Cámara de Comercio - Conecta Empleo (Fundación Telefónica)

ReadIt4me

Aplicación web para lectura e «interpretacion» de textos

Alexander López Borrajo
Jesús Couce Navas
Jonatan Mogan López
Alejandro Cebrián del Valle

18 de diciembre de 2019

Índice

Memoria del Proyecto

Nombre del proyecto	*ReadIt4me*
Empresa	OKB-4 Software Systems SCoop
Github	`https://github.com/AlejandroCebrianValle/Grupo4CursoJava`
Nombres de los participantes	*por orden alfabético:*
	Alejandro Cebrián del Valle — Diseño y coordinación
	Jesús Couce Navas — API
	Alexander López Borrajo — Web
	Jonatan Mogan López — Base de datos
Localidad	*Madrid*
Itinerario	Curso Aplicaciones en Java (Conecta Empleo - Cámara de Comercio)

1. Objetivo del proyecto

ReadIt4me es una aplicación *web* para el análisis y repositorio de textos en gran cantidad de formatos. Desarrollada en Java, pretende ser un servicio sencillo de usar e interpretar para aquellas personas u organizaciones que busquen resumir o simplificar textos en sus palabras clave o palabras más repetidas con significado (adjetivos, sustantivos, verbos, adverbios), pudiendo extraer sus propias conclusiones.

2. Esquema técnico

Las tecnologías utilizadas en este proyecto han sido:

- **Servidor**: Apache Tomcat (v 9.0.27).
- **Base de Datos**: MySQL (5.7.28).
- **Backend**: se utilizaron funcionalidades de Java EE con Maven como gestor de dependencias, Hibernate como librería JPA para proporcionar persistencia, JSLT y JDBC en las comunicaciones entre servlet y base de datos,
- **Frontend**: la parte visual de la aplicación se creo mediante JSP y HTML5, dandoles estilo mediante código en CSS y funcionalidades dinámicas mediante JavaScript con las librerías AJAX y JQuery.
- **IDE**: Eclipse.

2.1. Diseño de las interfaces *web*

El esquema técnico de la aplicación es simple pero resolutivo. La página inicial actua a modo de controlador o guardia de tráfico, permitiendo unicamente que los usuarios con nombre y contraseña acceder a los servicios. Si el usuario no está identificado o sus credenciales no son las correctas, no se dejará avanzar de este punto. Esto se consigue mediante un *servlet* conectado a la base de datos mediante JDBC.

Si el usuario está identificado, se permite acceder a una página donde podrá, gracias a JavaScript, JSTL y distintos servlets, acceder a los datos de los textos que haya subido a la aplicación, mediante una interfaz intuitiva (con botones fáciles de reconocer). El acceso a esta página únicamente se permite si asociada a la sesión, el usuario está identificado.

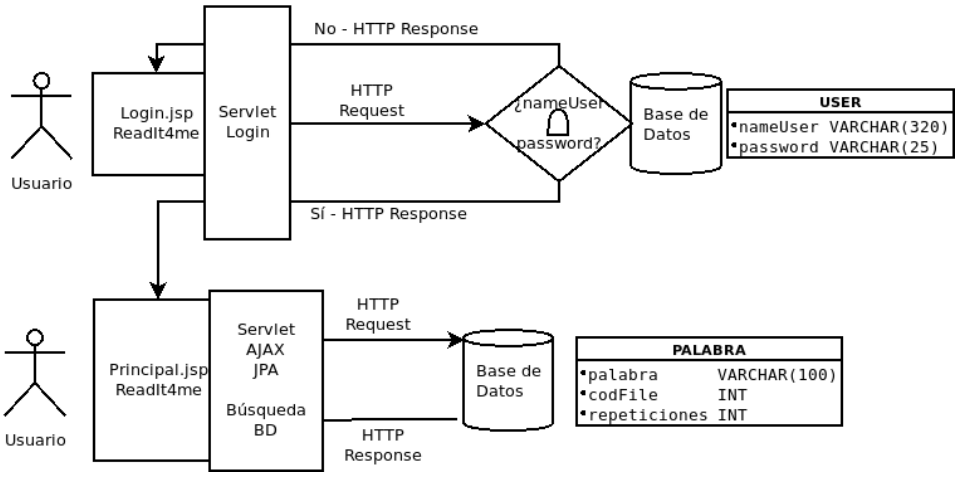

Figura 1: Esquema gráfico de la página de identificación.

2.1.1. Login

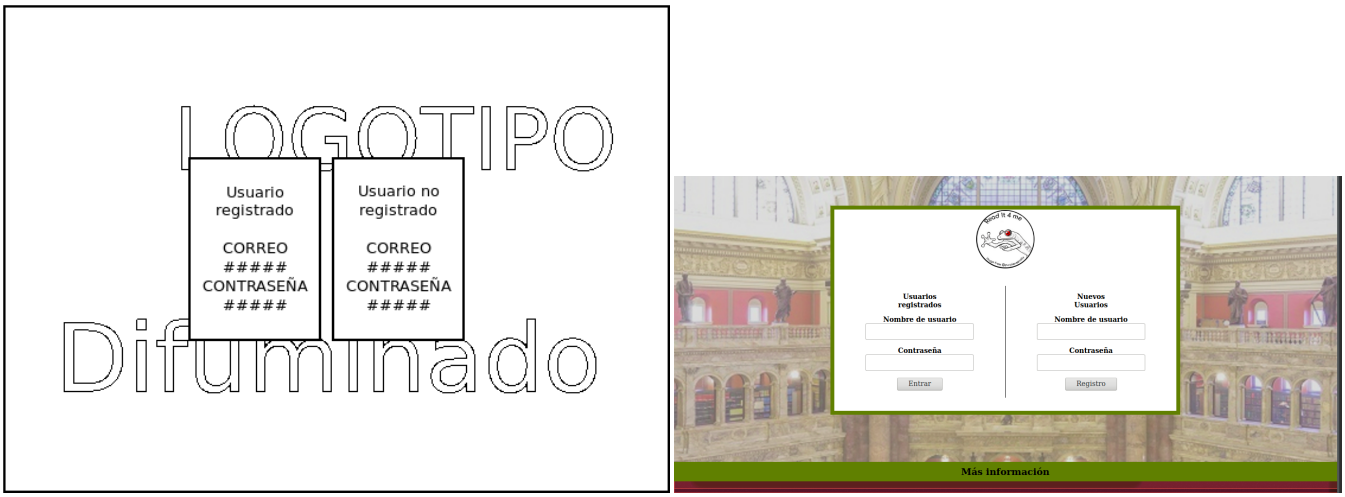

Figura 2: Esquema gráfico de la página de identificación y diseño final.

2.1.2. Página principal

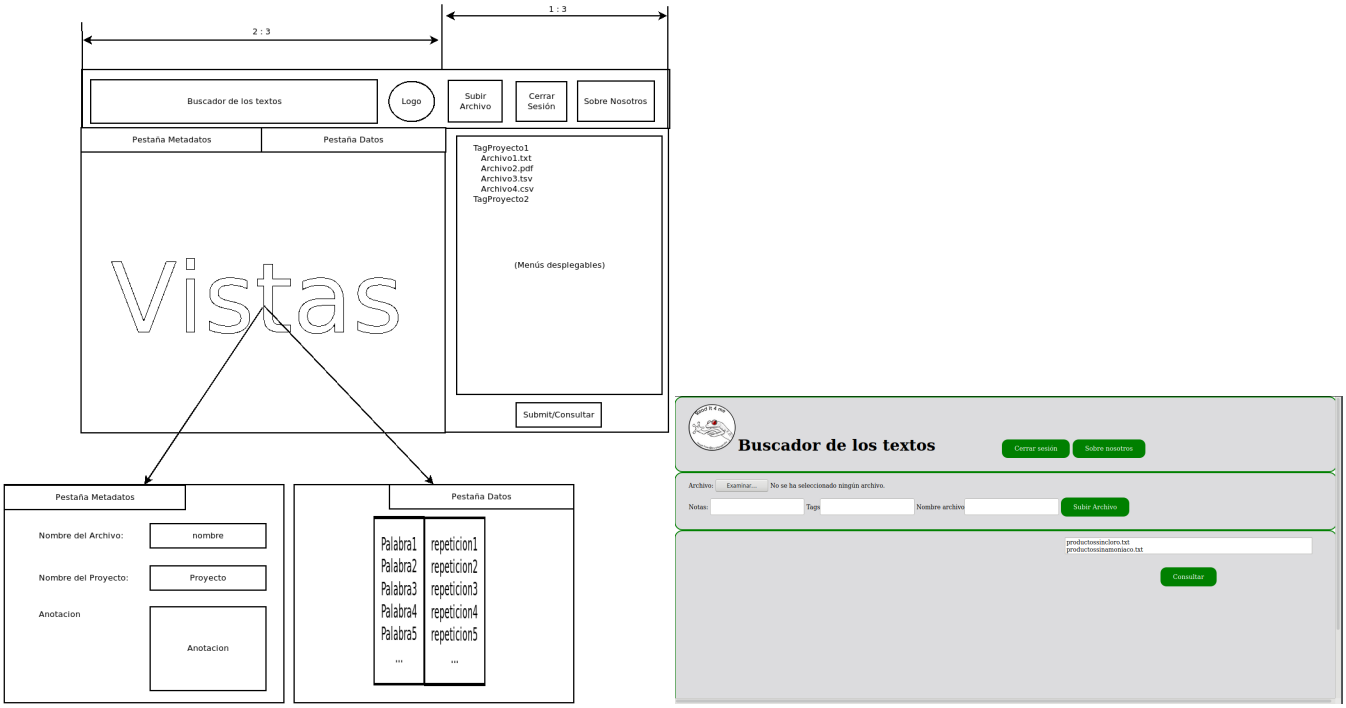

Figura 3: Esquema gráfico de la página principal y diseño final.

2.2. Base de datos

La necesidad de una base de datos se hace patente en el proyecto cuando se da un servicio de almacenamiento en servidor de documentos (manteniendo los documentos de forma privada) y cuando se espera una respuesta rápida del servicio ofertado: saber el número de repeticiones de las palabras en un documento. Para añadir la confidencialidad o privacidad de acceso a los documentos, se obligará a los usuarios a identificarse con un nombre de usuario y una contraseña para acceder a sus documentos. Porotra parte, usando técnicas de programación dinámica, los textos se procesan una vez y se crean una serie de registros en la base de datos que relacionan el documento con cada una de las palabras y su repetición, ahorrando al servidor capacidad de procesamiento.

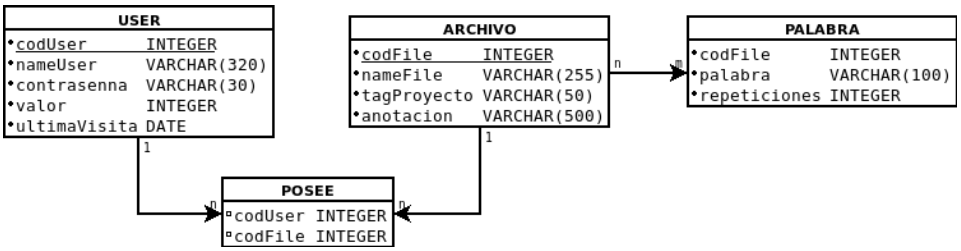

Figura 4: Modelo de entidad-relación.

- **Relaciones**:
  - USER-ARCHIVO: se da una relación de N a M en la que varios usuarios tienen un archivo y varios archivos son poseidos por un usuario, resolviendose con la tabla auxiliar POSEE, una tabla no indexada con relación 1 a N con USER y relación 1 a M con ARCHIVO.
  - ARCHIVO-PALABRA: se da una relación de N a M, en la que un sólo archivo es tiene varias palabras y una sola palabra es contenida en varios archivos.
- **Campos**:

Entidad	Campo	Tipo de Dato	Explicación
USER	codUser	INTEGER, PK	Índice que permite la indexación de la tabla de una forma sencilla.
	nameUser	VARCHAR(320)	Correo electrónico con el que se registra.
	contrasenna	VARCHAR(30)	Contraseña que identifica univocamente al usuario.
	valor	INTEGER	Cantidad de los logros conseguidos (Ver Gamificación)
	ultimaVisita	DATE	Fecha de la última visita
POSEE	codUser	INTEGER, FK	Índice que permite la indexación de la tabla de una forma sencilla.
	codFile	INTEGER, FK	Índice que permite la indexación de la tabla de una forma sencilla.
ARCHIVO	codFile	INTEGER, PK	Índice que permite la indexación de la tabla de una forma sencilla.
	nameFile	VARCHAR(255)	Nombre del archivo del que desea saber las repeticiones de palabras.
	tagProyecto	VARCHAR(50)	Proyecto al que el usuario asigna su archivo.
	Anotacion	VARCHAR(500)	Posibles anotaciones que quiera añadir el usuario sobre el texto.
PALABRA	codFile	INTEGER, FK	Índice que permite la indexación de la tabla de una forma sencilla.
	palabra	VARCHAR(100)	Palabra del que se registran sus repeticiones.
	repeticiones	INTEGER	Número de veces que se repite la palabra.

3. Metodología de trabajo

La metodología utilizada durante el desarrollo de la aplicación fueron reuniones semanales para concretar objetivos generales y desarrollo en el ámbito del hogar de las especificaciones habladas en esas reuniones. No se utilizaron metodologías ágiles debido al escaso número de miembros. Se creo por parte del organizador una guía de estilo con ánimo de armonizar el código asegurando al correcta organización y comprensibilidad de éste por los otros miembros del equipo de desarrollo. El equipo de desarrollo y la división del trabajo fue la que sigue:

Miembro	Funciones	
Alejandro Cebrián del Valle	Diseño:	Idea original y bases de la implementación de la aplicación.
	Coordinación:	Reparto de tareas a los miembros y supervisión de su ejecución.
Jesús Couce Navas	API:	Creación de código intermediario entre web y base de datos.
Alexander López Borrajo	Web:	Interfaz simple y intuitiva para el usuario.
Jonatan Mogan López	Base de Datos	Colección de datos y gestión de los mismos.

3.1. Guia de estilo

A efectos generales, el estilo a seguir es el llamado *Zen de Python* (PEP20).

Bello es mejor que feo,
Explícito es mejor que implícito,
Simple es mejor que complejo,
Complejo es mejor que complicado,
Plano es mejor que anidado,
Espaciado es mejor que denso,
La legibilidad es (hiper) importante,
Los casos especiales no son los suficientemente especiales para romper las reglas,
Sin embargo la practicidad le gana a la pureza,
Los errores no deberían pasar silenciosamente,
A menos que se silencien explicitamente,
Frente a la ambigüedad, evitar la tentación de adivinar,
Debería haber una, y sólo una, forma de hacerlo,
A pesar de que esa manera no sea obvia salvo que sea usted holandés,
Ahora es mejor que nunca,
*A pesar de que nunca es muchas veces mejor que * ahora * mismo,*
Si la implementación es difícil de explicar, es una mala idea,
Si la implementación es fácil de explicar, es una buena idea,
Los espacios de nombres son una gran idea, ¡tengamos más de esos!.

3.1.1. Resumen

- **Java**[1]:
  - **Comentarios**:
    1. Cada clase/programa comenzará con una explicación en un comentario en bloque incluyendo:
       a) Nombre de la clase
       b) Autor y fecha
       c) Para que se hace o qué ente representa.
       d) Descripción de métodos de la clase.
    2. Funciones o métodos que sean complicadas (más allá de un método get/set), se explicarán con un comentario en bloque que incluye:
       a) Nombre del método o función.
       b) Argumentos, indicando el tipo.
       c) Elementos que devuelve, indicando el tipo.
       d) *Opcional*: explicación de la función.
  - **Variables**:
    1. Los nombres de las variables estarán relacionados con aquello que guardan, evitando nombres absurdos y/o cómicos.
    2. Las constantes se escribirán con todas las letras mayúsculas, separando palabras mediante el carácter: _.
    3. Las clases tendrán el estilo *camel case*, cada palabra que forme el nombre se escribirá con la inicial mayúscula.
    4. Variables y métodos tendrán el estilo *camel case*, cada palabra que forme el nombre se escribirá con la inicial mayúscula, A EXCEPCIÓN de la primera.
  - **Estructura**:
    1. Los programas tendrán una extensión menor a las 2000 líneas.
    2. La extensión máxima de cada línea es de 80 caracteres. Si una expresión es demasiado larga, se romperá:
       - Después de una coma.
       - Después de un operador.
       - Se alineará la nueva línea al mismo nivel que que la línea anterior: comenzando tras el paréntesis que la encierra o en el caso de sentencias aritméticas de asignación, del operando de asignación.
    3. Tras el comentario inicial, se declararán todos los *package* y tras ellos, los *imports*, ordenandolos a ser posible en orden alfabético.
    4. Las indentaciones tendrán una longitud de 4 espacios.
    5. Las secciones dentro del código se marcarán con un comentario de línea antes de su comienzo.
    6. Las secciones tendrán un sentido, agrupando las funciones en bloques que trabajen en conjunto o que modifiquen conjuntos.

4. Proyecto

- **Privacidad de uso**: mediante el uso de una página de acceso obligatoria, se permite al usuario que sólo el usuario pueda acceder a los textos que tiene guardados y a los análisis ya efectuados.
- **Análisis de textos**: Esta aplicación *web* permite subir un archivo de texto plano y, en cuestión de unos pocos segundos, obtener un listado de las palabras usadas en el texto con su número de repeticiones. Este simple hecho estadístico, el de contar, permite a un investigador en mercados o académico medianamente experto o con poco entrenamiento, juzgar el contenido de un texto de forma mucho más agil a la habitualmente usada de lectura y anotación de palabras principales, ayudando y agilizando su tarea.

  (*Actualmente en proceso*): se está trabajando por implementar otras funcionalidades relacionadas como diagramas de porcentajes, Inteligencias Artificiales capaces de generar nubes de palabras y/o resumenes y su adaptación a la lengua inglesa y otras.
- **Agrupamiento por proyecto**: (*Actualmente en proceso*) El programa permitirá agrupar textos por temáticas para un análisis global de todos los textos.
- **Guardado de textos/Repositorio**: (*Actualmente en proceso*) El programa permitirá a quien lo use guardar en una carpeta a la que únicamente esa persona puede acceder, una colección de archivos guardados y ya analizados, lo que descarga la tarea de almacen o archivo digital de su forma física.

[1] Aquí se ofrece un pequeño resumen de la guía de estilo de Java, disponible aquí: `https://web.archive.org/web/20060228095122/http://developers.sun.com/prodtech/cc/products/archive/whitepapers/java-style.pdf`