UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

“DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA PARA EL MONITOREO DE TWITTER”

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

ALEJANDRO AGAPITO BAUTISTA

DIRECTOR DE TESIS

ING. JORGE ALBERTO RODRÍGUEZ CAMPOS

CIUDAD UNIVERSITARIA 15 /Agosto/ 2014.

A MI MADRE

Agradecimientos

A mi madre y padre por darme su apoyo constante e incondicional a lo largo de toda mi vida, por los valores que me han inculcado y por guiarme para lograr ser la persona que ahora soy, siempre encontraron la forma de estar ahí a pesar de las dificultades, no lo habría logrado sin su apoyo. Gracias por ser un excelente ejemplo de vida a seguir.

A mi hermano por ser la persona a la que más confianza tengo, por cuidarme y apoyarme en momentos difíciles, gracias por llenar mi vida de alegrías y amor cuando te he necesitado.

A mi pareja por tu paciencia, comprensión y cariño, me inspiraste a ser mejor para ti, gracias por estar siempre a mi lado.

A mis amigos y familiares por ser esas personas importantes en mi vida, por brindarme su ayuda y por compartir su vida conmigo, gracias por todos esos momentos felices.

A mi director de tesis y profesores en la universidad, por sus lecciones y experiencias que ayudaron a formarme y a desarrollar mis habilidades, a todos y cada uno de ellos dedico cada una de estas páginas de mi tesis.

A la Universidad Nacional Autónoma de México, fue un sueño realizar mis estudios en esta universidad, gracias por todo el conocimiento.

Al Programa de tecnología en cómputo PROTECO por ayudarme como complemento a lo largo de mi carrera así como por el apoyo y confianza que me brindaron, entrar fue la mejor decisión de mi carrera.

Desarrollo e implementación de un sistema dedicado al monitoreo, análisis y clasificación de Twitter

Por

Alejandro Agapito Bautista

Resumen

El concepto de redes sociales consiste en un conjunto de individuos y una relación social que los une. En esta tesis se crea un sistema capaz de realizar análisis entre los individuos de la red social y sus relaciones a través de algoritmos que permiten analizar el impacto de las publicaciones, así como conocer el sentimiento de las mismas. El sistema muestra de manera general el impacto que se tiene sobre algún actor, evento o compañía en un determinado momento y en un determinado lugar, todo esto utilizando mapas, reportes y gráficas que permiten un entendimiento claro de la información. El sistema es creado en primera instancia para el análisis de la red social Twitter, para los siguientes pasos se pretende realizar análisis de otras redes sociales tales como Facebook, Youtube y Foursquare.

Director de tesis: Jorge Alberto Rodríguez Campos.

Tabla de contenido

1. ANTECEDENTES 10

1.1. TEMA 11

1.2. PROBLEMA A RESOLVER 11

1.3. PREGUNTAS DE INVESTIGACIÓN 12

1.4. OBJETIVOS 12

1.5. HIPÓTESIS 12

2. MARCO TEÓRICO 13

2.1. REDES SOCIALES 14

2.2. TWITTER 14

2.3. ANÁLISIS DE REDES SOCIALES 17

2.4. ALGORITMOS PARA ANÁLISIS DE TEXTOS 19

2.1. JAVA ENTERPRISE EDITION 22

2.6. SPRING 32

2.7. SERVICIOS WEB 33

2.7. INGENIERÍA DE SOFTWARE 35

2.8. PRUEBAS DE SOFTWARE 39

3. ANÁLISIS Y DISEÑO 41

3.1. ¿QUÉ ES SOCIAL ANALYSIS? 42

3.2. ANÁLISIS DE REQUERIMIENTOS 42

3.3. ARQUITECTURA DEL SISTEMA 48

3.4. DISEÑO DE LA BASE DE DATOS 52

3.5. DISEÑO DETALLADO 55

4. DESARROLLO DEL SISTEMA 67

4.1. ESTRUCTURA DE LA APLICACIÓN 68

4.2. MODELO VISTA CONTROLADOR 71

4.3. UTILIZACIÓN DE SERVICIOS WEB REST 80

4.4. JAVASERVER FACES Y PRIMEFACES EN SOCIAL ANALYSIS 84

5. PRUEBAS Y ANÁLISIS DE RESULTADOS 96

5.1. PRUEBAS EN SOCIAL ANALYSIS 97

6. DESPLIEGUE DEL SISTEMA 125

6.1. DESPLIEGUE 126

6.2. DESPLIEGUE DE SOCIAL ANALYSIS 132

7. CONCLUSIONES 139

GLOSARIO DE TÉRMINOS 142

BIBLIOGRAFÍA 148

Tabla de figuras

Figura 1: Tuit sobre Barack Obama 16

Figura 2: Seguidores de personajes públicos. 17

Figura 3: Análisis de redes sociales. 19

Figura 4: Diseño de capas en Java Enterprise Edition 24

Figura 5: Inversion of Control 33

Figura 6: Interoperabilidad 35

Figura 7: Arquitectura del sistema 52

Figura 8: Diagrama de la base de datos del sistema 53

Figura 9: Módulo mapas, opción buscar 56

Figura 10: Módulo mapas, opción gráficas 56

Figura 11: Módulo de mapas, opción gráficas pt 2 58

Figura 12: Módulo de mapas, sección table de resultados 59

Figura 13: Módulo de búsqueda general 62

Figura 14: Módulo de Perfiles 64

Figura 15: Módulo de time line 66

Figura 16: Estructura del proyecto 68

Figura 17: Proyecto padre 69

Figura 18: Social analysis model 69

Figura 19: Social analysis core 70

Figura 20: Social analysis web 71

Figura 21: Modelo vista controlador 73

Figura 22: Modelo vista controlador en Social analysis 73

Figura 23: Consumo de web services REST utilizando RestTemplate..... 80
Figura 24: Uso de PrimeFaces en social analysis 91
Figura 25: Mapas de Google utilizando PrimeFaces 94
Figura 26: Manejo de eventos en los mapas de Google..... 94
Figura 27: Pruebas unitarias utilizando JUnit 100
Figura 28: Pruebas funcionales en el módulo de mapas prueba 1..... 101
Figura 29: Pruebas funcionales en el módulo de mapas, prueba 2..... 102
Figura 30: Pruebas funcionales en el módulo de mapas prueba 3..... 103
Figura 31: Pruebas funcionales en el módulo de mapas prueba 4..... 104
Figura 32: Pruebas funcionales en el módulo de mapas prueba 5..... 105
Figura 33: Pruebas funcionales módulo de mapas prueba 6..... 106
Figura 34: Pruebas funcionales módulo de mapas prueba 7..... 107
Figura 35: Pruebas funcionales módulo de mapas prueba 8..... 108
Figura 36: Pruebas funcionales módulo de mapas, prueba 9..... 109
Figura 37: Pruebas funcionales módulo de mapas, prueba 10..... 110
Figura 38: Pruebas funcionales módulo de mapas, prueba 11..... 110
Figura 39: Pruebas funcionales módulo de mapas, prueba 12..... 111
Figura 40: Pruebas funcionales módulo de mapas, prueba 13..... 112
Figura 41: Pruebas funcionales módulo de mapas, prueba 14..... 113
Figura 42: Pruebas funcionales módulo de mapas, prueba 15..... 114
Figura 43: Pruebas funcionales módulo de mapas, prueba 16..... 114
Figura 44: Pruebas funcionales módulo de mapas, prueba 17..... 115
Figura 45: Pruebas funcionales módulo de mapas, prueba 18..... 116
Figura 46: Pruebas funcionales módulo de mapas, prueba 19..... 117

Figura 47: Pruebas funcionales módulo de mapas, prueba 20 118
Figura 48: Pruebas funcionales módulo de mapas, prueba 21 118
Figura 49: Pruebas funcionales módulo de mapas, prueba 22 119
Figura 50: Pruebas funcionales módulo de mapas, prueba 23 119
Figura 51: Pruebas funcionales módulo de búsqueda general, prueba 24 . 120
Figura 52: Pruebas funcionales módulo de búsqueda general, prueba 25 . 120
Figura 53: Pruebas funcionales módulo de búsqueda general, prueba 26 . 121
Figura 54: Pruebas funcionales módulo de perfiles, prueba 27 122
Figura 55: Pruebas funcionales módulo de perfiles, prueba 28 123
Figura 56: Estructura de directories Apache Tomcat 128
Figura 57: Consola de Jelastic para la generación de ambientes 133
Figura 58: Generación de ambiente utilizando Jelastic 133
Figura 59: Manejo de memoria utilizando Jelastic 134
Figura 60: Precio por mes en Jelastic 135
Figura 61: Tiempo para la creación del entorno 135
Figura 62: Importación de la base de datos a la nube de Jelastic 136
Figura 63: Agregar un recurso al servidor de Jelastic 137
Figura 64: Despliegue de la aplicación utilizando la consola de Jelastic 137
Figura 65: Abrir la aplicación desde la consola de Jelastic 138
Figura 66: Aplicación desplegada en la web 138

Capítulo 1

1. Antecedentes

Antecedentes

1.1. Tema

Análisis de redes sociales, minería de textos, desarrollo de software, bases de datos e ingeniería de software.

1.2. Problema a resolver

Sin lugar a duda, una de las actividades más frecuentes en Internet es el uso de redes sociales tales como Facebook, Twitter e Instagram. Con esto se genera un gran cúmulo de información difícil de medir, clasificar, localizar y monitorear.

En la industria de la televisión y la radio se puede conocer el impacto de algún programa de televisión o alguna marca a través de factores como el rating y con esto saber a cuantas personas está llegando el mensaje que se desea transmitir. Este concepto es útil pero no es aplicable en redes sociales. Este mar de información que producen las redes, puede representar una fuente invaluable para predecir o inclusive influir de forma importante en la toma de decisiones acerca de una oportunidad de mejora, negocio, etc.

Para llegar a este punto es fundamental contar con una clasificación, análisis y procesamiento de la información. Los encargados del marketing de empresas o de personas públicas se encuentran con el problema de tener un número muy grande de personas hablando de ellos dentro de redes sociales y no contar con una forma de saber cuál fue realmente el impacto que se tuvo, si este fue positivo o negativo para ellos o inclusive conocer si se está hablando de ellos en otras partes del mundo. Este es un problema muy relevante, ya que el número de personas que escriben en redes sociales y la cantidad de información que

publican es muy alta y una persona no podría leer toda esa información para poder definir si esta es buena mala o neutra.

1.3. Preguntas de investigación

- ¿Es posible realizar una clasificación de la información que proviene de redes sociales?
- ¿De qué modo se puede clasificar la información de tal forma que genere valor para un cliente o usuario final?

1.4. Objetivos

- Desarrollar un método el cual permita clasificar la información proveniente de la red social Twitter.
- Generar una herramienta capaz de plasmar esa clasificación de tal forma que ofrezca valor a un cliente.
- Generar una herramienta capaz de representar el estado de un acontecimiento en diferentes partes del mundo.

1.5. Hipótesis

Utilizando algoritmos de análisis de textos se clasificará información proveniente de Twitter y se representará a través de una herramienta que brinde una perspectiva clara sobre marcas, hechos, personas o eventos de un cliente, además de poder representar su estado en diferentes partes del mundo.

Capítulo 2

2. Marco teórico

Marco teórico

2.1. Redes sociales

Una red social es una forma de representar una estructura social, asignándole un grafo. Si dos elementos del conjunto de actores (tales como individuos u organizaciones) están relacionados de acuerdo a algún criterio (relación profesional, amistad, parentesco, etc.), entonces se construye una línea que conecta los nodos que representan a dichos elementos. El tipo de conexión representable en una red social es un lazo interpersonal, que se puede interpretar como relaciones de amistad, parentesco, laborales, entre otros.

Investigaciones han mostrado que las redes sociales constituyen representaciones útiles en muchos niveles, desde las relaciones de parentesco hasta las relaciones de organizaciones a nivel estatal (se habla en este caso de redes políticas), desempeñando un papel crítico en la determinación de la agenda política y el grado en el cual los individuos o las organizaciones alcanzan sus objetivos o reciben influencias.

2.2. Twitter

Es una aplicación web gratuita de microblogging que reúne las ventajas de los blogs, las redes sociales y la mensajería instantánea. Esta nueva forma de comunicación, permite a sus usuarios estar en contacto en tiempo real con personas de su interés a través de mensajes breves de texto a los que se denominan actualizaciones o tuits, por medio de una sencilla pregunta: ¿Qué estás haciendo?. Los usuarios envían y reciben actualizaciones de otros usuarios

a través de breves mensajes que no deben superar los 140 caracteres, vía web, teléfono móvil, mensajería instantánea o a través del correo electrónico.

En la sección de la línea del tiempo se puede estar al día de las cuentas a las que sigue una cuenta, los seguidores pueden estar al tanto de las actualizaciones que realiza. Además de buscar amigos, familiares, compañeros u otras personas de tu interés, Twitter también ofrece otras opciones, como buscar en otras redes, invitar a amigos por email o seleccionar a usuarios recomendados.

Usuarios registrados pueden leer y escribir tuits pero usuarios no registrados solo pueden leerlos. Los usuarios pueden agrupar mensajes entre sí por tema o tipo de uso de hashtag (palabras o frases que inician con un símbolo "#") del mismo el símbolo "@" seguido de un nombre de usuario se utiliza para mencionar o responder a otros usuarios. Si a un usuario le agrada un tuit que leyó puede tomar la decisión de darle un retuit, esto permite compartirlo con sus propios seguidores o puede tomar la decisión de marcarlo como favorito para así poder verlo en otro momento.

Una palabra o una frase que se escribe constantemente dentro de la red social se dice que es un tema de tendencia *(trending topic*), estos temas son clasificados por región y son actualizados día a día.

Esta red social es ideal para el uso de personas públicas ya que el usuario es quien decide que tuits se mostrarán en su perfil, además de que se puede dar una idea de a cuántas les agradó una publicación y cuántas lo compartieron, en este momento Twitter no proporciona una idea clara del total de personas que pudieron haber leído una publicación o tuit o si esa publicación fue buena o mala.

Marco teórico

Un ejemplo sobre el impacto que tiene Twitter es que el 24 de abril del 2013 la agencia de noticias Associated Press (AP) sufrió un ataque en su cuenta de Twitter desde la cual se publicó lo siguiente:

Figura 1: Tuit sobre Barack Obama

"Breaking: Two Explosions in the White House and Barack Obama is injured"

"Dos explosiones en la Casa Blanca y Barack Obama se lesiona".

Por su parte MarketWatch, portal financiero del Wall Street Journal, aseguró que este tuit falso provocó la caída de las acciones estadounidenses Dow Jones Industrial Average (DJIA) cayeron colocándolas en 120 puntos.

El portavoz de la Casa Blanca, Jay Carney, confirmó que el presidente Barack Obama se encuentra bien.

A continuación se muestra la tabla en la que se presenta una relación entre personajes públicos y la cantidad de seguidores con los que cuenta cada uno:

Imagen	Personaje	Cuenta	Seguidores	Amigos
	Barack Obama	@BarackObama	41,4 millones	653 Mil
	Vladimir Putin	@KremlinRussia_E	148 Mil	19
	Papa Francisco	@Pontifex_es	4,74 Millones	8
	Elena Poniatowska	@Eponiatowska	416 Mil	95

Figura 2: Seguidores de personajes públicos.

Como se puede observar una publicación de Barack Obama podría llegar a 41,4 millones de personas, esto considerando que ni una sola de ellas va a compartirla a sus amigos, si eso sucede este número aumentaría dependiendo del número de seguidores con el que cuente la cuenta que lo compartió. Con esto se puede mostrar de manera clara el impacto que tiene esta red social en todo el mundo.

2.3. Análisis de redes sociales

¿Cómo podemos clasificar las páginas en la Web? ¿Cómo se transmite el VIH? ¿Cómo se explica el éxito o el fracaso de los empresarios en cuanto a sus contactos de negocios? Todas estas preguntas tienen en común, que pueden reformularse con el vocabulario del análisis de redes, una rama de la sociología y matemáticas que se aplica cada vez más, a las preguntas fuera del dominio social.

Marco teórico

Análisis de Redes Sociales (ARS) es el estudio de las relaciones sociales entre un conjunto de actores. La diferencia fundamental entre el análisis de la red y otros enfoques de la ciencia social es el enfoque en las relaciones entre los actores en lugar de los atributos de los actores individuales.

ARS es, pues, un enfoque diferente a los fenómenos sociales y por lo tanto requiere un nuevo conjunto de conceptos y nuevos métodos para la recolección y análisis de datos. El análisis de redes proporciona un vocabulario para la descripción de las estructuras sociales, proporciona modelos formales que capturan las propiedades comunes de todas las redes (sociales) y un conjunto de métodos aplicables para el análisis de redes en general.

Los conceptos y los métodos de análisis de redes se basan en una descripción formal de las redes en forma de gráficos. Los métodos de análisis se originan principalmente a partir de la teoría de grafos como éstos se aplican a la representación gráfica de datos de redes sociales como se muestra en la sifuiente figura. (Análisis de la red también se aplica estadística y probabilística métodos y, en menor medida, las técnicas algebraicas.)

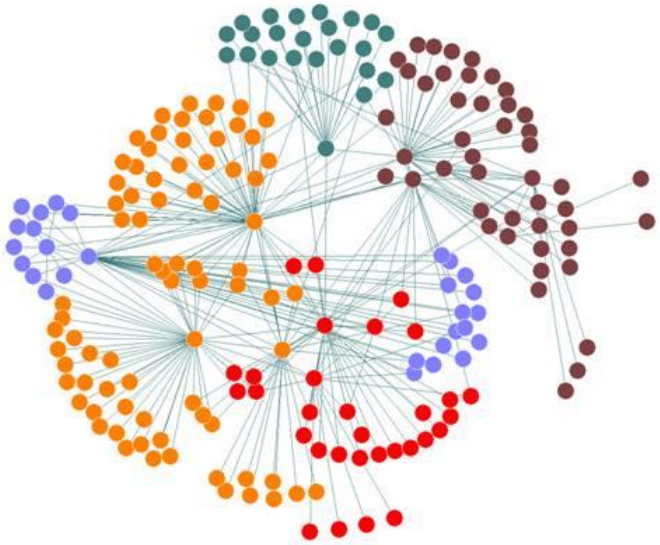


Figura 3: Análisis de redes sociales.

2.4. Algoritmos para análisis de textos

La minería de textos es el análisis de los datos contenidos en el texto en lenguaje natural, la aplicación de técnicas de minería de texto para resolver problemas de negocios se conoce como análisis de texto.

La minería de textos puede ayudar a una organización a derivar potencialmente valiosas ideas empresariales de contenido basado en texto, como documentos de texto, correo electrónico y mensajes de flujos de medios sociales como Facebook, Twitter y LinkedIn. La minería de datos no estructurados con el procesamiento del lenguaje natural (NLP), modelos estadísticos y técnicas de aprendizaje puede ser un reto, ya que el texto en lenguaje natural es a menudo incoherente. El lenguaje natural contiene ambigüedades provocadas por la sintaxis y la semántica inconsistentes, incluyendo argot, lenguaje específico para industrias verticales y grupos de edad, dobles sentidos y el sarcasmo.

El software de análisis de texto puede ayudar mediante la transposición de palabras y frases en los datos no estructurados en los valores numéricos que se

pueden vincular con la información estructurada en una base de datos y se analizan con las técnicas tradicionales de minería de datos. Con un enfoque iterativo, una organización puede utilizar con éxito el análisis de texto para profundizar en los valores de contenido específico, como el sentimiento, la emoción, la intensidad y relevancia. A pesar de que la tecnología de análisis de texto sigue siendo considerada como una tecnología emergente, los resultados y la profundidad de análisis pueden variar mucho de un proveedor a otro.

2.4.1. Análisis Sentimental

El auge de los medios sociales como blogs y redes sociales ha impulsado el interés en el análisis de los sentimientos. Con la proliferación de opiniones, valoraciones, recomendaciones y otras formas de expresión en la red, la opinión en línea se ha convertido en una especie de moneda virtual para empresas que buscan comercializar sus productos, identificar nuevas oportunidades y gestionar su reputación. A medida que las empresas buscan poder automatizar el proceso de los filtrados, la comprensión de las conversaciones, la identificación del contenido relevante y ejecutando de manera apropiada, muchos están mirando hacia el campo de análisis de los sentimientos.

Twitter es un servicio popular de microblog donde los usuarios crean mensajes de estatus llamados “tuits”. Esos tuits algunas veces expresan opinión sobre diferentes temas. Se propone un método para automáticamente extraer el sentimiento (positivo, negativo o neutro) de un tuit, esto será muy útil porque permitirá obtener retroalimentación sin necesidad de contar con una intervención de forma manual.

Marco teórico

La clasificación sentimiento es uno de los problemas más difíciles en el Procesamiento del Lenguaje Natural. Un clasificador sentimiento reconoce los patrones de uso de la palabra entre las diferentes clases y los intentos de poner texto sin título que en una de estas categorías de manera no supervisada. Por lo tanto, el intento es clasificar los documentos no por temas sino por el sentimiento general.

Los clientes pueden hacer uso del análisis sentimental para investigar sobre productos y servicios antes de hacer una compra. Los comerciantes pueden hacer uso de esto para investigar sobre la opinión pública sobre su compañía y productos o analizar la satisfacción de su cliente.

A diferencia del uso del raiting con el análisis sentimental podremos conocer no solo el impacto que tiene nuestra marca, sino también conocer si el impacto fue positivo o negativo para nosotros.

2.4.1.1. Análisis Sentimental en Social Analysis

El sistema Social Analysis utiliza una técnica llamada “Machine Learning”, este tipo de técnicas se caracterizan por trabajar mediante un extenso conjunto de textos –que conocemos como corpus – y que actúan como sistema de entrenamiento. Esto significa que, a partir de estos textos, el algoritmo es capaz de “aprender” a diferenciar el significado o polaridad de las opiniones y comentarios.

En el caso del “sentiment analysis”, los corpus de entrenamiento son previamente clasificados y ordenados entre positivos y negativos. El algoritmo los

procesa y extrae patrones, probabilidades y estructuras del texto, que posteriormente utiliza para realizar el análisis sentimental de los textos.

El algoritmo Hace una clasificación binaria donde se clasifican las opiniones en 2 categorías: positivas y negativas.

El algoritmo que se utiliza va un paso más allá, ya que realiza dos procesos: en un primer paso extrae las partes subjetivas y objetivas de forma separada.

Sobre las determinadas como subjetivas que se realiza el análisis para averiguar si su signo es positivo o negativo. Este sistema permite obtener mejores resultados facilitando, además, obtener de forma casi instantánea la categoría implícita de resultados neutrales (aquellos que no contienen opinión).

2.4.2. Retos y dificultades técnicas

El nivel de precisión y fiabilidad alcanzado por las técnicas de análisis semántico se sitúa en la actualidad, en torno al 80%.

¿Dónde se hallan las principales dificultades desde el punto de vista técnico?

El lenguaje textual no es matemático. Los significantes no siempre corresponden a los significados. Podemos encontrar textos claramente negativos sin la utilización de una sola palabra negativa, y viceversa.

El lenguaje es maleable. Puede ser ambiguo, confuso, impreciso. La utilización de metáforas y expresiones de ironía son difíciles de procesar. De una secuencia de palabras no se desprende un resultado unívoco. De ahí su riqueza.

2.1. Java Enterprise Edition

Los desarrolladores de hoy en día reconocen cada vez más la necesidad de crear aplicaciones distribuidas, transaccionales, portátiles, seguras y sobre todo

rápidas. En el mundo de las tecnologías de la información las aplicaciones empresariales deben ser diseñadas, construidas y desplegadas con menor presupuesto, mayor rapidez y menos recursos.

La plataforma Java EE es desarrollada a través del JCP que es la responsable de todas las tecnologías java y su objetivo es proporcionarnos un conjunto de API's que nos permitan realizar aplicaciones con excelente calidad, poca complejidad, rápido desarrollo y mejora en el rendimiento de las aplicaciones.

2.1.1. Aplicaciones distribuidas multicapa

La plataforma Java EE utiliza un modelo de aplicación multicapa para aplicaciones empresariales. El flujo de la aplicación se divide en componentes de acuerdo a su función, y los componentes que conforman una aplicación JEE se instalan en diversos servidores, dependiendo de la capa en la cual trabaja. Las capas que conforman una aplicación web son las siguientes

- Capa del cliente: Componentes que se ejecutan en la máquina del cliente y permiten al usuario interactuar con el servidor, esto puede ser un navegador web, un dispositivo móvil u otra aplicación capaz de utilizar el protocolos como HTTP o IIOP.
- Capa web: Componentes que se ejecutan sobre un servidor Java EE y son encargados de presentar la información generada de forma dinámica.
- Capa de negocio: Componentes que se ejecutan sobre un servidor Java EE y son encargados de realizar la lógica de negocio de la aplicación.
- Capa de Acceso a datos: Componentes encargados de interactuar con las bases de datos o los dispositivos de almacenamiento y de obtención de información.

Marco teórico

El uso de una arquitectura multicapa es una excelente práctica en el desarrollo de software ya que el desarrollo se puede llevar a cabo en varios niveles, en caso de cambios, solo se afecta al nivel requerido, sin tener que revisar todo el código además que reduce el acoplamiento de componentes e incrementa la cohesión de los mismos.

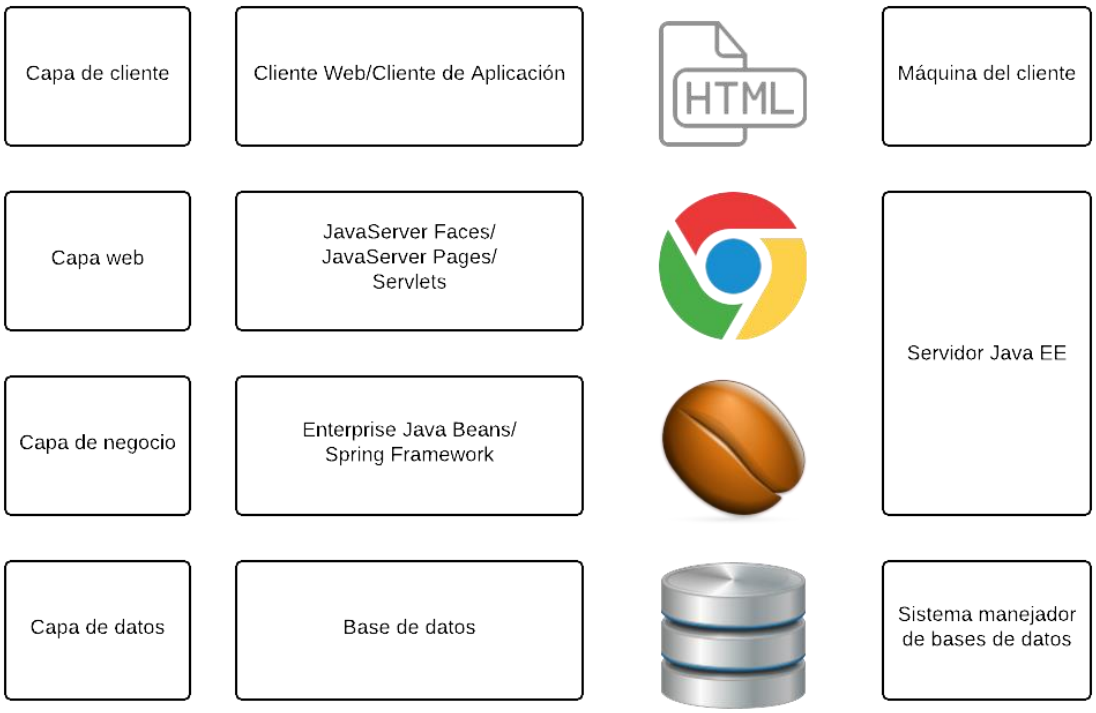


Figura 4: Diseño de capas en Java Enterprise Edition

2.1.2. Servlets

Un Servlet es un componente web cuyo objetivo principal es tomar una petición de un cliente generar y devolver una respuesta. Los Servlets son capaces de tomar una petición y devolver una respuesta simple como una página en formato HTML o compleja como una aplicación completa de compras por internet.

Los Servlets entran en el grupo llamado “Web Components” o componentes web, dichos componentes son administrados por un *Contenedor Web*. Dicho contenedor es el encargado de llevar a cabo el ciclo de vida de los Servlets.

Los Servlets son la base de muchas tecnologías web y *Frameworks* que funcionan sobre el lenguaje Java, ya que permiten realizar contenido dinámico en un servidor y devolverlo como respuesta a un cliente. Con esto el cliente no necesita tener instalada la JVM (Java Virtual Machine) ni ningún otro tipo de complemento.

2.1.3. JSP

JSP (JavaServer Pages) proporcionan una forma simple y rápida de crear contenido web dinámico. Esta tecnología permite el desarrollo rápido de aplicaciones web que funcionan en el servidor de plataforma independiente.

Esta tecnología está basada en Servlets y permite una forma mucho más simple de generar contenido web dinámico, un JSP es traducido a un Servlet y es administrado de la misma forma.

Las JSP’s permiten la utilización de código java y de expression language a través de scripts que son insertados en un documento HTML. A demás es posible utilizar bibliotecas de etiquetas que permiten el uso de etiquetas que mejoran la legibilidad del código y facilitan el desarrollo de software.

2.1.4. JavaServer Faces

JSF (JavaServer Faces) desarrollado por la Java Community Process bajo la JSR-314, es el estándar para el desarrollo de interfaces de usuario del lado del servidor. JSF es un framework web basado en componentes. Esto significa que el

desarrollador no tiene que generar código HTML para tareas rutinarias tales como la creación de una tabla, un calendario, mostrar una imagen, utilizar un link, etc. En lugar de eso utilizas un componente JSF dedicado a eso, JSF está compuesto por las siguientes partes:

- Un conjunto prefabricado de componentes UI (User Interface).
- Un modelo de programación dirigido por eventos.
- Un modelo que permite que otros usuarios generen nuevos componentes.
- Un conjunto de reglas de navegación
- Un soporte de internacionalización

Algunos componentes son muy sencillos, imágenes, checkbox, áreas de texto, etc. Otros son muy sofisticados como tablas de datos, paginadores o árboles.

JSF no es el único framework basado en componentes, pero es el estándar de Java EE, esto permite que este incluido en todos los servidores de aplicaciones Java EE y que existan tecnologías dedicadas a la expansión de sus componentes, algunas tecnologías dedicadas a esto son: PrimeFaces, RichFaces, OpenFaces y ICEFaces.

Estas tecnologías están dedicadas a generar componentes complejos JSF entre los que se pueden encontrar tablas de datos con paginadores, galerías de imágenes, cuadros de diálogos, menús y componentes que permiten una manera más sencilla de trabajar con tecnologías como *AJAX*.

2.5.6. PrimeFaces

JSF está basado en componentes, los cuales utilizamos para realizar de manera sencilla tareas que son repetitivas y complejas. PrimeFaces es un framework de código abierto dedicado a la expansión de componentes de JSF para realizar tareas complicadas de una forma sencilla y óptima. Además de que cuenta con la posibilidad de integrar sus componentes con otros por ejemplo los creados por RichFaces.

Dentro de PrimeFaces existen componentes para diversas tareas, entre los que se encuentran:

- Áreas de texto con posibilidad de autocompletar
- Tablas complejas de datos
- Paginadores
- Carruseles de imágenes
- Gráficas
- Paneles
- Diálogos
- Manejadores de archivos
- Componentes que facilitan el uso de AJAX dentro de la aplicación
- Cuenta con 25 temas prediseñados

En cuanto a la experiencia con usuarios finales estos componentes son amigables y cuentan con un diseño innovador, en cuanto a los programadores se cuenta con herramientas que facilitan el uso de tecnologías como AJAX y generan componentes nuevos constantemente.

2.5.7. Java Persistence Api

Las aplicaciones empresariales son caracterizadas por su necesidad de recolectar, transformar, borrar y reportar una gran cantidad de información. JPA

proporciona un modelo de persistencia basado en *POJO's.* JPA fue desarrollado por el grupo de expertos de EJB 3.0 como parte de *JSR* 220, aunque su uso no se limita a los componentes software EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones clientes; incluso fuera de la plataforma Java EE, por ejemplo, en aplicaciones Java SE.

En su definición, se han combinado ideas y conceptos de los principales frameworks de persistencia como Hibernate, Toplink y JDO, y de las versiones anteriores de EJB. Todos éstos cuentan actualmente con una implementación JPA.

2.5.7.1. Object-Relational Mapping

Dentro del dominio de java se tienen clases, dentro de la base de datos se tienen tablas. Estos dos dominios son muy similares y debería de haber una forma de realizar el salto entre estos dos de manera sencilla. Esto ya se ha realizado manualmente utilizando patrones de diseño como DAO (Data Access Object) utilizando JDBC (Java Database Connectivity), el problema con esto es que la transformación entre un modelo y otro es manual y repetitiva.

La técnica para brincar entre un modelo y otro de manera transparente es conocida como Object-Relational Mapping. Y se basa en realizar clases llamadas entidades que representen las tablas y relaciones que se encuentran dentro de la base de datos, esto se logra a través de simples anotaciones que representan relaciones.

2.5.7.2. Entidades

El concepto de entidad es simple y es una clase que representa a una tabla dentro de la base de datos, si se crea una entidad se tiene la posibilidad de persistirla, leerla, actualizarla o borrarla dentro de la base de datos. Son las entidades las que representan el ORM dentro de la aplicación, ya que son las que permiten el salto de un modelo al otro.

Las instancias de una entidad corresponden a una fila dentro de la base de datos. Un metadato dentro de una entidad se puede realizar a través de anotaciones.

2.5.7.3. Modelo de dominio

Un modelo de dominio es un modelo conceptual de todos los temas relacionados con un problema específico. En el se describen las distintas entidades, sus atributos, papeles y relaciones además de las restricciones que rigen el dominio del problema. El modelo de dominio se crea con el objetivo de representar el vocabulario y los conceptos claves del dominio del problema. El modelo de dominio también representa las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema.

Una ventaja importante de un modelo de dominio es que describe y limita el alcance del dominio del problema. El modelo de dominio puede ser usado efectivamente para verificar y validar la comprensión del dominio del problema entre las diversas partes interesadas. Define un vocabulario y es útil como herramienta de comunicación. Puede añadir precisión y enfoque para la discusión entre el equipo de negocios, así como entre los equipos técnicos y de negocios.

2.5.7.4. Relaciones múltiples de la entidad

Existen cuatro tipo de relaciones: uno a uno, uno a muchos, muchos a uno, y muchos a muchos.

Uno a uno: Cada entidad se relaciona con una sola instancia de otra entidad. Las relaciones uno a uno utilizan anotaciones de la persistencia de java "`OneToOne`".

Uno a muchos: Una entidad, puede estar relacionada con varias instancias de otras entidades. Las relaciones uno a muchos utilizan anotaciones de la persistencia de java "`OneToMany`" en los campos o propiedades persistentes.

Muchos a uno: Múltiples instancias de una entidad pueden estar relacionadas con una sola instancia de otra entidad. Esta multiplicidad es lo contrario a la relación uno a muchos. Las relaciones muchos a uno utilizan anotaciones de la persistencia de java "`ManyToOne`" en los campos o propiedades persistentes.

Muchos a muchos: En este caso varias instancias de una entidad pueden relacionarse con múltiples instancias de otras entidades.Este tipo de relación utiliza anotaciones de la persistencia de java "`ManyToMany`" en los campos o propiedades persistentes.

Estas entidades pueden ser utilizadas para representar el modelo de dominio de la aplicación, esto es un modelo conceptual en el cual se definen todos los temas relacionados con un problema específico, el modelo de dominio muestra (a los modeladores) clases conceptuales significativas en un dominio de problema. Es el artefacto más importante que se crea en el análisis orientado a objetos.

2.5.7.5. JPQL

JPQL (Java Persistence Query Language) Es un poderoso framework que permite realizar consultas a la base de datos utilizando el dominio de Java y obteniendo resultados en el mismo. Este lenguaje está basado en SQL pero tiene la capacidad de devolver los resultados en listas de Java que pueden ser utilizadas en cualquier parte de la aplicación sin necesidad de ningún tipo de transformación manual.

Dentro de JPQL se pueden realizar consultas como en SQL, teniendo la posibilidad de utilizar en las consultas funciones, subqueries, joins, restricciones, ordenamientos, agrupaciones, entre otras.

2.5.8. Enterprise JavaBeans

Un EJB (Enterprise Java Beans) es un componente que funciona del lado del servidor, es el encargado de ejecutar la lógica de negocio de una aplicación. La lógica de negocio dentro de la programación es el código que cumple con el propósito de la aplicación, por ejemplo, en una aplicación de compras por internet la lógica de negocio sería la encargada de persistir el cobro dentro de la base de datos, consultar los catálogos nuevos existentes, realizar actualizaciones a los productos, entre otras acciones que son indispensables para la aplicación.

Los EJB son componentes que permiten simplificar el desarrollo de aplicaciones complejas y distribuidas. Es importante aclarar que un EJB no funciona en un contenedor web simple, ya que requiere que se administre su ciclo de vida de manera diferente, es por esto que requieren de un *Servidor de Aplicaciones* completo que contenga un contenedor de EJB, responsable de los

servicios a nivel de sistema, tales como la administración de transacciones o las tareas de seguridad dentro de la aplicación.

Existen dos tipos de componentes EJB dentro de JEE: session beans y message-driven beans. Los session beans y los message-driven beans son encargados de realizar la lógica de negocio dentro de la aplicación.

2.6. Spring

Es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. A pesar de que no impone ningún modelo de programación en particular, este framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).

2.6.1. Inversión de control

El contenedor de Spring es el núcleo de Spring Framework. El contenedor creará los objetos, conectarlos juntos, configurarlos, y gestionar su ciclo de vida completo, desde la creación hasta la destrucción. El contenedor de Spring utiliza inyección de dependencias (DI) para gestionar los componentes que forman una aplicación. Estos objetos se llaman Spring Beans.

El contenedor recibe las instrucciones sobre que objetos instanciar, configurar y montar mediante la lectura de metadatos de configuración proporcionados. Los metadatos de configuración se puede representar ya sea

XML, anotaciones Java, o código Java. El siguiente diagrama es una vista de alto nivel de cómo funciona Spring. El contenedor de Spring IoC hace uso de clases Java POJO y metadatos de configuración para producir un sistema o aplicación totalmente configurada y ejecutable.

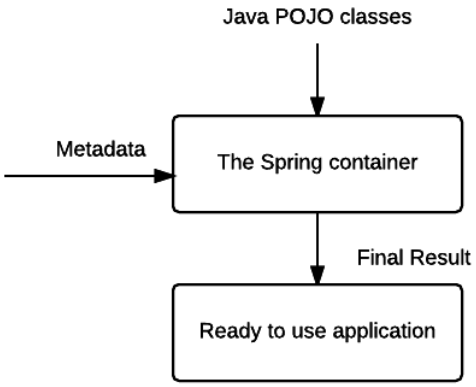


Figura 5: Inversion of Control

2.7. Servicios Web

Un servicio web es una aplicación la cual provee funcionalidad para un cliente. Los clientes son normalmente navegadores web pero pueden ser también aplicaciones estándar. La comunicación entre un cliente y un servicio web se realiza a través de protocolos de comunicación que conectan puntos finales. El término puntos finales representa un lugar específico que son usados para acceder al servicio web.

Desde la perspectiva de Java los servicios web pueden ser divididos en dos grandes partes, la primera categoría es Java API for XML Web Services (JAX-WS) la cual se basa en XML y en SOAP (Simple Object Access Protocol). Las capacidades de un servicio son publicadas en un documento XML basado en

WSDL (Web Service Descriptor Language). Éste soporta una arquitectura basada en mensajes y llamadas remotas. Este tipo de servicios son más complejos y robustos pero proveen a las aplicaciones de transacciones, seguridad y otras ventajas.

La segunda categoría es Representational State Transfer (RESTful) JAX-RS. Este tipo de servicio es útil para aplicaciones más simples que tienen una menor demanda. Este tipo de servicios son útiles cuando el rendimiento de la aplicación es un factor importante.

2.6.1. Servicios Web RESTful

Representational State Transfer (RESTful) es un estilo de arquitectura en el cual los web services son vistos como recursos y pueden ser identificados por un URI's (Uniform Resource Identifiers).

Un uso muy común para los RESTful web services es actuar como un intermediario a la base de datos, clientes RESTful pueden usar un servicio RESTful para realizar operaciones CRUD (Create, Read, Update y Delete) a la base de datos.

Este tipo de servicios son accesados a través de métodos del protocolo HTTP como GET, POST, PUT, TRANCE, DELETE, etc. Nosotros utilizamos anotaciones para identificar el método HTTP que va a atender un método Java.

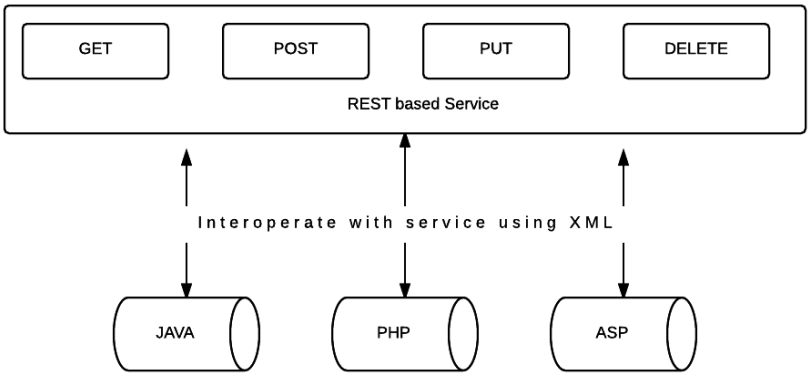


Figura 6: Interoperabilidad

2.6.2. Ventajas de utilizar web services

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades, plataforma en la que estén instaladas o el lenguaje de programación que se utilizó para desarrollarlas.
- Los servicios web fomentan los estándares y protocolos basados en textos, que hacen más simple acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

2.7. Ingeniería de software

Actualmente casi todos los países dependen de sistemas informáticos complejos. Infraestructuras nacionales y utilidades dependen de sistemas informáticos, y la mayor parte de los productos eléctricos incluyen una computadora y software de control. La fabricación industrial y distribución está

completamente automatizada, como el sistema financiero. Por lo tanto, producir software costeable es esencial para el funcionamiento de la economía nacional e internacional.

La ingeniería del software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software. Éste es abstracto e intangible. No está restringido por materiales, o gobernado por leyes físicas o por procesos de manufactura. De alguna forma, esto simplifica la ingeniería del software ya que no existen limitaciones físicas del potencial del software. Sin embargo, esta falta de restricciones naturales significa que el software puede llegar a ser extremadamente complejo y, por lo tanto, muy difícil de entender.

2.7.1. Sistemas críticos

Los fallos de funcionamiento del software son relativamente comunes. En la mayoría de los casos, estos fallos provocan molestias, pero no daños graves ni a largo plazo. Sin embargo, en algunos sistemas un fallo de funcionamiento puede ocasionar pérdidas económicas significativas, daño físico o amenazas a la vida humana. Estos sistemas se conocen como sistemas críticos.

Los sistemas críticos son sistemas técnicos o socio-técnicos de los cuales dependen las personas o los negocios. Si estos sistemas no ofrecen sus servicios de la forma esperada, pueden provocar graves problemas y pérdidas importantes.

Hay tres tipos principales de sistemas críticos:

- Sistemas de seguridad críticos. Son sistemas cuyo fallo de funcionamiento puede provocar perjuicio, pérdida de vidas o daños graves al medio ambiente. Un ejemplo de un sistema de seguridad crítico es un sistema de control para una planta de fabricación de productos químicos.

- Sistemas de misión críticos. Son sistemas cuyo fallo de funcionamiento puede provocar errores en algunas actividades dirigidas por objetivos. Un ejemplo de un sistema de misión crítico es un sistema de navegación para una nave espacial.
- Sistemas de negocio críticos. Son sistemas cuyo fallo de funcionamiento puede provocar costos muy elevados para el negocio que utiliza un sistema de este tipo. Un ejemplo de un sistema de negocio crítico es un sistema de cuentas bancarias.

2.7.2. Procesos del software

Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software. Estas actividades pueden consistir en el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C. Sin embargo, cada vez más, se desarrolla nuevo software ampliando y modificando los sistemas existentes y configurando e integrando software comercial o componentes del sistema.

Aunque existen muchos procesos diferentes de software, algunas actividades fundamentales son comunes para todos ellos:

- Especificación del software. Se debe definir la funcionalidad del software y las restricciones en su operación.
- Diseño e implementación del software. Se debe producir software que cumpla su especificación.
- Validación del software. Se debe validar el software para asegurar que hace lo que el cliente desea.
- Evolución del software. El software debe evolucionar para cubrir las necesidades cambiantes del cliente.

2.7.3. Modelos del proceso del software

Un modelo del proceso del software es una representación abstracta de un proceso del software. Cada modelo representa un proceso desde una perspectiva particular, y así proporciona sólo información parcial sobre ese proceso.

Estos modelos generales no son descripciones definitivas de los procesos del software. Más bien, son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo de software. Puede pensarse en ellos como marcos de trabajo del proceso que pueden ser extendidos y adaptados para crear procesos más específicos de ingeniería del software.

Los modelos de software más comunes son los siguientes:

- El modelo en cascada. Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etcétera.
- Desarrollo evolutivo. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Éste se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.
- Ingeniería del software basada en componentes. Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.

Aunque no existe un proceso del software «ideal», en las organizaciones existen enfoques para mejorarlos. Los procesos pueden incluir técnicas

anticuadas o no aprovecharse de las mejores prácticas en la ingeniería del software industrial. De hecho, muchas organizaciones aún no aprovechan los métodos de la ingeniería del software en sus desarrollos.

2.8. Pruebas de software

Una prueba es el proceso que permite revelar la calidad de un producto, se realiza para demostrar que un software no tiene errores y además cumple con las funciones para las cuales fue construido; las pruebas son integradas dentro de diferentes fases del ciclo de software dentro de la Ingeniería de software.

Para determinar el nivel de calidad se deben ejecutar una serie de medidas o pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema.

2.8.1. Tipos de Pruebas

Dentro del desarrollo de software existen diversos tipos de pruebas, y aunque no existe una clasificación oficial o formal acerca de ellos, existen dos vertientes fundamentales:

- Pruebas de caja negra (Black Box testing): Cuando el sistema es probado utilizando su interfaz externa, para este tipo no se tiene acceso al código fuente de las aplicaciones y solo se trabaja con entradas y salidas del sistema.
- Pruebas de caja blanca (White Box testing): Cuando un sistema es probado desde dentro, utilizando su lógica aplicativa y se trabaja conociendo sus entradas, sus salidas y el conocimiento interno del sistema.

A demás de este tipo de pruebas existen otras clasificaciones de pruebas en las que se incluyen las siguientes:

Marco teórico

- Pruebas unitarias: Es la forma de probar el correcto funcionamiento de la unidad mínima funcional del código, el cual no interactúa con otros componentes. Esto nos permite saber si cada uno de los módulos puede funcionar de manera independiente.
- Pruebas funcionales: El servicio de pruebas funcionales se centra en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente. Este servicio ayuda a su organización a detectar los posibles defectos derivados de errores en la fase de programación.
    - Pruebas de integración: Son aquellas que se realizan en el desarrollo de software una vez que se han aprobado las pruebas unitarias. Este tipo de pruebas comprueban las conexiones y comunicaciones entre los diferentes módulos del software desarrollado o con terceros (pasarelas de pago, sistemas publicitarios).
    - Integración no incremental: Se combinan todos los módulos por anticipado y se prueba todo el sistema en conjunto.
    - Integración incremental: El programa se construye y se prueba en segmentos pequeños.
- Pruebas de aceptación: Son procesos de validación que permiten saber si el sistema de software producido cumple con las especificaciones y con su cometido.

Capítulo 3

3. Análisis y diseño

Análisis y diseño

3.1. ¿Qué es Social Analysis?

Social Analysis es un sistema web desarrollado en Java cuyo objetivo principal es la búsqueda, análisis, clasificación y representación de información proveniente de Twitter a través de mapas, gráficas, reportes y tablas de datos.

La pregunta natural es ¿por qué realizar análisis sobre Twitter?, la respuesta es simple Twitter es una red social emergente que se estima que cuenta con cerca de 200 millones de usuarios en todo el mundo y que genera cerca de 65 millones de tuits (publicaciones) cada día. Un tuit es un texto de no más de 140 caracteres en el cual se plasma una idea, comentario o queja sobre algún personaje, evento, lugar o acontecimiento.

3.2. Análisis de requerimientos

Se presentarán los dos tipos de requerimientos del sistema tanto los funcionales como los no funcionales, los requerimientos funcionales de un sistema describen la funcionalidad o los servicios que se espera que éste provea. Son entendidos como capacidades que debe exhibir una aplicación con el fin de resolver un problema. Los requerimientos no funcionales se refieren a las implicaciones que esto conlleva como por ejemplo seguridad, tiempo de respuesta, escalabilidad, entre otros.

3.2.1. Requerimientos Funcionales

El sistema Social Analysis es un sistema dedicado a la búsqueda, análisis, clasificación, generación de estadísticas y representación de la información en

gráficas y reportes de la información proveniente de Twitter. El sistema está compuesto por los siguientes módulos:

3.2.1.1. Módulo de mapas

Éste es uno de los módulos principales de la aplicación y su objetivo principal es que a través de un mapa se conozca la opinión de los usuarios de Twitter sobre algún tema, personaje, compañía o evento en diferentes partes del mundo sin importar el idioma en el cual este escrita la información.

En este módulo la aplicación llevará a cabo las siguientes tareas:

- Análisis del lenguaje: La herramienta detecta en que lenguaje se encuentran escritos los tuits los clasifica a través de un algoritmo llamado análisis sentimental y otro llamado análisis de impacto.
- Presentación de trending topics: Adicionalmente a los resultados se genera un apartado que nos permite conocer los trending topics de la región seleccionada, con esto podremos saber cuáles son los temas de interés de la región en un momento determinado.
- Geocodificación de tuits: Dentro de un mapa se crearán puntos que reflejan el lugar en el que fueron escritos los tuits, es importante mencionar que esto solo funcionará para las personas que tienen activa su localización, esto nos permite conocer en qué lugar tiene impacto el tema o marca que estamos buscando.
- Gráficas de Sentimiento e Idioma: Se presentan 2 gráficas de pie que representan una lo positivo, negativo y neutro que se encontró

respectivamente y la otra el idioma en el cual se encontraron los resultados de la búsqueda de la aplicación.

- Comparación de resultados: Una vez realizada la búsqueda, la aplicación será capaz de realizar otra en modo “comparar”, la cual nos permitirá comparar los resultados de la nueva búsqueda con la anterior para así poder realizar distintos tipos de análisis tales como:
- Comparar una marca o producto en diferentes países y conocer en cuales se tiene un mayor o menor impacto
- Comparar una marca o producto con su competencia en diferentes regiones para conocer en qué lugares se tiene mayor impacto sobre la competencia y viceversa.

3.2.1.2. Módulo de búsqueda

El módulo de búsqueda permite realizar búsquedas a nivel global y obtener resultados sin importar el lugar en el que se encuentre. Su función principal es realizar tanto análisis sentimental como análisis de lenguaje para conocer en un contexto general como se está comportando un tema, empresa, personaje o marca desde una perspectiva general. El módulo de búsqueda realiza las siguientes tareas:

Realiza una búsqueda general de los tuits referentes a un término de búsqueda, éste puede ser un tema o marca de interés para el usuario que la esté realizando.

Clasificación de tuits: El sistema clasificará la información devuelta de acuerdo al sentimiento y al lenguaje, marcando rojos los tuits negativos, grises los

neutros y verdes los positivos. Los tuits devueltos serán presentados en una tabla de información, se debe permitir paginación, filtrado y reordenamientos de los tuits devueltos.

Generación de gráficas: Adicionalmente a la tabla de datos con los resultados aparecerán 2 gráficas de pie una representando el sentimiento (Positivo, Negativo o Neutro) y otra con el idioma (Español o inglés).

3.2.1.3. Módulo de perfiles

El módulo de perfiles permite ver los tuits de la cuenta con la que se inició sesión del mismo modo que se hace en la página de Twitter, adicionalmente se pueden ver los retuits y los favoritos de la cuenta, todo esto en la misma pantalla con la posibilidad de ordenarlos de acuerdo al número de retuits para así conocer cuáles fueron las publicaciones que tuvieron un mayor impacto dentro de tu cuenta. Adicionalmente se pueden crear tuits y ver tanto la información de la cuenta como el número de seguidores, amigos y número de tuits hechos desde la cuenta.

Se mostrará el nombre, el nombre de usuario, la fecha del último tuit además de que se debe tener la posibilidad de realizar filtros, ordenamientos y paginación sobre las tablas de datos que contienen los tuits.

3.2.1.4. Módulo Time line

Este módulo permite ver los tuits de las personas a las que sigue la cuenta con la que se inició sesión del mismo modo en el que se hace en la página de

Twitter, además de que se tiene la posibilidad de realizar tuits, ver los seguidores y los amigos de la cuenta.

Se mostrará el nombre, el nombre de usuario, la fecha del último tuit además de que se debe tener la posibilidad de realizar filtros, ordenamientos y paginación sobre las tablas de datos que contienen los tuits.

3.2.2. Requerimientos no funcionales

Los requerimientos no funcionales hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo. Estos requerimientos son adicionales a los requerimientos funcionales que debe cumplir el sistema, y corresponden a aspectos tales como la disponibilidad, mantenibilidad, flexibilidad, seguridad, facilidad de uso, etc.

3.2.2.1. Atributos de Calidad del Sistema

Desempeño:

El sistema deberá soportar 100 usuarios conectados de modo concurrente brindando una respuesta de máximo 2 segundos por cada petición realizada.

Disponibilidad:

Estar disponible 100% o muy cercano a esta disponibilidad durante el horario hábil laboral.

Escalabilidad:

El sistema debe ser construido sobre la base de un desarrollo evolutivo e incremental, de manera tal que nuevas funcionalidades y requerimientos relacionados puedan ser incorporados afectando el código existente de la menor

manera posible; para ello deben incorporarse aspectos de reutilización de componentes.

Facilidad de Uso e Ingreso de Información:

El sistema debe ser de fácil uso y entrenamiento por parte de los usuarios, así como de fácil adaptación de la entidad con el mismo.

El sistema no debe permitir el cierre de una operación hasta que todos sus procesos, subprocesos y tareas relacionados, hayan sido terminados y cerrados satisfactoriamente.

Flexibilidad:

El sistema debe ser diseñado y construido con los mayores niveles de flexibilidad en cuanto a la parametrización de los tipos de datos, de tal manera que la administración del sistema sea realizada por un administrador funcional del sistema

Seguridad:

El acceso al sistema debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Sólo podrán ingresar al sistema las personas que estén registradas, estos usuarios serán clasificados en varios tipos (roles) con acceso a las opciones de trabajo definidas para cada rol.

Por seguridad de los usuarios de Social analysis el sistema realizará la autenticación utilizando el protocolo OAuth (Open Autorization) el cual proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta. En otras palabras, OAuth permite a un usuario del sitio A compartir su información en el sitio A (proveedor de servicio) con el sitio B (llamado consumidor) sin compartir toda su identidad. De este modo Social

analysis no almacena información sensible de los usuarios como el password de su cuenta.

3.3. Arquitectura del sistema

La arquitectura del sistema constituye un modelo a través del cual será desarrollado el sistema, la plataforma elegida para el desarrollo de la aplicación es la plataforma JEE (Java Enterprise Edition) la cual fue diseñada para el desarrollo de aplicaciones distribuidas, con base en componentes o unidades funcionales de software que interactúan entre sí para formar parte de una aplicación empresarial. Los frameworks utilizados para el desarrollo de la aplicación fueron los siguientes:

JavaServer Faces : Se tomó la decisión de utilizar JavaServer Faces como framework para la capa web debido a diversas ventajas que brinda en la fase de desarrollo de software dentro de las cuales destacan: Componentes que facilitan el desarrollo, manejo de templates, manejo de navegación, AJAX, internacionalización, el uso de componentes personalizados, entre otras.

PrimeFaces: Se eligió PrimeFaces como complemento a JavaServer Faces debido a la gran diversidad de componentes que lo integran, así como el excelente manejo de AJAX, la capacidad de cambiar de temas dependiendo del diseño de la aplicación, así como la perfecta integración que tiene con JavaServer Faces.

Spring Framework: Framework utilizado para el desarrollo de aplicaciones empresariales que ayuda a realizar el desarrollo JEE de una manera más sencilla. A diferencia de otros frameworks, Spring Framework no trabaja sólo en una capa de la aplicación sino que tiene la capacidad de trabajar en todas las capas arquitectónicas ya que está compuesto de diversos módulos especializados en

diferentes aspectos del desarrollo. Los módulos utilizados dentro de la aplicación fueron los siguientes:

Spring Security: Este módulo fue utilizado para manejar la seguridad dentro de la aplicación. Con esto se lleva a cabo la autenticación, cifrado, autorización y la restricción de contenidos en base a roles.

IOC(Inversion of Control): Spring Framework está compuesto por un contenedor, el cual es el encargado de administrar los objetos java dentro de la aplicación, así como el tiempo de vida que tendrán dichos objetos, el uso Inversion of Control ayuda a tener un sistema con bajo acoplamiento y alta cohesión y por tanto un mejor diseño.

Spring Data: El acceso a datos es un problema común dentro de las aplicaciones empresariales, el API estándar para realizar el acceso a datos es JPA(Java Persistence Api) el cual brinda diversas funcionalidades que optimizan y facilitan el acceso a datos, Spring Data fue utilizado dentro del sistema como un complemento a JPA, el cual permite quitar la capa DAO de una aplicación y delegar la implementación de los métodos de acceso a datos al framework, con esto se consigue un código más limpio y un mejor diseño de la aplicación.

Rest Template: En el desarrollo de aplicaciones de hoy en día es muy común el uso de webservices para compartir información y comunicar aplicaciones empresariales. El propósito de la aplicación es realizar monitoreo y análisis de Twitter. Twitter provee un api basado en webservices rest para realizar búsquedas, realizar tuits, seguir a usuarios, entre otras acciones dentro de la aplicación. Rest Template es una implementación de spring que permite consumir

esos servicios de una manera simple y eficiente, además del api de Twitter se utiliza Rest Template para el consumo de un servicio encargado de realizar el análisis sentimental dentro de la aplicación del cual hablaremos más adelante.

Java Persistence API: JPA será utilizado como API para acceso a datos ya que es el estándar de Java EE para esta tarea, con JPA se realiza el ORM(Object Relational Mapping) de la aplicación, las consultas JPQL (Java Persistence Query Language), las entidades de la aplicación y en lugar de realizar DAO's(Data Access Object) JPA para el acceso a datos se comunicará con Spring Data para realizar la tarea de los DAO's de una manera más sencilla y eficiente.

Twitter4j: Es un API que simplifica de manera considerable el uso del api de Twitter, ya que brinda un modelo en java de las respuestas que brinda el API Rest de Twitter, con esto no tenemos que crear un modelo especial para las respuestas de las peticiones de Twitter, además de ofrecer una forma sencilla de realizar las llamadas a Twitter para acciones como crear un tuit, seguir a un usuario, dejar de seguirlo, generar listas, entre otras acciones.

API de Google Maps: Los mapas de google son los mapas más conocidos por los usuarios y dentro de la aplicación se tiene la posibilidad de realizar búsquedas a través de mapas y plasmar en estos el lugar del que proviene un tuit, además que nos ayudan a conocer el estatus de una búsqueda en diferentes partes del mundo, se utilizará el API de google maps para poder representar dichos mapas dentro de la aplicación.

JUnit: Las pruebas unitarias son necesarias dentro de la aplicación debido a la necesidad de probar los componentes de manera aislada, el api más poderoso

para realizar esta tarea es JUnit y tiene una perfecta integración con el contexto de Spring framework y fue por esto que se tomó la decisión de utilizarlo.

Maven: Se hará uso de Maven para construir el proyecto debido al número de frameworks que se tienen dentro de la aplicación y las dependencias que requiere cada uno de éstos. Además del uso de plugins para facilitar el desarrollo como el plugin de Jetty que permite realizar “Hot deploys” lo cual ayudará a realizar el proyecto de forma más rápida.

Apache Tomcat: Debido a que se está realizando una aplicación empresarial, ésta no puede ser instalada como una aplicación de escritorio común, para poder instalar dicha aplicación es necesario un contenedor web o un servidor de aplicaciones, debido a que no se está haciendo uso de EJB’s dentro de la aplicación se tomó la decisión de no usar un servidor de aplicaciones completo y usar simplemente un contenedor web, por esto de seleccionó Apache Tomcat para realizar esta tarea. Tomcat administra el pool de conexiones a la base de datos, la definición de roles y el despliegue de la aplicación en producción.

Base de datos: La base de datos de la aplicación hará uso del manejador de bases de datos Mysql ya que es un producto libre rápido y eficiente para la administración de la base de datos además de que es muy común su uso en ambientes de Cloud Computing, PAAS.

Netbeans: Netbeans es el IDE(Integrated Development Environment) recomendado por Oracle para el desarrollo de aplicaciones Java EE, en su versión 7.3 publicó mejoras en el debug de javascript, herramientas para probar javascript dentro del IDE, así como una integración directa con google chrome, además de

las ventajas que ya presentaba en sus versiones anteriores como su perfecta integración con Maven, Git, Apache Tomcat, Mysql, sus herramientas para medir el rendimiento de la aplicación y las facilidades que da al momento de realizar el desarrollo de software.

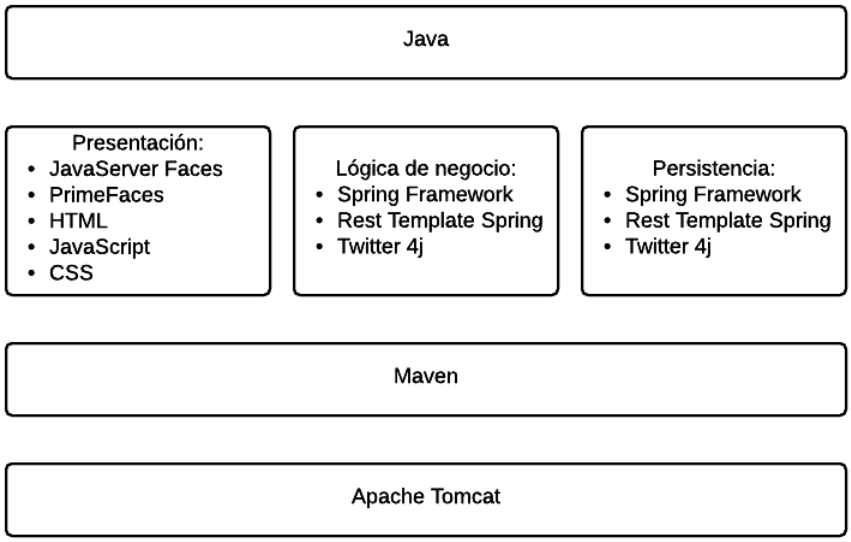


Figura 7: Arquitectura del sistema

3.4. Diseño de la base de datos

A continuación se presenta el diseño de la base de datos del sistema social análisis y una breve explicación de lo que se piensa almacenar en cada una de las tablas.

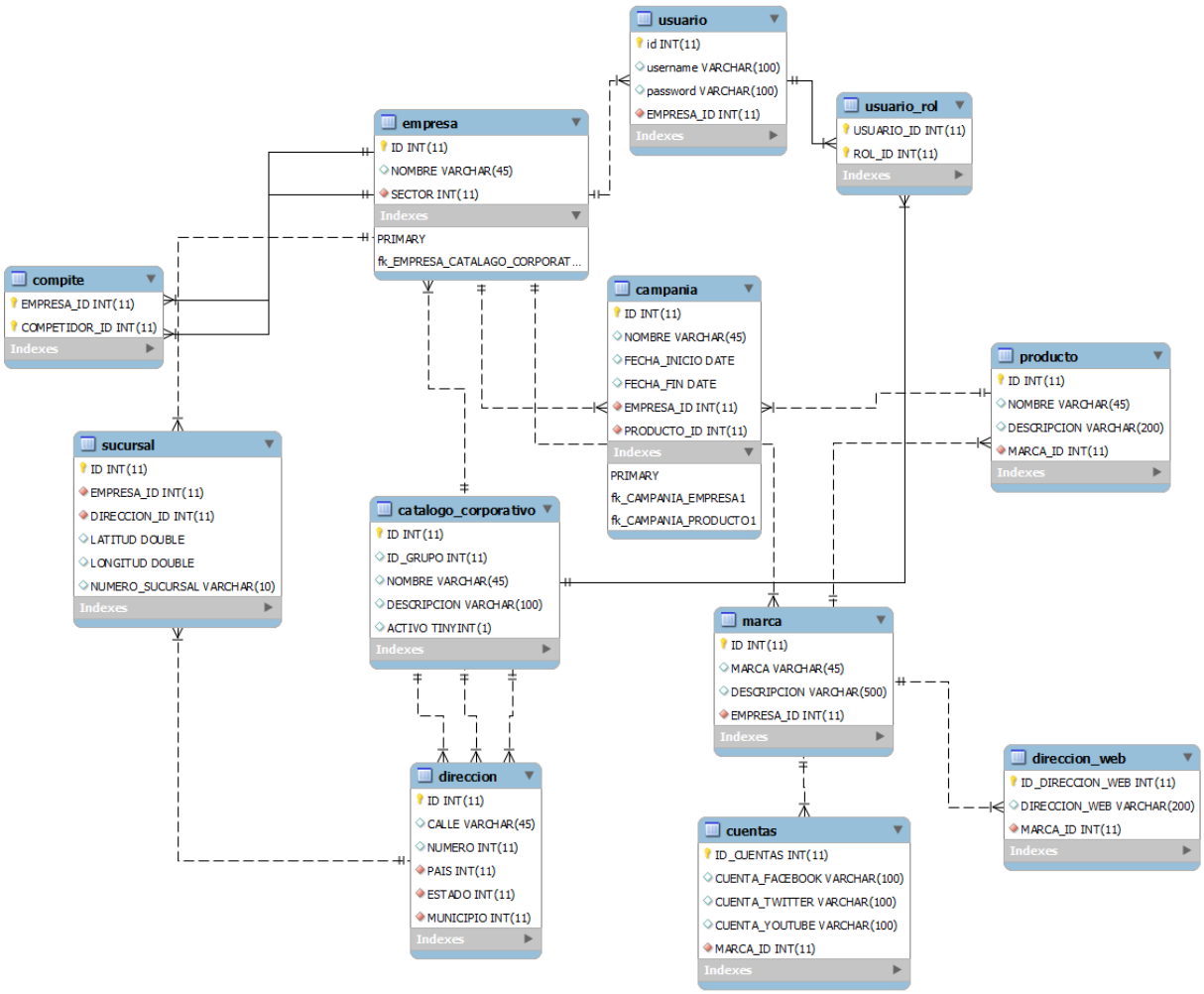


Figura 8: Diagrama de la base de datos del sistema

3.4.1 Descripción de las tablas

La base de datos del sistema Social Analysis esta compuesta por las tablas mostradas en el diagrama, a continuación se presenta una breve explicación de cada una de ellas:

- CATALOGO_CORPORATIVO: Esta tabla será la encargada de almacenar todos los catálogos utilizados dentro de la aplicación para reducir el número de tablas dedicadas a este propósito.

- EMPRESA: Será la responsable de almacenar el nombre de las empresas, así como el sector al que pertenece cada una de ellas, el sector se encuentra relacionado con la tabla CATALOGO_CORPORATIVO ya que dentro de esa tabla se encontrará el catálogo de sectores.
- COMPITE: Esta tabla nace de una relación recursiva de la tabla EMPRESA, esto es porque una empresa puede competir con muchas empresas.
- USUARIO: Cada usuario cuenta con un usuario y una contraseña, este está asociado con EMPRESA de tal modo que una empresa puede tener asignados muchos usuarios para entrar a la aplicación.
- USUARIO_ROL: Cada usuario tiene asociado un rol, el catálogo de roles se encuentra en la tabla CATALOGO_CORPORATIVO, es por esto que se tiene una relación a esa tabla.
- MARCA: Existen empresas que tienen muchas marcas asociadas a dicha empresa, por ejemplo la empresa “Procter & Gamble” tiene asociadas marcas como: Ace, Ariel, Vick, Pampers, etc. En la tabla marca se guarda la información de dichas marcas.
- CUENTA: Guarda información de las cuentas de redes sociales de la marca.
- DIRECCION_WEB: Guarda información de las direcciones web que tenga asociada la marca, son muchas por si existen direcciones específicas para cada país.

- CAMPANIA: Cada empresa puede dar de alta muchas campañas, para ser observadas en la aplicación, por cada campaña se tendrá asociado un producto sobre el cual se está haciendo la campaña por esto se tiene una relación a la tabla PRODUCTO.
- PRODUCTO: Guarda información sobre el nombre del producto que se quiere analizar así como la marca a la que pertenece.
- SUCURSAL: Almacena información sobre una sucursal en específico y la empresa a la cual está asociada.
- DIRECCIÓN: Almacena direcciones, cuenta con relaciones hacia la tabla CATALOGO_CORPORATIVO para valores como país, municipio y estado.

3.5. Diseño detallado

El sistema Social Analysis deberá ser un sistema web al que se pueda acceder a través de un navegador web y estará compuesto por los siguientes módulos: Mapas, búsqueda, perfiles y time line. Todos los módulos del sistema deberán de contar con un menú en la parte superior de la página que permitirán cambiar de un módulo a otro, así como el logo del sistema en la parte superior.

3.5.1. Módulo de mapas

El módulo de mapas representa al módulo principal del sistema y deberá contar sólo con un mapa en la parte central de la pantalla inicialmente. En caso de hacer click en algún lugar del mapa se deberá presentar un cuadro de diálogo que contenga los campos, “Término a buscar” y “Radio (Km)”, así como un botón de

"Buscar". Si se escribe algún "Término de búsqueda", un "Radio (km)" y oprime el botón de "Buscar" se llevarán a cabo las siguientes acciones:

Figura 9: Módulo mapas, opción buscar

Se actualiza el mapa y se pintan puntos que representan el origen de los tuits de los usuarios que cuenten con la función de localización activa.

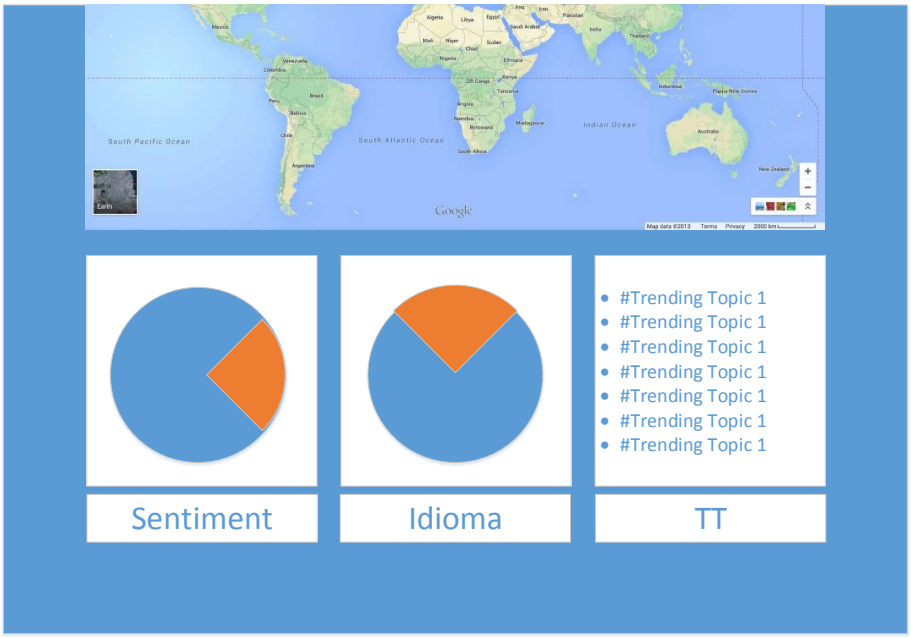


Figura 10: Módulo mapas, opción gráficas

- Se muestra una gráfica de pie que representa el sentimiento (Positivo, Negativo o Neutro).
- Se muestra una gráfica de pie que representa el idioma en el cual se encuentra la información, en este momento se tiene soporte solo para español e inglés.
- Se muestran los trending topics de la región a la cual se le dio click en el mapa.

- Se presenta una gráfica de barras con las siguientes series:
  - Tuits: Número de tuits encontrados.
  - Usuarios (x10,000): La suma de seguidores de todos los usuarios que realizaron tuits sobre el término de búsqueda (este número debe ser multiplicado por 10000 para obtener el número total ya que el número normalmente es muy grande con respecto a las demás series y se deben mostrar dentro de una misma gráfica).
  - Retuits (x1000): El número de retuits que se tiene de los tuits que cumplen con el término de búsqueda (este número debe ser multiplicado por 1000 para obtener el número total, ya que este número es muy grande con respecto a las demás series y se deben mostrar en una misma gráfica).
  - Positivos: El número total de tuits con contenido positivo que cumplen con el término de búsqueda.
  - Negativos: El número total de tuits con contenido negativo que cumplen con el término de búsqueda.

- Neutros: El número total de tuits con contenido neutro que cumplen con el término de búsqueda.

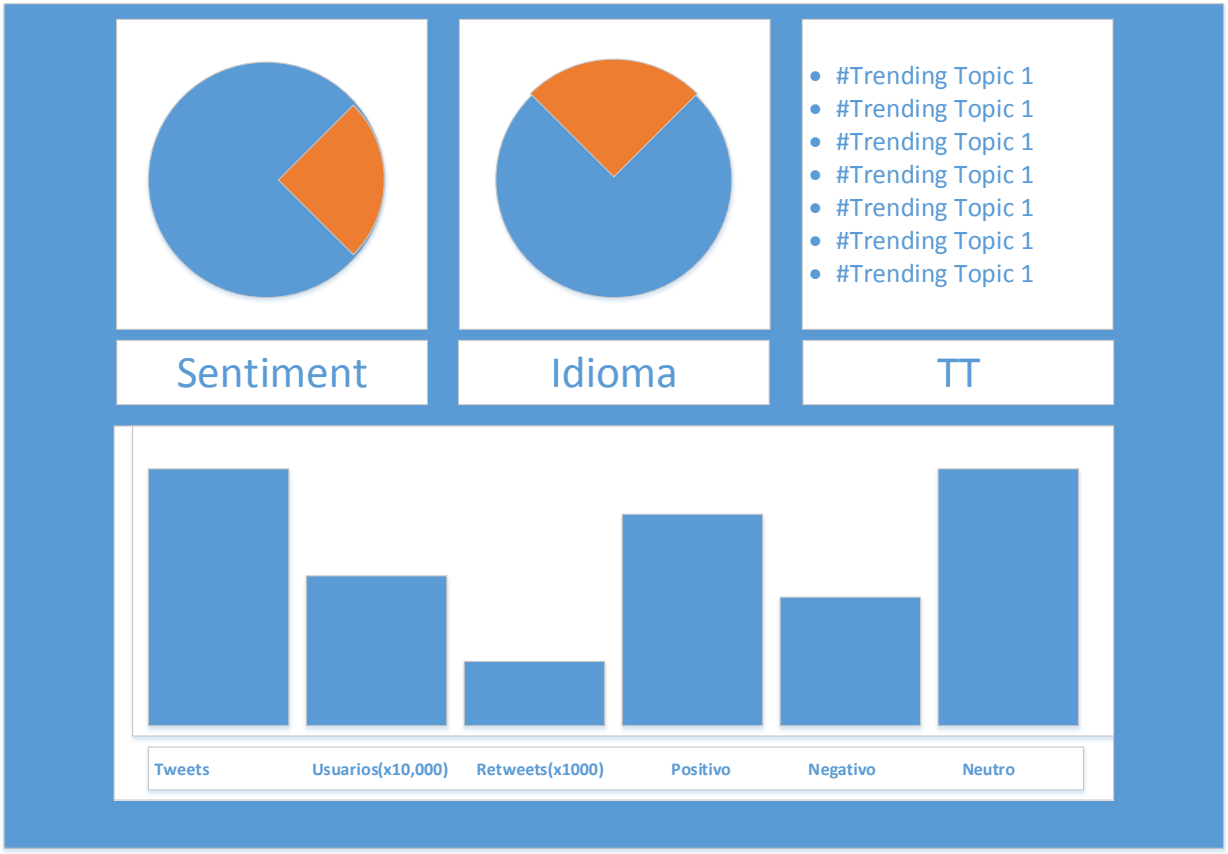


Figura 11: Módulo de mapas, opción gráficas pt 2

- Se presenta una tabla de datos con los tuits que cumplen con los términos de búsqueda realizados, esta tabla de datos debe cumplir con los siguientes requisitos;
  - La tabla de datos se debe poder ordenar de acuerdo al número de retuits que se tienen.
  - Los registros de la tabla de datos se deben pintar de colores de acuerdo lo siguiente: Si se tiene sentimiento Negativo se deberá

pintar rojo, si se tiene sentimiento Positivo se deberá pintar verde y si se tiene sentimiento Neutro se deberá pintar gris.

- Cada registro de la tabla deberá contar con la siguiente información: Fotografía del perfil de usuario, nombre, nombre de usuario, número de seguidores, número de amigos, tuit, fecha de creación, sentimiento e idioma.

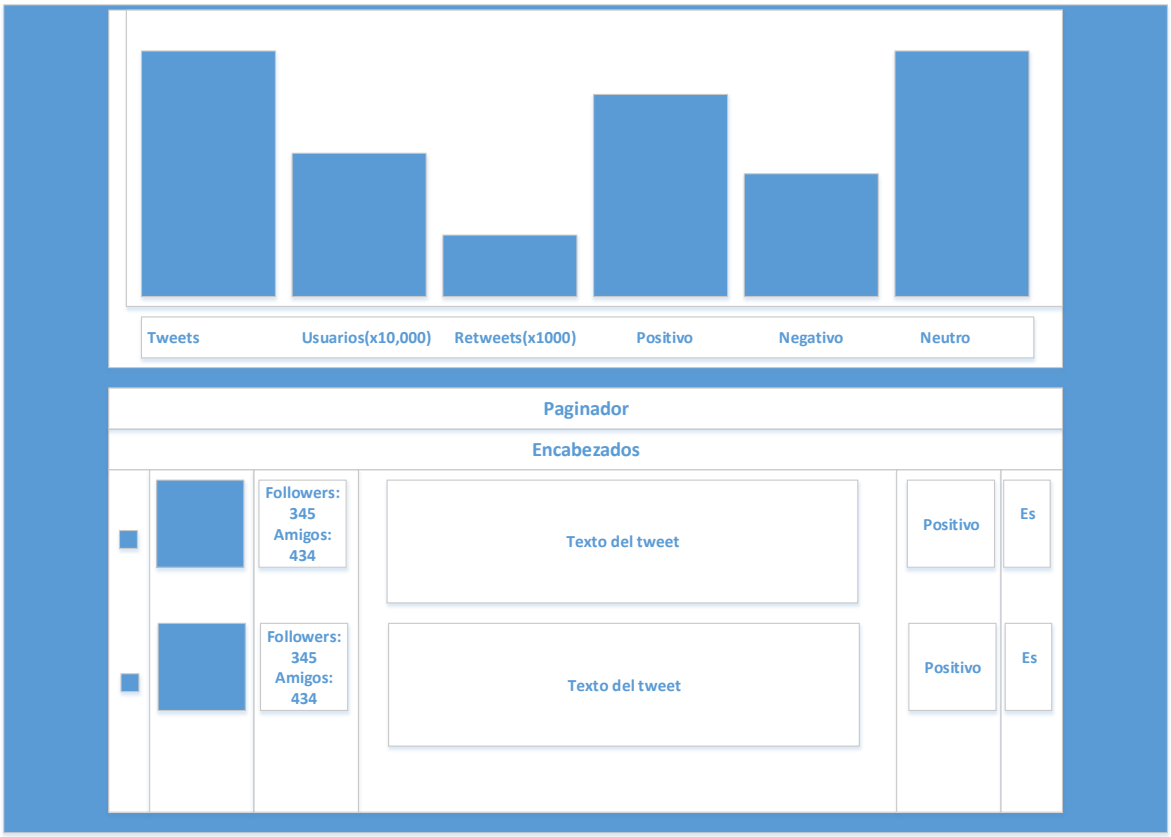


Figura 12: Módulo de mapas, sección table de resultados

Si ya se realizó una búsqueda y se quiere realizar una nueva, dentro del diálogo se tiene la posibilidad de seleccionar un checkbox llamado “Comparar” el cual realizará las mismas actualizaciones al momento de realizar la búsqueda con dos cambios.

1. En la gráfica de barras no se quitará la información y se pondrá nueva, sino que se agregará una serie nueva con otro color que cuente con la misma información. Con esto podremos comparar el número de tuits, el número de usuarios, el número de retuits, el número de tuits positivos, el número de negativos y el número de neutros de las 2 búsquedas.
2. En el mapa no se quitarán los puntos de la búsqueda anterior sino que se agregarán los nuevos puntos con un color diferente para poder comparar las dos búsquedas y diferenciar los resultados.

3.5.2. Módulo de búsqueda

El módulo de búsqueda permitirá realizar búsquedas generales sobre la información sin importar la región en la que se encuentran. En la parte izquierda de la página se mostrará un área de texto en la cual se escribirá el término de búsqueda y un botón con la leyenda “Buscar”. Una vez escrito el término de búsqueda y que se oprima el botón buscar se llevarán a cabo las siguientes acciones:

- Se muestra una gráfica de pie que representa el sentimiento (Positivo, Negativo o Neutro).
- Se muestra una gráfica de pie que representa el idioma en el cual se encuentra la información en este momento sólo se tiene soporte para español e inglés.
- Se presenta una tabla de datos con los tuits que cumplen con los términos de búsqueda realizados, esta tabla de datos debe cumplir con los siguientes requisitos;

Análisis y diseño

- La tabla de datos se debe poder ordenar de acuerdo al número de retuits que se tienen.
- Los registros de la tabla de datos se deben pintar de colores de acuerdo lo siguiente: Si se tiene sentimiento Negativo se deberá pintar rojo, si se tiene sentimiento Positivo se deberá pintar verde y si se tiene sentimiento Neutro se deberá pintar gris.
- Cada registro de la tabla deberá contar con la siguiente información: Fotografía del perfil de usuario, nombre, nombre de usuario, número de seguidores, número de amigos, tuit, fecha de creación, sentimiento e idioma.

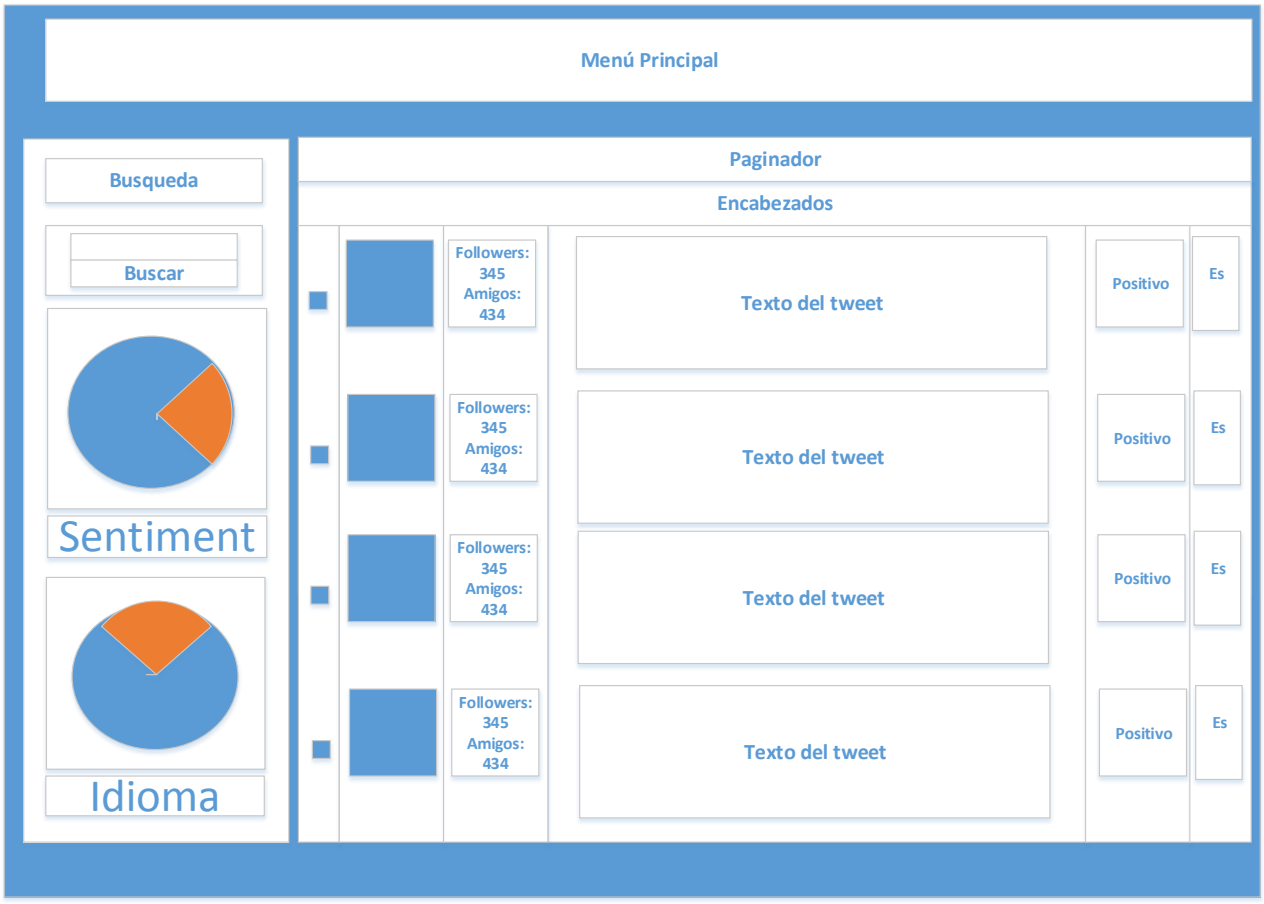


Figura 13: Módulo de búsqueda general

3.5.3. Módulo de perfiles

El módulo perfiles nos permitirá ver la información de la cuenta con la que se inició sesión, así como sus tuits, sus favoritos y sus retuits. La página estará compuesta por lo siguiente:

- La leyenda “Hola!” y el nombre de usuario de la cuenta con un link hacia la cuenta de la página de Twitter.
- La foto del usuario y la descripción de la cuenta.
- Una tabla de datos con la siguiente información:
    - “Tuits” El número de tuits realizados por el usuario.
    - “Siguiendo“ El número de usuarios a los que sigue la cuenta.

- “Seguidores” El número de seguidores que tiene la cuenta.

- Un área de texto con una longitud máxima de 140 caracteres y un botón con la leyenda Twittear. Una vez que se oprime el tuit y se oprime el botón se agregará a la lista de tuits del usuario y aparecerá en su cuenta oficial de Twitter.
- Una tabla de datos con los tuits favoritos del usuario, la tabla debe contar con una paginación de 5 en 5 registros, cada registro deberá contar con la siguiente información:
    - La imagen de la cuenta de usuario.
    - El nombre de usuario con un link hacia su cuenta de Twitter.
    - El nombre de la persona.
    - El texto del tuit.
    - La fecha en la que se escribió el tuit.
    - El número de retuits que tiene el tuit.
- Una tabla de datos con los tuits del usuario a los que se han hecho retuit, la tabla debe contar con una paginación de 5 en 5 registros, cada registro deberá contar con la siguiente información:
    - La imagen de la cuenta de usuario.
    - El nombre de usuario con un link hacia su cuenta de Twitter.
    - El nombre de la persona.
    - El texto del tuit.
    - La fecha en la que se escribió el tuit.
    - El número de retuits que tiene el tuit.

- Una tabla de datos con todos los tuits del usuario, la tabla debe contar con una paginación de 15 en 15 registros, cada registro deberá contar con la siguiente información:
    - La imagen de la cuenta de usuario.
    - El nombre de usuario con un link hacia su cuenta de Twitter.
    - El nombre de la persona.
    - El texto del tuit.
    - La fecha en la que se escribió el tuit.
    - El número de retuits que tiene el tuit.

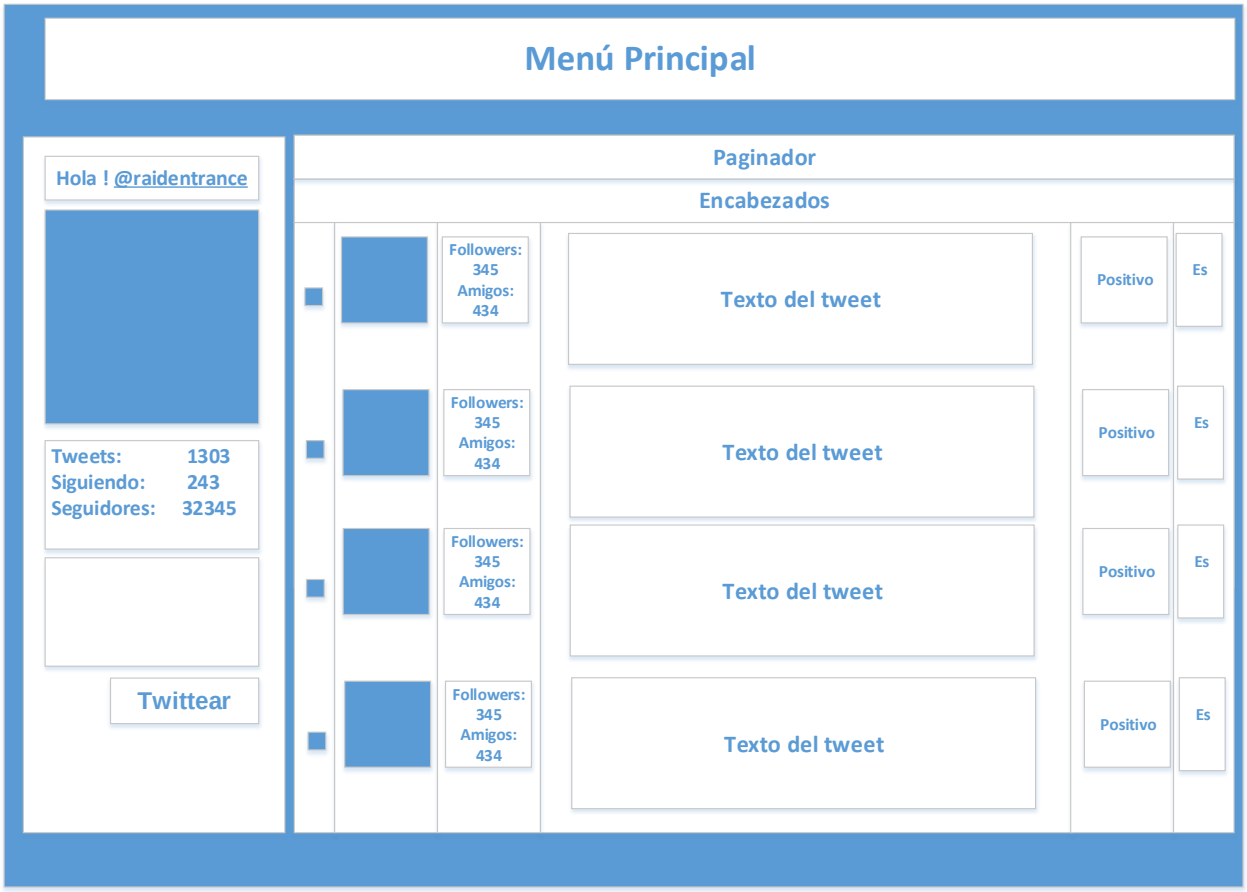


Figura 14: Módulo de Perfiles

3.5.4. Módulo de time line

El módulo “time line” muestra los tuits de los usuarios a los que sigue la cuenta con la que se inició sesión. La página está compuesta por lo siguiente:

- La leyenda “Hola!” y el nombre de usuario de la cuenta con un link hacia la cuenta de la página de Twitter.
- La foto del usuario y la descripción de la cuenta.
- Una tabla de datos con la siguiente información:
    - “Tuits” El número de tuits realizados por el usuario.
    - “Siguiendo“ El número de usuarios a los que sigue la cuenta.
    - “Seguidores” El número de seguidores que tiene la cuenta.
- Un área de texto con una longitud máxima de 140 caracteres y un botón con la leyenda Twittear. Una vez que se oprime el tuit y se oprime el botón se agregará a la lista de tuits del usuario y aparecerá en su cuenta oficial de Twitter.
- Una tabla de datos con todos los tuits de los usuarios a los que sigue la cuenta con la que se inició sesión, la tabla debe contar con una paginación de 15 en 15 registros, cada registro deberá contar con la siguiente información:
    - La imagen de la cuenta de usuario.
    - El nombre de usuario con un link hacia su cuenta de Twitter.
    - El nombre de la persona.
    - El texto del tuit.
    - La fecha en la que se escribió el tuit.

- El número de retuits que tiene el tuit.

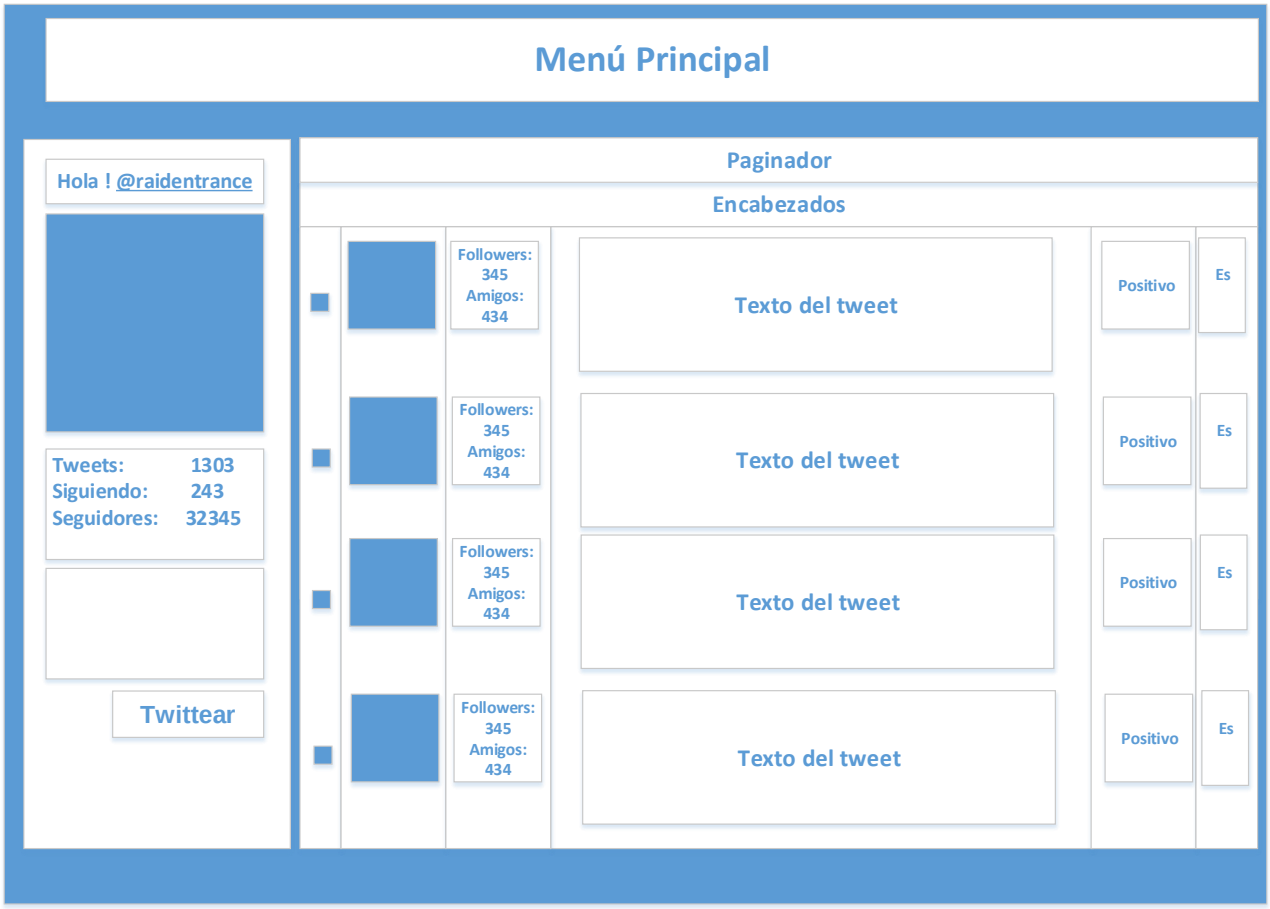


Figura 15: Módulo de time line

Capítulo 4

4. Desarrollo del sistema

Desarrollo del sistema

4.1. Estructura de la aplicación

El sistema Social Analysis fue construido utilizando Maven 2, esta es una herramienta de software para la gestión y construcción de proyectos Java, utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

El sistema Social Analysis es un sistema creado bajo la estructura Maven 2 a través de 4 módulos.

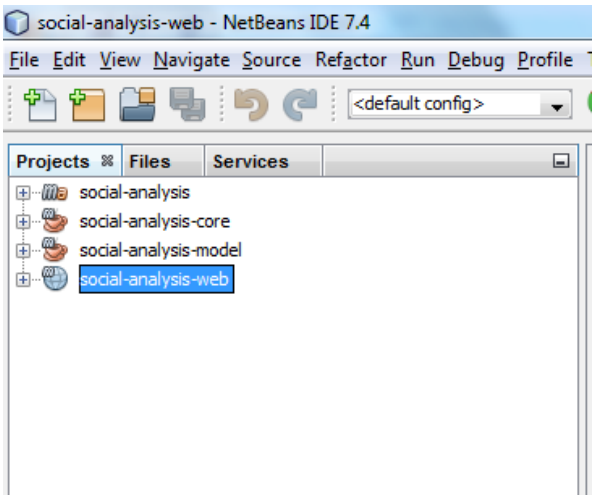


Figura 16: Estructura del proyecto

A continuación se describirán cada uno de los módulos del sistema:

- social-analysis: Es un proyecto de tipo POM, es utilizado como el padre de la aplicación y es en este donde se definen todos los módulos que formarán parte de la aplicación, así como las dependencias que serán comunes en todos los proyectos.

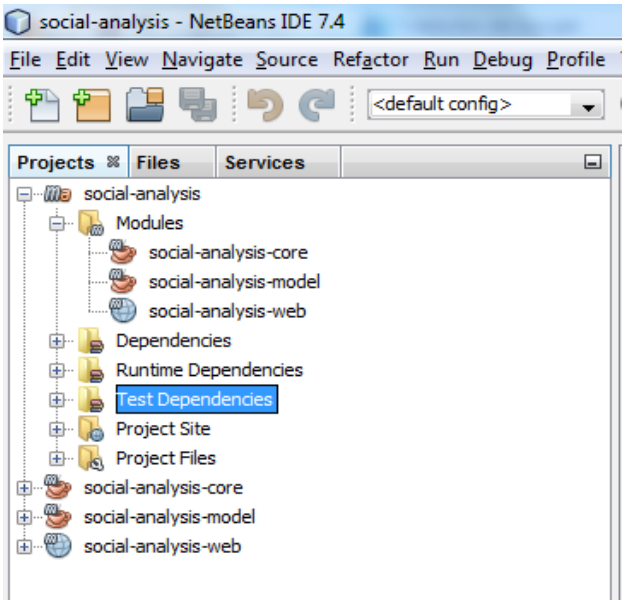


Figura 17: Proyecto padre

- social-analysis-model: Dentro de este proyecto se define el dominio de la aplicación, contiene clases tales como entidades, interfaces de los servicios (La implementación de dichas interfaces no se encuentra en este proyecto) y otras clases de dominio. Dentro de este proyecto se encuentra Java Persistence Api, Spring Test, Spring Core y Spring Data.

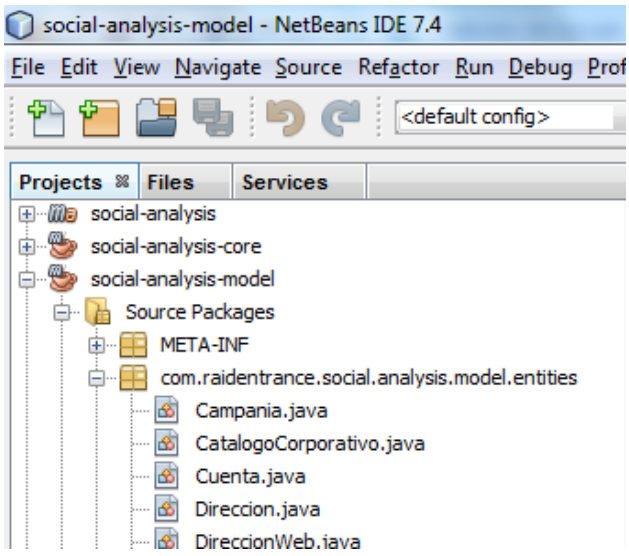


Figura 18: Social analysis model

- social-analysis-core: Dentro de este proyecto se definen los servicios que se utilizarán dentro de la aplicación, contiene la implementación de las interfaces que definidas en el proyecto social-analysis-model. En este proyecto se puede ver la aplicación de RestTemplate, Spring Core y Spring Test.

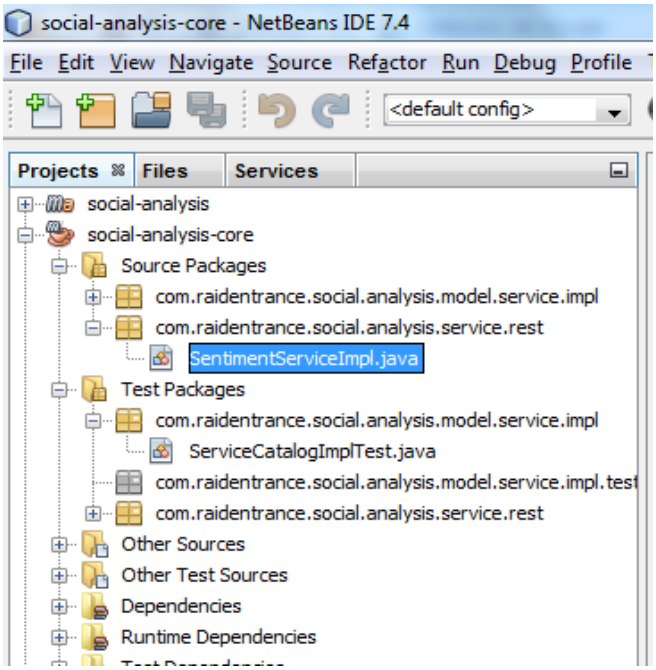


Figura 19: Social analysis core

- social-analysis-web: Dentro de este proyecto se crean los componentes referentes a la capa web de la aplicación. Incluye la implementación y uso de frameworks tales como JavaServer Faces, PrimeFaces, Spring Core y Spring Test.

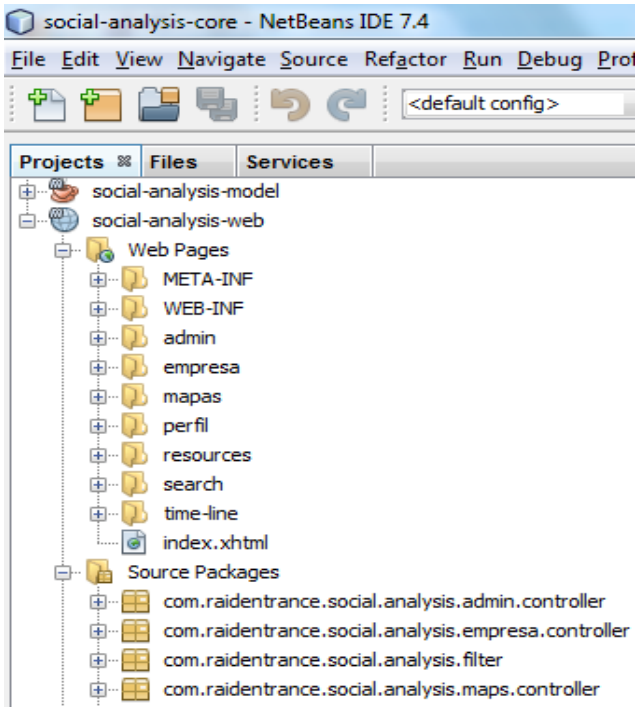


Figura 20: Social analysis web

4.2. Modelo Vista Controlador

MVC fue introducido por primera vez por Trygve Reenskaug, un desarrollador de Smalltalk en el Palo Alto Research Center de Xerox en 1979, y ayuda a separar el acceso a datos y la lógica de negocio de la manera en la que se muestra al usuario. MVC puede ser dividido en tres elementos:

Modelo - El modelo representa los datos y las normas que rigen el acceso y la actualización de estos datos. En el software de la empresa, un modelo a menudo sirve como una aproximación de software de un proceso del mundo real.

Vista - La vista hace el contenido de un modelo. En ella se especifica exactamente cómo se deben presentar los datos del modelo. Si los cambios en los datos de modelo, la vista debe actualizar su presentación según sea necesario. Esto se puede lograr mediante el uso de un modelo de empuje, en el que la vista

registra a sí mismo con el modelo para notificaciones de cambio, o un modelo de extracción, en el que la vista es responsable de llamar el modelo cuando se necesita para recuperar los datos más actuales.

Controlador - El controlador traduce las interacciones del usuario con la vista en acciones que el modelo funcionará. En un cliente GUI independiente, interacciones de los usuarios podrían ser clics de botón o selecciones de menú, mientras que en una aplicación web de la empresa, aparecen como peticiones GET y HTTP POST. Dependiendo del contexto, un controlador también puede seleccionar un nuevo punto de vista - por ejemplo, una página web de resultados - a presentar de nuevo al usuario.

4.2.1. Social Analysis y MVC

El sistema Social Analysis aplica el patrón de diseño Modelo vista controlador a través del uso del Framework JavaServer Faces aplicado como se muestra a continuación:

Vista.- La vista se realiza a través del uso de Facelets, son archivos `.xhtml` y representan las vistas de la aplicación.

Controlador.- Los controladores se realizan a través del uso de componentes llamados Managed Beans los cuales son los encargados de entablar comunicación con los Facelets comentados anteriormente realizar las operaciones correspondientes y devolviendo el resultado a otra vista como se muestra a continuación:

Modelo.- Las clases de modelo se realizan utilizando entidades como se muestra a continuación:

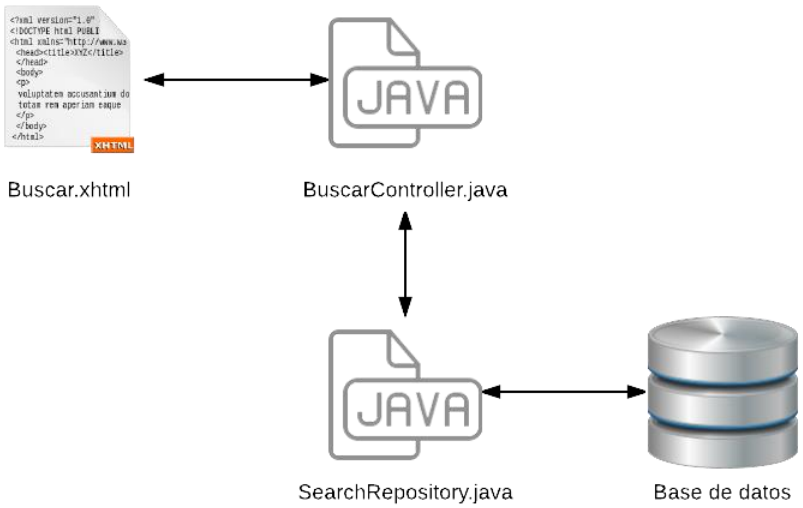


Figura 21: Modelo vista controlador

Dentro del sistema Social Analysis se aplica el patrón de diseño Modelo Vista Controlador. Cabe mencionar que se cuentan con diferentes fuentes de información (Base de datos y Twitter). El flujo que se presenta a continuación tiene como objetivo registrar un catálogo dentro de la aplicación. A continuación se muestra el flujo que se llevará a cabo para cumplir con el patrón.

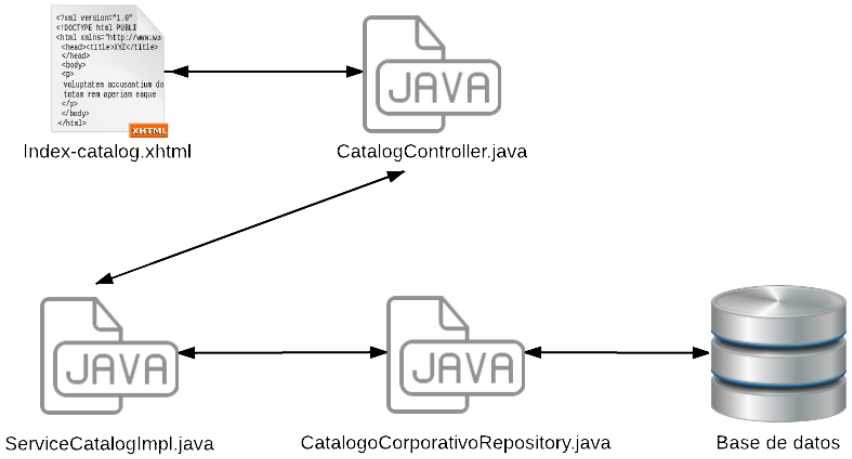


Figura 22: Modelo vista controlador en Social analysis

Desarrollo del sistema

A continuación se muestran cada uno de los componentes.

index-catalog.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:p="http://primefaces.org/ui"
      xmlns:f="http://java.sun.com/jsf/core">
    <body>
        <ui:composition template="./../resources/templates/template-one-col.xhtml">
            <ui:define name="meta">
                <f:loadBundle basename="admin/Admin" var="msgs_admin"/>
            </ui:define>
            <ui:define name="top">
                <h2>#{msgs_admin.admin_title}</h2>
            </ui:define>
            <ui:define name="content">
                <h1>#{msgs_admin.admin_title}</h1><br/>
                <h:form>
                    <p:panelGrid columns="3">
                        #{msgs_admin.admin_form_nombre}
                      <p:inputText id="nombre" required="true"
                    value="#{catalogController.catalogoCorporativo.nombre}"/>
                      <p:message for="nombre"/>
                        #{msgs_admin.admin_form_descripcion}
                      <p:inputText id="descripcion" required="true"
                    value="#{catalogController.catalogoCorporativo.descripcion}"/>
                      <p:message for="descripcion"/>
                        #{msgs_admin.admin_form_catalogo}
                      <p:selectOneMenu id="catalogo"
                    value="#{catalogController.catalogoCorporativo.idGrupo}" >
                          <f:selectItem itemLabel="Seleccionar uno" itemValue="-
1"/>
                          <f:selectItems
                    value="#{catalogController.verNombreCatalogo()}" var="_cat"
                    itemLabel="#{_cat.nombre}" itemValue="#{_cat.id}"/>
                      </p:selectOneMenu>
                      <p:message for="catalogo"/>
                      <p:commandButton action="#{catalogController.guardar()}"
                    value="#{msgs_admin.admin_form_boton_guardar}" ajax="false"/>
                    </p:panelGrid>
                </h:form>
                <p:dataTable style="width: 50%"
            value="#{catalogController.verTodosLosCatalogos()}" var="_catalog"
            rowKey="#{_catalog.id}"
                             paginator="true" rows="10">
                    <f:facet name="header">
                        <h:outputText value="#{msgs_admin.admin_table_header}"/>
                    </f:facet>
                    <p:column headerText="#{msgs_admin.admin_table_column_header_id}">
                        #{_catalog.id}
                    </p:column>
                    <p:column
headerText="#{msgs_admin.admin_table_column_header_idGrupo}">
                        #{_catalog.idGrupo}
                    </p:column>
                    <p:column
headerText="#{msgs_admin.admin_table_column_header_nombre}">
                        #{_catalog.nombre}
                    </p:column>
                      <p:column
                    headerText="#{msgs_admin.admin_table_column_header_descripcion}" >
                        #{_catalog.descripcion}
                    </p:column>
```

```
                        <p:column
headerText="#{msgs_admin.admin_table_column_header_activo}" >
                            #{_catalog.activo}
                        </p:column>
                    </p:dataTable>
                </ui:define>
            </ui:composition>
        </body>
    </html>
```

En el código fuente dentro de este archivo solo se presenta código cuyo objetivo es presentar y obtener información dentro de la aplicación siguiendo el patrón de diseño Modelo Vista Controlador. El archivo **index-catalog.xhtml** toma el papel de la vista, con este tipo de componentes se siguen buenas prácticas tales como el uso de templates para reducir el código duplicado y externar las etiquetas de la aplicación a archivos properties.

CatalogController.java

```
package com.raidentrance.social.analysis.admin.controller;

import com.raidentrance.social.analysis.model.entities.CatalogoCorporativo;
import com.raidentrance.social.analysis.model.service.ServiceCatalog;
import java.util.List;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;
import org.apache.commons.lang.builder.ToStringBuilder;

@ManagedBean
@RequestScoped
public class CatalogController {

    @ManagedProperty("#{serviceCatalogImpl}")
    private ServiceCatalog serviceCatalogImpl;
    @ManagedProperty("#{catalogoCorporativo}")
    private CatalogoCorporativo catalogoCorporativo;

    public String guardar() {
  catalogoCorporativo.setIdGrupo((catalogoCorporativo.getIdGrupo()==-
 1)?null:catalogoCorporativo.getIdGrupo());
        catalogoCorporativo.setActivo(Boolean.TRUE);
        serviceCatalogImpl.saveCatalog(catalogoCorporativo);
        return "admin";
    }

    public List<CatalogoCorporativo> verTodosLosCatalogos() {
        return serviceCatalogImpl.findAll();
    }

    public List<CatalogoCorporativo> verNombreCatalogo() {
        return serviceCatalogImpl.findCatalogNames();
    }

    public CatalogoCorporativo getCatalogoCorporativo() {
        return catalogoCorporativo;
    }
```

```
        public void setCatalogoCorporativo(CatalogoCorporativo catalogoCorporativo) {
            this.catalogoCorporativo = catalogoCorporativo;
        }

        public void setServiceCatalogImpl(ServiceCatalog serviceCatalogImpl) {
            this.serviceCatalogImpl = serviceCatalogImpl;
        }


    }
```

La clase `CatalogController` tiene como objetivo tomar la información de la vista `index-catalog.xhtml` colocarlas en el atributo `catalogoCorporativo` y a través del atributo `serviceCatalogImpl` se invoca su almacenamiento en la base de datos y regresa a la vista “admin” (este nombre se encuentra definido en el archivo de configuración `faces-config.xml`). Dentro de este código se pueden apreciar otras técnicas tales como la inyección de dependencias como podemos ver no se está creando el objeto a través del operador `new`, se está realizando inyección de dependencias utilizando el contenedor de Spring Framework.

`ServiceCatalogImpl.java`

```
package com.raidentrance.social.analysis.model.service.impl;

import com.google.common.collect.Lists;
import com.raidentrance.social.analysis.model.entities.CatalogoCorporativo;
import
com.raidentrance.social.analysis.model.repository.CatalogoCorporativoRepository;
import com.raidentrance.social.analysis.model.service.ServiceCatalog;
import java.io.Serializable;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service("serviceCatalogImpl")
@Transactional
public class ServiceCatalogImpl implements ServiceCatalog,Serializable{

    @Autowired
    private CatalogoCorporativoRepository catalogoCorporativoRepository;

    @Override
    public void saveCatalog(CatalogoCorporativo catalogo) {
        catalogoCorporativoRepository.save(catalogo);
    }

    @Override
    public List<CatalogoCorporativo> findByIdGrupo(Integer idGrupo) {
```

```
            return catalogoCorporativoRepository.findByIdGrupo(idGrupo);
        }

        @Override
        public List<CatalogoCorporativo> findCatalogNames() {
            return catalogoCorporativoRepository.findNamesOfCatalogs();
        }

        @Override
        public List<CatalogoCorporativo> findAll() {
            return Lists.newArrayList(catalogoCorporativoRepository.findAll());
        }

        @Override
        public CatalogoCorporativo findByName(String name) {
            return catalogoCorporativoRepository.findByNombre(name);
        }

        @Override
        public CatalogoCorporativo findById(Integer id) {
            return catalogoCorporativoRepository.findOne(id);
        }

    }
```

Como se puede observar la clase **CatalogController** utiliza una referencia de tipo **ServiceCatalog** la cual es una interfaz, la clase **ServiceCatalogImpl** es una implementación de dicha interfaz esta clase es la que define el método save(..) utilizado por **CatalogController.**

CatalogoCorporativoRepository.java

```
package com.raidentrance.social.analysis.model.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.raidentrance.social.analysis.model.entities.CatalogoCorporativo;
import java.util.List;
import org.springframework.data.jpa.repository.Query;

@Repository("catalogoCorporativoRepository")
public interface CatalogoCorporativoRepository extends
CrudRepository<CatalogoCorporativo, Integer> {

    public CatalogoCorporativo findByNombre(String nombre);

    public List<CatalogoCorporativo> findByIdGrupo(Integer idGrupo);

    @Query("SELECT c FROM CatalogoCorporativo c WHERE c.idGrupo = null")
    public List<CatalogoCorporativo> findNamesOfCatalogs();

}
```

La clase **CatalogoCorporativoRepository** es parte del módulo Spring Data de Spring Framework y es utilizada para evitar el uso del patrón de diseño

DAO (Data Access Object) ya que con el solo hecho de crear una interfaz que herede de `CrudRepository` y colocar tanto el tipo de dato que se desea persistir como el tipo de dato de la llave primaria de dicho tipo de dato, Spring Data realizará la implementación para realizar las operaciones llamadas CRUD (Create, Read, Update y Delete).

Spring Data cuenta con otras facilidades que vale la pena mencionar, por ejemplo si se desea crear un método que no exista en los métodos que proporciona `CrudRepository` podemos crear nuestros propios métodos y con solo nombrarlo findBy + nombre de un atributo de la entidad, Spring data deducirá la implementación que se desea, en la clase que se mostró se puede observar el siguiente método:

```
public CatalogoCorporativo findByNombre(String nombre);
```

El método se llama findByNombre y recibe como argumento String nombre, con esto Spring Data deduce que el método recibirá un nombre y buscará un `CatalogoCorporativo` a través de ese nombre y realizará la implementación por sí solo.

Otro tema que vale la pena apreciar es el uso de la anotación `@Query` la cuál es utilizada para crear métodos que no existan en los proporcionados por `CrudRepository` y que no puedan ser descritos en forma sencilla por el nombre, en la clase que se mostró se puede observar el siguiente método:

```
@Query("SELECT c FROM CatalogoCorporativo c WHERE c.idGrupo = null")
public List<CatalogoCorporativo> findNamesOfCatalogs();
```

En esta firma se coloca como argumento de la anotación de `@Query` una consulta en lenguaje JPQL (Java Persistence Query Language).

`CatalogoCorporativo.java`

```
package com.raidentrance.social.analysis.model.entities;

import com.google.common.base.Objects;
import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "catalogo_corporativo")
public class CatalogoCorporativo implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(unique = true, nullable = false)
    private Integer id;
    private Boolean activo;
    @Column(length = 100)
    private String descripcion;
    @Column(name = "ID_GRUPO")
    private Integer idGrupo;
    @Column(length = 45)
    private String nombre;

         // Getters and Setters

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 47 * hash + Objects.hashCode(this.id);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final CatalogoCorporativo other = (CatalogoCorporativo) obj;
        if (this.id != other.id && (this.id == null || !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

}
```

La clase **`CatalogoCorporativo`** es una entidad que se utiliza para realizar el ORM (Object Relational Mapping) de la aplicación esto es la

representación del modelo relacional en el modelo orientado a objetos. Es importante apreciar que además de representar la tabla o sus atributos es posible representar desde aquí los constraints o reglas que se tengan en la base de datos.

4.3. Utilización de servicios web Rest

Dentro de la aplicación se hace uso de servicios Rest para asignar el idioma y el sentimiento (positivo, negativo o neutro) a un tuit. Para consumir estos servicios web se hace uso de `RestTemplate` este es un módulo de Spring creado para consumir servicios web de tipo Rest de una forma simple y óptima haciendo uso de una implementación de JSON llamada Jackson la cual nos ayuda a transformar de un texto en formato json a clases java. El uso del módulo de Spring Framework `RestTemplate` se representa en el siguiente diagrama:

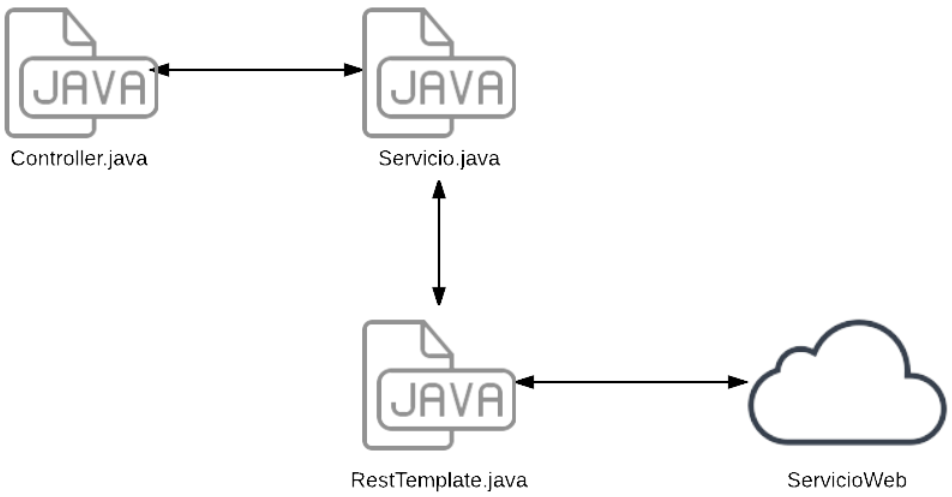


Figura 23: Consumo de web services REST utilizando RestTemplate

4.3.1. Utilidad del servicio REST de ejemplo

El servicio Rest que se va a invocar requiere de una URL a la cual se va a realizar la petición y una serie de parámetros los cuales serviran como entrada a ese servicio. La URL que vamos a invocar es http://www.sentiment140.com/api/bulkClassifyJson que es el lugar en el que se encuentra el servicio, y los parámetros que enviaremos serán los siguientes:

Id: Representa el identificador del texto a enviar.

Text: Representa el texto a evaluar.

Query: Representa una palabra clave a la cual se le desea dar énfasis.

Un ejemplo de una petición ya construida es el siguiente:

```
{"data": [{"text": "I love Titanic.", "query": "Titanic", "id": 1234},
        {"text": "I hate Titanic.", "query": "Titanic", "id": 4567}]}
```

Una vez que se envía a la URL mencionada los parámetros anteriores este servicio proporciona una respuesta como la que se muestra a continuación:

```
{"data": [{"text": "I love Titanic.", "id":1234, "polarity": 4},
        {"text": "I hate Titanic.", "id":4567, "polarity": 0}]}
```

En esta respuesta se puede observar lo siguiente, el texto que se envió como parámetro, el identificador del texto y un campo llamado polarity, a continuación se muestra el significado de este campo.

- Polarity 0: Negativo
- Polarity 2: Neutral
- Polarity 4: Possitivo

4.3.2. Configuración de RestTemplate

`serviceApplicationContext.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:jpa="http://www.springframework.org/schema/data/jpa"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
        ://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.0.xsd
        ://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.1.xsd
        ://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.1.xsd">

    <context:component-scan base-package="com.raidentrance"></context:component-scan>

    <bean id="restTemplate" class="org.springframework.web.client.RestTemplate  ">
        <property name="messageConverters">
            <list>
                <ref bean="jsonConverter" />
            </list>
        </property>
    </bean>

    <bean                                                          id="jsonConverter"
class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">
        <property name="supportedMediaTypes" value="application/json" />
    </bean>
</beans>
```

En este archivo de configuración registraremos los beans que deseemos que se encuentren en el contenedor de Spring Framework, como podemos observar se registró un bean llamado `jsonConverter` al cual le colocamos `supportedMediaTypes` el valor `application/json` para indicar que se requiere convertir de formato JSON a java. Este bean `jsonConverter` nos servirá para crear el bean `restTemplate` ya que se le asignará como convertidor. El bean `restTemplate` será el que se inyectará a las clases para poder consumir un servicio Rest.

`SentimentServiceImpl.java`

```
package com.raidentrance.social.analysis.service.rest;

import com.raidentrance.social.analysis.model.sentiment.Data;
import com.raidentrance.social.analysis.model.sentiment.SentimentService;
import com.raidentrance.social.analysis.model.sentiment.Status;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

@Service("sentimentServiceImpl")
public class SentimentServiceImpl implements SentimentService,Serializable {

    @Autowired
    private RestTemplate restTemplate;
   @Value("#{sentiment.url}")
    private String SENTIMENT_URL;

    private static String proccesaString(String input) {
        // Cadena de caracteres original a sustituir.
        String original = "áàäéèëíìïóòöúùuñÁÀÄÉÈËÍÌÏÓÒÖÚÙÜÑçÇ";
        // Cadena de caracteres ASCII que reemplazarán los originales.
        String ascii = "aaaeeeiiiooouuunAAAEEEIIIOOOUUUNcC";
        String output = input;
        for (int i = 0; i < original.length(); i++) {
            // Reemplazamos los caracteres especiales.
            output = output.replace(original.charAt(i), ascii.charAt(i));
        }//for i
        return output;
    }//remove1

    @Override
    public Data getData(List<Twitter4j.Status> status, String languaje,String query) {
        Data data = new Data();
        List<Status> lista = new ArrayList<Status>();
        for (Twitter4j.Status st : status) {
            Status s = new Status();
            s.setId(st.getId());
            s.setQuery(query);
            try {
                s.setText(new   String(proccesaString(st.getText()).getBytes(),   "UTF-
8"));
            } catch (UnsupportedEncodingException ex) {

Logger.getLogger(SentimentServiceImpl.class.getName()).log(Level.SEVERE, null, ex);
            }
            lista.add(s);
        }
        data.setData(lista);
        data.setLanguage(languaje);
         return restTemplate.postForObject(sentimentURL, data, Data.class);
     }

    public RestTemplate getRestTemplate() {
        return restTemplate;
    }

    public void setRestTemplate(RestTemplate restTemplate) {
        this.restTemplate = restTemplate;
    }
}
```

El servicio **SentimentServiceImpl** tiene dos atributos importantes `restTemplate` que es el bean que definimos dentro del archivo `serviceApplicationContext.xml` que nos ayudará a realizar peticiones rest y el `sentimentURL` el cual define la URL a la cual se le realizará la petición Rest.

El método `getData`(..) está encargado de obtener la lista de tuits (objetos de tipo Status) hacer procesos sobre éstos y mandarlos como argumento al servicio Rest al que vamos a invocar. Como se puede observar sólo se utiliza una línea para realizar la petición rest (línea remarcada en el código).

4.4. JavaServer Faces y PrimeFaces en Social Analysis

Dentro de Social Analysis se tomó la decisión de no utilizar Spring MVC y utilizar JavaServer Faces y PrimeFaces debido a las ventajas que brinda con respecto a Spring MVC, a continuación presentaremos algunas de estas aplicadas a Social Analysis.

Uso de templates

Los templates son plantillas en las cuales se determina la estructura base de un sitio y es común para distintas páginas esto nos permite separar el diseño del sitio del contenido que se presenta en el mismo, además de que si se desea realizar un cambio en varias páginas solo se necesita modificar un solo archivo y no todas las páginas a las que se desea realizar el cambio.

Para hacer uso de estos componentes es necesario crear un archivo llamado Facelet Template que es un archivo `.xhtml` en el cual colocaremos la estructura del sitio, se debe considerar que el contenido de éste será compartido por diferentes páginas, además del Facelet Template se tiene que crear un archivo llamado Facelet Template Cliente el cual es conocido como un cliente, éste va a heredar todo el estilo definido en el Facelet Template y sólo deberá contener el contenido específico de esa página. A continuación veremos un ejemplo:

Facelet Template

Desarrollo del sistema

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
   <h:head>
       <title></title>
       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
       <meta name="description" content="" />
       <meta name="keywords" content="" />
      <link
 href="http://fonts.googleapis.com/css?family=Open+Sans:400,600,700,300|Open+Sans+Con
 densed:300,700" rel="stylesheet" />

       <h:outputScript library="js" name="jquery.dropotron-1.2.js"/>
       <h:outputScript library="js" name="jquery.slidertron-1.2.js"/>
       <h:outputScript library="js" name="init.js"/>
       <ui:insert name="meta"/>
       <noscript>
           <h:outputStylesheet library="css" name="/5grid/core.css"/>
           <h:outputStylesheet library="css" name="/5grid/core-desktop.css"/>
           <h:outputStylesheet library="css" name="/5grid/core-1200px.css"/>
           <h:outputStylesheet library="css" name="/5grid/core-noscript.css"/>
           <h:outputStylesheet library="css" name="style.css"/>
           <h:outputStylesheet library="css" name="style-desktop.css"/>
       </noscript>
   </h:head>
   <h:body class="left-sidebar">
       <div id="header-wrapper">
           <div class="5grid-layout">
               <div class="row">
                   <div class="12u">
                       <nav id="nav" class="mobileUI-site-nav">
                           <ul>
                             <li            class="current_page_item"><h:form><h:link
                           outcome="index.xhtml" value="Inicio"/></h:form></li>
                                   <li><h:form><h:link                    outcome="mapas"
                             value="Mapas"/></h:form></li>
                                   <li><h:form><h:link                outcome="busqueda"
                             value="Búsqueda"/></h:form></li>
                                   <li><h:form><h:link   outcome="time-line"   value="Time
                             line"/></h:form></li>
                                   <li><h:form><h:link                    outcome="perfil"
                             value="Perfil"/></h:form></li>
                                <li><h:form><h:commandLink
                           action="#{cerrarSesionBean.invalidate()}"         value="Cerrar
                           Sesión"/></h:form></li>
                           </ul>
                       </nav>
                   </div>
               </div>
               <div class="row">
                   <div class="12u">
                       <header id="header">
                      <h1><a href="#" class="mobileUI-site-name">Raidentrance<span> -
                    -> Social Analysis</span></a></h1>
                       </header>
                   </div>
               </div>
           </div>
       </div>
       <div id="main-wrapper">
           <div id="main" class="5grid-layout">
               <div class="row">
                   <div id="sidebar" class="4u">
                       <ui:insert name="left"/>
                   </div>
```

```
                    <div class="8u mobileUI-main-content">
                        <article id="content">
                            <ui:insert name="content"/>
                        </article>
                    </div>

                </div>
            </div>
        </div>
    </h:body>
</html>
```

Como se puede observar este Facelet Template es un archivo xhtml común que cuenta con algunas peculiaridades, la que nos compete para el uso de templates es la etiqueta **<ui:insert name=" "/>** esta etiqueta nos indica que en ese lugar se va a inyectar código de un cliente de tal modo que el Facelet Template y el Facelet Template Client se van a combinar para hacer uno solo, es importante ver que cada etiqueta insert requiere un nombre para identificar el contenido que vendrá del cliente. A continuación se presentará el Facelet Template Client.

Facelet Template Client

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.org/ui"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:c="http://java.sun.com/jsp/jstl/core">
   <body>
      <ui:composition
 template="./../resources/templates/left-sidebar.xhtml">
         <ui:define name="left">
            <h2>Búsqueda</h2>
            <br/>
            <h:form prependId="false">
                <p:panelGrid columns="1" style="width: 100%;">
                  <p:inputText id="search" value="#{searchController.query}"
             required="true"/>
                    <p:commandButton value="Buscar"
              action="#{searchController.simpleSearch()}" ajax="false"/>
                </p:panelGrid>
            </h:form>
            <p:pieChart id="sentiment_chart" style="width: 100%; height:358px;"
          rendered="#{not (searchController.sentiment.size() eq 0)}"
          value="#{searchController.pieModelSentiment}" legendPosition="w"/>
```

```
                    <p:pieChart id="language_chart" style="width: 100%; height:358px;"
                rendered="#{not (searchController.pieModelLanguage.data.size() eq 0)}"
                value="#{searchController.pieModelLanguage}" legendPosition="w"/>
                </ui:define>
                <ui:define name="content">
                    <h2>Resultados</h2>
                    <br/>
                        <p:dataTable id="results_data"
                        value="#{searchController.resultados}" var="_resultado"
                        rowKey="#{_resultado.id}" paginator="true" rows="15"
                                sortBy="#{_resultado.retuitCount}"
                            rowStyleClass="#{(searchController.rateSentiment(_resultado.id
                            ).polarity eq
                            0)?'rojo':(searchController.rateSentiment(_resultado.id).polar
                            ity eq
                            2)?'gris':(searchController.rateSentiment(_resultado.id).polar
                            ity eq 4)?'verde':''}" sortOrder="descending"
                            selectionMode="single" >
                        <p:ajax event="rowSelect"
                    oncomplete="window.location.replace('http://Twitter.com/#{_resultado.use
                    r.screenName}/status/#{_resultado.id}');" />

                        <c:set var="sentiment_value"
                    value="#{searchController.rateSentiment(_resultado.id).polarity}"/>
                        <c:set var="sentiment_v" value="#{(sentiment_value eq
                    0)?'Negativo':(sentiment_value eq 2)?'Neutro':(sentiment_value eq
                    4)?'Positivo':''}"/>

                        <p:column >
                            <h:graphicImage
url="#{_resultado.user.biggerProfileImageURL}"/>
                            <br/>
                            <h3>
                          @<h:outputLink
                        value="#{'http://Twitter.com/'}#{_resultado.user.screenName}">#{_resu
                        ltado.user.screenName}</h:outputLink>
                            </h3>
                            #{_resultado.user.name}
                        </p:column>
                        <p:column headerText="Texto" sortBy="#{_resultado.retuitCount}">
                            <span>
                                #{_resultado.text}
                            </span>
                            <br/>
                            <h:outputText value="#{_resultado.createdAt}">
                                <f:convertDateTime pattern="dd-MM-yyyy"/>
                            </h:outputText>
                            <div><h3>#{_resultado.retuitCount} Retuits</h3></div>

                            <br/>
                        </p:column>
                        <p:column headerText="Sentiment" sortBy="#{sentiment_v}">
                            <h:outputText value="#{sentiment_v}"/>
                        </p:column>

                    </p:dataTable>
                </ui:define>
            </ui:composition>
        </body>
    </html>
```

Éste archivo es de tipo Facelet Template Client, como se puede observar, en este aquí no es necesario definir la maqueta del sitio ni el css (Cascading Style Sheets) el cual es el lenguaje utilizado para describir el aspecto y formato de la

aplicación ya que hereda del Facelet Template base, en este código es importante apreciar lo siguiente:

La etiqueta `<ui:composition template="./../resources/templates/left-sidebar.xhtml">` nos dice que este Facelet Template Client va a tomar como base el Facelet Template `left-sidebar.xhtml`.

La etiqueta `<ui:define name="left">` nos indica que todo lo que se encuentre dentro de ella será inyectado en el `<ui:insert name=" "/>` que definimos en el Facelet Template.

El uso de templates nos ayuda a separar en el Facelet Template el diseño y la estructura de la página y en el Facelet Template Client el código que presenta la información representativa de la página, es importante mencionar que un Facelet Template puede ser aplicado a muchos Facelet Template Clients.

4.4.1. Uso de componentes PrimeFaces

PrimeFaces es un perfecto complemento a JavaServer Faces ya que brinda un conjunto de componentes que permiten realizar tareas comunes de forma simple y limpia.

A continuación se presenta un ejemplo de su uso en el sistema para realizar gráficas y tablas de datos con paginación y ordenamientos complejos de forma simple y limpia.

Desarrollo del sistema

Index-busqueda.xhtml

```
<p:pieChart id="sentiment_chart" style="width: 100%; height:358px;"
rendered="#{not (searchController.sentiment.size() eq 0)}"
value="#{searchController.pieModelSentiment}" legendPosition="w"/>

<p:pieChart id="language_chart" style="width: 100%; height:358px;"
rendered="#{not (searchController.pieModelLanguage.data.size() eq 0)}"
value="#{searchController.pieModelLanguage}"
legendPosition="w"/>

<p:dataTable id="results_data" value="#{searchController.resultados}"
var="_resultado" rowKey="#{_resultado.id}" paginator="true" rows="15"
sortBy="#{_resultado.retuitCount}"
rowStyleClass="#{(searchController.rateSentiment(_resultado.id).polarity eq
0)?'rojo':(searchController.rateSentiment(_resultado.id).polarity eq
2)?'gris':(searchController.rateSentiment(_resultado.id).polarity eq
4)?'verde':''}" sortOrder="descending" selectionMode="single" >

<p:ajax event="rowSelect"
oncomplete="window.location.replace('http://Twitter.com/#{_resultado.user.scr
eenName}/status/#{_resultado.id}');" />

<c:set var="sentiment_value"
value="#{searchController.rateSentiment(_resultado.id).polarity}"/>
<c:set var="sentiment_v" value="#{(sentiment_value eq
0)?'Negativo':(sentiment_value eq 2)?'Neutro':(sentiment_value eq
4)?'Positivo':''}"/>
<p:column >
<h:graphicImage
url="#{_resultado.user.biggerProfileImageURL}"/>
<br/>
<h3>
@<h:outputLink
value="#{'http://Twitter.com/'}#{_resultado.user.screenName}">#{_resu
ltado.user.screenName}</h:outputLink>
</h3>
#{_resultado.user.name}
</p:column>
<p:column headerText="Texto" sortBy="#{_resultado.retuitCount}">
<span>
#{_resultado.text}
</span>
<br/>
<h:outputText value="#{_resultado.createdAt}">
<f:convertDateTime pattern="dd-MM-yyyy"/>
</h:outputText>
<div><h3>#{_resultado.retuitCount} Retuits</h3></div>

<br/>
</p:column>
<p:column headerText="Sentiment" sortBy="#{sentiment_v}">
<h:outputText value="#{sentiment_v}"/>
</p:column>

</p:dataTable>
```

La etiqueta **<p:dataTable>** es utilizada para crear una tabla de datos a partir de una lista de Java, a continuación se muestran algunos atributos interesantes:

- `paginator="true"` y `rows="15"` indican que se desea que la tabla de datos cuente con paginación y ésta sea de 15 en 15.
- `sortBy="#{_resultado.retuitCount}"` se indica que se tendrá un ordenamiento por el campo `retuitCount` que para la aplicación es el número de retuits que tenga una publicación.
- `rowStyleClass=""` indica que dependiendo del tipo de resultado se va a pintar de un color u otro la tabla de datos, para el caso del ejemplo se toma el sentimiento, si el sentimiento es positivo se va a pintar color verde si es negativo rojo y si es neutro será gris.

La etiqueta **`<p:pieChart>`** es utilizada para crear gráficos de tipo pie a partir de un objeto llamado pieModel , el título del gráfico y la posición en la que se desea mostrar.

El uso de los componentes de PrimeFaces nos permite utilizar complejos componentes de forma simple y sin utilizar javascript esto nos permite tener un código limpio y fácil de modificar. A continuación se muestra una imagen con el resultado del fragmento de código que se presentó.

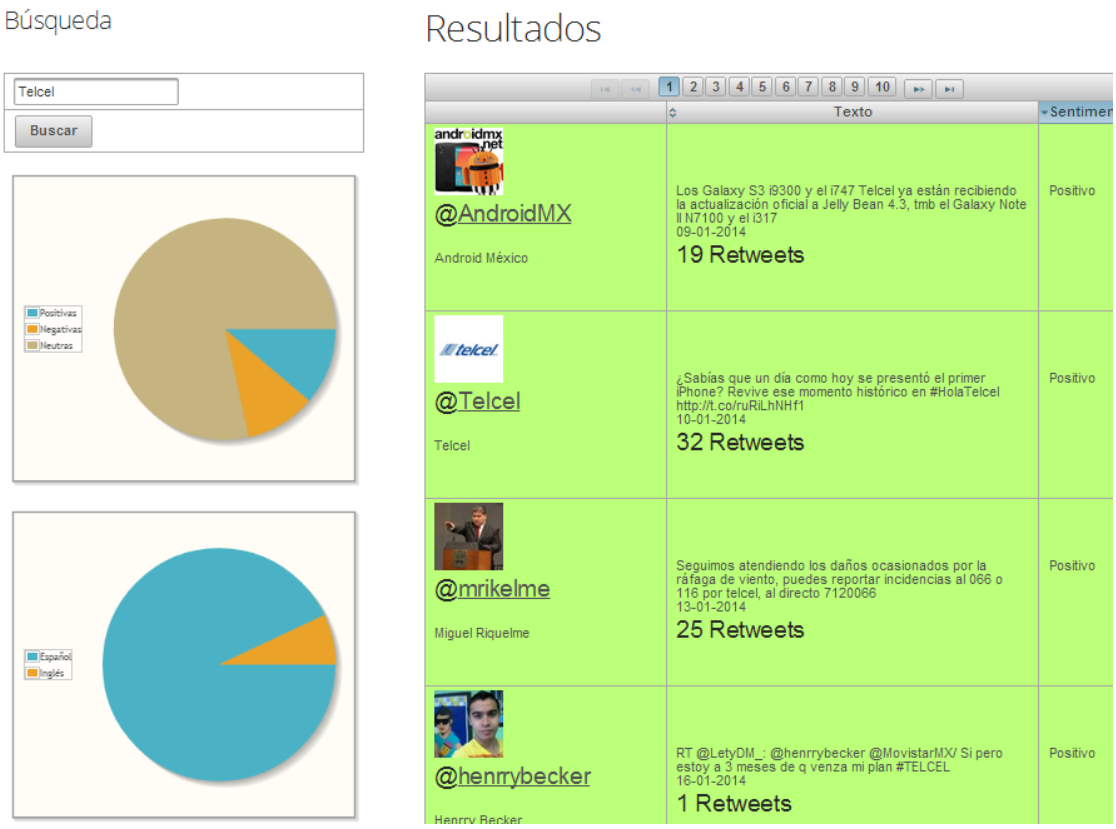


Figura 24: Uso de PrimeFaces en social analysis

4.4.2. PrimeFaces y Google Maps

PrimeFaces cuenta con un componente que permite trabajar con mapas de Google a través de etiquetas JavaServer Faces y con muy poco código JavaScript. A continuación se muestra la aplicación de este componente en el sistema Social Analysis.

```
<p:gmap id="gmap" center="19.273716,-99.138393" zoom="5"
                        style="width:100%;height:400px;"
                        model="#{mapController.emptyModel}"
                        onPointClick="handlePointClick(event);"
                        widgetVar="map" />
```

La etiqueta **<p:gmap>** representa el mapa de google que se va a presentar en el sitio, esta etiqueta cuenta con los siguientes atributos:

- `center="19.273716,-99.138393"` indica la latitud y longitud en la cual se desea que se presente el mapa inicialmente.

- `zoom="5"` indica el zoom en el cual se desea que se presente el mapa inicialmente.
- `style="width:100%;height:400px;"` indica el tamaño del mapa dentro de la página.
- `model="#{mapController.emptyModel}"` el modelo nos ayuda a representar objetos que se pueden mostrar dentro del mapa, para Social Analysis se requiere pintar puntos que representan el lugar en el cuál se escribió un tuit.
- `onPointClick="handlePointClick(event);"` es utilizado para invocar código JavaScript al momento de que se realiza un click en el mapa.
- `widgetVar="map"` indica el nombre con el cual se podrá trabajar con ese objeto en JavaScript.

El sistema Social Analysis requiere que cuando se haga click en el mapa se muestre un cuadro de diálogo en el cual se escribirá el término de búsqueda el radio de búsqueda y los botones para buscar y cancelar, para hacer esto se utiliza el siguiente código JavaScript.

```
<script>
    var currentMarker = null;
    function handlePointClick(event) {
        if (currentMarker == null) {
            document.getElementById('lat').value = event.latLng.lat();
            document.getElementById('lng').value = event.latLng.lng();

            currentMarker = new google.maps.Marker({
                position:          new          google.maps.LatLng(event.latLng.lat(),
event.latLng.lng())
            });

            map.addOverlay(currentMarker);
            dlg.show();
        }
    }

    function markerAddComplete() {
        var title = document.getElementById('query');
```

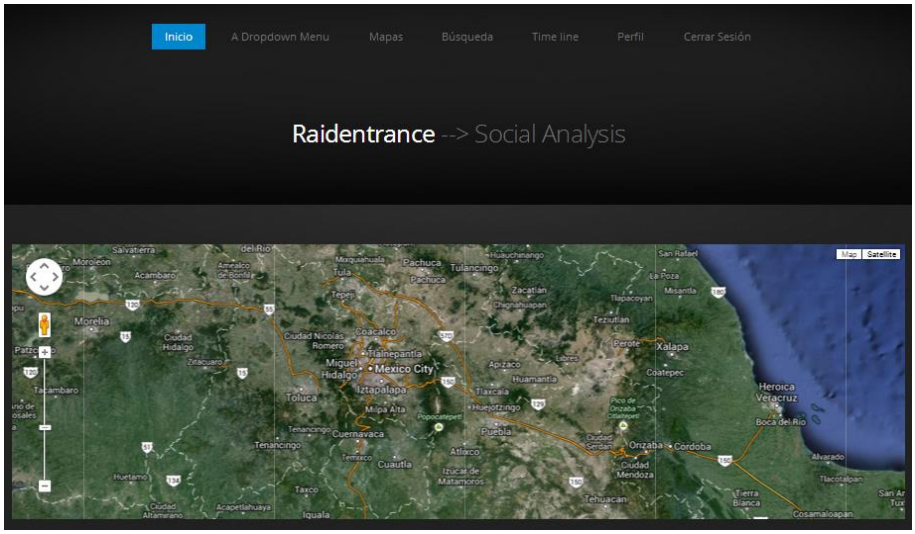
```
            currentMarker.setTitle(title.value);
            title.value = "";

            currentMarker = null;
            dlg.hide();
        }

        function cancel() {
            dlg.hide();
            currentMarker.setMap(null);
            currentMarker = null;

            return false;
        }
    </script>
```

Este código permite tomar la latitud y longitud en la cual se hizo click para poder utilizarlo al momento de realizar la consulta por región de Twitter. A continuación se muestra una imagen con el resultado del fragmento de código que se presentó



Mapa una vez que se hace click en el.

Desarrollo del sistema

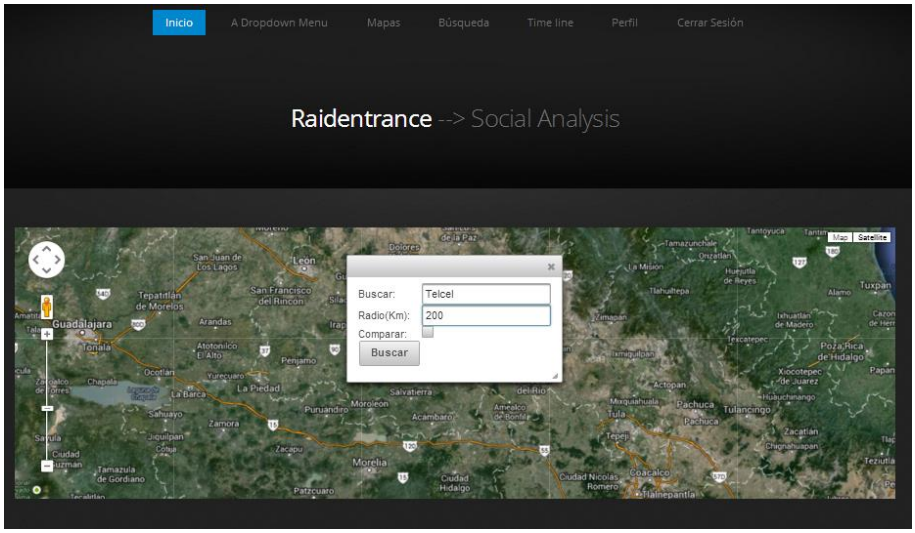


Figura 25: Mapas de Google utilizando PrimeFaces

Mapa después de que se hace click sobre él.

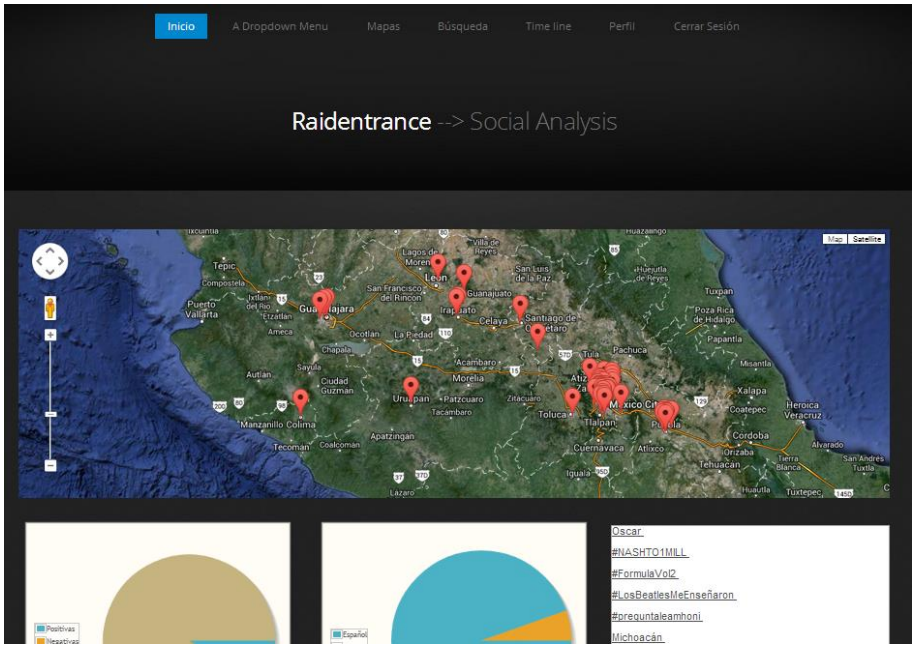


Figura 26: Manejo de eventos en los mapas de Google

Como se puede observar, dentro del sistema Social Analysis intervienen un conjunto muy amplio de herramientas, frameworks, API's, Patrones de diseño y

lenguajes de programación que trabajan en conjunto generando un sistema estable y capaz de implementar el diseño establecido en el capítulo anterior.

Capítulo 5

5. Pruebas y análisis de resultados

Pruebas y análisis de resultados

5.1. Pruebas en Social Analysis

5.1.1. Pruebas Unitarias

Se tomó la decisión de realizar las pruebas unitarias del sistema Social Analysis a través del framework JUnit en conjunto con el módulo de Spring Framework Spring Test, esto porque existen algunos aspectos importantes a considerar, entre ellos la necesidad de levantar un contexto de Spring para así poder probar los componentes que se encontrarán en dicho contexto. A continuación se presenta el ejemplo de un componente y como se realizó el test unitario sobre él.

El componente a probar es la interfaz **`CatalogoCorporativoRepository`** esta realiza operaciones sobre la base de datos en la tabla **`CATALOGO_CORPORATIVO`** y se compone por 3 métodos **`findByNombre, findByIdGrupo y findNamesOfCatalogs`** más los métodos que hereda de la interfaz **`CrudRepository`**.

```
    @Repository("catalogoCorporativoRepository")
    public interface CatalogoCorporativoRepository extends
CrudRepository<CatalogoCorporativo, Integer> {

        public CatalogoCorporativo findByNombre(String nombre);

        public List<CatalogoCorporativo> findByIdGrupo(Integer idGrupo);

        @Query("SELECT c FROM CatalogoCorporativo c WHERE c.idGrupo =
null")
        public List<CatalogoCorporativo> findNamesOfCatalogs();

    }
```

Los objetivos de las pruebas son los siguientes:

- Demostrar que el método **`findByNombre`** realiza búsquedas en la tabla a través de la cadena que recibe como argumento.

Pruebas y análisis de resultados

- Demostrar que el método **`findByIdGrupo`** realiza búsquedas en la tabla a través del idGrupo que recibe como argumento.
- Demostrar que el método **`findNamesOfCatalogs`** realiza búsquedas en la tabla a través de la consulta que se encuentra en la anotación `@Query`.

El proyecto se construyó utilizando Maven, por esto se colocaron los test unitarios en la carpeta “Test Packages” del proyecto, se creó un test unitario llamado **`TestCaseCatalagoCorporativoRepository`** el cual contiene métodos dedicados a probar los métodos anunciados anteriormente.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/jpaApplicationContext.xml")
public class TestCaseCatalagoCorporativoRepository {

    @Autowired
    private CatalogoCorporativoRepository catalogoCorporativoRepository;

    @Before
    public void beforeCatalago() {
        CatalogoCorporativo catalogo = new CatalogoCorporativo();
        catalogo.setActivo(true);
        catalogo.setDescripcion("Catálogo de paises");
        catalogo.setIdGrupo(null);
        catalogo.setNombre("Pais");
        catalogoCorporativoRepository.save(catalogo);
    }

    @Test
    public void tesFindByNombre() {
        assertNotNull(catalogoCorporativoRepository);
        CatalogoCorporativo corporativo = catalogoCorporativoRepository
                .findByNombre("Pais");
        assertNotNull(corporativo);
    }

    @Test
    public void tesFindCatalogNames() {
        assertNotNull(catalogoCorporativoRepository);
        List<CatalogoCorporativo>                corporativo                =
catalogoCorporativoRepository
                .findNamesOfCatalogs();
        assertNotNull(corporativo);
        for (CatalogoCorporativo catalogoCorporativo : corporativo) {
            System.out.println(catalogoCorporativo.getNombre());
        }
    }

    @Test
```

```
        public void testCatalago() {
            assertNotNull(catalogoCorporativoRepository);
            CatalogoCorporativo catalago = new CatalogoCorporativo();
            CatalogoCorporativo                    paises                    =
catalogoCorporativoRepository.findByNombre("Pais");
            assertNotNull(paises);
            catalago.setIdGrupo(paises.getId());
            catalago.setActivo(true);
           catalago.setDescripcion("Descripción de México");
          catalago.setNombre("México");
            catalogoCorporativoRepository.save(catalago);

        }

        @After
        public void afterCatalago() {

catalogoCorporativoRepository.delete(catalogoCorporativoRepository.findAll());
        }
    }
```

Dentro de la clase **`TestCaseCatalagoCorporativoRepository`** se pueden analizar los siguientes puntos:

- Las anotaciones `@RunWith y @ContextConfiguration` indican que se van a ejecutar las pruebas con el motor de JUnit de Spring y que se requiere configurar y cargar un Application Context de Spring con el archivo `"/jpaApplicationContext.xml"`.

```
@RunWith(SpringJUnit4ClassRunner.class)
      @ContextConfiguration("/jpaApplicationContext.xml")
public class TestCaseCatalagoCorporativoRepository {
```

- La anotación `@Autowired` indica que se va a inyectar un objeto de tipo `CatalogoCorporativoRepository` tomado desde el contenedor de Spring.

```
@Autowired
      private                          CatalogoCorporativoRepository
catalogoCorporativoRepository;
```

- La anotación `@Before` indica que se requiere la ejecución de ese método antes de cada prueba que se va a ejecutar.

```
@Before
public void beforeCatalago() {
```

- La anotación `@Test` indica que el método se comportará como una prueba unitaria.

```
@Test
public void tesFindCatalogNames() {
```

- La anotación `@After` indica que el método se ejecutará después de cada test unitario.

```
@After
public void afterCatalago() {
```

Este tipo de pruebas son útiles para monitorear los componentes de la aplicación y así conocer si un cambio en alguna parte del sistema puede afectar a otros componentes del mismo. Al ejecutar la prueba se pueden observar las pruebas que se ejecutaron, sólo se ejecutaron los métodos que están marcados con la anotación `@Test`.

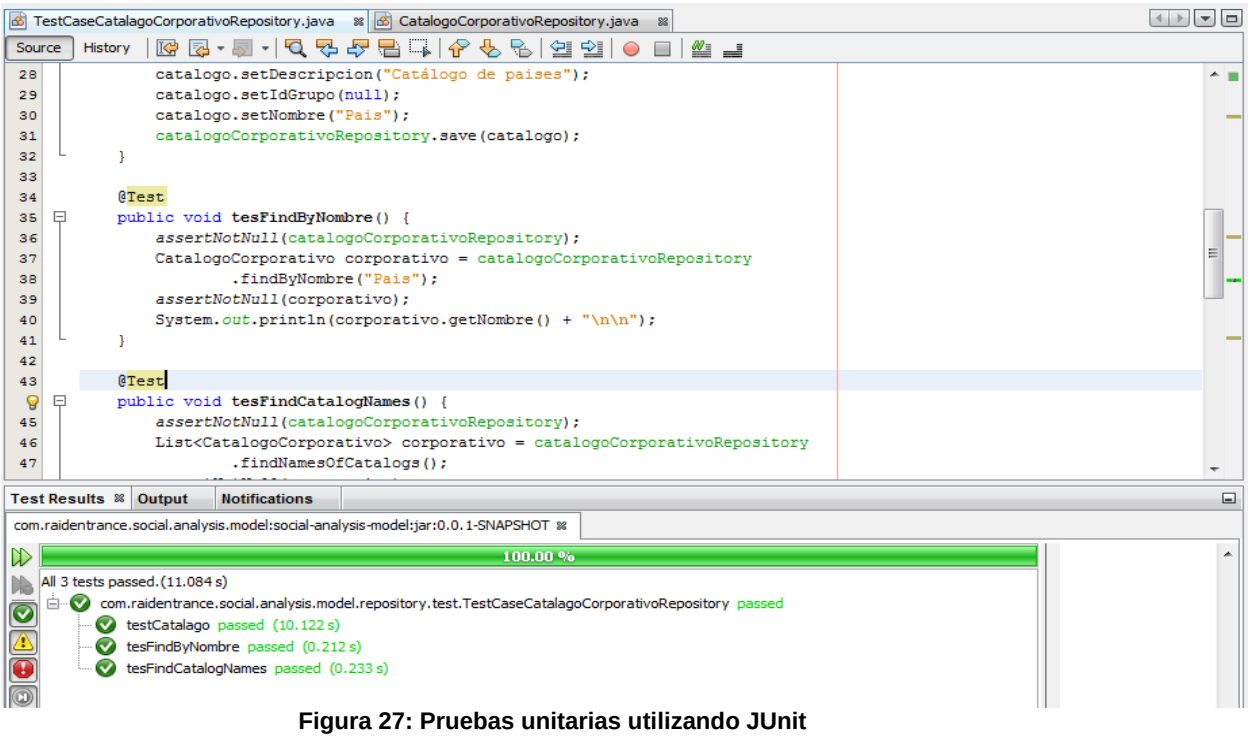


Figura 27: Pruebas unitarias utilizando JUnit

5.1.2. Pruebas funcionales

Pruebas funcionales del módulo de mapas:

Al ingresar al módulo de mapas se presenta la siguiente figura:

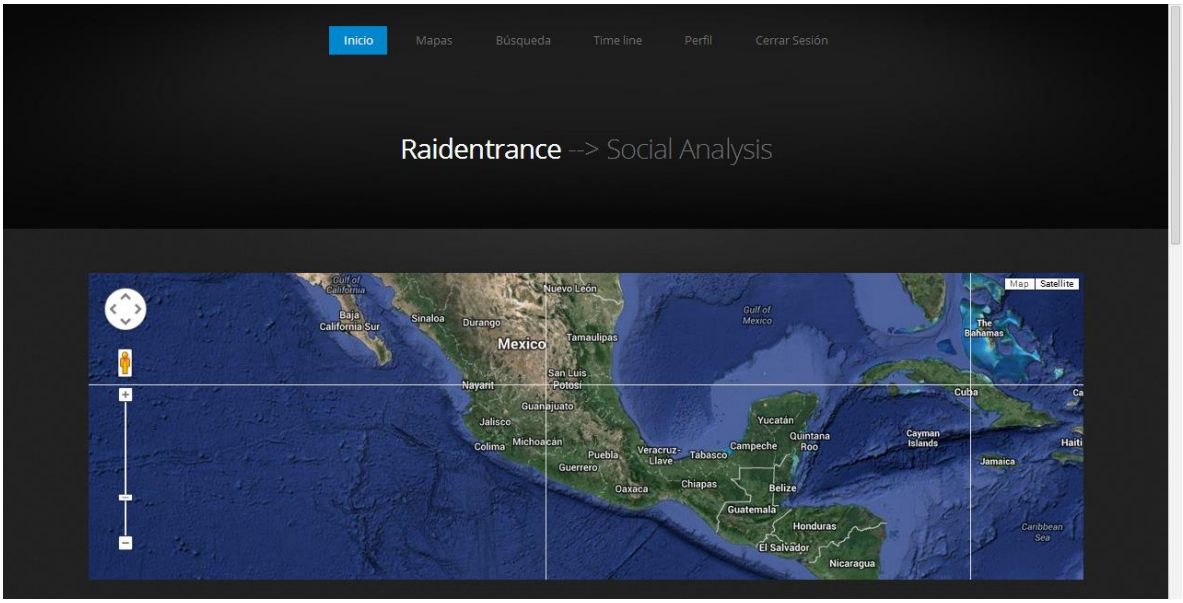


Figura 28: Pruebas funcionales en el módulo de mapas prueba 1

En ésta se presentan los siguientes menús:

- Inicio: Al hacer click en este menú se muestra la página de inicio del sistema.
- Mapas: Al hacer click en este menú se refresca el módulo de mapas del sistema.
- Búsqueda: Al hacer click en este menú se muestra el módulo de búsqueda del sistema.
- Time line: Al hacer click en este menú se muestra el módulo de búsqueda del sistema.
- Perfil: Al hacer click en este menú se muestra el módulo de perfil del sistema.

- Cerrar sesión: Al hacer click en este menú se termina la sesión del usuario activo en el sistema.

En la parte central de la pantalla se muestra un mapa tomado de Google Maps, dentro de este mapa se tiene la capacidad de ubicarse en diferentes partes del mundo desplazándolo con el mouse.

En el siguiente caso se ubicará en la Ciudad de México y se hará click sobre el mapa, con esto se presenta el siguiente cuadro de diálogo.

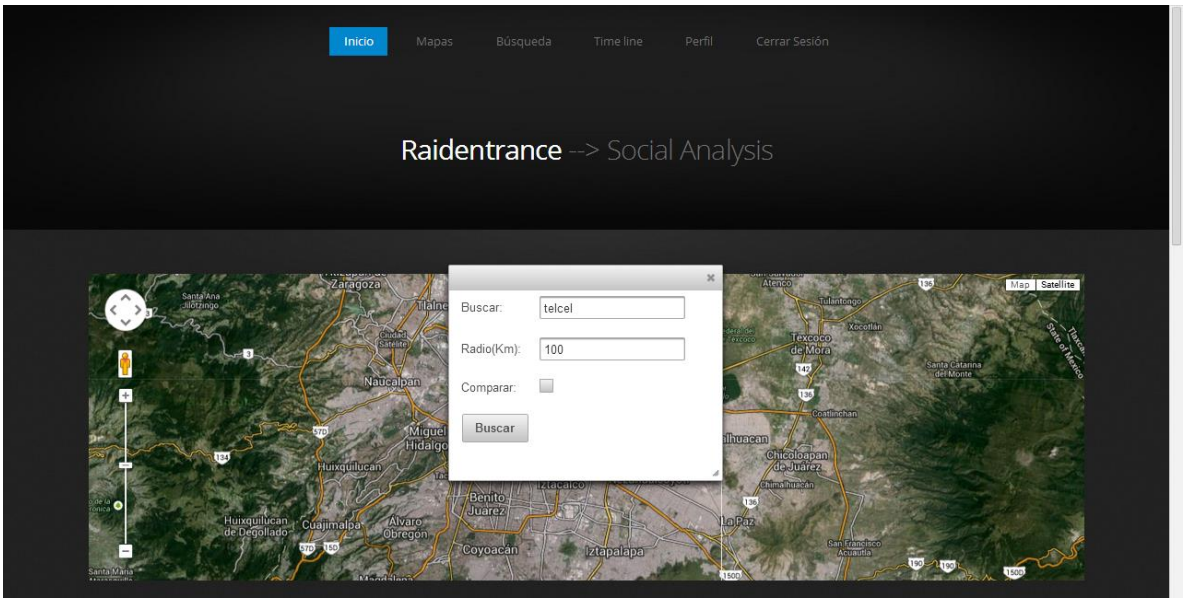


Figura 29: Pruebas funcionales en el módulo de mapas, prueba 2

Se capturó en el diálogo los siguientes valores:

- Buscar: Telcel
- Radio(Km): 100

Estos campos representan el término a buscar en Twitter y el radio expresado en Kilómetros a partir del que se buscará desde el punto en el que se

hizo click. Al hacer click en el botón “Buscar”. Se mostrará la siguiente pantalla (La pantalla será partida en 4 partes):

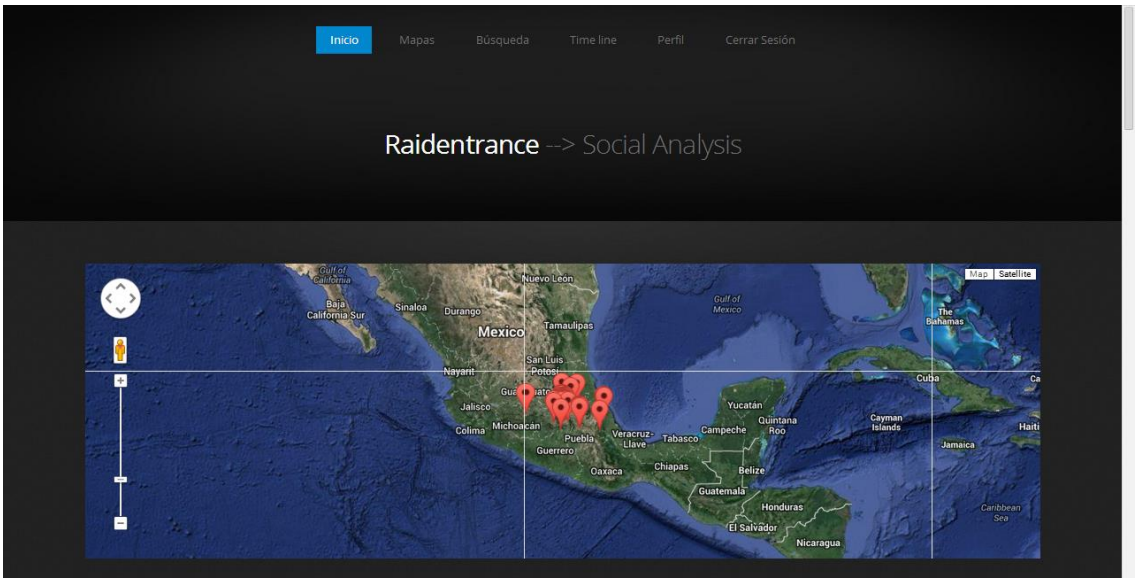


Figura 30: Pruebas funcionales en el módulo de mapas prueba 3

En esta zona de la pantalla se pueden observar puntos plasmados en el mapa, estos puntos representan la ubicación de los tuits (publicaciones) de aquellos usuarios que tienen la localización activa en su dispositivo al momento de realizar la publicación. A esta búsqueda se le asignó el color rojo, más adelante se detallará la importancia del color de estos puntos en el mapa.

Pruebas y análisis de resultados

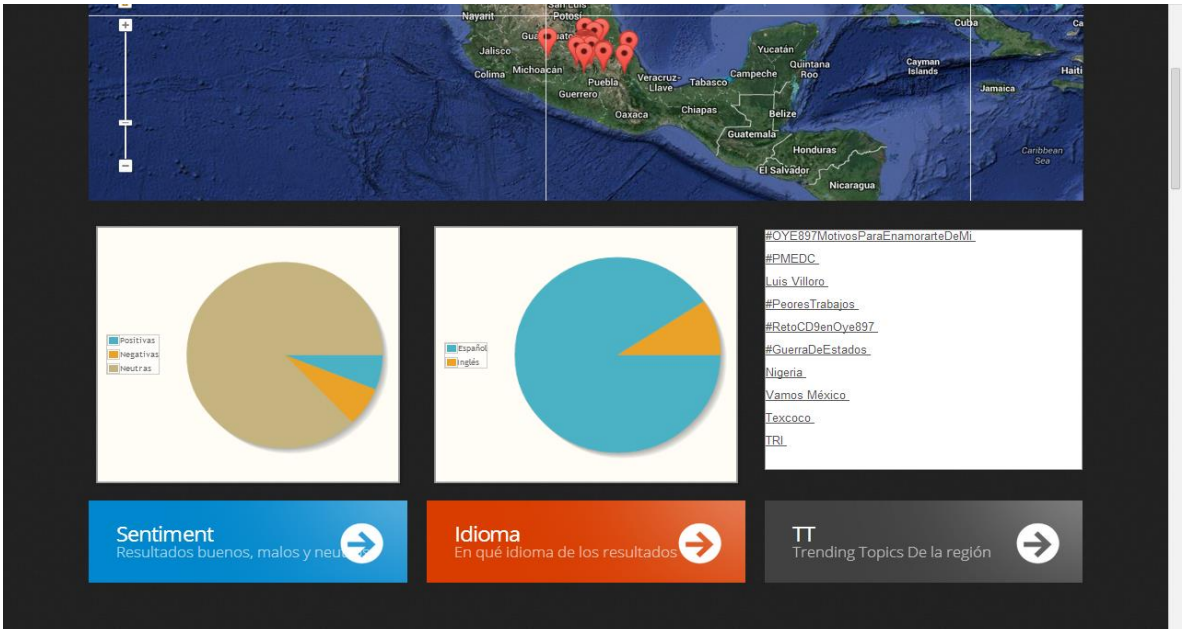


Figura 31: Pruebas funcionales en el módulo de mapas prueba 4

En esta zona de la pantalla se pueden apreciar 3 segmentos en los cuales se presenta lo siguiente:

- Sentiment: En esta gráfica de pie se presentan los resultados positivos, negativos y neutros de los resultados obtenidos de la búsqueda realizada.
- Idioma: En esta gráfica de pie se presentan los idiomas encontrados en los resultados obtenidos, en este momento el sistema cuenta con soporte para los idiomas (español e inglés).
- TT: En este segmento se muestran los Trendig Topics de la región sobre la cual se realizó la búsqueda.

Pruebas y análisis de resultados

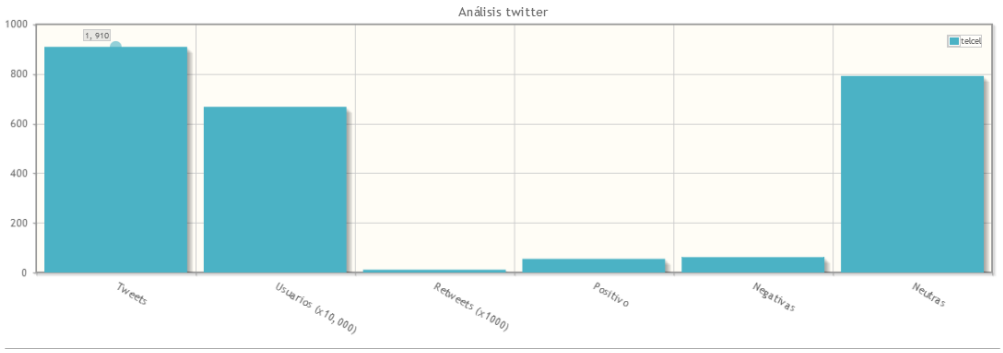


Figura 32: Pruebas funcionales en el módulo de mapas prueba 5

En la gráfica Análisis de Twitter se presentan los siguientes resultados:

- Tuits: Número de tuits (publicaciones) encontradas en la búsqueda.
- Usuarios(x10,000): Este número representa al número de usuarios a los que llegaron los tuits(publicaciones) relacionados con el término de búsqueda.
- Retuits(x1000): Número de retuits totales encontrados en todos los resultados obtenidos en la búsqueda.
- Positivo: Número de resultados positivos encontrados en la búsqueda.
- Negativo: Número de resultados negativos encontrados en la búsqueda.
- Neutro: Número de resultados neutros encontrados en la búsqueda.

Pruebas y análisis de resultados

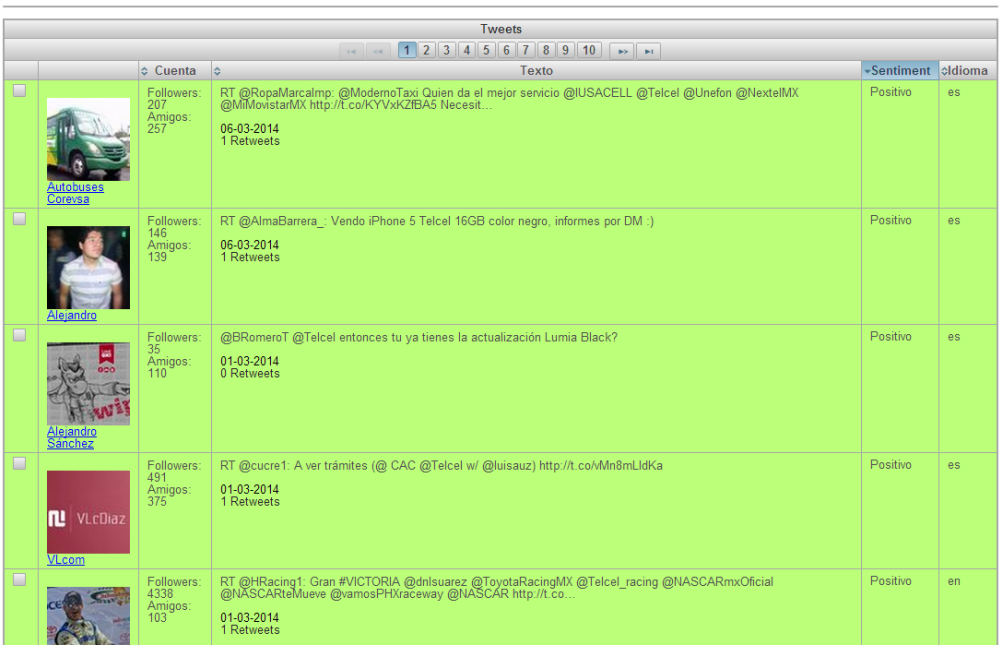


Figura 33: Pruebas funcionales módulo de mapas prueba 6

En esta pantalla se muestran los resultados encontrados en la búsqueda con la siguiente información:

- Imagen del perfil de Twitter del usuario.
- Nombre de usuario de Twitter.
- Número de seguidores del usuario.
- Número de amigos del usuario.
- Tuit (Publicación).
- Fecha de la publicación.
- Número de Retuits del tuit.
- Sentiment.
- Idioma.

Pruebas y análisis de resultados

Es importante resaltar que la publicación se mostrará en color verde si el tuit es positivo, rojo si el tuit es negativo o gris si el resultado es neutro, además de que la tabla se puede ordenar para mostrar solo los resultados positivos, negativos o neutros, así como para ver primero los resultados con mayor número de retuits, con mayor número de seguidores o por idioma. A continuación se presenta la tabla ordenada de modo que muestre primero los resultados negativos.

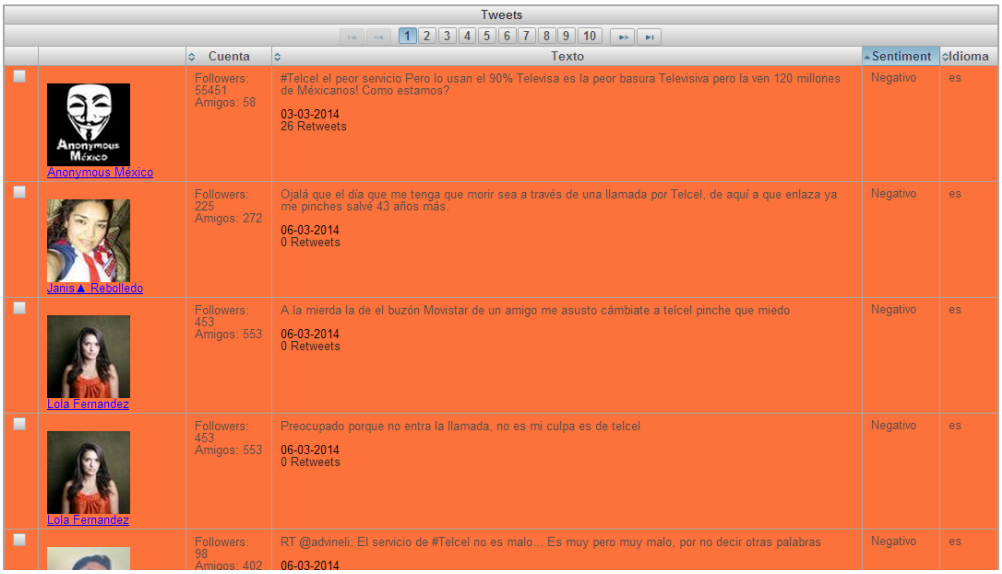


Figura 34: Pruebas funcionales módulo de mapas prueba 7

Pruebas y análisis de resultados

A continuación se presenta la pantalla mostrando los resultados con mayor número de retuits:

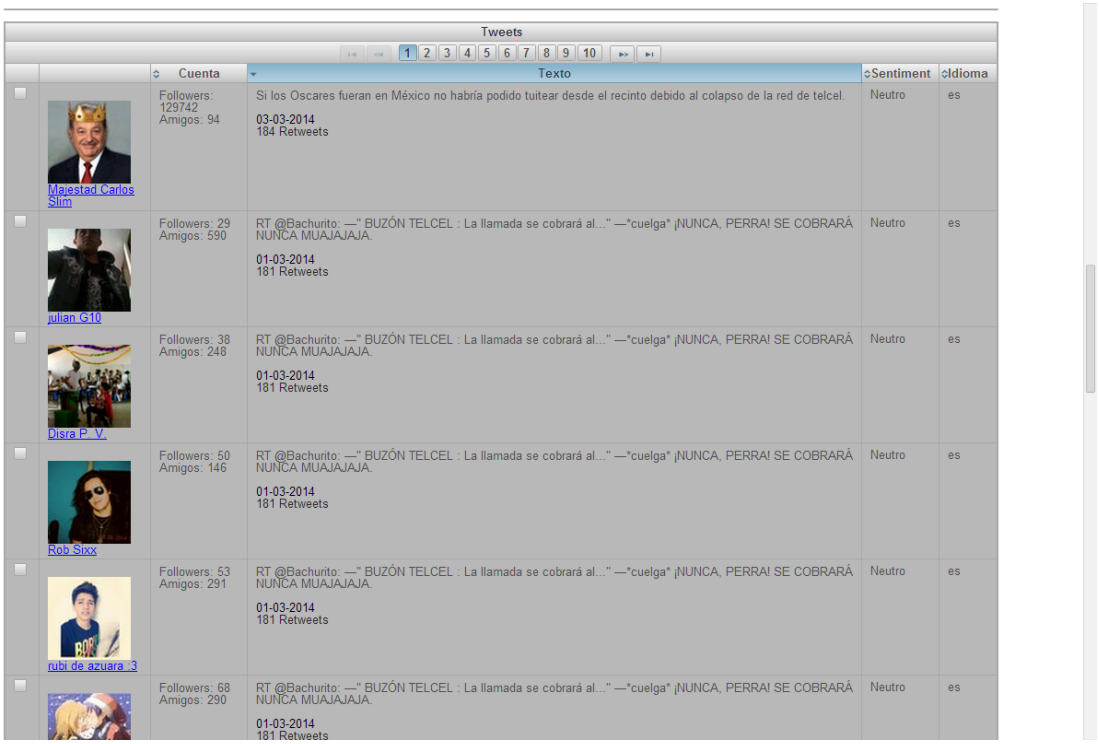


Figura 35: Pruebas funcionales módulo de mapas prueba 8

A continuación se muestra otra búsqueda del mismo modo, haciendo click en el mapa y capturando el término de búsqueda pero ahora activando el checkbox llamado “Comparar”.

Pruebas y análisis de resultados

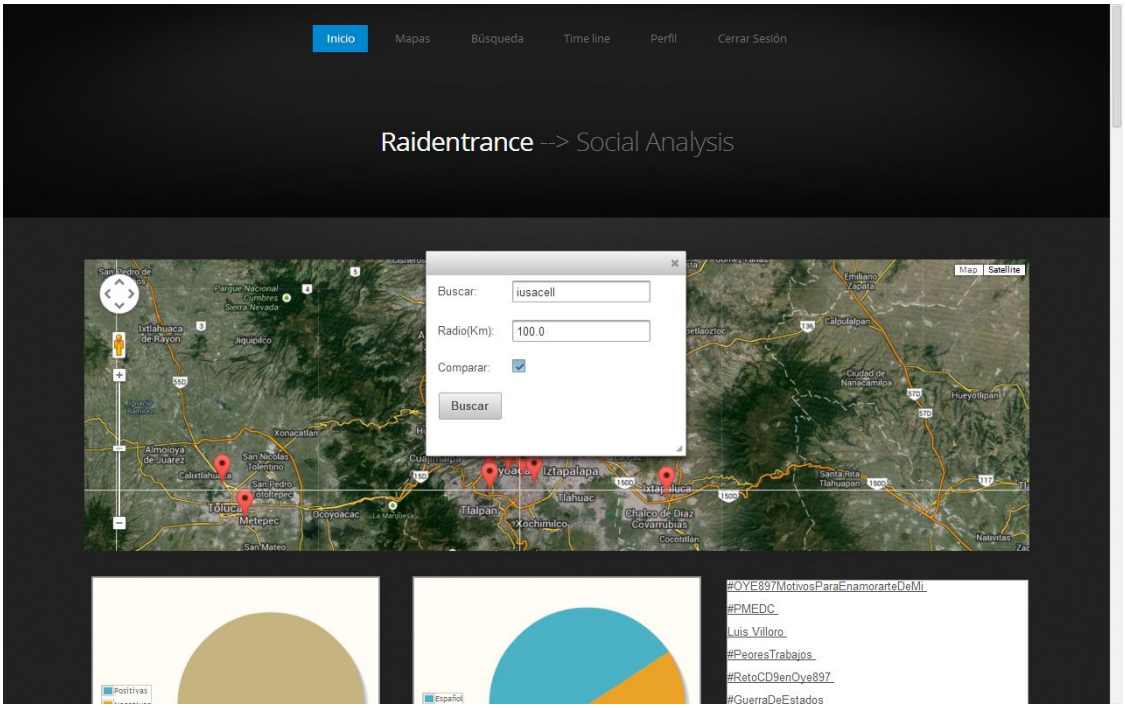


Figura 36: Pruebas funcionales módulo de mapas, prueba 9

Al hacer el botón “Buscar” se actualizará la pantalla quedando del siguiente modo, la ventana se partirá para mejor apreciación.

Pruebas y análisis de resultados

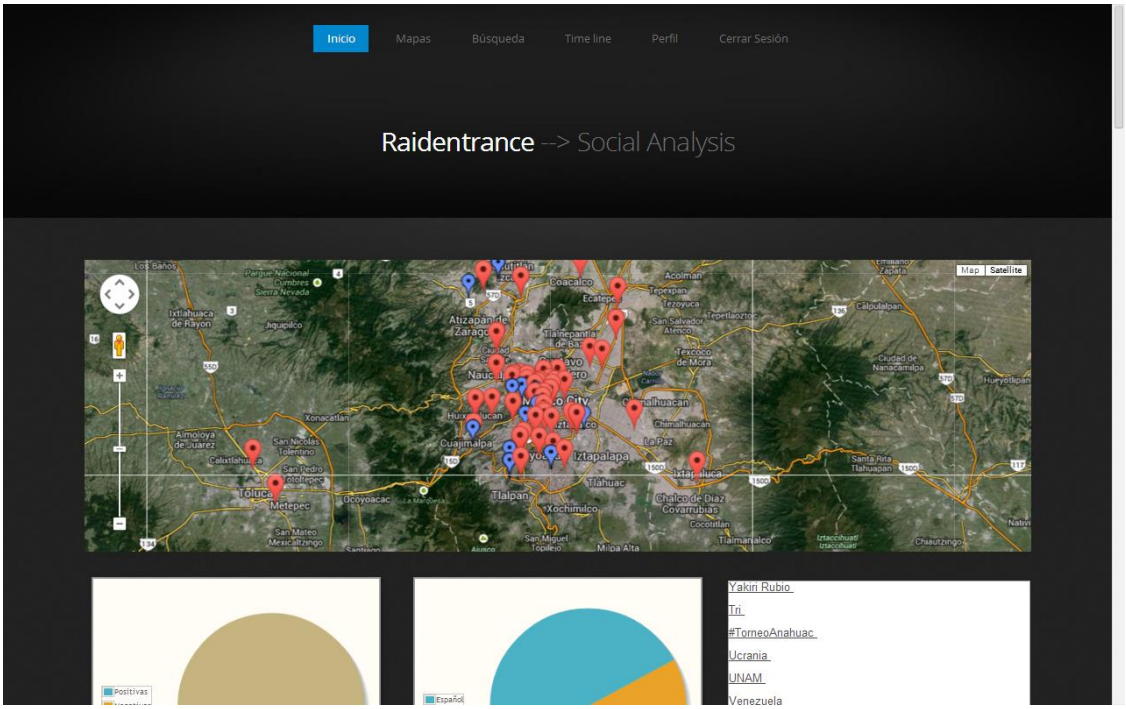


Figura 37: Pruebas funcionales módulo de mapas, prueba 10

Como se puede observar se presentan los puntos en el mapa con el nuevo término de búsqueda en otro color, permitiendo diferenciar la búsqueda actual de la anterior.

Figura 38: Pruebas funcionales módulo de mapas, prueba 11

Pruebas y análisis de resultados

Las gráficas de pie son actualizadas para mostrar los resultados de la nueva búsqueda.

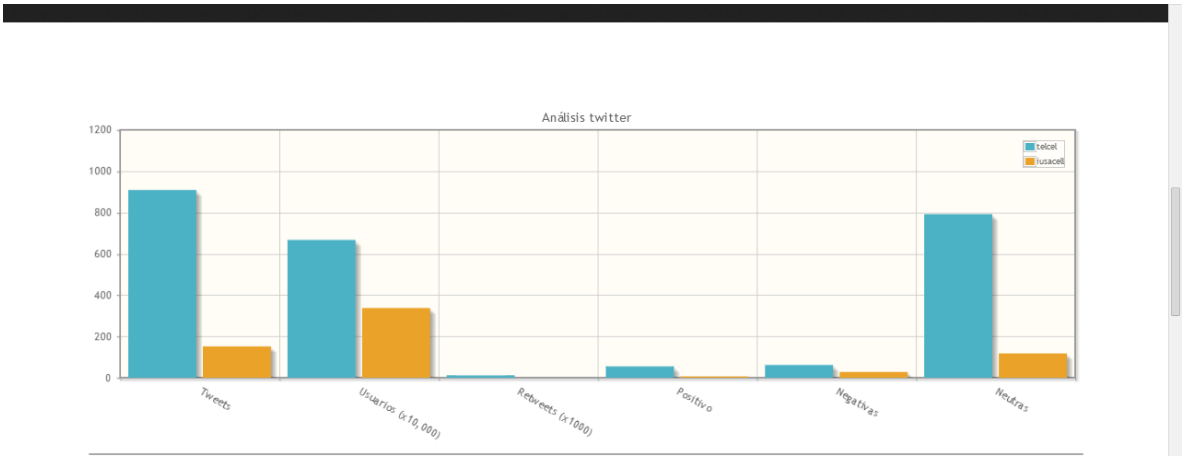


Figura 39: Pruebas funcionales módulo de mapas, prueba 12

En la gráfica análisis de Twitter se puede observar que se agregó una nueva serie con otro color que representa la nueva búsqueda, esto nos permite comparar la búsqueda actual contra la anterior, en esta se puede analizar el comportamiento de un término con respecto al otro.

Pruebas y análisis de resultados

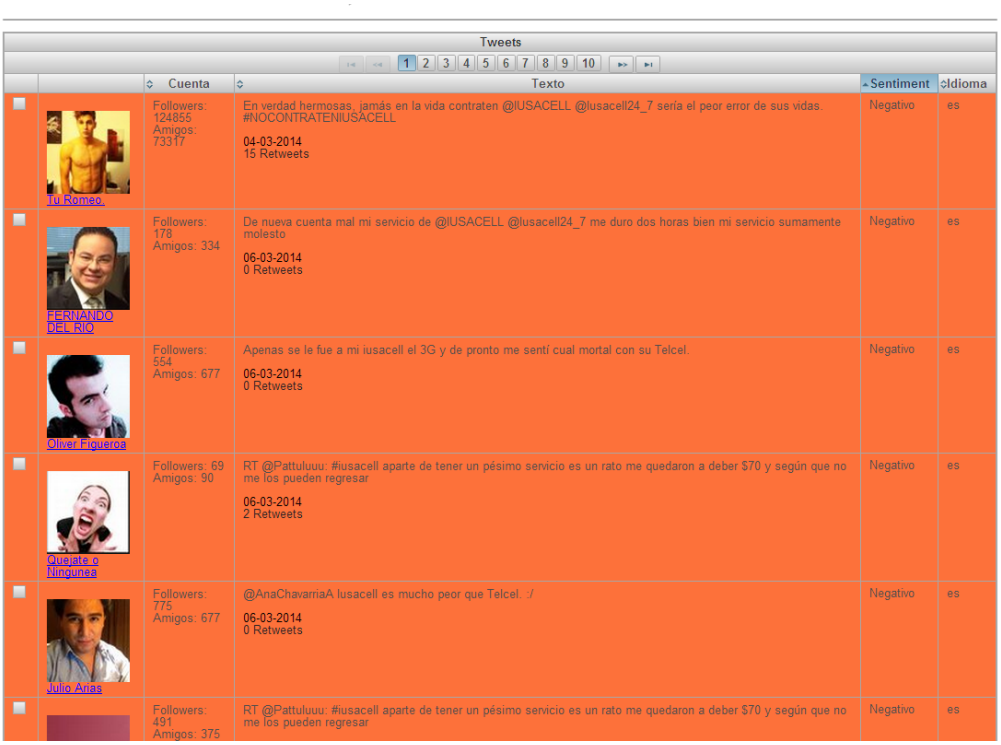


Figura 40: Pruebas funcionales módulo de mapas, prueba 13

Por último la tabla de resultados es actualizada para presentar los resultados de la nueva búsqueda permitiendo el comportamiento presentado anteriormente.

Este tipo de búsquedas nos permite analizar dos términos de búsqueda en una misma región o un término en diferentes regiones independientemente del lenguaje. Esto puede ser aplicado para los siguientes casos:

- Comparar el comportamiento de dos marcas siendo una la competencia de otra en una misma región o en diferentes regiones.
- Comparar el comportamiento de una marca en diferentes regiones.
- Comparar el comportamiento de un actor político con respecto a otro en una determinada región.
- Comparar el comportamiento de un actor político en diferentes regiones.

Pruebas y análisis de resultados

A continuación se presenta un ejemplo donde ejemplificará el soporte de inglés del sistema. Para esto se realiza una búsqueda sobre Huston con el término de búsqueda “Barack Obama”.

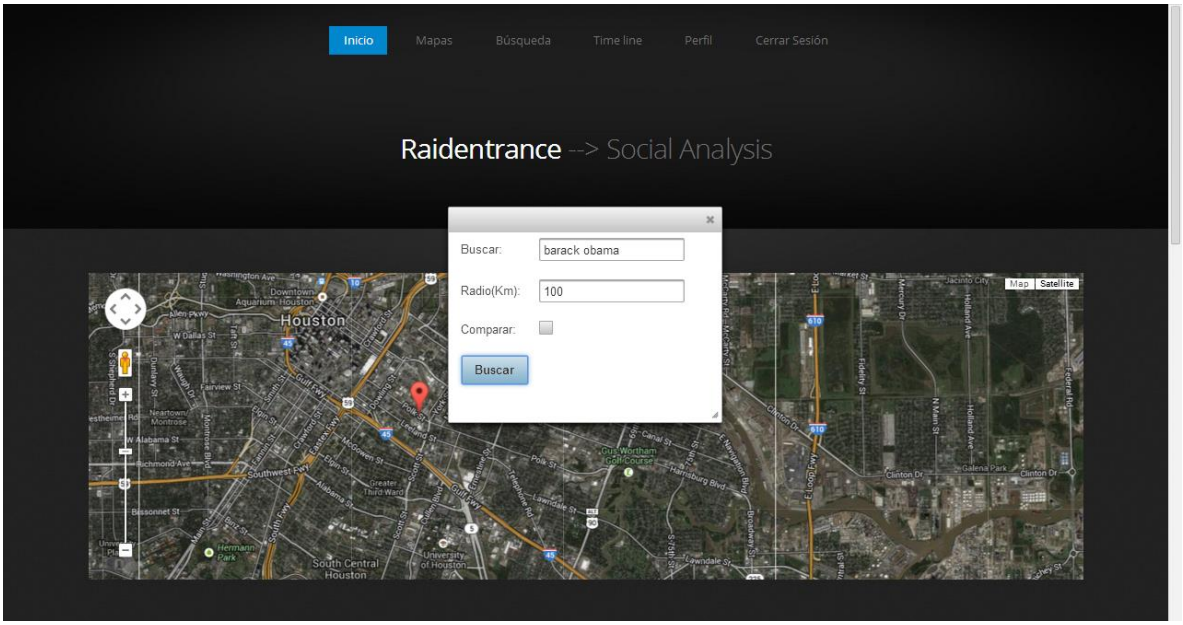


Figura 41: Pruebas funcionales módulo de mapas, prueba 14

Se capturó en el diálogo los siguientes valores:

- Buscar: Barack Obama
- Radio(Km): 100

Pruebas y análisis de resultados

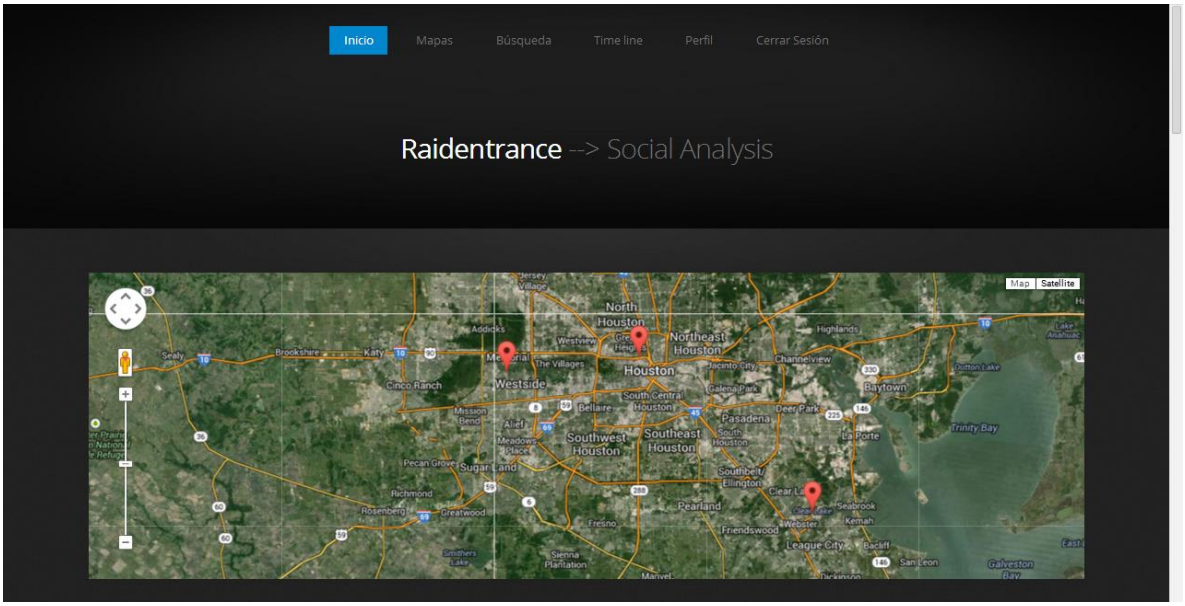


Figura 42: Pruebas funcionales módulo de mapas, prueba 15

Como se puede observar en esta región existe una cantidad menor de usuarios que tienen activa la función de localización en su dispositivo al momento de realizar publicaciones en Twitter.

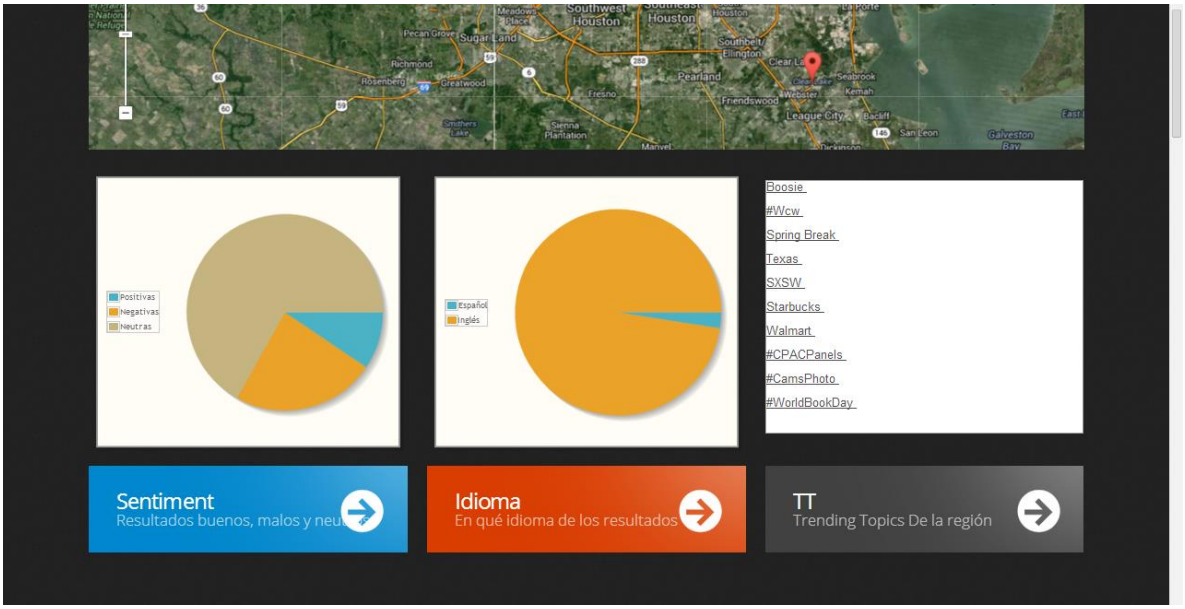


Figura 43: Pruebas funcionales módulo de mapas, prueba 16

Pruebas y análisis de resultados

En esta zona de la pantalla se pueden apreciar 3 segmentos en los cuales se presenta lo siguiente:

- Sentiment: En esta gráfica de pie se presentan los resultados positivos, negativos y neutros de los resultados obtenidos de la búsqueda realizada.
- Idioma: En esta gráfica de pie se presentan los idiomas encontrados en los resultados obtenidos, como se puede observar al realizar la búsqueda en Houston Texas los resultados son en su mayoría en idioma inglés.
- TT: En este segmento se muestran los Trendig Topics de la región sobre la cual se realizó la búsqueda, para este caso se puede observar que están en idioma inglés ya que son los TT de Houston Texas.

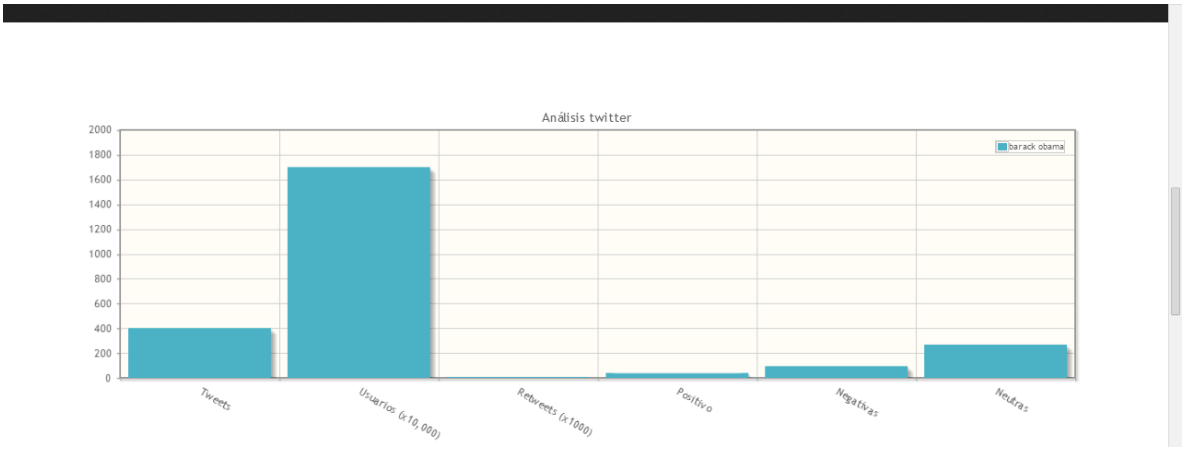


Figura 44: Pruebas funcionales módulo de mapas, prueba 17

En la gráfica Análisis de Twitter se presentan los siguientes resultados:

- Tuits: Número de tuits (publicaciones) encontradas en la búsqueda.

Pruebas y análisis de resultados

- Usuarios(x10,000): Este número representa al número de usuarios a los que llegaron los tuits(publicaciones) relacionados con el termino de búsqueda.
- Retuits(x1000): Número de retuits totales encontrados en todos los resultados obtenidos en la búsqueda.
- Positivo: Número de resultados positivos encontrados en la búsqueda.
- Negativo: Número de resultados negativos encontrados en la búsqueda.
- Neutro: Número de resultados neutros encontrados en la búsqueda.

A continuación se presenta la tabla ordenada de modo que muestre primero los resultados negativos.

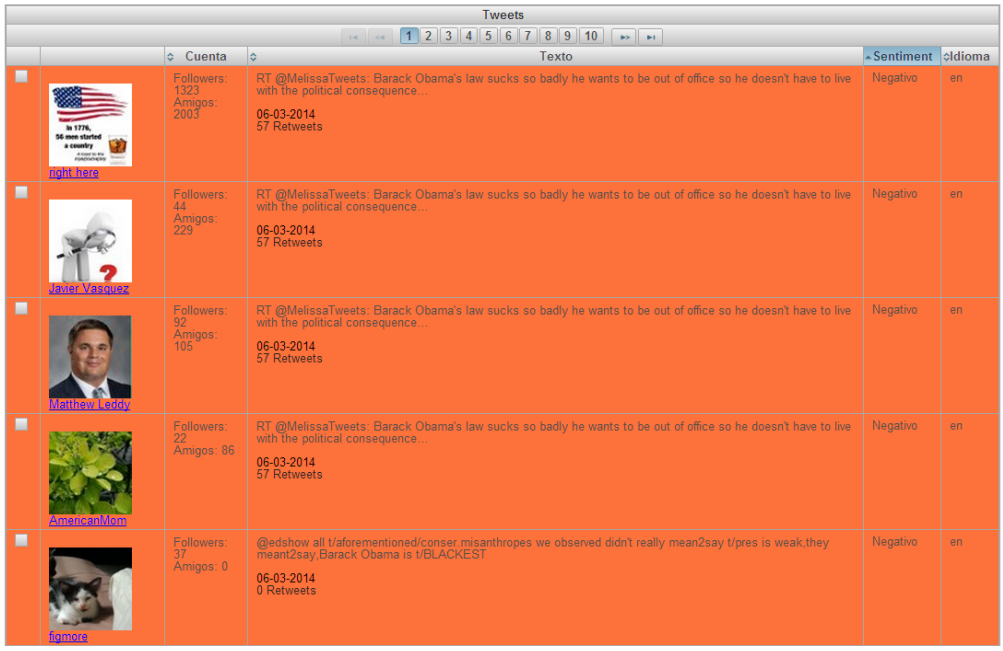


Figura 45: Pruebas funcionales módulo de mapas, prueba 18

Pruebas y análisis de resultados

Por último se realiza una búsqueda en la Ciudad de México con el mismo término de búsqueda para así comparar el comportamiento del término “Barack Obama” en México con Houston Texas.

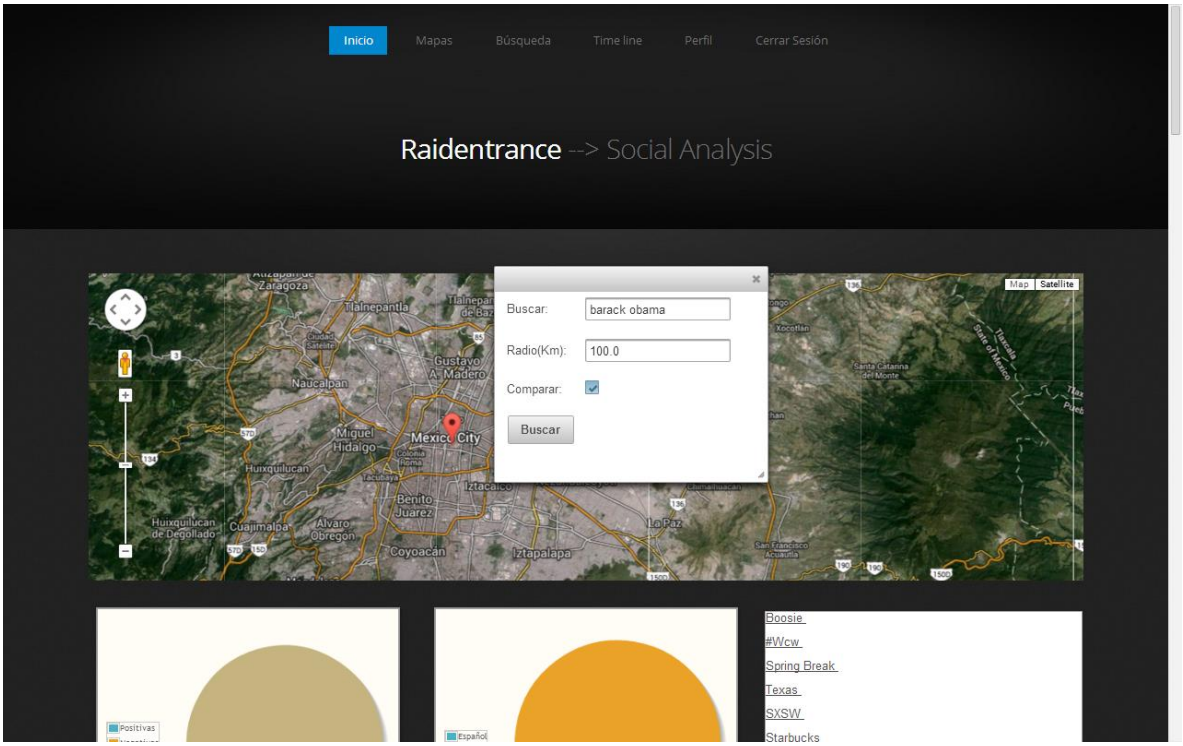


Figura 46: Pruebas funcionales módulo de mapas, prueba 19

A continuación se presentan los puntos desde los cuales se realizaron las publicaciones en la Ciudad de México.

Pruebas y análisis de resultados

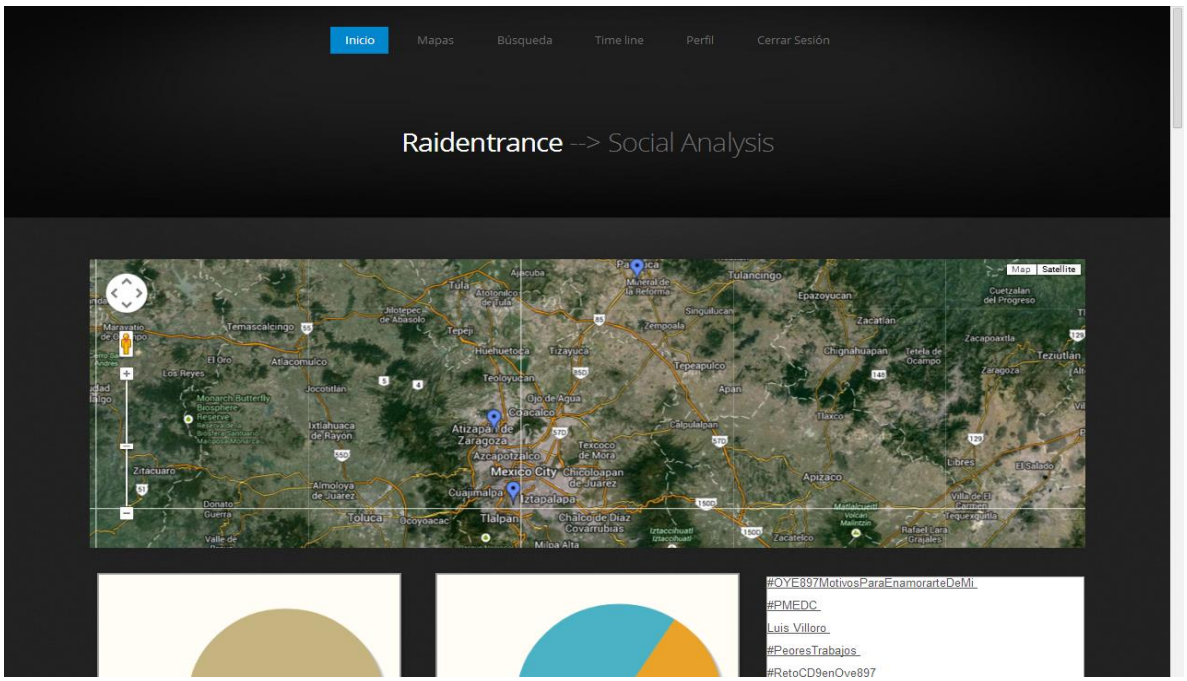


Figura 47: Pruebas funcionales módulo de mapas, prueba 20

Como se puede observar el Sentiment el idioma y los Trending Topics en la Ciudad de México varían con respecto a la búsqueda inicial en la Ciudad de Houston.

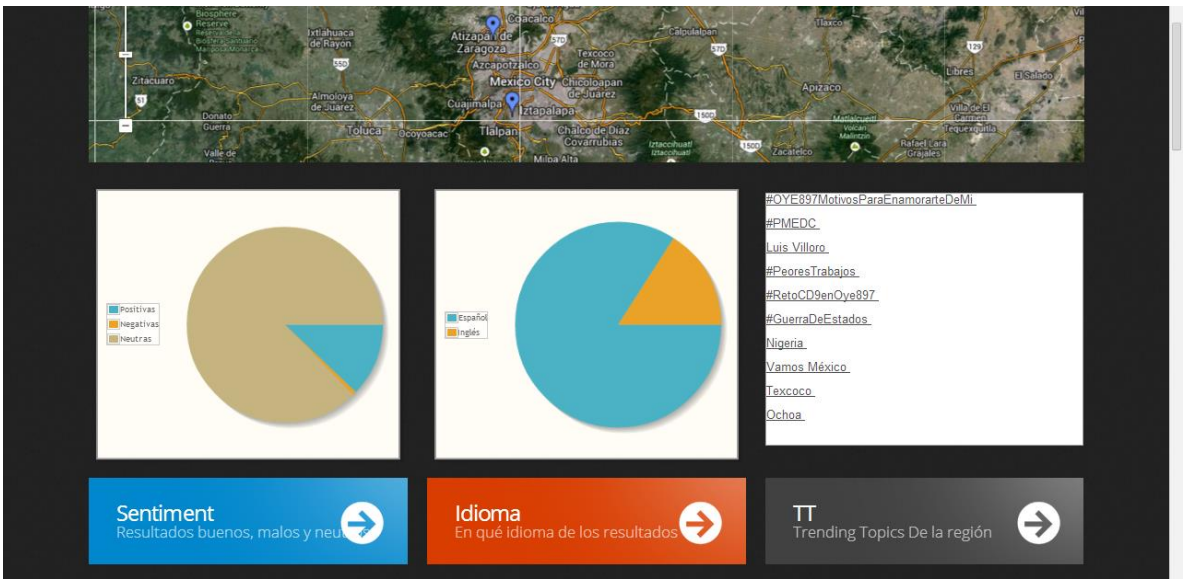


Figura 48: Pruebas funcionales módulo de mapas, prueba 21

Pruebas y análisis de resultados

Como se puede observar en la gráfica de análisis de Twitter el impacto en México es más positivo que negativo y en Houston es más negativo que positivo.

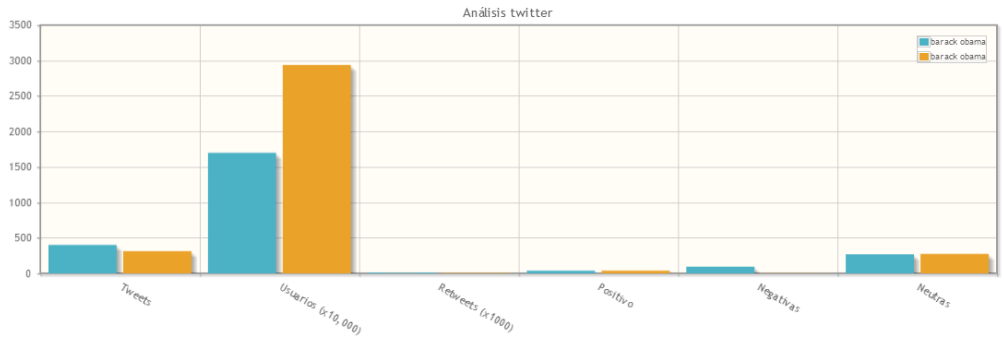


Figura 49: Pruebas funcionales módulo de mapas, prueba 22

Para mostrar la gráfica de una manera más clara se cuenta con la posibilidad de hacer zoom en ciertas zonas, a continuación se muestra el zoom para la parte del Sentimiento.

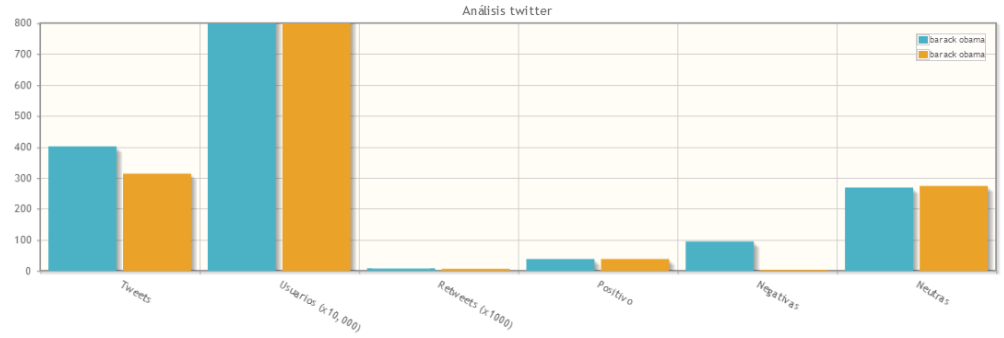


Figura 50: Pruebas funcionales módulo de mapas, prueba 23

Pruebas funcionales del módulo de Búsqueda:

El módulo de búsqueda tiene como objetivo realizar búsquedas generales sin importar la zona en la que se realizó la publicación. Al entrar al módulo se presenta la siguiente pantalla:

Pruebas y análisis de resultados

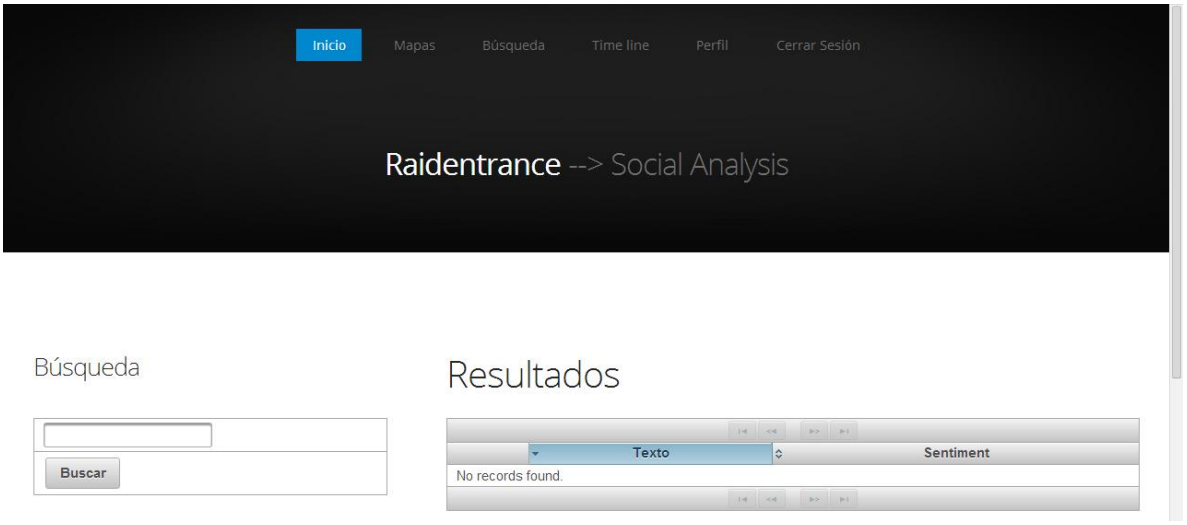


Figura 51: Pruebas funcionales módulo de búsqueda general, prueba 24

A continuación se escribe el término de búsqueda “Barack Obama” y se hace click en el botón Buscar.

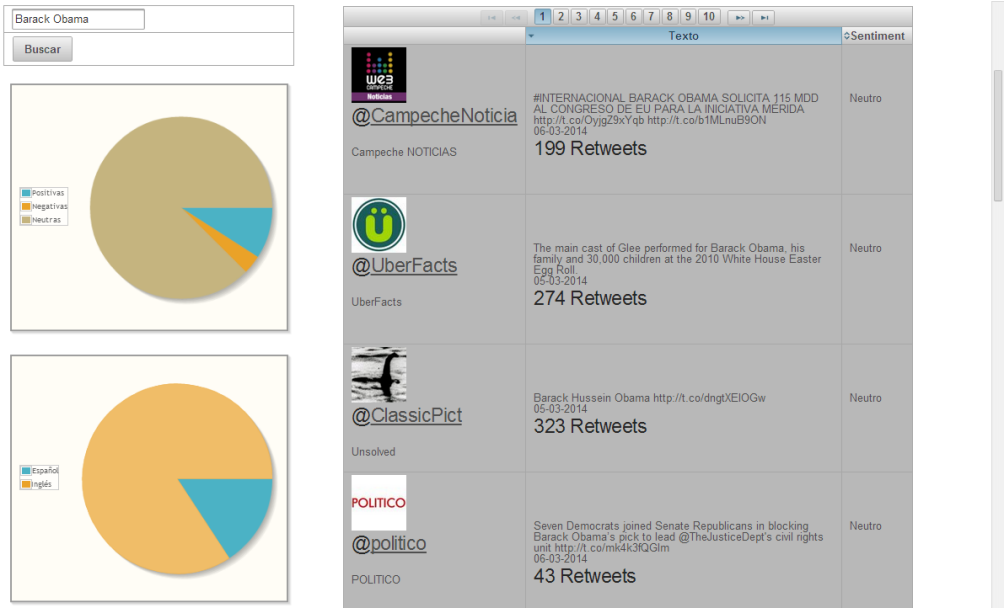


Figura 52: Pruebas funcionales módulo de búsqueda general, prueba 25

Como se puede observar, en la parte central se presenta una tabla de datos con los resultados obtenidos en Twitter, estos resultados pueden ser ordenados por el sentiment o la cantidad de retuits de la publicación y son marcados como

verde, gris o rojo dependiendo del sentiment. En la parte izquierda se presentan las gráficas que representan el sentimiento y el idioma, estas gráficas se generan con base en los resultados obtenidos en la búsqueda. A continuación se presenta la tabla de datos ordenada por el sentimiento negativo.

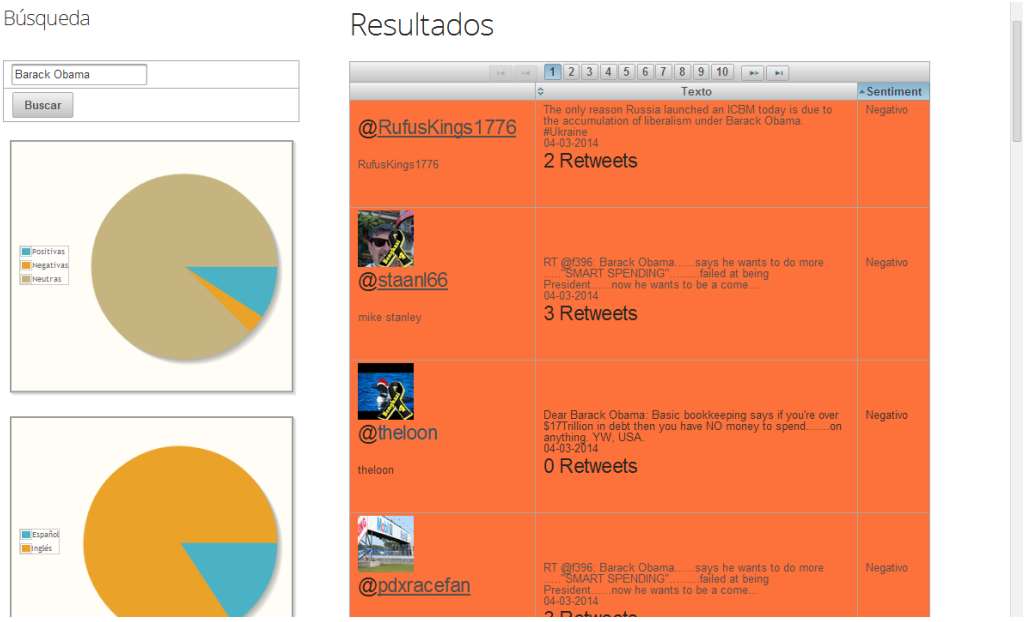


Figura 53: Pruebas funcionales módulo de búsqueda general, prueba 26

Pruebas funcionales del módulo de Time line:

Dentro de este módulo se pueden observar las publicaciones de los usuarios a los que sigue la cuenta con la que se inició sesión, además de que se cuenta con la posibilidad de realizar publicaciones y ver información de la cuenta.

Pruebas y análisis de resultados

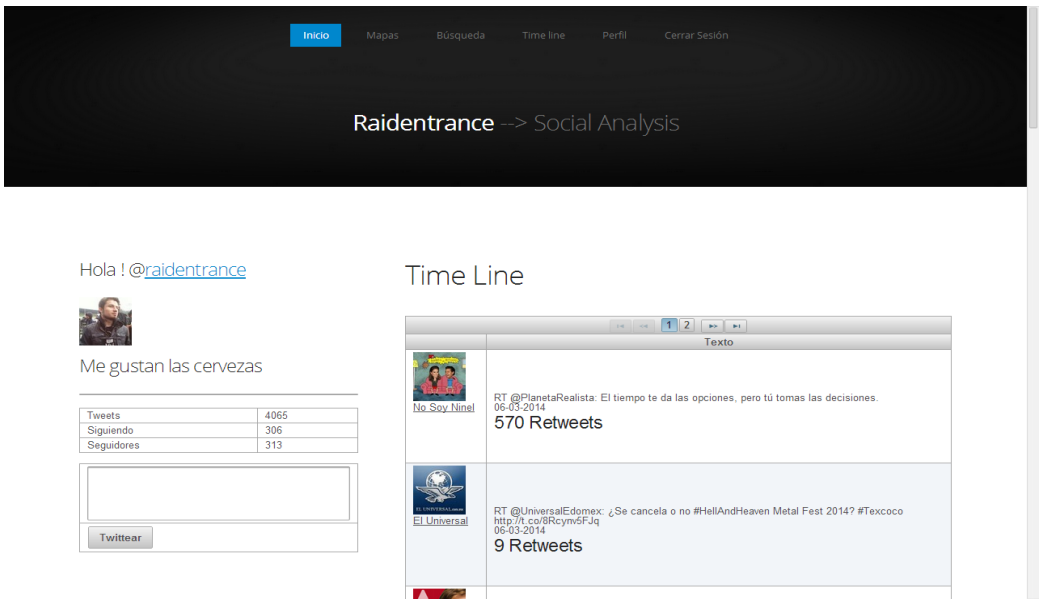


Figura 54: Pruebas funcionales módulo de perfiles, prueba 27

En la parte izquierda se presenta lo siguiente:

- Texto “Hola!” y el nombre del usuario que inició sesión con link a la cuenta oficial de Twitter.
- La descripción del perfil de acuerdo como lo haya escrito el usuario.
- Número de publicaciones del usuario.
- Número de seguidores del usuario.
- Número de amigos del usuario.
- Área de texto para escribir una publicación nueva.
- Botón Twittear.

Del lado central se pueden observar las publicaciones de los usuarios a los que sigue la cuenta, esto en una tabla de datos que cuenta con paginadores para ver más publicaciones mostrando lo siguiente:

- Imagen de perfil.

Pruebas y análisis de resultados

- Nombre de usuario.
- Publicación.
- Fecha de publicación.
- Número de retuits de la publicación.

Pruebas funcionales del módulo de Perfil:

Dentro de este módulo se tiene la posibilidad de ver las publicaciones hechas por la cuenta que inició sesión, así como de ver las publicaciones favoritas y a las que ha dado retuit como se muestra en la siguiente pantalla.

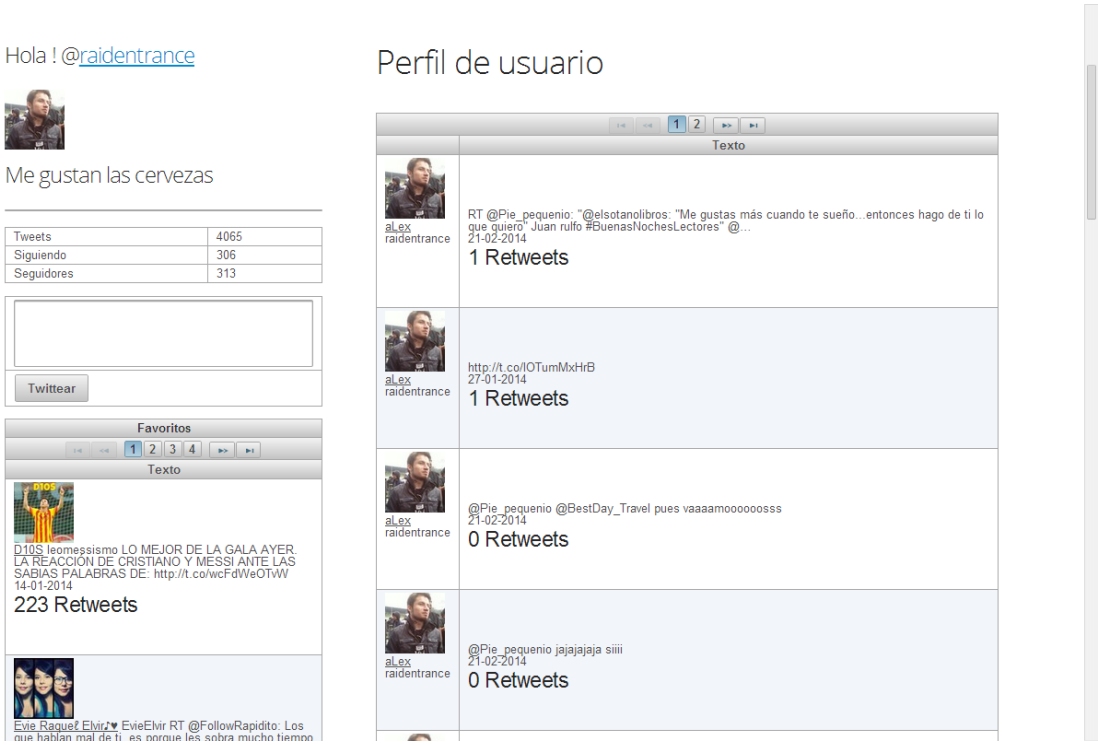


Figura 55: Pruebas funcionales módulo de perfiles, prueba 28

Dentro de esta sección se tiene la posibilidad de lo siguiente:

- Ver la siguiente información de la cuenta: Número de publicaciones, número de amigos, número de seguidores y datos de la cuenta.

Pruebas y análisis de resultados

- Área de texto para realizar publicaciones nuevas.
- Área de favoritos en la cual se muestran las publicaciones favoritas del usuario.
- Área de publicaciones personales del usuario.

Capítulo 6

6. Despliegue del sistema

Despliegue del sistema

6.1. Despliegue

El sistema Social Analysis al ser una aplicación web bajo la arquitectura Java EE requiere para su despliegue un servidor que cuente con lo siguiente:

6.1.1. Apache Tomcat

El sistema requiere de un contenedor de servlets para su despliegue, se tomó la decisión de utilizar Apache tomcat ya que funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o Weblogic. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

La estructura de directorios de la instalación de Tomcat es la siguiente:

- Bin - Contiene los scripts y códigos requeridos para que se ejecute el servidor. Es el principal utilizado para arrancar y parar tomcat.

- conf - Contiene los ficheros necesarios para configurar y administrar Tomcat, es lo primero que lee al arrancar y se basa en su contenido para cargar el resto. Contiene estos ficheros imprescindibles:
  - catalina.policy - Las políticas de seguridad, prevalece este fichero sobre el java.policy que viene con la instalación de Java. No obstante sólo se usa cuando tomcat se ejecuta con el parámetro –security.
  - catalina.properties - Contiene un listado de los paquetes Java que no pueden ser anulados por la programación, ya que podría ser un agujero de seguridad.
  - context.xml - el contexto que es usado en todas las aplicaciones web, normalmente se usa para configurar donde acceder al web.xml.
  - logging.properties - La configuración por defecto para logging en Tomcat.
  - server.xml - el principal fichero de configuración de Tomcat, aquí configuramos los puertos que arranca este servicio y muchísima información más.
  - tomcat-users.xml - Seguridad para el acceso a aplicaciones de administración. Por defecto viene todo comentado para que no se pueda acceder.
  - web.xml - Fichero que es utilizado por todas las aplicaciones web, se configuran los timeout de sesiones y los ficheros principales como index.htmllogs - logs de Catalina y de las aplicaciones.
- Lib - Contiene todos los JAR (Java ARchive) que usa el servidor.

Despliegue del sistema

- Logs - El directorio usado para logging, es decir las bitácoras del server.
- work - Ficheros temporales y con los que trabajamos. Es el usado por los JSP para compilarse, convertirse en un Servlet, que es el miniprograma que está arrancado para cuando un usuario haga una petición específica, el servlet que corresponde sirve comprueba y manda la información al navegador.
- Webapps: En esta carpeta se encuentran las aplicaciones desplegadas en Apache Tomcat.

  Ejemplo de una aplicación desplegada en Apache Tomcat:

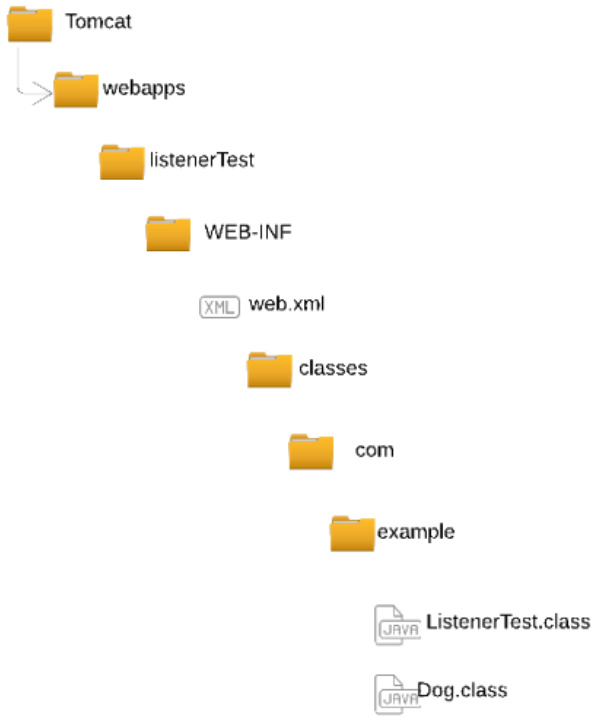


Figura 56: Estructura de directories Apache Tomcat

6.1.2. MySQL

Sistema manejador de base de datos Open source que nos permite almacenar la información del sistema Social Analysis, a continuación se presentan algunas características del manejador:

- MySQL software es Open Source
- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de Sistemas Operativos
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet
- El software MySQL usa la licencia GPL

6.1.3. Jelastic

La implementación del sistema se realizará en el servicio jelastic, este es un servicio de cómputo en la nube de tipo PAAS (Platform as a service) dicho servicio proporciona lo siguiente:

Despliegue del sistema

- Un balanceador de carga que permite crear entornos con más de una instancia del contenedor de Servlets.
- Uno o muchos contenedores de Servlets.
- Un servidor de MySQL para el almacenamiento de la información.

Se tomó la decisión de utilizar PAAS para desplegar el sistema ya que nos brinda las siguientes ventajas:

- Instalación del sistema en minutos: El entorno se configura a través de herramientas simples y prototipos ya hechos que permiten crear complejos entornos en muy poco tiempo.
- No se requiere instalación ni configuración: No se requiere la instalación de ningún tipo de software en el servidor ya que todo se realiza a través de herramientas en línea.
- No se requieren APIs adicionales: No se requiere agregar código a la aplicación para que funcione en este tipo de entorno.
- Amplia gama de paquetes de software: Los servicios de la nube proporcionan diversas opciones de Servidores de aplicaciones, servidores de bases de datos o lenguajes de programación.
- Fácil configuración de ambiente de cluster: Se puede seleccionar fácilmente el número de instancias de servidores de aplicaciones y la memoria con la que contará cada uno de ellos.

Despliegue del sistema

- Fácil despliegue de aplicaciones: El despliegue de las aplicaciones se realiza a través de herramientas con lo cual sólo se necesita subir el archivo war y realizar las configuraciones necesarias de la aplicación.
- Se puede desplegar el proyecto desde GIT o Subversion: Si se cuenta con un servidor de GIT o subversión se pueden utilizar plugins para desplegar la aplicación en la nube.
- Fácil de escalar: Se cuenta con herramientas que te permiten subir o bajar la memoria de las instancias bajo demanda.
- Fácil de administrar: Se cuenta con diversos cuadros de mando para realizar distintas tareas para la administración del entorno.

Despliegue del sistema

6.2. Despliegue de Social Analysis

El primer paso para desplegar el sistema Social Analysis es construir el archivo .war del proyecto, para esto se deben se seguir los siguientes pasos:

- Instalar y configurar Maven.
- Abrir una consola de MS-DOS.
- Posicionarse en la carpeta en la que se encuentra el proyecto.
- Ejecutar el comando: `mvn clean install`.

Con esto se creará en la carpeta del proyecto una nueva carpeta llamada `target`. En esta carpeta se creará un archivo .war que representa la aplicación Social Analysis.

Una vez creada una cuenta en Jelastic el servicio brinda un tablero de trabajo en el cual se tiene la posibilidad de crear un entorno de trabajo para el despliegue, un administrador de despliegues, un monitor de los despliegues entre otras herramientas. El primer paso es crear un ambiente de trabajo en el que se va a desplegar la aplicación. Para hacer esto se selecciona la opción “Create environment” la cual nos presentará lo siguiente.

Despliegue del sistema

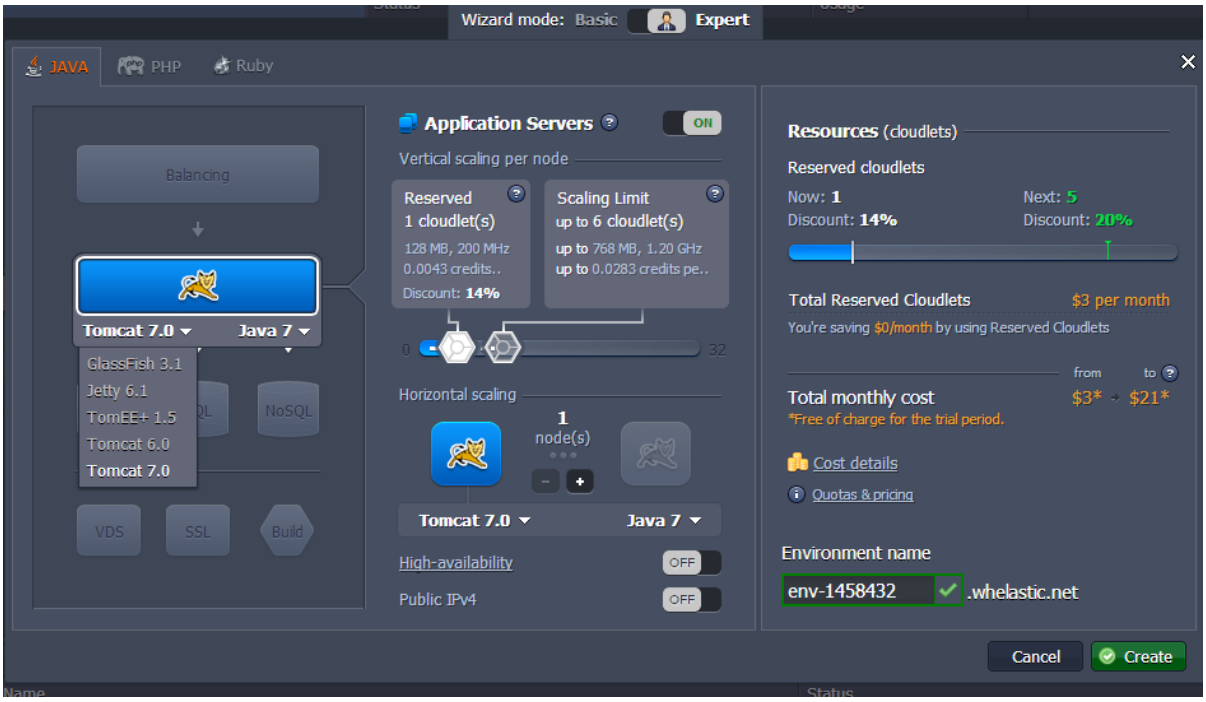


Figura 57: Consola de Jelastic para la generación de ambientes

El creador de entornos está separado en 3 secciones a través de las cuales se realizan las configuraciones correspondientes.

Primera sección

Se presenta el entorno de trabajo que se desea utilizar para realizar el despliegue, Es en este lugar en el que se seleccionarán aspectos como el lenguaje de programación, contenedor web, la base de datos, el balanceador de carga, servidores dedicados, SSL y la herramienta que se utilizará para la construcción del proyecto. Para el sistema Social Analysis se tomó el siguiente entorno de trabajo:

Figura 58: Generación de ambiente utilizando Jelastic

Despliegue del sistema

Se tomó Java como lenguaje de programación, un balanceador de carga Nginx, el contenedor Apache Tomcat, MySQL como manejador de base de datos y Maven como herramienta de construcción del proyecto.

Segunda sección

En la segunda sección se presentan las configuraciones de cada uno de los aspectos del despliegue seleccionados, permitiendo seleccionar aspectos como el procesador y la memoria que se le brindarán a cada uno.

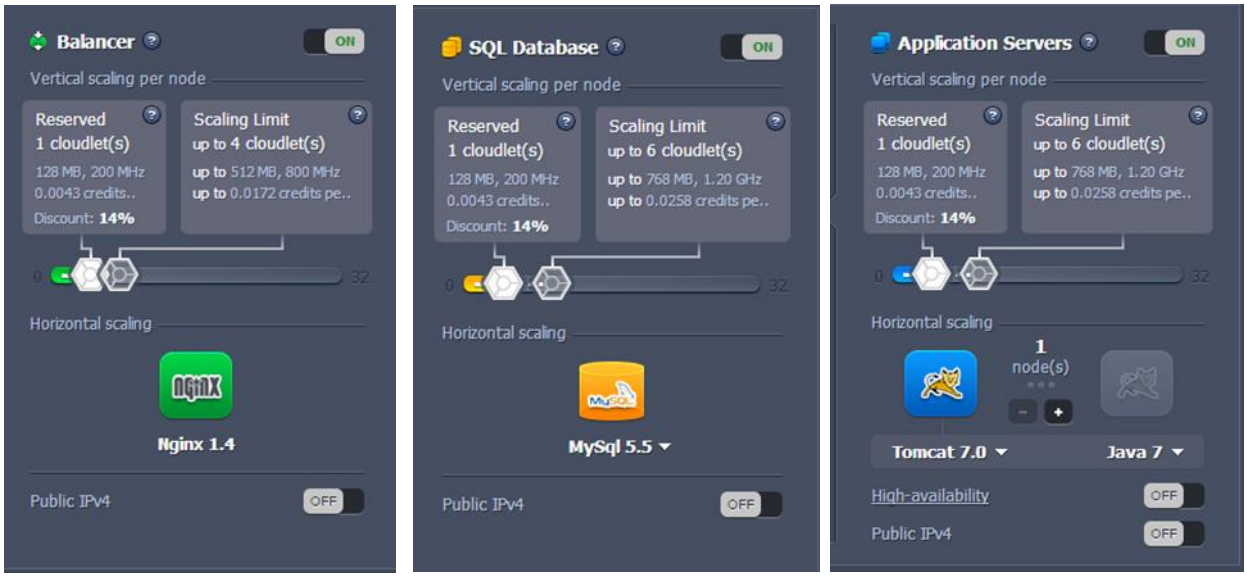


Figura 59: Manejo de memoria utilizando Jelastic

Despliegue del sistema

En esta sección se presenta el precio por mes del entorno de desarrollo seleccionado con el detalle desglosado y la dirección del entorno.

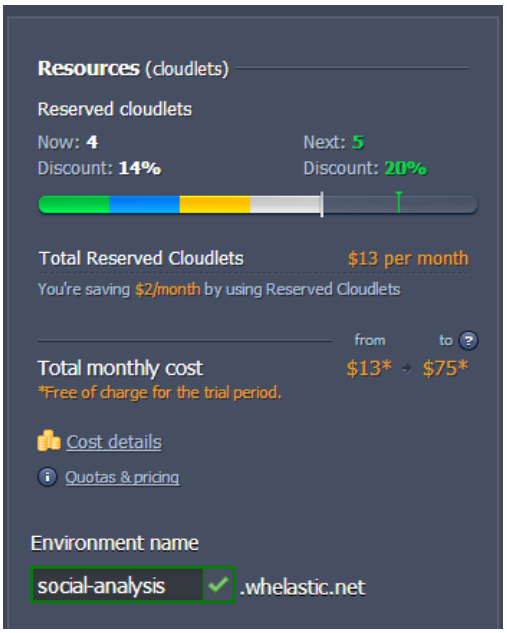


Figura 60: Precio por mes en Jelastic

Una vez configurado el entorno de trabajo se hace click en “Create” y ya se tiene un entorno de despliegue para el sistema, entre 5 y 10 minutos después se presenta el entorno como se muestra a continuación.

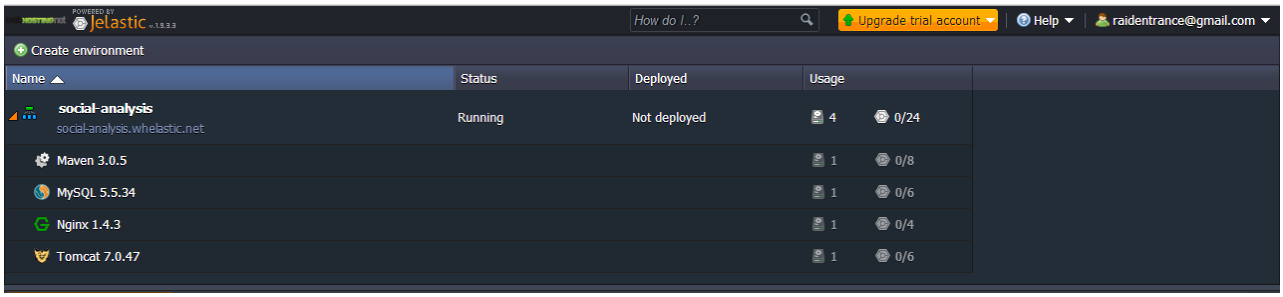


Figura 61: Tiempo para la creación del entorno

Para el Servidor de MySQL y el de Tomcat se nos brinda un usuario y un password para entrar a la consola de Tomcat y a un PhpMyAdmin de Mysql. El siguiente paso para hacer el despliegue es entrar al PhpMyAdmin con las

credenciales Jelastic envía por correo y crear la base de datos. Para esto se crea una bsae de datos nueva se selecciona en la pestaña importar y se toma el archivo .sql que contiene la base de datos y se importa quedando como se muestra en la imágen.

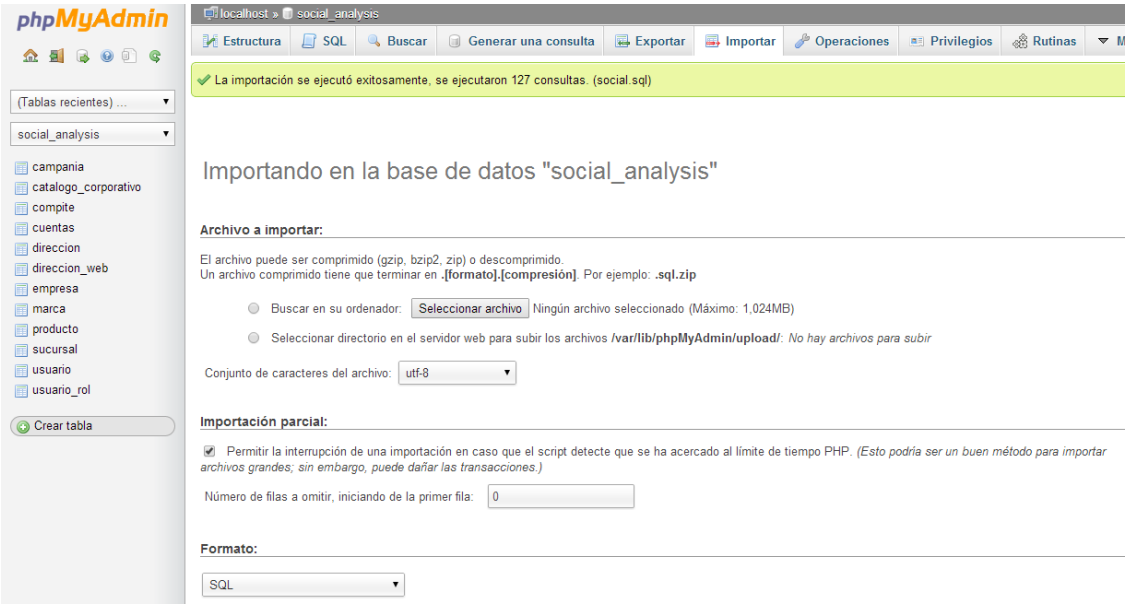


Figura 62: Importación de la base de datos a la nube de Jelastic

Por último para subir la aplicación se selecciona en la pestaña “Deployment manager” el botón “Upload” y se selecciona el archivo .war a desplegar se escribe un comentario sobre el despliegue y se selecciona el botón “Upload”.

Despliegue del sistema

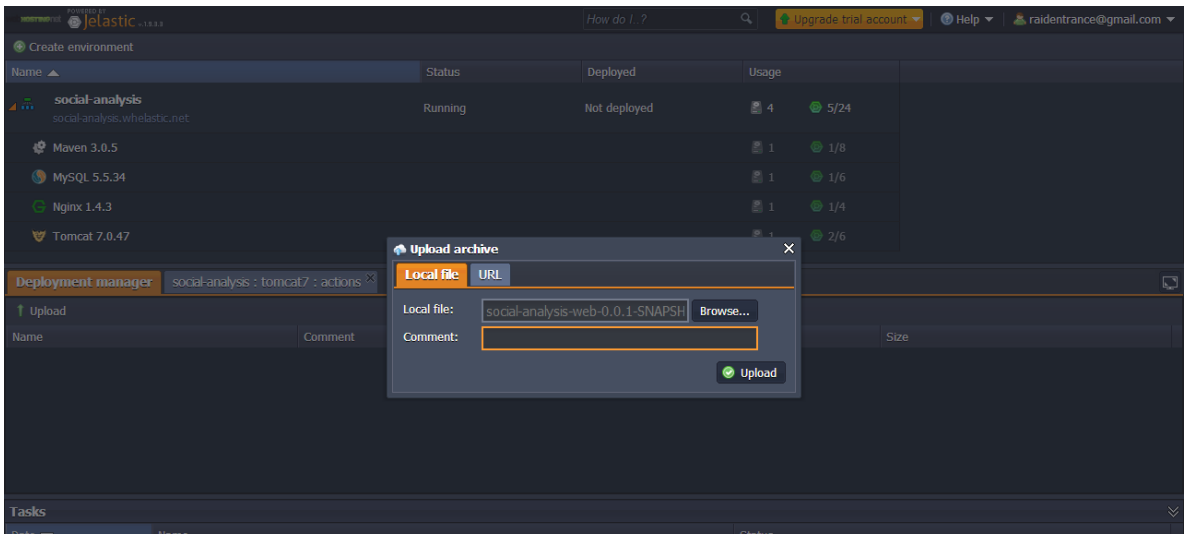


Figura 63: Agregar un recurso al servidor de Jelastic.

Este proceso puede tardar unos minutos, una vez terminado se selecciona el botón deploy social-analysis y se escribe el nombre de la aplicación, para este caso será “social-analysis”.

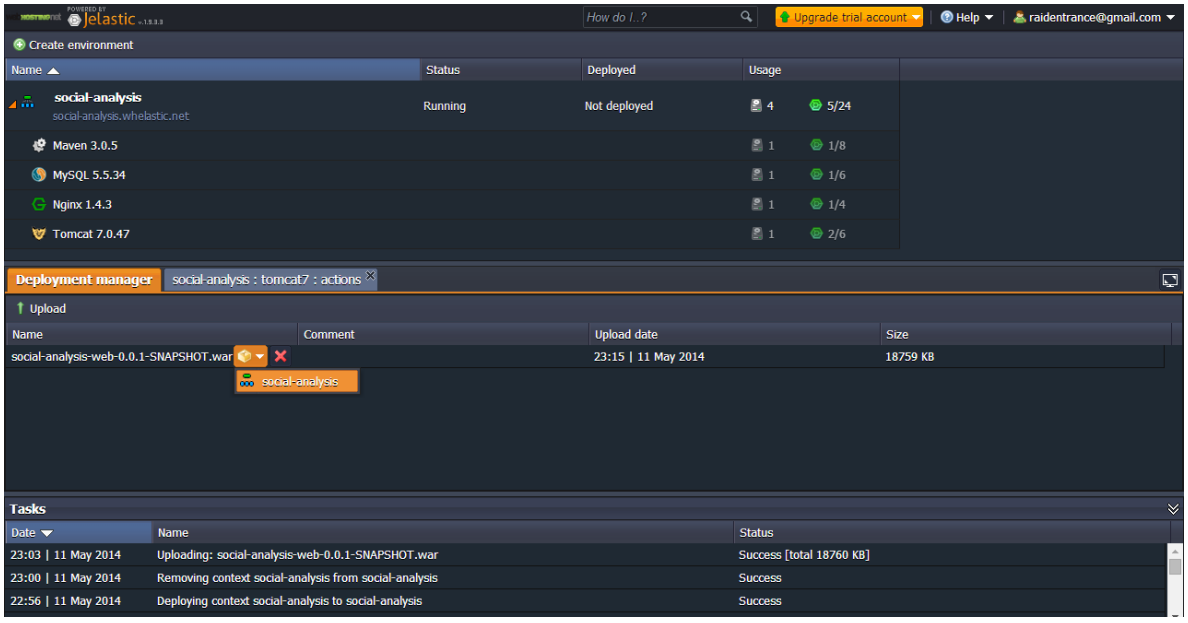


Figura 64: Despliegue de la aplicación utilizando la consola de Jelastic

Una vez concluido este proceso simplemente se oprime el botón “Open in browser” para ver la aplicación desplegada.

Despliegue del sistema

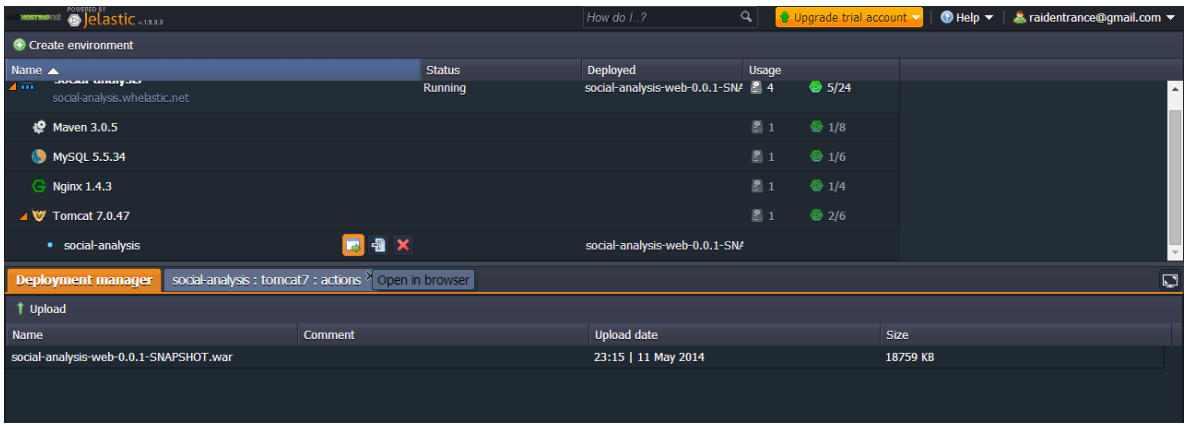


Figura 65: Abrir la aplicación desde la consola de Jelastic

Con esto queda concluido el proceso de despliegue, para poder observar la aplicación simplemente se tiene que hacer click en el link http://social-analysis.whelastic.net/social-analysis/mapas/.

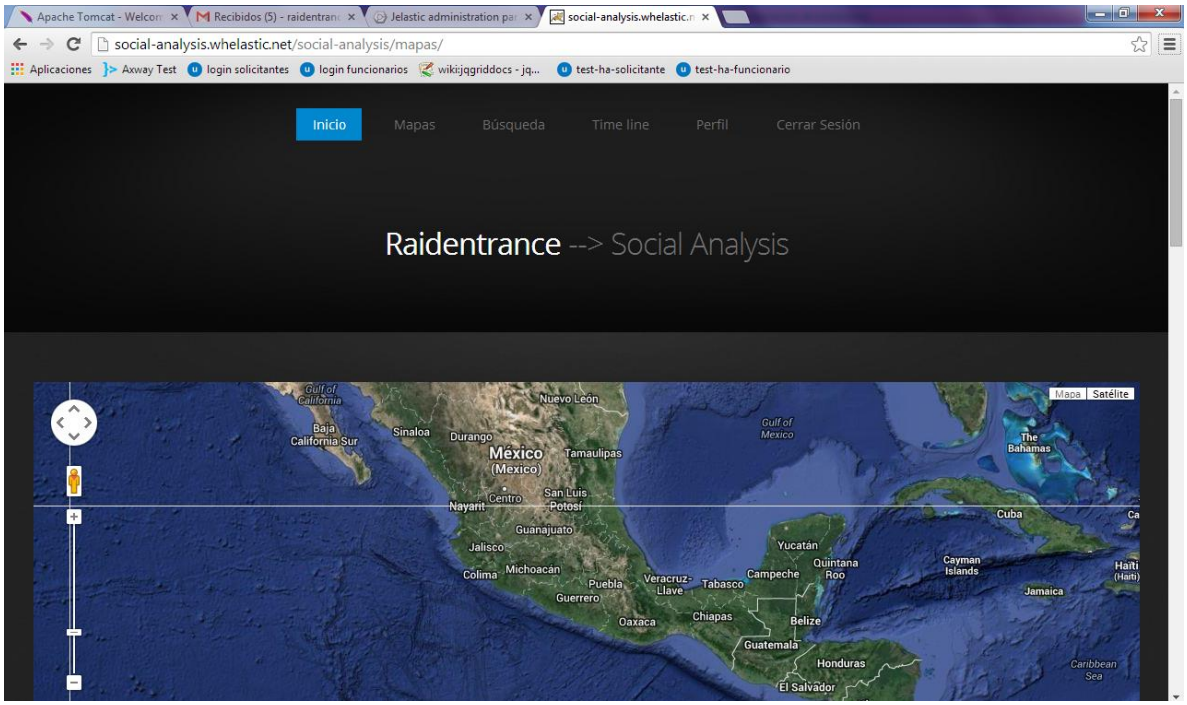


Figura 66: Aplicación desplegada en la web

Capítulo 7

7. Conclusiones

Conclusiones

El objetivo fundamental de esta tesis fue el desarrollo de un método a través del cual se permitiera clasificar la información proveniente de la red social Twitter, así como el desarrollo de una herramienta que, a través de ese método fuera capaz de representar el estado de un acontecimiento en diferentes partes del mundo.

Así pues, la aportación principal de este trabajo representa el análisis, diseño, desarrollo e implementación de un sistema que a través del uso de algoritmos de análisis de textos fue capaz de clasificar de la información proveniente de Twitter en diferentes partes del mundo.

Este trabajo fue desarrollado a través del uso del lenguaje de programación Java haciendo uso de los frameworks más nuevos y representativos del lenguaje, así como de protocolos estándares para el desarrollo de aplicaciones empresariales tales como HTTP y entandares como RESTful para el desarrollo de los servicios web. Todo el desarrollo fue basado en patrones de diseño los cuales permiten que la aplicación se adapte a estándares internacionales, sea fácilmente expandible y adaptable a diferentes entornos.

Este tema de tesis toca diversos temas de Ingeniería en computación tales como bases de datos, redes, análisis de textos, computo en la nube, ingeniería de software, sistemas operativos entre otros.

El sistema Social Analysis es solo el inicio de una plataforma, los siguientes pasos es la expansión de esta herramienta a otras redes sociales tales como Facebook, Instagram y Foursquare. Esta plataforma tiene diversas aplicaciones en

diferentes rubros tales como la política, el marketing y la seguridad, lo cual representa una ventaja para fines comerciales.

El objetivo a corto plazo es la distribución de este sistema haciendo uso de una estrategia comercial llamada SAS (Software as a service), en este modelo se pretende la distribución de un software como un servicio a través de un pago mensual, de tal modo que un cliente pague un precio por el uso de la herramienta para una cantidad n de usuarios. Existen diversas empresas que distribuyen el software de este modo e incluso existen tiendas comerciales en la nube donde se puede realizar la distribución del sistema de un modo simple y rápido.

El desarrollo del sistema Social Analysis representa una muestra clara de la aplicación que tienen los algoritmos de análisis de textos, los algoritmos utilizados en este sistema fueron análisis sentimental, extracción de entidades y reconocimiento de lenguaje.

Glosario

Glosario de términos

Microblogging: El microblogging, también conocido como nanoblogging, es un servicio que permite a sus usuarios enviar y publicar mensajes breves, generalmente sólo de texto. Las opciones para el envío de los mensajes varían desde sitios web, a través de SMS, mensajería instantánea o aplicaciones ad hoc.

Grafo: En matemáticas y ciencias de la computación, un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto. Son objeto de estudio de la teoría de grafos.

Hashtag: Es una cadena de caracteres formada por una o varias palabras concatenadas y precedidas por una almohadilla o gato (#). Es, por lo tanto, una etiqueta de metadatos precedida de un carácter especial con el fin de que tanto el sistema como el usuario la identifiquen de forma rápida.

Trending topics: Es una de las palabras o frases más repetidas en un momento concreto en Twitter. Los diez más relevantes se muestran en la página de inicio, pudiendo el usuario escoger el ámbito geográfico que prefiera, mundial o localizado, o personalizadas, en función además de a quién sigue el propio usuario.

Algoritmo: es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

HTTP: Es el protocolo usado en cada transacción de la World Wide Web.

Glosario de términos

SQL: (Structured Query Language) Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

Servidor de Aplicaciones: Se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones.

Protocolo: Es un conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos para transmitir información por medio de cualquier tipo de variación de una magnitud física.

PAAS: (Platform as a Service), en español Plataforma como Servicio. Aunque suele identificarse como una evolución de SaaS, es más bien un modelo en el que se ofrece todo lo necesario para soportar el ciclo de vida completo de construcción y puesta en marcha de aplicaciones y servicios web completamente disponibles en la Internet. Otra característica importante es que no hay descarga de software que instalar en los equipos de los desarrolladores. PasS ofrece mútliples servicios, pero todos provisionados como una solución integral en la web.

JCP: (Java Community Process) es el mecanismo para el desarrollo estándares técnicos de especificaciones de tecnologías de java. Cualquier persona puede inscribirse para convertirse en un miembro de la JCP y luego participar en el grupo de expertos de la JSR.

JSR: (Java Specification Request) son documentos formales que describen las especificaciones y tecnologías propuestas para que sean añadidas a la

plataforma java. Las revisiones publicas formales JRS'S son controladas antes de que los JRS se conviertan en finales y sean votados por el Comité Ejecutivo JCP.

POJO: (Plain Old Java Object) es una sigla utilizada por programadores java para enfatizar el uso de clases simples y que no dependen de un framework en especial. Este acrónimo surge como reacción al mundo Java de los frameworks cada vez más complejos, y que requieren un complejo proceso de desarrollo que esconde el problema que realmente se está modelando.

GUI: (Graphical User Interface) interfaz gráfica de usuario, generalmente creada utilizando SWING o Abstract Window Toolkit(AWT).

HTML: (HyperText Markup Language) Lenguaje marcado de hipertexto, es un lenguaje basado en etiquetas capaz de asignar un formato a la información utilizada en el protocolo HTTP.

DAO: (Data Access Object) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.

JSON: (JavaScript Object Notation) es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

AJAX: (Asynchronous JavaScript And XML) Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad

de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

API: (Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

Framework: (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Mysql: Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

IDE: (Integrated Development Environment) Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Maven: Es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant.

Google Maps: Es un servidor de aplicaciones de mapas en la web que pertenece a Google. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o

imágenes a pie de calle Google Street View. Desde el 6 de octubre de 2005, Google Maps es parte de Google Local.

Bibliografía

Bibliografía

Bibliografía

SATNAM, A. (2008) COLLECTIVE INTELLIGENCE IN ACTION. ESTADOS UNIDOS DE AMÉRICA : MANNING PUBLICATIONS.

TOBY, S. (2007) PROGRAMMING COLLECTIVE INTELLIGENCE: BUILDING SMART WEB 2.0 APPLICATIONS. ESTADOS UNIDOS DE AMÉRICA: O'REILLY MEDIA.

DAVID ,G. ,CAY, S. (2010) CORE JAVASERVER FACES (3RA EDICIÓN). UNITED STATES OF AMERICA: PRENTICE HALL; 3 EDITION.

RICHARD, M. (2011) EJB 3.1 COOKBOOK . ESTADOS UNIDOS DE AMÉRICA: PACKT PUBLISHING.

MIKE, K. , MERRICK, S. (2009) PRO JPA 2: MASTERING THE JAVA™ PERSISTENCE API. UNITED STATES OF AMERICA: APRESS.

JONATHAN, R. (2010) THE AGILE SAMURAI: HOW AGILE MASTERS DELIVER GREAT SOFTWARE. ESTADOS UNIDOS DE AMÉRICA: PRAGMATIC BOOKSHELF.