GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2015 / 2016

TÍTULO DEL TFG

Twitter Finder

nombre del documento

MEMORIA TFG

DATOS DE LA ALUMNA O DEL ALUMNO	**DATOS DEL DIRECTOR O DE LA DIRECTORA**
NOMBRE Lander	NOMBRE Juan Antonio
APELLIDOS Gil Blázquez	APELLIDOS Pereira Varela
DNI	DEPARTAMENTO Lenguajes y Sistemas Infórmaticos
FDO.:	FDO.:
FECHA:	FECHA:

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

¿Quién no ha tenido nunca una queja o problema que consultar? Ya sea sobre la compañía eléctrica, tráfico, compañía telefónica, falta de cobertura, etc. Como por ejemplo llevar días sin internet. Casi todo el mundo ha tenido algún conflicto de este tipo y nos gustaría saber si somos los únicos a los que nos ocurre o si hay más gente en nuestra zona a la que le está pasando lo mismo.

¿Y si Twitter puede facilitarnos esta información? Cada vez hay más gente que usa las redes sociales y Twitter es una de las que más utilizadas, situándose en el quinto puesto en el ranking mundial.

Ilustración 1. Ranking redes sociales

http://www.multiplicalia.com/redes-sociales-mas-usadas-en-2016/

Twitter cuenta con más de 500 millones de usuarios de los cuales 332 millones están activos en 2016, generando 65 millones de tweets al día y manejando más de 800.000 peticiones de búsqueda diarias[1]. Solo en España hay más de 1,4 millones de usuarios activos en Twitter[2].

[1] *https://es.wikipedia.org/wiki/Twitter*
[2] *http://mediaesfera.com/estudio-sobre-los-usuarios-de-facebook-y-twitter-en-espana*

La gran cantidad de información que se maneja en ésta red social puede ayudarnos con nuestros problemas cotidianos. De eso trata este proyecto, es una aplicación web llamada Twitter Finder que nos ayuda a averiguar a través de tweets si hay más gente cerca con el mismo conflicto.

Siguiendo con el ejemplo que se comenta en el primer párrafo (llevar varios días sin internet), esta aplicación nos ayudaría a saber si cerca de nuestra ubicación habría más personas que están sin internet. Ayudándonos a conocer mejor el problema, pudiendo saber si sería un problema global (siendo la compañía telefónica la que no funciona correctamente en la zona. Ilustración 2) o solo nos pasa a nosotros (fallo en la instalación de casa por ejemplo).

Ilustración 2. Queja "sin internet"

Además de poder ayudar con las "quejas" Twitter Finder también puede tener otras aplicaciones, como por ejemplo para enterarnos de eventos como fiestas, conciertos, etc (Ilustración 3).

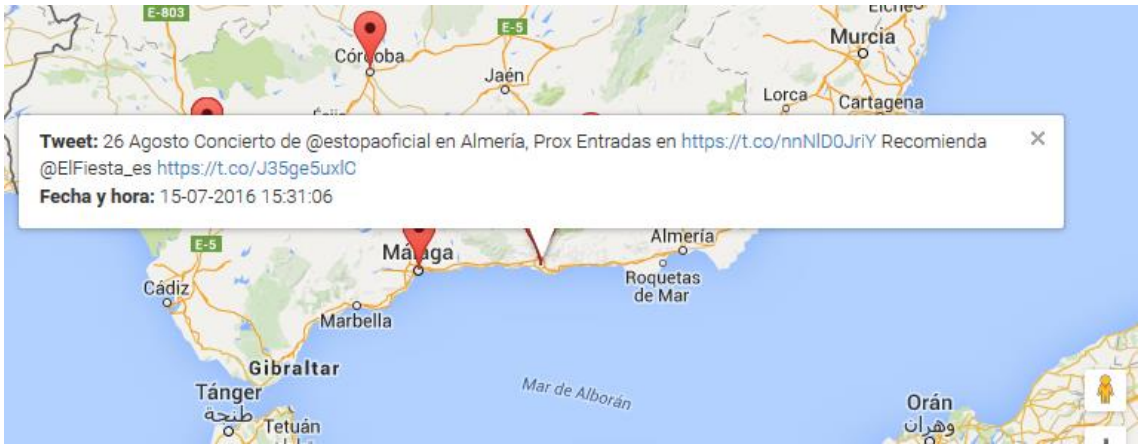


Ilustración 3. Búsqueda "concierto"

1.1. Origen del proyecto

La idea de la realización de este proyecto es de mi director de proyecto y profesor de la asignatura DAWE (Desarrollo de Aplicaciones Web Enriquecidas) Juan Antonio Pereira.

La asignatura DAWE, en la que aprendí a desarrollar aplicaciones web utilizando los lenguajes HTML, CSS y JavaScript, fue la que más me gustó y motivó de todas las asignaturas impartidas durante la carrera. El problema era, ¿qué aplicación web desarrollar? Por lo que al encontrarme sin ideas para el TFG (Trabajo de Fin de Grado) fui a consultar a Juanan para ver si podía orientarme. En cuanto me propuso el tema de este proyecto me llamó la atención y acepte sin dudarlo.

1.2. Motivaciones para la elección del proyecto

La primera motivación para la realización de este proyecto ha sido ver como los conocimientos que he adquirido durante la carrera me permiten hacer correctamente este trabajo.

La segunda motivación y la que considero más importante, es que en un futuro me gustaría dedicarme al diseño y desarrollo de páginas/aplicaciones web. Por lo que consideré que elegir este proyecto me ayudaría a crecer profesionalmente en este campo y podría encontrar un trabajo que antes veía más complicado.

La tercera motivación es la de tratar de ayudar a los usuarios de mi web a encontrar información o respuestas sobre quejas, problemas, eventos, etc.

Por último, el uso de lenguajes diferentes a los estudiados y manejados durante la carrera (como PHP), así como los nuevos conocimientos adquiridos (como la utilización de la API de Twitter) en el desarrollo del TFG han sido una motivación más.

2. Planteamiento inicial

2.1. Descripción

La aplicación web a desarrollar se llama Twitter Finder.

Se trata de una página web en la que cabe destacar como partes más importantes un buscador y un mapa.

El usuario podrá hacer búsquedas de una o varias palabras clave a través del buscador. Con la búsqueda realizada y con la ayuda de la API de Twitter, se obtendrán los últimos tweets escritos que contengan el texto de la búsqueda, almacenándolos en la base de datos.

Como la API de Twitter limita las peticiones diarias, la aplicación web no pide tweets a Twitter con cada búsqueda, sino que implementa un sistema de caché. Lo que se hace es pedir tweets la primera vez que se busca un texto concreto en el día de su búsqueda o cada vez que se busque diez veces en ese mismo día, el resto de las veces simplemente se cogerá la información que se ha guardado en la base de datos. Aparte, cada una hora se actualizará automáticamente la base de datos con las búsquedas que se han hecho ese día, pidiendo nuevamente a Twitter más tweets.

Con los tweets almacenados en la base de datos, se pintará en el mapa un marcador con cada uno de ellos. Esto se consigue utilizando la API de Google Maps; con la geolocalización de cada tweet pintamos un marcador en el mapa. En caso de que el tweet no tenga geolocalización, se usa la población en la que fue registrado el usuario que ha escrito el tweet. Cada marcador contendrá el texto del tweet, la hora y fecha en la que fueron publicados, además de un enlace al tweet original.

En el caso de tener que utilizar la ubicación de registro del usuario para ubicar el marcador en el mapa, se irá almacenando dicha ubicación con sus coordenadas en la base de datos. Esto se hace porque con cada ubicación se pide su latitud y longitud a Google Maps mediante su API. Por lo tanto, para no hacer muchas peticiones de ubicación y ralentizar así la velocidad de respuesta de la aplicación web, se usarán los datos almacenados en la base de datos, usando de nuevo un sistema de caché. Otro detalle a destacar cuando se produce este caso, es que las coordenadas de la ubicación de la localidad se moverán un poco al dibujarlas en el mapa, ya que si todos tienen la misma latitud y longitud solo se visualizará el último marcador en ser dibujado, esta técnica se conoce como jittering.

Ilustración 4. Ejemplo técnica jittering

Aparte de la información de los tweets, las ubicaciones de cada tweet y las búsquedas, también habrá una lista de quejas en la base de datos que serán insertadas por el administrador de la aplicación en base al criterio del mismo. Esta lista irá creciendo a medida que el administrador vea que palabras o frases usan los usuarios para quejarse.

Para la realización de la búsqueda se podrán aplicar distintos filtros. Uno de ellos es un rango temporal, con esta opción se consigue no mostrar los tweets más antiguos que el rango seleccionado, dependiendo de la antigüedad del tweet su marcador se verá de un color diferente. Otro filtro es el que nos permite ver el mapa de España o seleccionar una región de éste. Por último, como opción avanzada existe la búsqueda por nombre de usuario, si marcamos esta opción veremos los tweets escritos por un usuario en concreto.

Aparte de la funcionalidad principal descrita antes, la página web ofrece otras funcionalidades, como ver tu búsqueda en la página oficial de Twitter, ver la cantidad de tweets encontrados al realizar una consulta, búsquedas recomendadas para orientar al usuario mostrando distintas listas con enlaces a cada búsqueda, una guía con los distintos tipos de consulta y una sección que explica que es Twitter Finder.

En el apartado de búsquedas recomendadas se usará la lista de quejas que se menciona anteriormente, apareciendo las quejas que los usuarios han escrito ese día.

2.2. Objetivos

El objetivo principal es desarrollar una aplicación web, para comprobar mi capacidad de llevar a cabo un proyecto de principio a fin.

Por otra parte, también pretendo ganar mayor experiencia en los lenguajes HTML, CSS, JavaScript y PHP, familiarizarme con las APIs de Twitter y Google Maps, siendo la primera totalmente nueva para mí, y manejar información con una base de datos MySQL para utilizarla en una aplicación web.

De cara al usuario, lo que se pretende con esta aplicación web, es intentar ayudar en las posibles quejas o cualquier otro asunto que puedan surgirles, aportando información mediante tweets, ubicando cada tweet en un mapa en el punto donde se escribió para conocer la ubicación de la consulta.

2.3. Arquitectura

Se utilizará una arquitectura cliente/servidor. El usuario a través de la aplicación web ejecutará sentencias contra la base de datos mediante llamadas a ficheros PHP alojados en el servidor (Ilustración 5).

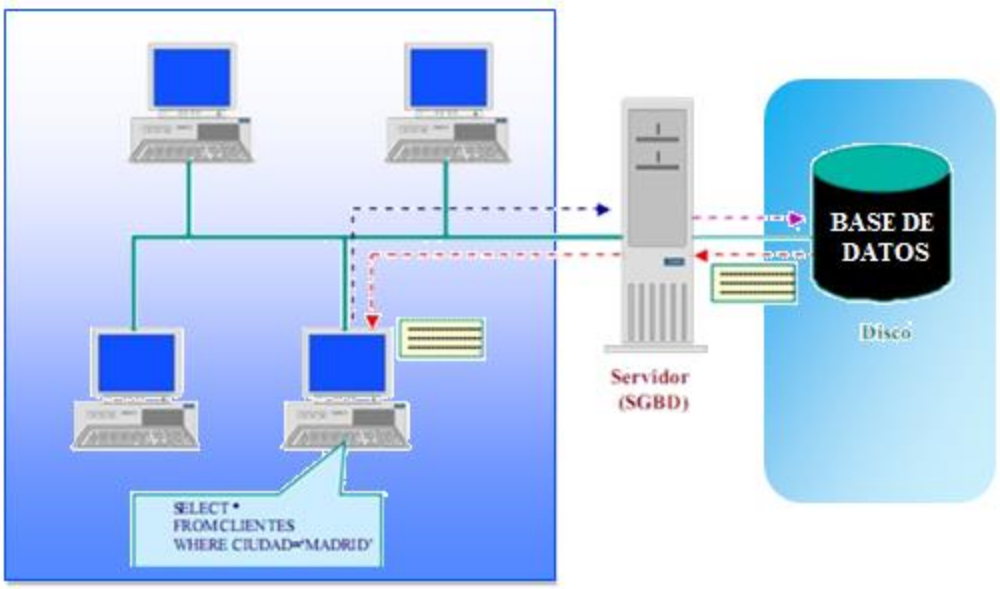


Ilustración 5. Arquitectura

https://commons.wikimedia.org/wiki/File:Arquitectura_Cliente_Servidor.png

2.4. Herramientas

En esta sección se nombraran todas las herramientas de software y de hardware que se utilizan para el desarrollo del proyecto.

Herramientas hardware:

- Ordenador personal Asus con sistema operativo Windows 8.

Herramientas software:

- Wamp

  Es un entorno de desarrollo web que nos va a permitir tener nuestro propio servidor o host local instalado en nuestro ordenador. Sus principales características son:

  - Manejo de Bases de datos con MySQL
  - Software para servidor web Apache
  - Software para poder programar script con PHP
  - Permite el manejo sencillo de Bases de Datos con PHPMyAdmin y SQLiteManager
  - Es software libre y completamente gratuito.

- Microsoft Office 2010

  Paquete de programas informáticos para oficina. Son una serie de aplicaciones que sirven para realizar tareas ofimáticas. En este caso para el desarrollo de la memoria, creación de tablas y elaboración de la presentación.

- MySQL Workbench

  Es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

- Microsoft Azure

  Microsoft Azure es una plataforma general que tiene diferentes servicios para aplicaciones, desde servicios que alojan aplicaciones en alguno de los centros de procesamiento de datos de Microsoft para que

se ejecute sobre su infraestructura (Cloud Computing), hasta servicios de comunicación segura y federación entre aplicaciones. Se usará para alojar la aplicación web y la base de datos en la nube.

- Adobe Photoshop

  Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente "taller de fotos". Se ha usado para la edición de distintas imágenes de la aplicación web.

- Dropbox

  Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y con tabletas y móviles. Se usará para alojar tanto el proyecto como la memoria para así tener una copia de seguridad.

- Filezilla

  Es un cliente FTP multiplataforma de código abierto y software libre, licenciado bajo la Licencia Pública General de GNU. Soporta los protocolos FTP, SFTP y FTP sobre SSL/TLS (FTPS).

- Cacoo

  Es una herramienta online para hacer diagramas de diversos tipos, desde el esquema de una oficina hasta diagramas UML, pasando por el prototipado de pantallas.

- Google Chrome

  Navegador web desarrollado por Google. Se usará este navegador para comprobar si la web se ve correctamente en él.

- Mozilla Firefox

Navegador web libre, de código abierto y multiplatafoma desarrollado por la fundación Mozilla. Se usará este navegador para comprobar si la web se ve correctamente en él.

- Internet explorer

Es un navegador web desarrollado por Microsoft para el sistema operativo Microsoft Windows. Se usará este navegador para comprobar si la web se ve correctamente en él.

- Notepad++

Es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. De soporte nativo a Microsoft Windows. Se usará para editar ficheros HTML, CSS, PHP y JavaScript.

- Twitter Bootstrap

Es un framework o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML y CSS, así como extensiones JavaScript para opciones adicionales.

- Gantt Project

Es una herramienta gratuita que sirve para crear la planificación de los proyectos. Se usará para crear la gráfica de la planificación temporal.

2.5. Alcance

Para la realización del proyecto se ha elegido modelo en cascada simple, en el que hasta no finalizar una tarea no se comienza con la siguiente. Excepto en la fase de implementación, en la cual se utilizará un modelo iterativo, este modelo es una derivación del ciclo de vida en cascada, que busca reducir el riesgo que surge entre las necesidades del usuario y el producto final. Consiste en la iteración de varios ciclos de vida en cascada. Al final de cada iteración se consigue una versión mejorada con más funcionalidades que la anterior. Al final de cada iteración, el desarrollador evaluará el producto y en caso de ser necesario, ejecutará modificaciones o mejoras en él. Este proceso se repetirá hasta obtener un producto que satisfaga las expectativas que se quieren conseguir.

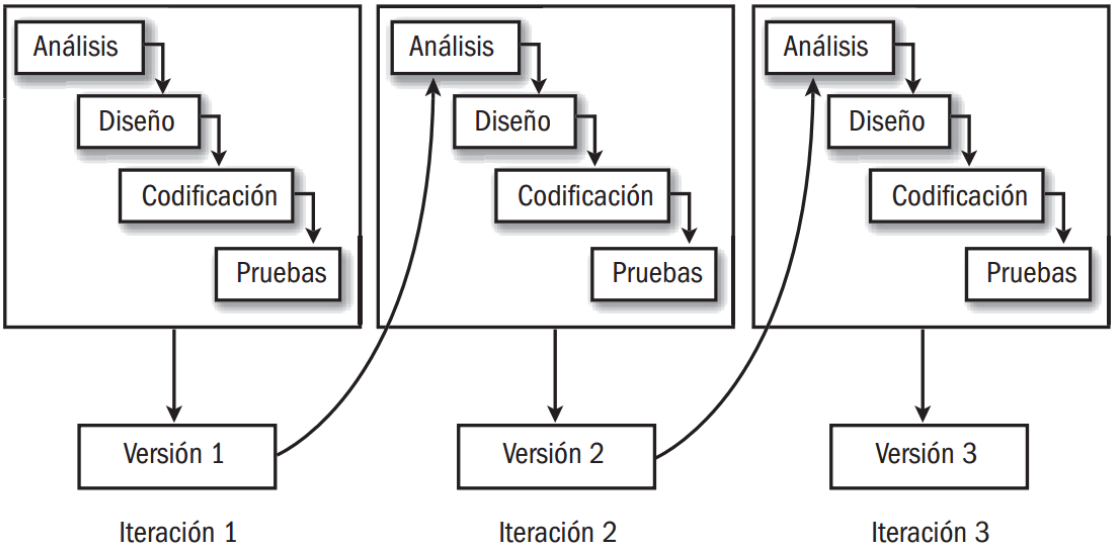


Ilustración 6. Modelo de ciclo de vida iterativo

http://sings-ufps.blogspot.com.es/2012/04/ciclo-de-vida-conceptos.html

A continuación se explicarán las distintas etapas del proyecto, que son representadas en la Estructura de Descomposición del Trabajo (EDT) y se dividen en seis partes:

1. **Gestión**: Búsqueda de información sobre las tecnologías que se van a utilizar y planificación del proyecto.

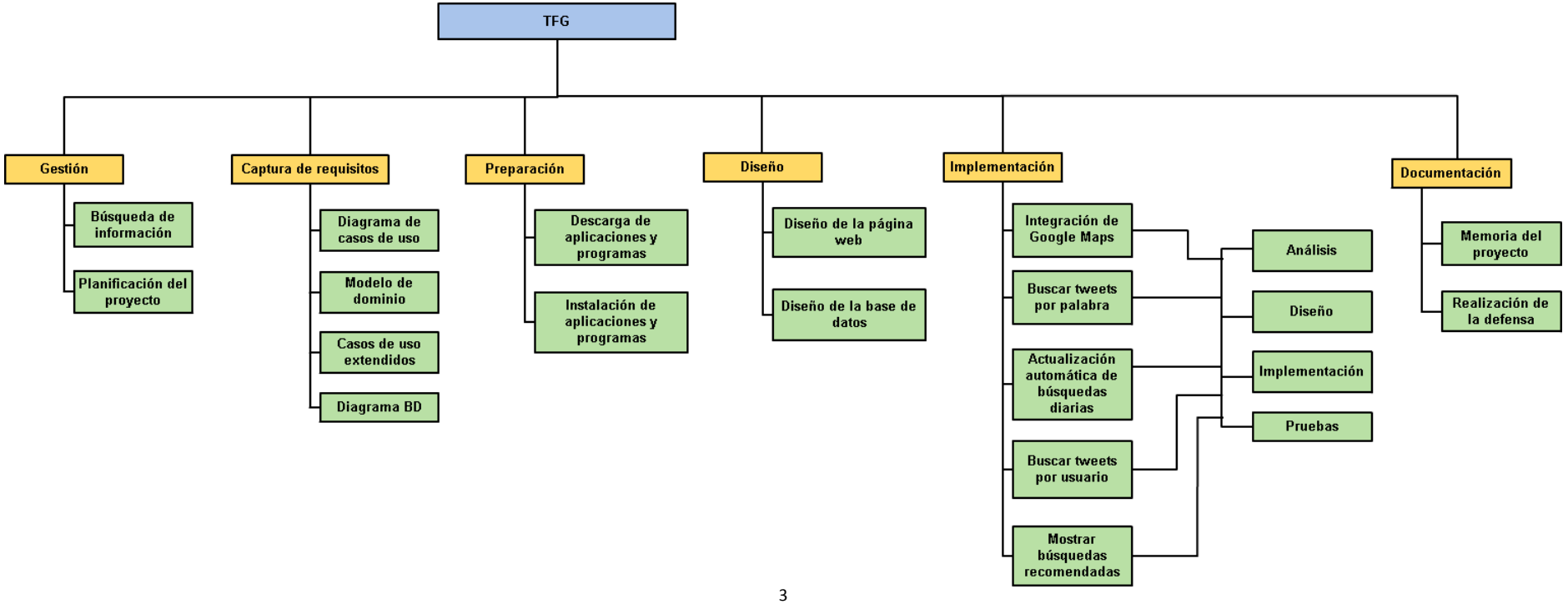
2. **Captura de requisitos**: Realización del diagrama de casos de uso, modelo de dominio, casos de uso extendidos y transformación del modelo de dominio a base de datos.

3. **Preparación**: Descarga e instalación de las aplicaciones y programas necesarios para la realización del proyecto.

4. **Diseño**: Diseño de la página web y de la base de datos.

5. **Implementación**: En este módulo se encuentran todas las iteraciones que hemos explicado anteriormente. Las iteraciones se irán repitiendo, generando distintas versiones de la aplicación cada vez más completas.

6. **Documentación**: Elaboración del documento de objetivos del proyecto y de la memoria del proyecto.



[3]

Ilustración 7. Estructura de Descomposición del Trabajo

[3] En la fase de implementación se refleja el carácter iterativo de estas tareas.

Paquete de trabajo: Búsqueda de información
Duración estimada: 30 horas (10 días). **Descripción**: Búsqueda de información sobre la idea que se quiere desarrollar, ver si existe algo similar y de ser así, que novedades introducirle para que sea diferente a lo existente. Por otra parte búsqueda de las herramientas que se utilizarán para la realización del proyecto. **Entradas**: / **Salidas/Entregables**: Documento Propuesta TFG por parte del alumnado. **Recursos necesarios**: Un ordenador con conexión a internet y Microsoft Word. **Precedencias**: /

Tabla 1. Búsqueda de información

Paquete de trabajo: Planificación del proyecto
Duración estimada: 9 horas (3 días). **Descripción**: Identificar y planificar todas las tareas que se van a realizar a lo largo del proyecto. **Entradas**: / **Salidas/Entregables**: / **Recursos necesarios**: Un ordenador y Microsoft Word. **Precedencias**: Búsqueda de información.

Tabla 2. Planificación del proyecto

Paquete de trabajo: Diagrama de casos de uso
Duración estimada: 9 horas (3 días). **Descripción**: Realización del diagrama que representa las funcionalidades de la aplicación web. **Entradas**: / **Salidas/Entregables**: Diagrama de casos de uso. **Recursos necesarios**: Un ordenador y Cacoo. **Precedencias**: /

Tabla 3. Diagrama de casos de uso

Paquete de trabajo: Modelo de dominio
Duración estimada: 9 horas (3 días). **Descripción**: Realización del diagrama que representa las entidades cuyos datos se necesitan almacenar para desarrollar la aplicación. **Entradas**: / **Salidas/Entregables**: Modelo de dominio. **Recursos necesarios**: Un ordenador y Cacoo. **Precedencias**: Diagrama de casos de uso.

Tabla 4. Modelo de dominio

Paquete de trabajo: Casos de uso extendidos
Duración estimada: 30 horas (10 días). **Descripción**: Realización de los casos de uso extendidos que se centra en la representación de la interfaz gráfica y en la descripción de cada una de las funcionalidades de los casos de uso. **Entradas**: Diagrama de casos de uso. **Salidas/Entregables**: Casos de uso extendidos. **Recursos necesarios**: Un ordenador y Cacoo. **Precedencias**: Diagrama de casos de uso.

Tabla 5. Casos de uso extendidos

Paquete de trabajo: Diagrama BD
Duración estimada: 15 horas (5 días). **Descripción**: En esta tarea transformaremos en una base de datos el modelo de dominio hecho anteriormente. **Entradas**: Modelo de dominio. **Salidas/Entregables**: Base de datos. **Recursos necesarios**: Un ordenador y Cacoo. **Precedencias**: Modelo de dominio.

Tabla 6. Diagrama BD

Paquete de trabajo: Descarga de aplicaciones y programas
Duración estimada: 6 horas (2 días). **Descripción**: Descargar todas las herramientas necesarias para el desarrollo del proyecto. **Entradas**: / **Salidas/Entregables**: / **Recursos necesarios**: Un ordenador con conexión a internet. **Precedencias**: /

Tabla 7. Descarga de aplicaciones y programas

Paquete de trabajo: Instalar aplicaciones y programas
Duración estimada: 15 horas (5 días). **Descripción**: Instalar todas las herramientas descargadas y configurarlas para su uso. **Entradas**: / **Salidas/Entregables**: / **Recursos necesarios**: Un ordenador. **Precedencias**: Descarga de herramientas y componentes.

Tabla 8. Instalar aplicaciones y programas

Paquete de trabajo: Diseño de la página web
Duración estimada: 60 horas (20 días). **Descripción**: Diseño y desarrollo de cada página de la aplicación web. **Entradas**: / **Salidas/Entregables**: Archivos HTML, CSS y JavaScript. **Recursos necesarios**: Un ordenador, Notepad++, Google Chrome, Mozilla Firefox, Internet Explorer, Twitter Bootstrap y Wamp. **Precedencias**: /

Tabla 9. Diseño de la página web

Paquete de trabajo: Diseño de la base de datos
Duración estimada: 6 horas (2 días). **Descripción**: Diseño y desarrollo de la base de datos a partir de su diagrama. **Entradas**: / **Salidas/Entregables**: Base de datos. **Recursos necesarios**: Un ordenador, Wamp y MySQL Workbench. **Precedencias**: Diagrama BD.

Tabla 10. Diseño de la base de datos

Paquete de trabajo: Integración de Google Maps
Duración estimada: 36 horas (12 días). **Descripción**: Integrar y configurar Google Maps para las necesidades requeridas por la aplicación web. También se incluye análisis, diseño, implementación y pruebas. **Entradas**: / **Salidas/Entregables**: Ficheros JavaScript. **Recursos necesarios**: Un ordenador, Notepad++, Google Chrome, Mozilla Firefox, Internet Explorer y Wamp. **Precedencias**: Captura de requisitos.

Tabla 11. Integración de Google Maps

Paquete de trabajo: Buscar tweets por palabra
Duración estimada: 51 horas (17 días). **Descripción**: Creación de la funcionalidad de buscar tweets por una o varias palabras aplicando los filtros de búsqueda, recibiendo los últimos tweets que contengan la palabra o palabras escritas por el usuario y almacenando la información necesaria de cada búsqueda en la base de datos. También se incluye análisis, diseño, implementación y pruebas. **Entradas**: Casos de uso extendidos de la búsqueda de tweets. **Salidas/Entregables**: Archivos PHP. **Recursos necesarios**: Un ordenador, Notepad++, Google Chrome, Mozilla Firefox, Internet Explorer y Wamp. **Precedencias**: Captura de requisitos.

Tabla 12. Buscar tweets por palabra

Paquete de trabajo: Actualización automática de búsquedas diarias
Duración estimada: 9 horas (3 días). **Descripción**: Creación de la funcionalidad que actualiza la base de datos cada una hora utilizando las búsquedas que se han hecho ese día. También se incluye análisis, diseño, implementación y pruebas. **Entradas**: Casos de uso extendidos de la actualización automática. **Salidas/Entregables**: Archivo PHP. **Recursos necesarios**: Un ordenador, Wamp y Notepad++. **Precedencias**: Captura de requisitos.

Tabla 13. Actualización automática de búsquedas diarias

Paquete de trabajo: Buscar tweets por usuario
Duración estimada: 27 horas (9 días). **Descripción**: Creación de la funcionalidad de buscar tweets por usuario aplicando los filtros de búsqueda, recibiendo los últimos tweets que publicados por ese usuario y almacenando la información necesaria de cada búsqueda en la base de datos. También se incluye análisis, diseño, implementación y pruebas. **Entradas**: Casos de uso extendidos de la búsqueda de tweets. **Salidas/Entregables**: Archivo PHP. **Recursos necesarios**: Un ordenador, Notepad++, Google Chrome, Mozilla Firefox, Internet Explorer y Wamp. **Precedencias**: Captura de requisitos.

Tabla 14. Buscar tweets por usuario

Paquete de trabajo: Mostrar búsquedas recomendadas
Duración estimada: 9 horas (3 días). **Descripción**: Creación de la funcionalidad de mostrar búsquedas que sirvan de guía al usuario. También se incluye análisis, diseño, implementación y pruebas. **Entradas**: Casos de uso extendidos de búsquedas recomendadas. **Salidas/Entregables**: Archivo PHP. **Recursos necesarios**: Un ordenador, Notepad++, Google Chrome, Mozilla Firefox, Internet Explorer y Wamp. **Precedencias**: Captura de requisitos.

Tabla 15. Mostrar búsquedas recomendadas

Paquete de trabajo: Memoria del Proyecto
Duración estimada: 60 horas (20 días). **Descripción**: Realización de la memoria del proyecto. **Entradas**: / **Salidas/Entregables**: Memoria del Trabajo de fin de grado. **Recursos necesarios**: Un ordenador, Microsoft Word, Microsoft Excel y Cacoo. **Precedencias**: Análisis, diseño e implementación.

Tabla 16. Memoria del Proyecto

Paquete de trabajo: Preparación y ejecución de la defensa de la defensa
Duración estimada: 9 horas (3 días). **Descripción**: Preparación y ejecución de la defensa del proyecto. **Entradas**: / **Salidas/Entregables**: Presentación del proyecto. **Recursos necesarios**: Un ordenador y Microsoft PowerPoint. **Precedencias**: Memoria del proyecto

Tabla 17. Preparación y ejecución de la defensa

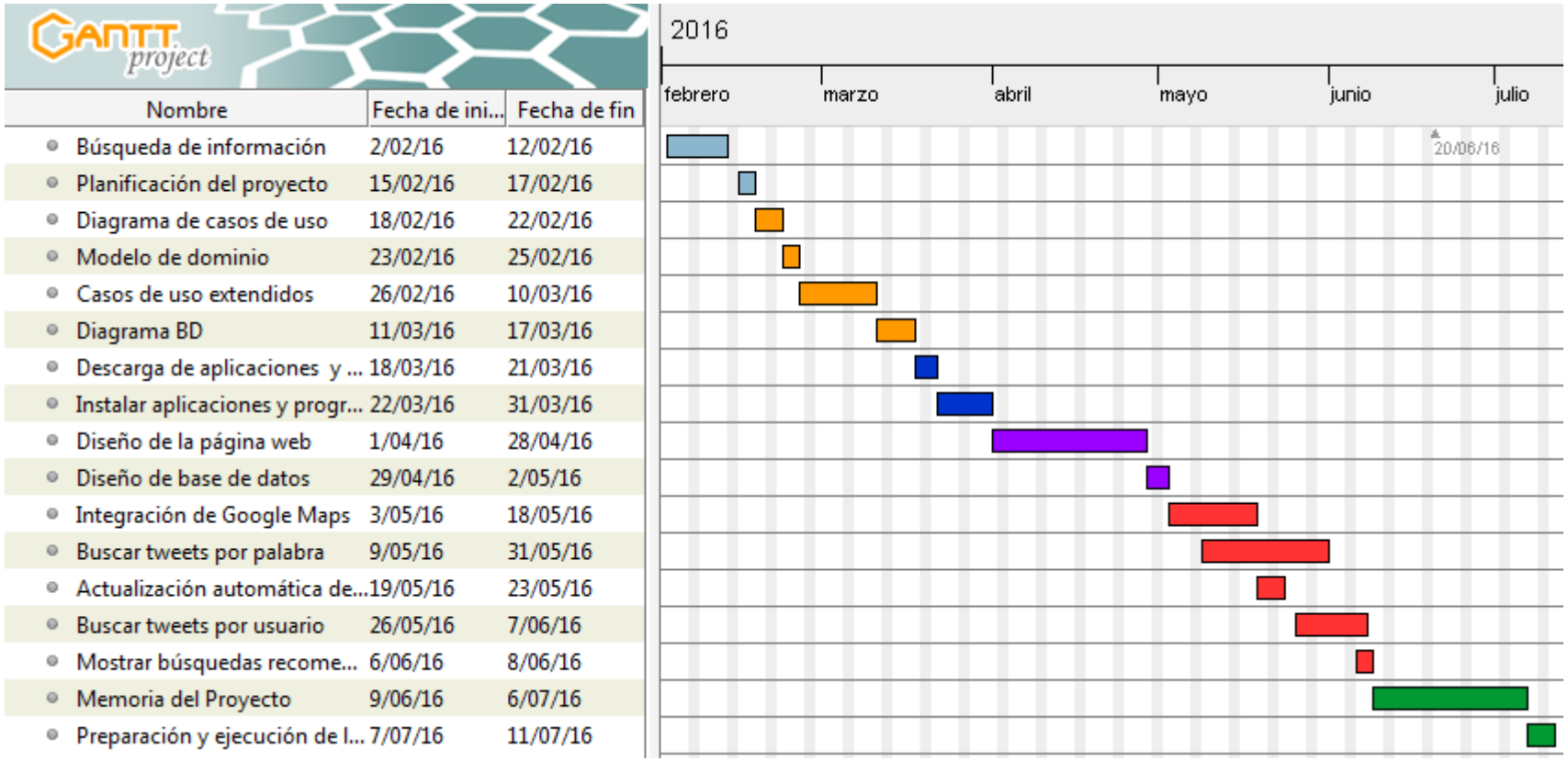
2.6. Planificación

Una vez definidas las tareas del proyecto, hay que calcular como se van a dividir en el tiempo. El tiempo de dedicación previsto se representa mediante un diagrama Gantt (Ilustración 8).

Al trabajar por las mañanas de 8:00-15:00, solo disponiendo de las tardes para la realización de TFG, se estima una carga de trabajo diaria de 3 horas salvo en la parte de implementación, en la que se hará un esfuerzo y habrá días en los que se trabajarán 6 horas. Los fines de semana y los días festivos como semana santa en principio no se trabajarían, a no ser que haga falta terminar una tarea pendiente.

Por lo tanto, poniendo que el trabajo comienza el 2 de febrero, si se sigue el plan previo éste terminaría el 11 de Julio.

El siguiente diagrama está dividido en las tareas generadas en el EDT. En el grafico se muestran todas las tareas que se van a llevar a cabo y la duración de cada una de ellas. Hasta que no se termine una tarea no se empezará con la siguiente, excepto en la fase de implementación, en el que se ha elegido un modelo iterativo.



Nombre	Fecha de ini...	Fecha de fin
Búsqueda de información	2/02/16	12/02/16
Planificación del proyecto	15/02/16	17/02/16
Diagrama de casos de uso	18/02/16	22/02/16
Modelo de dominio	23/02/16	25/02/16
Casos de uso extendidos	26/02/16	10/03/16
Diagrama BD	11/03/16	17/03/16
Descarga de aplicaciones y ...	18/03/16	21/03/16
Instalar aplicaciones y progr...	22/03/16	31/03/16
Diseño de la página web	1/04/16	28/04/16
Diseño de base de datos	29/04/16	2/05/16
Integración de Google Maps	3/05/16	18/05/16
Buscar tweets por palabra	9/05/16	31/05/16
Actualización automática de...	19/05/16	23/05/16
Buscar tweets por usuario	26/05/16	7/06/16
Mostrar búsquedas recome...	6/06/16	8/06/16
Memoria del Proyecto	9/06/16	6/07/16
Preparación y ejecución de l...	7/07/16	11/07/16

Ilustración 8. Diagrama de Gantt

A continuación se muestra con más detalle la duración de las tareas:

TAREA	FECHA	DURACIÓN (horas)
Gestión		
Búsqueda de información	01/02/2016 - 12/02/2016	30
Planificación del proyecto	15/02/2016 - 17/02/2016	9
Captura de requisitos		
Diagrama de casos de uso	18/02/2016 - 22/02/2016	9
Modelo de dominio	23/02/2016 - 25/02/2016	9
Casos de uso extendidos	26/02/2016 - 10/03/2016	30
Diagrama BD	11/03/2016 - 17/03/2016	15
Preparación		
Descarga de aplicaciones y programas	18/03/2016 - 21/03/2016	6
Instalar aplicaciones y programas	22/03/2016 - 31/03/2016	15
Diseño		
Diseño de la página web	01/04/2016 - 28/04/2016	60
Diseño de base de datos	29/04/2016 - 02/05/2016	6
Implementación		
Integración de Google Maps	03/05/2016 - 18/05/2016	36
Buscar tweets por palabra	09/05/2016 - 31/05/2016	51
Actualización automática de búsquedas diarias	19/05/2016 - 23/05/2016	9
Buscar tweets por usuario	26/05/2016 - 07/06/2016	27
Mostrar búsquedas recomendadas	06/06/2016 - 08/06/2016	9
Documentación		
Memoria del Proyecto	09/06/2016 - 06/07/2016	60
Preparación y ejecución de la defensa	07/07/2016 - 11/07/2016	9
TOTAL		**390**

Tabla 18. Fechas y duración total de las tareas

2.7. Gestión de riesgos

Pérdida de información
Descripción: Pérdida de datos por avería o fallo del equipo utilizado para el desarrollo el proyecto. **Prevención**: Hacer una copia de seguridad del código y la documentación en un pendrive y Dropbox. **Plan de contingencia**: Utilizar la última versión guardada. **Probabilidad**: Baja. **Impacto**: Alto.

Tabla 19. Pérdida de información

Enfermedad o lesión
Descripción: No poder realizar el proyecto por una causa médica. **Prevención**: Tomar precauciones para evitar exponerse a enfermedades o lesiones. **Plan de contingencia**: Acudir al médico para diagnosticar el alcance de la lesión. Reorganizar la planificación temporal para recuperar el tiempo perdido. **Probabilidad**: Media. **Impacto**: Baja.

Tabla 20. Enfermedad o lesión

“Quedarse en blanco”
Descripción: Llegar a un momento en el que no se sabe continuar en una cierta tarea. **Prevención**: Prepararse correctamente para la realización de este trabajo. **Plan de contingencia**: Buscar en libros de la temática o en internet sobre el problema. **Probabilidad**: Media. **Impacto**: Baja.

Tabla 21. “Quedarse en blanco”

Realizar una mala planificación
Descripción: Realizar una mala planificación temporal y sufrir por ello pérdidas de tiempo en los plazos estimados. **Prevención**: Realizar una planificación lo más precisa posible. **Plan de contingencia**: Reorganizar el tiempo para recuperar el tiempo perdido. **Probabilidad**: Alta. **Impacto**: Medio.

Tabla 22. Realizar una mala planificación

Ordenador estropeado
Descripción: Daños imprevistos en el equipo de trabajo, ya sea de hardware como de software, debido acciones relacionadas con el trabajo, transporte de material o agentes externos. **Prevención**: Hacer un uso responsable del equipo, manteniéndolo protegido contra agentes externos. Realizar copias de seguridad periódicamente. **Plan de contingencia**: Usar las copias de seguridad para restaurar los datos perdidos en otro equipo. **Probabilidad**: Baja. **Impacto**: Alto.

Tabla 23. Ordenador estropeado

Caída del servidor
Descripción: Caída del servidor en el que se alojan la aplicación web y la base de datos. Si esto ocurriese no podría utilizarse la aplicación. **Prevención**: Guardar los datos en más de un servidor. **Plan de contingencia**: Utilizar otro servidor que tenga los datos de la aplicación y la base de datos. **Probabilidad**: Baja. **Impacto**: Alto.

Tabla 24. Caída del servidor

Pérdida de internet
Descripción: Pérdida de la conexión a internet en el equipo. Esto provocaría la incapacidad de buscar información o probar la aplicación en el servidor. **Prevención**: Contratar un servicio de internet en una compañía fiable, así como mantener el router en un entorno alejado de posibles incidencias. **Plan de contingencia**: Conectarse a alguna red WiFi disponible, como la de la universidad. **Probabilidad**: Baja. **Impacto**: Baja.

Tabla 25. Pérdida de internet

2.8. Evaluación económica

A continuación se detalla la evaluación económica del proyecto:

- Personal:

  Se ha calculado que un programador gana e media 20€/hora. Se ha estimado que número total de horas son 390. Por lo tanto el coste bruto del personal del proyecto es:

  20€ * 390h = 7.800€

  Total: 7.800€

- Hardware:

  Para el desarrollo del proyecto se ha utilizado un ordenador portátil Asus valorado en 522€. Teniendo en cuenta que la vida útil de un portátil son 5 años y que la duración del proyecto es de 6 meses y medio:

  522€ / 5 años = 104,4€
  104,4€ en 6,5 meses = 56,55€

  Total: 56,55€

- Software:

  Casi todos los componentes software han sido gratuitos gracias a la licencia que tiene la universidad con Microsoft Azure. Para realizar de la memoria se ha utilizado Microsoft Office 2010, que ha supuesto un gasto de 70€ con una licencia de 3 años. También se ha utilizado Adobe Photoshop, el coste para poder usar esta aplicación durante un año es de 290€. Por lo que:

  70€ / 3 años = 23,33€
  290€ * 1 años = 290€
  313,33€ en 6,5 meses = 169,72€

  Total: 169,72€

- Materiales:

  Aproximadamente el gasto en otro tipo de materiales son de 20€.

MOTIVO	COSTE
Personal	7.800,00 €
Hardware	56,55 €
Software	169,75 €
Materiales	20 €
COSTE TOTAL	**8.046,30 €**

Tabla 26. Coste total

Como se puede observar después de calcular los gastos, la creación de una aplicación web no es barata. Por este motivo es conveniente pensar de qué manera se van a conseguir los ingresos necesarios para que la aplicación salga rentable.

Después de investigar sobre el tema y haber encontrado varias maneras para amortizar el coste de la aplicación, me he decantado por la siguiente:

Publicidad online: Incluir publicidad en la página web, por ejemplo mediante banners o anuncios de texto. Es un método lento y los ingresos son pequeños. Hay tres maneras para ganar dinero con publicidad. Que el anunciante pague por visualización, clics o acción del banner. A continuación se explica más detalladamente las diferentes opciones:

1. Coste por Mil Impresiones (CPM)**:** se paga un precio "x" por 1.000 visualizaciones de un banner en un sitio web. Es el método más común de todos.

2. Coste por Clic (CPC)**:** el CPC es un modelo donde un anunciante paga por cada clic que un navegante realiza en un anuncio de texto o banner. Suena sencillo pero no existe un estandár común entre los diferentes actores. Esto hace que se ha creado un poco de desconfianza en este modelo de pago.

3. Coste por Acción (CPA)**:** el CPA es el modelo más complejo de los tres. El anunciante paga por ventas o registros. Según el tipo de "acción" deseada se diferencia entre CPS (Cost per Sale, también conocido como CPV, Coste por Venta) cuando se trata de una venta o CPL (Cost per Lead) cuando se trata de un registro que requiere p.ej. rellenar algún tipo de formulario.

Para esta aplicación se ha escogido que el anunciante pague con el método CPM. Considerando la estructura de la interfaz principal de la web, se calcula que podría albergar tres banners: uno en la cabecera, otro en el lateral izquierdo y otro al final de la página. Una vez establecidos estos datos se expondrá el siguiente ejemplo.

Imaginando, y siendo muy optimistas, que la web tiene 100.000 visitas al mes y que cada uno de los banners publicitarios tiene un CPM de 0,5€ de media, se calculan los siguientes ingresos:

Ingresos (€ por mes) = nº de visualizaciones x nº de banners x CPM medio

Ingresos = (100.000 x 3 x 0,5€) / 1.000 = 150€ al mes

Con este cálculo para llegar a los 8.046,3€ se necesitarían casi 4 años y medio.

Un usuario habitual de una web la visita de media 5 veces al día. Teniendo en cuenta que con tres banners se reciben 1,5€ por 1.000 visualizaciones de la página, se necesitarían 5.364.200 visitas para recuperar el dinero invertido. Sabiendo que un año consta de 365 días:

*5.364.200 / (365 * 5) = 2.939,28*

Se necesitarán 2.940 usuarios habituales para recuperar la inversión en un año.

3. Antecedentes

Una vez definidas las tareas a realizar, se procede a buscar aplicaciones similares con el fin de compararlas de cara a añadir o mejorar funcionalidades de Twitter Finder.

Para este caso, las aplicaciones que se han buscado son aplicaciones que combinan el uso de Twitter y Google Maps. De las encontradas, ninguna me pareció completa, por lo que considero que Twitter Finder contendrá funcionalidades que según mi criterio, les faltan a las demás.

- Twaps:

  Es una aplicación web que muestra los tweets en un mapa alrededor de la localización del usuario o de una escogida por él. Es una aplicación limitada, ya que solo muestra los últimos 50 tweets emitidos en un radio de 2 kilómetros de la dirección elegida. Los tweets son representados con la fotografía del usuario que los escribió, y al pulsar en ella se lee el texto del tweet. Para poder utilizar esta aplicación, antes de nada hay que compartir la localización del usuario con la aplicación. Puede encontrarse en la siguiente dirección: *http://www4.itsatwap.com.*

- Tworldy:

  Tworldy permite conocer mediante Google Maps los trending topics por ciudad o país. En el mapa aparecen pájaros azules sobre distintos países. Al pulsar en un pájaro nos aparecen los temas más comentados de ese país, también nos da la opción de seleccionar por ciudad. Al hacer click en alguno de los hashtags, automáticamente la página se redirige a Twitter para poder ver los tweets que se están comentando sobre esa temática. Puede encontrarse en la siguiente dirección: *http://tworldy.in.*

- Twitmap:

  Aplicación móvil. Twitmap muestra tweets en un mapa con marcadores tal y como hace Twitter Finder. Permite buscar los tweets en un radio concreto del mapa, buscar tweets por palabra clave y también por nombre de usuario. Al pulsar el marcador se ve la foto y el nombre del usuario que ha publicado el tweet, además del texto del tweet. Puede descargarse de la plataforma para móviles *Play Store*.

- Maps for Twitter:

  Otra aplicación móvil. Esta aplicación es la aplicación más completa que se ha encontrado. Es muy parecida a la anterior. Las funcionalidades son prácticamente idénticas, salvo que añade la opción de ver los tweets en el mapa o en una lista. Además es más intuitiva y cuenta con un mejor diseño de interfaz. Se puede descargar en *Play Store*.

Después de analizar todas ellas, no encontré ninguna que tuviera todas las funcionalidades que me habría gustado encontrarme en una aplicación de este estilo, como por ejemplo una sección con búsquedas recomendadas o que distinguiese cada tweet por antigüedad. Twitter Finder tendrá lo bueno que he encontrado en las otras aplicaciones y añadirá las funcionalidades que me habría gustado encontrarme.

4. Captura de requisitos

4.1. Modelo de casos de uso

La aplicación tiene dos actores: usuario y reloj. Cada uno de ellos realiza distintas funcionalidades. El usuario es el que hace uso de la aplicación web. Mientras que el reloj será una tarea programada, ejecutará su tarea cada cierto tiempo. Se han elaborado los siguientes diagramas para representar las funcionalidades que pueden hacer los distintos actores (Ilustración 9 y 10).

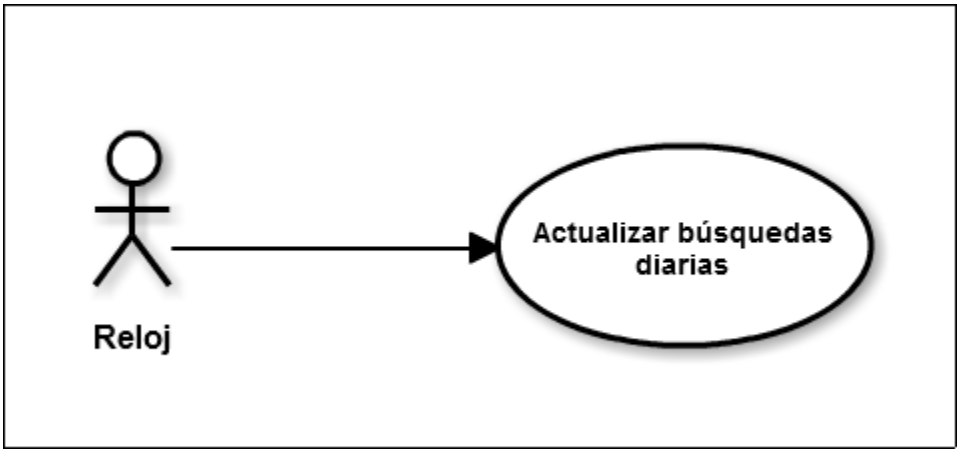


Ilustración 9. Modelo de casos de uso Reloj

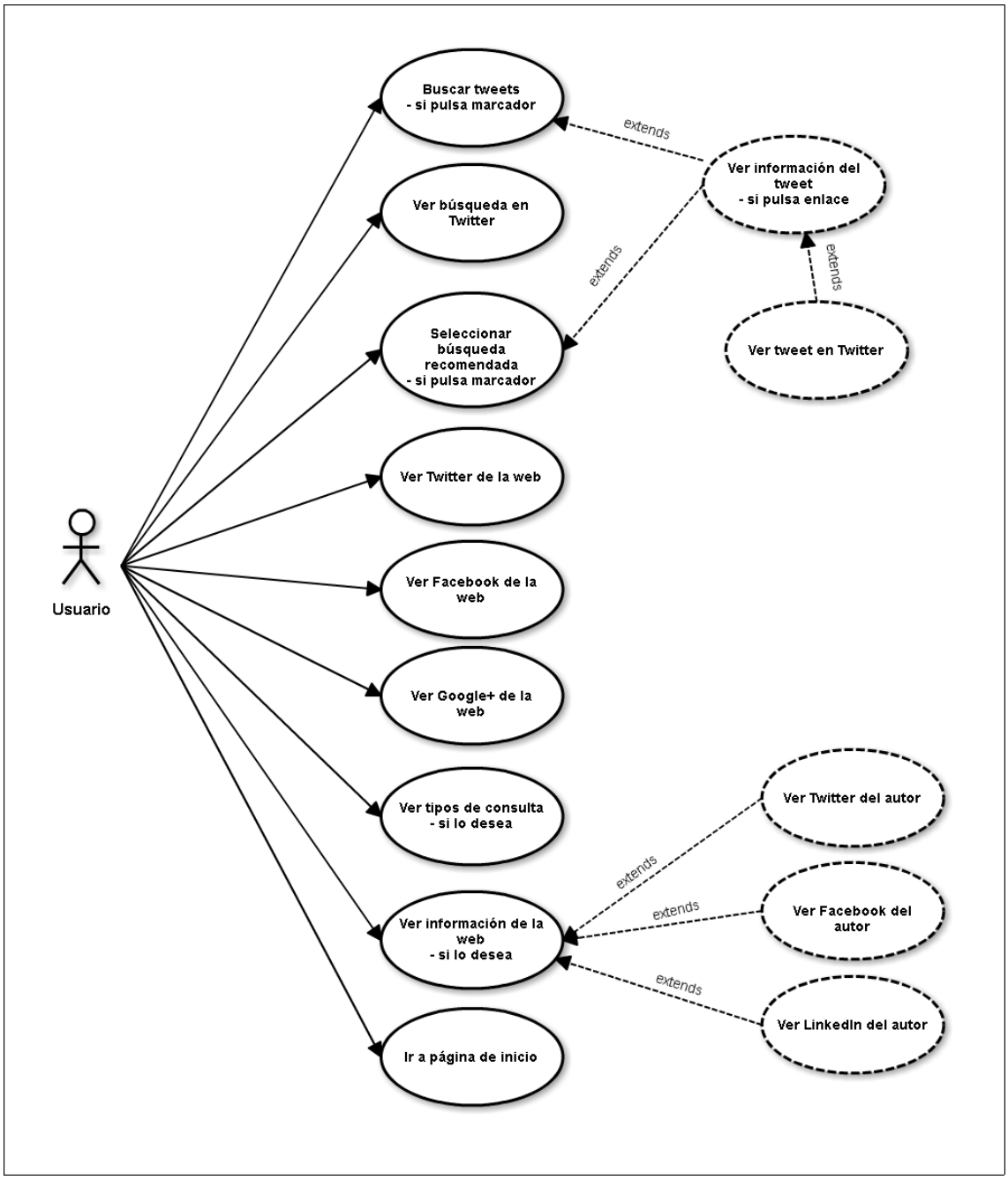


Ilustración 10. Modelo de casos de uso Usuario

En este caso ninguna de las partes tiene funcionalidades en común. Por lo tanto, la jerarquía de actores quedaría sin ninguna unión entre ambos.

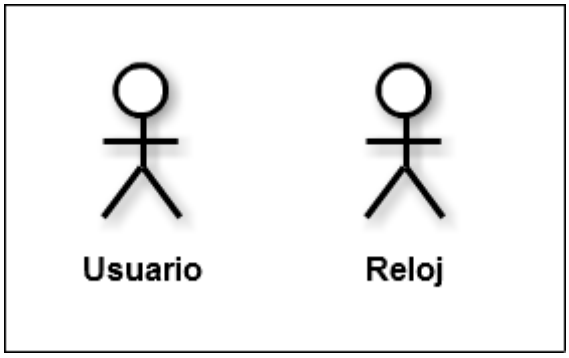


Ilustración 11. Jerarquía de actores

4.2. Casos de uso extendidos (Anexo I)

Los casos de uso extendidos describen cada una de las funcionalidades que parecen en el modelo de casos de uso de manera mucho más precisa, ayudándose de interfaces gráficas. Describen el proceso completo de las funcionalidades teniendo en cuenta todas las opciones posibles que se pueden dar. Debido a su gran tamaño en cuanto a espacio de documentación se refiere, este apartado es presentado aparte en forma de anexo.

4.3. Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos del dominio pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio.

El objetivo del modelo de dominio es comprender y describir las clases más importantes dentro del contexto del sistema.

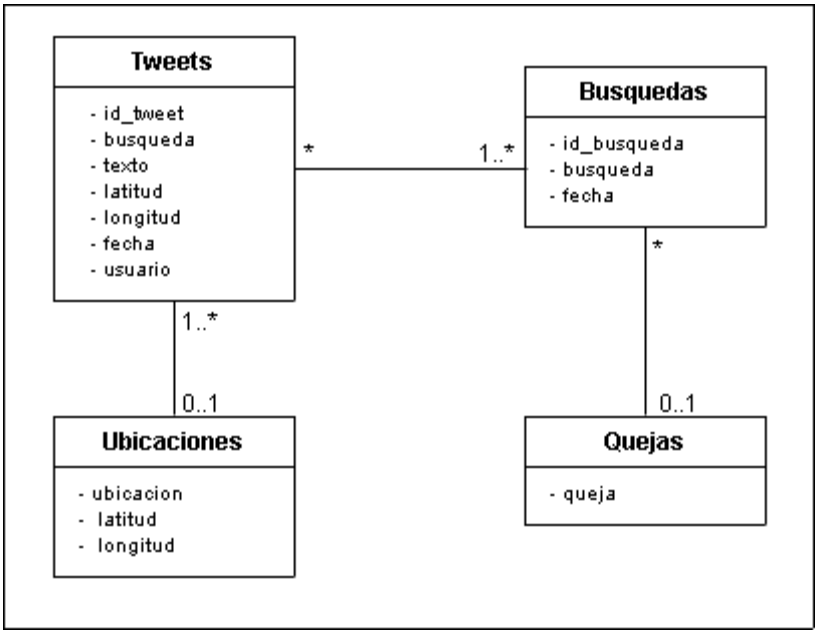


Ilustración 12. Modelo de dominio

A continuación se explica con más detalle la función cada una de las entidades:

- Tweets:

  El primer campo de esta entidad es el id_tweet, que será el identificador del tweet. El segundo es la búsqueda introducida por el usuario, la cual ha encontrado el tweet. También contendrá el texto del tweet, la latitud y longitud en la cual fue escrito, la fecha y la hora en la que se publicó, y el usuario que lo escribió. Se utilizará para dibujar los marcadores en el mapa que representan los tweets, así como para generar la lista de las 10 palabras con más tweets y crear el enlace al tweet original. Un tweet puede ser la respuesta de 1 o más búsquedas, además puede o no contener las coordenadas de una ubicación.

- Búsquedas:

  La entidad búsquedas estará formada por el id_busqueda, que es un número auto-incremental, la búsqueda que realizada por el usuario y la fecha en la que la hizo. Sirve para contabilizar las búsquedas diarias, actualizar dichas búsquedas cada hora y mostrar las siguientes listas: palabras más buscadas, usuarios más buscados, más buscado este mes y quejas de hoy. Una búsqueda puede generar 0 o más tweets como respuesta, además puede o no ser una queja.

- Ubicaciones:

  Esta entidad contiene una ubicación con sus coordenadas (latitud y longitud). Se usará para tener la información de las distintas localidades en las que los usuarios se han registrado, de esta manera no se pedirá a Google Maps continuamente esta información y se pintarán los marcadores con mayor rapidez. Las coordenadas de una ubicación pueden estar en 1 o más tweets.

- Quejas:

  Entidad que solo tiene un campo, la queja, es el único campo que será introducido a mano por el administrador. Se utiliza para crear la lista “Quejas de hoy”. Una queja puede haber sido buscada 0 o más veces.

5. Análisis y diseño

5.1. Diseño de la base de datos

El gestor de bases de datos utilizado en la aplicación es MySQL. Es un sistema gestor de bases de datos OpenSource, además, la licencia de este gestor es gratuita.

La realización del diseño de las tablas de la base de datos de la aplicación se ha desarrollado con el programa phpMyAdmin. Es una herramienta gratuita que facilita la elaboración del diseño de la base de datos de manera visual, para posteriormente generar el código de manera automatizada.

Para poder empezar a trabajar con una base de datos, primero es necesario realizar una transformación basándose en el modelo de dominio de la aplicación. Lo que se hace es transformar las entidades y las relaciones en tablas para la base de datos. Las entidades que pasarán a ser tablas son:

- Tweets
- Búsquedas
- Ubicaciones
- Quejas

Una vez que obtenemos las tablas, normalizados la base de datos. Se asignan las claves primarias y foráneas, en este caso se ha decidido que no habrá claves foráneas por comodidad. En la siguiente ilustración se muestra el diseño final de la base de datos.

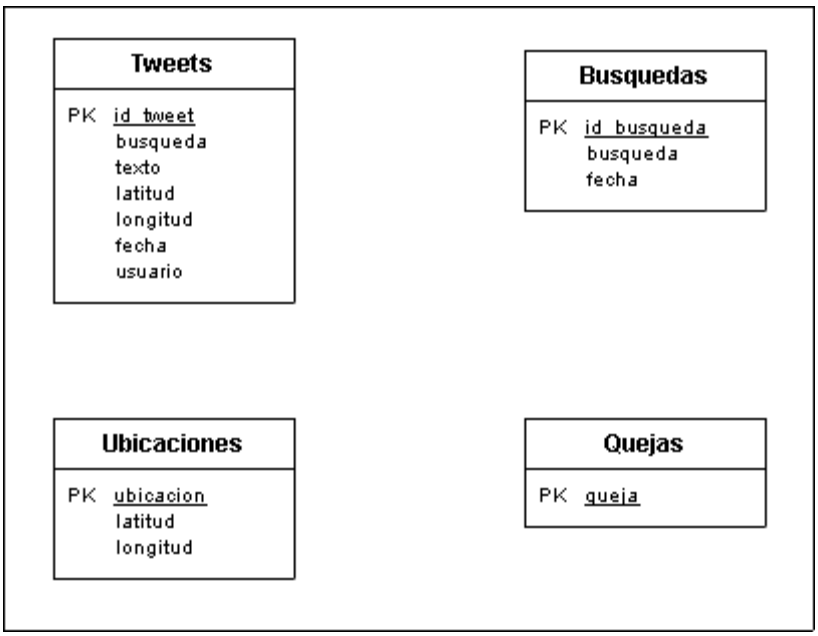


Ilustración 13. Diseño de la base de datos

5.2. Diseño de la aplicación web

El objetivo del presente apartado pretende ofrecer una visión lo más generalizada posible de la aplicación web. Esta aplicación funcionará de forma autónoma sobre cualquier plataforma, valiéndose para ello de cualquier navegador web de actualidad y obviamente, de una conexión a Internet.

Las tecnologías principales empleadas en la implementación de los módulos serán:

- Lenguaje HTML y PHP dado su carácter dinámico.
- Plantillas CSS para optimizar la legibilidad de la información mostrada.
- JavaScript para moldear la forma en que se muestran algunas páginas web.
- MYSQL como sistema de gestión de base de datos.

Durante el desarrollo y pruebas se utilizará un servidor local (Wamp) para alojar la aplicación. Una vez terminada, se alojará en el servidor gratuito Mirosoft Azure y así poder acceder a ella desde cualquier ordenador.

Todos los archivos generados para la creación de esta aplicación web se encontraran en una misma carpeta. La estructura de esta carpeta es la siguiente:

- La interfaz principal llamada “index.php” se encuentra en la raíz.
- Las otras dos interfaces, “consultas.html” y “about.html” están dentro de la carpeta “pages”.
- Las hojas de estilo que darán el buen aspecto a las páginas web se ubican en la carpeta “css”.
- Los scripts que se usarán para conectarse e interactuar con la base de datos y gestionar las actividades de Twitter se encuentran en la carpeta “php”.
- Los scripts que integran y gestionan las actividades necesarias de Google Maps en la aplicación están en la carpeta “js”.
- Las imágenes que usa la aplicación se encuentran en la carpeta “images”.

Para darle mejor aspecto a la web, se usa una plantilla descargada de Twitter Bootstrap que contiene las siguientes carpetas: bower_components, dist y less. Todos los archivos que están dentro de estas carpetas no han sido modificados.

5.3. Diagramas de secuencia (Anexo II)

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Contienen objetos y clases que se usan para la implementación. Debido a su gran tamaño en cuanto a espacio de documentación se refiere, este apartado es presentado aparte en forma de anexo.

6. Desarrollo

En este apartado se va a explicar el desarrollo que se ha realizado para llevar a cabo la aplicación Twitter Finder. Se explicarán algunos de los problemas encontrados y las soluciones utilizadas para corregirlos.

Primero se desarrolló la base de datos para poder introducir todos los datos necesarios y después la aplicación.

6.1. Base de datos

Como se ha explicado anteriormente, para desarrollar la base de datos de la aplicación se utilizó una base de datos MySQL alojada un servidor local (Wamp), cuando el desarrollo de Twitter Finder terminó, se alojó en el servidor gratuito Microsoft Azure a través de MySQL Workbench.

Conviene destacar, que la primera versión del diseño de la base de datos fue algo diferente a la versión final, debido a que, a lo largo del proyecto, fueron necesarias realizar modificaciones para solucionar problemas que iban surgiendo.

Una de las modificaciones más destacadas, fue la de añadir un campo “usuario” a la tabla “tweets”. Cuando se añadió la funcionalidad de acceder al tweet original, era necesario tener almacenado al autor del propio tweet. Por lo que se sumó el campo para almacenar a cada autor de los tweets guardados.

Otra modificación bastante importante fue la de añadir la tabla “ubicaciones”. En un principio solo se almacenaban los tweets con la geolocalización activa, esto es, que los campos latitud y longitud del tweet no estuviesen vacíos. Al ver que eran muy pocos tweets los que aparecían con cada búsqueda, se decidió guardar los tweets que contasen con la ubicación de registro del autor del tweet. Aquí surgió el problema, ya que para conseguir las coordenadas de la localización de registro de cada autor, hay que interactuar con Google Maps mediante su API. Esto provocaba una gran itinerancia de datos y en ocasiones bloqueaba la aplicación. Para solucionar este problema se creó la tabla “ubicaciones”, para guardar cada localidad con sus respectivas coordenadas, haciendo que así no haya que utilizar siempre la API de Google Maps, evitando que la aplicación se colapse.

Por último, se decidió añadir la tabla “quejas” con la idea de mostrar una lista de quejas cada día, para que el usuario sepa de qué se está quejando la gente en la actualidad.

6.2. Aplicación web

6.2.1. Introducción

Como se ha comentado al inicio, es una aplicación web creada con las tecnologías web HTML, CSS, JavaScript y PHP. Para desarrollar la aplicación se utilizó el servidor local Wamp, y los archivos que la componen fueron editados con Notepad++.

Se decidió usar Wamp ya que es una aplicación que permite instalar y configurar en el sistema lo último del servidor Web Apache, el lenguaje de programación PHP y el servidor de base de datos MySQL muy fácilmente. Además, Wamp es muy útil cuando se trabaja en la creación de una página o aplicación web, ya que te permite trabajar y editarla localmente sin necesidad de subirla a un servidor, como fue este caso.

Ilustración 14. Wamp Server

http://falconhive.com/host-website-wamp-server/

Para darle un mejor diseño y visualización a la página web, se usó una plantilla de Twitter Bootstrap. Esta plantilla contiene librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. De esta manera, se consigue que cada página web de la aplicación pueda verse correctamente en cada ordenador.

Ilustración 15. Twitter Bootstrap

http://twoggle.com/blog/twitter-bootstrap-benefits/

Tal y como se hizo con la base de datos, la aplicación web también se alojó en el servidor Microsoft Azure. De esta manera se puede acceder desde cualquier ordenador a Twitter Finder, lo que sirvió para comprobar su correcto funcionamiento de sus distintas funcionalidades, así como para asegurarse de su correcta visualización en las distintas pantallas de ordenador. Para el intercambio de archivos con el servidor se utilizó Filezilla. La url para acceder a la aplicación es la siguiente:

http://proyectotwitter.azurewebsites.net

6.2.2. Diseño e implementación de la interfaz gráfica

El diseño de la aplicación responde a la necesidad de tener la información de una manera clara y ordenada, y a su vez gozar de un aspecto trabajado y original. El diseño se realiza buscando constantemente la simplicidad en el uso de la aplicación, a la vez de llamar la atención del usuario. A continuación podemos ver una imagen que refleja la estructura de la página web:

Ilustración 16. Estructura página web

http://librosweb.es/libro/css/capitulo_12/estructura_o_layout.html

Una vez escogida la plantilla de Twitter Bootstrap, se comienza con el diseño de la interfaz gráfica de la aplicación web, confeccionando la página principal y sus subsecciones, que son “Tipos de consulta” y “¿Qué es Twitter Finder?”.

Toda la aplicación se controla desde un menú superior que siempre es visible, desde que el usuario accede a la aplicación. La estructura está dividida en bloques o módulos accesibles desde el menú, que hacen fácil e intuitivo su uso.

La página web tiene una interfaz principal (Ilustración 17) y en la cabecera contará con un menú que permitirá navegar por la aplicación, este menú estará

presente en todas las páginas. Igualmente, tiene un pie de página que contiene los enlaces a las distintas redes sociales en las que Twitter Finder está presente.

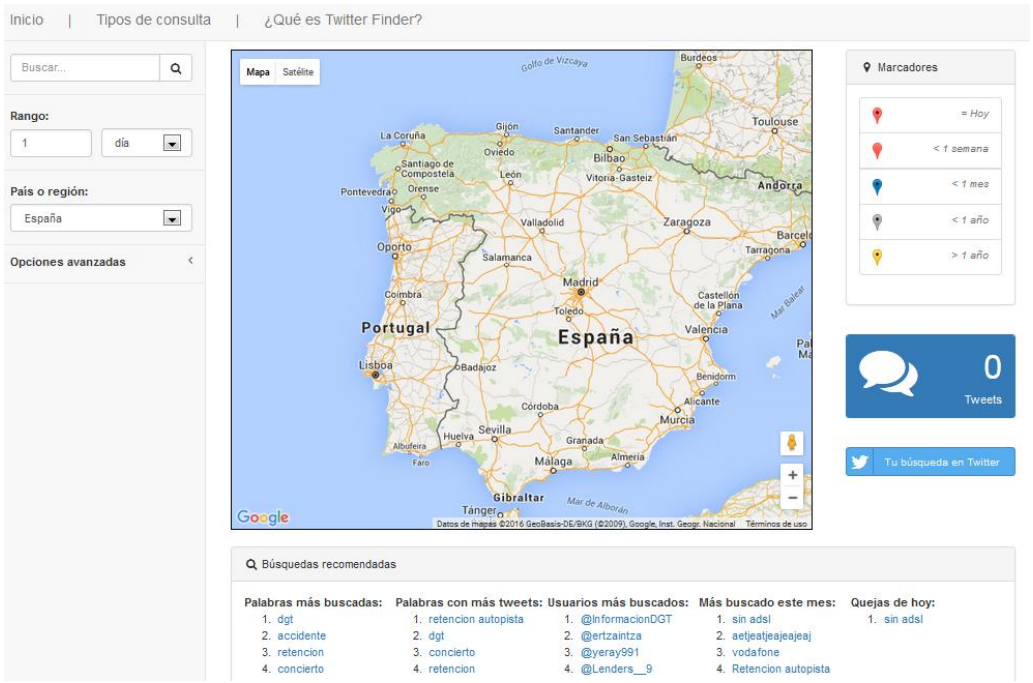


Ilustración 17. Interfaz principal de Twitter Finder

Esta aplicación web tiene otras dos subsecciones, una de ellas es en la que puede verse los distintos tipos de consulta que puede hacerse (Ilustración 18).

Ilustración 18. Página "Tipos de consulta"

Por último, cuenta con una sección en la que puede verse información explicativa sobre la aplicación (Ilustración 19).

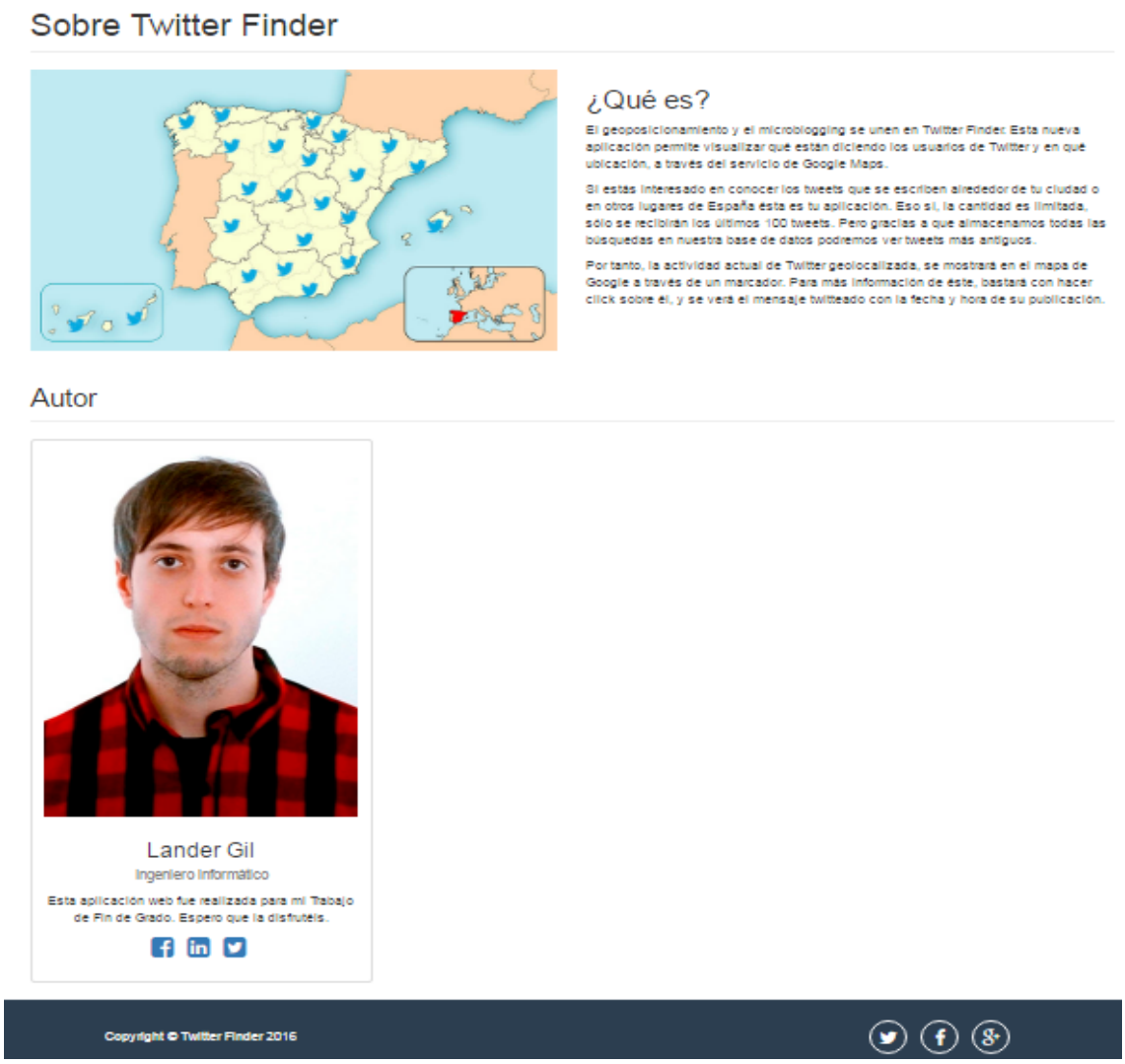


Ilustración 19. Página "¿Qué es Twitter Finder?"

6.2.3. Conexión con la base de datos

Como ya se ha explicado anteriormente, la base de datos que se utiliza es una base de datos *MySQL* que está alojada en el servidor de Microsoft Azure.

Para obtener e insertar la información que hay guardada en la base de datos se requiere usar la tecnología PHP. En los archivos PHP está el código necesario para que la aplicación pueda conectarse a la base de datos y así poder ejecutar operaciones escritas en él. En este caso van a realizarse operaciones de tipo SELECT, INSERT y REPLACE. Las operaciones SELECT permitirán traer a la aplicación los datos que se quieren mostrar en la interfaz. Los INSERT servirán para almacenar información en la base de datos. Los REPLACE tienen una función muy parecida al INSERT, la única diferencia es que al guardar la información en la base de datos, si esos datos ya existen, se reemplazan.

```
$link = mysql_connect($servidor,$usuario,$password);
mysql_query("SET NAMES utf8");
mysql_select_db("twitter",$link);
```

Ilustración 20. Conexión a la base de datos

En la imagen anterior podemos ver como se realiza la conexión a la base de datos alojada en el servidor, con un usuario que tiene los permisos necesarios para realizar operaciones en ella, y la contraseña de dicho usuario. Esta información para mayor seguridad no se muestra en el código, es alojada en otro archivo PHP llamado "secret.php". Después de realizar la conexión con el servidor se escoge el cotejamiento de la base de datos, que será *utf8*, y se selecciona la base de datos a utilizar, en este caso "twitter".

```
mysql_query("INSERT INTO busquedas(busqueda, fecha) VALUES ($buscarBD, CURDATE())",$link);

$num_busquedas = mysql_query("SELECT COUNT(busqueda) FROM `busquedas` WHERE (busqueda = $buscarBD AND fecha = CURDATE())");
```

Ilustración 21. Operaciones con la base de datos

En esta imagen vemos un ejemplo de dos operaciones en la base de datos. La primera es un INSERT donde se están guardan datos. La segunda es un SELECT COUNT, en la que se obtiene información de la base de datos y se guarda en la variable *$num_busquedas*, para poder utilizarla posteriormente.

6.2.4. Conexión con Twitter

La funcionalidad más importante de la aplicación es la búsqueda tweets mediante palabras clave. Esto se consigue extrayendo la información de Twitter, para ello ha sido necesario establecer conexión con la aplicación de Twitter y así poder interactuar con ella.

Hasta que llegó la última actualización de versión, en la anterior, la v1.0, Twitter permitía acceso a los desarrolladores a determinados parámetros sin requerir que sus aplicaciones estuvieran autentificadas, esto quería decir que, podían acceder a determinada información pública de Twitter sin saber quiénes son. Para prevenir el uso malicioso, se actualizo a la versión 1.1 en la que se exige a cada solicitud de conexión ser autentificada. Estas autentificaciones se realizan mediante *OAuth*, que es un protocolo que permite la autorización segura de una API para aplicaciones de escritorio, móviles y web.

Para poder usar Twitter en nuestra aplicación es necesario tener la autentificación *OAuth.* Primero hay que crear una cuenta de desarrollador de Twitter, una vez creada, hay que crear los Tokens de acceso, que son claves que vamos a necesitar para comunicarnos con Twitter. Después de tener creadas todas las claves, hay que descargar la librería twitteroauth, que puede encontrarse en *https://github.com/abraham/twitteroauth.* Esta librería está preparada para realizar conexiones con Twitter mediante PHP. Debe de estar almacenada en el servidor con el resto e archivos para que el archivo PHP que vaya a realizar la comunicación con Twitter pueda utilizarla.

```
$twitter = new twitteroauth($consumer_key,$consumer_secret,$oauth_access_token,$oauth_access_token_secret);

$tweets = $twitter->get('https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name='.$buscar.'&count=100');
```

Ilustración 22. Obtener tweets

En el ejemplo anterior se puede ver como primero se realiza la conexión con Twitter con las claves que se han mencionado en el párrafo anterior, estas claves se guardaran por seguridad en el archivo “secret.php”. Después se pide a Twitter mediante la función “get” los últimos tweets usando como palabra clave la búsqueda que ha introducido el usuario, obteniendo los tweets en formato JSON y se almacenándolos en la variable $tweets.

6.2.5. Integración de Google Maps

Para poder mostrar el mapa en la interfaz principal, así como poner los marcadores en dicho mapa hay que utilizar la API de Google Maps. La API de Google Maps se encuentra alojada en los servidores de Google. Para poder cargar la API de Google Maps se debe hacer una referencia desde el archivo "index.php" hacia el lugar en el que se encuentra ésta (Ilustración 23).

```
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>
```

Ilustración 23. Referencia a API de Google Maps

La API de Google Maps requiere que se indique si la aplicación que la está usando utiliza algún tipo de sensor, por ejemplo para determinar la ubicación del usuario a través de un localizador de GPS. Esto es especialmente útil en las aplicaciones pensadas para dispositivos móviles. En este caso no se utiliza ningún tipo de sensor por lo que el valor utilizado será "false". En caso contrario se pondría el valor "true".

Una vez hecha la referencia, a través del archivo JavaScript "localización.js", se manejarán las funciones necesarias para interactuar con Google Maps.

```
geocoder.geocode( { 'address': region}, function(results, status) {
    if (status == google.maps.GeocoderStatus.OK) {

        //alert(results[0].geometry.location);
        var googleLatAndLong = results[0].geometry.location;

        var mapOptions = {
            zoom: zoom,
            center: googleLatAndLong,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };

        var mapDiv = document.getElementById("mapa");
        map = new google.maps.Map(mapDiv, mapOptions);
```

Ilustración 24. Cargar mapa

En la Ilustración 24 puede verse como se carga el mapa en la interfaz principal de la aplicación. Lo que se hace es geocodificar una dirección en Google Maps, de este modo se obtiene un mapa de ubicación desde una simple dirección, en este caso la dirección irá en la variable región. Al iniciar la página, la dirección por defecto será España, cuando un usuario realiza una búsqueda puede elegir otra región si quiere, cambiando así el mapa que se muestra en pantalla. Para poder mostrar el mapa, es necesario definir sus propiedades de visualización:

- zoom: Define el nivel inicial de zoom del mapa.
- center: Define el centro del mapa mediante unas coordenadas.
- mapTypeId: Define el tipo de mapa que será cargado inicialmente.

6.2.6. Mostrar tweets en el mapa

Una vez explicado cómo hacer la conexión y obtener tweets de Twitter, además de cómo mostrar un mapa de Google Maps, esta sección explicará cómo unir las dos funcionalidades.

Para mostrar los tweets en el mapa habrá que hacer una búsqueda previa. Como ya se ha dicho, no siempre que se realiza una búsqueda se hace una solicitud a la API de Twitter, ya que ésta limita las peticiones a 150 a la hora. Por lo que se optó por pedir los tweets con la primera búsqueda del día de la palabra clave, o cada vez que se busque 10 veces. Además de la actualización que hace cada hora el "reloj" con todas las búsquedas diarias. Una vez obtenidos los datos nuevos de Twitter, éstos se tratarán e introducirán en la base de datos.

Al tratar con los tweets surgió un dilema. No todos los tweets tenían geolocalización, ya que para ello es necesario que el usuario de Twitter active la opción en la configuración de su cuenta "Añadir una ubicación a mis tweets", aparte de tener el GPS activado si tuitea desde el móvil. Por este motivo, muchos tweets no eran guardados en la base de datos y por lo tanto no se mostraban en el mapa. Para resolver este problema, se decidió usar la ubicación de registro del usuario para poder obtener las coordenadas del tweet. Esto se consigue utilizando la API de Google Maps en el archivo PHP que trata los tweets, se le pasará una dirección y Google devolverá sus coordenadas (Ilustración 25).

```php
$address = str_replace(" ", "+", $address); // replace all the white space with "+" sign to match with
google search pattern

$url = "http://maps.google.com/maps/api/geocode/json?sensor=false&address=$address";

$response = file_get_contents($url);

$json = json_decode($response,TRUE); //generate array object from the response from the web

if (isset($json['results'][0]['geometry']['location']['lat'])) {

    $lat = $json['results'][0]['geometry']['location']['lat'];
    $lng = $json['results'][0]['geometry']['location']['lng'];
    mysql_query("INSERT INTO ubicaciones(ubicacion,latitud,longitud) VALUES ($address, $lat, $lng)",$link);

    return array($lat, $lng);
} else {
    return null;
}
```

Ilustración 25. Pedir coordenadas de una ubicación

A partir de aquí volvió a surgir otro problema, ya que como eran muchos los tweets que no tenían coordenadas, había que pedir a Google Maps muchas veces las coordenadas de distintas ubicaciones. Para solucionar esto, se creó una tabla en la base de datos que almacenase las distintas ubicaciones con su latitud y longitud. De esta manera, si la ubicación ya existía en la base de datos, no se pedían sus coordenadas a Google.

Otro asunto a tratar fue el siguiente, si varios tweets compartían ubicación, al tener las mismas coordenadas se pondrían en el mismo lugar del mapa, viéndose solo el último en ser puesto. Para arreglar esto, lo que se hizo fue mover un poco la

ubicación de los tweets que comparten ubicación, así el usuario podría ver todos los tweets en el mapa. Después de realizar varias pruebas, se escogió un desplazamiento que mueve la latitud y la longitud sumando un número aleatorio que va de -0,001 a 0,001, se consideró adecuado este rango ya que mueve el marcador a una distancia cercana de su posición original, pero lo suficientemente grande como para poder ver todos los marcadores movidos al hacer zoom sobre el mapa. Este proceso (jittering[4]) puede verse en la siguiente imagen (Ilustración 26).

```php
function getCoordinates($address){
    include ("secret.php");
    $link = mysql_connect($servidor,$usuario,$password);
    mysql_query("SET NAMES utf8");
    mysql_select_db("twitter",$link);

    $ubicacion = mysql_query("SELECT latitud,longitud FROM `ubicaciones` WHERE ubicacion=$address");

    if (mysql_num_rows($ubicacion)>0) {
        /*echo 'si';
        echo '<br>';*/
        $row_tb=mysql_fetch_assoc($ubicacion);
        $lat = $row_tb['latitud'];
        $lng = $row_tb['longitud'];
        $lat = $lat + (rand(0,200)-100)/100000;
        $lng = $lng + (rand(0,200)-100)/100000;
        return array($lat, $lng);
```

Ilustración 26.Pedir y mover coordenadas

Una vez introducida la información necesaria en la base de datos, a continuación se recogerán todos los tweets que cumplan las condiciones de búsqueda hecha por el usuario de la base de datos. Con estos datos se crea una variable PHP ($res), que contendrá un array (puntos) en formato JavaScript con toda la información necesaria de los tweets obtenidos. Esto se hace porque al cargar la página "index.php" nuevamente con la búsqueda, se cargará la variable $res, pero para poder manejar los datos y pintar los marcadores en el mapa es necesario que la información esté en formato JavaScript, ya que como se ha dicho antes las funciones de la API de Google Maps se manejan desde JavaScript. De esta manera se consigue que la funcionalidad de obtener tweets de Twitter y/o de la base de datos, realizada en PHP, y otra como dibujar marcadores en un mapa de Google Maps realizada en JavaScript, puedan interactuar entre ambas. En la Ilustración 27 se realiza este proceso.

[4] Few, S. (2008). Solutions to the Problem of Over-plotting in Graphs. Visual Business Intelligence Newsletter. https://www.perceptualedge.com/articles/visual_business_intelligence/over-plotting_in_graphs.pdf

```php
$puntos = mysql_query("SELECT id,texto,latitud,longitud,fecha,usuario FROM `tweets` WHERE (busqueda LIKE $buscarBD OR texto LIKE $buscarBD)
AND fecha>=DATE_ADD(NOW(),$tiempo)");

if (mysql_num_rows($puntos) != false) {
    $res = "var puntos = [";
    while ($row_tb=mysql_fetch_assoc($puntos)) {

        $rec_id = $row_tb['id'];
        $rec_texto = $row_tb['texto'];
        $rec_latitud = $row_tb['latitud'];
        $rec_longitud = $row_tb['longitud'];
        $rec_fecha = $row_tb['fecha'];
        $rec_usuario = $row_tb['usuario'];

        $rec_texto = convertUrlToLink($rec_texto);

        $res .= "{latitud: ". $rec_latitud . ",  longitud: ". $rec_longitud . ", texto: '". $rec_texto . "', usuario: '". $rec_usuario . "',
        id: '". $rec_id . "', fecha: '". $rec_fecha . "'},";

    }
    $res = rtrim($res, ","); // eliminar última coma
    $res .= "];";
```

Ilustración 27. Creación de variable $res

En la siguiente imagen puede observarse como se cargan variables en la interfaz principal, aparte de la variable $res que contiene toda la información de los tweets obtenidos en una búsqueda, se cargan otras variables. Una de ellas es la región que se ha seleccionado en los filtros de la búsqueda, que se utilizará cargar el mapa de ese lugar. Otra es la alerta, que si está inicializada significará que no hay resultados para la búsqueda realizada, con lo que debe saltar una alerta para avisar al usuario. Por ultimo está la búsqueda, qué recuerda al usuario la última búsqueda realizada y que filtros utilizó para ella.

```php
<script><?php
    echo $res,$region,$alerta,$busqueda;
?></script>
```

Ilustración 28. Cargar variables en la interfaz principal

Cuando se recarga la página, el archivo localizacion.js comprobará si la variable "puntos" está inicializada, si es así, recorrerá el array dibujando un marcador por cada tweet introducido en dicha variable.

```javascript
for (var i = 0;i < puntos.length;i++) {
    googleLatAndLong = new google.maps.LatLng(puntos[i].latitud,puntos[i].longitud);
    fecha = puntos[i].fecha;
    porcion_fecha = fecha.substring(8, 10) + fecha.substring(4, 7) + "-" + fecha.substring(0, 4);
    fecha = porcion_fecha + fecha.substring(10, 19);
    if (puntos[i].usuario != "") {
        contenido = "<b><a target="+'"'+"_blank"+'"'+ "href="+'"'+"https://twitter.com/".concat(puntos[i].usuario).concat("/status/")
        .concat(puntos[i].id)+'"'+ ">Tweet: </a></b>" + puntos[i].texto + "</br>" + "<b>Fecha y hora: </b>" + fecha;
    } else {
        contenido = "<b>Tweet: </b>" + puntos[i].texto + "</br>" + "<b>Fecha y hora: </b>" + fecha;
    }
    addMarker(map, googleLatAndLong, titulo, contenido, porcion_fecha, hoy);

}
```

Ilustración 29. Poner marcadores en el mapa

Como se puede ver en la Ilustración 29, se pone un marcador en el mapa de la interfaz principal en una posición concreta (usando latitud y longitud) mediante la función “addMarker”. Como ya se ha comentado, el marcador contiene el texto del tweet que lo representa y la fecha y hora en la que fue escrito. También contará con un enlace al tweet original.

```
function addMarker(mapa, latlong, titulo, contenido, fecha, hoy)
{
    var pinImage;
    var difFecha = restaFechas(fecha, hoy);

    if (difFecha == 0) {
        //pinImage = new google.maps.MarkerImage("http://www.gooc
    } else if (difFecha <= 7) {
        pinImage = "images/spotlight-poi2.png";
    } else if (difFecha <= 30) {
        pinImage = "images/spotlight-poi_blue.png";
    } else if (difFecha <= 365) {
        pinImage = "images/spotlight-poi_grey.png";
    } else {
        pinImage = "images/spotlight-poi_yellow.png";
    }
```

Ilustración 30. Distintos tipos de marcador

En la imagen anterior, se puede apreciar como utilizando la fecha actual y la fecha en la que fue escrito el tweet, se obtiene la diferencia en días de las dos fechas. Dependiendo de cuánto tiempo haya pasado se pondrá un marcador diferente (Ilustración 31). Terminando así el proceso de mostrar los tweets en el mapa.

Ilustración 31. Distintos tipos de marcadores

7. Verificación y Evaluación

En este apartado se va a proceder a especificar las pruebas que se han realizado para comprobar que la aplicación funciona correctamente y cumple con las funcionalidades descritas anteriormente.

Para conseguir la conformidad total en cada una de las funcionalidades, se han hecho diversas iteraciones en todas ellas hasta conseguir el resultado esperado. Una vez terminada una funcionalidad, se da por terminada y se comienza con la siguiente.

Una vez terminada la aplicación, se han vuelto a repetir todas las pruebas para comprobar que siguen superándose cada una de ellas.

A continuación se detallan las pruebas realizadas por cada funcionalidad.

7.1. Integración de Google Maps

Funcionalidad que integra un mapa de Google Maps en la aplicación.

Iteración 1

Código de prueba	Descripción	Resultado esperado	Resultado
1.1	Ver mapa de España al iniciar la aplicación utilizando la API de Google Maps.	Se muestra el mapa en la interfaz principal de Twitter Finder.	**Correcto**.

Tabla 27. Integración Google Maps – Iteración 1

Iteración 2

Una vez comprobado que el mapa se visualiza correctamente, se añade la funcionalidad de mostrar el mapa por regiones de España.

Código de prueba	Descripción	Resultado esperado	Resultado
1.1	Ver mapa de España al iniciar la aplicación utilizando la API de Google Maps.	Se muestra el mapa en la interfaz principal de Twitter Finder.	**Correcto**.
1.2	El usuario al realizar una búsqueda, puede elegir mostrar el mapa de España o de sus distintas regiones.	Se muestra el mapa de la región seleccionada por el usuario al realizar una búsqueda en la interfaz principal.	**Error**. En algunas regiones como “La Rioja” se muestra otro lugar. Existe “La Rioja” en Argentina.

Tabla 28. Integración Google Maps – Iteración 2

Iteración 3

Se modifica el valor de las opciones del filtro "País o región", especificando que son las comunidades autónomas de España las que tienen que mostrarse en el mapa (Ejemplo: La Rioja, España).

Código de prueba	Descripción	Resultado esperado	Resultado
1.1	Ver mapa de España al iniciar la aplicación utilizando la API de Google Maps.	Se muestra el mapa en la interfaz principal de Twitter Finder.	**Correcto**.
1.2	El usuario al realizar una búsqueda, puede elegir mostrar el mapa de España o de sus distintas regiones.	Se muestra el mapa de la región seleccionada por el usuario al realizar una búsqueda en la interfaz principal.	**Correcto**.

Tabla 29. Integración Google Maps – Iteración 3

7.2. Buscar tweets por palabra clave

Funcionalidad que incluye la recogida de tweets por palabra clave de Twitter, inserción y obtención de los datos necesarios de la base de datos para realizar esta funcionalidad y mostrar cada tweet en el mapa.

Iteración 1

Código de prueba	Descripción	Resultado esperado	Resultado
2.1	Añadir las búsquedas realizadas en la base de datos.	Se añaden las búsquedas en la base de datos correctamente.	**Error**. Las palabras con tildes o "ñ" se introducen mal, aparecen con caracteres "raros" a pesar de que el cotejamiento de la base de datos y de los campos es "UTF-8".
2.2	Obtener los últimos tweets de Twitter que contengan una palabra clave, utilizando la API de Twitter.	Recibir todos los tweets que tengan la palabra clave de la búsqueda y mostrarlos en pantalla.	**Correcto**.
2.3	Añadir los tweets válidos (que contengan latitud y longitud) a la base de datos.	Se añaden los tweets correctamente en la base de datos.	**Error**. Los textos de los tweets con tildes o "ñ" se introducen mal, aparecen con caracteres "raros" a pesar de que el

			cotejamiento de la base de datos y de los campos es “UTF-8”.
2.4	Mostrar todos los tweets en el mapa que coincidan con los filtros de búsqueda.	Se coloca un marcador en el mapa por cada tweet.	**Correcto.**
2.5	Introducir en cada marcador la información requerida del tweet.	Al pulsar en un marcador se verá el texto del tweet, además de la fecha y la hora en la que fue escrito.	**Correcto**.
2.6	Dependiendo del rango de tiempo que haya pasado entre el día de la búsqueda y la publicación del tweet, los tweets se mostrarán diferentes en el mapa.	El tipo de marcador es el adecuado para cada tweet.	**Correcto**.

Tabla 30. Buscar tweets por palabra clave – Iteración 1

Iteración 2

Se soluciona el error de cotejamiento, después de hacer la conexión a la base de datos en el archivo PHP pertinente, se escoge UTF-8 como cotejamiento para la base de datos mediante la query “mysql_query("SET NAMES utf8")”. Para los tweets que no tienen coordenadas, se usará la ubicación de registro de usuario para obtener una la latitud y longitud y así poder posicionar el tweet en el mapa.

Código de prueba	Descripción	Resultado esperado	Resultado
2.1	Añadir las búsquedas realizadas en la base de datos.	Se añaden las búsquedas en la base de datos correctamente.	**Correcto**.
2.2	Obtener los últimos tweets de Twitter que contengan una palabra clave, utilizando la API de Twitter.	Recibir todos los tweets que tengan la palabra clave de la búsqueda y mostrarlos en pantalla.	**Correcto**.
2.3	Añadir los tweets válidos (que contengan latitud y longitud) a la base de datos.	Se añaden los tweets correctamente en la base de datos.	**Error**. El texto de los tweets que contienen una comilla simple, o nº impar de comillas simples, provocan un error al insertarlos en

			la base de datos. Ya que el campo "texto" es un varchar y se introduce entre dos comillas simples.
2.4	Mostrar todos los tweets en el mapa que coincidan con los filtros de búsqueda.	Se coloca un marcador en el mapa por cada tweet.	**Correcto.**
2.5	Introducir en cada marcador la información requerida del tweet.	Al pulsar en un marcador se verá el texto del tweet, además de la fecha y la hora en la que fue escrito.	**Correcto**.
2.6	Dependiendo del rango de tiempo que haya pasado entre el día de la búsqueda y la publicación del tweet, los tweets se mostrarán diferentes en el mapa.	El tipo de marcador es el adecuado para cada tweet.	**Correcto**.
2.7	En los tweets sin geolocalización, se usa la localización de registro de usuario para obtener unas coordenadas.	Se obtienen unas coordenadas de una dirección utilizando la API de Google Maps.	**Error**. Cuando hay muchos tweets sin coordenadas, se hacen muchas llamadas a la API de Google Maps. Esto provoca un error de tiempo de espera en la aplicación.

Tabla 31. Buscar tweets por palabra clave – Iteración 2

Iteración 3

Se decide sustituir todas las comillas simples por espacios en blanco en los textos de los tweets. Para evitar posibles errores de este tipo al almacenar las búsquedas, se restringe el campo de búsqueda no permitiendo hacer una búsqueda con comillas simples. En cuanto al error por tiempo de espera, para no hacer tantas llamadas a la API de Google Maps se decide ir almacenando todas las ubicaciones y sus coordenadas en la base de datos. De manera que solo se haría la llamada cuando La ubicación no exista en la base de datos. Se añaden, un contador de los tweets que se consiguen al hacer una búsqueda y un enlace al tweet original en el marcador que representa dicho tweet.

Código de prueba	Descripción	Resultado esperado	Resultado
2.1	Añadir las búsquedas realizadas en la base de datos.	Se añaden las búsquedas en la base de datos correctamente.	**Correcto**.
2.2	Obtener los últimos tweets de Twitter que contengan una palabra clave, utilizando la API de Twitter.	Recibir todos los tweets que tengan la palabra clave de la búsqueda y mostrarlos en pantalla.	**Correcto**.
2.3	Añadir los tweets válidos (que contengan latitud y longitud) a la base de datos.	Se añaden los tweets correctamente en la base de datos.	**Correcto**.
2.4	Mostrar todos los tweets en el mapa que coincidan con los filtros de búsqueda.	Se coloca un marcador en el mapa por cada tweet.	**Correcto.**
2.5	Introducir en cada marcador la información requerida del tweet.	Al pulsar en un marcador se verá el texto del tweet, además de la fecha y la hora en la que fue escrito.	**Correcto**.
2.6	Dependiendo del rango de tiempo que haya pasado entre el día de la búsqueda y la publicación del tweet, los tweets se mostrarán diferentes en el mapa.	El tipo de marcador es el adecuado para cada tweet.	**Correcto**.
2.7	En los tweets sin geolocalización, se usa la localización de registro de usuario para obtener unas coordenadas.	Se obtienen unas coordenadas de una dirección utilizando la API de Google Maps.	**Correcto**.
2.8	Se cuentan todos los tweets encontrados en la base de datos al realizar una búsqueda.	Se muestra el número de tweets encontrados en una búsqueda.	**Correcto**.
2.9	Se añade un enlace al tweet original en el marcador.	Al pulsar el enlace, se abre una pestaña nueva en el navegador donde aparece el tweet original.	**Correcto**.

Tabla 32. Buscar tweets por palabra clave – Iteración 3

7.3. Actualización automática de las búsquedas diarias

Se actualizará cada hora automáticamente la información de las búsquedas que se han hecho en un día, guardando esa información en la base de datos.

Iteración 1

Código de prueba	Descripción	Resultado esperado	Resultado
3.1	Recoger de la base de datos todas las búsquedas realizadas en el día actual.	Se recogen en una variable todas las búsquedas del día.	**Correcto**.
3.2	Obtener los últimos tweets que contengan en su texto las búsquedas recogidas de la base de datos.	Recibir todos los tweets que tengan en su texto las búsquedas recogidas de la base de datos.	**Correcto**.
3.3	Añadir todos los tweets con geo-posición o con una ubicación de registro de usuario válida (que exista esa localidad) a la base de datos.	Se añaden los tweets correctamente en la base de datos.	**Correcto**.

Tabla 33. Actualización automática de las búsquedas diarias – Iteración 1

7.4. Buscar tweets por usuario

Funcionalidad que incluye la recogida de tweets por usuario, inserción y obtención de los datos necesarios de la base de datos para realizar esta funcionalidad y mostrar cada tweet en el mapa.

Iteración 1

Código de prueba	Descripción	Resultado esperado	Resultado
4.1	Añadir el nombre de usuario introducido en la búsqueda en la base de datos.	Se añaden el usuario en la base de datos correctamente.	**Correcto**.
4.2	Obtener los últimos tweets que haya escrito un usuario, utilizando la API de Twitter.	Recibir los últimos tweets publicados por el usuario que se ha buscado.	**Correcto**.
4.3	Añadir los tweets con coordenadas a la base de datos.	Se añaden los tweets correctamente en la base de datos.	**Correcto**.

4.4	Mostrar todos los tweets en el mapa que coincidan con los filtros de búsqueda.	Se coloca un marcador en el mapa por cada tweet.	**Correcto.**
4.5	Introducir en cada marcador la información requerida del tweet.	Al pulsar en un marcador se verá el texto del tweet, además de la fecha y la hora en la que fue escrito.	**Correcto**.
4.6	Dependiendo del rango de tiempo que haya pasado entre el día de la búsqueda y la publicación del tweet, los tweets se mostrarán diferentes en el mapa.	El tipo de marcador es el adecuado para cada tweet.	**Correcto**.
4.7	En los tweets sin geolocalización, se usa la localización de registro de usuario para obtener unas coordenadas.	Se obtienen unas coordenadas de una dirección utilizando la API de Google Maps.	**Correcto**.
4.8	Se cuentan todos los tweets encontrados en la base de datos al realizar una búsqueda.	Se muestra el número de tweets encontrados en una búsqueda.	**Correcto**.
4.9	Se añade un enlace al tweet original en el marcador.	Al pulsar el enlace, se abre una pestaña nueva en el navegador donde aparece el tweet original.	**Correcto**.

Tabla 34. Buscar tweets por palabra clave – Iteración 1

7.5. Mostrar búsquedas recomendadas

Se mostrará en la aplicación una serie de búsquedas recomendadas basándose en la información almacenada en ese momento en la base de datos.

Iteración 1

Código de prueba	Descripción	Resultado esperado	Resultado
5.1	Recoger de la base de datos las 10 palabras más buscadas.	Se muestra en la interfaz principal una lista con las 10 palabras más buscadas.	**Correcto**.
5.2	Recoger de la base de datos las 10 palabras que más tweets tengan.	Se muestra en la interfaz principal una lista con las 10 palabras con más tweets.	**Correcto**.
5.3	Recoger de la base de datos los 10 usuarios más buscados.	Se muestra en la interfaz principal una lista con los 10 usuarios más buscadas.	**Correcto**.
5.4	Recoger de la base de datos las 10 palabras más buscadas en el último mes.	Se muestra en la interfaz principal una lista con las 10 palabras más buscadas el último mes.	**Correcto**.

Tabla 35. Mostrar búsquedas recomendadas – Iteración 1

Iteración 2

Se añade una lista de quejas a la base de datos, cuando un usuario realice una búsqueda de una queja que esté en la base de datos, aparecerá en la lista de quejas de ese día.

Código de prueba	Descripción	Resultado esperado	Resultado
5.1	Recoger de la base de datos las 10 palabras más buscadas.	Se muestra en la interfaz principal una lista con las 10 palabras más buscadas.	**Correcto**.
5.2	Recoger de la base de datos las 10 palabras que más tweets tengan.	Se muestra en la interfaz principal una lista con las 10 palabras con más tweets.	**Correcto**.
5.3	Recoger de la base de datos los 10 usuarios más buscados.	Se muestra en la interfaz principal una lista con los 10 usuarios más buscadas.	**Correcto**.
5.4	Recoger de la base de datos las 10 palabras más buscadas en el último mes.	Se muestra en la interfaz principal una lista con las 10 palabras más buscadas el último mes.	**Correcto**.

5.5	Recoger de la base de datos las quejas que se han buscado ese día.	Se muestra en la interfaz principal una lista de quejas buscadas ese día.	**Error**. No se muestra la lista de quejas, ya que no hay espacio en el contenedor de las "Búsquedas recomendadas".

Tabla 36. Mostrar búsquedas recomendadas – Iteración 2

Iteración 3

Se modifica el diseño de la interfaz principal para darle más espacio de ancho al contenedor de "Búsquedas recomendadas".

Código de prueba	Descripción	Resultado esperado	Resultado
5.1	Recoger de la base de datos las 10 palabras más buscadas.	Se muestra en la interfaz principal una lista con las 10 palabras más buscadas.	**Correcto**.
5.2	Recoger de la base de datos las 10 palabras que más tweets tengan.	Se muestra en la interfaz principal una lista con las 10 palabras con más tweets.	**Correcto**.
5.3	Recoger de la base de datos los 10 usuarios más buscados.	Se muestra en la interfaz principal una lista con los 10 usuarios más buscadas.	**Correcto**.
5.4	Recoger de la base de datos las 10 palabras más buscadas en el último mes.	Se muestra en la interfaz principal una lista con las 10 palabras más buscadas el último mes.	**Correcto**.
5.5	Recoger de la base de datos las quejas que se han buscado ese día.	Se muestra en la interfaz principal una lista de quejas buscadas ese día.	**Correcto.**

Tabla 37. Mostrar búsquedas recomendadas – Iteración 3

8. Conclusiones

Una vez finalizado el proyecto, llega la hora de exponer las conclusiones personales sobre el desarrollo de la aplicación. Se comparará el trabajo realizado con el que se previó al comienzo del proyecto, así como se comentará un posible trabajo futuro que se podría realizar para mejorar Twitter Finder.

8.1. Conclusiones de gestión

Respecto a la planificación inicial realizada al comienzo, cabe destacar que ha habido retrasos por temas personales. Entre los que destacan haber sufrido una lesión de espalda y tener el ordenador estropeado.

Como estos contratiempos estaban planificados como posibles riesgos, no han tenido una gran repercusión en el desarrollo del proyecto.

Por otra parte, la fase de gestión ha llevado más tiempo del esperado, concretamente en la tarea “Búsqueda de información”. Esto ha sido debido a que se han tenido que utilizar herramientas nuevas y lenguajes de programación en los que no se tenían muchos conocimientos.

De la misma manera, la fase de implementación también ha sufrido retrasos. A medida que avanzaba el proyecto han surgido errores con los que no se contaban y la solución de los mismos ha llevado tiempo. También se han ido añadiendo nuevas funcionalidades y mejoras para conseguir una aplicación más completa, por lo que se han invertido más horas de las planificadas.

En cuanto a la redacción de la memoria, la realización de cada apartado ha llevado más tiempo del esperado. El hecho de hacer tablas, insertar imágenes, corregir fallos de escritura, etc. Han hecho que se aumenten las horas reales de la memoria.

Como “Realizar una mala planificación” era un riesgo a asumir, se logró reducir el atraso al mínimo.

A pesar de los retrasos sufridos, como había bastante holgura desde la fecha prevista para la finalización del TFG (11/07/2016) hasta la entrega de la memoria, resumen y archivos que componen el proyecto (09/09/2016), se ha podido presentar el TFG en la convocatoria de septiembre tal y como se había planeado desde el principio.

A continuación pueden verse una tabla y un gráfico con las diferencias entre las horas estimadas y las reales (aproximadas) que se han invertido en el proyecto.

TAREA	DURACIÓN estimada (horas)	DURACIÓN Real Aprox. (horas)
Gestión		
Búsqueda de información	30	40
Planificación del proyecto	9	9
Captura de requisitos		
Diagrama de casos de uso	9	12
Modelo de dominio	9	9
Casos de uso extendidos	30	35
Diagrama BD	15	15
Preparación		
Descarga de aplicaciones y programas	6	6
Instalar aplicaciones y programas	15	15
Diseño		
Diseño de la página web	60	70
Diseño de base de datos	6	10
Implementación		
Integración de Google Maps	36	40
Buscar tweets por palabra	51	60
Actualización automática	9	15
Buscar tweets por usuario	27	30
Mostrar búsquedas recomendadas	9	15
Documentación		
Memoria del Proyecto	60	100
Realización de la defensa	9	9
TOTAL	**390**	**490**

Tabla 38. Duración total estimada de las tareas vs real

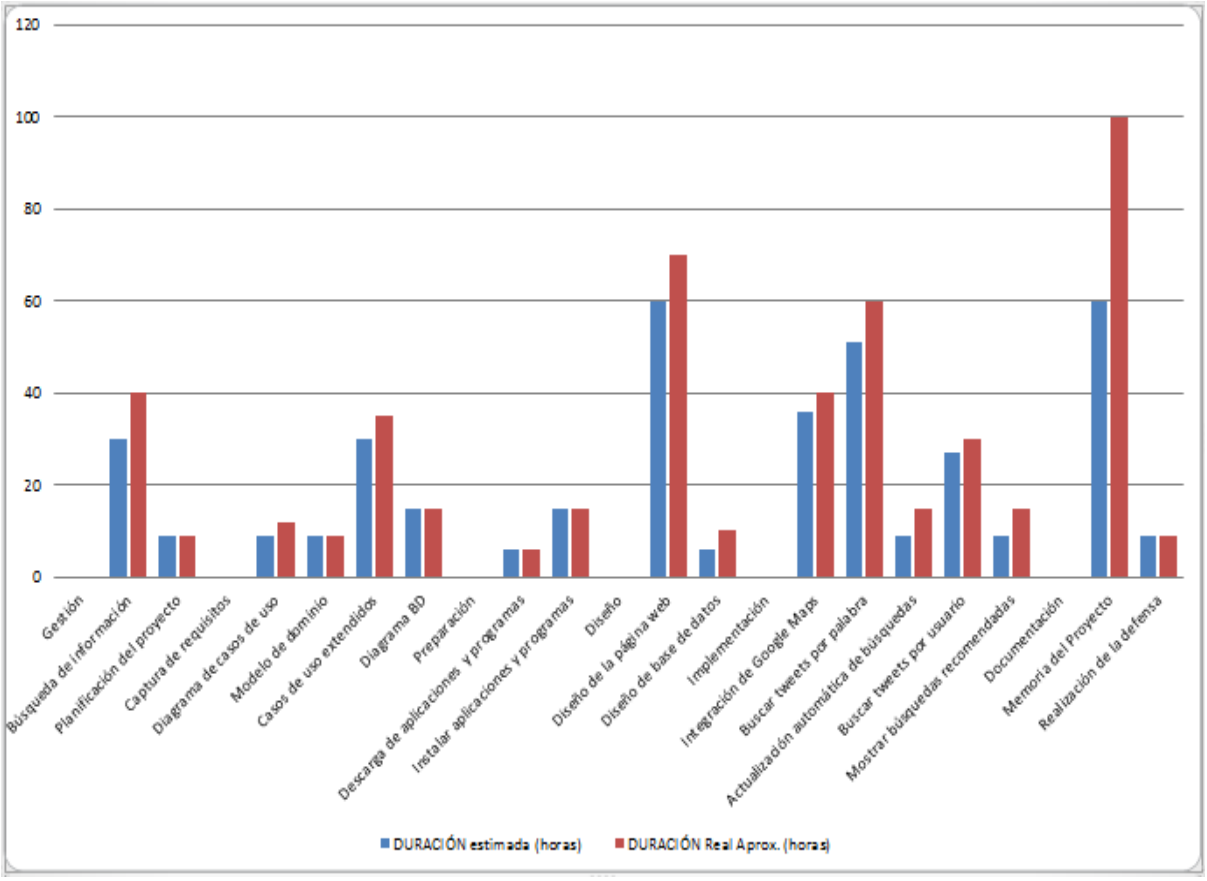


Ilustración 32. Gráfico de planificación

8.2. Cumplimiento de objetivos

Podría decirse que se han cumplido todos los objetivos establecidos al comienzo del proyecto.

- Se ha conseguido ganar mayor experiencia en los lenguajes de programación utilizados para el desarrollo de una aplicación web desde cero, ganando los conocimientos necesarios para realizar otras aplicaciones similares.

- De la misma importancia, es la utilidad de la aplicación web para los usuarios. Presentada a usuarios que residan en España que tengan alguna queja o consulta y quieran saber si hay más gente con el mismo problema y en qué lugar ocurre.

8.3. Conclusiones personales

Es difícil echar la vista atrás y pensar en todo lo realizado para llegar hasta donde estoy ahora. Ha sido un camino duro, lleno de baches y altibajos. Pero el poder realizar un proyecto de principio a fin por cuenta propia y ver el resultado final, me hace estar muy satisfecho.

También he conseguido mejorar mi concentración y mi capacidad de auto-aprendizaje, puntos que considero muy importantes para futuros trabajos en el sector de la informática. Por otra parte, el haber realizado este proyecto, me da la motivación necesaria para afrontar nuevas aplicaciones web con los conocimientos adquiridos.

8.4. Líneas futuras

No se descarta la opción de seguir trabajando en Twitter Finder. Se considera que la aplicación podría mejorarse en los siguientes aspectos para que sea más completa:

- Que la aplicación sea a nivel mundial. En lugar de poder filtrar por región de España, poder elegir cualquier país en el que se use Twitter. De esta forma se conseguirían un mayor número de usuarios.

- Conseguir una mejor visualización de la aplicación en los dispositivos móviles.

- Dar la opción de registrarse al usuario y así que pueda disponer de más opciones, como por ejemplo guardar sus búsquedas favoritas o añadir sugerencias de quejas para el apartado “Quejas de hoy”.

- Añadir banners de publicidad y así empezar a tener algún ingreso.

De las mejoras mencionadas, la que más opciones tiene para su implementación es la primera de ellas. Se considera que no tendrá mucha dificultad y ayudaría a conseguir un mayor número de ingresos al llegar a un mayor número de usuarios.

9. URLs consultadas

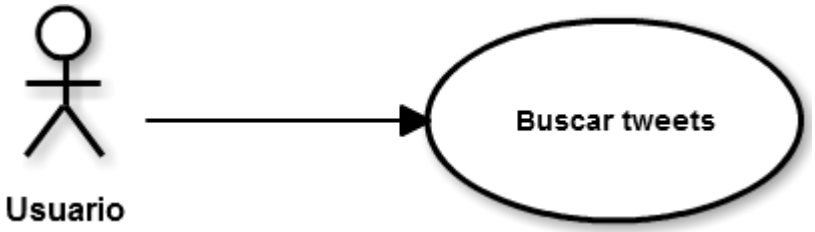
- Documentación sobre la API de Twitter:
  https://dev.twitter.com/overview/documentation
- Manual de PHP:
  http://www.php.net/manual/es/
- Sitio web para encontrar soluciones a problemas de programación:
  http://stackoverflow.com/
- Documentación sobre la API de Google Maps:
  https://developers.google.com/maps/?hl=es
- Tutorial de CSS:
  http://www.w3schools.com/css/
- Tutorial de JavaScript:
  http://www.w3schools.com/js/
- Tutorial de SQL:
  http://www.w3schools.com/sql/
- Tutorial de HTML:
  http://www.w3schools.com/html/
- Videotutorial de introducción a la API de Twitter:
  https://www.youtube.com/watch?v=ZHlpoOZRc0g
- Videotutorial de cómo obtener tweets con la API de Twitter:
  https://www.youtube.com/watch?v=T6v3qUMPawc
- Información sobre campañas publicitarias en internet:
  http://www.marketingguerrilla.es/campanas-de-publicidad-en-internet-cuando-elegir-cpc-cpm-o-cpa-2/
- Documentación sobre phpMyAdmin:
  https://www.phpmyadmin.net/docs/
- Documentacion sobre FileZilla:
  https://wiki.filezilla-project.org/Documentation
- Tutorial de MySQL Workbench:
  http://coba.dc.fi.udc.es/~bd/bd2/MySQLWB/tutorialWB.html

10. Bibliografía

- Cabeza, Y. (2015). Gestión de un taller mecánico. https://addi.ehu.es/bitstream/10810/15204/1/MemoriaTFG_Yasmin_Cabeza.pdf
- Castro, J (2016). Portal de búsqueda de vulnerabilidades. https://www.dropbox.com/s/02mywxoxtqxonr4/PBVW_JCastro_Memoria.pdf?dl=0
- Calzada, I. A. (2016). SurfBizkaiApp.
- González, Y. (2014). Football Manager App.
- Few, S. (2008). Solutions to the Problem of Over-plotting in Graphs. Visual Business Intelligence Newsletter. https://www.perceptualedge.com/articles/visual_business_intelligence/over-plotting_in_graphs.pdf

ANEXO I: CASOS DE USO EXTENDIDOS

Usuario

Buscar tweets

Descripción: El usuario realiza una búsqueda de tweets con los filtros que considere oportunos
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario introducirá un texto en el campo de búsqueda. Ilustración 33. [Si el usuario quiere, cambiará los distintos filtros] 2a. Puede elegir el número del rango de la búsqueda. Ilustración 34. 3a. Puede elegir la unidad temporal del rango de búsqueda. Ilustración 35. 4a. Puede elegir el país o región para la búsqueda. Ilustración 36. 5a. Pulsando en "Opciones avanzadas", puede elegir buscar por nombre de usuario. Ilustración 37. 6. Para realizarla búsqueda el usuario pulsará la lupa situada a la derecha del campo de búsqueda. Ilustración 38. [Si hay tweets] 7. a. Aparecerá en mapa un marcador por cada tweet. Ilustración 39. [Si no hay tweets] 7. b. Saldrá un aviso diciendo que no existen resultados para la búsqueda. Ilustración 40.
Postcondiciones: La búsqueda se guardará en la base de datos. Si hay tweets, tanto la información de los tweets, como las ubicaciones de ellos que no existan en la base de datos también se almacenarán
Interfaz gráfica:

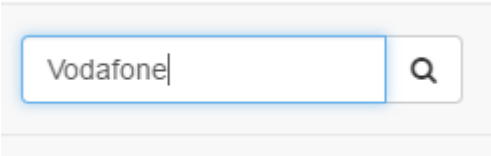


Ilustración 33. Rellenar campo de búsqueda

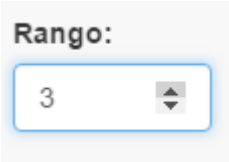


Ilustración 34. Elegir valor del rango

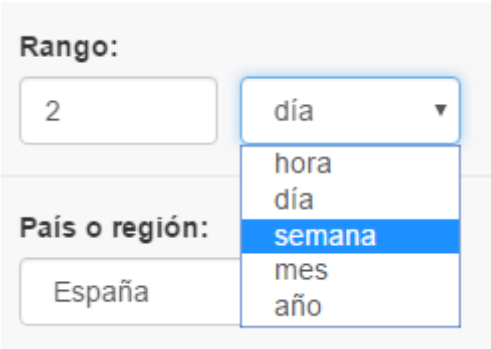


Ilustración 35. Elegir unidad del rango

Ilustración 36. Elegir país o región

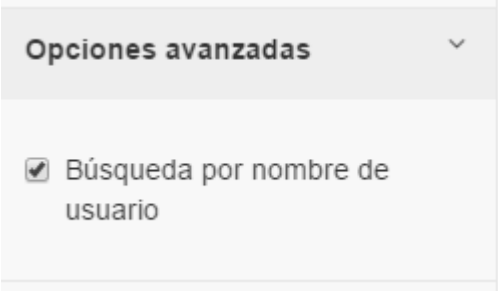


Ilustración 37. Seleccionar buscar por usuario

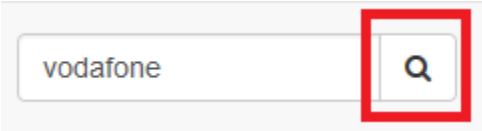


Ilustración 38. Realizar búsqueda

Ilustración 39. *Mapa con tweets*

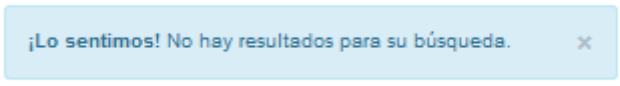
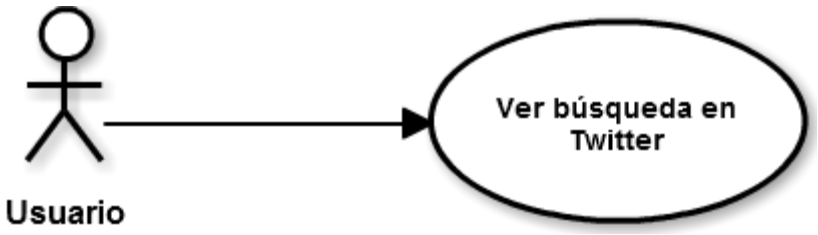


Ilustración 40. Aviso "sin resultados"

Ver búsqueda en Twitter

Descripción: Se abre una pestaña nueva que se redirecciona a la web oficial de Twitter y se verá la búsqueda que se ha realizado desde ahí
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario pulsará sobre el botón “Tu búsqueda en Twitter”. Ilustración 41. [Si se ha hecho una búsqueda previa] 2. a. Se abre la web de Twitter con nuestra búsqueda. Ilustración 42. [Si no se ha hecho una búsqueda previa] 2. b. Se accede a la página principal de Twitter. Ilustración 43.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 41. Ver búsqueda en Twitter*

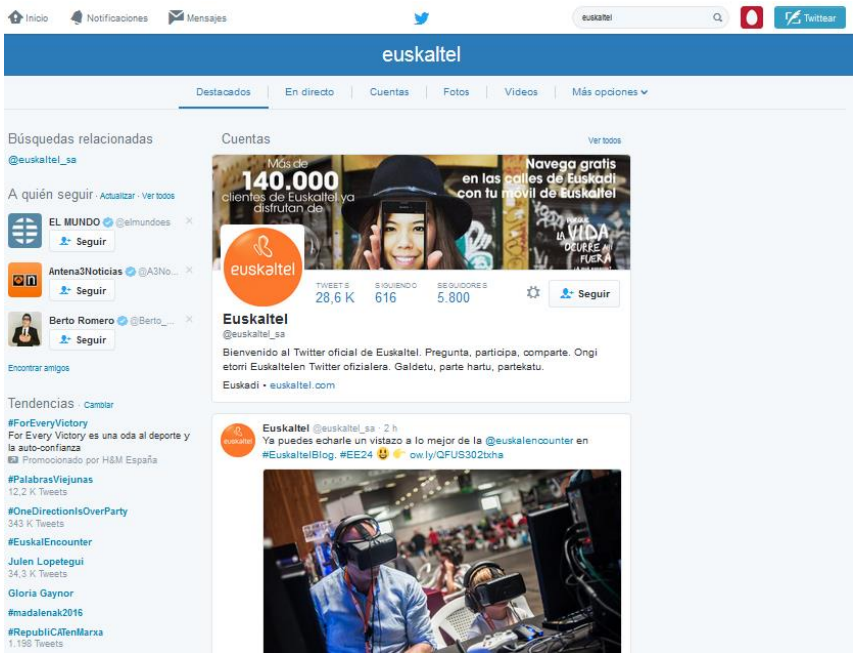


Ilustración 42. Búsqueda "euskaltel" en Twitter

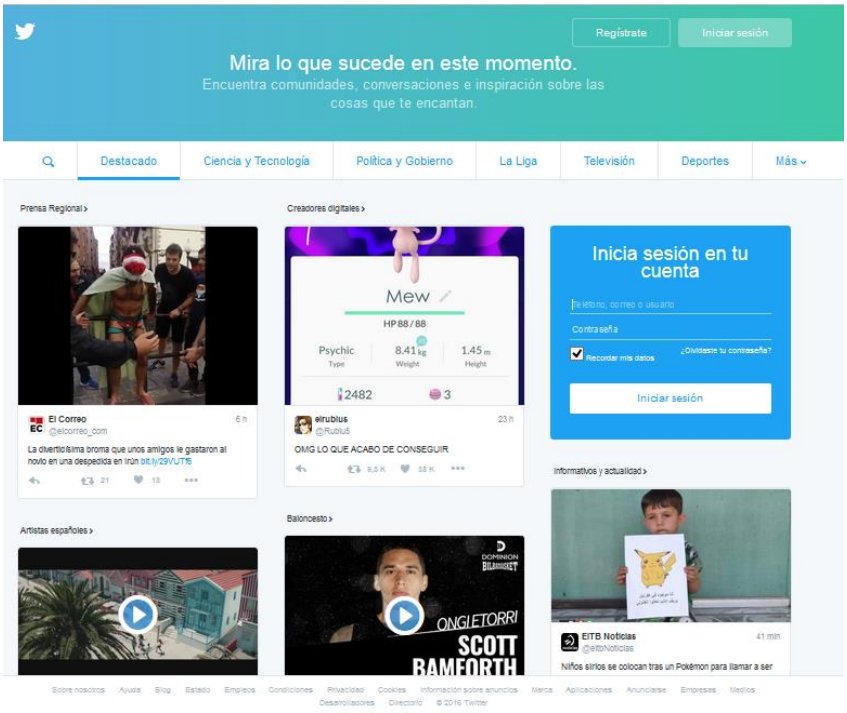


Ilustración 43. Página principal de Twitter

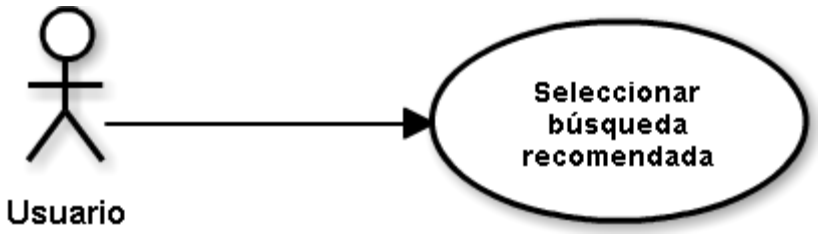
Seleccionar búsqueda recomendada

Descripción: El usuario selecciona una de las opciones que salen en búsquedas recomendadas, se verán ver los tweets del texto seleccionado con un rango de 1 día en España
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre uno de los enlaces que aparecen en el apartado "Búsquedas recomendadas". Ilustración 44. [Si hay tweets] 2. a. Aparecerá en mapa un marcador por cada tweet. Ilustración 39. [Si no hay tweets] 2. b. Saldrá un aviso diciendo que no existen resultados para la búsqueda. Ilustración 40.
Postcondiciones: La búsqueda recomendada se guardará en la base de datos. Si hay tweets, tanto la información de los tweets, como las ubicaciones de ellos que no existan en la base de datos también se almacenarán
Interfaz gráfica: Búsquedas recomendadas **Palabras más buscadas:** 1. dgt 2. accidente 3. retencion 4. concierto 5. vodafone 6. retencion autopista 7. sin adsl 8. adsl 9. movistar 10. sin internet **Palabras con más tweets:** 1. retencion autopista 2. dgt 3. concierto 4. retencion 5. accidente 6. incdgt 7. fiesta 8. vodafone 9. euskaltel 10. orange **Usuarios más buscados:** 1. @InformacionDGT 2. @ertzaintza 3. @yeray991 4. @Lenders__9 5. @juanan 6. @Bizkaibus 7. @evitemosradares 8. @Yerai991 9. @Pablo_Iglesias_ 10. @MyOLLGpurpose **Más buscado este mes:** 1. sin adsl 2. aetjeatjeajeajeaj 3. vodafone 4. Retencion autopista 5. concierto 6. pokemon go error 7. pokemon go fallo 8. liasjgteripayj 9. incdgt 10. sin internet **Quejas de hoy:** 1. sin adsl

Ilustración 44. Seleccionar búsqueda recomendada

Ilustración 39. Mapa con tweets

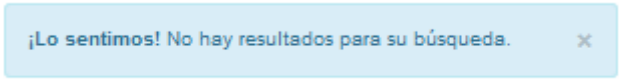
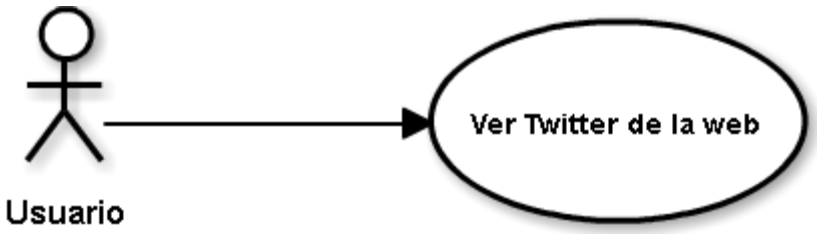
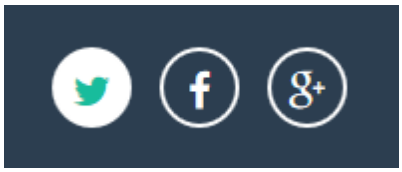


Ilustración 40. Aviso "sin resultados"

Ver Twitter de la web

Descripción: Se abrirá una pestaña nueva en la que se cargará el Twitter de la aplicación web
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre el icono de Twitter, situado en la parte inferior derecha de la aplicación. Ilustración 45. 2. Se abre la página de Twitter de Twitter Finder. Ilustración 46.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 45. Acceder a Twitter de Twitter Finder*

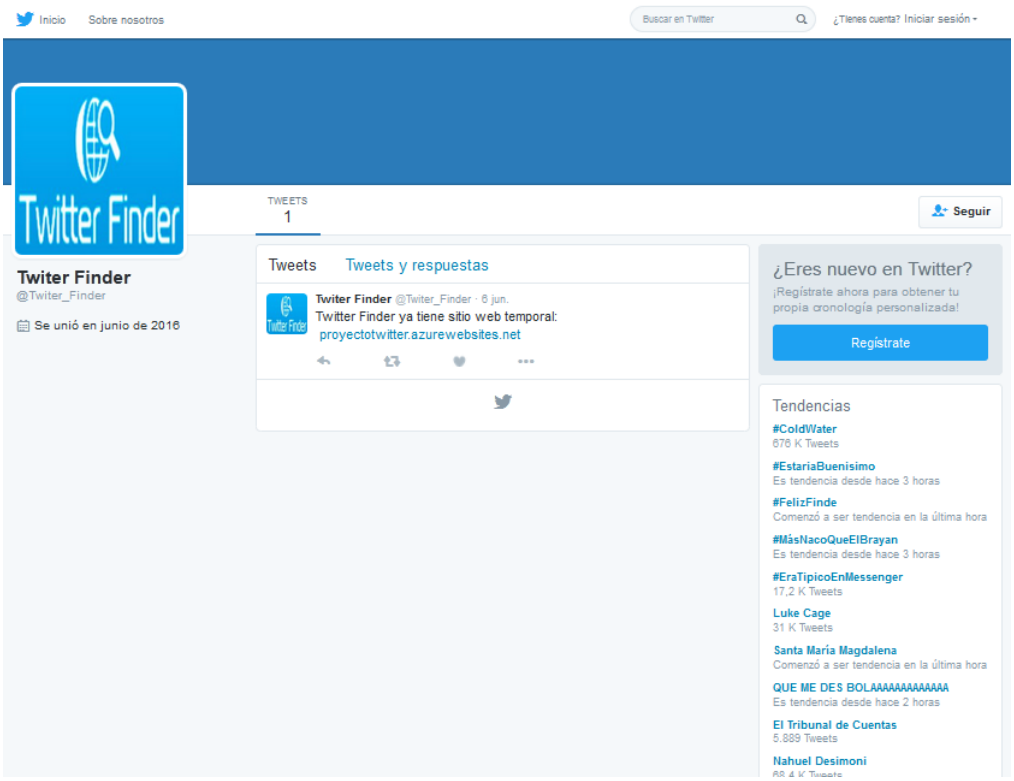


Ilustración 46. Twitter de Twitter Finder

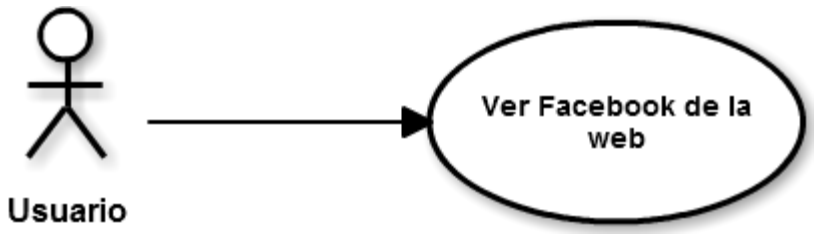
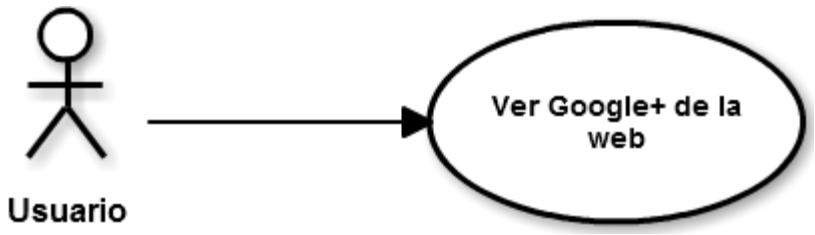
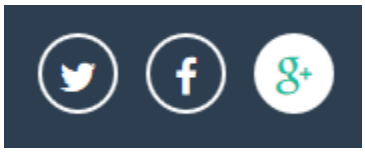
Ver Facebook de la web

Descripción: Se abrirá una pestaña nueva en la que se cargará el Facebook de la aplicación web
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre el icono de Facebook, situado en la parte inferior derecha de la aplicación. Ilustración 47. 2. Se abre la página de Facebook de Twitter Finder. Ilustración 48.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 47. Acceder a Facebook de Twitter Finder*

Ilustración 48. Facebook de Twitter Finder

Ver Google+ de la web

Descripción: Se abrirá una pestaña nueva en la que se cargará el Google+ de la aplicación web
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre el icono de Google+, situado en la parte inferior derecha de la aplicación. Ilustración 49. 2. Se abre la página de Google+ de Twitter Finder. Ilustración 50.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 49. Acceder a Google+ de Twitter Finder*

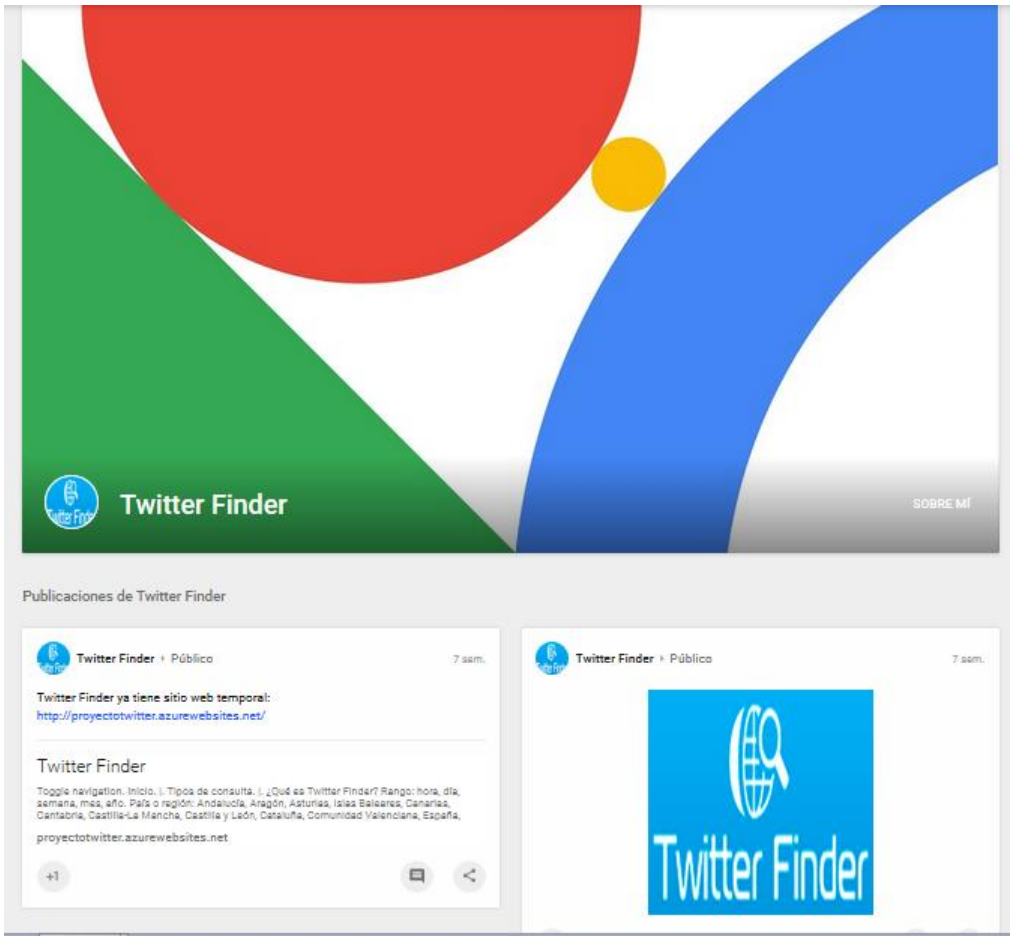


Ilustración 50. Google+ de Twitter Finder

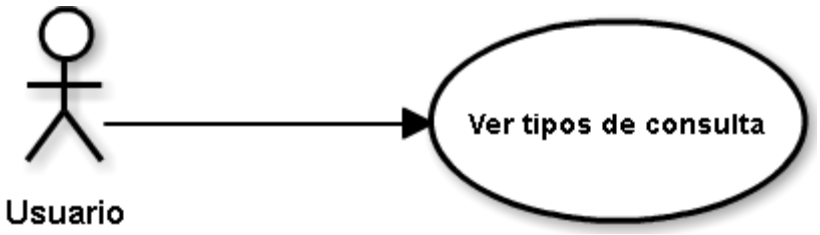
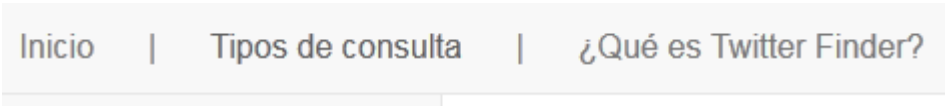
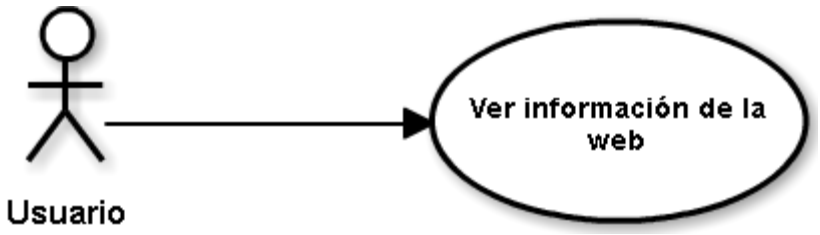
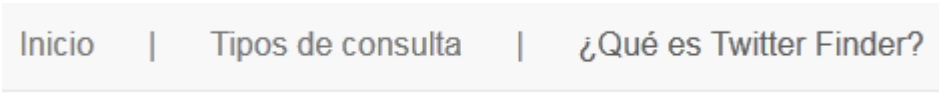
Ver tipos de consulta

Descripción: Se abrirá una página nueva en la que el usuario podrá ver los distintos tipos de consulta que hay para realizar una búsqueda
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre el enlace situado en la cabecera de la página “Tipos de consulta”. Ilustración 51. 2. Se abre una página de la aplicación en la que se muestran los tipos de consulta. Ilustración 18.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 51. Ver tipos de consulta*

Ilustración 18. Página "tipos de consulta"

Ver información de la web

Descripción: Se abrirá una página nueva en la que el usuario podrá ver información de la aplicación web, como de que trata y datos del autor
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre el enlace situado en la cabecera de la página "¿Qué es Twitter Finder?". Ilustración 52. 2. Se abre una página en la que se muestra información de la aplicación web. Ilustración 19.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 52. Ver información de la web*

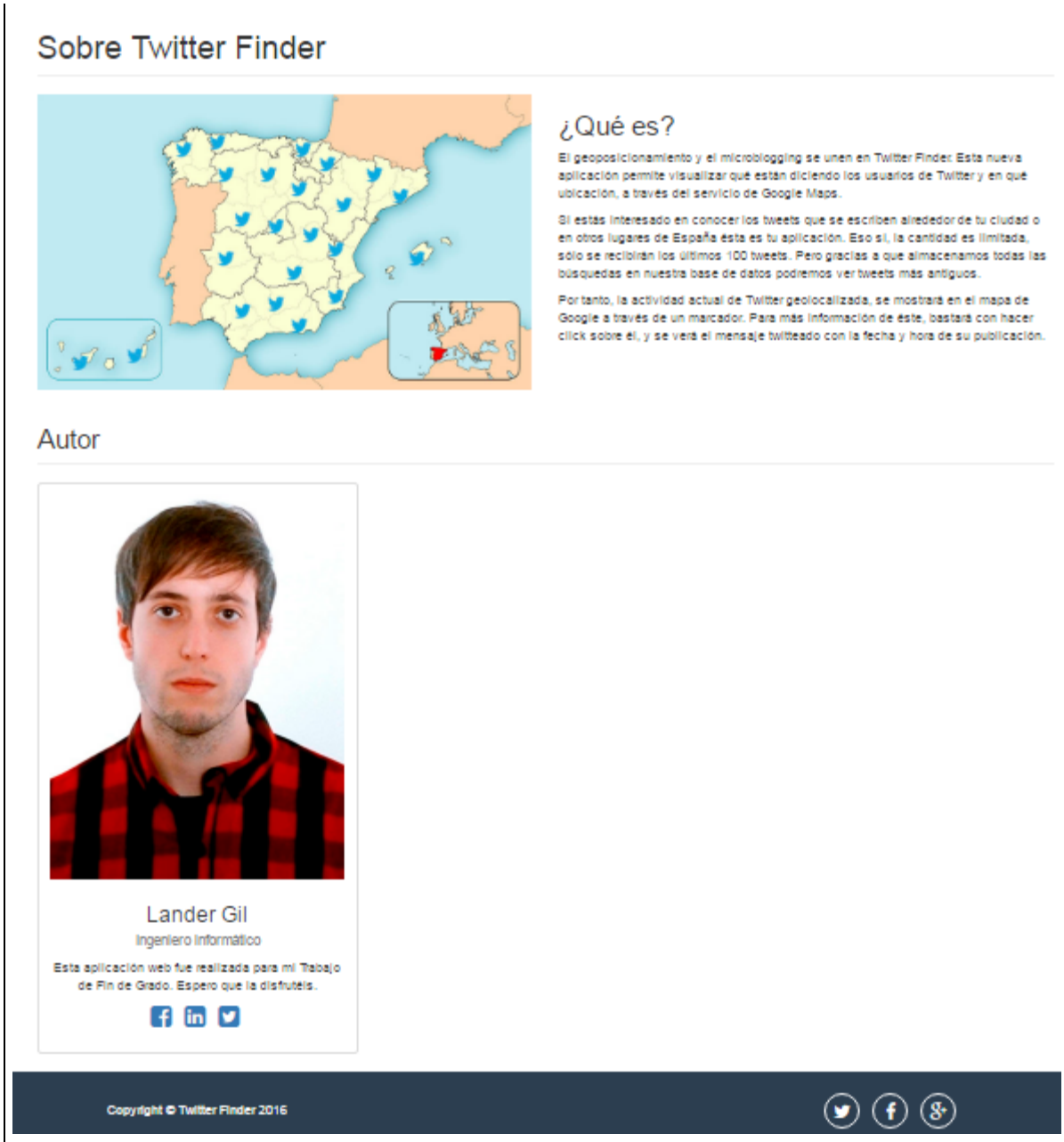
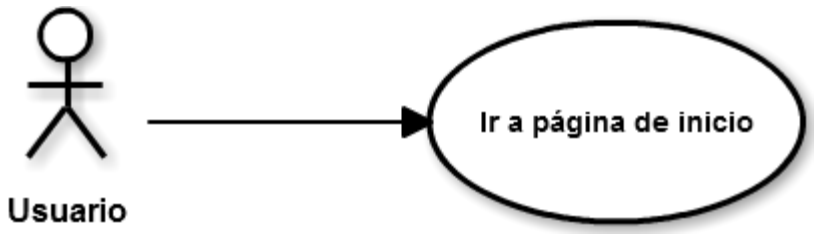
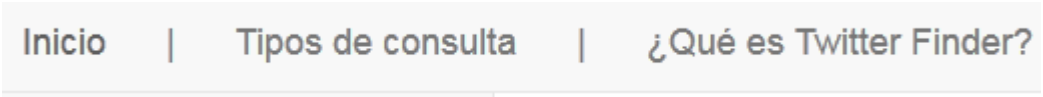


Ilustración 19. Página "¿Qué es Twitter Finder?"

Ir a página de inicio

Descripción: Funcionalidad que llevará al usuario a la página de inicio de Twitter Finder
Actores: Usuario
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click sobre el enlace situado en la cabecera de la página "Inicio". Ilustración 53. 2. Se abrirá la página de inicio de la aplicación. Ilustración 54.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 53. Ir a página de inicio*

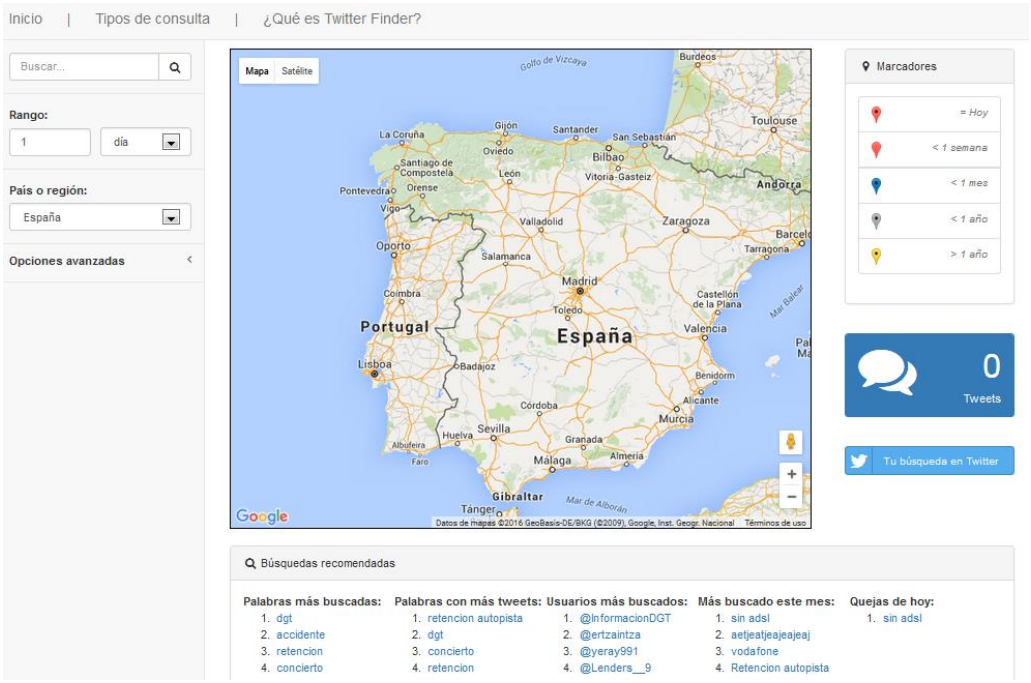
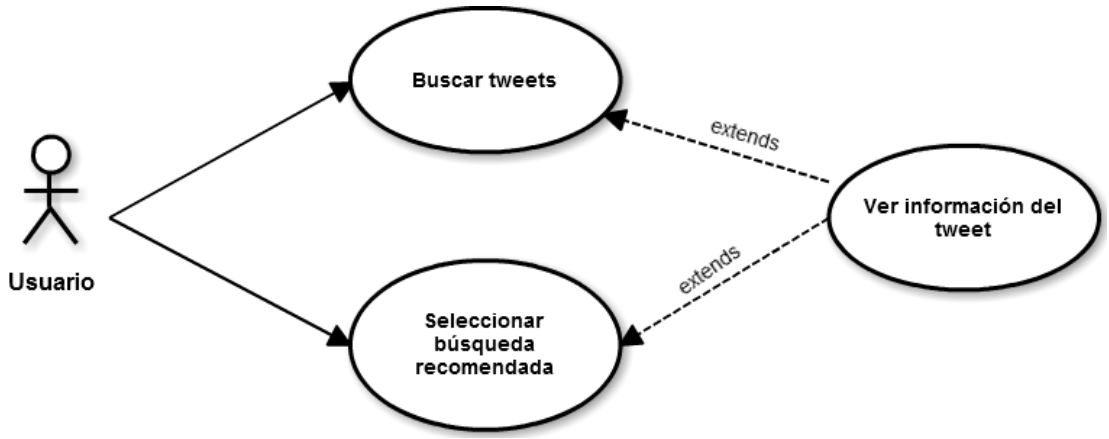
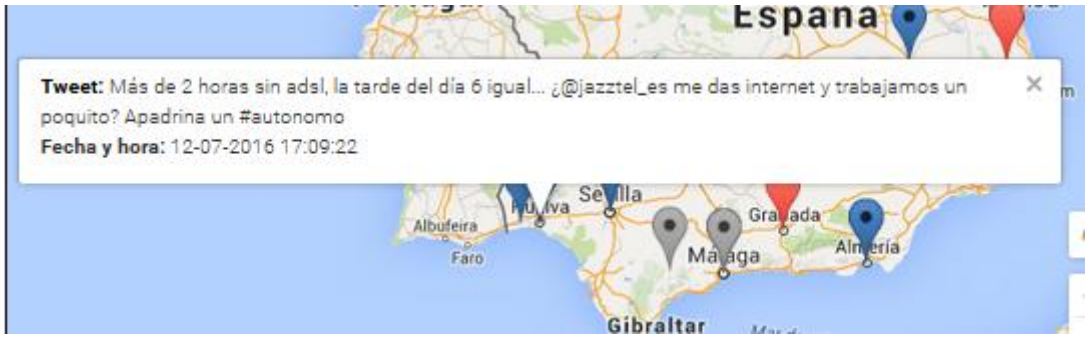


Ilustración 54. Página inicial de Twitter Finder

Ver información del tweet

Descripción: Subcaso de uso. El usuario verá el texto, la fecha y la hora de un tweet
Actores: Usuario
Precondiciones: Haber realizado una búsqueda o seleccionado una búsqueda recomendada, obteniendo algún resultado
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click en uno de los marcadores y podrá ver la información del tweet que representa dicho marcador. Ilustración 55.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 55. Ver información del tweet*

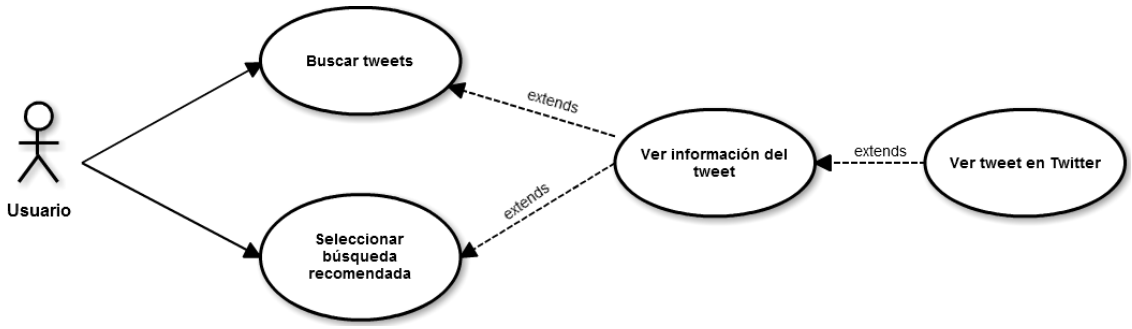
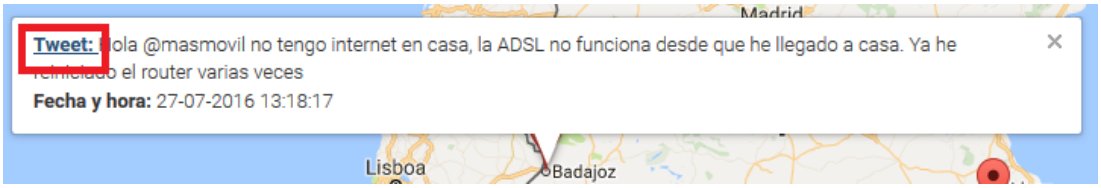
Ver tweet en Twitter

Descripción: Subcaso de uso. El usuario verá tweet original en Twitter
Actores: Usuario
Precondiciones: Haber pulsado un marcador para ver la información del tweet
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click en el enlace que contendrá el marcador. Ilustración 56. 2. Se abrirá una pestaña nueva donde podremos ver el tweet original. Ilustración 57.
Postcondiciones: Ninguna
Interfaz gráfica:    *Ilustración 56. Ver tweet original*

Ilustración 57. Tweet en Twitter

Ver Facebook del autor

Descripción: Subcaso de uso. Se abrirá una pestaña nueva en la que se cargará el Facebook del autor de la aplicación web
Actores: Usuario
Precondiciones: Haber accedido previamente a la página "¿Qué es Twitter Finder?"
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click en el icono de Facebook, situado en la sección "Autor", debajo del nombre del autor de la aplicación web. Ilustración 58. 2. Se abre la página de Facebook del autor. Ilustración 59.
Postcondiciones: Ninguna
Interfaz gráfica:  *Ilustración 58. Acceder a Facebook del autor*

Ilustración 59. Facebook del autor

Ver LinkedIn del autor

Descripción: Subcaso de uso. Se abrirá una pestaña nueva en la que se cargará el LinkedIn del autor de la aplicación web
Actores: Usuario
Precondiciones: Haber accedido previamente a la página "¿Qué es Twitter Finder?"
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click en el icono de LinkedIn, situado en la sección "Autor", debajo del nombre del autor de la aplicación web. Ilustración 60. 2. Se abre la página de LinkedIn del autor. Ilustración 61.
Postcondiciones: Ninguna
Interfaz gráfica: Lander Gil Ingeniero informático Esta aplicación web fue realizada para mi Trabajo de Fin de Grado. Espero que la disfrutéis. *Ilustración 60. Acceder a LinkedIn del autor*

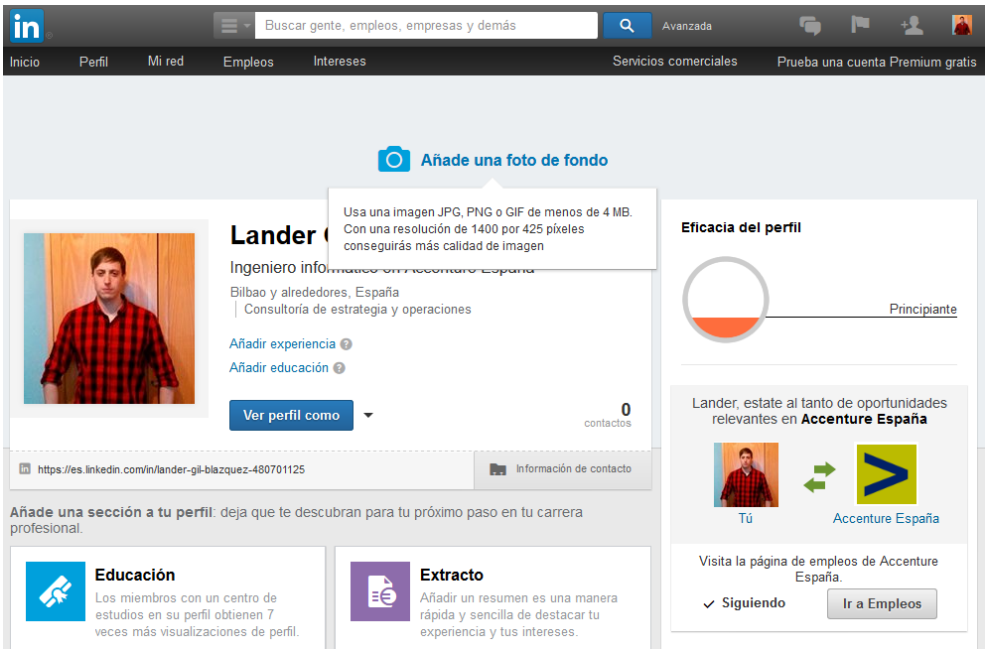


Ilustración 61. LinkedIn del autor

Ver Twitter del autor

Descripción: Subcaso de uso. Se abrirá una pestaña nueva en la que se cargará el Twitter del autor de la aplicación web
Actores: Usuario
Precondiciones: Haber accedido previamente a la página “¿Qué es Twitter Finder?”
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario hará click en el icono de Twitter, situado en la sección “Autor”, debajo del nombre del autor de la aplicación web. Ilustración 62. 2. Se abre la página de Twitter del autor. Ilustración 63.
Postcondiciones: Ninguna
Interfaz gráfica: Lander Gil Ingeniero informático Esta aplicación web fue realizada para mi Trabajo de Fin de Grado. Espero que la disfrutéis. *Ilustración 62. Acceder a Twitter del autor*

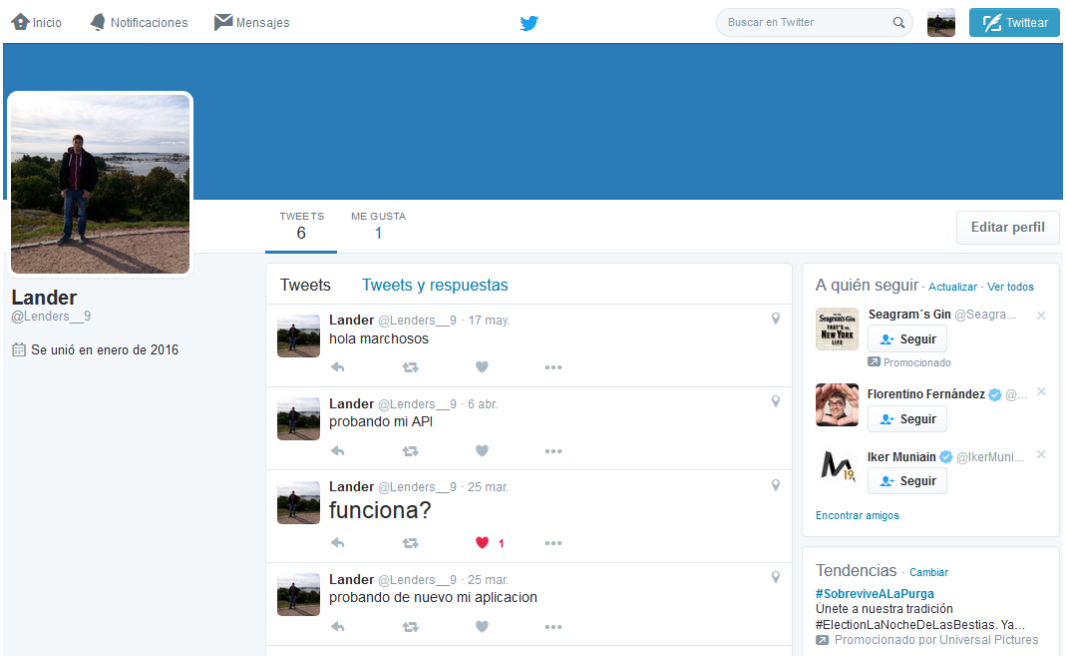
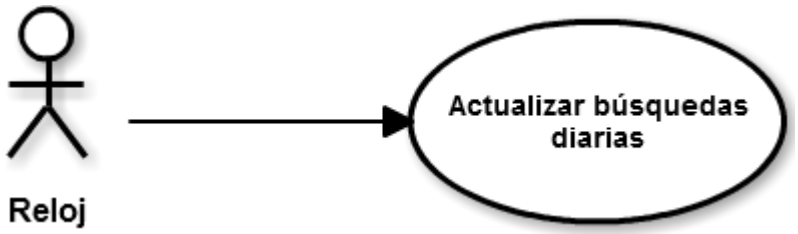


Ilustración 63. Twitter del autor

Reloj

Actualizar búsquedas diarias

Descripción: Tarea programada que se ejecuta cada hora, actualizando todas las búsquedas que se han hecho en el día que se está ejecutando.
Actores: Reloj
Precondiciones: Que en un día se haya realizado como mínimo una búsqueda en la web
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El reloj (script en php), cada una hora coge de la base de datos todas las búsquedas realizadas en el día actual. 2. Con cada una de las búsquedas, a través de la API de Twitter pide tweets nuevos. 3. Los nuevos tweets recibidos los insertará en la base de datos.
Postcondiciones: Se actualizará la base de datos con nueva información
Interfaz gráfica: No existe

ANEXO II: DIAGRAMAS DE SECUENCIA

Mostrar mapa

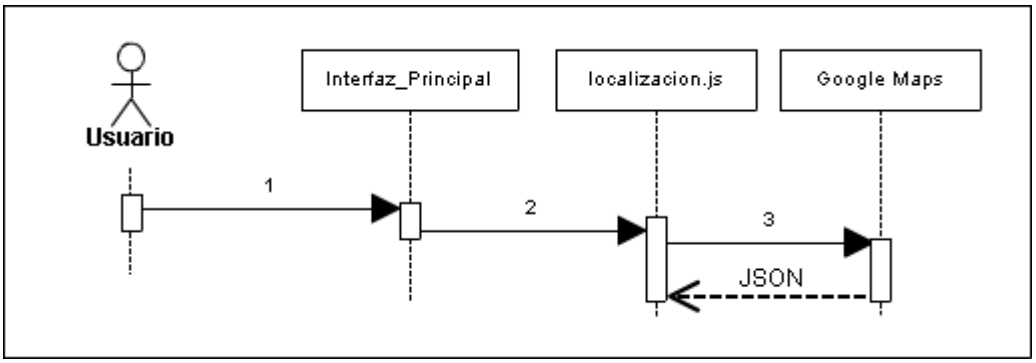


Ilustración 64. Diagrama de secuencia – Mostrar mapa

1. El usuario accede a la interfaz principal de la web o realiza una búsqueda

2. ShowMap()

(Se muestra un mapa a partir de unas coordenadas. Si se acaba de acceder a la web principal, por defecto mostrará el mapa de España. Si se ha hecho una búsqueda, se mostrará el mapa seleccionado en el filtro “País o región”)

3. getGoogleLatAndLong(address)

(Utilizando la API de Google Maps, se obtiene la latitud y longitud de una ubicación)

Buscar tweets

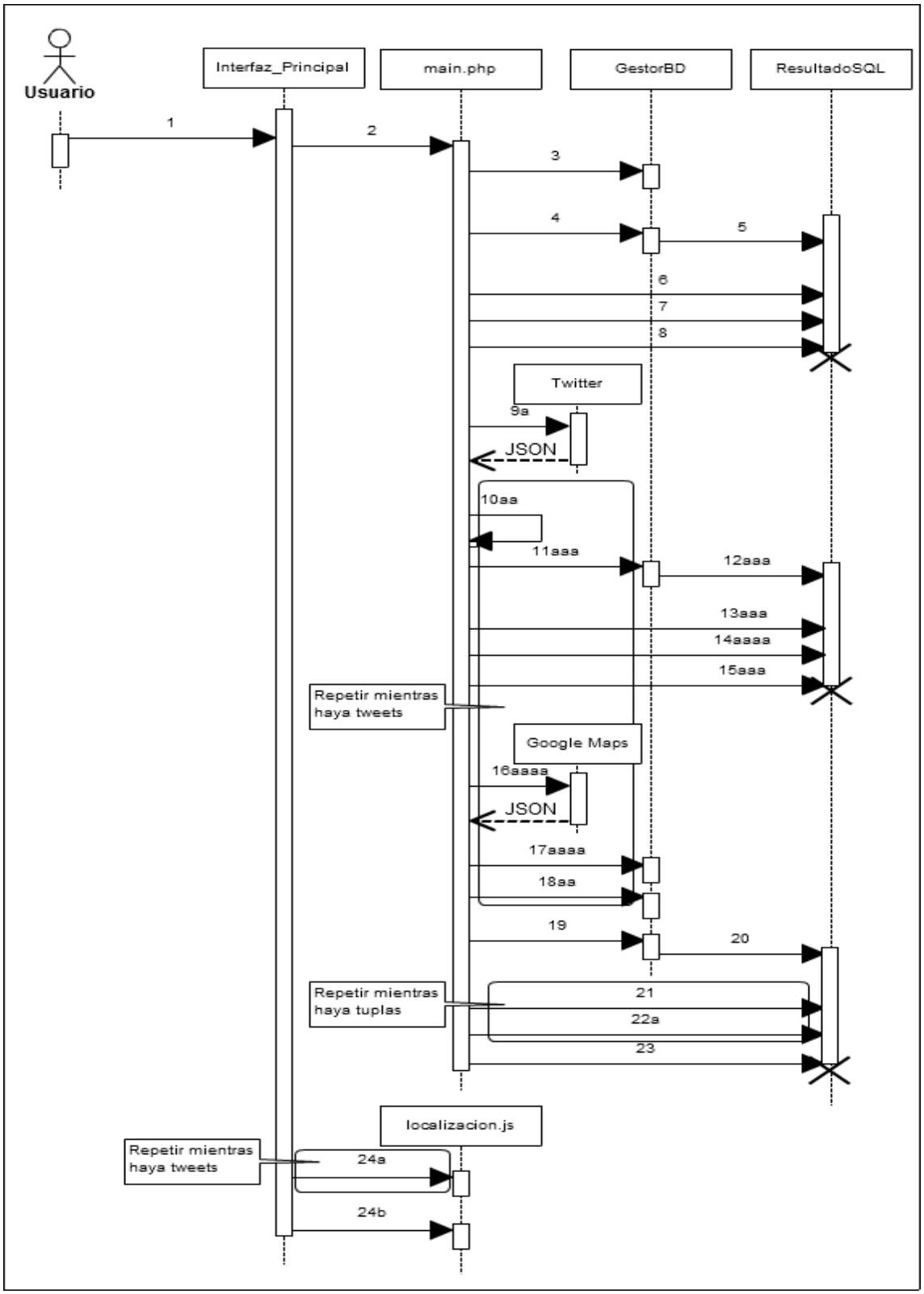


Ilustración 65. Diagrama de secuencia – Buscar tweets

1. El usuario realiza una búsqueda estándar
2. getTweets()
3. execSQL(“INSERT INTO busquedas(busqueda, fecha) VALUES ($buscarBD, CURDATE())”)
   (Se inserta la búsqueda en la base de datos)
4. execSQL(“SELECT COUNT(busqueda) FROM `busquedas` WHERE (busqueda = $buscarBD AND fecha = CURDATE())”)
   (Hace un recuento de las mismas búsquedas que hay ese día)
5. new()
6. next()
7. getNumBusquedas()
8. close()

[Si es la primera búsqueda o múltiplo de 10]

9a.
get('https://api.twitter.com/1.1/search/tweets.json?q='.$buscar.'&result_type=recent

&count=500&lang=es&geocode=39.86283160000001,-4.02732309999999,800km')

(Mediante la API de Twitter obtenemos los tweets más recientes en España que contengan el texto de la búsqueda)

[Si hay tweets]

10aa. getTweet()

[Si el tweet no tiene geolocalización]

11aaa. execSQL(“SELECT latitud,longitud FROM `ubicaciones` WHERE ubicacion=$address”)

(Se cogen las coordenadas de la ubicación)

12aaa. new()

13aaa. next()

[Si hay tuplas]

14aaaa. getLatLong()

15aaa. close()

[Si la ubicación no existe en la base de datos]

16aaaa. getCoordinates($address)

(Usando la ubicación de registro del usuario que ha escrito el tweet y con la ayuda de la API de Google Maps, se obtienen las coordenadas)

17aaaa. execSQL("INSERT INTO ubicaciones(ubicacion,latitud,longitud) VALUES ($address, $lat, $lng)")

(Se inserta la ubicación con sus coordenadas en la base de datos)

18aa. execSQL("REPLACE INTO tweets(id,busqueda,texto,latitud,longitud,fecha,usuario) VALUES ($id,$buscarBD, $texto, $latitud, $longitud, $fecha, $usuario)")

(Se inserta el tweet con todos sus datos en la base de datos, si ya existe se reemplaza)

19. execSQL("SELECT id,texto,latitud,longitud,fecha,usuario FROM `tweets` WHERE (busqueda LIKE $buscarBD OR texto LIKE $buscarBD) AND fecha>=DATE_ADD(NOW(),$tiempo)")

(Se seleccionan los tweets que coinciden con los filtros de búsqueda introducidos por el usuario)

20. new()

21. next()

[Si hay tuplas]

22a. getDatosTweet()

23. close()

[Si hay tweets]

24a. addMarker(map, googleLatAndLong, titulo, contenido, porcion_fecha, hoy)

(Se añade un marcador al mapa con el texto, la fecha y hora de publicación del tweet, con un enlace al tweet original)

[Si no hay tweets]

24b. mostrarAviso()

(Aparecerá un aviso advirtiendo de que no se han encontrado resultados para la búsqueda realizada)

Mostrar búsquedas recomendadas

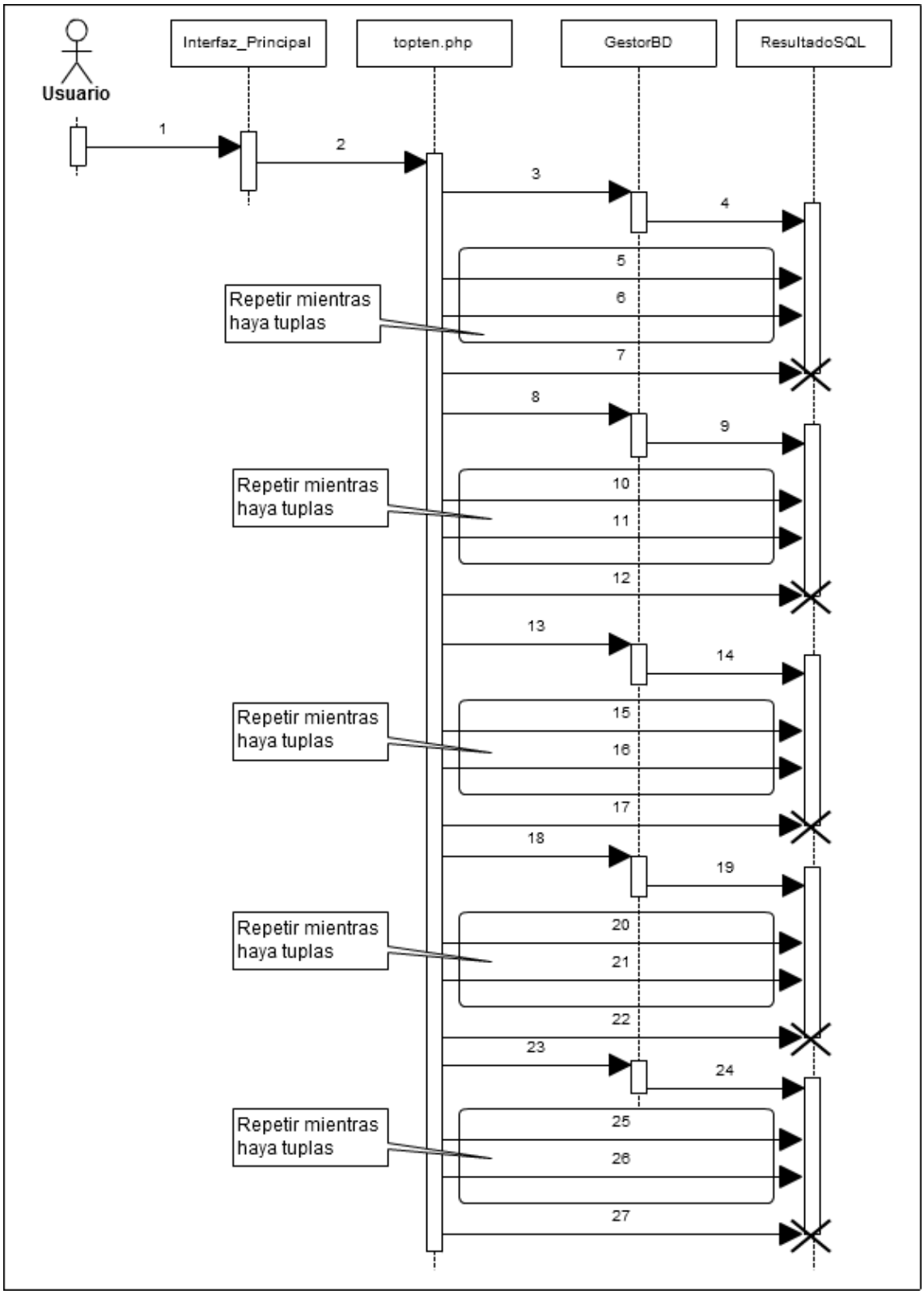


Ilustración 66. Diagrama de secuencia – Mostrar búsquedas recomendadas

1. El usuario accede a la página principal de la aplicación
2. getBusquedasRecomendadas()
3. execSQL(“SELECT busqueda, COUNT(busqueda) FROM busquedas WHERE NOT (busqueda LIKE '@%') GROUP BY busqueda ORDER BY COUNT(busqueda) DESC LIMIT 0, 10”)
   (Se cogen las 10 palabras más buscadas por los usuarios)
4. new()
5. next()
6. getPalabrasMasBuscadas()
7. close()
8. execSQL(“SELECT busqueda, COUNT(id) FROM tweets WHERE NOT (busqueda LIKE '@%') GROUP BY busqueda ORDER BY COUNT(id) DESC LIMIT 0, 10”)
   (Se cogen las 10 palabras que más tweets tienen)
9. new()
10. next()
11. getParalabrasConMasTweets()
12. close()
13. execSQL(“SELECT busqueda, COUNT(busqueda) FROM busquedas WHERE busqueda LIKE '@%' GROUP BY busqueda ORDER BY COUNT(busqueda) DESC LIMIT 0, 10”)
    (Se cogen los 10 usuarios más buscados)
14. new()
15. next()
16. getUsuariosMasBuscados()
17. close()
18. execSQL(“SELECT busqueda, COUNT(busqueda) FROM busquedas WHERE (fecha>=DATE_ADD(NOW(),INTERVAL -1 MONTH)) GROUP BY busqueda ORDER BY COUNT(busqueda) DESC LIMIT 0, 10”)
    (Se cogen las 10 palabras más buscadas en el último mes)
19. new()
20. next()
21. getPalabrasMasBuscadasMes()
22. close()
23. execSQL(“SELECT busqueda, COUNT(busqueda) FROM busquedas WHERE fecha = CURDATE() AND busqueda IN (SELECT queja FROM quejas) GROUP BY busqueda ORDER BY COUNT(busqueda) DESC LIMIT 0 , 10”)
    (Se cogen las 10 palabras de queja más buscadas en ese día)
24. new()
25. next()
26. getQuejas()
27. close()

Actualizar búsquedas diarias

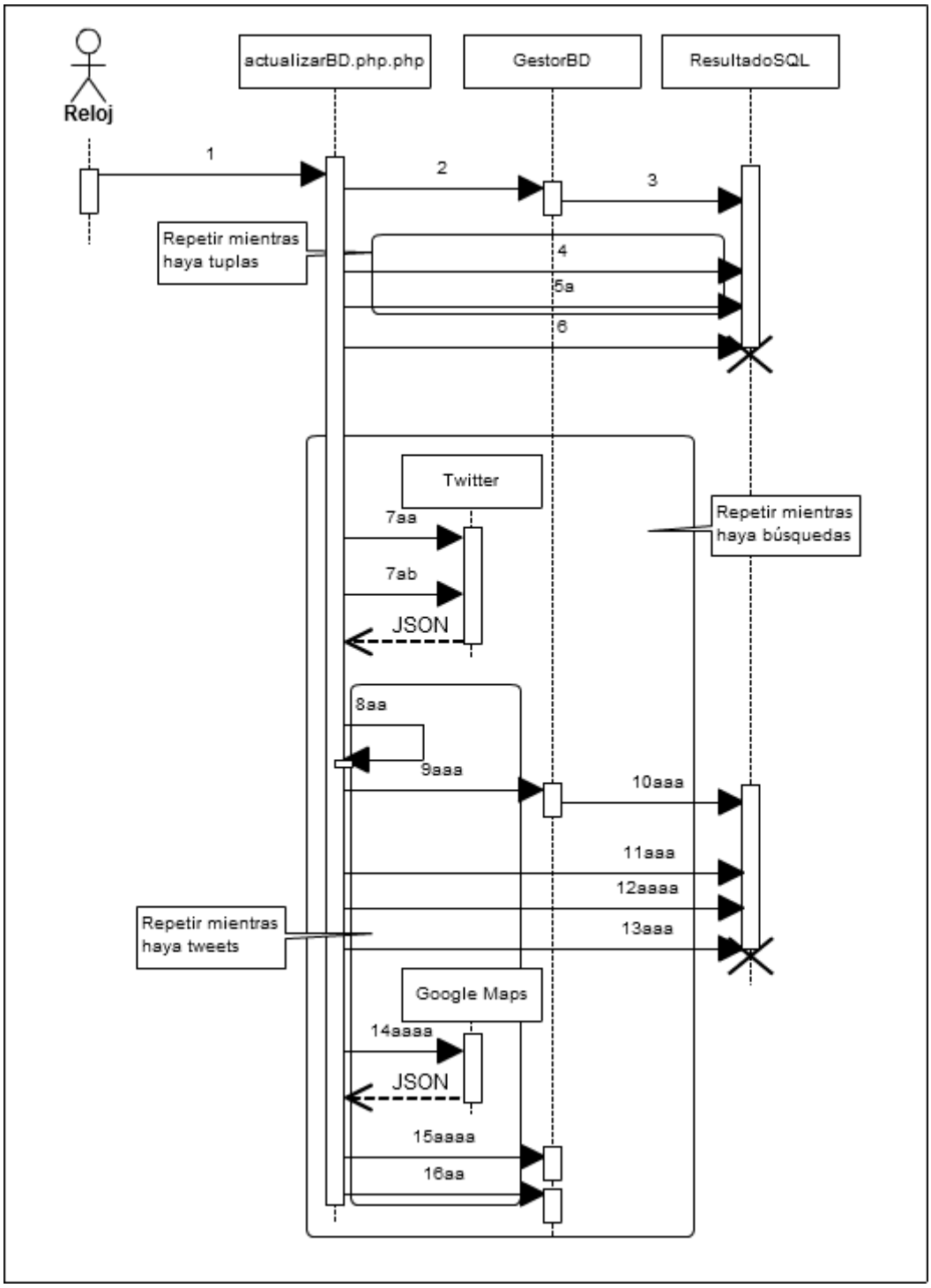


Ilustración 67. Diagrama de secuencia – Actualizar búsquedas diarias

1. actualizarBusquedasDiarias()
2. execSQL(“SELECT busqueda FROM `busquedas` WHERE fecha=CURDATE() GROUP BY busqueda”)
   (Se seleccionan todas las búsquedas que se han hecho ese día)
3. new()
4. next()

[Si hay tuplas]

5a. getBusqeda()

6. close()

[Si hay búsquedas]

[Si es una búsqueda estándar]

7aa.
get('https://api.twitter.com/1.1/search/tweets.json?q='.$buscar.'&result_type=recent

&count=500&lang=es&geocode=39.86283160000001,-4.02732309999999,800km')

(Mediante la API de Twitter obtenemos los tweets más recientes en España que contengan el texto de la búsqueda)

[Si es una búsqueda por usuario]

7ab.
get('https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name='.$busqueda.'&count=100')

(Mediante la API de Twitter obtenemos los tweets más recientes que ha escrito un usuario)

[Si hay tweets]

8aa. getTweet()

[Si el tweet no tiene geolocalización]

9aaa. execSQL(“SELECT latitud,longitud FROM `ubicaciones` WHERE ubicacion=$address”)

(Se cogen las coordenadas de la ubicación)

10aaa. new()

11aaa. next()

[Si hay tuplas]

12aaaa. getLatLong()

13aaa. close()

[Si la ubicación de registro del usuario no existe en la base de datos]

14aaaa. getCoordinates($address)

(Usando la ubicación de registro del usuario que ha escrito el tweet y con la ayuda de la API de Google Maps, se obtienen las coordenadas)

15aaaa. execSQL("INSERT INTO ubicaciones(ubicacion,latitud,longitud) VALUES ($address, $lat, $lng)")

(Se inserta la ubicación con sus coordenadas en la base de datos)

16aa. execSQL("REPLACE INTO tweets(id,busqueda,texto,latitud,longitud,fecha,usuario) VALUES ($id,$buscarBD, $texto, $latitud, $longitud, $fecha, $usuario)")

(Se inserta el tweet con todos sus datos en la base de datos, si ya existe se reemplaza)

Buscar tweets por usuario

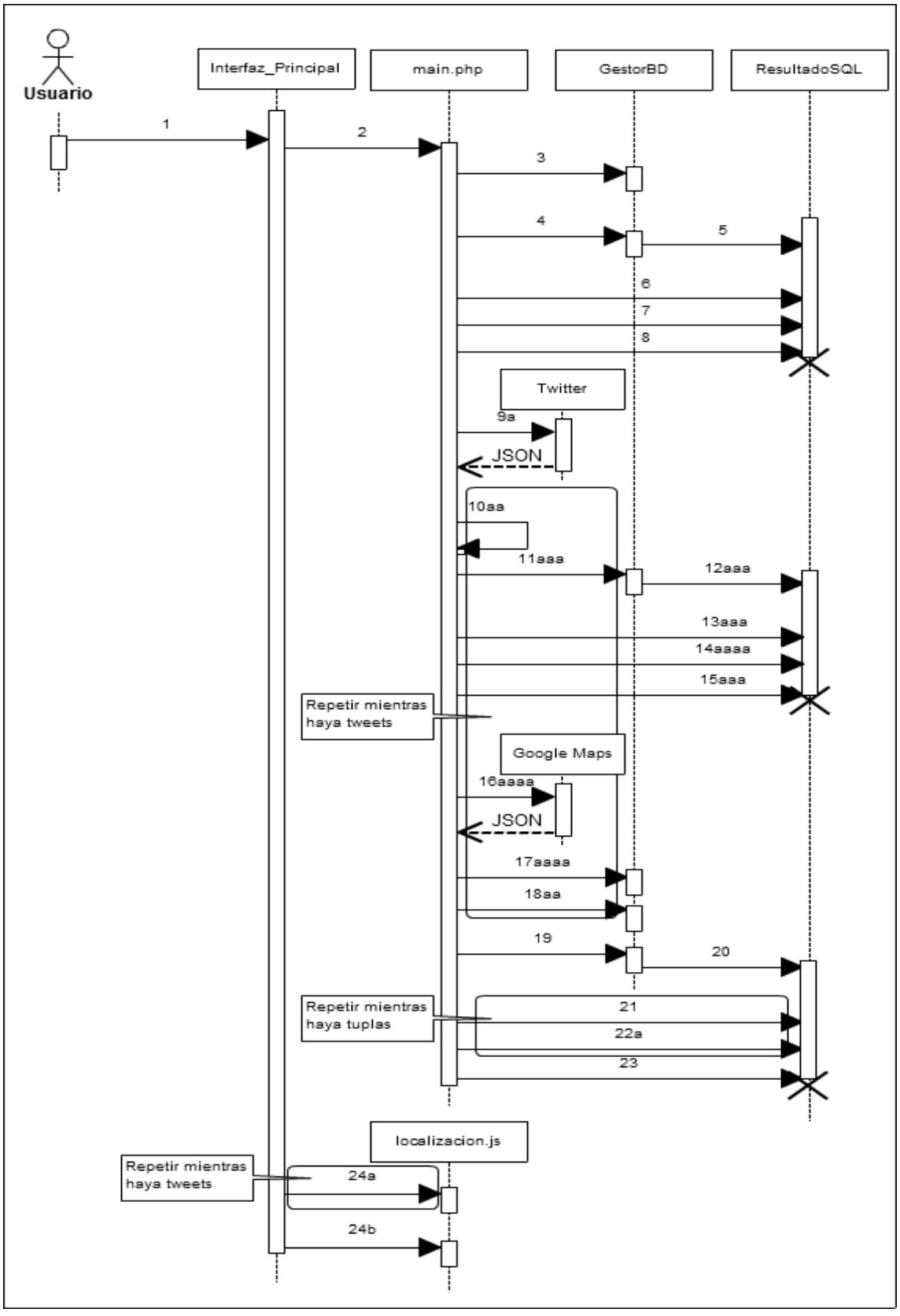


Ilustración 68. Diagrama de secuencia – Buscar tweets por usuario

1. El usuario selecciona “Búqueda por nombre de usuario” y realiza una búsqueda
2. getTweetsUsuario()
3. execSQL(“INSERT INTO busquedas(busqueda, fecha) VALUES ($buscarBD, CURDATE())”)
   (Se inserta la búsqueda en la base de datos)
4. execSQL(“SELECT COUNT(busqueda) FROM `busquedas` WHERE (busqueda = $buscarBD AND fecha = CURDATE())”)
   (Hace un recuento de las mismas búsquedas que hay ese día)
5. new()
6. next()
7. getNumBusquedas()
8. close()

[Si es la primera búsqueda o múltiplo de 10]

9a.
get('https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name='.$buscar.'&count=100')

(Mediante la API de Twitter obtenemos los tweets más recientes que ha escrito un usuario)

[Si hay tweets]

10aa. getTweet()

[Si el tweet no tiene geolocalización]

11aaa. execSQL(“SELECT latitud,longitud FROM `ubicaciones` WHERE ubicacion=$address”)

(Se cogen las coordenadas de la ubicación)

12aaa. new()

13aaa. next()

[Si hay tuplas]

14aaaa. getLatLong()

15aaa. close()

[Si la ubicación no existe en la base de datos]

16aaaa. getCoordinates($address)

(Usando la ubicación de registro del usuario que ha escrito el tweet y con la ayuda de la API de Google Maps, se obtienen las coordenadas)

17aaaa. execSQL("INSERT INTO ubicaciones(ubicacion,latitud,longitud) VALUES ($address, $lat, $lng)")

(Se inserta la ubicación con sus coordenadas en la base de datos)

18aa. execSQL("REPLACE INTO tweets(id,busqueda,texto,latitud,longitud,fecha,usuario) VALUES ($id,$buscarBD, $texto, $latitud, $longitud, $fecha, $usuario)")

(Se inserta el tweet con todos sus datos en la base de datos, si ya existe se reemplaza)

19. execSQL("SELECT id,texto,latitud,longitud,fecha,usuario FROM `tweets` WHERE (busqueda LIKE $buscarBD OR texto LIKE $buscarBD) AND fecha>=DATE_ADD(NOW(),$tiempo)")

(Se seleccionan los tweets que coinciden con los filtros de búsqueda introducidos por el usuario)

20. new()

21. next()

[Si hay tuplas]

22a. getDatosTweet()

23. close()

[Si hay tweets]

24a. addMarker(map, googleLatAndLong, titulo, contenido, porcion_fecha, hoy)

(Se añade un marcador al mapa con el texto, la fecha y hora de publicación del tweet, con un enlace al tweet original)

[Si no hay tweets]

24b. mostrarAviso()

(Aparecerá un aviso advirtiendo de que no se han encontrado resultados para la búsqueda realizada)