# Midiendo "distància sintáctica" de variedades lingüísticas en zonas bilingües a partir de los datos masivos

Sergi Merino Robledo

**Resum–** El proyecto "Midiendo "distància sintáctica" de variedades lingüísticas en zonas bilingües a partir de los datos masivos"trata sobre como un solicitante requiere que su actual sistema de Base de datos por uno que tenga un rendimiento eficiente con su cantidad masiva de datos. Para ello se analizara el sistema actual de solicitante para saber que problemas tenia para posteriormente crear una nueva base de datos con una arquitectura distribuida propia y acompañada por una interfaz que permitirá a los solicitantes hacer las consultas que ellos demanden de una manera rápida y sencilla sin tener que usar un SGBD. A parte se trabajara con una metodología que gestionara el tiempo del proyecto y dividirá el proyecto en partes para que el desarrollo sea mas dinámico y eficiente.

**Paraules clau–** base de datos, migración, adaptación, mejora, optimización, docker, contenedores, fragmentación, interfaces, Big Data, metodología ágil, C, WPF.

**Abstract–** The project "Measuring "syntactic distància.ºf linguistic varieties in bilingual zones based on mass data"deals with how an applicant requires that your current database system by one that has an efficient performance with its massive amount of data. For this purpose the current system of the applicant will be analyzed to know that problems had to later create a new database with its own distributed architecture and accompanied by an interface that will allow the applicants to make the queries that they demand quickly and easily without having to use a SGBD. It will also work with a methodology that manages the time of the project and will divide the project into parts so that the development is more dynamic and efficient.

**Keywords–** database, migration, adaptation, improvement, optimization, Docker, containers, fragmentation, interfaces, Big Data, agile methodology, C, WPF.

## 1 Introducción

En este documento se muestran la planificación, desarrollo y la búsqueda de información para realizar el proyecto "Mesurant la "distància sintáctica"de variant lingüístiques en zones bilingües a partir de dades massius (Big Data)". Este proyecto surge del Filólogo Angel J. Gallego que solicita que se haga una migración de su actual base de datos a un nuevo sistema, ya que su sistema actual que utiliza MySQL no les proporciona las características que necesita una base de datos con mas de 20 millones de datos de tweets realizado en diferentes países de habla hispana, obtenido un rendimiento no aceptable, hace falta aclarar que esta base de datos MySQL no tiene ningún tipo de optimización y todos los datos se guardan en una tabla. A parte quieren una nueva interfaz para realizar consultas en esta nueva base de datos, que sea sencilla y si la utilización de comandas para hacerlas, como ejemplo se nos propociono un buscador de la RAE [1].

Para realizar dicho proyecto sera necesario observar y analizar el sistema actual para luego no repetir los mismos errores. Después de analizar el sistema se comenzara una búsqueda de tecnologías de base de datos, tanto relacionales como no relacionales, para tras obtener un gran abanico de posibilidades elegir una concreta para empezar a trabajar.

Tras ya tener una base para comenzar a desarrollan antes sera imprescindible desarrollar una metodología de trabajo

• E-mail de contacto: sergi.merino@e-campus.uab.cat
• Mención realizada: Ingeniería del Software
• Trabajo tutorado por: Oriol Ramos Terrades (Dpt. Ciències de la Computació )
• Curso 2018/19

y una gestión del tiempo lo mas optima posible, ya que esto representa una gran parte del proyecto y mas si se quiere que este se desarrolle correctamente.

A parte este proyecto da acceso a vivir lo que es realmente un proyecto de verdad, debido a una continua comunicación con los solicitantes para tenerlos informados y para saber si es necesario un cambio no esperado. A continuación se explicaran los objetivos del proyecto a completar en el apartado 1.1.

### 1.1. Objetivos

Tras analizar el proyecto se podrian definir 2 objetivos que se tendran que cumplir para que este proyecto sea considerado exitoso:

**[O-01] Primer objetivo: Creación estructura de Base de Datos** - Siendo este el objetivo mas importante y pilar principal del proyecto.

**[O-02] Segundo objetivo: Creación de Interfaz de consultas** - Este objetivo sera centrado en como el usuario tiene que hacer las consultas y como se ha creado de interfaz.

En la planificación inicial existía un tercer objetivo, que consistía en el desarrollo de mapas de calor en la web actual de los solicitantes, pero debido a que era con otra base de datos y no tenia relación con los otros objetivos este fue descartado.

## 2 Planificación del proyecto

En este apartado se explicaran las estrategias para gestionar este proyecto, empezando por la metodología utilizara en el apartado 2.1 y la gestión del tiempo 2.2 para poner llevar un control de desarrollo.

### 2.1. Metodología

La metodología seleccionada, la cual fue complicado de saber como gestionarla al ser un proyecto individual, fue una metodología ágil [2] organizando el proyecto en 3 etapa, creación de la estructura de base de datos, creación de la interfaz y una ultima etapa de testing y obtención de resultados, reflexión sobre ellos y conclusiones.

La idea es desarrollar las 2 primeras etapas en manera paralela, ya que dependen cada una de la otra, para que así al terminar poder hacer testing general a todo el proyecto. Así de esta manera te evitas bloqueos, ya que puedes dedicar tiempo a la otra parte para enfocar el problema desde otro angulo.

### 2.2. Gestión del tiempo

Para comenzar a gestionar el tiempo de desarrollo comenzaremos separando el proyecto en 3 partes con tareas asignadas a cada una:

**[P-01]** Creación de la estructura de la base de datos mediante Docker. Esta parte completa el objetivo **[O-01]**. Tareas en tabla 1.

| | |
|---|---|
| T-10 | Elección de tipo de BD y levantamiento default. |
| T-11 | Diseño y configuración de la base de datos. |
| T-12 | Creación de la Arquitectura distribuida de la base de datos. |
| T-13 | Creación del Docker-compose. |
| T-14 | Migración de los datos actuales. |

Tabla 1: Tareas **[P-01]**

**[P-02]** Creación de una Interfaz de consultas para la estructura de la parte 1.Esta parte completa el objetivo **[O-02]**. Tareas en tabla 2.

| | |
|---|---|
| T-20 | Familiarización con el entorno. |
| T-21 | Crear la conexión contra la estructura de la base de datos. |
| T-22 | Paso de las consultas MySQL a lenguaje WPF/C. |
| T-23 | Diseño de la interfaz |
| T-24 | Creación de la parte visual. |
| T-25 | tratamiento de los resultados de la consultas. |

Tabla 2: Tareas **[P-02]**

**[P-03]** Parte final del proyecto, que tiene como finalidad entregar un software fiable, realizando testing en la interfaz y pruebas a la estructura de base de datos. Tareas en tabla 3.

| | |
|---|---|
| T-30 | Retoques de Interfaz final. |
| T-31 | Testing y corrección de errores. |
| T-32 | Recolección de resultados y análisis. |

Tabla 3: Tareas **[P-03]**

Para llegar a cabo estas tareas se creara un diagrama de Gantt fig. 1 por semanas para la correcta gestión del tiempo de desarrollo, ya que es la parte mas complicada de este proyecto.

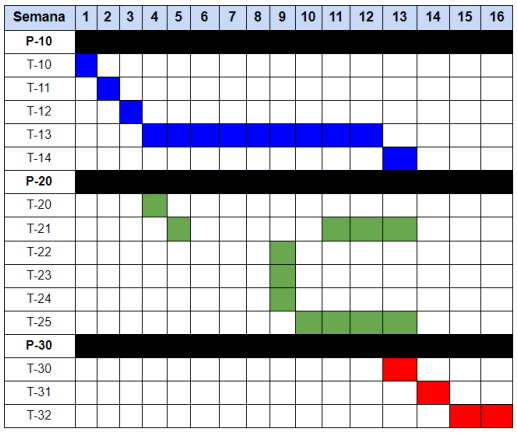


Fig. 1: Diagrama de Gantt del proyecto

## 3 Tecnologías existentes

En este apartado se mostrara la información buscada de las herramientas relacionadas con el proyecto, se comen-

zara explicando diferentes tipos de bases de datos ya que es objetivo principal. Se explicaran bases de datos relacionales 3.1 y no relacionales 3.2. Después se explicaran las herramientas de docker 3.3 y finalmente se explicara que herramientas se van a utilizar y las razones 3.4.

## 3.1. Bases de datos relacionales

Este es el tipo de base de datos que actualmente da problemas y que la característica principal que se requiere para solucionar el problema es el escalado horizontal [3] no se dará tanto énfasis en este apartado y se explicaran las características de este tipo.

Este tipo de bases de datos se compone por tablas o también denominadas relaciones, no puede existir dos tablas con el mismo nombre ni tampoco datos iguales, ya que este tipo de base de datos cumple las condiciones ACID (Atomicidad, Consistencia, Aislamiento, Permanencia). Estas relaciones o tablas tienen claves primarias para cumplir estas condiciones y también tablas ajenas para que datos de diferentes tablas estén referencia-das.

Nos proporciona ventajas tales como garantizar evitar la duplicidad de registros, garantizar la integridad relacional y favorecer la normalización para ser mas comprensible y aplicable.

Entre muchos gestores de bases de datos relacionales nos podemos encontrar a los siguiente: MySQL, PostgreSQL, Oracle, DB2, INFORMIX, Interbase, FireBird, Sybase, Microsoft SQL Server.

## 3.2. Bases de datos no relacionales

Para la búsqueda de bases de datos no relacionales hemos seleccionado las más importante y conocidas [4], para así tener un número claro de tipos a investigar.

**Cassandra**

Cassandra, teniendo como origen Facebook fue liberado y pasar a ser un proyecto open source es una base de datos NoSQL distribuida y fácilmente escalable teniendo como capacidad más reconocida su capacidad de escalar linealmente así como el soporte para multi data center o comunicaciones peer-to-peer entre nodos. Su Arquitectura y características nos proporciona bastantes buenos resultados, para empezar el nivel de consistencia es modificable según interese y también el nivel de query.En estas características [5] podríamos definir que es distribuida, escala linealmente y de forma horizontal y introduce una arquitectura Peer-to-Peer para eliminar fallos de patrones maestro-esclavo.

**Redis**

[6] Es una base de datos en memoria a partir de almacenamiento mediante hashes usada como una base de datos persistente. Escrito en ANSI C, software de código abierto y con un rendimiento elevado sin diferencias entre el tiempo de lectura y escritura. Ha diferencia de otras bases de datos que funcionan con hashes está también dispone de datos complejos como tablas, colas, pilas,etc. Además de ser persistente en disco así que el reinicio de la máquina no pierde la información. La escalabilidad se genera con un sistema de replicación maestro-esclavo haciendo que los servidores esclavos obtengan copias exactas del maestro.

**MongoDB**

Diseñada para ser rápida, flexible , escalable y fácil de aprender con herramientas de análisis de datos.Siendo esta la base de datos NoSQL líder del mercado, nos puede dar los siguientes beneficios [7] tal es como poder almacenar todo tipo de datos dando un gran rendimiento de escalabilidad horizontal y procesado de datos. Esta orientada a documentos con estructura JSONs para almacenar la información. Utiliza Map-Reduce, se puede actualizar sin detener el servicio y los servidores se sincronizan los datos de manera periódica.

**CouchDB**

CouchDB es una base de datos NoSQL, es capaz de replicarse en una amplia gama de entornos cliente/servidor. Basado en tecnología web es una base de datos distribuida con la capacidad de adaptarse a los servidores como a clientes de diferentes tipos. Las características [8] [9] son tales como que esta orientada a documentos escritos en SON y implementa la semántica ACID por lo cual se puede escribir y leer en múltiples lugares al mismo tiempo. Usa REST sobre HTTP y te permite trabajar con el de manera offline y al conectarse se actualizara el trabajo realizado.

## 3.3. Docker

Docker [10] es un proyecto de código abierto que te permite crear contenedores, también conocidos como maquinas virtuales ligeras, que son menos exigentes con que equipos pueden ejecutarlos. Las características principales es su gran portabilidad, ya que los contenedores pueden ser movidos a otro equipo fácilmente. Que sean maquinas ligeras quiere decir que ocupan poco espacio. Y su ultima característica va enfocada en su autosuficiencia ya que un contenedor no necesita de terceros para funcionar.

**Docker compose**

Docker compose [13] es una herramienta que permite crear scripts que facilitan el diseño y la construcción de servicios para así simplificar el uso de Docker. De esta manera se puede hacer un levantamiento simultaneo de múltiples contenedores fácilmente.

## 3.4. Definición de herramientas

Después de investigar las herramientas anteriormente mencionada y de estudiar los datos de la base de datos que tenemos que guardar se ha hecho una selección de herramientas.

Desde el punto de vista de la estructura de base de datos el tipo de base de datos seleccionado son las no relacionales, ya que nuestro objetivo primario es mejorar el escalado horizontal y el tipo de no relacional a sido [11] **MongoDB** ya que es una base de datos que suele funcionar a buen rendimiento y con un escalado horizontal muy bueno y con muchas opciones de configuración. Otro motivo de esta selección es que anteriormente yo ya he trabajado con este tipo de base de datos, lo cual me ahorrara tiempo de formación con las herramientas. Esta sera levantada en múltiples contenedores docker simultáneamente gracias al uso de docker compose.

Y después para realizar la interfaz de consulta usaremos como lenguaje de programación C# usando una aplicación

WPF, esta selección a sido realizada porque este tipo de aplicaciones son muy cómodas de crear y con el cual ya había trabajado y en un proyecto donde el problema mas grande es la gestión del tiempo va bien ahorrar tiempo en las cosas en las que sea posible.

# 4 DESARROLLO DEL PROYECTO

En este apartado se mostrara en primer lugar el diseño para la estructura de la base de datos y la interfaz en el apartado 4.1 seguido de el tipo de datos que se van a almacenar en el apartado 4.2. Después se comenzara a explicar la creación de la estructura en el apartado 4.3 y los criterios de la fragmentación de los datos en el 4.5 y terminaremos explicando la creación del interfaz de consultas en el apartado 4.7.

## 4.1. Diseño

### 4.1.1. Diseño de la BD

La idea de la estructura consta de 3 partes distintas o mejor dicho 3 tipos de contenedores mongo distintos. Existirán los contenedores que se ocuparan de almacenar las particiones de los datos o también llamados shards, estos se agrupan en este caso de 3 en 3 haciendo que cada shard sea un replica set, de esta manera podemos asegurar un nivel de disponibilidad mas gran en caso de que un conteneos se desconectara o se anulara. Existirán los contenedores de configuración que se ocuparan de mantener la configuración al re-levantar la estructura y es quien se ocupa de guardar los logs de la estructura. Para terminar tenemos un contenedor individual que usaremos como salida y al cual se tendrá que conectar la interfaz o un SGBD para poder acceder a todos los datos. Esta información se puede mostrar mas gráficamente en la figura 2 .

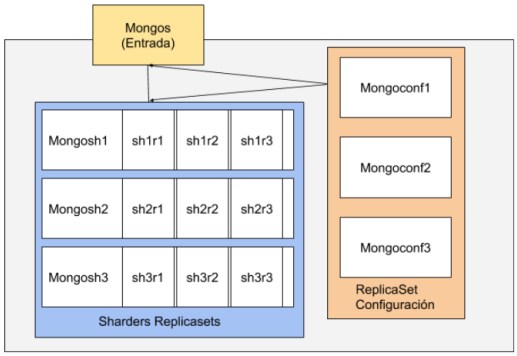


Fig. 2: Arquitectura de la Base de datos

En la figura 2 se muestran en el cuadrado azul los 3 shards formados por 3 contenedores formando 3 replica sets, en el cuadrado naranja el replica set de configuración y en amarillo el mongo de salida.

### 4.1.2. Diseño de la Interfaz

En cuanto se refiere al diseño de la interfaz es bastante simple, existiendo 4 pantallas, un menú principal que tendrá 3 opciones: Comenzar a hacer las consultas, ir al menú de configuración o salir de la aplicación. En el menú de consultas se pedirán los parámetros para hacer las consultas y al pulsar iniciar te llevara a la ventana de resultados donde se mostraran los resultados y se dará opción a guardar los datos en un archivo de texto. Y para terminar el menú de configuración te muestra los parámetros de la conexión de la interfaz y te permite editarlos. Todas estas pantallas menos en el menú principal tienen un botón de volver que te redirige a la ventana anterior. En la figura 3 se muestra el diagrama de flujo que representa lo anteriormente explicado.

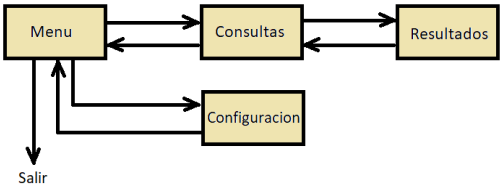


Fig. 3: Diagrama de flujo del Interfaz.

## 4.2. Estructura de los datos

Antes de comenzar a crear la estructura necesitamos cambiar el formato de los datos al nuevo formato que en el caso de MongoDB es JSON. para eso mantendremos los nombres de los atributos pero modificados para que el formato sea correcto.Los atributos serian: **texto** que representa el texto escrito en el tweet, **lema** que representa el texto escrito de la manera correcta, **etiquetado** que guarda la etiqueta de lo que representa cada palabra del tweet (verbo, sujeto, etc.) y es una array de strings que las etiquetas están en el mismo orden que en el texto, **conjunto** y **conjunto_lema** son una array de documentos que agrupa cada palabra del texto o lema con su etiqueta respectivamente y para terminar el **pais** que indica de que país es el tweet.

## 4.3. Creación de la estructura

Para la creación de la estructura explicada en al apartado 4.1.1 se creara un documento docker-compose para el levantamiento simultaneo de los contenedores. De estos contenedores solo necesitamos que el mongo de salida tenga un puerto de la maquina en el que este instalada redireccionado al puerto 27017 del contenedor, en el cual se encuentra la entrada al mongos. Estos contenedores guardan en todo momento el estado de la base de datos en volumes de docker, de esta manera si se para el servicio y se eliminan los contenedores si se vuelven a crear usando el mismo volumen la base de datos no se vera afectada. Este docker compose se puede ver en el apéndice A.1 .

Una vez levantados los contenedores tendremos que hacer saber a cada uno de ellos a que replica set pertenecen. De esta manera tras tener los 3 shards y el replica set de configuración se van a añadir los shards a la entrada mongos, pero una vez añadidos antes de añadir migrar los datos hace falta que se determine la clave de sharding para saber cual va a ser el criterio de fragmentacion.

## 4.4. Criterios de fragmentación

Teniendo ya los contenedores en marcha, los shards creados y preparados para hacer sharding de los datos que en-

tren por el mongos de entrada pero antes necesitamos definir cual sera el atributo que dividirá los datos en bloques y los repartirá por los shards. Tras analizar los datos explicados en el apartado 4.2 y tras hablar con los solicitantes de que tipo de consultas van a realizar se llego a la conclusión que el atributo **pais** es la mejor clave de sharding, ya que todas las consultas que se quieren realizar comienzan seleccionando en primer lugar el país. De esta manera se creara la colleccion que se desea en este caso se llama ”twitter”, se establecerá la clave de sharding para y se activara el sharding , debido a que no es un Integer no se puede crear rangos para que de dividan los datos, sino que se tiene que hacer mediante una función de hash interna de MongoDB haciendo que la información no esta totalmente dividida de manera igual en cada shard sino que depende de que si los paquetes de datos con el mismo país tienen un tamaño parido o no.

## 4.5. Migración de datos

La migración de datos fue algo costoso, ya que donde esta actualmente la base de datos MySQL no admite conexiones externas no puedo usar el SGBD NoSQLBooster para hacer una copia de los datos directamente. Así que se ha desarrollado un pequeño software que a partir de los documentos de texto de que le pedí a los solicitantes con una cantidad decente de datos con el formato JSON de la nueva base de datos los inserta en la nueva estructura. Fue muy importante que las string que inserte estén en formato iso-8859-1 que acepta todo tipo de caracteres, que siendo una base de datos para hacer estudios sobre el lenguaje es totalmente prioritario que los datos estén correctos.

El gran inconveniente de este método es que requiere bastante tiempo, pero como la migración solo se tendrá que realizar una vez se puede permitir. Aun así esto a sido un gran problema a la hora de hacer pruebas con diferentes configuraciones ya que se tardaba unas 12 horas en tener listo un escenario de pruebas con 20 millones de datos.

## 4.6. Seguridad

Para que esta estructura no sea totalmente vulnerable se ha instaurado un sistema de autenticación de usuario y contraseña para acceder al mongo de salida. No se especificaron temas de seguridad con los solicitantes, pero como buena practica tiene que existir aun que sea un mínimo. Para realizar dicha autenticación solo se ha tenido que apagar la estructura y añadir el termino –auth en el comando que levanta el contenedor mongo de salida.

## 4.7. Creación del interfaz

Tras poder conectar la aplicación WPF con la estructura anteriormente creada se empezara a desarrollar la [12] lógica y la parte visual de la interfaz, empezando por el menú principal, seguido de el menú de configuración y terminando con la ventana de consultas y resultados.Estas ventanas se pueden encontrar en el apéndice A.2.

**Menú Principal**

En esta ventana, al ser la primera, se definirán los colores y la sensación que transmitirá la interfaz al usuario. Para eso se han usado colores pastel y amarillo para transmitir tranquilidad y que al ser el fondo no destaque, mientras que los botones funcionan como si fuera un semáforo: rojo sales del software o vuelves a la pantalla anterior, verde sigues adelante con las consultas y ámbar para la configuración. Esto se puede ver en la figura 4.

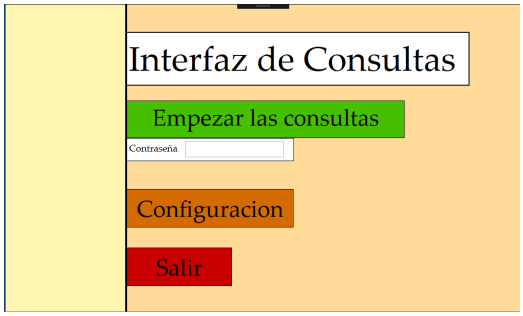


Fig. 4: Menú principal.

La única característica aparte de los 3 botones de empezar consultas, configuración y salir es que debajo de el botón de empezar consultas hay una passwordBox en la que el usuario podrá introducir la contraseña del usuario que esta guardado en la configuración. La contraseña se introduce cada vez que se quiere empezar a hacer consultas para evitar que la contraseña se guarde en el documento de configuración. Si el usuario o la contraseña no es correcta se le notificara al usuario.

**Configuración**

En este menú se muestran 4 TextBox en los cuales se le pide al usuario que introduzca la ip donde esta la estructura, el puerto, el usuario con el que va a ingresar y en el ultimo la base de datos y la colección que va a utilizar separados por un punto. Estos datos son guardados en un documento de texto el cual es guardado al pulsar el botón verde de Aplicar y se notificara que se han guardado correctamente. Al abrir esta ventana los datos del documento son escritos en las TextBox para que el usuario sepa que configuración tiene en ese momento.

**Consultas**

En esta ventana se encuentra realmente la mayor parte de la lógica de la interfaz. Existen 4 tipos de consultas distintas la cual puedes elegir en un menú desplegable y depende de que selecciones aparecerán unos inputs o otros para realizar dicha consultas, esto se muestra en la figura 5 . A parte tal y como se explica en el apartado 4.4 el atributo país es el primero que se elige, así que al entrar en esta ventana se consultara a la base de datos para obtener un listado ordenado alfabéticamente de los países y se introducirán en un menú desplegable para que el usuario pueda elegir. La primera y la segunda consulta son las mas simples, estas dos solo necesitan un input, ya que buscan a partir del texto o el lema respectivamente, no tiene porque ser exacto sino que solo hace falta que contenga lo escrito en el input siendo prácticamente la misma consulta cambiando que el atributo de búsqueda es el texto o el lema. A la tercera se le ha llamado Etiquetas Cercanas que es una consulta que busca documentos que tengan las etiquetas pasadas por el input

en el mismo orden en alguna posición de la array del atributo etiquetado separadas en este caso por comas dentro del TextBox donde se introduce el input. Y por ultimo la

cuarta consulta se llama Etiquetas Lejana”que en este caso tiene 3 inputs diferentes tales como Etiqueta 1, Etiqueta 2 y distancia. Esta consulta devuelve los documentos que tienen la etiqueta 1 a una distancia concreta de la etiqueta 2, encontrándose la etiqueta 1 siempre en primer lugar, a la inversa no.

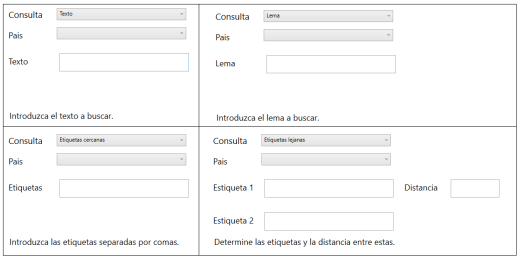


Fig. 5: Inputs para cada tipo de consulta

A parte gracias a realizar User Testing con usuarios de testing se ha introducido en la parte inferior un mensaje para saber como funciona cada consulta.

**Resultados**

En esta ventana se mostraran los resultados de la consulta con los parámetros introducidos en la ventana de consultas. Estos resultados son impresos en una tabla por la cual se puede hacer scroll vertical y horizontal y donde en la parte inferior hay un TextBox donde tras poner un nombre para el archivo de texto y pulsar el botón verde guardar se te guardara un archivo con dicho nombre en la carpeta de resultados del interfaz con un pequeño resumen de los parámetros de la consulta y luego todos los datos. Se le notifica al usuario cuando se ha guardado correctamente. Si un documento ya tiene dicho nombre este se sobrescribiera. El resumen de la consulta se puede ver en la figura 6 .

```
---- Datos de la busqueda ----
Numero de elementos: 919
Pais: AR
Tipo de busqueda: Texto
Texto: ejemplo
```

Fig. 6: Cabecera del documento de resultados.

Todas las ventanas se puede ver en el apéndice A.2.

### 4.7.1. Testing

El testing realizado en la interfaz esta enfocada en todos los inputs posibles dividiéndolos en 2 dependiendo donde se encuentran, ya que estos son los que no aceptan toda clase de caracteres, son los inputs de configuración y consultas.

Comenzando por el menú de configuración podemos ver como existen 4 inputs: dirección, puerto , usuario y BD.colleccion. En Dirección y Usuario se permite cualquier tipo de carácter mientras que en puerto solo se permiten numero. En el ultimo se permite todo menos tener mas de un punto ya que es lo que se usa para separa la base de datos de la collecion. Si cualquier cosa de las anteriormente mencionadas no se cumple se le notifica al usuario. Se puede ver mas gráficamente en la Figura 7.

| | Int | Char | String | Caracteres especiales |
|---|---|---|---|---|
| Dirección | | | | |
| Puerto | | | | |
| Usuario | | | | |
| BD.Colleccion | | | | Si, pero solo puede 1 “.” |

Fig. 7: Testing de inputs de la ventana Configuración.

A continuación seguiremos con la ventana de consultas que tiene en total 6 inputs.

Siendo estos el inputs de texto de la consulta por texto que acepta todo tipo de caracteres y lo mismo con el input de lema del tipo de consulta por lema.

Para la consulta de etiquetas cercanas se aceptan todo tipo de caracteres menos signos de puntuación, porque como usamos las comas para partir las etiquetas podría dar errores.

Y para terminar en el tipo de consulta de etiquetas lejanas se acepta en los inputs de etiqueta de todo menos símbolos especiales tales como signos de puntuación y en el input de distancia solo se permiten números. Se puede ver mas gráficamente en la Figura 8.

Si no se cumple cualquiera de estas condiciones se notifica al usuario.

| | Int | Char | String | Caracteres especiales |
|---|---|---|---|---|
| Texto | | | | |
| Lema | | | | |
| Etiquetas | | | | |
| Etiqueta 1 | | | | |
| Etiqueta 2 | | | | |
| Distancia | | | | |

Fig. 8: Testing de inputs de la ventana Consultas.

Hace falta destacar que al terminar la interfaz se invito a diferentes testers para que realizaran unas tareas con la interfaz hasta completar una checklist y que apuntaran todo lo que no les quedaba claro.

La checklist era bastante simple, era comenzar introduciendo la configuración para la conexión y luego realizar una consulta de cada tipo y guardar los resultados en diferentes documentos de texto. Fue un gran éxito, y gracias a esto se añadió los mensajes de información a la hora de rellenar los inputs de las consultas ya que algún usuario quedo algo bloqueado al hacer el test.

## 5 Análisis de resultados

Tras finalizar la creación de la estructura se van a realizar pruebas en esta para determinar si su estructura es óptima

haciendo levantamientos de estructuras como esta pero con 2, 3, 4, 5 shard para ver si 3 shards era el numero mas optimo.

Todas estas pruebas serán realizadas utilizando NoSQLBooster, ya que la interfaz al tener la necesidad de imprimir en pantalla los resultados esto hace que el tiempo se dispare. Así que se recomendaba que cuando se quieran hacer consultas de 10 millones aproximadamente se utilice el gestor NoSQLBooster.

Para hacer estas pruebas se han preparado 2 grupos de consultas, ambos tienen 3 consultas que devuelven diferente numero de documentos:

**Consulta leve**: poco procesamiento (100.000 documentos).

**Consulta media**: procesamiento medio (1,5 millones de documentos).

**Consulta elevada**: procesamiento extremo (20 millones de documentos).

Y la diferencia entre los 2 grupos es que el primer grupo hace consultas usando correctamente la fragmentación de los datos, filtrando primero el país y el 2n grupo son consultas cross database usando directamente el atributo texto. Los resultados se muestran en las figuras 9 y 10 respectivamente.

| | 2 Shards | 3 Shards | 4 Shards | 5 Shards |
|---|---|---|---|---|
| Consulta leve | 0,086 s | 0,085 s | 1,020 s | 1,153 s |
| Consulta media | 4,125 s | 4,561 s | 4,243 s | 4,436 s |
| Consulta extrema | 23,412 s | 15,472 s | 16,492 s | 15,325 s |

Fig. 9: Resultados comparación de numero de shards utilizando correctamente las particiones.

| | 2 Shards | 3 Shards | 4 Shards | 5 Shards |
|---|---|---|---|---|
| Consulta leve | 1,302 s | 0,302 s | 0,325 s | 0,253 |
| Consulta media | 6,325 s | 6,103 s | 7,361 s | 5,902 s |
| Consulta extrema | 49,632 s | 23,461 s | 24,262 s | 23,762 s |

Fig. 10: Resultados comparación de numero de shards sin utilizar correctamente las particiones (cross).

Tras realizar dichas comparaciones podemos concluir dos cosas principalmente: tener 2 Shards nos da un resultado bastante parecido al actual pero tiene más dificultades al hacer consultas a datos masivos y que aumentar en número de shards a 4 o 5 podrías ser positivo, pero no lo suficientes como para tener aumentar el coste de hardware que necesita dicho aumento. De esta manera podemos ver como realizar una estructura de 3 shards tiene un rendimiento muy acertado y se mantendrá. Los que sí se ha podido ver es que el tiempo en el grupo 2 es igual que en el grupo uno pero con tiempos más elevados por lo cual no cambia la conclusión de los resultados.

Tras realizar dichas pruebas con el numero de shards se comprobó si realmente el atributo pais es la mejor clave de sharding para hacer la fragmentación, por esta razón se probo hacer las mismas consultas que en las pruebas anteriores, usando solo el primer grupo en este caso, para ver si el resultado nos podía recomendar otra clave distinta. Tras realizar las pruebas con 3 shards en las estructura los resultados se encuentras en la figura 11.Se eligieron a parte de pais los atributos etiquetado, conjunto y conjunto_lema ya que son los únicos que se repiten en algunos documentos.

| | pais | etiquetado | conjunto | conjunto_lema |
|---|---|---|---|---|
| Consulta leve | 0,086 s | 1,832 s | 2,362 s | 2,472 s |
| Consulta media | 4,125 s | 13,275 s | 15,392 s | 16,946 s |
| Consulta extrema | 23,412 s | 65,834 s | 73,526 s | 69,273 s |

Fig. 11: Resultados comparación de claves de sharding.

Pero tras analizar los datos se puede ver claramente como el atributo país es el le mejor criterio de fragmentación posible en este proyecto.

## 6 Conclusiones

Tras la realización del proyecto y la obtención de resultados se puede llevar a cabo algunas reflexiones sobre qué se podría mejorar de este, tanto en la parte lógica como en el ámbito de la gestión del tiempo y las tareas.

Empezando con este último fue bastante difícil seguir la planificación inicial, siendo así necesario cambiarla hasta 3 veces a lo largo del proyecto, haciendo que el tercer objetivo quedase fuera por falta de tiempo. Aun así el poder gestionar un proyecto de unas proporciones más grandes de las que estoy acostumbrado y con libertad para desarrollar ha sido interesante e instructivo para siguientes proyectos.

Comentando los resultados como tal del proyecto que serían por un lado la estructura de contenedores que crean la nueva base de datos y el software para realizar consultas. La estructura fue lo más complicado, porque aunque ahora mismo se tarde unos minutos en levantar la estructura antes era un suplicio y no conocía del todo el funcionamiento de docker compose, lo cual tiene muchísimas opciones y creo que posiblemente con más tiempo y profundizamiento en el tema podría ser mejorable, pero por temas de tiempo tuve que pasar a la segunda parte que es el interfaz de consultas.

Esta interfaz ha sido curiosa de realizar porque se ha realizado mediante demanda de información al cliente y plasmando lo que el cliente pide en la lógica de la interfaz. Si, la interfaz podría tener muchas más opciones y puede que no guste visualmente a todo el mundo, aunque visualmente es aceptable, pero es funcional para el cliente, que en este caso de proyectos es lo más importante.

Estoy muy contento al ver los resultados, tanto el rendimiento de la estructura al realizar consultas con un resultado de millones de documentos y con la interfaz que aun tardando un poco mas te deja visualizar dichos resultados y plasmarlos en un documento para tenerlo a mano si se requiere. Ha sido una experiencia que aunque en algunos momentos ha sido bastante estresante y difícil ha dejado un buen sabor de boca.

## AGRAÏMENTS

Tengo que agradecer en primer lugar a mi tutor Oriol Ramos Terrades por ayudarme en este proyecto y indicarme no solo mis errores durante el seguimiento sino también aconsejarme cuando no sabia por que camino tirar. Ha sido agradable tenerlo como tutor ya que si le pides ayuda no duda en dártela. Cambien quiero agradecer al solicitante del proyecto Ángel J. Gallego por confiar en nuestra escuela para realizar dicho proyecto y por confiar en mi persona para este proyecto. Quiero agradecer a Juan Pablo Fuentes, actual ingeniero que se ocupa de la base de datos actual, por enviarme cualquier información que le pedía rápidamente y por cambiar los datos al formato nuevo y enviármelos para la migración. Y por ultimo quiero agradecer a todos los testers que realizaron user testing para la interfaz y realizar un trabajo tan bueno como el que realizaron.

Y a parte quiero dar las gracias a la UAB y a la ETSE por darme la oportunidad de enfrentarme a este proyecto y por hacerme aprender durante el proceso.

## REFERENCIAS

[1] http://web.frl.es/CORPES/view/inicioExterno.view;jsessionid=A939A077E19FD79901011C3F651518DC

[2] https://www2.deloitte.com/es/es/pages/technology/articles/waterfall-vs-agile.html

[3] https://www.oscarblancarteblog.com/2017/03/07/escalabilidad-horizontal-y-vertical/

[4] https://es.wikipedia.org/wiki/NoSQLTabla_Comparativa_de_SGBD_NoSQL

[5] https://www.paradigmadigital.com/dev/cassandra-la-dama-de-las-bases-de-datos-nosql/

[6] http://www.antweb.es/servidores/redis-todo-lo-que-debes-saber

[7] http://juanroy.es/es/mongodb-caracteristicas-y-futuro/

[8] https://www.arsys.es/blog/programacion/cloud-couchdb-bbdd/

[9] https://es.wikipedia.org/wiki/CouchDB

[10] https://openwebinars.net/blog/docker-que-es-sus-principales-caracteristicas/

[11] Mastering MongoDB 4.x: Expert techniques to run high-volume and fault-tolerant database solutions using MongoDB 4.x, 2nd Edition, Autor: Alex Giamas

[12] Programming WPF: Building Windows UI with Windows Presentation Foundation

[13] https://docs.docker.com/compose/compose-file/compose-file-v2/

## APÉNDICE

### A.1. Docker compose de la estructura

Para la creación de un shard y sus 3 replicas se utilizaría el siguiente docker compose que se muestra en la figura 12.

```
mongors1shard2:
  container_name: mongors1shard2
  image: mongo
  command: mongod --shardsvr --replSet mongors2 --dbpath /data/db --port 27017
  ports:
    - 52492:27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /mongo_data2/3/data1:/data/db
mongors2shard2:
  container_name: mongors2shard2
  image: mongo
  command: mongod --shardsvr --replSet mongors2 --dbpath /data/db --port 27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /mongo_data2/3/data2:/data/db
mongors3shard2:
  container_name: mongors3shard2
  image: mongo
  command: mongod --shardsvr --replSet mongors2 --dbpath /data/db --port 27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /mongo_data2/3/data3:/data/db
```

Fig. 12: Docker compose de un shard.

Para la creación de el mongo de configuración y sus 3 replicas se utilizaría el siguiente docker compose que se muestra en la figura 13 .

```
mongoconf1:
  container_name: mongoconf1
  image: mongo
  command: mongod --configsvr --replSet mongorsconf --dbpath /data/db --port 27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /mongo_conf1/3/config1:/data/db
mongoconf2:
  container_name: mongoconf2
  image: mongo
  command: mongod --configsvr --replSet mongorsconf --dbpath /data/db --port 27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /mongo_conf1/3/config2:/data/db
mongoconf3:
  container_name: mongoconf3
  image: mongo
  command: mongod --configsvr --replSet mongorsconf --dbpath /data/db --port 27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /mongo_conf1/3/config3:/data/db
```

Fig. 13: Docker compose de un mongo de configuración.

Y para la creación del mongo de entrada se utilizaría el siguiente docker compose que se muestra en la figura 14 .

```
mongos:
  container_name: mongos
  image: mongo
  depends_on:
    - mongoconf1
    - mongoconf2
    - mongoconf3
  command: mongos --configdb mongorsconf/mongoconf1:27017,mongoconf2:27017,mongodb
conf3:27017 --bind_ip_all --auth
  ports:
    - 52495:27017
  expose:
    - "27017"
  volumes:
    - /etc/localtime:/etc/localtime:ro
```

Fig. 14: Docker compose de un mongo de entrada.

## A.2. Pantallas de la Interfaz

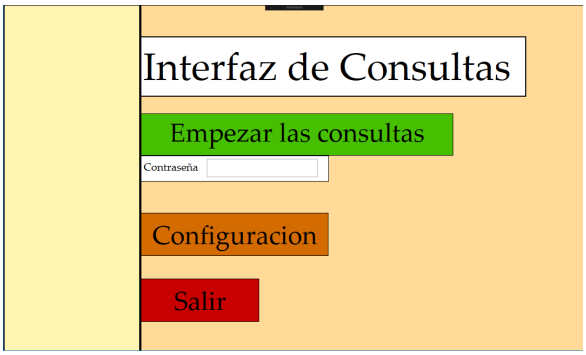


Fig. 15: Menú principal.


Fig. 16: Configuración.

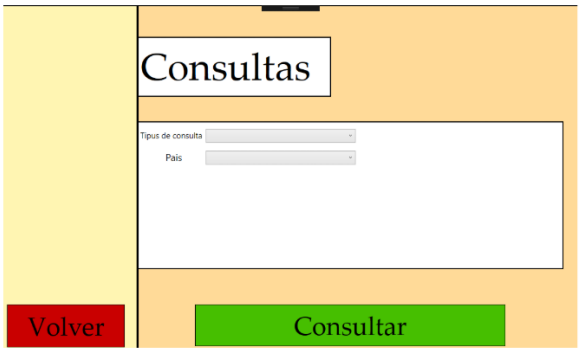


Fig. 17: Consultas.

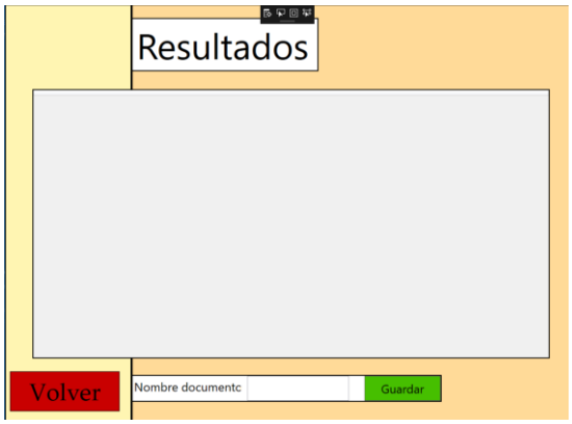


Fig. 18: Resultados.