Proyecto Final

Sistema de procesamiento de estadísticas de redes sociales

Estudiante

Pablo Ezequiel Horst

DNI: 32.512.140

Número de legajo: 3316

Modalidad: práctica en empresa

Docente tutor por parte de la Facultad

Dra. Gabriela Minetti

Tutor de la práctica por parte de la empresa

Ing. Leonardo Torres

CARRERA

Ingeniería en Sistemas

Plan 2004

Facultad de Ingeniería

Universidad Nacional de La Pampa

General Pico, La Pampa

1 de junio de 2016

Tabla de Contenidos

Índice de figuras

Resumen

El presente proyecto se realizó como práctica final supervisada de la carrera Ingeniería en Sistemas (plan 2004) de la Universidad Nacional de La Pampa. El mismo muestra el proceso de desarrollo de una aplicación web de nombre "Twoorl" cuyo objetivo es reunir datos de cuentas de *Facebook* y *Twitter* de un individuo particular utilizando las APIs públicas de dichos sitios, para luego obtener estadísticas que permitan de forma intuitiva y ordenada comprender mejor el impacto que está teniendo dicho usuario en las redes sociales.

El proyecto fue desarrollado en el marco de la empresa Ipaddress S.R.L., dedicada al desarrollo de aplicaciones web a medida. Se utilizó un *framework* propio de la empresa escrito en PHP, razón por la cual fue ese el lenguaje elegido para la programación junto con otras tecnologías web, como lo son Javascript, CSS, AJAX, entre otros.

Debido a la necesidad de obtención de datos de las redes sociales, se realizó una investigación de las APIs públicas y los correspondientes SDK (*Software Development Kit* o Kit de desarrollo de Software) escritos en PHP que permiten una fácil utilización de las mismas. Además, dado al gran volumen de información que manejará el sistema, se diseñó un subsistema de obtención y procesamiento de datos que utilizará una base de datos NoSQL para tal fin.

Como interfaz con el usuario, se desarrolló un *front-end* de la aplicación, desde el cual se pueden asociar cuentas de redes sociales a un usuario del sistema, administrarlas y observar sus estadísticas.

Agradecimientos

Quiero aprovechar la oportunidad para agradecer a mi familia, especialmente a mis padres, Nancy y Daniel, que siempre me apoyaron durante toda mi carrera desde el primer día hasta el último. A mis compañeros de estudios y amigos que siempre estuvieron cuando los necesité, dándome una mano o un “tirón de orejas” cuando hizo falta. También agradezco a mi directora de tesis, Gabriela Minetti, por brindarme su acompañamiento este último tiempo y su enorme paciencia para con mis tiempos. A ellos y a todos los que estuvieron presentes en este camino que recorrí quiero decirles, gracias y mil gracias.

1. Introducción

Hoy en día es cada vez más frecuente el uso de redes sociales llámense *Facebook*, *Twitter*, entre otras, por parte de organizaciones como medio de comunicación y de promoción. Muchas empresas, además, utilizan y analizan la retroalimentación obtenida en dicho medio para evaluar el impacto de sus productos y servicios, así como también de las diferentes campañas publicitarias relacionadas a los mismos [1]. Tanto *Facebook* como *Twitter* poseen plataformas que brindan estadísticas sobre sus cuentas a sus usuarios, ya sea variación de la cantidad de "Me gusta" de una página en *Facebook* o cantidad de seguidores de una cuenta de *Twitter* [2, 3, 4]. Pero muchas veces la información brindada es limitada. *Twitter* nos muestra los mejores *tweets* de la cuenta ordenados por la cantidad de veces que fueron reenviados o *"retweeteados"*, pero no por otros criterios importantes; tales como la cantidad de respuestas y las veces que fue marcado como favorito. La idea es brindar información y estadísticas que las plataformas no muestran y, así, analizar qué contenido fue el más exitoso mientras se provee información para hacer ajustes y así mejorar la recepción y el impacto del producto o servicio publicado en las redes sociales.

En el presente proyecto se desarrolla una aplicación web llamada "Twoorl" que obtiene datos de cuentas de redes sociales que una organización o un individuo particular posean (denominadas de ahora en más usuarios del sistema), con el objetivo de obtener información relevante para el usuario. La obtención de datos se realiza por medio de las APIs de dichas redes sociales, con su correspondiente autorización. Como resultado del procesamiento de estos datos se obtienen estadísticas que permiten a los usuarios comprender mejor diferentes aspectos de la cuenta de red social analizada, tales como: crecimiento e información demográfica de seguidores, alcance y rendimiento de los contenidos, según tipo y cantidad, datos sobre los consumidores del contenido más activos, entre otros.

Este proyecto fue desarrollado en el marco de la empresa IpAddress S.R.L., la cual posee dos departamentos: uno encargado de *web hosting* que brinda servicios de alojamiento de aplicaciones web y administración de sistemas, y otro de desarrollo de software cuyo objetivo es crear software web a medida. Es en este último que el proyecto fue desarrollado.

1.1 Propósito, objetivos y alcance

El propósito del sistema Twoorl es el de proveer estadísticas sobre diferentes aspectos de una cuenta de *Twitter* o página de *Facebook*. Dado el crecimiento actual de las redes sociales, muchas empresas, organizaciones e individuos las utilizan como medio de promoción. Twoorl brinda información de calidad para que dichos usuarios puedan comprender cómo los diferentes contenidos publicados en la red social impactan en el público, evaluar el crecimiento de la cuenta y de sus seguidores. Esto permitiría realizar un análisis y evaluar dicha cuenta, pudiendo luego, tomar decisiones que impacten en la misma.

Los objetivos específicos del sistema se detallan a continuación:

- Relevar y analizar las redes sociales *Facebook* y *Twitter*, sus APIs públicas.
- Analizar los potenciales datos a obtener y los diferentes resultados estadísticos a alcanzar a partir de tales datos.
- Diseñar y programar un sistema de obtención y procesamiento de datos de las diferentes redes sociales mediante APIs públicas de dichos sitios.
- Diseñar y programar un *front-end* para la aplicación web, el cual será accedido por los usuarios y les permitirá visualizar las diferentes estadísticas de sus cuentas de *Twitter* y *Facebook* conectadas a su cuenta.

1.2 Contexto del sistema

En una primera instancia, el sistema estará alojado en un servidor presente en internet, el cual tendrá el *back-end*, el *front-end*, la base de datos relacional MySQL, los procesos de obtención y procesamiento de datos y la base de datos NoSQL MongoDB utilizada para tal fin. Luego, en función del crecimiento del sistema, se prevé migrar la base de datos relacional MySQL a un servidor separado y, a su vez, mover el proceso de obtención y procesamiento de datos y la base de datos NoSQL MongoDB a uno o más servidores (replicación).

En cuanto al software, en este caso se utilizó un *framework* implementado por la empresa de desarrollo de software (Ipaddress S.R.L.) para la realización de software a medida. Dicho *framework* sigue el patrón de arquitectura de software conocido como Modelo Vista Controlador (MVC), cuyo propósito es separar los datos y la lógica de negocio del software, de la interfaz de usuario y del proceso encargado de gestionar los eventos y las comunicaciones. Para tal fin, se desarrollan tres componentes distintos que son el modelo, la vista y el controlador, que definen componentes para la representación de la información, por un lado, y la interacción del usuario por el otro. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento [4, 5].

1.3 Recursos

Para desarrollar el sistema se utilizaron computadoras personales, conectadas en red con un servidor de la empresa como ambiente de pruebas. Este cuenta con un sistema operativo Linux que posee un motor de base de datos MySQL, un motor de base de datos NoSQL MongoDB y un servidor SVN para control de versiones y cambios.

La computadora empleada para desarrollar cuenta con un sistema operativo Microsoft Windows y se utiliza el siguiente software:

- Zend Studio 10 [6]: IDE de programación para PHP.
- HeidiSQL [7]: administrador de Base de Datos SQL.
- Tortoise SVN [8]: software para utilización de SVN como método de control de versiones.
- Apache [9]: utilizado como servidor web.
- Navegadores Web: Google Chrome, Mozilla Firefox, Internet Explorer versión 9 y 10.

- MongoDB [10]: sistema de base de datos de tipo NoSQL.
- MySQL Workbench [11]: software de gestión de base de datos, que en este caso, fue utilizado para la creación de diagramas de base de datos.
- *Framework* propio de la empresa Ipaddress S.R.L.

Como lenguaje de programación fue elegido PHP, utilizado con anterioridad por la empresa y, además, el *framework* está escrito en dicho lenguaje. También se utilizan tecnologías web tales como HTML, hojas de estilo en cascada CSS, *Javascript*, AJAX y *JQuery*.

1.4 Estimación del proyecto de software

El proyecto está dividido en instancias que están formadas por tareas y marcan hitos dentro del proceso de desarrollo del sistema. A continuación se describen cada una de ellas:

- La **primera** instancia del trabajo está comprendida por una investigación sobre las APIs públicas de *Facebook* y *Twitter*, los datos a obtener y potenciales estadísticas a partir de ellos.
- En **segunda** instancia se diseña el esquema de bases de datos del sistema, el sistema de obtención y procesamiento de datos y el *front-end* del sistema.
- En la **tercera** etapa se implementa el esquema de base de datos previamente diseñado.
- En la **cuarta** etapa se desarrolla un subsistema que permita la automatización de la obtención de los datos, almacenamiento, posterior proceso y guardado de los datos obtenidos en las redes sociales.
- La **quinta** parte consta del desarrollo de un *front-end* de la aplicación web, el cual será accedido por los usuarios del sistema.

- La **sexta** instancia consiste en la redacción del informe correspondiente al trabajo realizado.

En la Figura 1 se puede apreciar el diagrama de Gantt que describe gráficamente el cronograma de actividades agrupadas en las instancias nombradas anteriormente.

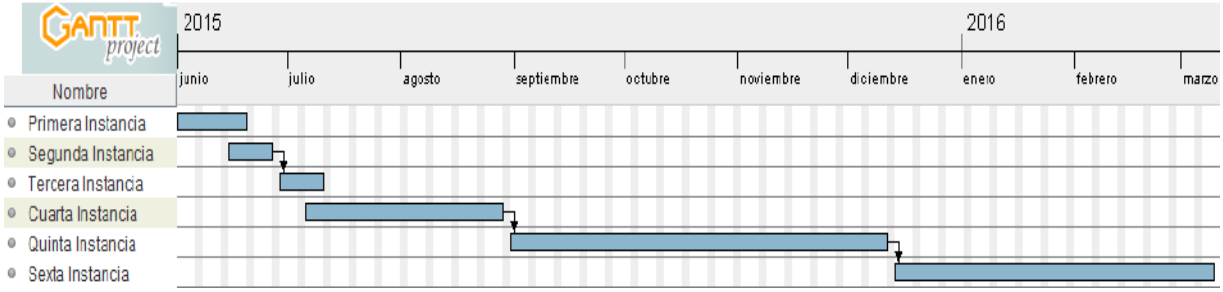

Figura 1: Diagrama de Gantt del proyecto.

1.5 Comunicación con el cliente

La comunicación inicial con el cliente la cual incluye planificación y negociaciones iniciales fueron realizadas por los directivos de la empresa Ipaddress S.R.L. Durante el desarrollo, la comunicación con el cliente se realizó mediante llamadas a través de Skype, cuya periodicidad varía según la necesidad de realizar consultas y mostrar avances. Se acuerda el cronograma de conferencias enviando correos electrónicos al cliente. Si el objetivo de la comunicación es mostrar avances, se prepara una pequeña demostración utilizando la función de compartir pantalla de Skype en la cual el cliente puede observar los cambios realizados al software.

2. Tareas de desarrollo

En el siguiente capítulo se detallan las diferentes tareas de relevamiento, diseño e implementación realizadas. Es importante aclarar que por cuestiones de confidencialidad, no se incluirán aspectos monetarios y solamente algunos aspectos del código del sistema.

2.1 Investigación de las APIs públicas de *Facebook* y *Twitter*

La primera instancia del trabajo consistió en investigar las APIs públicas de *Facebook* y *Twitter*. Ambas redes sociales cuentan con amplia y detallada información y documentación ahondando en las características, el sistema de autenticación, de comunicación y los métodos que pueden utilizarse tanto para obtener como para enviar datos. Se hizo hincapié en investigar cómo obtener los datos necesarios para la confección de las estadísticas que solicitó el cliente y además, en buscar un SDK para utilizar las APIs de ambas redes sociales para PHP y *Javascript* [12, 13].

Luego de realizada la investigación, se obtuvieron SDKs oficiales para PHP y *Javascript* tanto de *Facebook* como de *Twitter* y, también la información necesaria para su utilización y obtención de los datos requeridos por el sistema.

2.1.1 API de *Facebook*

Facebook provee una API que permite interactuar con la red social de diferentes maneras. Está formada por 3 componentes principales: *Atlas* API, *Marketing* API y *Graph* API. La primera es una plataforma relativamente nueva y compleja que está orientada al marketing pago dentro de la red social y su acceso se encuentra restringido para ciertas aplicaciones en una lista blanca conformada por *Facebook*. Luego, la llamada *Marketing* API, de manera similar a la descrita anteriormente, se utiliza para obtener información sobre publicidad paga, automatizar procesos de publicidad y obtener reportes al respecto. La *Graph* API, a diferencia de las anteriores, no está diseñada para marketing, sino que es la forma principal de obtener e ingresar información en la

red social. Es una API de bajo nivel basada en HTTP con la cual se puede obtener datos, postear nuevas historias, subir fotos, entre otras posibilidades. Luego de una breve lectura de los 3 componentes de la API de *Facebook*, se concluyó que la API a utilizar para obtener los datos necesarios para el desarrollo del sistema es *Graph* API [14, 15, 16]. Para poder usar esta última se debe crear una Aplicación de *Facebook* (*Facebook App*) desde la página de desarrolladores [15]. Se creó una aplicación de nombre Twoorl para ser utilizada por el sistema como se puede observar en la Figura 2. Ésta tendrá asociado un número de identificación de aplicación y un número secreto similares a un par de claves pública y privada respectivamente, para ser utilizados en el sistema de autenticación. Para esto último se debe utilizar *Facebook Login*.

Figura 2: Tablero (*dashboard*) en el que se muestra la *Facebook App* creada.

2.1.1.1 *Facebook Login*

Facebook Login permite a los usuarios crear cuentas y *loguearse* en una *Facebook App* a través de múltiples plataformas. Además, *Facebook Login* se utiliza como método de autenticación mediante el cual se le solicita a un usuario *logueado* en *Facebook*, los permisos correspondientes que la *Facebook App* pretende utilizar (ver Figura 3). De ser aceptada la solicitud, el usuario de *Facebook* quedará registrado en la aplicación permitiendo a esta última obtener un *token* de acceso. Un *token* de acceso es una cadena de texto que identifica a un usuario y puede, y en muchos casos debe, ser utilizado para realizar peticiones a una API. En el caso de *Facebook*, incluye información sobre su fecha de expiración y el identificador de la *Facebook App* que lo creó [17].

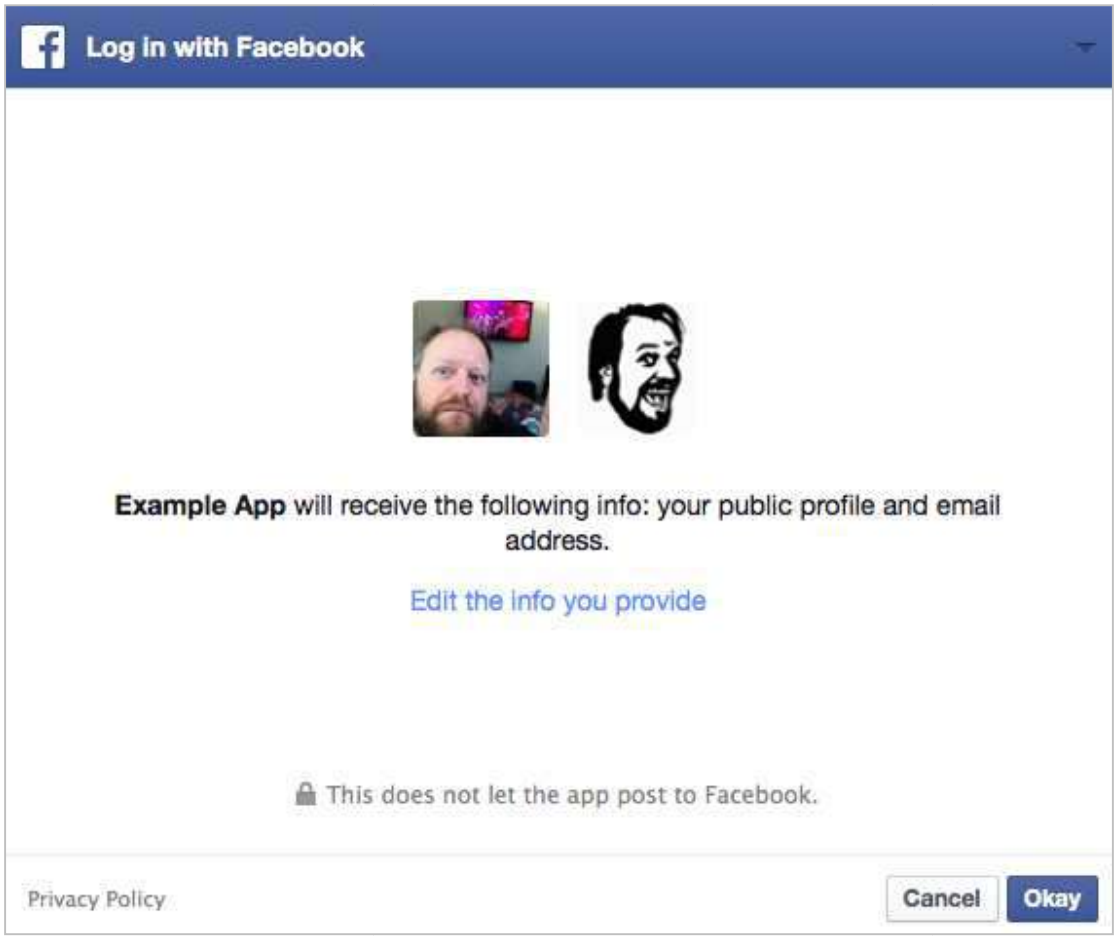

Figura 3: Ejemplo de utilización de *Facebook Login*.

2.1.1.2 *Graph* API

La *Graph* API (gráfico en inglés) fue nombrada de esa manera por representar a través de un "gráfico social" la información que compone a *Facebook* [18]:

- Nodos: entidades como usuarios, páginas, comentarios, videos y fotos.
- Aristas: conexiones entre las entidades, como pueden ser las fotos de un usuario o comentarios de una foto.
- Campos: información sobre las entidades, como por ejemplo el nombre de un usuario o su cumpleaños.

Todos los nodos y aristas pueden ser leídos con una petición HTTP Get a un *endpoint* (*url* de la API) correspondiente. Cabe aclarar que dichas peticiones deben estar "firmadas", en la mayoría de los casos, por un *token* de acceso con permisos suficientes para poder acceder a la información. Por ejemplo, para obtener información del usuario actualmente *logueado* a través de la API y del cual se obtuvo el *token* de acceso se debe hacer una petición HTTP (ver Figura 4).

```
GET /v2.5/me HTTP/1.1
Host: graph.facebook.com
```

Figura 4: Petición HTTP a la *Graph* API de *Facebook*.

También se pueden especificar los campos a obtener en una llamada a la API, permitiendo mejorar la velocidad y la performance al utilizar la plataforma. Por ejemplo, en la Figura 5, se puede observar que la llamada retornará solamente el id, nombre y foto del perfil del usuario.

```
GET graph.facebook.com
 /bgolub?
   fields=id,name,picture
```

Figura 5: Llamada a la *Graph* API utilizando parámetros.

La respuesta a la petición se devuelve en un formato general y varía según el nodo o arista que se esté leyendo (ver Figura 6).

```
{
    "nombre del atributo": {valor del atributo},
    .....
}
```

Figura 6: Ejemplo genérico de respuesta a una petición.

2.1.1.3 *Facebook* PHP SDK

El SDK de *Facebook* para PHP es una librería que permite a desarrolladores integrar *Facebook Login* y realizar llamadas a la *Graph* API. Gracias a esto, facilita operaciones con cuentas de *Facebook* como subir fotos y videos, realizar peticiones por lotes, entre otras cosas. A su vez, es compatible con el SDK de *Javascript* para *Facebook*, lo que permite el uso de ambas librerías convenientemente en un entorno web.

Para presentar un ejemplo *Facebook Login*, podríamos tener en cuenta el siguiente caso: Se inicializa la API con los datos de la *Facebook App* a utilizar (identificador y clave secreta de la aplicación) y luego se obtiene una URL de *logueo* utilizándolos (ver Figura 7). De ser necesario se pueden especificar los permisos a solicitar durante el *logueo* mediante parámetros en la llamada a la API. Este procedimiento se puede implementar en un *script* PHP o en dos, para mayor separación y más control sobre las respuestas (ver Figuras 8 y 9).

```
$fb = new Facebook\Facebook([
  // Facebook App id
  'app_id' => '985231965432290426',
  // App secret
  'app_secret' => 'b58796ejdg687521056c6006c1898e5',
  // Version de Graph API
  'default_graph_version' => 'v2.2',]
);

$helper = $fb->getRedirectLoginHelper();
$loginUrl = $helper->getLoginUrl('https://ejemplo.com/fb-callback.php', ['email']);

echo '<a href="' . htmlspecialchars($loginUrl) . '">Login</a>';
```

Figura 7: Ejemplo sencillo de script implementando Facebook Login.

```
$fb = new Facebook\Facebook([
  'app_id' => '985231965432290426', // Facebook app id
  'app_secret' => 'b58796ejdg687521056c6006c1898e5', // App secret
  'default_graph_version' => 'v2.2',
  ]);

$helper = $fb->getRedirectLoginHelper();

try {
  $accessToken = $helper->getAccessToken();
} catch(Facebook\Exceptions\FacebookResponseException $e) {
  // When Graph returns an error
  echo 'Graph returned an error: ' . $e->getMessage(); exit;
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  // When validation fails or other local issues
  echo 'Facebook SDK returned an error: ' . $e->getMessage();
  exit;
}

if (! isset($accessToken)) {
  if ($helper->getError()) {
    header('HTTP/1.0 401 Unauthorized');
    echo "Error: " . $helper->getError() . "\n";
    echo "Error Code: " . $helper->getErrorCode() . "\n";
} else {
    header('HTTP/1.0 400 Bad Request');
    echo 'Bad request';
  }
  exit;
}
// Logueado
echo '<h3>Token de acceso</h3>';
var_dump($accessToken->getValue());
```

Figura 8: Ejemplo de *script* de PHP que maneja la respuesta (*callback*) del *logueo* en *Facebook* (parte 1).

```
// Manejador de cliente OAuth 2.0 que ayuda a administrar tokens de acceso
$oAuth2Client = $fb->getOAuth2Client();

// Get the access token metadata from /debug_token
$tokenMetadata = $oAuth2Client->debugToken($accessToken);
echo '<h3>Metadatos</h3>';
var_dump($tokenMetadata);

// Validación de la Id de Facebook App
$tokenMetadata->validateAppId('985231965432290426');

// Pregunta si el token es de larga duración
if (! $accessToken->isLongLived()) {
  // Intercambia un token de corta duración por uno de larga duración
  try {
    $accessToken = $oAuth2Client->getLongLivedAccessToken($accessToken);
  } catch (Facebook\Exceptions\FacebookSDKException $e) {
    echo "<p>Error obteniendo token de acceso de larga duración: " . $helper->getMessage() . "</p>\n\n";
    exit;
  }
  echo '<h3>Token de acceso de larga duración</h3>';
  var_dump($accessToken->getValue());
}
// Guardo el token de acceso en sesión
$_SESSION['fb_access_token'] = (string) $accessToken;
```

Figura 9: Ejemplo de *script* de PHP que maneja la respuesta (*callback*) del *logueo* en *Facebook* (parte 2).

En el siguiente ejemplo se muestra cómo obtener datos del perfil del usuario *logueado* utilizando la *Graph* API con el SDK de PHP de *Facebook*: presumiendo que ya se ha obtenido un *token* de acceso mediante el proceso de *logueo*, se realiza la llamada a la *url* "/me" relativa a la API incluyendo el *token* mencionado con anterioridad. Adicionalmente, se pueden especificar los campos a obtener del perfil del usuario como nombre, identificador, *url* de la foto de perfil, entre otros (ver Figura 10).

```
$fb = new Facebook\Facebook([
  'app_id' => '{app-id}',
  'app_secret' => '{app-secret}',
  'default_graph_version' => 'v2.2',
  ]);

try {
  // Returns a `Facebook\FacebookResponse` object
 // Obtengo el id y el nombre del usuario logueado
  $response = $fb->get('/me?fields=id,name', '{access-token}');
} catch(Facebook\Exceptions\FacebookResponseException $e) {
  echo 'Graph returned an error: ' . $e->getMessage();
  exit;
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  echo 'Facebook SDK returned an error: ' . $e->getMessage();
  exit;
}
```

Figura 10: Obtención del identificador y nombre de perfil del usuario de Facebook actualmente *logueado*.

2.1.1.4 *Facebook Javascript* SDK

El SDK de *Facebook* para *Javascript* provee interacción con la API y varias funcionalidades del lado del cliente que permiten, entre otras cosas, crear botones de "me gusta" y similares asociados a la red social, utilizar *Facebook Login* y realizar llamadas a *Graph* API.

Para cargarlo, se realiza una llamada asíncrona que incluirá como parámetro el identificador de la *Facebook App* y el número de versión de la API a utilizar (ver Figura 11).

```
 window.fbAsyncInit = function() { FB.init({ appId : '985231965432290426', xfbml : true, version : 'v2.5' }); };


(function(d, s, id){
    var js, fjs = d.getElementsByTagName(s)[0];


   if (d.getElementById(id)) {return;}


   js = d.createElement(s);
    js.id = id;
    js.src = "//connect.facebook.net/en_US/sdk.js";
    fjs.parentNode.insertBefore(js, fjs);
  }(document, 'script', 'facebook-jssdk'));
```

Figura 11: Ejemplo de inicialización de Api de *Facebook* a través de *Javascript*.

Para implementar *Facebook Login* con el SDK de *Javascript* basta con utilizar, luego de la inicialización expuesta con anterioridad, el método "FB.login(callback, params)". Este método permite especificar parámetros tales como permisos y la posibilidad de indicar qué función se utilizará para el manejo de la respuesta *callback*. Similar es el caso de la utilización de la *Graph* API, ya que basta con usar la función "FB.api(url, callback)", especificando la *url* de API de *Facebook* a utilizar y una función de manejo de respuesta para lograr realizar una petición. En la Figura 12 se muestra una combinación, en la cual se utiliza *Facebook Login*, para luego, de ser exitoso el procedimiento, intentar obtener los datos del perfil del usuario *logueado*, imprimiendo el nombre del mismo en pantalla.

```
FB.login(function(response) {
   if (response.authResponse) {
     FB.api('/me', function(response) {
       console.log('Good to see you, ' + response.name + '.');});
   } else { console.log('User cancelled login or did not fully authorize.');  }
}, {scope: 'manage_pages, read_insights', return_scopes: true});
```

Figura 12: Ejemplo de utilización de Facebook Login y Graph API a través de Javascript.

2.2 Diseño de esquema de bases de datos, sistema de obtención y procesamiento de datos y *front-end* del sistema

Para el presente proyecto se estableció MySQL como motor de base de datos. Esto es debido a que en la empresa Ipaddress S.R.L. se acostumbra a utilizar dicho motor de base de datos. Tanto para el *back-end* como para la parte principal del *front-end* se utilizaron tablas y procedimientos almacenados que están presentes en todos los proyectos de la empresa Ipaddress S.R.L. y, por cuestiones de confidencialidad, no se brindarán detalles al respecto.

En cuanto a la sección específica del sistema, se crearon las tablas *fbaccount* y *twaccount* para guardar la información de cuentas de *Facebook* y *Twitter* asociadas a un usuario (ver Figura 13)

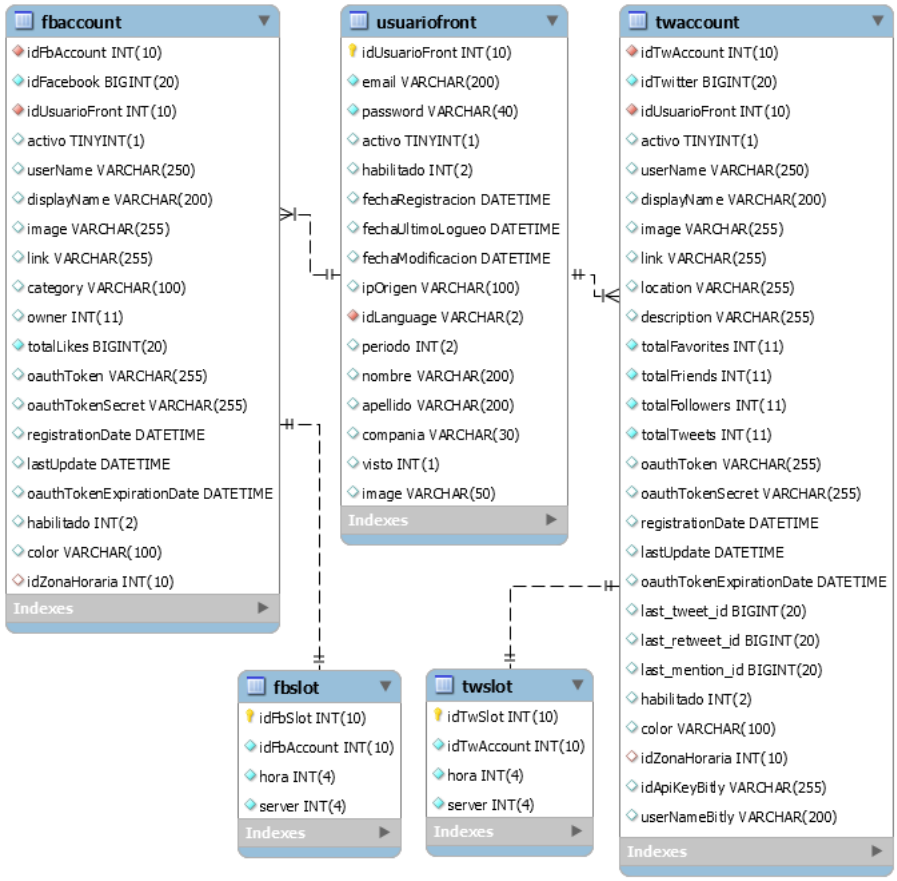

Figura 13: Diagrama de entidad relación: Sección usuario *front-end*.

2.2.1 Base de datos: Facebook API

Para guardar la información procesada de *Facebook* se crea la estructura de tablas que se puede observar en la Figura 14. Vale aclarar que no debe confundirse la tabla *fbaccount* (recordar Figura 13) y la tabla *fbpage* (ver Figura 14). *Fbaccount* guarda la información básica de la cuenta de *Facebook* asociada, así como también los datos del *token* de acceso correspondiente a la misma. En cambio, *fbpage*, guarda la información de una página de *Facebook* asociada a la cuenta de dicha red social. El resto de las tablas presentes en la Figura 14 almacenan información procesada sobre diferentes aspectos y secciones de la página de *Facebook* como por ejemplo los usuarios más importantes que interactúan con la página (tabla *fbpagetopfans*), los datos demográficos agregados de los usuarios de la misma (tabla *fbpagegenderage*), entre otros.

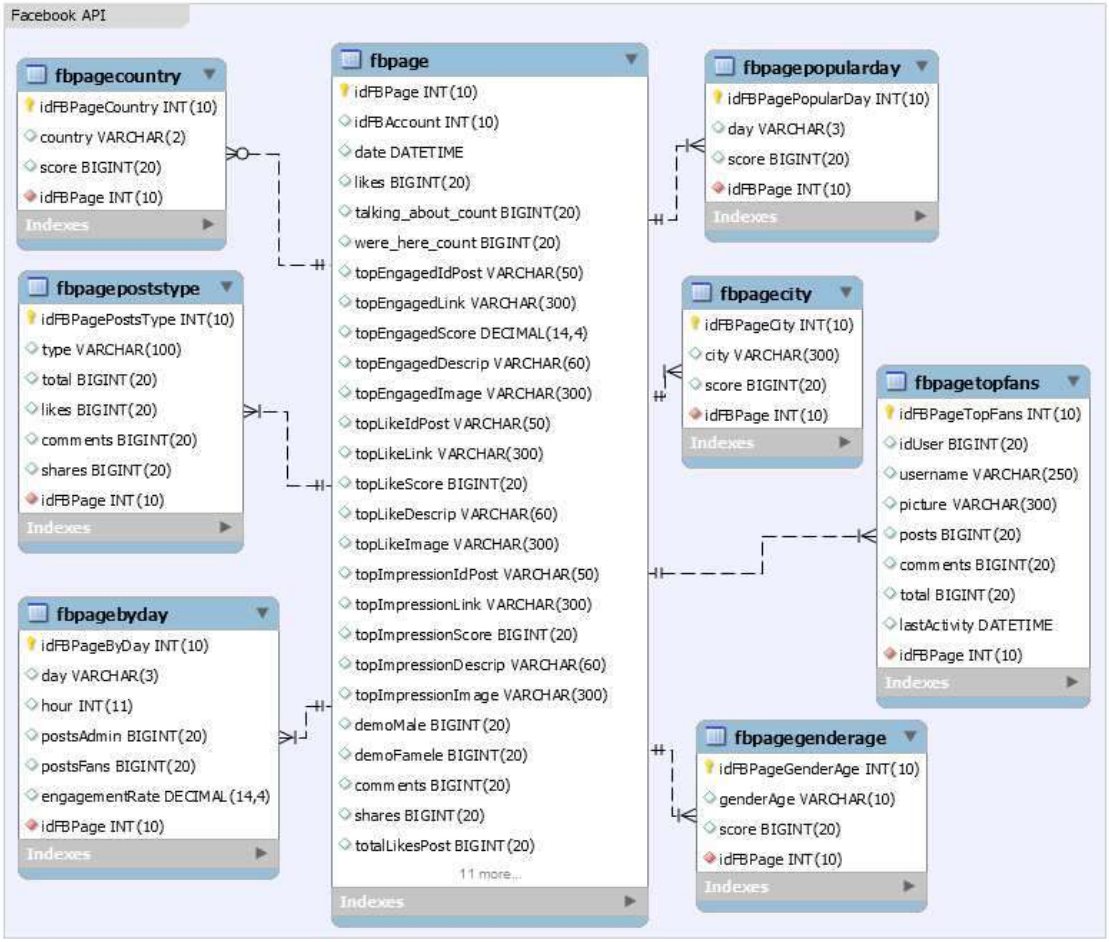

Figura 14: Sección "usuario *front-end*" del diagrama de entidad relación.

2.3 Implementación del esquema de base de datos previamente diseñado

En esta etapa del desarrollo se implementó el diseño de base de datos expuesto en la sección 2.2. Para ello, se crearon las tablas y procedimientos correspondientes utilizando *scripts* SQL mediante la herramienta HeidiSQL. Luego a través de un generador del *framework* de Ipaddress S.R.L., se generaron automáticamente los procedimientos almacenados que permiten que el sistema obtenga y guarde datos en dichas tablas. A pesar de la utilización del generador, existieron algunos errores lógicos y se tuvieron que modificar manualmente varios de los *scripts* auto-generados, aunque, debido al tiempo que se ganó por el propio proceso de "autogeneración", el costo-beneficio fue positivo. En la Figura 15 se puede apreciar un ejemplo de procedimiento almacenado auto-generado el cual permite guardar o actualizar un registro de la tabla *fbpagegenderage*.

```
CREATE PROCEDURE `FBPageGenderAgeSave`(IN `pIdFBPageGenderAge` int(10) unsigned, IN `pGenderAge` varchar(10), IN `pScore` bigint(20), IN `pIdFBPage` int(11))
BEGIN
  IF (EXISTS(SELECT * FROM `FBPageGenderAge` WHERE `FBPageGenderAge`.`idFBPageGenderAge` = pIdFBPageGenderAge)) THEN
    UPDATE `FBPageGenderAge` SET
      `FBPageGenderAge`.`genderAge` = pGenderAge,
      `FBPageGenderAge`.`score` = pScore,
      `FBPageGenderAge`.`idFBPage` = pIdFBPage
    WHERE
      `FBPageGenderAge`.`idFBPageGenderAge` = pIdFBPageGenderAge;
  ELSE
    INSERT INTO `FBPageGenderAge` (`idFBPageGenderAge`, `genderAge`, `score`, `idFBPage`)
    VALUES (pIdFBPageGenderAge, pGenderAge, pScore, pIdFBPage);
  END IF;
END
```

Figura 15: Ejemplo de script SQL auto-generado.

Además de los procedimientos almacenados nombrados anteriormente, mediante el generador del *framework* de Ipaddress S.R.L. se auto-generaron archivos de PHP que, mediante un administrador de conexiones y operaciones sobre base de datos propio del *framework*, permiten conectar con la base de datos, y realizar operaciones sobre una tabla específica. Tales operaciones pueden ser: obtener un registro de una tabla buscándolo según su clave primaria, guardar un registro en la tabla, borrar un registro de la tabla, entre otros. Al igual que en el caso de los procedimientos almacenados, se realizaron modificaciones al código generado con el fin de evitar errores y también, se agregaron métodos nuevos que permiten realizar diferentes operaciones sobre la tabla que no fueron incluidas durante la generación (ver Figura 16).

```
class FBPageGenderAgeDal extends Dal {
  [...]
   public static function save(FBPageGenderAge $entidad)
   {
      $connection = DBManager::getDatabaseConnection();
      $connection->begin();
      $statement = $connection->prepare('CALL `FBPageGenderAgeSave`(:pIdFBPageGenderAge,
            :pGenderAge, :pScore, :pIdFBPage)');

      $statement->bind(':pIdFBPageGenderAge', $entidad->getIdFBPageGenderAge(), Connection::INT);
      $statement->bind(':pGenderAge', $entidad->getGenderAge(), Connection::STRING);
      $statement->bind(':pScore', $entidad->getScore(), Connection::INT);
      $statement->bind(':pIdFBPage', $entidad->getIdFBPage(), Connection::INT);

      $exception = $statement->execute();
      if (!$entidad->getIdFBPageGenderAge())
      {$entidad->setIdFBPageGenderAge($connection->lastInsertedId());}
      $connection->commit();

      return $exception;
   }
}
```

Figura 16: Porción de código PHP auto-generado por el *framework*.

2.4 Desarrollo de subsistema de automatización de obtención, almacenamiento y procesamiento de datos obtenidos en redes sociales

El subsistema de obtención y procesamiento de datos tiene varias funciones. Por un lado, debe obtener datos sobre una cuenta de red social en particular (utilizando *tokens* de acceso y permisos acordes). Para ello se implementó un sistema de colas, en el cual las cuentas a analizar se irán insertando para que el proceso de obtención de datos vaya analizándolas. Los datos resultantes de este último proceso serán almacenados para luego ser procesados. De esta forma se obtiene la información pertinente a mostrar en el sistema que será guardada en la base de datos MySQL.

Al momento de diseñar este subsistema, se destacó la necesidad de que el mismo sea escalable. Esto surge debido a la cantidad de información analizada y almacenada por la aplicación. Es por esto que cada cuenta de *Facebook* analizada tiene asociado un *slot* de servidor que indica en qué horario se realizará el proceso de obtención de datos y qué servidor lo realizará. En un principio, será un solo servidor, pero a medida que la cantidad de usuarios crezca, se podrán agregar nuevos servidores dedicados al análisis y procesamiento de datos con relativa facilidad. Para almacenar los datos obtenidos y albergar el sistema de cola, se pensó en utilizar un sistema de almacenamiento separado de la base de datos MySQL para mejorar el rendimiento de todo el procesamiento de datos. Para tal fin se decidió utilizar un motor de base de datos NoSQL amigable con PHP llamado MongoDB (ver Figura 17).

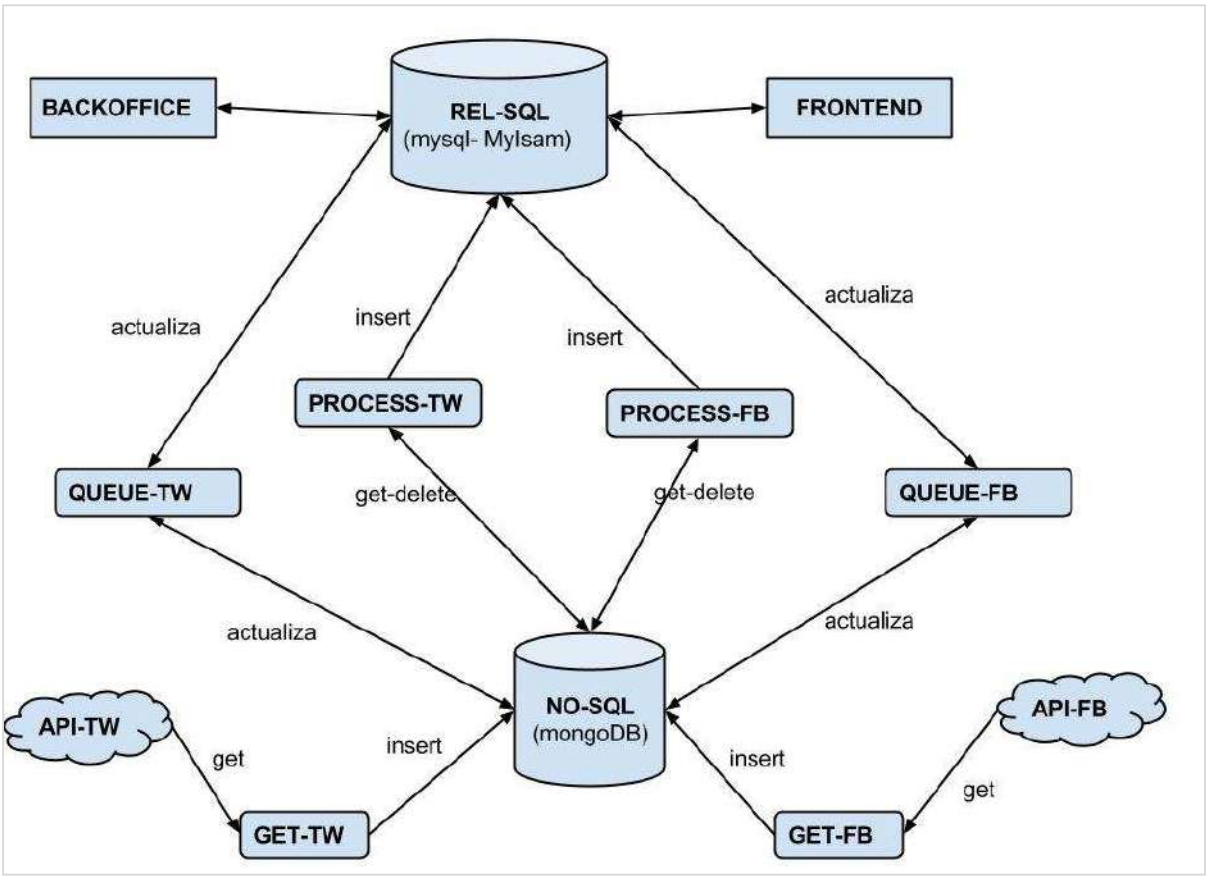

Figura 17: Estructura de subsistema de obtención y procesamiento de datos.

2.4.1 NoSQL y MongoDB

El término NoSQL originalmente se utilizaba para hacer referencia a sistemas de base de datos no relacionales. Ese tipo de base de datos existen desde finales de la década de 1960, pero el término no se acuñó hasta principios de 2009, donde grandes compañías como *Facebook* y *Google* necesitaban sistemas de base de datos no relacionales, especializadas en el manejo de gran cantidad de información. Luego de eso, el movimiento NoSQL empezó a crecer y comenzaron a agregarse nuevas características tales como: soporte para replicación sencilla, consistencia eventual, independencia de esquema, entre otras. Es por esto que hoy en día, si bien el término NoSQL sigue vigente, el actual significado del mismo podría traducirse como "no solamente SQL" [19, 20].

En la actualidad, las bases de datos NoSQL son desarrolladas en respuesta a diferentes necesidades presentes en la construcción de aplicaciones modernas tales como trabajar con aplicaciones dedicadas a crear una cantidad masiva de datos que cambian rápidamente (datos estructurados, semi-estructurados, desestructurados o polimórficos). Las bases de datos relacionales tradicionales, como MySQL, no fueron diseñadas para trabajar con la escala y agilidad con las que deben hacerlo muchas aplicaciones modernas ni tampoco para tomar ventaja del gran poder de proceso y capacidad de almacenamiento que existen hoy en día [21].

Para este proyecto se eligió utilizar un sistema NoSQL para almacenar los datos crudos (sin procesar) obtenidos de las redes sociales, debido al volumen de información que este sistema manejará y las ventajas en agilidad y rendimiento que presentan las bases de datos NoSQL para afrontar dicho problema. Se eligió el sistema de base de datos MongoDB debido a que el mismo puede ser utilizado con el lenguaje PHP ya que existe un conector desarrollado para tal fin. Por lo tanto integrar y utilizar MongoDB en el proyecto sería una tarea relativamente sencilla.

MongoDB es una base de datos NoSQL de código abierto basada en documentos que provee alta performance, alta disponibilidad y escalamiento automático. Para guardar la información se crean base de datos, que contienen colecciones compuestas por documentos [22].

Un registro en MongoDB es denominado documento y es básicamente una estructura de datos compuesta por pares de campos y valores. Los valores de los campos pueden incluir otros documentos, *arrays*, y *arrays* de documentos (ver Figura 18) [23].

```
{
   "_id" : ObjectId("54c955492b7c8eb21818bd09"),
   "nombre" : "María",
   "apellido" : "Pérez",
   "edad" : 30,
   "fecha_ingreso" : "2012-05-01",
   "permisos" : ["usuarios",  "administración"]
}
```

Figura 18: Ejemplo de un documento en MongoDB.

MongoDB guarda los documentos en colecciones. Estas son análogas a las tablas en las base de datos relacionales pero, a diferencia de estas últimas, las colecciones no requieren que los documentos tengan ese mismo esquema, solamente un campo _id único que actúa como clave primaria [22].

En cuanto a las operaciones que se pueden realizar en MongoDB, éstas son similares a las de cualquier base de datos. Algunas de ellas son: Insertar un documento, buscar un documento, actualizar un documento y remover un documento. En la Figura 19, se puede observar la inserción de un documento a modo de ejemplo.

```
# Selecciono base de datos
use mydb;

# Inserto documento en la colección usuarios
db.usuarios.insert (
    "nombre" : "Maria",
    "apellido" : "Perez",
    "edad" : 30,
    "fecha_ingreso" : ISODate("2014-10-01T00:00:00Z"),
    "permisos" : ["usuarios",  "administracion"]
)
```

Figura 19: Ejemplo de inserción en MongoDB.

Para integrar a MongoDB en el proyecto, se utilizó un controlador de MongoDB para PHP. Luego de esto, con instanciar la clase MongoClient incluyendo como parámetros los datos de conexión a la base de datos y ya se puede comenzar a operar sobre ella. En la Figura 20 se puede observar un ejemplo en el cual se instancia el cliente de MongoDB para PHP, luego se crea un documento y se lo inserta en la colección "usuarios", perteneciente a la base de datos "mydb".

```
<?php
  // connect to mongodb
  $m = new MongoClient("mongodb://usuario:password@127.0.0.1: 27017/mydb");
  echo "Conexión exitosa";

  // seleccionar una base de datos
  $db = $m->mydb;
  echo "Base de datos mydb seleccionada";
  $collection = $db->usuarios;
  echo "Collection selected succsessfully";

  $document = array(
    "nombre" => "Maria",
   "apellido" => "Perez",
   "edad" => 30,
   "fecha_ingreso" => date("Y-m-d"),
   "permisos" => ["usuarios",  "administración"]
  );

  $collection->insert($document);
  echo "Documento insertado exitosamente";
?>
```

Figura 20: Ejemplo de utilización de MongoDB en PHP.

2.4.2 Sistema de colas

El sistema de colas diseñado para ordenar la obtención de datos de las redes sociales consta de un proceso, que se ejecuta una vez por hora como tarea programada en el servidor donde se aloja Twoorl. Este obtiene de la base de datos MySQL, las cuentas de red social que deben ser analizadas en esa hora y en ese servidor en particular (para posibilitar escalabilidad). Luego, la información sobre estas cuentas (identificador de cuenta, *token* de acceso, entre otros) se inserta en una colección de MongoDB que hará las veces de cola. En el caso de *Facebook* esa colección se llama *fbqueue* y en *Twitter twqueue*. Por cuestiones de confidencialidad, solo se mostrará una

pequeña porción del código que corresponde a *Facebook* (ver Figura 21).

```
<?php
// Selecciono la colección de la cola para páginas de Facebook
// utilizando una conexión con MongoDB
 $fb_page_posts_queue = $this->connection->selectCollection('twoorl', FacebookHelper::FB_PAGE_ QUEUE);
 $fb_page_posts_queue->ensureIndex('page_id', array('background' => true));

// Proceso todas las cuentas de Facebook
 if(is_array($this->fbAccounts) )
{
   foreach($this->fbAccounts as $fbAccount)
   {
      $page_id = $fbAccount->getIdFacebook();
      $postUrls = $this->getPostUrls($page_id);

      $postDocument = new FbDocument($fbAccount);
      $postDocument->setUrls($postUrls);
      $postDocument->setSince($this->since);
      $postDocument->setUntil($this->until);

      // Inserto documento en la colección
      $r = $fb_page_posts_queue->insert($postDocument);
   }
}
?>
```

Figura 21: Otro ejemplo de utilización de MongoDB en PHP.

2.4.3 Obtención y procesamiento de datos

La obtención de datos en *Facebook* y *Twitter* se realiza en dos procesos separados ejecutados como tareas programadas. Estos leen las cuentas insertadas en la cola y secuencialmente realizan peticiones a la API correspondiente. La información obtenida se guarda en una colección de

MongoDB donde se alojan todos los datos sin procesar y, posteriormente se borran las cuentas analizadas de la cola (recordar Figura 17).

El procesamiento de datos es realizado mediante procesos separados para cada red social. Al igual que la obtención de datos, dichos procesos se ejecutan como tareas programadas en el servidor de la aplicación. Los datos recolectados de la red social son leídos desde de la colección de MongoDB correspondiente y se calculan las estadísticas deseadas. Finalmente el resultado de este proceso es guardado en tablas de la base de datos de MySQL en tanto que los datos utilizados de la colección de MongoDB son eliminados de la misma.

2.5 Desarrollo del *front-end* de la aplicación web

El *front-end* fue desarrollado utilizando el *framework* propio de la empresa Ipaddress S.R.L. que provee, entre otras cosas, soluciones sencillas para operaciones cotidianas de aplicaciones web como *login*, registración de nuevos usuarios, listados y pantallas de ABM (alta, baja y modificación), lo cual agilizó en gran medida el proceso de desarrollo. El sistema posee una página maestra que contiene texto, controles estáticos y carga las librerías que aparecen en todo el sistema. Dentro de esta página maestra se cargan las diferentes secciones del sitio. También se utiliza AJAX para servir contenido y realizar operaciones asíncronamente.

El diseño gráfico del sistema fue tercerizado y, como resultado de dicho proceso, fueron proporcionados por el cliente archivos en formato PSD de *Adobe Photoshop* con las plantillas correspondientes a las diferentes pantallas del sistema. Estas debieron ser integradas al proyecto y maquetadas (extrayendo gráficos y desarrollando archivos de estilo en cascada CSS).

2.5.1 Registración de usuarios y acceso al sistema

El registro de usuarios se desarrolló con una pantalla sencilla en la cual se ingresan datos básicos del usuario a registrar y, luego de validaciones tales como el correcto formato del correo electrónico ingresado, se procede a enviar un correo a la dirección ingresada, el cual contiene un enlace al sitio que permitirá validar la cuenta recientemente creada. Para el acceso al sitio se desarrolló la pantalla de *logueo* en la cual se utiliza un correo electrónico como "nombre de usuario" y se solicita la contraseña ingresada por primera vez en el proceso de registro (ver Figura 22). Como medida de seguridad se implementó la librería recaptcha, para evitar ataques por fuerza bruta [23].

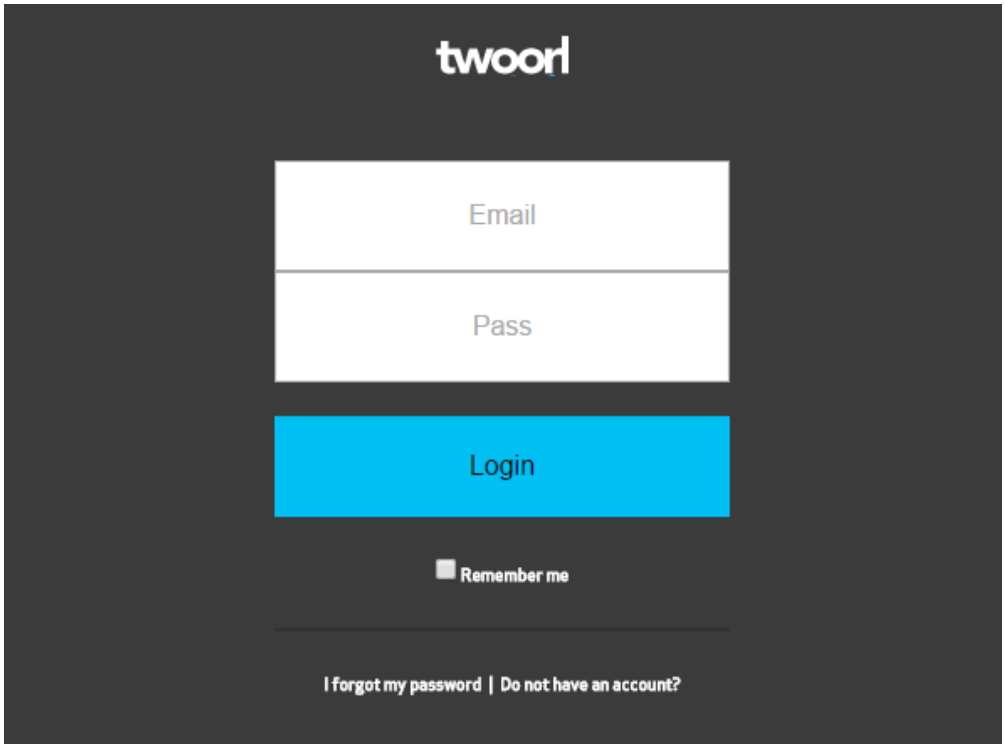

Figura 22: Captura de pantalla del login de Twoorl.

2.5.2 Listado y administración de cuentas asociadas

El menú principal del sistema es accedido inmediatamente luego de completar el proceso de *logueo, y* está compuesto por varias solapas con diferentes secciones (ver Figura 23). La primera y segunda corresponden a listados de cuentas de *Facebook* y *Twitter* (respectivamente) asociadas al usuario *logueado.* Desde dichos listados se puede acceder a la página de análisis de cuenta, administrar las mismas o asociar una nueva. La tercer solapa permite agregar colaboradores que podrán acceder a la cuenta de usuario actual y acceder a las estadísticas de redes sociales obtenidas. Finalmente, la última solapa llamada "*My Account*", está destinada a modificar la información de perfil del usuario *logueado* tal como su nombre, apellido, foto de perfil, entre otros.

Figura 23: Captura de pantalla menú principal en Twoorl.

2.5.3 Asociación de cuenta de red social

Como fue descrito anteriormente, desde el menú principal se pueden asociar nuevas cuentas de redes sociales al usuario actual. Al iniciar dicho proceso, la red social elegida, ya sea *Facebook* o *Twitter*, solicitará que inicie sesión (ver Figura 24). Posteriormente, mostrará los permisos que la aplicación Twoorl pretende pedir al usuario para obtener los datos estadísticos pertinentes. En el caso de *Facebook*, de aceptarse dicha solicitud, se mostrará un listado con las páginas que la cuenta de *Facebook* posee, pudiendo estas asociarse a la cuenta de usuario de Twoorl (ver Figura 25).

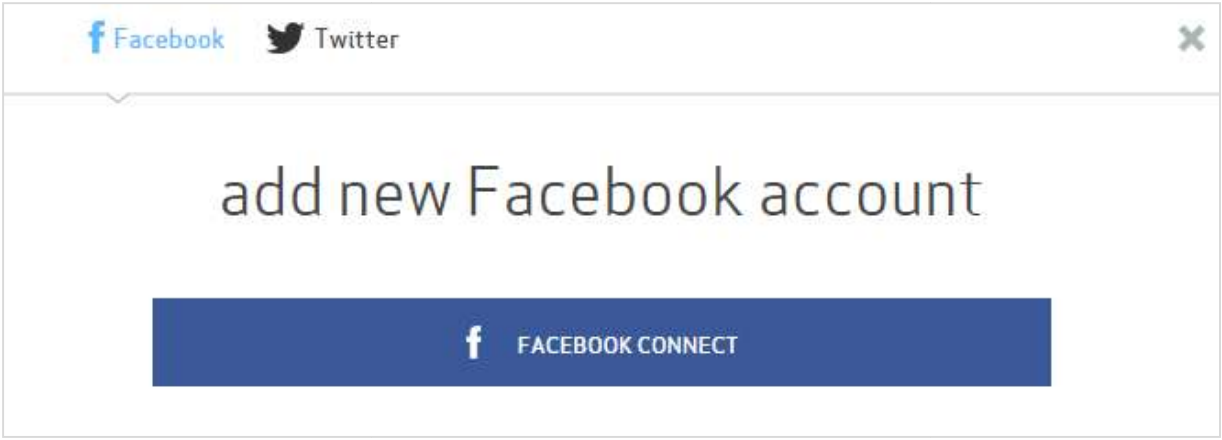

Figura 24: Captura de pantalla conexión con *Facebook* para agregar cuenta en Twoorl.

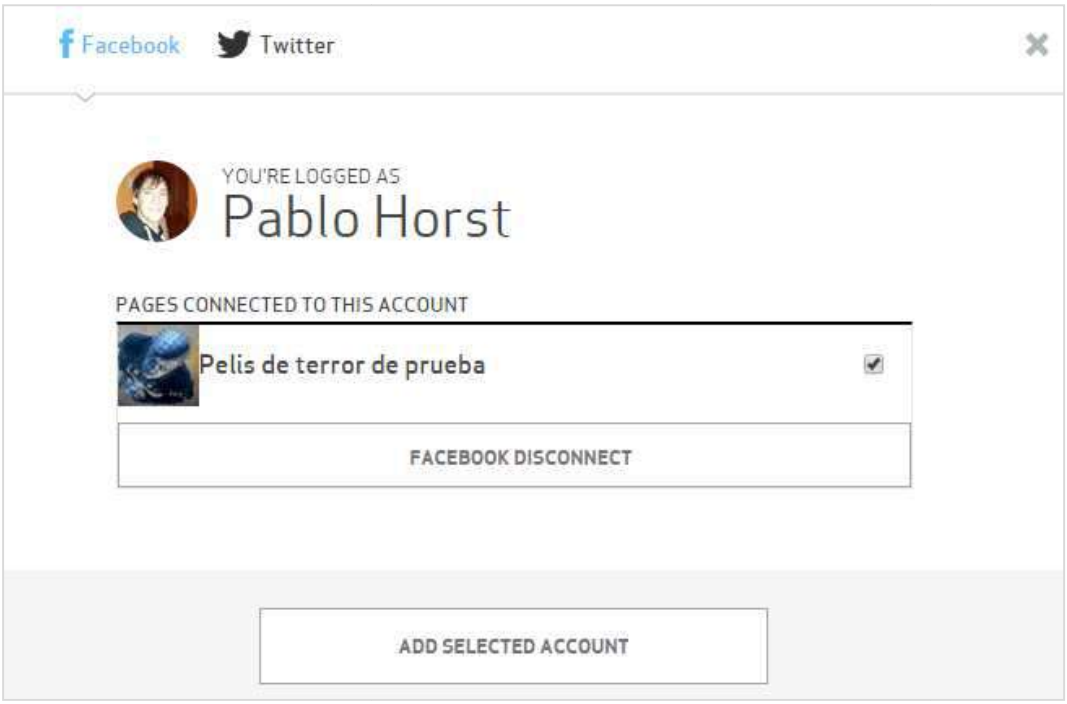

Figura 25: Captura de pantalla para agregar cuenta de *Facebook* en Twoorl.

2.5.4 Estadísticas de cuenta de red social

Al ingresar a la página de estadísticas de una cuenta de red social, se puede apreciar que la información está separada en diferentes secciones que varían según si la cuenta es de *Facebook* o *Twitter*. En ellas se pueden observar las diferentes estadísticas en diversos formatos gráficos (ver figs. 26 y 27). Dicha información se construye a partir de lo almacenado en la base de datos MySQL como fue explicado en el punto 2.4 del presente documento y es filtrada por una fecha inicial y una fecha final indicada en la sección superior de la página.

Figura 26: Captura de pantalla listado de cuentas de *Facebook* en Twoorl.

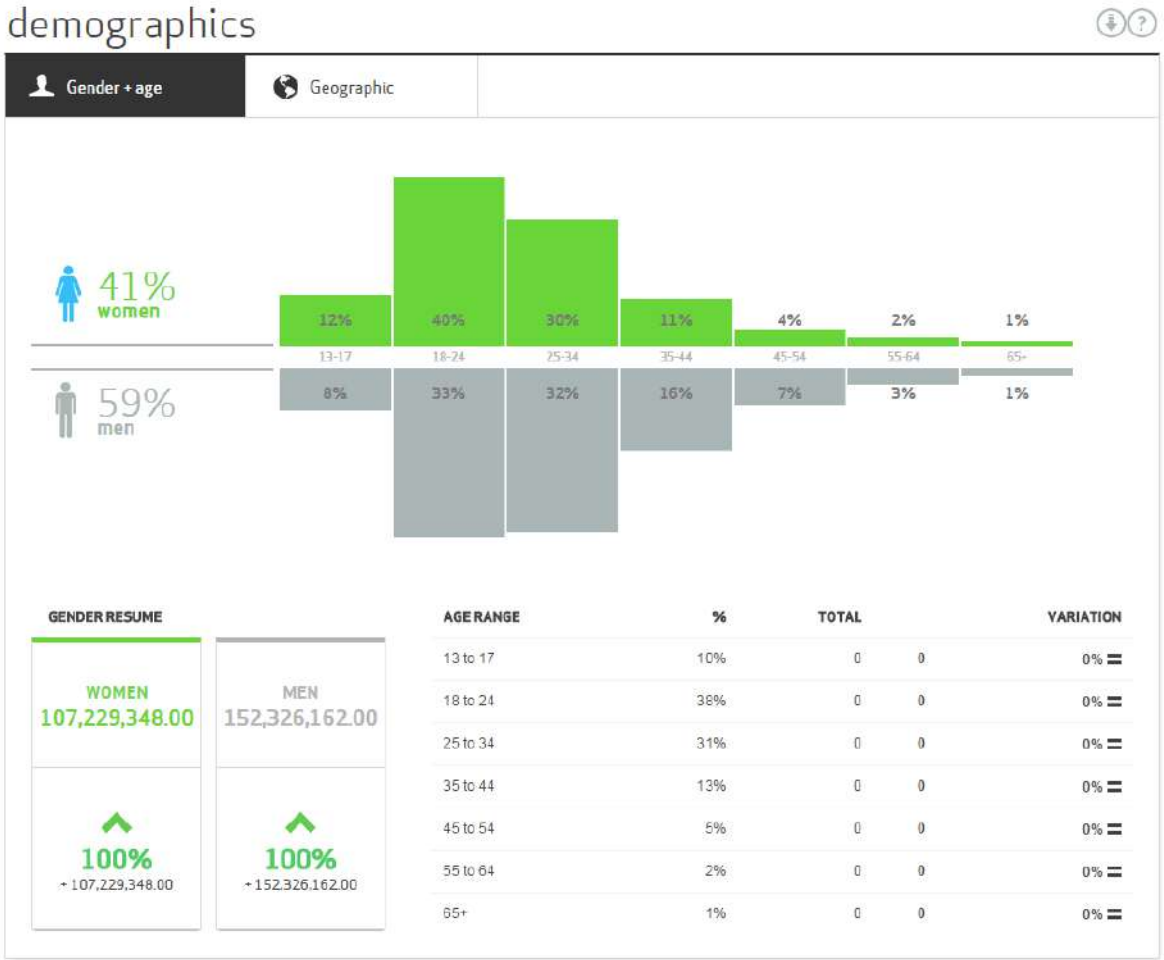

Figura 27: Captura de pantalla de estadísticas demográficas en cuenta de *Facebook*.

3. Conclusiones

El sistema de procesamiento de estadísticas de redes sociales "Twoorl" permite obtener estadísticas que brindan información útil sobre una cuenta de red social. El proyecto fue desarrollado en el marco de la empresa Ipaddress S.R.L., la cual cuenta con un departamento de desarrollo de software que se especializa en la creación de aplicaciones web a medida. Para la obtención de los datos se realizó una investigación sobre las APIs de *Facebook* y *Twitter*, lo que permitió la posterior utilización de las mismas. Para el proceso de desarrollo, se usó un *framework* creado por Ipaddress S.R.L. y un diseño gráfico tercerizado, junto con diferentes tecnologías web.

3.1 Lecciones aprendidas

Como resultado de mi participación en este proyecto, pude adquirir nuevos conocimientos. Por un lado, aprendí a utilizar APIs en general y, en particular, las de *Facebook* y *Twitter*. También logré conocer el movimiento NoSQL aprendiendo a utilizar MongoDB como base de datos NoSQL. A su vez, vale agregar, pude consolidar mis conocimientos de programación web en PHP y *Javascript*, así como también, la creación y administración de una base de datos MySQL.

Durante el desarrollo, el hecho de contar con un *framework* facilitó en gran medida la tarea de programación. Esto se debe a que permite reducir mucho el tiempo de desarrollo al contener gran cantidad de código reutilizable y, además, posibilita realizar tareas complejas con pocas líneas de código.

También hubo inconvenientes durante este proceso, debido que a la mitad del mismo hubo un cambio de versión de la API de *Facebook*, lo que ocasionó que se deba volver a investigar y realizar los cambios pertinentes al código. Otro inconveniente estuvo dado con la comunicación con el cliente. Si bien las demostraciones y conferencias por Skype estaban pautadas de antemano, muchas veces el cliente las pospuso, postergando las entregas y la retroalimentación que surge de las mismas.

Como conclusión, puedo decir que en el proceso de desarrollo se confeccionó muy poca documentación. Se crearon documentos durante etapas previas pero, durante el mismo, dicho proceso prácticamente no existió. Esta situación creó algunos problemas, demostrando que si bien el exceso de "burocracia" y documentación en el proceso puede llegar a ser perjudicial, el otro extremo también lo es.

3.2 Trabajos a futuro

Existen trabajos a realizar a futuro sobre Twoorl. Se prevé realizar pruebas de estrés, de carga y concurrencia sobre el sistema de obtención y procesamiento de datos de las redes sociales para determinar la capacidad y velocidad de respuesta. Esto permitirá conocer los límites bajo los cuales el sistema se comportará de manera apropiada y así saber cuándo se necesitará ajustar la configuración del servidor o realizar balanceo de carga. A su vez se tiene planeado a futuro mejorar las peticiones a las APIs de las redes sociales con el fin de lograr mejor rendimiento.

Bibliografía

[1] Stelzner, Michael E. (2014). *2014 Social Media Marketing Industry Report*. Publicado por Social Media Examiner.
Enlace: http://www.socialmediaexaminer.com/SocialMediaMarketingIndustryReport2014.pdf. Última fecha de acceso: 30 de mayo de 2016.

[2] Aspectos básicos de la medición de Facebook. Publicado por Facebook.com. (2015).
Enlace: https://www.facebook.com/business/learn/facebook-ads-measuring-results/. Última fecha de acceso: 30 de mayo de 2016.

[3] Benson, Buster. (2014) Mide el impacto de tus tweets con Twitter Analytics. Publicado en blog oficial de Twitter.
Enlace: https://blog.twitter.com/es/2014/mide-el-impacto-de-tus-tweets-con-twitter-analytics/. Última fecha de acceso: 10 de enero de 2016.

[4] Trygve Reenskaug y James Coplien. (20 de Marzo de 2009). The DCI Architecture: A New Vision of Object-Oriented Programming.
Enlace: http://www.artima.com/articles/dci_vision.html. Última fecha de acceso: 5 de enero de 2016.

[5] Best MVC Practices
Enlace: http://www.yiiframework.com/doc/guide/1.1/en/basics.best-practices/. Última fecha de acceso: 5 de enero de 2016.

[6] Zend Studio - IDE de desarrollo de Software
Enlace: http://www.zend.com/en/products/studio/. Última fecha de acceso: 5 de enero de 2016.

[7] HeidiSQL - Software de administración de base de datos
Enlace: http://www.heidisql.com/. Última fecha de acceso: 5 de enero de 2016.

[8] Tortoise SVN - Software para utilización de SVN como método de control de versiones
Enlace: https://tortoisesvn.net/. Última fecha de acceso: 5 de enero de 2016.

[9] Apache Web Server
Enlace: https://httpd.apache.org/. Última fecha de acceso: 5 de enero de 2016.

[10] MongoDB - Sistema de base de datos de tipo noSQL
Enlace: https://www.mongodb.org/. Última fecha de acceso: 7 de febrero de 2016.

[11] MySQL Workbench - Software de administración de base de datos
Enlace: https://www.mysql.com/products/workbench/. Última fecha de acceso: 5 de enero de 2016.

[12] Facebook for Developers
Enlace: https://developers.facebook.com/. Última fecha de acceso: 6 de febrero de 2016.

[13] Twitter Developers
Enlace: https://dev.twitter.com/. Última fecha de acceso: 7 de febrero de 2016.

[14] Facebook for Developers - Facebook App Management
Enlace: https://developers.facebook.com/apps. Última fecha de acceso: 6 de febrero de 2016.

[15] Atlas API for Facebook - Overview
Enlace: https://developers.facebook.com/docs/atlas-api/reference/overview/v2.5. Última fecha de acceso: 6 de febrero de 2016.

[16] Marketing API for Facebook
Enlace: https://developers.facebook.com/docs/marketing-apis?locale=es_LA . Última fecha de acceso: 6 de febrero de 2016.

[17] Facebook Login - Access Tokens
Enlace: https://developers.facebook.com/docs/facebook-login/access-tokens/. Última fecha de acceso: 6 de febrero de 2016.

[18] Graph API for Facebook - Overview
Enlace: https://developers.facebook.com/docs/graph-api/overview/. Última fecha de acceso: 6 de febrero de 2016.

[19] Leavitt, Neal. (2010). Will NoSQL Databases Live Up to Their Promise? IEEE Computer.
Enlace: http://www.leavcom.com/pdf/NoSQL.pdf. Última fecha de acceso: 30 de enero de 2016.

[20] No SQL Database
Enlace: http://nosql-database.org/. Última fecha de acceso: 30 de enero de 2016.

[21] NoSQL Explained. MongoDB.
Enlace: https://www.mongodb.com/nosql-explained. Última fecha de acceso: 30 de enero de 2016.

[22] Introduction to MongoDB. MongoDB
Enlace: https://docs.mongodb.org/manual/core/introduction/. Última fecha de acceso: 30 de enero de 2016.

[23] Google Recaptcha. Google.
Enlace: https://www.google.com/recaptcha/intro/index.html. Última fecha de acceso: 5 de enero de 2016.