COLEGIO SALESIANO SAN JOSÉ

CFGS DESARROLLO DE

APLICACIONES MULTIPLATAFORMA

PROYECTO DE FIN DE CICLO

Emilio Vicente Medina

Mayo, 2018

0. ÍNDICE

1. INTRODUCCIÓN

1.1. INTRODUCCIÓN AL TEMA

El planteamiento consiste en el desarrollo de una plataforma en línea que permita a sus usuarios participar en comunidades basadas en sus mismos intereses, ya sean profesionales, personales, de aficiones o cualquier otra temática, al estilo de una “red social”.

Dichas comunidades son creadas y mantenidas por los mismos usuarios y conforman lugares en los que poner en común información y debatir acerca de temas de interés para los usuarios que forman parte de las mismas. Cada una de las comunidades consta en sí misma de una estructura similar a la de un foro en el sentido en que contienen hilos que, a su vez, pueden contener comentarios.

1.2. ENFOQUE

Los usuarios pueden suscribirse a las comunidades que les interesen, así como votar tanto positiva como negativamente los hilos que quieran y los comentarios de los mismos en base a si consideran que contribuyen a la comunidad o no. De esta manera se fomenta el buen comportamiento en cada comunidad gracias al criterio de los propios usuarios que la frecuentan.

El sistema de usuarios no está enfocado en la identificación personal, sino en el uso de pseudónimos y en el anonimato. No se trata de que los usuarios se comuniquen con las personas que ya conocen, sino con cualquier persona independientemente de sus características, primando las ideas, el diálogo y el compartir conocimiento e información.

La arquitectura de la plataforma sigue un modelo cliente-servidor, de tal manera que para su funcionamiento requiere de un sistema servidor que está a la espera, atiende y procesa las peticiones realizadas por los clientes, y un software cliente, que es el encargado de mostrar la información al usuario, permitir su interacción con la misma, y realizar las peticiones necesarias al servidor.

2. OBJETIVOS DEL PROYECTO

2.1. EL FENÓMENO DE LAS REDES SOCIALES

De un tiempo a esta parte, más concretamente en los últimos diez años, se han popularizado enormemente en la sociedad plataformas digitales para que las personas se mantengan en contacto digitalmente mediante Internet. Ésto no era nada nuevo, ya que los medios ya existían previamente: correo electrónico, chats IRC, foros, etc. Sin embargo, las denominadas “redes sociales” llevaron esos conceptos al público general ofreciendo un servicio unificado por medio de plataformas en las que compartir contenido con todos tus círculos personales, consiguiendo un gran éxito y dejando atrás los vetustos medios usados hasta entonces en Internet.

Por otra parte, también comenzaron a recopilar gran cantidad de información de sus usuarios, en mayor medida sin su conocimiento ni su consentimiento expreso. Dado que el servicio que ofrecen siempre ha sido gratuito, su modelo de negocio se ha basado en la publicidad y, sobre todo, en el tráfico de la información recopilada a sus usuarios.

2.2. LA IMPORTANCIA DE LA INFORMACIÓN

Mientras los gigantes tecnológicos actuales pujan por mantener su monopolio de la información, cada vez se pone más de manifiesto la necesidad de que los usuarios del mundo digital la protejamos. Es difícil cambiar algo que ya se ha establecido como el estándar en un campo: los usuarios de Facebook, WhatsApp, Twitter, Google, etc. no van a cambiar en masa a otra plataforma aunque existan alternativas que ofrezcan un mejor servicio o mayor privacidad. Por otra parte, la existencia de dichas alternativas siempre ejerce una presión, aunque sea pequeña, en beneficio de todo el conjunto de usuarios, ya que obligan a los gigantes a no acomodarse en su posición actual de mayor cuota de mercado.

Hoy día, y aún más a la luz de los recientes escándalos de Facebook sobre su tratamiento de la información que posee de usuarios, es palpable el hecho de que hacen falta más alternativas que protejan la privacidad y el anonimato.

2.3. ANONIMATO COMO CONTRAPARTE

Debido a todo aquello expuesto en los dos apartados anteriores, lo que se pretende no es otra cosa que llevar a cabo un tipo de plataforma que ofrezca esa privacidad y anonimato tan necesarias y escasas actualmente, dando todo el control a los usuarios sobre sus datos y, lo que es más importante, no usándolos para ningún otro objetivo ajeno a la prestación del servicio, ni para la identificación personal.

Por otra parte, tanto la práctica totalidad del software y herramientas usadas para la realización del proyecto, como el proyecto en sí mismo, son software libre; con lo que se pretende apoyar la filosofía que ello conlleva y demostrar que el código abierto y el software libre son algo altamente beneficioso para el progreso de la tecnología y la sociedad en su conjunto.

Ante todo, se pretende ofrecer una plataforma enfocada en la simplicidad, en la rapidez y en la accesibilidad; que fomente la conversación directa entre usuarios y que sea un lugar agradable en el que compartir información de interés común.

3. HIPÓTESIS DE PROYECTO

3.1. LIMITACIONES Y CONDICIONANTES

Dada la limitación de tiempo debido a la realización simultánea de la Formación en Centros de Trabajo, no ha sido posible implementar características tales como el uso de imágenes subidas por los usuarios como imágenes de perfil en sus cuentas ni como iconos en las comunidades; la encriptación de toda la comunicación entre el cliente y el servidor mediante TLS (Transport Layer Security, o anteriormente llamado SSL, Secure Sockets Layer); el establecer color principal y de fondo a la hora de crear una comunidad, ni la posibilidad de eliminación o edición de comunidades, hilos o comentarios.

Así mismo, funcionalidades completas planteadas en un inicio, como las salas de chat y el desarrollo de un cliente web paralelo al cliente para Android han debido quedarse fuera por la misma razón.

3.2. TECNOLOGÍAS EMPLEADAS

3.2.1. Base de datos

PostgreSQL, Sistema de Manejo de Bases de Datos de objetos relacionales de software libre y código abierto que cumple con el estándar SQL y ofrece un rendimiento, fiabilidad y rapidez excelentes.

Versión 10.3-1 para Arch Linux AMD64.

3.2.2. Servidor

Arch Linux, distribución del Sistema Operativo GNU/Linux de software libre y código abierto cuyo objetivo es la simplicidad, la eficiencia y la personalización del sistema.

Carece de versión, ya que se basa en un modelo de desarrollo continuo. Kernel Linux 4.16.10-1 y arquitectura AMD64 (ó x86-64, ó 64bits).

3.2.3. Servicio intermediario

Scripts PHP programados por mí y servidos mediante Apache, el servidor web HTTP de software libre y código abierto más usado, con la ayuda del paquete de software PHP.

Versión de PHP 7.2.5-2 y de Apache 2.4.33-3, ambos para Arch Linux AMD64.

3.2.4. Cliente

Programado en Android Studio, el entorno de desarrollo oficial de Google para el Sistema Operativo Android de licencia freeware o gratuita.

Versión 3.1.2 para GNU/Linux AMD64.

3.3. EQUIPO FÍSICO EMPLEADO

Tanto para el desarrollo de la totalidad del proyecto, como para la implementación del servidor, se ha usado mi ordenador principal de sobremesa, montado por mí con los siguientes componentes de hardware:

- Microprocesador AMD FX 8350, de 8 núcleos a 4.2GHz y arquitectura AMD64.
- Memoria RAM 12GB Kingston, tres módulos de 4GB DDR3 a 1600MHz.
- Procesador gráfico Nvidia GeForce GTX 960, con 2GB GDDR5 de vRAM.
- Almacenamiento: SSD de 120GB, SSD de 500GB y RAID5 de 3x HDD de 3TB.

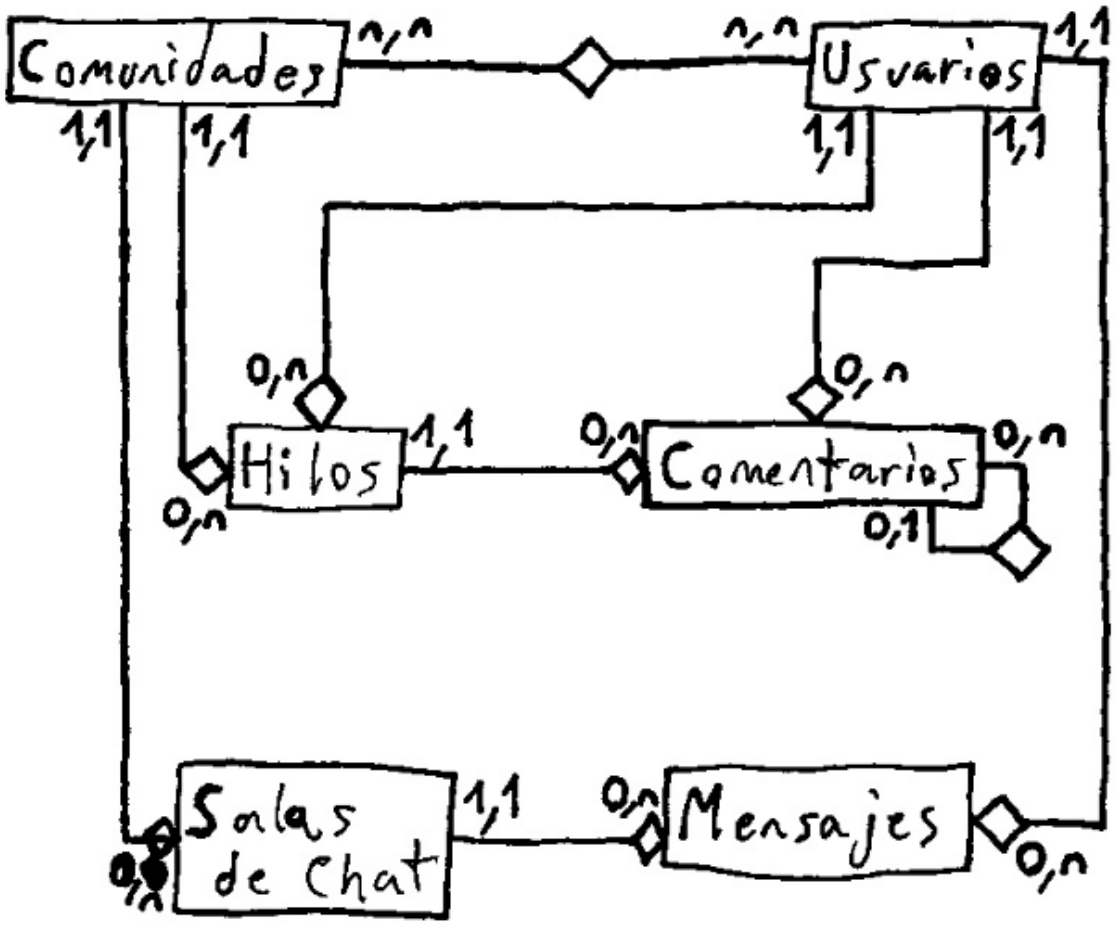
Para el control de versiones y copias de seguridad se ha usado un repositorio personal en GitHub creado para tal efecto: https://github.com/evicentemedina/esperpento

La realización de pruebas se ha llevado a cabo con mi teléfono móvil personal, modelo BQ Aquaris U con sistema Android 7.1.1.

4. DESARROLLO DEL PROYECTO

4.1. DISEÑO DE LA BASE DE DATOS

Diseño inicial del Diagrama de Entidad-Relación:



* Las salas de chat finalmente no han sido implementadas, así que las entidades de Salas de Chat y Mensajes no tienen utilidad en el resultado final del proyecto.

La creación de tablas mediante SQL para PostgreSQL, utilizando tipos de datos propios como 'serial' y sintaxis de restricciones (constrains) propias de PostgreSQL, se realizó consultando la documentación oficial de PostgreSQL [Bibliografía nº 1].

Archivo: **`./src/db/posrgresql.sql`**

4.2. PREPARACIÓN Y CONFIGURACIÓN DEL SERVIDOR

Los pasos realizados en este apartado se realizaron consultando la documentación sobre el sistema Arch Linux disponible en ArchWiki.

Instalación y configuración de PostgreSQL [Bibliografía nº 2].

Instalación y configuración de Apache HTTP Server [Bibliografía nº 3].

Instalación y configuración de PHP [Bibliografía nº 4].

Configuración de Apache para uso de PHP mediante las directivas ‘php-fpm’ y ‘mod_proxy_fcgi’ [Bibliografía nº 5].

Creación de la base de datos ‘esperpento’ y de un usuario homónimo para la realización de operaciones de selección, inserción, actualización y eliminación de registros en las tablas. Concesión de esos permisos por parte del usuario administrador de la base de datos. Éste es el usuario empleado para la conexión a la base de datos y la realización de las operaciones necesarias a petición de los clientes. No cuenta con permisos para crear, alterar ni eliminar tablas, así como sólamente cuenta con permisos de conexión a la base de datos, no puede alterarla, eliminarla, ni crear bases de datos nuevas.

Para que el usuario ‘esperpento’ pudiese insertar registros en las tablas que tienen como clave primaria un campo tipo ‘serial’ (autoincremental propio de PostgreSQL), fue necesario darle permiso específico para ello usando el siguiente comando en la terminal de administración de PostgreSQL:

```
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO esperpento
```

Por último, para que el servidor pudiese ser accesible desde fuera de la red local en la que está y recibir las peticiones HTTP, fue necesario abrir el puerto 80 en el enrutador o router y redirigirlo a la dirección IP local del ordenador que actúa como servidor.

Por comodidad y conveniencia, ya que la dirección IP externa del router es dinámica y puede cambiar, también fue registrado un nombre de dominio dinámico o DDNS con la dirección ‘esperpento.ddns.net’ que apunta a la dirección IP actual del router.

4.3. DESARROLLO DEL SERVIDOR

En un principio se inició su desarrollo en Python 3.6 usando la librería Psycopg 2.7.5, con el planteamiento de utilizar hilos y sockets para las comunicaciones entre el servidor y el cliente. Sin embargo, debido a la mayor flexibilidad y rapidez de implementación y testeo de realizar la comunicación mediante peticiones HTTP usando Apache y scripts PHP, se optó por desarrollarlo de esa manera.

El estado inicial del desarrollo y las pruebas realizadas con Python se pueden observar en el directorio **`./src/server`** y la documentación consultada fue la oficial de la librería Psycopg [Bibliografía nº 6].

El desarrollo en PHP ha sido realizado usando el soporte integrado de PHP para PostgreSQL consultando la documentación oficial de PHP al respecto [Bibliografía nº 7].

Los scripts se pueden consultar en el directorio **`./src/php`** , siendo cada uno usado para la funcionalidad de una llamada diferente desde el cliente y usando dos scripts auxiliares para la conexión con la base de datos (**`./src/php/conn.php`**) y la desconexión y codificación de la respuesta en un objeto JSON (**`./src/php/close.php`**).

El funcionamiento de los scripts está basado en la realización de las consultas SQL correspondientes en la base de datos con los datos que se reciban mediante GET, tras su debida validación y haber llevado a cabo las comprobaciones necesarias, y la devolución de un resultado en formato JSON que contenga una clave “s” (success, o éxito) de valor numérico determinado por si la operación ha tenido éxito (1) o no (0) o, si es necesario, si ha ocurrido otro tipo de error (-1). En caso de éxito, se incluye otra clave, “c” (content o contenido) de valor otro objeto o array de objetos JSON con los datos que sea necesario devolver.

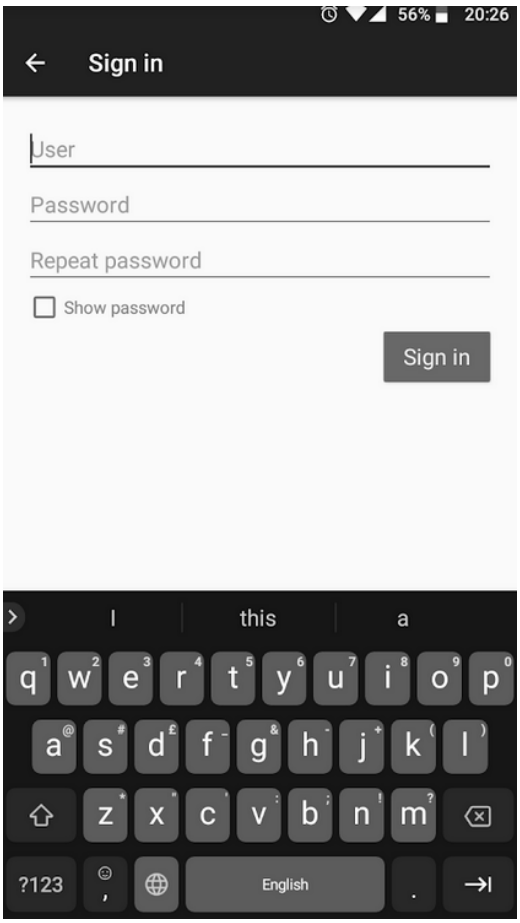
En caso de no recibir los parámetros GET imprescindibles, ni se realiza acción alguna, ni se devuelve ningún resultado para no causar carga innecesaria en el servidor.

4.4. DESARROLLO DEL CLIENTE

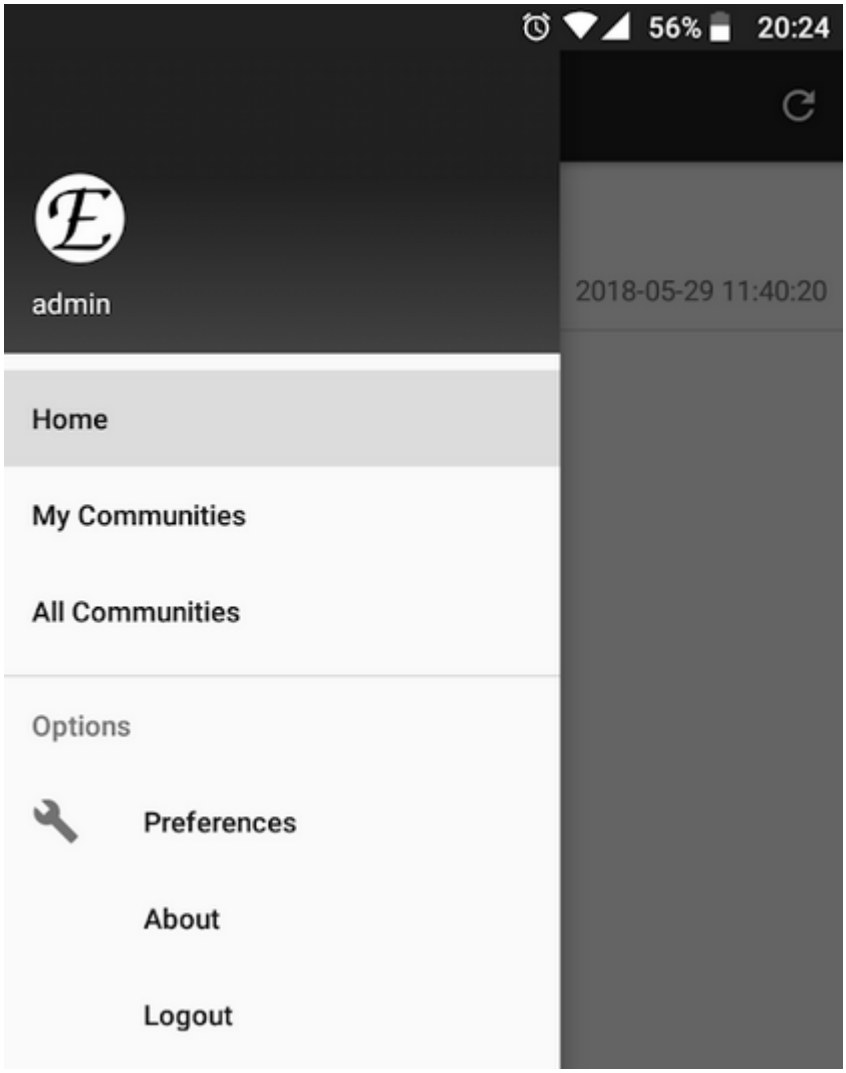
Llevado a cabo consultando la documentación oficial sobre desarrollo para Android disponible en Android Developers.

El lenguaje de programación utilizado ha sido Java para la funcionalidad y XML para la interfaz de usuario y archivos auxiliares. El proyecto en formato de Android Studio se encuentra en el directorio **`./src/android`** . La versión de API utilizada ha sido la última estable: la 27 (Android 8.1), y la más antigua soportada la 21 (Android 5.0).

La interfaz se ha desarrollado teniendo como primera pantalla una actividad de inicio de sesión con una actividad hija de creación de cuenta:



Una vez iniciada la sesión, la pantalla principal implementa un menú de navegación lateral propio de Material Design, cuyo contenido carga por defecto el fragmento ‘Home’, que muestra los hilos recientes de las comunidades a las que el usuario esté suscrito:



La comunicación con el servidor ha sido implementada mediante la librería Volley [Bibliografía nº 8], usando una cola de peticiones (RequestQueue) con la ayuda de un objeto estático construido para su uso en toda la aplicación (VolleySingleton) [Bibliografía nº 9] y añadiendo peticiones de objetos JSON a dicha cola desde el punto de la aplicación que sea necesario [Bibliografía nº 10].

De este modo, se controla eficientemente tanto la posible concurrencia de peticiones, como el que las mismas se realicen de manera asíncrona y no bloqueen la ejecución de la aplicación esperando una respuesta.

Un ejemplo de la estructura usada para las peticiones:

```
changeFragment(R.layout.fragment_loading);
VolleySingleton.getInstance().addToRequestQueue(new JsonObjectRequest(
    Request.Method.GET, Constants.getUrlMyComm(user),  jsonRequest: null,
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            try {
                if (response.getInt( name: "s") == 1) {
                    changeFragment(R.layout.fragment_all_communities,
                            response.getJSONArray( name: "c").toString());
                } else
                    changeFragment(R.layout.fragment_all_communities);
            } catch (JSONException e) {
                changeFragment(R.layout.fragment_error, getString(R.string.bad_response));
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            error.printStackTrace();
            String msg;
            if (error.networkResponse != null)
                msg = getString(R.string.error) + " " + error.networkResponse.statusCode;
            else
                msg = getString(R.string.connection_error);
            changeFragment(R.layout.fragment_error, msg);
        }
    }
));
```

Básicamente, se añade a la cola una petición de objeto JSON pasándole los parámetros de método de petición (GET), URL (con los parámetros necesarios), objeto jsonRequest nulo, escuchador de respuesta (Response.Listener, se ejecutará al obtener una respuesta) y un escuchador de error (Response.ErrorListener, se ejecutará si no se obtiene respuesta).

Para construir las URLs con los parámetros debidamente formateados (los caracteres especiales han de codificarse para ser válidos en una URL) y hacer más ordenado el manejo de tantas URLs distintas, se ha utilizado una clase ‘Constants’ con todas ellas como variables constantes y métodos estáticos que las devuelven construidas con los parámetros que se les pase.

`./src/android/app/src/main/java/evicentemedina/esperpento/objects/Constants.java`

La clase encargada de manejar la cola, VolleySingleton, es realmente sencilla: obtiene el contexto general de la aplicación con la ayuda de la clase Esperpento (clase vacía usada simplemente para mantener el contexto general de la aplicación) para mantener la cola durante toda la vida útil de la aplicación y tiene dos métodos, uno para añadir peticiones a la cola, y otro para obtener el estado actual de la misma.

```
public class VolleySingleton {
    private static VolleySingleton instance = null;
    private RequestQueue requestQueue;

    private VolleySingleton() {
        requestQueue = getRequestQueue();
        requestQueue = Volley.newRequestQueue(Esperpento.getAppContext());
    }

    public static synchronized VolleySingleton getInstance() {
        if(instance == null) {
            instance = new VolleySingleton();
        }
        return instance;
    }

    public RequestQueue getRequestQueue() {
        return requestQueue;
    }

    public <T> void addToRequestQueue(Request<T> req) {
        req.setTag("cancelable");
        getRequestQueue().add(req);
    }
}
```

`./src/android/app/src/main/java/evicentemedina/esperpento/objects/VolleySingleton.java`

La imagen de icono de la aplicación ha sido diseñada usando Gimp, un editor de imágenes de software libre, e importada en Android Studio mediante la herramienta integrada para su adaptación a los diferentes tamaños y formatos necesarios para su uso como icono.

Archivos: **`./img/icon.png`** y **`./img/icon.xcf`**

El resto de iconos usados en la aplicación han sido obtenidos en formato de vectores SVG de la página web de Material Design [Bibliografía nº 11] y también han sido importados en Android Studio mediante la herramienta para su adaptación como recursos dibujables o 'drawables' del proyecto. Se pueden encontrar en el directorio **`./img`** .

5. CONCLUSIONES Y PROPUESTAS

Estoy bastante satisfecho con el resultado obtenido, ya que todo aquello que está implementado funciona correctamente y con una eficiencia considerable, algo que valoro con creces y opino que tiene mayor importancia de la que usualmente le es dada.

Así mismo, la comunicación entre el cliente y el servidor se ha implementado con una alta eficiencia en cuanto al volumen de información para minimizar lo máximo posible el tráfico de red, permitiendo unos tiempos de carga mínimos y una respuesta casi instantánea, a pesar del poco ancho de banda con el que cuenta la conexión ADSL que usa el servidor (tan sólo 0.7 Mbps de subida, lo que equivale a 0.0875 MB/s u 89.6 KB/s).

Lógicamente, la plataforma en su estado actual dista de ofrecer un servicio completo que pueda ser puesto a disposición del público general, pero sirve como prueba de concepto de una alternativa diferente a las plataformas comunicativas digitales actuales.

Por otra parte, en mi tiempo libre seguiré trabajando en esta idea e implementando las características que tenía en mente desde un principio. Además de que, siendo un proyecto de software libre, cualquiera es libre de consultar el código fuente y aprender y coger ideas del mismo.

6. BIBLIOGRAFÍA

Android Developers: https://developer.android.com/

ArchWiki: https://wiki.archlinux.org/

Init.d.org: http://initd.org/psycopg/

Material Design: https://material.io/

PHP.net: http://php.net/

PostgreSQL.org: https://www.postgresql.org/

Stack Overflow: https://stackoverflow.com/