

Evolución y Gestión de la Configuración

Agor@Us

Autenticación

Grupo 1 – ID de Opera 7
Curso 2016/2017

Fidel Mazo Delgado
Alejandro Garrido Resina
Alejandro Guerrero Montoro
Alejandro Román Rodríguez
Julio Márquez Castro
María Remedios Dans Caballero

Control del Documento

Registro de Cambios en el Documento

Versión	Motivo	Realizado Por	Fecha
1.0	Creación	Alejandro Román	20/12/16
2.0	Mejora 1	Alejandro Guerreo	26/01/17
3.0	Mejora 2	Alejandro Guerreo	31/01/17

Índice

I. Índce de figuras.	4
1. Resumen.	5
2. Introducción y Contexto.	6
3. Descripción del sistema.	7
3.1. Interfaz y configuración.	7
3.2. Login sin DNle.	8
3.3. Login con Twitter.	9
3.4. Login con Facebook.	12
3.5 Login con Google +.	15
3.6. API Rest.	22
3.7. Modificaciones del proyecto.	23
3.8. Planificación del proyecto.	23
4. Elementos de Control.	25
5. Entorno de desarrollo.	25
6. Gestión del código fuente.	26
7. Gestión de la integración continua.	28
8. Gestión de incidencias.	30
9. Gestión de liberaciones y despliegue.	31
10. Mapa de herramientas.	32
11. Conclusiones y trabajos futuros.	34

Índice de Figuras

- Figura 1. Logotipo de AgoraVoting modificado para AgoraUs.
- Figura 2. Imagen del Conceptual Model.
- Figura 3. Página Principal del subproyecto de Autenticación.
- Figura 4. Pantalla de Login.
- Figura 5. Pantalla de Control de Acceso mediante segunda verificación.
- Figura 6. Pantalla de creación de nueva app de twitter.
- Figura 7. Pantalla de creación de nueva app de twitter de manera local.
- Figura 8. Pantalla de creación de nueva app de twitter para integración.
- Figura 9. Pantalla de acceso a AgoraUS mediante app de twitter.
- Figura 10. Pantalla de acceso a AgoraUS mediante app de twitter.
- Figura 11. Fichero index.php del proyecto.
- Figura 12. Fichero loginFacebook.php del proyecto.
- Figura 13. Fichero fc-callback.php del proyecto.
- Figura 14. Creación de nuevo proyecto.
- Figura 15. Nombre del nuevo proyecto.
- Figura 16. Google.
- Figura 17. Credenciales para Google.
- Figura 18. Credenciales para Google.
- Figura 19. Pantalla de autorización.
- Figura 20. Pantalla de autorización.
- Figura 21. Credenciales.
- Figura 22. Credenciales. URIs.
- Figura 23. Cliente de OAuth.
- Figura 24. Estructura de comunicaciones.
- Figura 25. Repositorio de AgoraUS.
- Figura 26. Visualización gráfica de las ramas implementadas en el repositorio.
- Figura 27. Travis. Configuración.
- Figura 28. Travis. Salida por pantalla.
- Figura 29. Travis. Successfull.
- Figura 30. Jenkins. Estado actual.
- Figura 31. ObjectMapped.
- Figura 32. Apartado de Issuess de GitHub.
- Figura 33. Formulario para la creación de una nueva Issue.
- Figura 34. Mapa de herramientas.

1. Resumen

AgoraUs es una aplicación de gestión de elecciones y votaciones electrónica, basada en Agora Voting, una aplicación que está en funcionamiento por Europa en la actualidad. AgoraUs está estructurada en diferentes secciones que abarca:

- **Autenticación.**
- **Verificación.**
- **Accesibilidad.**
- **Cabinas.**
- **Sistema de voto.**
- **Test de IG.**
- **Visualización.**
- **Avanzar IG.**
- **Recuento.**
- **Censos.**
- **Cabina Telegram.**
- **Almacenamiento.**



Figura 1: Logotipo de AgoraVoting modificado para AgoraUS.

A lo largo del presente documento se verán las diferentes modificaciones y aportaciones al proyecto, más concretamente del subsistema de **Autenticación**, el cual es el responsable del registro y del control de acceso. Se explicarán con detalle cómo se ha reemplazado el antiguo método de acceso o cómo se ha habilitado el acceso a la plataforma desde diferentes logins desde diversas redes sociales.

A continuación mostramos los enlaces más importantes:

- [En el enlace que a continuación ofrecemos están toda la documentación y código del proyecto de Autenticación 16/17.](#)

- [Enlace a la página de incidencias del subproyecto.](#)
- [Enlace a la Wiki de los miembros del proyecto.](#)
- [Sistema beta desplegado.](#)
- [Sistema estable desplegado.](#)

2. Introducción y Contexto.

En la actualidad existe la necesidad de un sistema de voto electrónico seguro que nos proporcione la confianza que ello requiere. En esta asignatura se pretende tomar un proyecto existente con el fin de agilizar la capacidad de trabajar en grupo con código heredado.

Nuestra labor consiste en estudiar y mejorar el subproyecto de **Autenticación** para el proyecto de votación electrónica asignada por esta asignatura, AGORA@US.

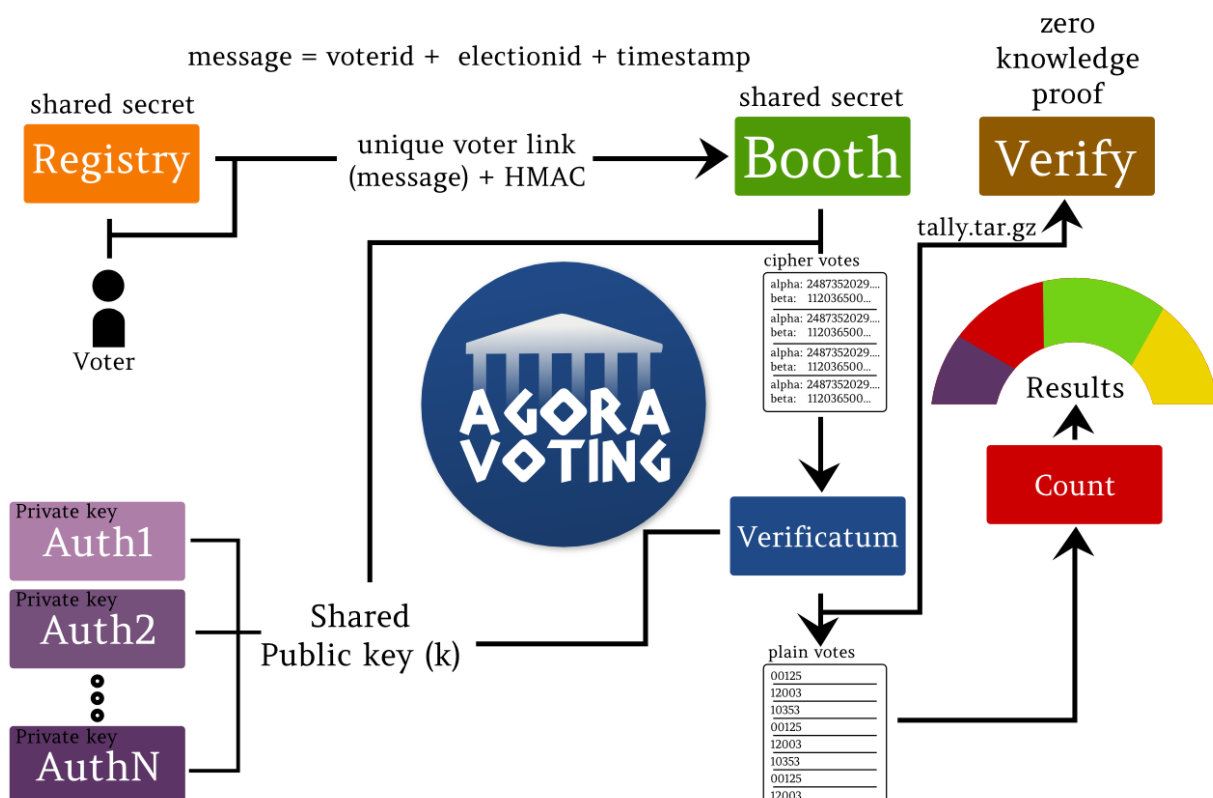


Figura 2: Imagen tomada de <https://1984.lsi.us.es/wiki-egc/images/8/8c/AgoraConceptualMap.png>

Se han realizado ampliaciones de la funcionalidad para el control de acceso mediante **Facebook** y **Twitter**. También se ha reemplazado el anterior sistema de control de acceso por un de **dobles login**, el cual nos proporciona mayor seguridad a la hora de frenar a ciberdelincuentes.

3. Descripción del sistema.

La autenticación es el proceso mediante el cual una persona accede a una aplicación informática. Este proceso se puede llevar a cabo de distintas maneras, las implementadas en esta aplicación son:

- **Interfaz y configuración.**
- **Login sin DNIE.**
- **Login con Twitter.**
- **Login con Facebook.**
- **Login con Google +.**
- **API Rest**

3.1. Interfaz y configuración.

Para cubrir distintos niveles de seguridad, la aplicación tiene distintos modos de login implementados estos son:

Acceso desde cualquier opción: se podrá acceder a la aplicación desde todas las formas de login es decir con DNIE, desde redes sociales, login sin DNIE.

Acceso solo desde redes sociales: se podrá acceder a la aplicación desde las redes sociales implementadas y sin DNIE.

Acceso con DNIE: Se podrá acceder a la aplicación con DNIE y sin él.

Acceso sin Dnie: esta opción solo permite acceder con el login sin DNIE.

Para poder facilitar la labor de configuración de los distintos tipos de conexiones se han creado dos variables globales para estas son:

\$socialNet: variable que configura si se desea o no poder loguearse desde las redes sociales.

\$DNIE: variable que configura la opción de acceso con el DNIE.

Estas variables se han definido en el archivo variables.php, el cual es un archivo de configuración de la aplicación y son usadas en index.php. Indicar que ambas son variables booleanas las cuales toman valor true o false dependiendo de la configuración que deseemos usar.

3.2. Login sin DNle.

Una de las tantas funcionalidades añadidas a este proyecto es la posibilidad de acceder al sistema sin DNle, sencillamente con las credenciales que tenemos en la base de datos.

Para ello se ha realizado un archivo nuevo, simple y legible en el cual se realizan comprobaciones en otros ficheros y a los cuales se ha unido la funcionalidad de doble login, una característica que está al orden del día fortificando más la entrada a personal no autorizado y ciberdelincuentes.

En primer lugar, se ha añadido un botón en la página principal con el que poder acceder al login sin DNle. En la siguiente imagen podemos ver el cambio mencionado:

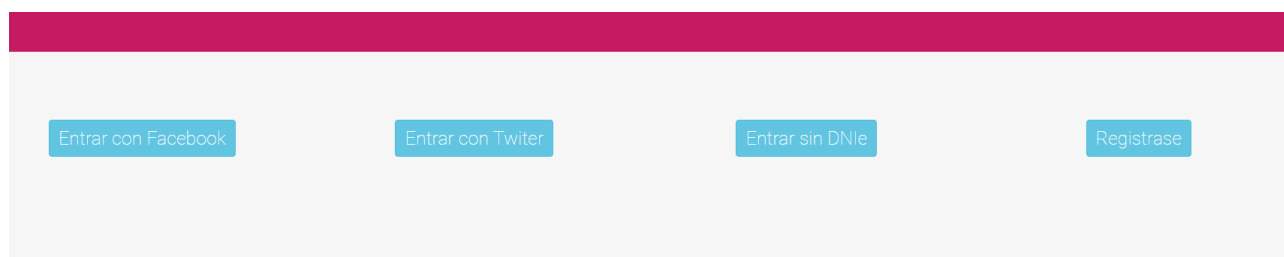


Figura 3: Página principal del subproyecto Autenticación.

Una vez que accedemos al login desde la página principal nos encontramos con una página que nos pide acceso mediante nombre de usuario y contraseña sin la posibilidad de dejar sesión abierta por motivos de seguridad. A continuación, podemos apreciar la vista descrita:

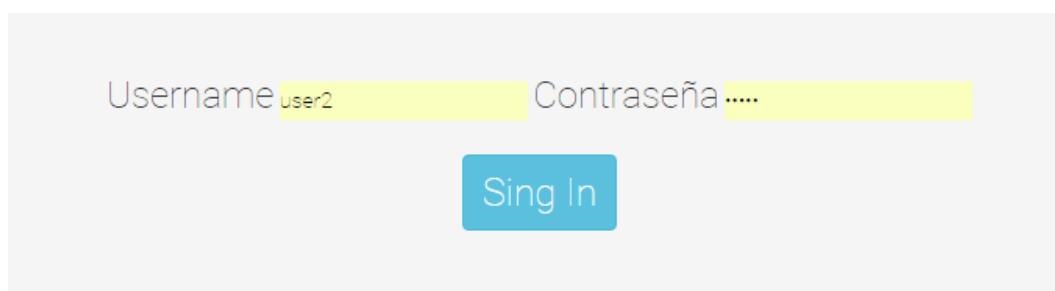


Figura 4: Pantalla de login.

Después presionar el botón para loguearse, si los datos introducidos son correctos, accederemos a la zona para introducir el código de verificación que, supuestamente, deberemos recibir por correo o por SMS (mejor éste último para mayor seguridad). En caso de no haber introducido bien los datos nos redirigirá la página principal. En la siguiente imagen podemos ver una ilustración de lo explicado:



Figura 5: Pantalla de control de acceso mediante la segunda verificación.

Llegados a este punto, si el código introducido coincide con el código recibido podremos acceder a nuestra zona habilitada para el voto. Podemos ver, en la imagen que se muestra, cómo se ha accedido de manera satisfactoria. De no haber introducido un código de verificación correcto se nos habría redirigido a la página principal.

3.3. Login con Twitter.

Otro sistema de control de acceso implementado en el proyecto es el login mediante Twitter. Este proceso no es más que acceder al sistema de votación electrónica mediante las credenciales que, previamente, se han creado en la red social de Twitter.

Para ello, y en primer lugar, se accedió a la URL '<http://apps.twitter.com>', la cual nos lleva a un sitio web que, si estamos logueados, nos mostrará la siguiente pantalla:

Twitter Apps

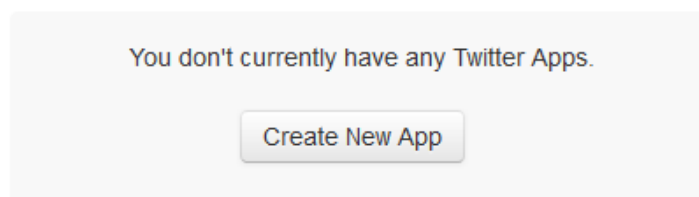
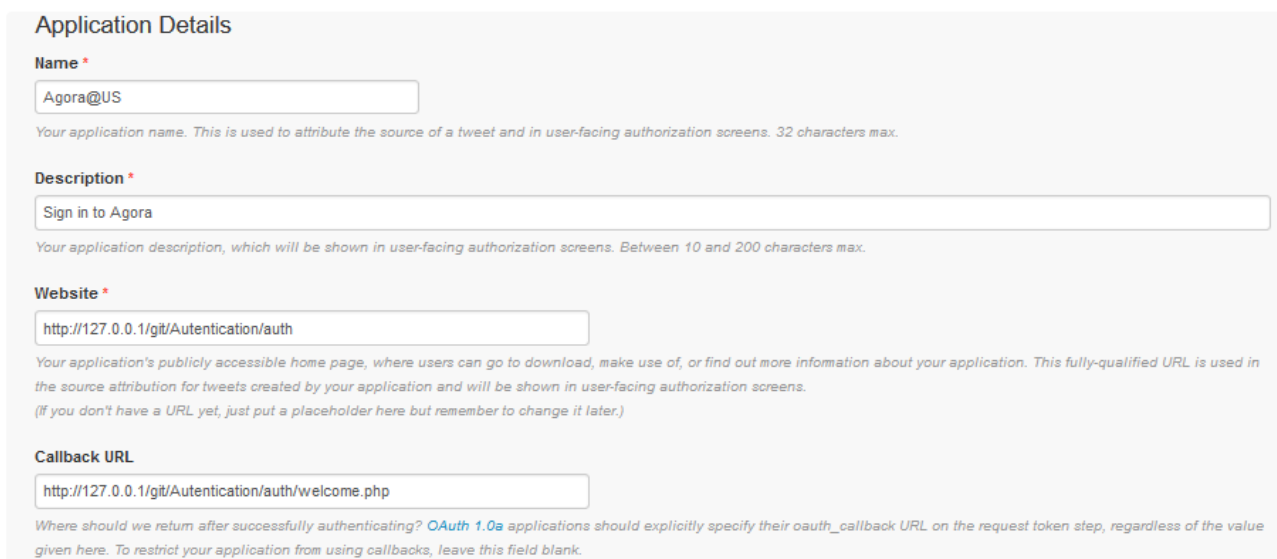


Figura 6: Pantalla de creación de Nueva App de Twitter.

Una vez que procedemos a la creación de la App, procederemos al formulario con el que iniciar dicho proceso. A continuación vemos una figuras en las que vemos la creación de la app para Twitter de manera local y otra para la integración.



The screenshot shows the 'Application Details' form for creating a new Twitter app. It includes four main sections: Name, Description, Website, and Callback URL. Each section has a text input field and a descriptive note below it.

Application Details

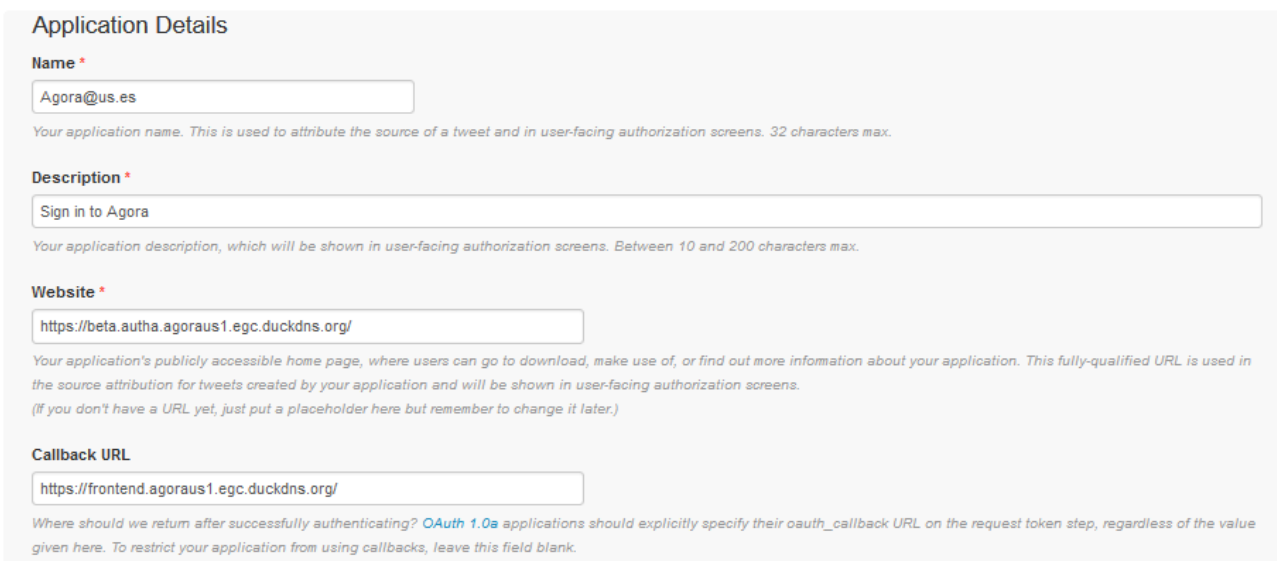
Name *
Agora@US
Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *
Sign in to Agora
Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *
http://127.0.0.1/git/Autentication/auth
Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL
http://127.0.0.1/git/Autentication/auth/welcome.php
Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Figura 7: Pantalla de creación de Nueva App de Twitter de manera local.



The screenshot shows the 'Application Details' form for creating a new Twitter app, similar to the previous one but with different values for the Website and Callback URL fields.

Application Details

Name *
Agora@us.es
Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *
Sign in to Agora
Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *
https://beta.autha.agoraus1.egc.duckdns.org/
Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL
https://frontend.agoraus1.egc.duckdns.org/
Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Figura 8: Pantalla de creación de Nueva App de Twitter para la integración.

Una vez hecho esto la api de twitter nos proporcionará una serie de tokens que necesitaremos posteriormente para sincronizar el sistema de información web con el servicio de la api de Twitter. Seguidamente tendremos que implementar, por medio de una serie de archivos, la autenticación de Twitter. Para acceder nos dirigiremos a la página principal, 'index.php', en la que podremos ver el botón de acceso, el cual nos redirigirá al archivo 'process.php' con el que empezaremos a acceder mediante la app creada.

Además de los ficheros mencionados, tendremos que configurar los siguientes:

- **config.php:** En este archivo tendremos la configuración de los tokens de twitter (Consumer Key y Consumer Secret), y la Callback url, que será donde nos redirigirá la página una vez que nos hayamos autenticado.
- **twitteroauth.php:** En este archivo primero tendremos que cargar la librería de OAuth, la cual se puede encontrar en <http://oauth.net>, posteriormente estará la clase TwitterOAuth donde tendremos una serie de funciones para configurar la autenticación gracias a los servicios que ofrece la api de Twitter.
- **functions.php:** Este archivo tiene la clase Users con algunas funciones. En la función `__construct()` hemos configurado la base de datos y la conexión con la base de datos. Sólo necesita cambiar el valor de las variables `$dbServer`, `$dbUsername`, `$dbPassword` y `$dbName` con los detalles de su servidor MySQL.

Posteriormente veremos si la variable global `'oauth_token'` está definida y no es null, también veremos si el token de sesión es distinto de la variable `'oauth_token'`. Si esto se cumple haremos una nueva clase `'TwitterOAuth'` donde le incluiremos las variables necesarias de configuración.

Una vez hecha la conexión sacaremos el acceso al token y veremos si el código de la conexión es `'200OK'`, si es correcto redirigiremos el usuario a Twitter, insertaremos el usuario en la base de datos (chequeando su información) y volveremos al index. Si no es correcta la conexión daremos un mensaje de error.

Por último, si la variable `'oauth_token'` no está definida, veremos si la variable `'global denied'` está definida. Si está definida iremos al index y si no, haremos una conexión nueva con los elementos necesarios e incluiremos los nuevos tokens de `'oauth'`. Si la conexión se establece correctamente redigiremos al usuario a la URL de Twitter. Si falla mostraremos un mensaje de error.

En la siguiente figura vemos que al acceder, desde index.php, haciendo click en el botón de Twitter, nos dirigimos a una página web perteneciente a Twitter la cual nos pide las credenciales necesarias para acceder.

¿Autorizas a Agora@us.es para utilizar tu cuenta?



Agora@us.es

beta.autha.agoraus1.egc.duckdns.org/

Sign in to Agora

☐ Recordar mis datos · ¿Olvidaste tu contraseña?

Iniciar sesión

Cancelar

Esta aplicación podrá:

- Leer Tweets de tu cronología.
- Ver a quién sigues.

No podrá:

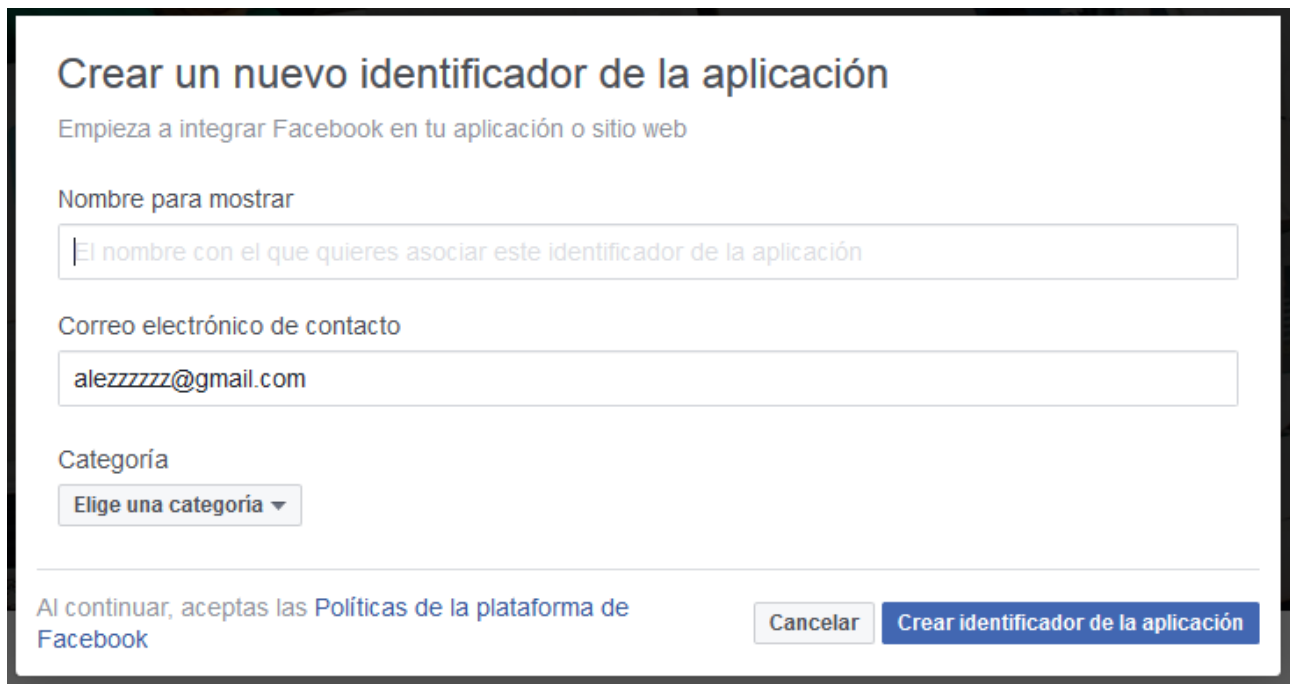
- Sigue a nuevas personas.
- Actualizar tu perfil.
- Publicar Tweets por ti.
- Acceder a tus mensajes directos.
- Mira tu dirección de correo electrónico.
- Ver tu contraseña de Twitter.

Figura 9: Pantalla de acceso a Agor@US mediante app de Twitter.

3.4. Login con Facebook.

Por último, se implementó un control de acceso mediante la red social de Facebook. Este proceso no es más que acceder al sistema de votación electrónica mediante las credenciales que, previamente, se han creado en la red social de Facebook.

Para ello, y en primer lugar, se accedió a la URL 'https://developers.facebook.com', la cual nos lleva a un site web que, si estamos logueados, nos mostrará la siguiente pantalla:



Crear un nuevo identificador de la aplicación

Empieza a integrar Facebook en tu aplicación o sitio web

Nombre para mostrar

El nombre con el que quieres asociar este identificador de la aplicación

Correo electrónico de contacto

alezzzzzz@gmail.com

Categoría

Elige una categoría ▼

Al continuar, aceptas las [Políticas de la plataforma de Facebook](#)

Cancelar Crear identificador de la aplicación

Figura 10: Pantalla de acceso a Agor@US mediante app de Twitter.

Introducimos el nombre de nuestra aplicación, en nuestro caso introduciremos 'Autenticación y Autorización'.

Tendremos que instalar el SDK de Facebook para PHP siempre y cuando tengamos en cuenta unos requisitos del sistema básicos:

- Tener instalados **PHP 5.4** o superior. Esto lo sabremos ejecutando el método 'phpversion()'. Bastará con escribir: 'echo phpversion();'
- Tener la extensión '**mbstring**'.
- Tener instalado '**Composer**', aunque este requisito es opcional.

Tenemos la posibilidad de instalar el **SDK de Facebook**:

- **Mediante 'Composer'**, que es la opción que se ha utilizado en este proyecto.
- De forma **manual**, método menos automatizado pero con más control sobre lo que se está haciendo.

Volvemos al punto en el que creamos la app. Tendremos que dirigirnos al proyecto que tenemos abierto en Aptana y abriremos los siguientes archivos para su modificación:

- **index.php**: Aquí no haremos más que introducir un botón que nos enlace a 'loginFacebook.php'.

```
<div class="col-md-3">
    <input onClick="location.href = 'loginFacebook.php' "
            id="loginFacebook"
            type="button"
            value = "Entrar con Facebook"
            class="btn btn-info"/>
</div>
```

Figura 11: Fichero 'index.php' del proyecto.

- **loginFacebook**: En este fichero lo que haremos es redirigir a 'fb-callback.php' donde obtendremos el identificador de acceso con las credenciales necesarias para el acceso mediante las credenciales que se tienen en la red social de Facebook.

```
#!/usr/bin/php

/* Iniciando la sesión*/
session_start();

/* Cambiar según la ubicación de tu directorio*/
require_once __DIR__ . '/vendor/facebook/graph-sdk/src/Facebook/autoload.php';

$fb = new Facebook\Facebook(array(
    'app_id' => '1065487940245693',
    'app_secret' => '1a934054901ddcecf5095a53172e0d07',
    'default_graph_version' => 'v2.4',
));

$helper = $fb->getRedirectLoginHelper();

$permissions = array('email'); // Permisos opcionales
$loginUrl = $helper->getLoginUrl('http://egcaj.tk/fb-callback.php', $permissions);

/* Link a la página de login*/
echo '<a href="' . htmlspecialchars($loginUrl) . '">Login con Facebook!</a>';

?>
```

Figura 12: Fichero 'loginFacebook.php' del proyecto.

- **fb-callback**: Cargamos la configuración de la aplicación e iniciamos sesión mediante token.

```

fb-callback.php x
46 // Get the access token metadata from /debug_token
47 $tokenMetadata = $oAuth2Client->debugToken($accessToken);
48 //echo '<h3>Metadata</h3>';
49 //var_dump($tokenMetadata);
50
51 // Validation (these will throw FacebookSDKException's when they fail)
52 $tokenMetadata->validateAppId('1065487940245693'); // Replace {app-id} with your app id
53 // If you know the user ID this access token belongs to, you can validate it here
54 // $tokenMetadata->validateUserId('123');
55 $tokenMetadata->validateExpiration();
56
57 if (! $accessToken->isLongLived()) {
58     // Exchanges a short-lived access token for a long-lived one
59     try {
60         $accessToken = $oAuth2Client->getLongLivedAccessToken($accessToken);
61     } catch (Facebook\Exceptions\FacebookSDKException $e) {
62         echo "<p>Error getting long-lived access token: " . $helper->getMessage() . "</p>\n\n";
63         exit;
64     }
65
66     echo '<h3>Long-lived</h3>';
67     var_dump($accessToken->getValue());
68 }
69
70 $_SESSION['fb_access_token'] = (string) $accessToken;
71
72 // User is logged in with a long-lived access token.
73 // You can redirect them to a members-only page.
74 //header('Location: https://example.com/members.php');
75 header('Location: https://frontend.agoraus1.egc.duckdns.org');

```

Figura 13: Fichero 'fb-callback.php' del proyecto.

Llegados a este punto podemos acceder a nuestra aplicación web de voto electrónico mediante las credenciales que posee Facebook de nosotros.

3.5 Login con Google +

El login de los productos de Google se realiza a través de *OAuth2*, esto es un mecanismo propio de Google para las aplicaciones a través de sus sistemas de login.

Para poder usar *OAuth2* es necesario dar de alta la aplicación a través de la consola de desarrolladores de Google <https://console.developers.google.com/iam-admin/projects>. En esta página debes de seguir los siguientes pasos:

Crear nuevo proyecto: para ello pulsamos el botón Crear proyecto que hay en la página, rellenos el formulario que aparece y pulsamos crear.

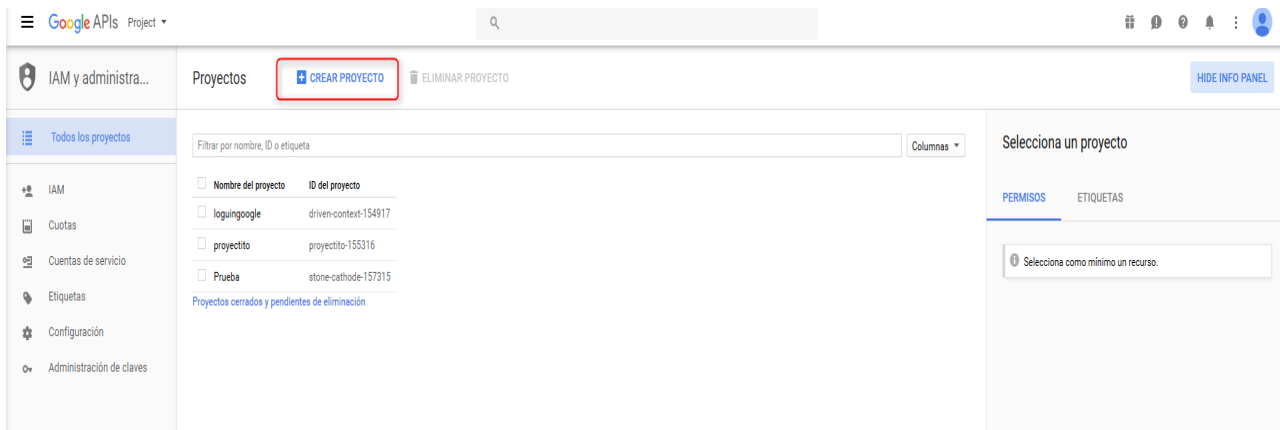


Figura 14: Creación de nuevo proyecto.

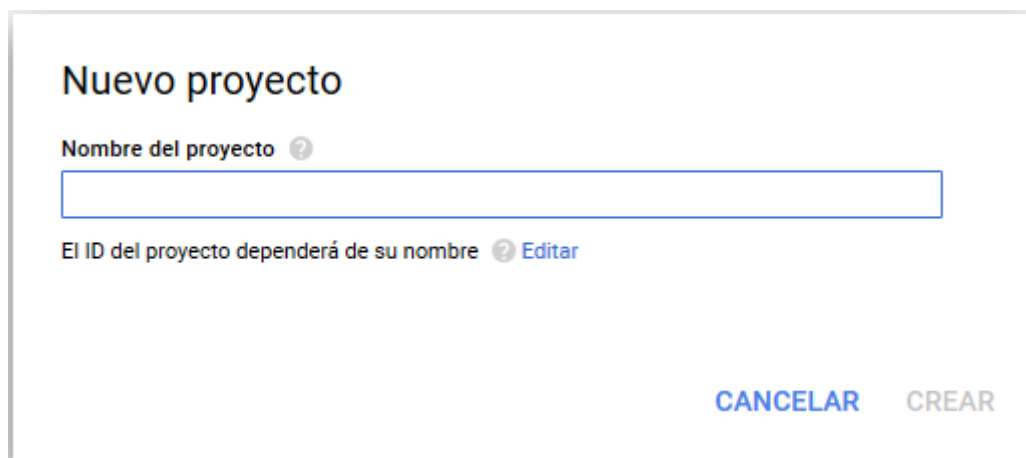


Figura 15: Nombre del nuevo proyecto

Seleccionamos las apis que necesitemos añadir.

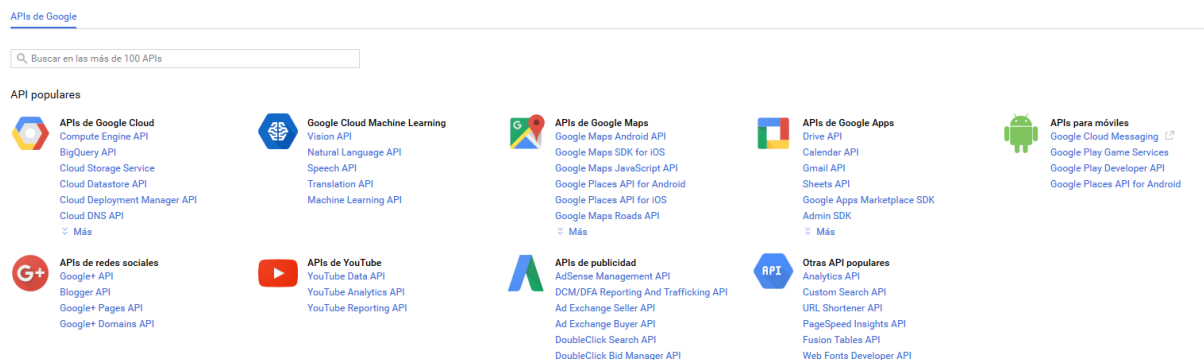


Figura 16: Google.

Posteriormente seleccionamos la opción *credenciales* de del menú del lado izquierdo.

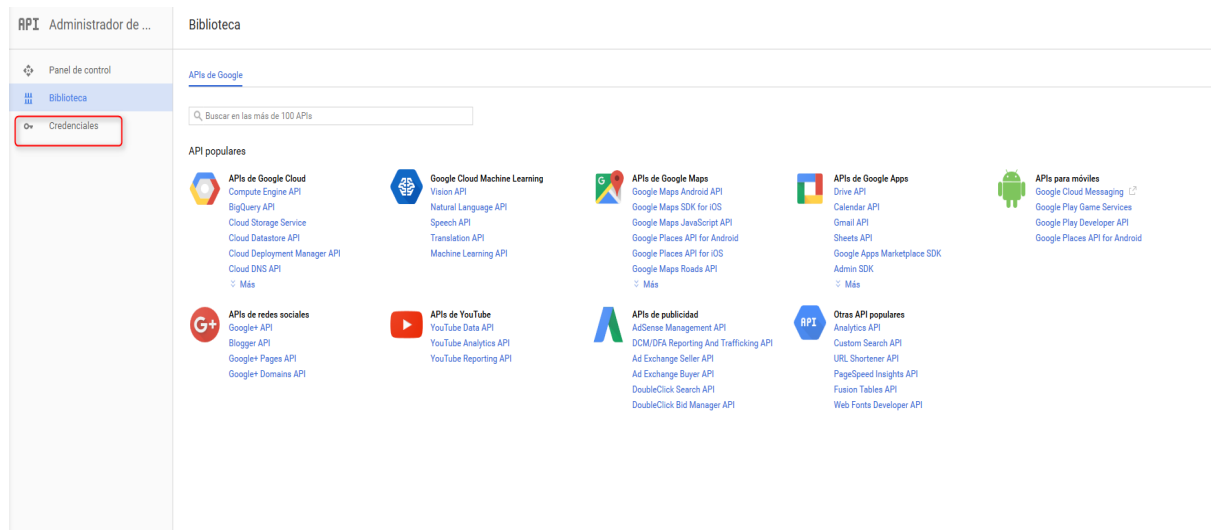


Figura 17: Credenciales para Google.

En esta pantalla seleccionamos *Pantalla de autorización de OAuth* y en el formulario que aparece introducimos el campo *Nombre de producto mostrado a los usuarios*.

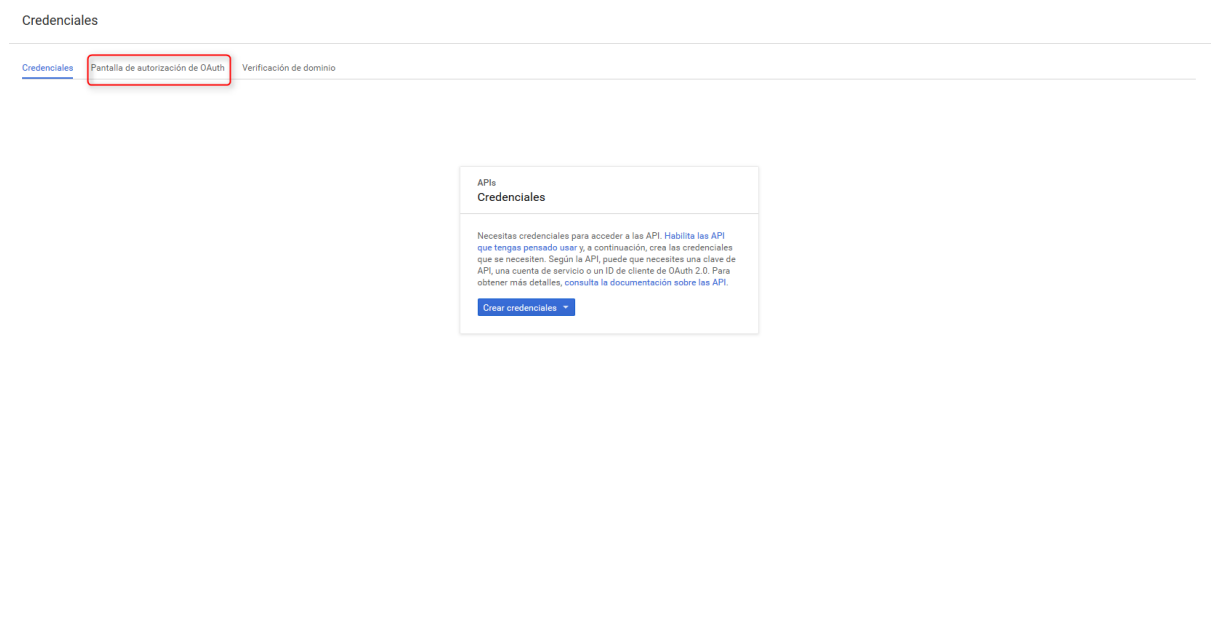


Figura 18: Credenciales Google.


Credenciales **Pantalla de autorización de OAuth** Verificación de dominio

Dirección de correo electrónico ⓘ
ozam85@gmail.com

Nombre de producto mostrado a los usuarios
Nombre del producto

URL de página principal (Opcional)
https:// o http://

URL de logotipo de producto (Opcional) ⓘ
http://www.example.com/logo.png


 Así es como verán los usuarios finales tu logotipo
Tamaño máximo: 120 x 120 píxeles

URL de la Política de Privacidad
Es opcional hasta que despliegues tu aplicación
https:// o http://

URL de las Condiciones de Servicio (Opcional)
https:// o http://

Guardar

Cancelar



La pantalla de autorización se muestra a los usuarios cuando solicitas acceso a sus datos privados mediante tu ID de cliente. Se muestra para todas las aplicaciones registradas en este proyecto.

Debes proporcionar una dirección de correo electrónico y un nombre de producto para que OAuth funcione.

Figura 19: Pantalla de autorización.

Tras esto vamos a generar las credenciales de nuestro proyecto, para ello pulsamos en *credenciales* y en el botón que aparece seleccionamos de la lista desplegable *ID de cliente de OAuth*.

Credenciales

Pantalla de autorización de OAuth


Verificación de dominio

Dirección de correo electrónico ?
ozam85@gmail.com

Nombre de producto mostrado a los usuarios
Nombre del producto

URL de página principal (Opcional)
https:// o http://

URL de logotipo de producto (Opcional) ?
http://www.example.com/logo.png




Así es como verán los usuarios finales tu logotipo
Tamaño máximo: 120 x 120 píxeles

URL de la Política de Privacidad
Es opcional hasta que despliegues tu aplicación
https:// o http://

URL de las Condiciones de Servicio (Opcional)
https:// o http://

Guardar

Cancelar



La pantalla de autorización se muestra a los usuarios cuando solicitas acceso a sus datos privados mediante tu ID de cliente. Se muestra para todas las aplicaciones registradas en este proyecto.

Debes proporcionar una dirección de correo electrónico y un nombre de producto para que OAuth funcione.

Figura 20: Pantalla de autorización.

APIs

Credenciales

Necesitas credenciales para acceder a las API. [Habilita las API que tengas pensado usar](#) y, a continuación, crea las credenciales que se necesiten. Según la API, puede que necesites una clave de API, una cuenta de servicio o un ID de cliente de OAuth 2.0. Para obtener más detalles, [consulta la documentación sobre las API](#).

Crear credenciales

Clave de API

Identifies your project using a simple API key to check quota and access

ID de cliente de OAuth

Requests user consent so your app can access the user's data

Clave de cuenta de servicio

Enables server-to-server, app-level authentication using robot accounts

Ayúdame a elegir

Te haremos unas preguntas para decidir qué tipo de credencial puedes usar

Figura 21: Credenciales.

Seleccionamos el tipo de aplicación de la lista que aparece en nuestro caso *web*.

Insertamos el nombre del cliente web y lo que nos haga falta en las Restricciones ya sean los *Orígenes de JavaScript autorizados* o *URLs de redireccionamiento autorizados*, en esta última opción meteremos todas las direcciones de páginas que queramos para que nuestra aplicación redirija al login y al logout. Si la dirección no está activa no se podrá usar redireccionar.

Credenciales



Crear ID de cliente

Tipo de aplicación

- ☒ Web
- ☐ Android [Más información](#)
- ☐ Chrome [Más información](#)
- ☐ iOS [Más información](#)
- ☐ PlayStation 4
- ☐ Otro

Nombre

Cliente web 1

Restricciones

Introduce los orígenes de JavaScript, los URI de redireccionamiento o ambos

Orígenes de JavaScript autorizados

Para su uso en las solicitudes de navegador. Se trata del URI de origen de la aplicación cliente. No puede contener caracteres comodín (`http://*.example.com`) ni una ruta (`http://example.com/subdir`). Si utilizas un puerto no estándar, deberás incluirlo en el URI de origen.

`http://www.example.com`

URLs de redireccionamiento autorizados

Para usarse con las solicitudes de un servidor web. Es la ruta de la aplicación a la que se redirecciona a los usuarios después de autenticarse en Google. A dicha ruta se añadirá el código de autorización de acceso. Debe tener un protocolo. No puede incluir fragmentos de URL ni rutas relativas. No puede ser una dirección IP pública.

`http://www.example.com/oauth2callback`

Crear

Cancelar

Figura 22: Credenciales. Uris.

Tras completar esto pulsamos en el botón crear lo que nos generará el *ID de cliente* y el *cliente secreto* con esto finalizamos el proceso de activación.

Cliente de OAuth

Este es tu ID de cliente

380954900648-fjmb8q2pg4kmpctdqhtjdveofrb389.apps.googleusercontent.com

Este es tu secreto de cliente

r2Ie1-SiEEs7f-W7iCwwi-1y

ACEPTAR

Figura 23: Cliente de OAuth.

Los datos de la redirección y los clientes interno se introducen en los campos respectivos de OAuth2, el proceso habrá terminado.

Seguidamente se muestra un esquema de las comunicaciones entre cliente los servidores y OAuth2.

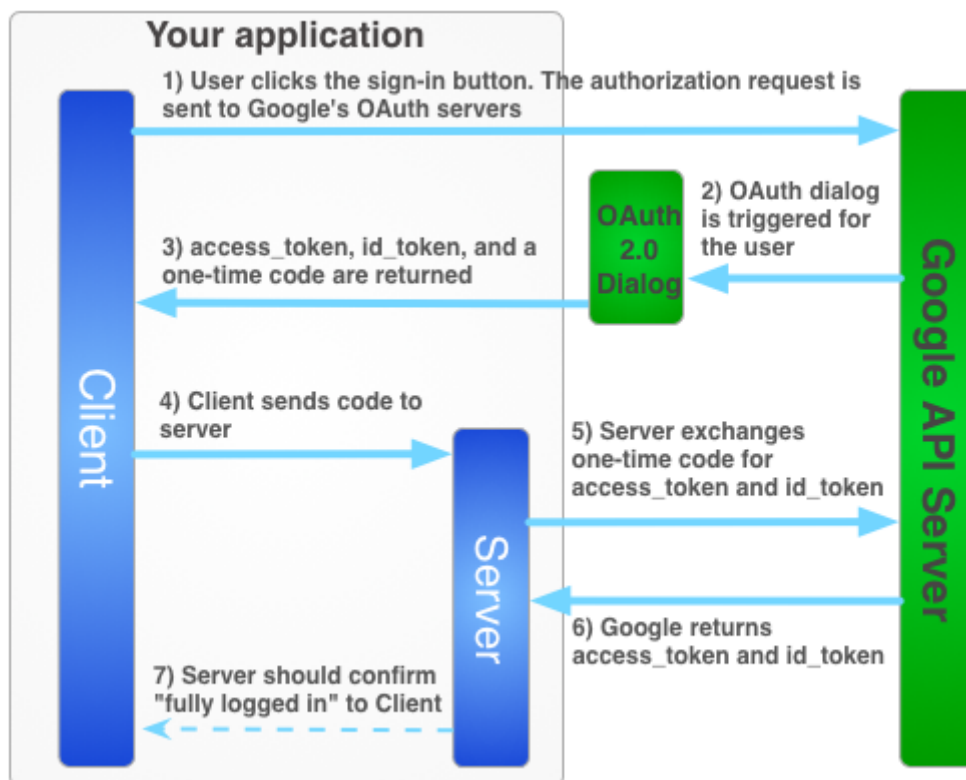


Figura 24. Estructura de comunicaciones sacada de: <https://developers.google.com/identity/sign-in/web/server-side-flow>

Finalmente, no se ha terminado de realizar este sistema de autenticación ya que nos hemos centrado en el proceso de gestión de código, incidencias, despliegue, etc.

3.6. API Rest.

Nuestro subsistema incluye una API REST para facilitar a otros subsistemas el consumir de autenticación. De esta manera es posible acceder a datos de login y usuario de manera programática. En concreto la API proporciona datos en formato JSON de los usuarios del sistema y "tokens" generados al conectarse. Se accede a la API a través de la siguiente URL: <https://autha.agoraus1.egc.duckdns.org/api/>

Los comandos soportados son los siguientes:

- GET USERS

Devuelve información de los usuarios

Parámetros: username - (opcional) - El nombre de usuario del que se desea obtener la información.

Sin éste parámetro se devolverán todos los usuarios.

- GET TOKEN

Devuelve si el token especificado es correcto

Parámetros: token - (obligatorio) - El token a comprobar.

Nota: La API está programada teniendo en cuenta que el módulo "Rewrite" de apache esté activado, de forma que sea accesible mediante URLs amigables. En el caso de que no estuviese activado, la manera de acceder es de la siguiente forma:

<https://autha.agoraus1.egc.duckdns.org/api/index.php?method=METHOD>

Ejemplos:

Obtener usuarios: `https://autha.agoraus1.egc.duckdns.org/api/index.php?method=getUsers`

Obtener un usuario: `https://autha.agoraus1.egc.duckdns.org/api/index.php?method=getUser&user=USERNAME`

Comprobar token: `https://autha.agoraus1.egc.duckdns.org/api/index.php?method=checkToken&token=TOKEN`

Además de la API, desde el subsistema se generan dos cookies: una de nombre "token" que guarda el token generado del usuario que se ha logueado, y otra "user" que guarda el nombre de usuario del mismo.

3.7. Modificaciones del proyecto.

El código heredado poseía algunas implementaciones de funcionalidades para un sistema de autenticación a la plataforma de voto electrónico. Ha sido nuestro cometido el de ampliar las funcionalidades que venían e incluso modificar algunas que ya traían y que a nuestro parecer eran deficientes. A continuación se enumeran las modificaciones realizadas.

- La implementación del acceso, ya descrito con anterioridad, a la plataforma mediante las credenciales que la red social de **Twitter** posee de un posible votante.

- La implementación del acceso, ya descrito previamente, a la plataforma de voto electrónico mediante los datos de acceso que la red social de **Facebook** pudiera poseer de un usuario del sistema de información web de Agor@Us.

- La implementación de un acceso al sistema mediante **doble verificación** de credenciales, en la cual se implementa un nuevo sistema de login independiente al del código heredado y además se añade la funcionalidad del envío de un código de verificación por “SMS” o por email, y decimos “SMS” ya que en la práctica se ha procedido a mostrar por pantalla el código para que el usuario lo copie y lo pegue, ya que sería muy engorroso montar un sistema de envío de SMS real.

- Se ha implementado una API con la que ofrezca una biblioteca para ser usada por otro software u otro subproyecto de Agor@US, con el fin de que haya la posibilidad de integración entre ellos.

En cuanto a la integración, el sistema no estaba integrado con ningún otro subsistema, aunque todos dependen de éste para poder acceder de manera correcta a sus servicios. En nuestro caso se han utilizado dos tecnologías para la integración continua:

- Se ha implementado el uso de **Jenkins** para la integración de manera local, ya que el servidor de integración ya tienen sus propios métodos de integración continua.

- Se ha procedido a implementar **Travis**, para hacer testing al código implementado en nuestro proyecto sincronizados con GitHub.

3.8. Planificación del proyecto.

En nuestro caso, para la planificación se ha tenido en cuenta que las reuniones en persona siempre son más productivas ya que no se pierde tanto el tiempo en la escritura de mensajes instantáneos, aunque éste método haya sido el segundo más usado, Telegram.

A la hora de repartir el trabajo el Jefe de Proyecto ha tenido en cuenta las habilidades de cada persona para asignar tareas, teniendo en cuenta siempre la carga de trabajo de cada tarea.

Para ser más exactos, a continuación se muestra una lista detallada con las tareas realizadas por cada integrante:

· **Alejandro Garrido Resina:** Alejandro realizó una **evaluación del código** heredado con el fin de discernir qué o qué no estaba hecho en el código heredado del año anterior. Una vez que esta labor terminó se preocupó de realizar '**issues**', en GitHub, de problemas encontrados. Seguidamente se centró en la incorporación de **Jenkins**, de manera local, para nuestro proyecto. Paralelamente se dedicó a perfeccionar la **API** que el año pasado no lograron pulir. Para la mejora del 2 de Enero está dedicando todo el tiempo a pulir las funcionalidades de la **API** para mejorar y perfeccionar, dentro de lo posible, el funcionamiento de ésta.

· **Alejandro Guerrero Montoro:** Se encargó principalmente de la supervisión de la seguridad implementada en el proyecto, asesorando de las buenas prácticas que habría que seguir en el proyecto para mantener un **bastionado** severo en este subproyecto de autenticación. Una vez concluida esta parte de adquisición de información y de la correcta entrega al resto de integrantes del grupo, se dedicó a la implementación del **doble login**. Una vez realizada la implementación de la funcionalidad antes mencionada, se dedicó a realizar la **documentación** explicativa referente al doble login y de Jenkins. Ayudó en la comprobación del correcto funcionamiento de la API realizando labores de **testing**. Una vez acabada ésta labor, para la ampliación del proyecto para el 2 de Enero, procedió a revisar la **documentación** de todo el proyecto y a realizarla acorde con las bases estipuladas por la asignatura, corrigiendo, modificando y mejorando el estado del documento a entregar.

· **María Remedios Dans Caballero:** En un principio se dedicó a la búsqueda de **información** para la correcta implementación de la funcionalidad de acceso mediante la **API de Facebook**. Para concluir se dedicó a la realización de la **documentación** referente a Facebook. Para la ampliación del 2 de Enero se encargó de la creación del **diario de grupo**.

· **Alejandro Román Rodríguez:** En un principio se encargó de la realización de la **configuración** de la **máquina virtual**, instalación de git, xampp y aptana. A continuación procedió a investigar la implementación del **certificado digital**. Esta labor, bastante difícil, finalmente no llegó a realizarse dado que el profesor de la asignatura recomendó no complicar el proyecto. Debido a lo mencionado se dedicó a implementar el control de acceso a la plataforma mediante las credenciales que **Twitter** guarda de un posible votante. Esto se realizó implementando utilizando servicios de la API de Twitter. Finalmente se dedicó a crear la documentación de esta implementación y de otros elementos. Para la mejora del 2 de Enero, se centró en la implementación de la tecnología de **Travis** para el testing de código de GitHub y a rellenar la documentación correspondiente.

· **Julio Márquez Castro:** Empezó con la corrección de la **configuración** de la máquina virtual, puliendo algunos detalles necesarios que previamente no se habían tenido en cuenta. Una vez acabada dicha tarea procedió al estudio de la **integración** continua del proyecto entablando relaciones con otros miembros de diferentes subproyectos. Para terminar se centró en completar la tarea de implementación del acceso a la plataforma de voto electrónico mediante **Facebook**. Para la mejora del 2 de Enero se centró en la implementación de la tecnología de **Travis** para el testing de código de GitHub.

4. Elementos de Control.

Los llamados Configuration Items (CI) o elementos de configuración son esos productos finales, intermedios o finales, productos a entregar.

A continuación, se muestra una lista de los elementos de configuración:

- Código fuente.
- El proyecto en formato ejecutable.
- Todos los datos que hagan referencia al proyecto en sí.
- La documentación técnica.
- En el caso de que sea procedente, la máquina virtual con el proyecto desplegado.

5. Entorno de desarrollo.

En este apartado describiremos cómo hemos trabajado y sobre qué herramientas lo hemos hecho.

- Para empezar decir que se ha montado una máquina virtual con un **Windows 7** y para ejecutarla se ha usado el software **VMware**. Esto se ha hecho así ya que es beneficioso para el propósito que nos ocupa. El uso de una máquina virtual es beneficioso porque nos permite usar un equipo con unas herramientas determinadas sin que pueda existir conflicto de versiones con otras herramientas que se utilicen para otros proyectos. Otro beneficio es que todo el equipo puede trabajar con el mismo entorno de trabajo y que todo sea compatible. Por último, se ha tenido en cuenta que a la hora de configurar herramientas que se hayan incorporado en un estado avanzado del proyecto, si no ha salido de la manera esperada y se han cometido fallos, solo hay que tomar una copia de la máquina virtual y probar desde cero, como si de un punto de restauración se tratase.

- Se ha montado un servidor de manera local para probar el proyecto antes de la integración, y esto ha sido posible gracias al **XAMPP**.

- Para trabajar con el código, se ha utilizado **Aptana** como editor de texto formateado.

- Para testear la API realizando pruebas se ha usado el software **Insomnia**.

- Para la realización de tests en integración se ha usado **Travis**.

- Para el despliegue de la app se ha usado **Jenkins**.

- Por último se ha procedido a usar **Git** para la gestión del código fuente.

6. Gestión del código fuente.

A la hora de trabajar con código, una buena práctica es la de centralizar el código en un repositorio que tenga control de versiones y la posibilidad de editar conflictos de concurrencia. Con esta medida conseguimos que todos los desarrolladores puedan trabajar con el código de todos. Centralizar el código es una práctica que beneficia al proyecto por su inmediata disponibilidad y la gestión de conflictos que éstas herramientas nos ofrecen.

Es por ello que para la gestión de código, como ya se ha dicho con anterioridad, se ha querido usar un repositorio de **GitHub**, concretamente llamado '[AgoraUS-G1-1617/Autentication](https://github.com/AgoraUS-G1-1617/Autentication)' (<https://github.com/AgoraUS-G1-1617/Autentication>). En la imagen que se muestra a continuación podemos apreciar una vista previa del repositorio mencionado.

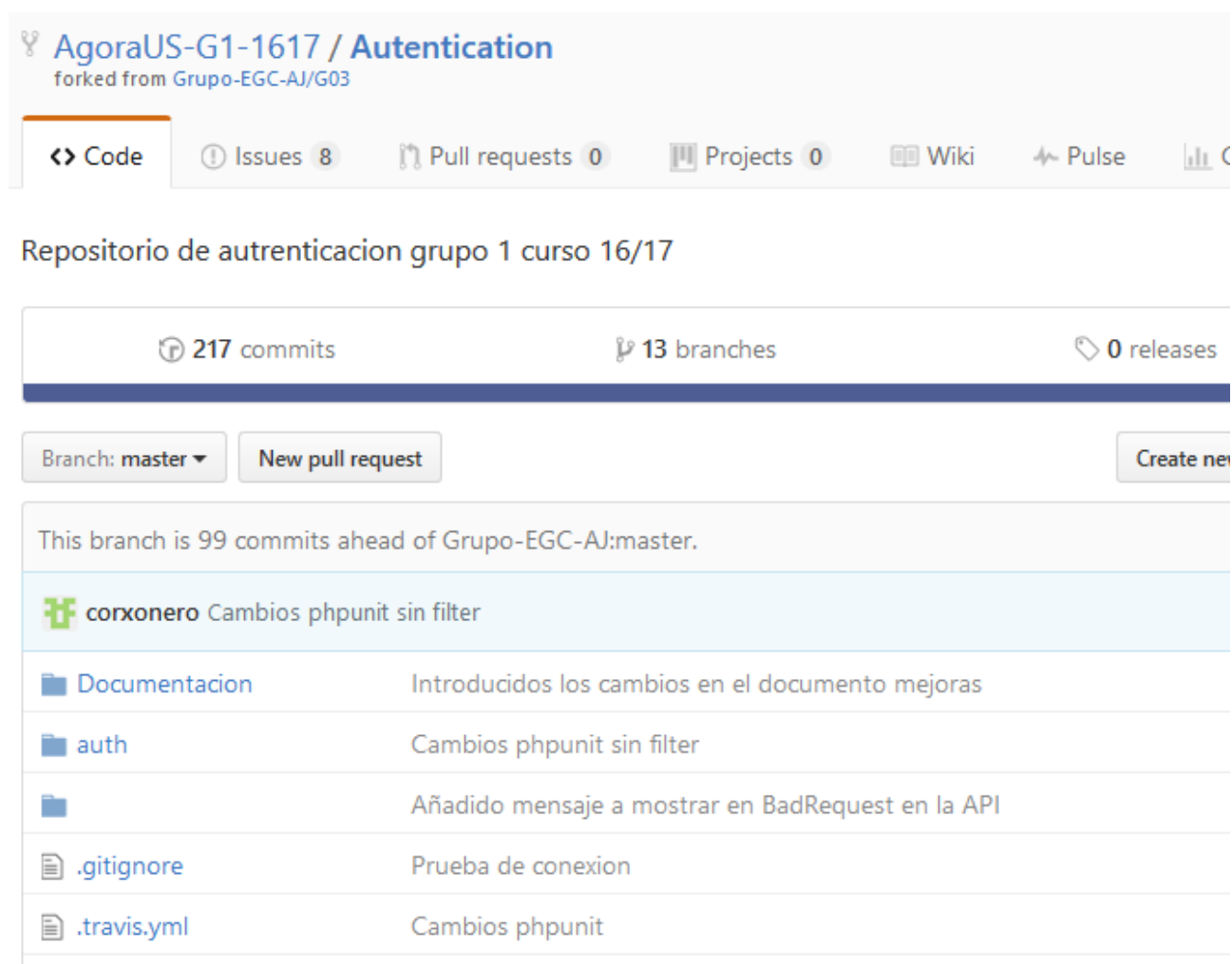


Figura 25: Repositorio de 'AgoraUS-G1-1617/Autentication'.

Una vez llegados a este punto, podemos asegurar que es el método a seguir. Pero para asegurarnos del buen funcionamiento de este método se decidió dedicar tiempo a la planificación de la estructura del repositorio. Esto es así porque se decidió generar diferentes **ramas** o **branches** para las diferentes tareas asignadas a diferentes desarrolladores. En cada rama se trabaja con una tarea en concreto y se va subiendo al repositorio las tareas terminadas. Con las tareas terminadas dentro de sus respectivas ramas se procede a unir las diferentes ramas con la rama principal(master) si pasan los tests correspondientes y así llegamos a tener una rama unificada con todo el trabajo realizado y actualizado.

Para tener un mayor entendimiento del método usado se ilustrará con un ejemplo:

- Alejandro Guerrero Montoro tiene que realizar una tarea específica, la de implementar el doble login para el control de acceso a la plataforma de voto electrónico.
- Alejandro crea una rama llamada 'security' para poder trabajar con comodidad con la certeza de no entorpecer el trabajo de ningún otro integrante del grupo.
- Una vez que tiene la rama creada procede a investigar cómo hacer su tarea, la realiza y la sube al repositorio, concretamente a su rama 'security'.
- Vemos que pasan correctamente los tests.
- Ya sabemos que funciona todo correctamente, lo siguiente que se va a hacer es fusionar la rama 'security' y la rama 'master' que es la rama principal.
- Por último sabemos que la rama 'master' ya está unificada con la 'security' y posee todas las implementaciones que tenía anteriormente y las nuevas con el doble login.
- Cada usuario hará lo mismo con sus ramas pertinentes y así la rama 'master' estará actualizada siempre.

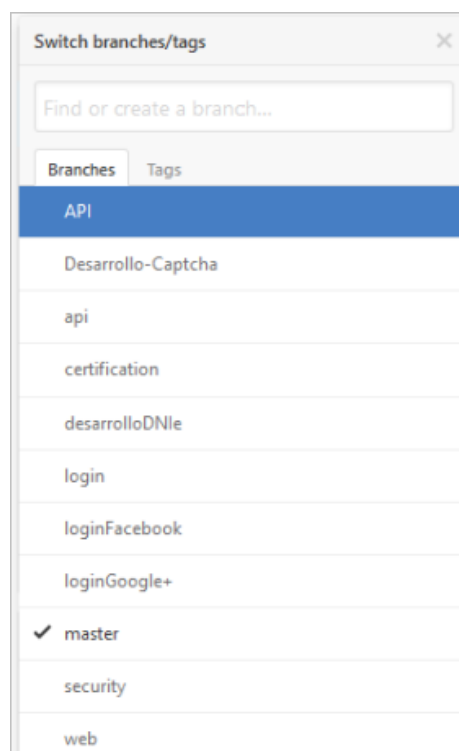


Figura 26: Visualización gráfica de las ramas implementadas en el repositorio.

7. Gestión de la integración continua.

Nuestra gestión de integración continua comienza en el momento que algún integrante del grupo realiza cambios en alguna de sus ramas, posteriormente la aplicación Travis realiza los test de nuestro proyecto. Una vez que la aplicación Travis realiza los test procederemos a fusionar nuestra rama con la rama “master” siempre que los tests haya sido satisfactorios. Una vez actualizada en master, el sistema se despliega a través del software Jenkins implementado en un servidor de la Universidad de Sevilla, que usamos todos los subsistemas, lo que facilita la integración entre los diferentes grupos.

Para las pruebas configuramos nuestro repositorio, mediante el fichero “.travis.yml” para que Travis ejecute los test de nuestro sistema usando mysql, el cual, crea nuestra base de datos. Phpunit se usará para ejecutar los test, y bootstrap, para una correcta visualización del resultado de los tests.

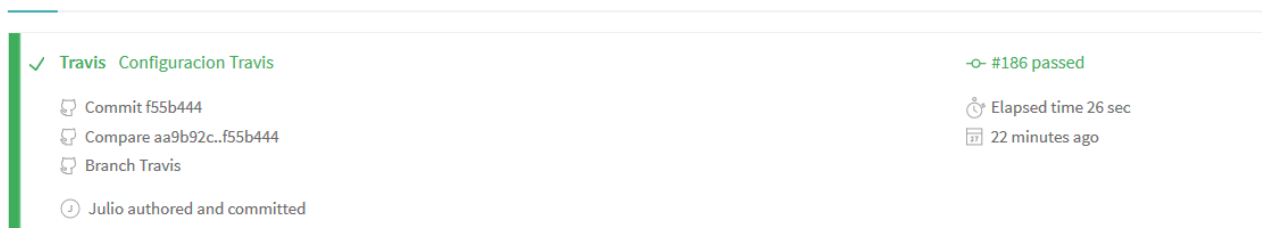


Figura 27: Travis. Configuración.

```
260 $ echo 'Before script'
263 $ mysql -e 'create database IF NOT EXISTS egcajtk_egcdb;'
265 $ mysql -u root -e "USE egcajtk_egcdb;"
267 $ mysql -u root -e "CREATE USER 'egcajtk_egc1617'@'localhost' IDENTIFIED BY 'egc1617aj';"
269 $ mysql -u root -e "GRANT ALL PRIVILEGES ON egcajtk_egcdb.* TO 'egcajtk_egc1617'@'localhost' IDENTIFIED BY 'egc1617aj';"
271 $ phpunit --configuration $TRAVIS_BUILD_DIR/phpunit.xml
272 PHPUnit 5.5.0 by Sebastian Bergmann and contributors.
273
274 .....
275
276 Time: 1.05 seconds, Memory: 15.25MB
277
278 [Progress bar]
279
280 The command "phpunit --configuration $TRAVIS_BUILD_DIR/phpunit.xml" exited with non-zero status 1
281 $ ./marge_script.sh
```

Figura 28: Travis. Salida por pantalla.

```
272 PHPUnit 5.5.0 by Sebastian Bergmann and contributors.
273
274 .....
275
276 Time: 1.05 seconds, Memory: 15.25MB
277
278 [Progress bar]
279
280 The command "phpunit --configuration $TRAVIS_BUILD_DIR/phpunit.xml" exited with non-zero status 1
281 $ ./marge_script.sh
```

Figura 29: Travis. Successfull.



Figura 30: Jenkins. Estado actual.

Necesitamos configurar el entorno de despliegue de Jenkins y dado que hay diferentes configuraciones para cada subsistema, se ha optado por la implementación de dockers para facilitarnos a nosotros mismos y al propio servidor, la construcción y despliegue de los distintos entornos.

El despliegue se divide en tres fases:

- **Make:** Se copia y se descarga el código del repositorio cuando haya cambios y se prepara para ser lanzado.
- **Beta:** Se ejecuta automáticamente una vez realizada la fase **Make**. Primero elimina la aplicación ya creada anteriormente y despliega la nueva aplicación preparada en la fase **Make**.
- **Stable:** Esta fase es básicamente igual que la fase **Beta**, pero se ejecuta manualmente y la principal diferencia que debemos tener en cuenta es que la aplicación desplegada en esta fase debe ser una versión estable, con todas las pruebas necesarias y revisiones oportunas para una buena integración con el resto de subsistemas.

Nuestros ficheros de configuración están disponibles para su revisión en el siguiente enlace. Son ficheros de comandos de Shell, donde se ejecutan los dockers necesarios para nuestro sistema.

https://github.com/AgoraUS-G1-1617/continuous-integration-agoraus_g1/tree/master/AgoraUS/G7-AutenticacionA

En cuanto a la integración con otros subsistemas, otros subsistemas consumen de nosotros para realizar logins. Por ejemplo, le subsistema Deliberations, que hace uso de nuestra api para realizar logueo en su aplicación. Podemos ver en su controlador "UserController" ([enlace a la clase](#)) como invoca nuestro sistema.

```
// Se recupera la respuesta a la petición

response = objectMapper.readValue(new URL("https://autha.agoraus1.egc.duckdns.org/api/index.php?method=checkToken&token=" + tokenToVerify),
    Token.class);
```

Figura 31: ObjectMapper.

URL de la aplicación en Jenkins: <https://jenkins.egc.duckdns.org/>

URL del sistema beta desplegado: <https://beta.autha.agoraus1.egc.duckdns.org/>

URL del sistema estable desplegado: <https://autha.agoraus1.egc.duckdns.org/>

8. Gestión de incidencias.

Una buena manera de proceder en un proyecto de software es llevar a cabo una gestión de incidencias organizada, y en nuestro caso se ha realizado mediante el propio gestor de **incidencias** de **GitHub**.

El gestor que esta plataforma nos ofrece viene preparada para avisar por correo al desarrollador que tenga asignada la tarea, tanto en la asignación como en el momento en el cual la tarea ha sido cerrada. Viene preparada con códigos de **colores** para diferentes tipos de incidencias, como cambios de funcionalidad o errores o alertas, por ejemplo. En la siguiente imagen podemos ver el aspecto que tiene.



Figura 32: Apartado de Issues de GitHub.

Las incidencias las puede crear cualquier usuario accediendo al formulario con el aspecto que se ilustra en la siguiente imagen.

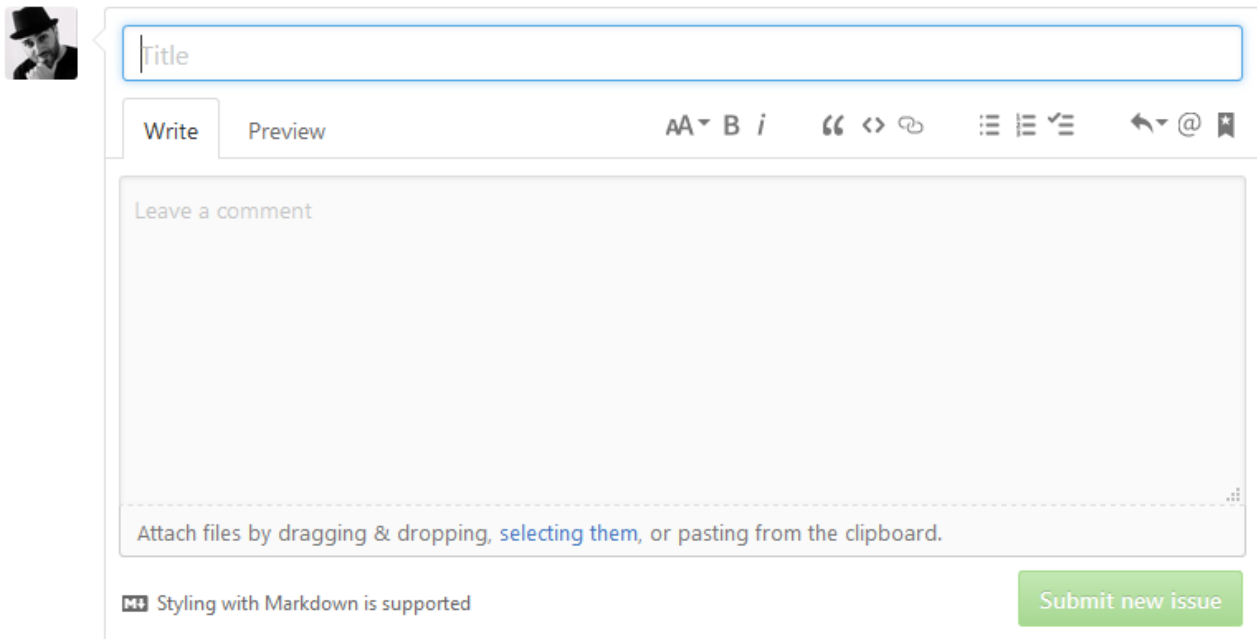
The image shows a web form for creating a new issue. At the top left is a small profile picture of a man wearing a hat. To its right is a text input field labeled 'Title'. Below the title field are two tabs: 'Write' (active) and 'Preview'. To the right of the tabs is a rich text editor toolbar with icons for bold, italic, quote, code, link, list, and other formatting options. Below the tabs is a large text area with a placeholder text 'Leave a comment'. At the bottom of the text area is a dashed line and a note: 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' Below the text area is a small icon and the text 'Styling with Markdown is supported'. At the bottom right is a green button labeled 'Submit new issue'.

Figura 33: Formulario para la creación de una nueva issue.

Además podemos asignar un commit a la incidencia correspondiente gracias al código # y el número de la incidencia correspondiente.

9. Gestión de liberaciones y despliegue.

En un proyecto de software es muy común que, después de la fase de producción, se decida desplegar la aplicación para su posterior portabilidad y despliegue. En nuestro caso este proceso se llevará a cabo mediante **Jenkins** para el servidor de la facultad y usamos **FileZilla** para subir la versión estable de nuestra aplicación a un servidor aparte del servidor común de todos los subsistemas.

Para nuestro servidor aislado usamos <https://www.hostminio.es/>, desplegamos usando ftp mediante el programa **FileZilla**. El sistema queda desplegado en la siguiente dirección: <http://egcaj.tk/>.

Como ya se ha explicado con anterioridad, en la rama master convergerán todas las ramas de pruebas que estaban asignadas según la tarea específica a realizar, es en estas ramas donde se procederán a ejecutar los tests y si pasan los tests por la aplicación de Travis se procederá a subir el código a la rama master donde se desplegará la aplicación en Jenkins. Una vez desplegada y testeada la versión beta, procedemos a contactar con el administrador del Jenkins común y lanzamos la fase estable. El sistema queda desplegado en la siguiente dirección: <https://autha.agoraus1.egc.duckdns.org/>.

10. Mapa de herramientas.

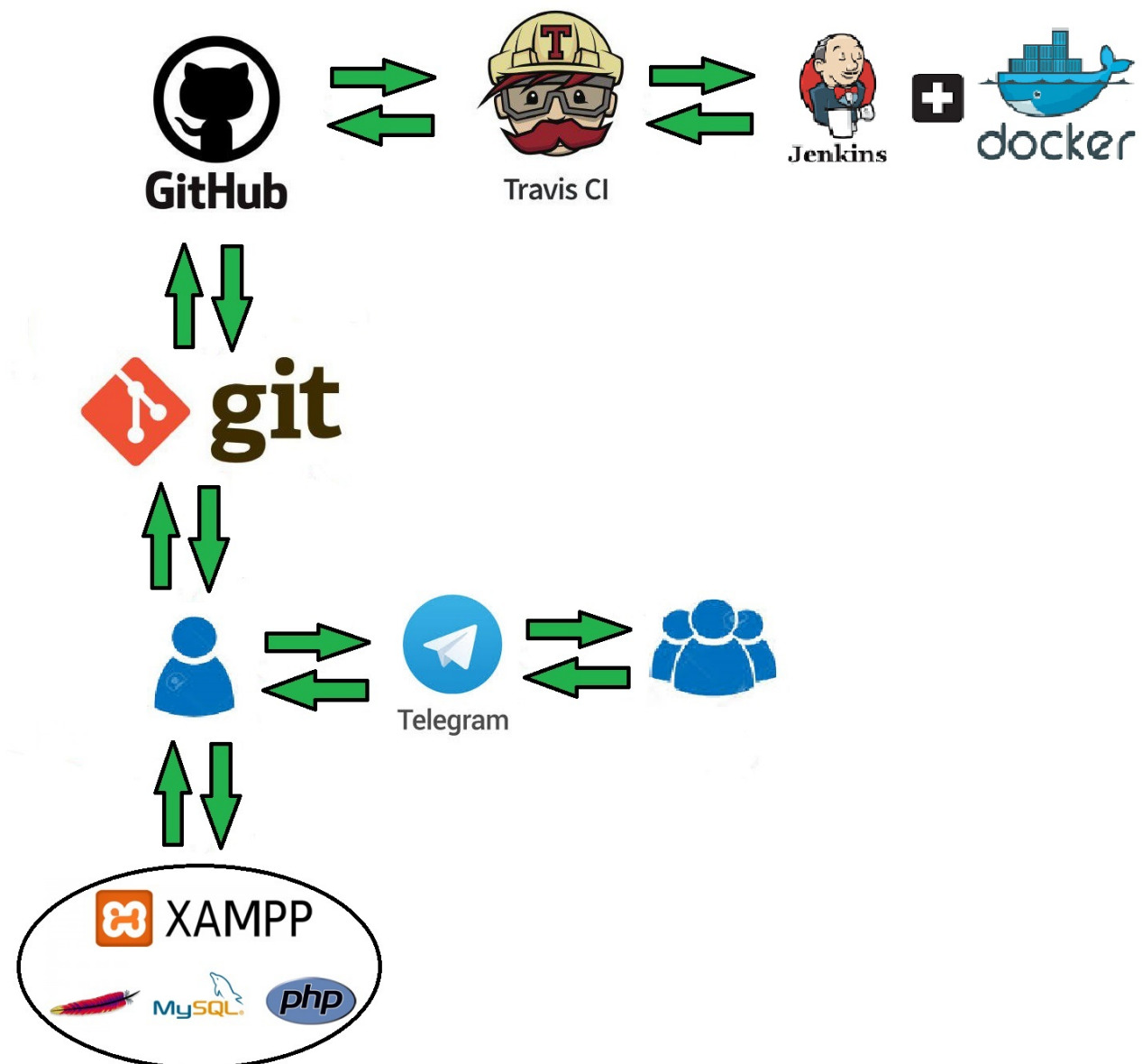


Figura 34: Mapa de herramientas.

Herramientas que utilizamos en cada uno de nuestros PCs:

- **Xampp:** Es una herramienta de desarrollo que te permite probar tu trabajo en tu propio ordenador, gestiona base de datos MySQL, servidor web Apache e intérpretes para lenguajes de script: PHP y Perl. Lo utilizamos junto a las herramientas mencionadas para implementar funcionalidades.
- **Git:** Es un software de control de versiones que utilizamos para poder hacer cambios sin miedo a perderlo.
- **GitHub:** Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git, que utilizamos para que todos los componentes del grupo podamos trabajar en el proyecto.

Herramientas que utilizamos en el servidor de la asignatura:

- **Jenkins:** Es un software que proporciona integración continua para el desarrollo de software.
- **Docker:** Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software.
- **Bootstrap:** Conjunto de herramientas HTML y CSS de código abierto que contienen tipografía, formularios, botones, cuadros, menús de navegación, entre otros, así como extensiones de JavaScript opcionales adicionales.

Herramientas para las pruebas:

- **Travis:** Software de integración continua. Brinda un servicio web para ejecutar pruebas en proyectos alojados en GitHub.

Herramientas de comunicación:

- **Telegram:** Es un servicio de mensajería por internet enfocado en la gestión de mensajes de texto y multimedia. Lo utilizamos para la comunicación entre los miembros del grupo.

11. Conclusiones y trabajos futuros.

Han sido muchas las lecciones aprendidas gracias a la elaboración de este proyecto. Con una visión más general podríamos ver que gracias a la asignatura al completo hemos aprendido a trabajar con una **gran cantidad de personas**, coordinando tareas de manera efectiva y con buenos resultados.

Centrándonos en el proyecto, podemos decir que hemos aprendido a trabajar, de manera más **profesional**, con un grupo elevado de personas, haciendo que todo esté coordinado y funcione como debiera.

Hemos aprendido a trabajar en un entorno profesional de gestión de código gracias a **Git y GitHub**. Con esta herramienta hemos aprendido a gestionar las tareas e **incidencias** de código de una manera mucho más eficiente hasta el punto de pensar en aplicar estas metodologías para otros proyectos y asignaturas. Es difícil pensar en trabajar con código de otra manera diferente a la que se ha usado en este proyecto.

Se ha desarrollado una metodología de aprendizaje **autodidacta**, sobre todo para el aprendizaje de las **nuevas tecnologías**. Gracias a la necesidad de aprender a integrar nuevas tecnologías se ha tenido que mejorar la capacidad de investigación y profundización en las tecnologías a implementar.

De las principales lecciones aprendidas a destacar, de entre todas, es el trabajar con **código heredado**. A lo largo de nuestro aprendizaje académico siempre había sido de alta dificultad el hecho de interpretar código ajeno, o incluso propio pero con cierta antigüedad. Gracias a este proyecto las peculiaridades que se solían encontrar a la hora de interpretar este tipo de código han ido disminuyendo hasta el punto de llegar a cierto punto de comodidad trabajando con el código.

En cuanto a las futuras direcciones que debería tomar este subproyecto, resaltar que el tema principal a tratar es la **seguridad** y debido a la estructura a seguir de la asignatura, no se ha podido llevar a cabo de manera estricta y debería de ser un punto a tener en cuenta. Un ejemplo de ello sería la **eliminación** de las implementaciones del acceso mediante **redes sociales**, ya que una persona puede llegar a tener varias cuentas activas en una misma red social y puede votar varias veces. O puede tener una cuenta en diferentes redes sociales y poder votar varias veces. La mejor forma de acceder sería implementando el acceso mediante **DNi**, **certificado digital** y acceso mediante **doble login**, siendo el doble login transmitido por una vía segura y personal como lo puede ser el envío de **SMS**.