CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Matemáticas e Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

EXTRACCIÓN AUTOMÁTICA Y PUBLICACIÓN DE CONOCIMIENTO MÉDICO A TRAVÉS DE CUESTIONARIOS EN LA RED SOCIAL TWITTER

Autor: Sandra Garrido Romero

Director: Alejandro Rodríguez González

MADRID, JUNIO 2016

RESUMEN

En este trabajo de fin de grado se ha desarrollado una herramienta en lenguaje de programación Java con el fin de estudiar la sabiduría de masas en la red social Twitter.

El desarrollo del estudio ha necesitado la recopilación de mensajes en la red social de personas relacionadas con el ámbito médico. Para ello, se ha creado una cuenta en la red social Twitter que publica un tweet diario en forma de pregunta y cuatro posibles respuestas de las que cada usuario deberá elegir una. Todas estas respuestas se recopilan en una base de datos para su posterior estudio.

Además, la aplicación desarrollada se conecta con una interfaz web en la que los usuarios podrán registrarse indicando su profesión o ámbito de estudio. De esta manera, en el estudio de los datos se podrán comparar las respuestas dadas separándolas por grado de especialidad de los usuarios.

ABSTRACT

This final dissertation is focused on the development of an analytic tool in the programming language Java with the purpose of study the wisdom of crowds in the social network Twitter.

The study has needed the compilation of messages in the social network of people related with the medicine field. As a consequence, a profile in the social network Twitter has been created. This profile publishes a daily tweet with a question and four possible answers which each user should choose one of these. All of these answers have been gathered in a database for a later study.

Finally, the application develop connects with a web interface where the users can be registered inserting their profession or study area. Therefore, in the study of data the answers can be compared depends on the users' grades of specialty.

Índice

Índice de ilustraciones

1. INTRODUCCIÓN

Este trabajo inicialmente estaba centrado en la extracción de información de textos descriptivos en páginas web especializadas en medicina. De manera que se pretendía obtener datos de enfermedades a través de sus síntomas. La realización de este análisis requería conocimiento de herramientas y tecnologías complejas necesarias para el desarrollo del proyecto. El director consideró adecuado un cambio de orientación en el trabajo.

La nueva temática planteada sigue los mismos objetivos iniciales que la propuesta original relativos a la extracción de conocimiento médico. La diferencia radica en la fuente de obtención de datos. En el nuevo planteamiento del trabajo, los datos se consiguen a través de la publicación y extracción automática de cuestionarios médicos en una red social, en concreto, Twitter.

Se pretende obtener datos de las respuestas que los usuarios den a los cuestionarios publicados. De manera que se pueda estudiar la sabiduría de masas. Esta teoría defiende que en el proceso de resolución de un problema, es preferible partir de la opinión de un grupo de individuos independientes con cierto conocimiento en la materia en cuestión que de la opinión de un experto en la materia.

Para realizar el estudio se publicarán preguntas de ámbito médico diariamente con cuatro posibles respuestas. Los usuarios deberán publicar un tweet con el *hashtag* que corresponda a la respuesta correcta, estas respuestas se almacenarán en una base de datos para su posterior estudio. En la realización del estudio se quiere comprobar que porcentaje alto de usuarios llegan a la respuesta correcta a la pregunta planteada así como otros posibles análisis posteriores que puedan indicar información adicional como por ejemplo las relaciones existentes entre los usuarios que contestan a las preguntas, etc.

Además los usuarios cuentan con una interfaz web desde la que podrán gestionar sus propias respuestas o responder a las preguntas. Esta interfaz web ha sido desarrollada por mi compañero Juan Aguado en su propio TFG.

El almacenaje de todas las respuestas, aparte de servir para el estudio mencionado, también puede ser utilizado para realizar estadísticas en función de la rama de medicina en la que esté especializado cada usuario o, según otros factores como la edad. Este tipo de estudio será posible gracias a los datos que los usuarios tendrán que dar en el registro en la interfaz web.

1.1. ESTRUCTURA DE LA MEMORIA

Este trabajo está dividido en cinco capítulos:

- **Capítulo 1**: Se expone una introducción al trabajo y la estructura elegida para el trabajo.

- **Capítulo 2**: Se especifican los objetivos impuestos al trabajo.

- **Capítulo 3**: Se resume la situación actual en la que se encuentran los temas a tratar.

- **Capítulo 4**: Se especifican los requisitos necesarios para el desarrollo de la aplicación. Además se desarrollan las decisiones tomadas en cuanto a la implementación para conseguir los objetivos marcados. También se detalla el desarrollo de los distintos módulos de los que se compone la aplicación.

- **Capítulo 5**: Se presentan los problemas encontrados en el desarrollo realizado junto a los resultados obtenidos. Finalmente, se exponen las conclusiones a las que se ha llegado.

- **Capítulo 6**: Se adjunta un manual de usuario de la aplicación para que futuros usuarios puedan usarla sin problemas.

- **Capítulo 7**: Se detalla la bibliografía utilizada como apoyo para el trabajo realizado.

2. OBJETIVOS

El objetivo general de este trabajo es extraer conocimiento de Twitter para poder realizar un estudio posterior en el que se consiga visualizar la sabiduría de masas. Para ello, se desarrollará una herramienta en lenguaje *Java* que automatice el proceso de publicar *tweets* y recopilar respuestas.

El estudio es realizado lanzando una serie de preguntas desde una cuenta de Twitter (@medquizzes) creada para este objetivo. Cada pregunta y sus respuestas tienen un hashtag asociado que se utiliza en la búsqueda de las respuestas dadas por los usuarios.

Otro de los objetivos es conectar la aplicación con una interfaz web para que trabajen en conjunto. La comunicación se ha conseguido con la creación de un *socket* por el que se realiza el intercambio de mensajes.

3. ESTADO DEL ARTE

En este apartado se va a proceder a mostrar lo encontrado en la literatura existente respecto al tema de inteligencia colaborativa y sus implicaciones.

Además, se expondrá el uso actual que se hace de las redes sociales a nivel de usuario y la manera en que esos datos pueden ser utilizados para múltiples estudios.

3.1. SABIDURÍA COLECTIVA

Este concepto nació de la mano del autor James Surowiecki, columnista de la revista *New Yorker*, en su libro "*The Wisdom of Crowds*" cuyo título en español es "*Cien mejor que uno*" en 2004. Aunque en 1906 Francis Galton, importante estadístico, hizo el descubrimiento de la sabiduría de masas. La anécdota del descubrimiento es conocida con el "buey de Galton" y se desarrolla en una feria de ganado de Plymouth. En la feria, existía una competición donde había que adivinar el peso de un buey tras ser descuartizada y preparada. Los participantes, que casi llegaron a 800 personas, debían escribir su predicción en un ticket y el ganador se llevaría un premio. Galton recopiló todas las predicciones y realizó un análisis estadístico descubriendo que la estimación media se encontraba cerca del peso real del buey que era 543 kilogramos, sólo se desviaba por una libra. Lo más sorprendente era que esta estimación era mejor que la del ganador individual de la competición y que las cifras que varios expertos habían proporcionado.

La sabiduría colectiva o sabiduría de masas puede definirse como la resolución de un problema a partir de un grupo de personas con cierto conocimiento de la materia. El conocimiento que posee este grupo de personas puede ser muy variado, desde tener un conocimiento general hasta ser expertos en el tema.

Las cuatro condiciones que debe cumplir un grupo de personas para poder hablar de sabiduría colectiva son:

- **Diversidad de opinión.** Cada persona cuenta con su propia información privada aunque sea una mera interpretación de los hechos conocidos.

- **Independencia.** Los individuos que forman el grupo no tienen comunicación entre sí a la hora de llegar a la solución del problema planteado. De esta forma, la opinión de cada persona no es determinada por las opiniones de las personas que la rodean.

- **Descentralización.** Las personas pueden especializarse y recurrir al conocimiento local existente.

- **Agregación.** Mecanismo que transforma los juicios individuales en una decisión colectiva.

La sabiduría de las masas es un análisis que podría sustituir a los análisis de mercado actuales ya que mejora las predicciones. Simplemente, haciendo que un grupo de gente, que cumpla las cuatro propiedades descritas anteriormente, conteste a una serie de preguntas.

Desde hace décadas en EEUU existen mercados de predicciones como la empresa *The Iowa Electronic Markets* que está especializada en cotización de eventos políticos. Esta empresa se hizo bastante famosa tras publicar la predicción de las elecciones de Obama contra Romney. Este último intentó manipular el resultado de los mercados de predicciones para poder influir en la acción mediante la predicción.

Otra gran empresa que se dedica a los mercados de predicciones es *Hollywood Stock Exchange* que se encarga de cotizar la taquilla de estrenos de cine y música. Trabaja con dinero virtual y actúa como barómetro determinante de las decisiones de inversión en el mundo del cine.

En territorio nacional, nació hace relativamente poco tiempo la página web http://www.futuramarkets.com/ que aspira a introducir juicio humano en las predicciones. Para ello, trabaja con dinero virtual y los participantes invierten acciones en realizar su propia predicción. Si aciertan con la predicción, reciben más acciones. Si fallan, no reciben nada.

Aunque el análisis de predicciones aún no se encuentra demasiado extendido parece que poco a poco comienza a emerger en distintos puntos del planeta.

3.2. USO DE LAS REDES SOCIALES

Las redes sociales están muy de moda desde hace unos años aunque sus inicios se remontan mucho tiempo atrás.

El nacimiento de las redes sociales puede situarse en el año 1971 cuando se envió el primer correo electrónico. Aunque la primera red social con un concepto parecido al que tenemos hoy en día apareció en 1994 cuando se fundó *GeoCities*. En ella, los usuarios creaban sus páginas web y las alojaban en barrios (Silicon Valley, Hollywood, Wallstreet…) dependiendo de su contenido.

Un año después, en 1995, se funda la página *theglobe.com* donde los usuarios podían personalizar sus experiencias online publicando su propio contenido e interactuando con personas con sus mismos intereses.

En 1997, se crea *AOL Instant Messenger* programa de mensajería instantánea que se encuentra actualmente en uso. En ese mismo año, aparece la página web *sixdegrees.com* en la que se podían crear perfiles personales.

En 2002 se lanza el portal *Friendster* que permitía la conexión online de amigos con la que se podía compartir contenido online.

En 2003 nace *MySpace* concebida como clon del portal mencionado en el párrafo anterior.

Un año más tarde aparece *Facebook* de la mano de Mark Zuckerberg y en 2006 se inaugura *Twitter*, la red social que se usa en el desarrollo de este trabajo.

Todas las redes sociales tienen varias aplicaciones aparte de su objetivo principal de crear una comunidad online. Algunas de estas aplicaciones son:

- **Aplicaciones de negocio**. Las empresas utilizan las redes sociales para darse publicidad lo que incrementa las posibilidades de darse a conocer a más personas en el mundo. También, la empresa y los clientes pueden contactar entre ellos fácilmente a través de perfiles sociales creados para ello.

- **Aplicaciones médicas**. Actualmente, existe una red social llamada *Sermo* creada exclusivamente para profesionales médicos. En ella, pueden compartir sus conocimientos sobre enfermedades, experiencias o pedir opinión sobre casos complicados.

- **Investigaciones**. En múltiples investigaciones policiales se usa el contenido subido a redes sociales de los implicados en el caso como prueba.

Tras ver las posibles aplicaciones que pueden tener las redes sociales, llegamos a la sabiduría colectiva que es la base de este trabajo. Por ejemplo, las empresas usan las redes sociales como sabiduría colectiva pidiendo la opinión de sus productos a los clientes. O, como se ha visto en las aplicaciones médicas, los propios profesionales piden consejo a un grupo de gente.

También la información subida a las redes sociales es usada en estudios de *Data analytics* en los que se trabaja con un gran volumen de datos para sacar información.

Un ejemplo del uso de las redes sociales se encuentra en el artículo "*Automatic extraction and identification os users' responses in Facebook medical quizzes*"[1]. En este estudio, se analizan las respuestas que los usuarios realizan a una serie de *quizzes* publicados por una cuenta relacionada con el ámbito médico en *Facebook*. De esta manera, se pretende estudiar la inteligencia colectiva, lo cual es el objetivo del presente trabajo pero en la red social *Twitter*.

Para terminar, enfatizar el hecho de que toda la información presente en las redes sociales es usada para múltiples fines y estudios.

[1] http://www.ncbi.nlm.nih.gov/pubmed/26777433

4. DESARROLLO

Este capítulo se centra en las decisiones tomadas para conseguir desarrollar la aplicación.

Primero, se detallan los requisitos que se han tenido que cumplir en la implementación de la aplicación. Estos requisitos, a su vez, se dividen en varios subapartados dependiendo del tipo.

Más adelante, se especifica la infraestructura que se ha seguido en el proyecto.

Y para terminar, se explican todas las decisiones pertinentes a la implementación como el lenguaje de programación elegido y los distintos módulos en los que se divide la aplicación.

4.1. ESPECIFICACIÓN DE REQUISITOS

En este apartado se proceden a detallar los distintos requisitos que han sido necesarios para el desarrollo del trabajo. Han sido divididos en requisitos funcionales, requisitos de desarrollo, requisitos tecnológicos y atributos.

4.1.1. REQUISITOS FUNCIONALES

Publicar tweet diario

REQ(01) publishDailyTweet: Prioridad: Alta. Permitir publicar tweet compuesto de una imagen con una pregunta y cuatro posibles respuestas. La publicación se realizará diariamente a la misma hora.

Publicar tweet

REQ(02) publishTweet: Prioridad: Alta. Permitir al administrador de la plataforma web publicar en Twitter una pregunta automáticamente.

Recoger respuestas

REQ(03) recoverTweets: Prioridad: Alta. Permitir recopilar las respuestas que los usuarios hayan publicado en Twitter. El proceso se realizará diariamente a la misma hora.

4.1.2. REQUISITOS DE DESARROLLO

Requisito(x): Prioridad: Alta. El ciclo de vida elegido para desarrollar la aplicación es de prototipo evolutivo, de manera que se puedan introducir de forma sencilla cambios y nuevas funciones.

4.1.3. REQUISITOS TECNOLÓGICOS

Requisito(x+1): Prioridad: Alta. La infraestructura para la aplicación es la siguiente:

- Sistema de Gestión de base de datos MySQL.
- Java

4.1.4. ATRIBUTOS

Extensibilidad

Requisito (x+2): Prioridad Alta. Se tendrá en cuenta la posibilidad de extender la base de datos. Toda decisión tecnológica sobre el sistema deberá tener este hecho en cuenta, para anticipar posibles problemas.

Disponibilidad

Requisito (x+3): Prioridad Alta. El sistema estará accesible las 24 horas del día los 7 días de la semana 24/7.

Seguridad

Requisito (x+4): Prioridad Alta. El sistema deberá cumplir la Ley Orgánica de Protección de Datos.

4.2. INFRAESTRUCTURA DEL PROYECTO

En este apartado se describirá la infraestructura que se ha diseñado para llevar a cabo el proyecto explicando bases de datos, frameworks y demás tecnologías utilizadas durante el desarrollo del proyecto.

Como ya se ha comentado anteriormente, este proyecto está dividido en dos partes bien diferenciadas. Por un lado, la aplicación desarrollada en *Java* expuesta en este trabajo y por otro, una aplicación web desarrollada por mi compañero Juan Aguado en la que no entraremos en detalles exhaustivos.

El proyecto desarrollado ha requerido la creación de una base de datos en la que se pudiesen guardar las preguntas y respuestas de los *quizzes*. Se ha decidido utilizar *MySQL* como motor de la base de datos. *MySQL* es un sistema de gestión de bases de datos relacional desarrollado por *Oracle Corporation*. Es uno de los sistemas más popular para gestión de bases de datos ya que es usado por sitios webs bastante famosos como *Wikipedia*, *Google*, *Facebook*, *Twitter*, *Flickr* y *Youtube*.

Hoy en día, varias interfaces de programación de aplicaciones permiten acceder a las bases de datos de *MySQL* a aplicaciones escritas en diversos lenguajes de programación. Entre estos lenguajes destacar *Java* y *PHP* ya que el primero es el lenguaje usado en la aplicación en la que se basa este trabajo, y el segundo, es uno de los lenguajes utilizados por mi compañero Juan Aguado para la creación de la aplicación web.

El empleo de la base de datos ha implicado la creación de una serie de tablas para almacenar las encuestas que se deseen publicar y guardar las respuestas que los usuarios de *Twitter* realicen, para su posterior análisis. Es una base de datos bastante sencilla ya que sólo cuenta con tres tablas:

- **Quiz**. Guarda las preguntas que se van publicando.

- **Quiz_hashtag**. Contiene las respuestas a las preguntas guardadas en la tabla anterior.

- **Tweet**. Incluye las respuestas recopiladas de *Twitter*.

En la siguiente imagen se muestra el diagrama entidad relación de la base de datos compartida entre los dos proyectos:

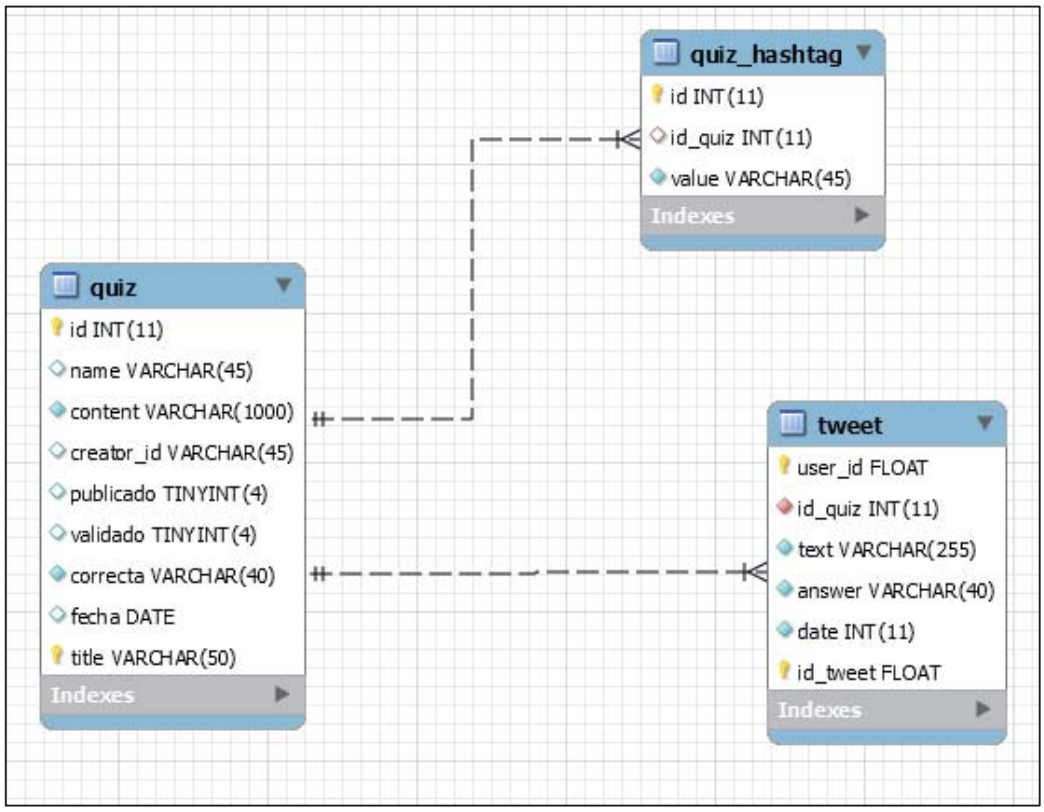

Ilustración 1: Diagrama entidad relación

A continuación se realiza una descripción de las tablas que aparecen en el diagrama entidad relación mostrado.

4.2.1. TABLA QUIZ

Esta tabla almacena las distintas preguntas de las encuestas que se van a publicar. El servicio que se ejecuta cada veinticuatro horas y publica un *tweet* consulta en esta tabla cuál es la primera pregunta que está pendiente de publicar comprobando una de las columnas de la tabla cómo se verá más adelante. Tras la consulta, publicará en *Twitter* la imagen correspondiente.

Esta tabla también es consultada cuando se quiere publicar un *tweet* bajo demanda para buscar la pregunta correspondiente al identificador pasado como parámetro.

A continuación se procede a realizar una explicación de cada columna que compone esta tabla:

- **Id**. Identificador de la pregunta. Es una columna autoincremental y forma parte de la clave primaria de la tabla.

- **Name**. Se utiliza como identificador de la encuesta en *Twitter*. Se almacena un componente de tipo *String* compuesto por la letra "Q" más el identificador de la encuesta. Además, este nombre se utiliza también para generar la imagen correspondiente y poder acceder a ella en el futuro. El nombre del archivo queda de la siguiente forma "Q"+identificador+".jpg".
  Como ejemplo, para la pregunta con identificador cinco, la columna *name* tendrá el valor Q5, y la imagen que genere la aplicación se llamará "Q5.jpg".

- **Content**. Esta columna almacena el texto que se desea publicar. Este texto es el que se utiliza como pregunta para generar la imagen de la encuesta. En función de su extensión, la imagen tendrá un tamaño u otro. Esto es debido a que el programa calcula el ancho de la imagen en función del número de caracteres con los que cuenta la pregunta y el alto dependiendo del número de líneas de las que conste la pregunta junto con las respuestas.

- **Creator_id**. Guarda el identificador del usuario que propuso la encuesta. Por defecto, el administrador de la aplicación web es el encargado de generar las preguntas, pero existe la opción de que cada usuario proponga nuevas encuestas.

- **Publicado**. Indica la situación de cada pregunta. La columna tendrá como valor por defecto el número cero. Una vez que sea publicada en *Twitter* el valor cambiará a uno.

- **Validado**. Esta columna sólo tiene sentido para las preguntas propuestas por los usuarios de la aplicación. Cuando una pregunta es propuesta, el valor de esta columna será cero hasta que el administrador la revise y valide, en este momento, pasa a tener valor uno y podrá ser publicada. Las preguntas cargadas directamente desde un archivo csv tendrán por defecto esta columna a uno.

- **Correcta**. Esta columna almacena la letra correspondiente a la respuesta correcta de la pregunta. Los valores que puede tomar son A, B, C, o D.

- **Fecha**. Indica la fecha en la que se añaden las encuestas al sistema.

- **Title**. Representa el título de la encuesta que aparecerá en la aplicación web. Forma parte de la clave primaria de la tabla.

4.2.2. TABLA QUIZ_HASHTAG

Esta tabla es utilizada para almacenar las posibles respuestas que puede tener cada pregunta. Aunque pueda parecer redundante su uso, es necesario para la aplicación *Java*. En un principio, se pensó que esta tabla era innecesaria ya que en la tabla anterior se guarda la pregunta con la letra que corresponde a la respuesta correcta, pero al intentar poner en marcha la aplicación y publicar encuestas el texto de las respuestas no se encontraba almacenado en ningún sitio lo que imposibilitaba la publicación. A partir de este punto, se llegó a la conclusión que se debía crear una tabla para almacenar el texto de las respuestas. A continuación, se detalla el contenido de cada columna de la tabla:

- **Id**. Identificador de cada pregunta, está establecido como autoincremental. Corresponde a la clave primaria de esta tabla.

- **Id_quiz**. Esta columna es una clave foránea que corresponde a la columna id de la tabla *quiz*. Se utiliza para relacionar el *hashtag* de la respuesta dada por el usuario con la pregunta correspondiente.

- **Value**. Almacena el texto de las respuestas que aparecerá en la imagen que se publique en *Twitter*.

4.2.3. TABLA TWEET

Esta tabla se utiliza para almacenar los *tweets* que publican los usuarios de la red social *Twitter* relacionados con las encuestas almacenadas en la base de datos. La aplicación *Java* es la encargada de alimentar esta tabla, conectándose a *Twitter* y buscando los *tweets* que contienen los *hashtags* de las preguntas publicadas. Esta tabla cuenta con las siguientes columnas:

- **User_id**. Esta columna almacena el identificador del usuario que publicó el *tweet* recogido. Este identificador es el proporcionado por *Twitter* para cada usuario. Este valor forma parte de la clave primaria de la tabla.

- **Id_quiz**. Esta columna corresponde a una clave foránea de la columna identificador de la tabla *quiz*, su función es relacionar cada *tweet* recogido como respuesta con la encuesta a la que responde. Esta relación también se puede conocer a través de la tabla *quiz_hashtag*. La aplicación *Java* detecta automáticamente a qué pregunta está respondiendo cada *tweet* para poder almacenar en esta columna su identificador.

- **Text**. Texto del que está compuesto el *tweet* recogido.

- **Answer**. Esta columna almacena el valor de la respuesta dada, es decir, los valores A, B, C o D. El valor de la respuesta se detecta a través del *hashtag* con el que se busca el *tweet*. Por ejemplo, el *hashtag* #Q10B representa una respuesta al *quiz* con el valor del identificador igual a diez y la respuesta con valor B.

- **Date**. Esta columna contiene la fecha en la que se publicó el *tweet*.

- **Id_tweet**. Esta columna contiene el identificador que *Twitter* aporta a cada uno de los *tweets* que se publican en él.

La gran mayoría de columnas de esta tabla (user_id, text, date, id_tweet) representan valores que se obtienen directamente desde *Twitter*, a través de la aplicación *Java* que se conecta con la librería oficial de la red social mediante una serie de funciones *get()*. Esta librería permite obtener mucha más información acerca de cada *tweet*, y por tanto, cabe la posibilidad de ampliar esta tabla en un futuro para almacenar más información, y así realizar un análisis más exhaustivo.

4.3. ARQUITECTURA

Este punto muestra la arquitectura de la que se compone la aplicación en funcionamiento incluyendo el trabajo desarrollado en *Java*, *Twitter* y la interfaz web de mi compañero. En la imagen que se muestra a continuación se puede ver un diagrama que representa esta arquitectura.

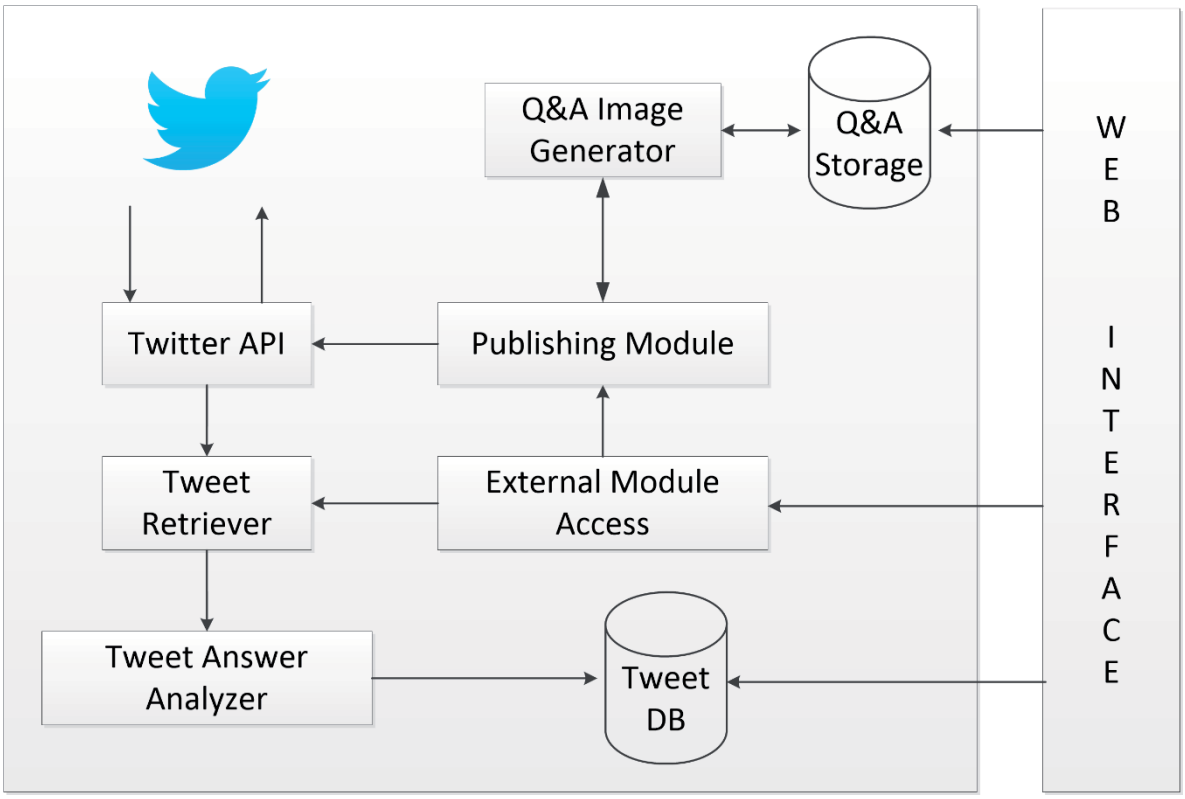

Ilustración 2: Diagrama arquitectura

La arquitectura se compone de varios módulos como se puede ver en la imagen de arriba. A continuación, se procede a explicar la función de cada uno de ellos:

- **Web interface**. Trabajo realizado por mi compañero Juan Aguado consiste en una interfaz web que se conecta con la aplicación que he desarrollado yo.

- **Twitter**. Red social *Twitter*, la cual es usada como escenario del proyecto para la publicación de preguntas.

- **Twitter API**. Librería desarrollada en *Java* usada para establecer la conexión con *Twitter*

- **Tweet Retriever**. Código de la aplicación que se encarga de recoger las respuestas a los usuarios todos los días a las once de la noche.

- **Tweet Answer Analyzer**. Proceso que guarda las respuestas que los usuarios han publicado en *Twitter* en la base de datos de proyecto. Concretamente, se guardan en la tabla *tweet*, tal y como se ha explicado en el apartado anterior. La interfaz web accede a estos datos para poder mostrar estadísticas personalizadas de las respuestas dadas a cada usuario registrado.

- **External Module Access**. Módulo externo de la aplicación que se conecta vía *Telnet*. Tiene dos funciones, por un lado, obtiene los *tweets* que corresponden a respuestas de usuarios conectándose al *Tweet Retriever*. O, publica un *tweet* concreto de la base de datos conectándose a *Publishing Module*.

- **Publishing module**. Se encarga de publicar *tweets*. Para conseguirlo, se conecta con el módulo *Q&A Image generator* y este, a la base de datos para obtener las preguntas y respuestas solicitadas. Una vez que cuenta con todos los datos necesarios se conecta a *Twitter* y publica el *tweet* correspondiente.

- **Q&A Image generator**. Se encarga de generar la imagen que contiene la pregunta con las cuatro posibles respuestas que va a ser publicada. Necesita conectarse a la base de datos para conseguir la información.

- **Q&A storage**. Base de datos que contiene las preguntas y las respuestas que se van a publicar. Además en ella se guardan las respuestas que los usuarios publican a cada pregunta formulada. La interfaz web también accede a esta base de datos.

Algunos de los módulos tienen una relación bidireccional ya se intercambian información entre ellos. Estas relaciones pueden vislumbrarse en la imagen que se ha mostrado al principio del capítulo.

4.4. IMPLEMENTACIÓN

En este apartado de la memoria, se va a proceder a explicar los recursos de programación necesarios para la implementación y a exponer cada uno de los módulos que integran el proyecto en el que se ha trabajado. El proyecto se ha dividido en un total de tres módulos que corresponden a los tres principales objetivos que persigue el proyecto desarrollado.

En primer lugar, se explica el módulo de publicación de *tweets* cuya misión es, como su nombre indica, conectarse a *Twitter* y publicar un *tweet* determinado.

Después se desarrollará la función del módulo de búsqueda de respuestas. En este caso, la aplicación se conecta a *Twitter* para recuperar las respuestas que los usuarios dan a las preguntas publicadas.

Por último, se encuentra el módulo de conexión que realiza la acción de crear un servidor para comunicarse con la interfaz web.

En la siguiente imagen se muestra el diagrama UML del programa llevado a cabo. Como se puede ver está compuesto por cinco clases, de las que cuatro de ellas se realizan llamadas entre sí.

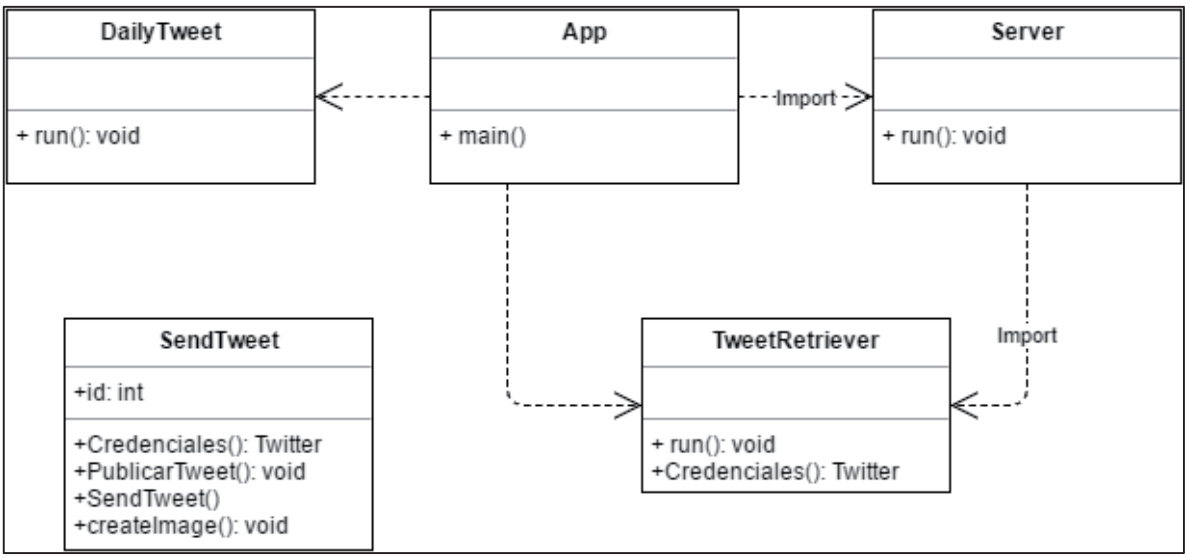

Ilustración 3: Diagrama UML

4.4.1. CONTROL DE VERSIONES

Este proyecto se encuentra alojado en *GitHub*, en un repositorio abierto a la comunidad para que pueda ser accesible. El repositorio ha sido creado por mi compañero Juan Aguado Peña con nombre de usuario de *GitHub* juanekele[2]. El código de la aplicación web y la aplicación *Java* puede consultarse en la siguiente dirección:

[2] https://github.com/juanekele/medQuizs

README.md

medQuizs

Trabajo Fin de Grado 2015-2016 Universidad Politécnica de Madrid

Juan Aguado Peña (Frontend) Drupal + WebApp

Sandra Garrido Romero (Java App -> conection with Twitter and Image Generation)

Directories:

sandra/

Source code binaries, and conf files of the Java application developed By Sandra. PHP files that connects with Java application

sandra/images/

The Java application saves the autogenerated images in this directory

sites/all/themes/medquizs/

Theme for the web. Info file set resources (css,js). /assets/css/ and /assets/js/ are the directories for this resources. /templates/ directory has the templates for each page

sites/all/modules/custom

All the necesary modules are installed in this directory (ctools,doneQuizs,edir_profile,hybridauth,oauth,pendingQuizs,quizs,sidebar,super_login)

Autogenerated Files and directories

Other directories and files are autogenerated by the drupal installation (Drupal 7.43)

Ilustración 4: Archivo Readme de GitHub

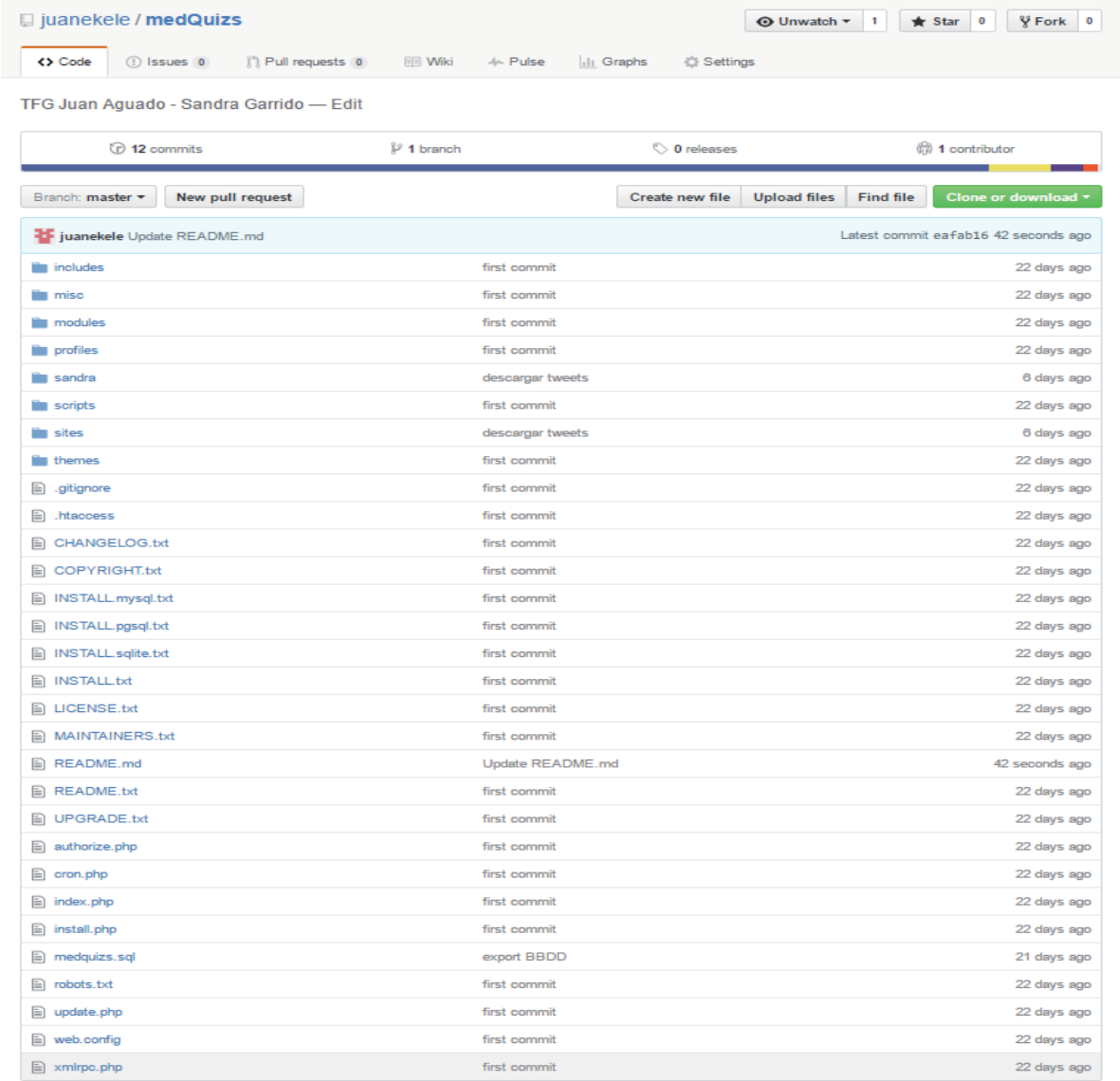

Ilustración 5: Repositorio de GitHub

El uso de la herramienta GitHub lleva implantado desde el día dos de mayo, tras una reunión con el tutor en la que se pensó que podía ser útil para complementar los dos proyectos y trabajar de manera más cómoda en equipo, y además ayudaría de cara a un despliegue futuro a un entorno real por parte del tutor.

Esta herramienta ha sido muy útil para poder trabajar en el proyecto sin necesidad de tener el equipo que se venía usando normalmente, simplemente realizando un *fork* del proyecto en *GitHub*, editando los cambios y añadiendo los cambios al repositorio.

4.4.2. LENGUAJE DE PROGRAMACIÓN

El lenguaje de programación elegido ha sido *Java* por varias razones. Una de ellas es que es el lenguaje que mejor controlo porque ha sido con el que más he practicado a lo largo de mi carrera universitaria. Y, la razón principal es que existen librerías que facilitan la integración de *Java* con *Twitter* y con *MySQL*.

Las principales ventajas de *Java* frente a otros lenguajes de programación han sido las siguientes:

- **Multihilos**. *Java* permite la existencia de varios hilos de ejecución simultáneamente. En la aplicación desarrollada se necesitan al menos, dos hilos de ejecución simultáneos. Uno de ellos que corresponde a la parte del servidor que se comunica con la interfaz web, este se mantiene siempre en ejecución, Y, el segundo para publicar un *tweet* por ejemplo.

- **Existencia clase String**. Este tipo de clase no existe en lenguajes como *C* y el manejo de cadenas de caracteres se vuelve más complicado. Sin embargo, en *Java* se facilita este trabajo al disponer de una clase *String* sólo para cadenas de caracteres. En la aplicación se manejan varias variables de clase *String* por lo que la existencia de esta clase es una gran ventaja.

- **Librería *Twitter4J***. Esta librería desarrollada en Java permite la conexión de la aplicación con cualquier cuenta de *Twitter*.

- **Librería JDBC (*Java DataBase Connectivity)***. Librería que permite la conexión a una base de datos desde una aplicación desarrollada en *Java*.

Aunque *Java* cuenta con multitud de ventajas más, las cuatro mencionadas han sido las más relevantes para su elección en el desarrollo del proyecto. A continuación, se van a explicar las librerías mencionadas con más detalle.

Twitter4J es una librería no oficial para *Java* gratuita de código abierto. Cualquier persona que se haga miembro de la comunidad puede solucionar errores del código o desarrollar nuevas funciones. El código se encuentra integro subido en la página web *github.com* y continuamente se está integrando código nuevo gracias a la herramienta *Jenkins*.

La librería se descarga desde la página web y se añade al proyecto de *Java* en el que se quiera usar. Sin embargo, antes de poder comenzar a programar es necesario registrar nuestra aplicación en la sección de desarrollo de *Twitter*. De esta manera, se obtienen la *Consumer Key*, la *Consumer Secret*, la *Access Token* y la *Access Token Secret*. Estas claves son necesarias para poder realizar la conexión de la aplicación con *Twiter* ya que son únicas para cada cuenta. En este trabajo, la cuenta utilizada es *@medquizzes*.

Las claves obtenidas se encuentran guardadas en un fichero de configuración al que la aplicación accede a la hora de conectarse a *Twitter* en cada una de las tareas que desarrolla. Al estar en un fichero externo, la modificación es bastante sencilla si en el futuro se realiza un cambio de cuenta.

Las acciones que permite la librería *Twitter4j* y son usadas en este trabajo son:

- **Publicar un *tweet***. Gracias al método *updateStatus()* puede publicarse un tweet. El texto de la publicación se pasa como parámetro y si se quiere añadir multimedia tan sólo hay que usar la función *setMedia()* sobre un objeto *StatusUpdate* pasando como parámetro el archivo multimedia que queramos insertar.

- **Búsqueda de *tweets***. El método necesario para llevar a cabo una búsqueda de *tweets* es *search(twitter4j.Query)*. A este método hay que pasarle como parámetro un objeto de la clase *Query* indicando qué es exactamente lo que se quiere buscar. En mi aplicación, el objeto *Query* contiene tan sólo el *hashtag* deseado.

La librería permite muchas más acciones aparte de las usadas, como buscar los *tweets* favoritos de una cuenta, mandar mensajes directos, recuperar los últimos *tweets* de una cuenta determinada y mucho más.

JDBC es una librería cuyo objetivo es ejecutar sentencias SQL. Estas sentencias pueden ser enviadas a cualquier base de datos relacional sin importar del tipo que sean, pueden ser *Oracle* o *Access* pero el código a ejecutar será el mismo. Se puede descargar fácilmente desde la página de descargas de MySQL. Sintetizando, *JDBC* es capaz de realizar las siguientes tareas:

- **Establecer conexión con base de datos**. Gracias a un conjunto de funciones, se puede crear una conexión a la base de datos desde *Java* insertando algunos identificadores de la base de datos como el nombre y la contraseña.

- **Enviar sentencias SQL**. La consulta de la base de datos es bastante sencilla ya que la sintaxis en la que se escriben las consultas es la misma que en SQL.

- **Procesar resultados**. Los resultados de las consultas realizadas se pueden procesar con varios métodos. Uno de los ejemplos es obtener cada elemento de una entrada de una tabla de la base de datos.

Las clases de esta librería usadas en este trabajo son:

- **DriverManager**. Esta clase es la capa gestora de *JDBC* que trabaja entre el usuario y el controlador. Establece la conexión entre la base de datos y el controlador apropiado.

- **Connection**. Un objeto de esta clase representa una conexión a una base de datos que incluye las sentencias SQL ejecutadas y los resultados devueltos. Una aplicación puede tener varias conexiones a la vez con la misma base de datos o distintas.

- **Statement**. Objeto usado para enviar las sentencias SQL a una base de datos una vez que se ha establecido la conexión.

- **ResultSet**. Contiene todos los registros que cumplen las condiciones impuestas en la sentencia SQL. Además proporciona acceso a los datos que se guardan en ellos mediante un conjunto de métodos *get* que permiten acceder a los campos del registro que contiene el objeto *ResulSet*.

La utilización de las clases listadas ha permitido poder obtener los *tweets* a publicar de la base de datos. También ha permitido guardar los *tweets* procedentes de las búsquedas realizadas por *hashtag*.

4.4.3. MÓDULO DE PUBLICACIÓN DE TWEETS

Uno de los objetivos del trabajo es publicar un *tweet* diario en una imagen junto a una pregunta y cuatro posibles respuestas de las que el usuario deberá elegir la que crea correcta.

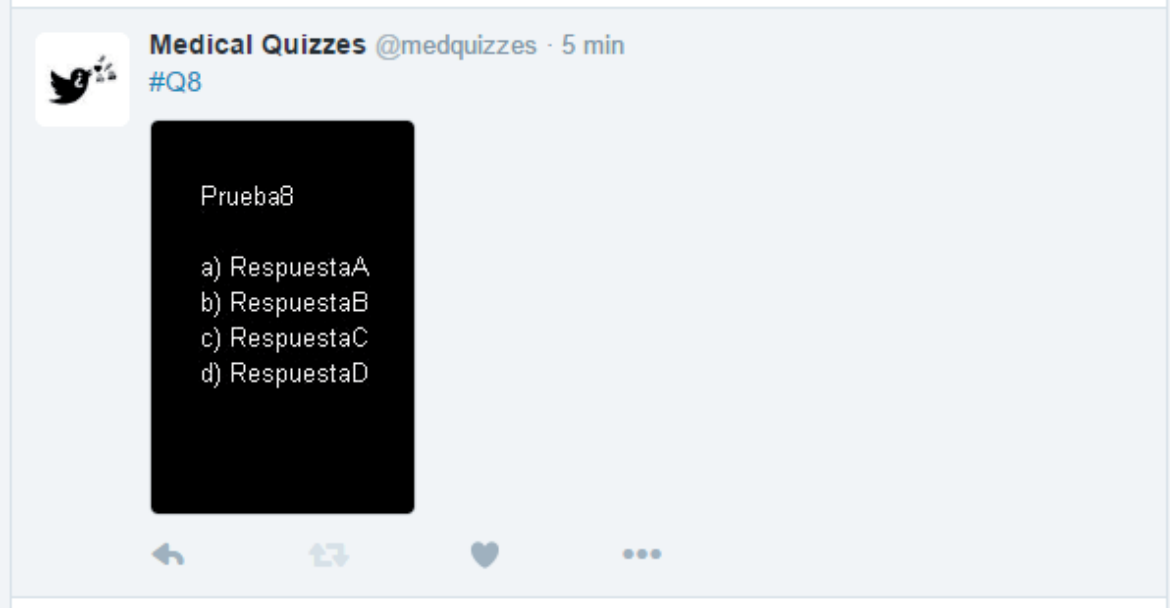

Ilustración 6: Tweet publicado

Aparte de la publicación del *tweet* diario, otra de las funcionalidades es que debe publicar un *tweet* bajo demanda. Esta petición es realizada por la interfaz web en cualquier momento.

Para la implementación de este objetivo se han creado dos clases en el proyecto de *Java*. Por un lado, existe una clase llamada *DailyTweet* cuya misión es conectarse a la base de datos existente y comprobar el identificador del primer *tweet* que aparece como no publicado. El *tweet* correspondiente a este identificador será el que se publicará. Para proceder a su publicación, este identificador se pasa como argumento a la otra clase que comprende este módulo, *SendTweet*.

La segunda clase mencionada se encarga de conectarse a la base de datos para recuperar el texto de la pregunta que corresponde al identificador pasado como parámetro. Además, también obtiene las cuatro posibles respuestas almacenadas en la base de datos. Con toda la información crea una imagen. Después, se conecta a *Twitter* y publica un tweet con la imagen y el *hashtag* correspondiente.

A continuación, se va a realizar un análisis más técnico de cada una de las dos clases mencionadas anteriormente.

La clase *DailyTweet* cuenta con un método en su interior. El método se llama *run()*. La existencia de este método se justifica ya que la clase *DailyTweet* extiende *TimerTask* para lograr programar la ejecución de la tarea a la misma hora todos los días. Y cualquier clase que extienda *TimerTask* necesita el método *run()* para poder ejecutarse. El principal objetivo de este método es buscar en la base de datos el *tweet* que corresponde publicar. Es una función que no devuelve ningún valor. El paso a paso que sigue este método es el siguiente:

- **Conexión a base de datos**. Se conecta a la base de datos mediante JDBC leyendo los datos de acceso desde un archivo externo.

- **Realización de la consulta**. Se crea una consulta que obtenga la primera entrada con la columna “publicado” a cero.

- **Llamada a otra clase**. Se realiza la llamada a la clase *SendTweet* con el identificador obtenido en la consulta.

La clase *SendTweet()* es ejecutada tras la llamada de la clase explicada anteriormente o a petición de la interfaz web. Esta clase cuenta con cuatro métodos. El primero de ellos *SendTweet (int id)* es el método al que llama la clase explicada anteriormente. No devuelve nada. Las acciones que sigue esta función son las siguientes:

- **Conexión a base de datos**. Se conecta a la base de datos mediante JDBC leyendo los datos de acceso desde un archivo externo.

- **Realización de la consulta**. En este caso, se crean dos consultas. La primera, para buscar la entrada de la base de datos cuyo identificador coincida con el identificador con el que se ha llamado a la función. Y la segunda para buscar las cuatro posibles respuestas de la pregunta que se va a publicar.

- **Llamada a otro método**. Se realiza la llamada al método *PublicarTweet(String mensaje, String [] respuestas)*. El primer parámetro corresponde a la pregunta obtenida de la consulta a la base de datos, y el array guarda las cuatro respuestas.

El método *PublicarTweet(String mensaje, String [] respuestas)* es el encargado de, como su nombre indica, publicar la imagen creada en *Twitter*. Los pasos que sigue esta función son:

- **Llamada a otro método**. Llamada al método *Credenciales()* que se explicará a continuación.

- **Llamada a otro método**. Llamada al método *createImage(String imgstr, String pregunta, String[] respuestas)* donde el primer parámetro representa la imagen que servirá de base, el segundo es la pregunta y el tercero corresponde a las respuestas. Este método será comentado más adelante.

- **Creación hashtag**. Contiene una variable que contiene el nombre de hashtag correspondiente dependiendo de la pregunta.

- **Conexión con Twitter**. Se conecta a *Twitter* para actualizar el estado lo que se traduce en la publicación de un nuevo *tweet*.

El método *Credenciales()* se encarga de configurar un objeto *Twitter* incluyendo todas las claves necesarias para poder conectarse a una cuenta especifica de *Twitter*. Las claves mencionadas son leídas desde un fichero externo para facilitar posibles modificaciones en el futuro. El objeto configurado es devuelto por el método para su posterior utilización.

Por último, el método *createImage(String imgstr, String pregunta, String[] respuestas)* es utilizado para crear la imagen que se publica en *Twitter*. Tras el ajuste de varios parámetros, se guarda la imagen en una carpeta para su posterior consulta.

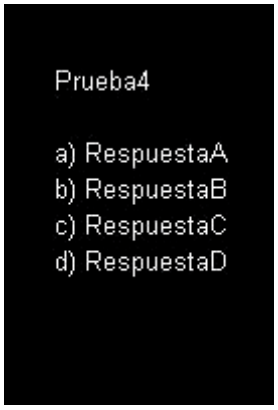

Ilustración 7: Imagen generada con la aplicación

El ancho de la imagen se encuentra ajustado a 50 caracteres. Si el texto de la pregunta es superior, realiza un algoritmo de manera que las palabras no se corten. El largo de la imagen también es variable dependiendo del número de líneas que ocupen la pregunta junto con las respuestas. En la imagen se puede ver cómo las palabras no se cortan al final de línea.

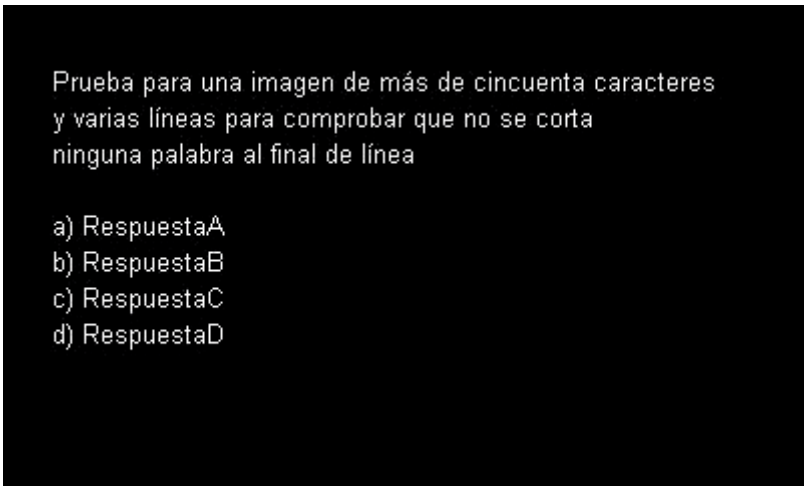

Ilustración 8: Imagen generada con la aplicación con saltos de línea

Explicado el proceso de publicación de *tweets*, en el siguiente punto se lleva a cabo la exposición de la búsqueda de respuestas dadas por los usuarios a cada una de las preguntas formuladas.

4.4.4. MÓDULO DE BÚSQUEDA DE RESPUESTAS

Una vez que se ha publicado una pregunta con sus cuatro posibles respuestas, es necesario obtener las respuestas que los usuarios van escribiendo a cada pregunta. La búsqueda de respuestas se realiza mediante los *hashtags* de *Twitter*. Tal y cómo se explicaba en el apartado anterior, cada *tweet* publicado cuenta con una imagen y un *hashtag* que lo identifica ya que está determinado por el identificador de la pregunta. Con este *hashtag* es con el que se hará la búsqueda. Cada usuario debe contestar cada pregunta con el *hashtag* especificado seguido de la letra que corresponda con su respuesta.

La recuperación de respuestas se realiza diariamente a la misma hora, pero al igual que en caso de la publicación, se puede ejecutar en cualquier momento tras petición en el servidor.

La implementación de este módulo se ha llevado a cabo con una clase en *Java* llamada *TweetRetriever()*. Dado que la tarea tiene está programada para ejecutarse diariamente extiende la clase *TimerTask* de Java como se hacía con la anterior. Esta clase cuenta con tan sólo dos métodos.

Uno de ellos llamado *Credenciales()*, este tiene la misma función que el método con el mismo nombre de la clase *SendTweet()* explicada en el apartado anterior. Su función es configurar la conexión de la aplicación con *Twitter* mediante las claves obtenidas de un fichero externo.

El segundo método lleva por nombre *run()*, ya que como se explicó en el apartado anterior, las clases que extienden *TimerTask* necesitan contar con este método. En esta caso, todo el peso de la clase cae sobre el método *run()*. El paso a paso de este método es el siguiente:

- **Conexión a base de datos**. Se conecta a la base de datos mediante JDBC leyendo los datos de acceso desde un archivo externo.

- **Búsqueda**. Mediante dos bucles anidados se realiza la búsqueda de *hashtags*, buscando por identificador de la pregunta y por letra posible de las respuestas (A, B, C o D). Todos los *tweets* encontrados se van guardando en un objeto de tipo *QueryResult*. Tras terminar la recuperación de *tweets*, se insertan uno a uno en la base de datos para proceder a su estudio en el futuro.

Todas las respuestas recuperadas son las usadas en el estudio de la sabiduría colectiva. Además los datos obtenidos y guardados en la base de datos son accesibles desde la interfaz web de manera que se pueden visualizar.

4.4.5. MÓDULO DE CONEXIÓN CON INTERFAZ

La aplicación desarrollada se conecta a una interfaz web desarrollada por mi compañero Juan Aguado Peña. A través de la interfaz de web se puede ejecutar mi aplicación solicitando la publicación *tweets* o la búsqueda de nuevas respuestas de usuarios bajo demanda. Este servicio puede ser solicitado en cualquier momento y cuantas veces se desee.

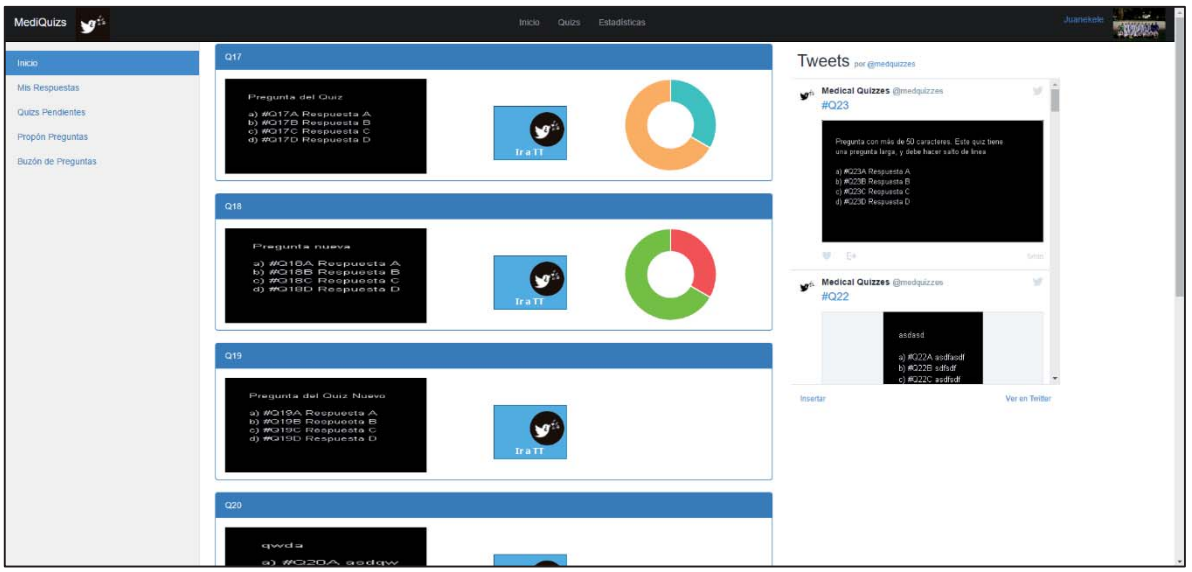

Ilustración 9: Interfaz web desarrollada por Juan Aguado

La aplicación en *Java* hace el papel de servidor y la interfaz web realiza el papel de cliente del servicio. La conexión se ha llevado a cabo mediante la implementación de *sockets* por los que se realiza un intercambio de mensajes. El servidor se encuentra en ejecución constantemente y cuando el cliente decide conectarse, el servidor muestra un mensaje pidiendo la contraseña.

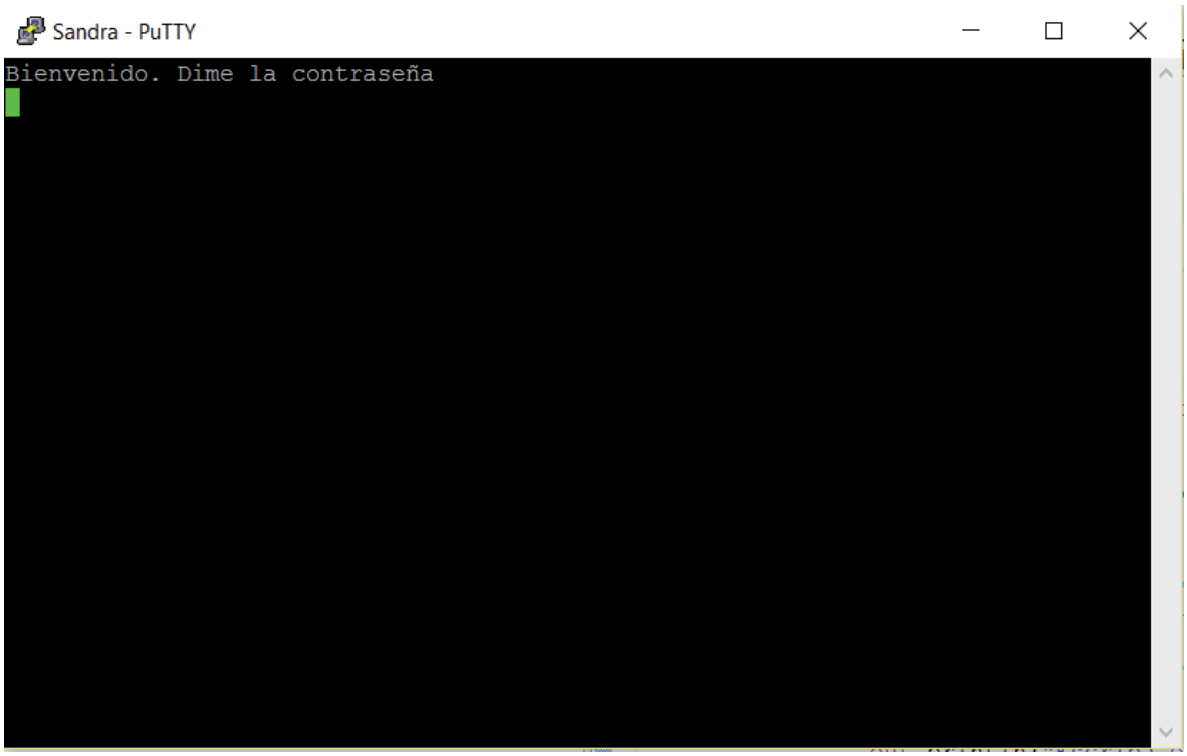

Ilustración 10: Primer mensaje servidor

La contraseña pedida corresponde a "1234". Si el cliente inserta una contraseña incorrecta, el servidor le envía un mensaje dándole la oportunidad de insertar la contraseña correcta. Esta acción se repite tantas veces como contraseñas incorrectas teclee el usuario.

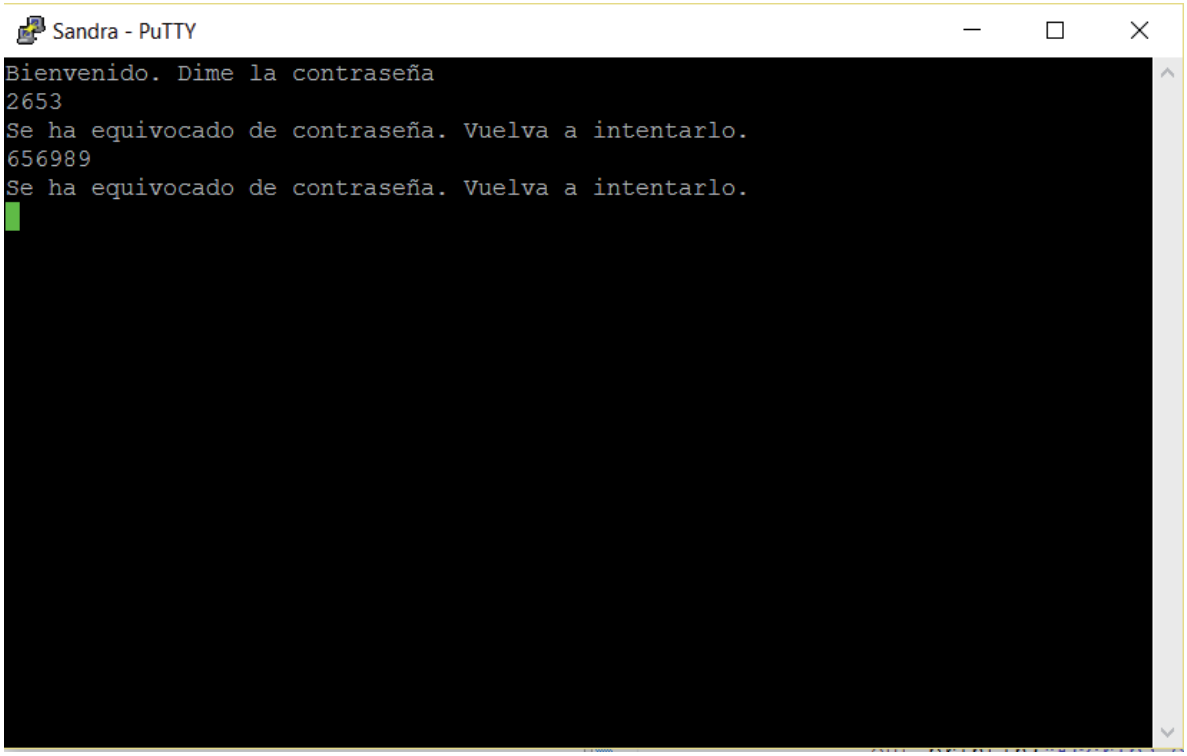

Ilustración 11: Inserción de contraseña incorrecta

Una vez insertada la contraseña correcta, el servidor da a elegir entre dos opciones. Por un lado, da la posibilidad de publicar un *tweet* al momento, como se ha detallado en apartados anteriores, pulsando el número uno. Por otro lado, se puede proceder a una búsqueda de respuestas por parte de los usuarios al pulsar el número dos.

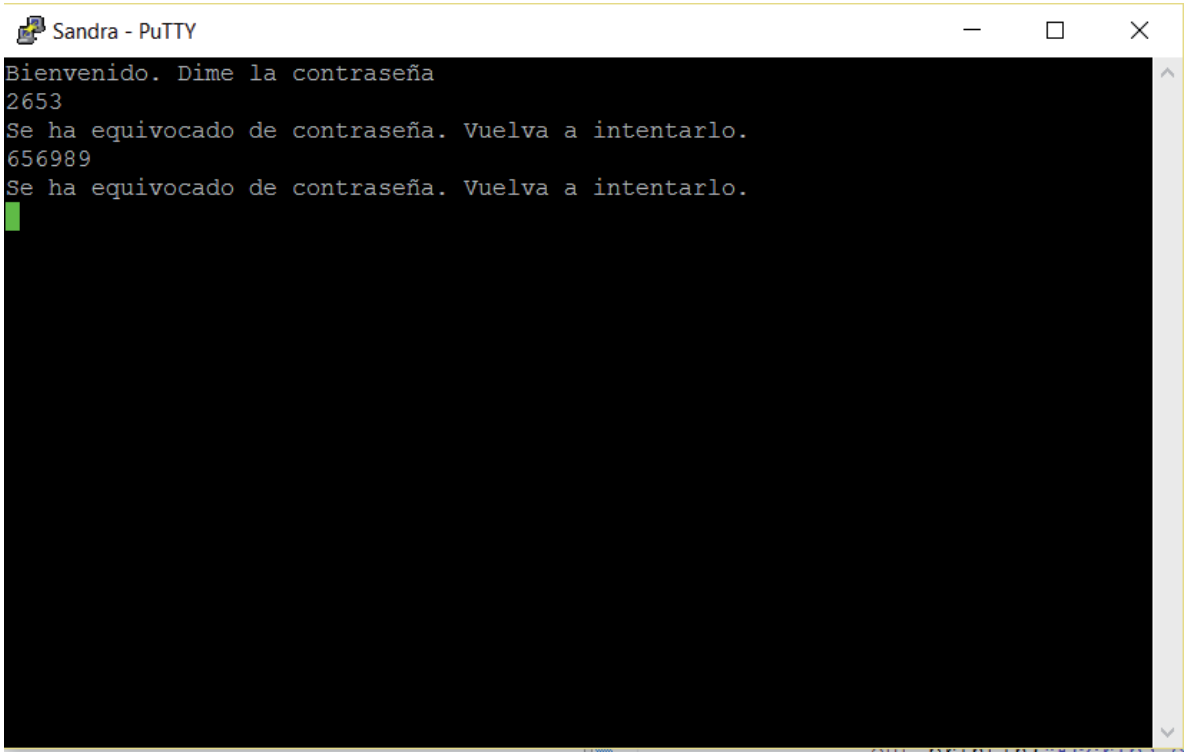

Ilustración 12: Opciones a elegir

Pulsando el número dos se procede a la búsqueda de respuestas. Tras terminar con la búsqueda e insertar los resultados en la base de datos, la conexión se cierra y el servidor se queda esperando a que un nuevo cliente se conecte.

En el caso de pulsar el número dos, el servidor solicita un identificador del *tweet* que se quiere publicar para poder proceder a su búsqueda en la base de datos.

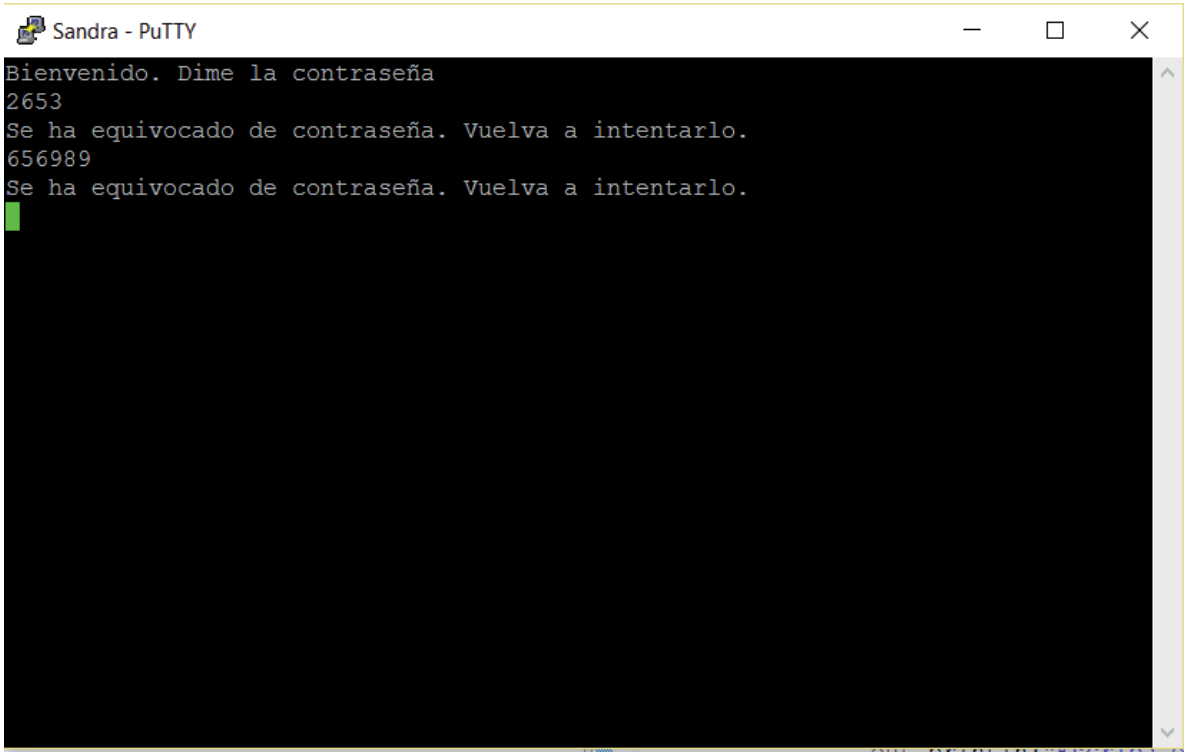

Ilustración 13: Petición de identificador

Una vez que se ha tecleado el identificador correspondiente, la aplicación procede a buscar el *tweet* en la base de datos y este es publicado en *Twitter*. La conexión se cierra cuando la publicación ha terminado.

Si el número introducido no es correcto, el servidor manda un mensaje informando al cliente de su error. Este mensaje se manda tantas veces como el cliente se equivoque.

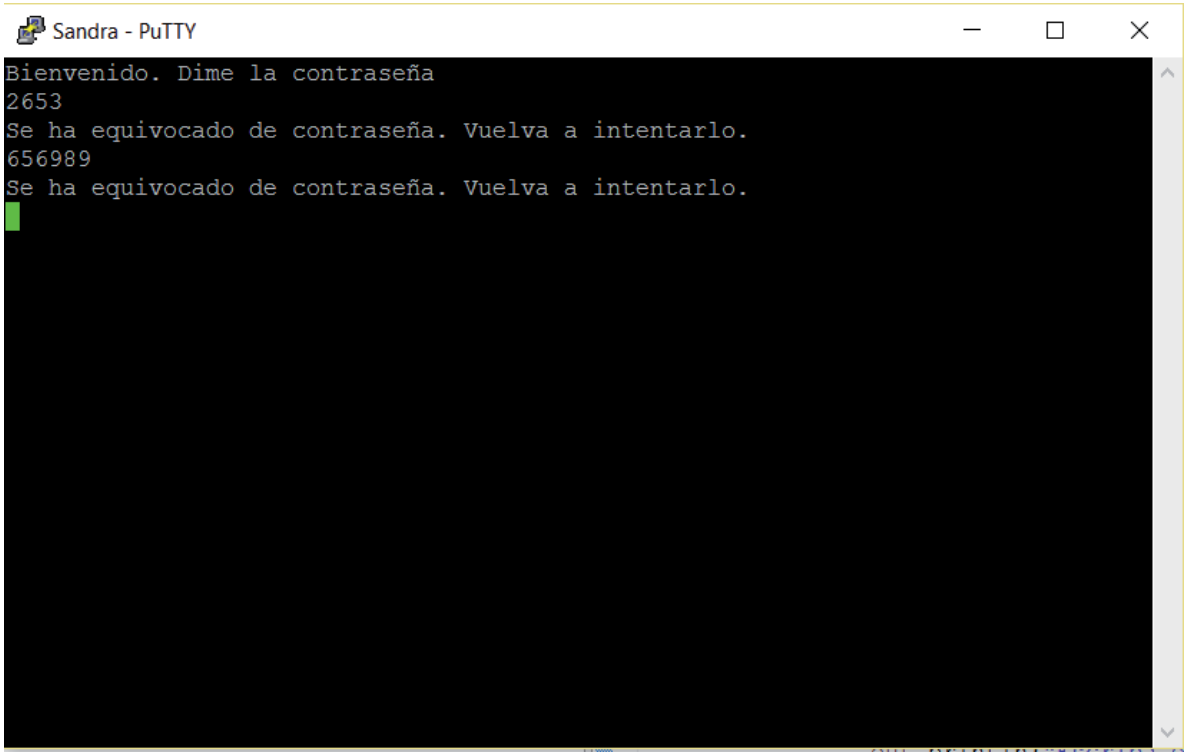

Ilustración 14: Inserción de número incorrecto

La implementación de esta funcionalidad se ha realizado con dos clases de *Java* especializadas en la creación y manejo de *socket*: la clase *Socket* y la clase *ServerSocket*.

El servidor contiene un *socket* asociado a un número de puerto específico a la escucha esperando que se conecte un cliente. En mi caso, el puerto utilizado para la escucha es veintidós. El cliente debe de conocer el nombre de la máquina en la que se encuentra el servidor y el número de puerto al que debe de conectarse. En mi aplicación, el nombre de la máquina no es necesario ya que cliente y servidor se encuentran ejecutando en la misma máquina.

Cuando el cliente quiere solicitar un servicio al servidor debe intentar estableces conexión con este último. Cuando el servidor está preparado, acepta la conexión. En un caso general, el servidor debería de abrir otro *socket* en un puerto diferente para atender la petición y mantener libre otro para escuchar futuras peticiones de otros clientes. En este trabajo no ha hecho falta mantener dos *sockets* abiertos mientras se atiende una petición ya que sólo se

cuenta con un cliente y mientras se le da servicio no existe otro cliente que vaya a solicitar permiso.

La clase *Socket* es la utilizada para la creación de los clientes y la clase *ServerSocket* se utiliza para la creación de los servidores. Este módulo ha sido implementado en una sola clase de *Java* llamada Server usando las clases mencionadas antes. Esta clase extiende la clase *Thread* ya que para que el servidor se encuentre ejecutando en todo momento, la llamada a la función se realiza dentro de un bucle infinito donde cada vez que se inicia una iteración del bucle se crea un *thread* y este tiene que esperarse a terminar antes de comenzar otra iteración. La clase *Server* sólo cuenta con un método llamado *run()*, igual que ocurría que en casos anteriores, este método es necesario porque es lo que se va a comenzar a ejecutar cuando se invoque la función *start()* de la clase *Thread*. Los pasos que sigue el método son los siguientes:

- **Creación socket**. Se crea un *socket* servidor asignado al número de puerto decidido previamente.

- **Conexión**. Se acepta la conexión con el *socket* cliente.

- **Intercambio de mensajes**. Durante la conexión el servidor manda una serie de mensajes y recoge los devueltos por el cliente para verificar que son correctos como se ha visto en las imágenes anteriores. Además, dependiendo de los mensajes mandados por el cliente, el servidor tiene que llevar a cabo distintas llamadas a otras clases.

- **Cierre de conexión**. Cuando el intercambio de mensajes finaliza, se debe de cerrar la conexión para que el *socket* quede liberado. De esa manera, vuelve a esperar quedándose a la escucha de futuras peticiones.

Los pasos descritos se ejecutan una y otra vez siempre que termina una petición y comienza otra.

4.4.6. EJECUCIÓN

La ejecución de la aplicación corre a cargo de una clase llamada *App* que sólo contiene un método *main()*. Así se ha creado una clase con el objetivo de llevar a cabo la ejecución del programa sin sobrecargarla con otras funciones.

Dentro del método *main()* se encuentra el bucle infinito que pone a funcionar la clase *Server* mediante la función *start()* de la clase *Thread*. Antes de terminar una iteración del bucle se llama a la función *join()*, también de la clase *Thread*, para evitar que se inicie otra iteración del bucle creando otro hilo de ejecución, lo que llevaría a errores por compartición de variables.

La otra tarea que se lleva a cabo en esta clase es la de programar la publicación de *tweets* y la búsqueda de respuestas a la misma hora todos los días. Para ello, las clases a ser programadas extienden la clase *TimerTask* de manera que todo el código que se encuentre dentro de sus métodos *run()* será ejecutado en la fecha programada. Lo único que se necesita crear en la función *main()* es un objeto *Timer* con el que se llamará a la tarea indicada. Se crean dos objetos *Date* en los que se determina la hora de llamada a cada uno. La clase *DailyTweet()* se ha determinado que se ejecute diariamente a las nueve de la mañana por lo que su objeto *Date* asociado es creado con esa hora. La clase *TweetRetriever()* se ejecuta a las once de la noche, así que su objeto *Date* tiene esta hora. Tras crear las fechas basta con llamar al método *schedule()* de la clase *Timer* que programará las tareas

5. RESULTADOS

Este capítulo se centra en los resultados obtenidos de la aplicación desarrollada. Primero, se van a detallar los problemas encontrados durante su implementación y posterior puesta en marcha. Después se expondrán los resultados que se han obtenido del funcionamiento de la aplicación. Para finalizar, se resumen las conclusiones a las que se ha llegado.

5.1. PROBLEMAS ENCONTRADOS

Las dificultades y obstáculos que han ido apareciendo durante estos tres meses de trabajo han sido abundantes por lo que no se van a exponer todos. En su lugar, se van a detallar aquellos que más han afectado al proyecto y a su desarrollo.

La red social *Twitter* impone un límite de *tweets* que se pueden obtener diariamente realizando consultas. En la aplicación desarrollada, este límite podía ocasionar un problema en la búsqueda de respuestas de usuarios ya que cruzado el límite no devuelve respuestas a las consultas realizadas. Aunque el límite permite obtener una gran cantidad de *tweets* y no ocasiona problema en la búsqueda programada para realizarse todas las noches, sí que se ha llegado a ese límite en algunos momentos del desarrollo. Concretamente, durante la codificación del módulo encargado de esta búsqueda han sido necesarias pruebas unitarias en las que se comprobase que el módulo funcionaba correctamente. Tras llamar al método encargado de ello unas cuantas veces, se llegaba al límite lo que no permitía proseguir con el desarrollo o con la comprobación del funcionamiento durante ese día. Lo mismo ocurrió cuando se realizaron pruebas integradas junto a la aplicación web por la misma razón.

Uno de los objetivos de este trabajo era el estudio de la inteligencia colectiva realizando un análisis de las respuestas dadas a las preguntas publicadas. Un estudio de estas características necesita una gran cantidad de datos de los que extraer información con los que difícilmente se ha podido contar ya que la aplicación ha sido imposible tenerla puesta en marcha durante todo el día. Además, tampoco se contaba con preguntas médicas que poder publicar. En un futuro, cuando se ponga la aplicación en funcionamiento, se podrá llevar a cabo el estudio de la sabiduría colectiva respecto a datos médicos.

La creación de la imagen fue una dificultad que estuvo presente durante casi todo el desarrollo de la aplicación. La imagen se generaba correctamente el problema recaía en que no aparecían los saltos de línea por lo que no se distinguían las preguntas y las respuestas. Además si la pregunta era muy larga aparecía entera en la imagen con la letra demasiado reducida. El problema se pudo solventar con ayuda del tutor del trabajo y como se puede comprobar la imagen se genera perfectamente.

Aunque han aparecido otros problemas, han sido fácilmente solucionables. La mayoría de ellos han sido en relación con la base de datos y su utilización. Por ejemplo, problemas al insertar *tweets* procedentes de la búsqueda, aparición de excepciones por el formato en el que se escriben algunos *tweets* entre otros.

5.2. RESULTADOS

El apartado anterior exponía el problema de falta de tiempo para la obtención de un gran número de respuestas por parte de los usuarios que se pudiesen estudiar en el futuro. Debido a ello, en este apartado tan sólo se pueden explicar los resultados obtenidos de las pruebas unitarias e integradas realizadas durante el proyecto.

Durante las pruebas unitarias de la aplicación se han publicado *tweets* bajo demanda del servidor. De esta manera, se comprobaba que las dos funcionalidades trabajaban correctamente. Los *tweets* publicados cumplían todos los requisitos pedidos, es decir, contaban con el *hashtag* correspondiente y la imagen aparecía legible y con los textos de la pregunta que se había solicitado.

Por otro lado, se realizaron pruebas para la obtención de respuestas por parte de los usuarios. Como ya se ha explicado, no se ha contado con respuestas reales por lo que al principio se han realizado pruebas con *hashtags* que fuesen *trending topic* para comprobar que el módulo realizaba las búsquedas correctamente e insertaba los datos en la base de datos. Todas las tareas se realizan correctamente.

La última funcionalidad probada unitariamente ha sido la programación de tareas. Cada una de las dos tareas a ejecutar diariamente han sido programadas y la aplicación las ha ejecutado a la hora pedida.

Tras la integración con la aplicación web, se han vuelto a realizar las mismas pruebas completándose satisfactoriamente.

Aunque no se pueden mostrar resultados del análisis de las respuestas de los usuarios, puede verse como resultado todo el trabajo realizado durante estos últimos meses se ejecuta correctamente.

5.3. LINEAS FUTURAS

Este proyecto ha servido como punto de inicio de una investigación que se desarrollará en el futuro. Entre mi compañero Juan Aguado y yo hemos creado la semilla de una aplicación que cumplirá todas las funcionalidades explicadas en este trabajo. Primeramente, la base de datos debe ser llenada con preguntas relacionadas con el ámbito médico aportadas por un profesional sobre el tema. Después, la aplicación tiene que ser alojada en un servidor en el que pueda ejecutarse sin interrupción de manera que publique una encuesta y recoja respuestas diariamente.

Cuando la aplicación lleve un tiempo en ejecución y existan varios usuarios registrados en la aplicación web, se podrá contar con múltiples respuestas para su análisis.

Además, futuros desarrolladores podrán añadir más funcionalidades al software para conseguir mejores resultados o búsqueda más eficientes.

Como conclusión, después del desarrollo de esta aplicación queda un gran trabajo por delante

6. ANEXOS

Este capítulo recoge el manual de usuario necesario para poner en funcionamiento la aplicación que ha sido desarrollada.

6.1. MANUAL DE USUARIO

La aplicación se encuentra en un fichero con extensión jar por lo que para ponerlo en funcionamiento tan sólo hay que hacer doble click sobre él. Una vez que se encuentra ejecutando, a las ocho de la mañana publicará un *tweet* con la pregunta correspondiente. A las once de la noche se producirá una búsqueda de respuestas y se podrá ver en la base de datos como la tabla *tweet* contiene *tweets* recogidos.

Mientras la aplicación se encuentra en ejecución se puede pedir que publique *tweets* o realice búsquedas. Para ello, se ha utilizado la herramienta *PuTTY* que actúa como cliente para conectarse al servidor. Como puerto, se especifica el número 22 y como *host name* escribimos la palabra *localhost* ya que servidor y cliente se encuentran en la misma máquina. Para el tipo de conexión se utiliza una conexión *raw*.

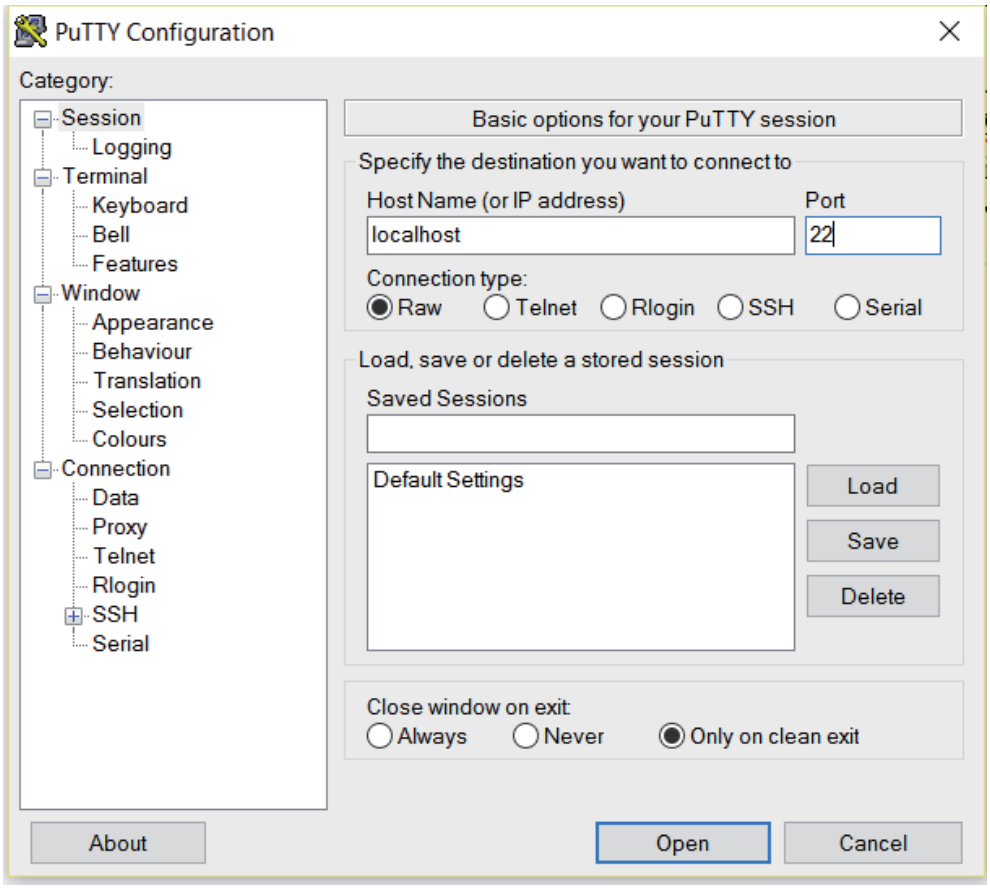

Ilustración 15: Configuración de PuTTY

Pulsando el botón *Open* se abrirá la conexión con la aplicación *Java* y podrá dar comienzo el intercambio de mensajes.

7. BIBLIOGRAFÍA

Surowiecki, J. (2005). *Cien mejor que uno*. Barcelona: Urano.

Twitter4j.org. (2016). *Twitter4J - A Java library for the Twitter API*. [online] Available at: http://twitter4j.org/en/index.html [Accessed 1 Mar. 2016].

Oracle.com. (2016). *Java SE Technologies - Database*. [online] Available at: http://www.oracle.com/technetwork/java/javase/jdbc/index.html [Accessed 10 Mar. 2016].

Infor.uva.es. (2016). *JAVA*. [online] Available at: http://www.infor.uva.es/~jmrr/tgp/java/JAVA.html [Accessed 1 Mar. 2016].

Rodríguez-González, A., Menasalvas Ruiz, E. and Mayer Pujadas, M. (2016). Automatic extraction and identification of users' responses in Facebook medical quizzes. *Computer Methods and Programs in Biomedicine*, 127, pp.197-203.

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Sun Jun 05 17:09:00 CEST 2016
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)