canales y grupos, esto viene bien cuando hay que enviar un mismo mensaje a la mayoría de usuarios, así en vez de utilizar el mensaje y enviárselo uno por uno a los tutores, se puede usar el canal para enviárselo a todos, así que solo habría que cambiar un par de líneas de código del script de Python.

### Conclusion y valoracion personal

La aplicación web me ha ayudado a entender mejor el tema de servidores, y sobre todo la forma de interactuar con aplicaciones ajenas a las dadas por el grado, simplemente con que la aplicación tenga versión Linux se puede hacer cualquier cosa, y con Telegram me ha ayudado a dar perspectiva y a buscar otras formas de usarlo, por ejemplo, crear un canal de difusión y programar mensajes automáticos de anuncios, o de noticias.

Desde mi punto de vista, a pesar de que la aplicación es útil y me ha ayudado a abrir la mente y dar perspectiva creo que las herramientas que ofrece Telegram de por si son suficientes, es cierto que el mensaje llegaría a todas las personas que estuvieran en el canal de difusión y mi aplicación ayuda a solucionarlo, pero bueno, es bastante útil, el desarrollo ha sido caótico, a pesar de que lo miro ahora con perspectiva y es súper sencillo de hacer, pero la forma de trabajar con dos servidores, uno por Xampp y otro con la máquina virtual de Ubuntu, de hecho el proyecto tiene casi mas código CSS que php, y eso que siempre se puede optimizar mas el código, podía haber añadido rangos de usuarios para la aplicación, y más cosilla pero no era necesario porque no dependía de mí, sino del login que tiene la UNED Melilla, aun así creo que después de terminar el proyecto y entregarlo voy a seguir agregando cosas porque es algo que me ha gustado bastante y es entretenido trastear. Además, como el año que viene voy a hacer el grado de desarrollo multiplataforma puedo coger ideas para usarlas en aplicaciones futuras.

Me han dado más problemas como hacer las consultas porque no lo veía con perspectiva y lo intentaba hacer más complicado de lo que era, pero bueno, en general ha sido una experiencia satisfactoria. Me hubiera gustado hacer el CSS con una navegación más atrevida, pero al final he ido a lo clásico.

### Bibliografia y fuentes de consulta

* He aprendido a usar el GitHub de forma rápida:

<https://www.youtube.com/watch?v=E3usf4UdkAI>

* Descarga de Telegram-cli e instalación:

<https://github.com/vysheng/tg>

* Instalación Apache MyQL PHP PHPMyadmin:

https://www.youtube.com/watch?v=juPhP1iHWSs

* Comandos de Linux:

<http://blog.desdelinux.net/mas-de-400-comandos-para-gnulinux-que-deberias-conocer/>

* Telegram y Python:

<https://geekytheory.com/tutorial-raspberry-pi-uso-de-telegram-con-python/>

* Compartir carpeta Ubuntu server y Windows:

<https://sliceoflinux.wordpress.com/2009/05/13/compartir-directorios-con-virtualbox-en-ubuntu-server/>

* Instalacion de pexpect:

<https://www.howtoinstall.co/en/ubuntu/trusty/python-pexpect>

* Instalar MYSQL Python:

<http://stackoverflow.com/questions/7459766/installing-mysql-python>

* Orientación de código Python:

<http://stackoverflow.com/questions/12325234/python-tuple-indices-must-be-integers-not-str-when-selecting-from-mysql-table>

* Orientacion de codigo Python 2:

<http://www.maestrosdelweb.com/guia-python-bases-de-datos-mysql/>

* Manejar consultas en Python:

<http://librosweb.es/libro/python/capitulo_12/conectarse_a_la_base_de_datos_y_ejecutar_consultas.html>

* Telegram, bot y Python:

https://www.youtube.com/watch?v=FeNzlT1npLg

* Programando un bot en Python:

<https://geekytheory.com/telegram-programando-un-bot-en-python/>

* Instalar mysql en Ubuntu:

<https://castudillob.wordpress.com/2007/08/15/instalar-mysql-en-ubuntu/>

* Cambiar directorio por defecto de apache:

<http://vensign.com/ubuntu-apache-2-como-cambiar-el-directorio-por-defecto/>

* Manual shell\_exec:

<http://php.net/manual/es/function.shell-exec.php>

### Fecha de terminacion del proyecto

El proyecto está en la versión 1.0, ahora mismo cumple el cometido de enviar mensajes, pero voy a añadirle diversas mejoras e intentar pasarla a clases y objetos para optimizarlo mejor, el proyecto llego a su versión 1.0 el dia 27/05/2016.