Carrera: Licenciatura en Sistemas

Asignatura: Bases de Datos II

Trabajo Práctico 5

MongoDB

Construyendo un Blog

En este trabajo práctico vamos a construir un Blog. La temática del blog será elegida por el alumno. Utilizaremos MongoDB, un base de datos basada en documentos, como mecanismo de persistencia.

El blog estará compuesto por los siguientes elementos:

- páginas: Las páginas, por lo general, en los blogs se utilizan para describir detalles del autor del blog, describir la temática, realizar anuncias o promocionar eventos (siempre todo relacionado con la temática del blog). Cada página debe conformada por los siguientes datos:
  - título: Un título de la página
  - texto: El texto completo de la página
  - autor: El autor de la página.
  - fecha: La fecha de publicación de la página en el blog.
- posts: Los post son artículos específicos sobre la temática del blog. Cada post debe estar conformado por los siguientes datos:
  - título: Un título del post
  - texto: El texto completo del post.
  - tags: Palabras o frases cortas que definen la temática específica del post.
  - links-relacionados: URLs de sitios web que hablen sobre el mismo tema (o relacionados) del que habla el post.
  - autor: El autor del post.
  - Fecha: La fecha de publicación del post en el blog.

Como parte del trabajo práctico, el profesor entregará:

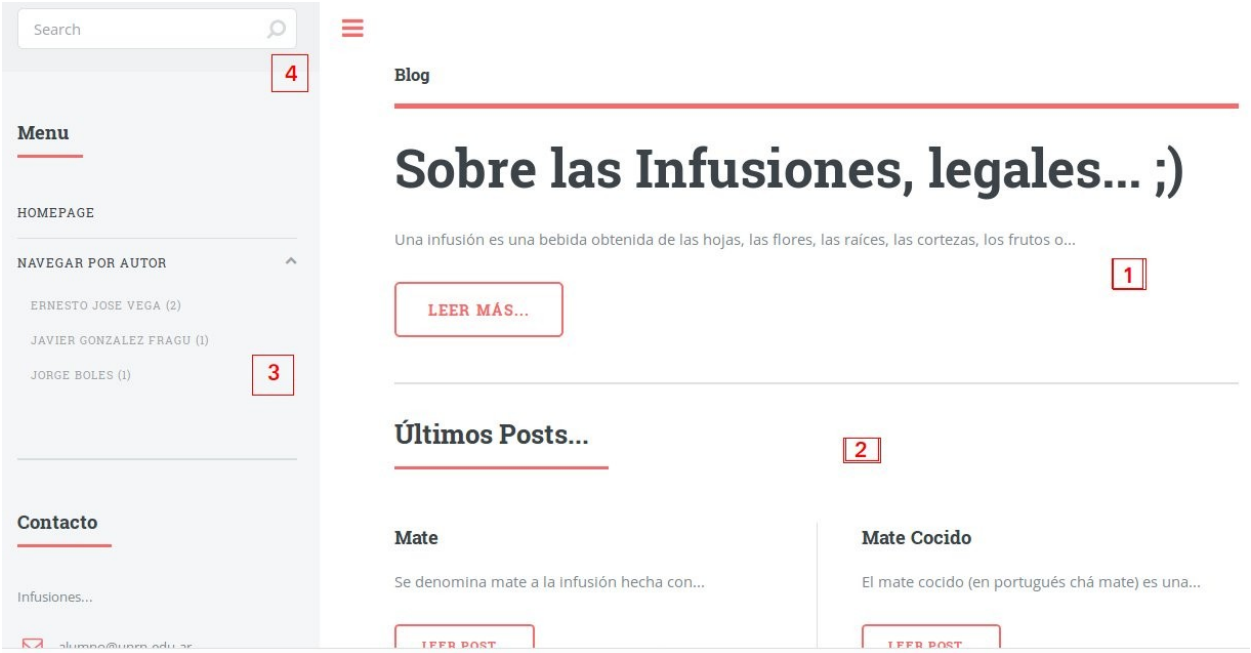
- Un proyecto Eclipse con el front-end del Blog. Escrito 100% en html+javascript (utiliza Ajax (https://es.wikipedia.org/wiki/AJAX) para consumir Servicios Web).
- Un proyecto Eclipse configurado para utilizar MongoDB Driver (http://mongodb.github.io/mongo-java-driver/3.4/driver/getting-started/quick-start/), más Spark (http://sparkjava.com/) que nos permite publicar servicios Web (API Web). Estos servicios son los consumidos por el front-end utilizando ajax.

El Alumno deberá:

- Diseñar la base de datos basada en documentos para persistir la definición del blog (páginas y posts).
- Generar datos de al menos 10 posts, y 2 páginas.
- Implementar la API Web definida para el trabajo práctico, verificando que el blog quede funcionando.

Especificación de la API Web

La siguiente imagen muestra la página web de inicio (home page) del blog.



Observe que cada sección esta enumerada en rojo. A continuación se describe cada una de las secciones y la API Web que se utiliza en cada caso.

1. Esta sección muestra una página la cual debe contar con una descripción del contenido/temática del blog y del autor. El ID de dicha página se debe poner en el archivo *html/assets/js/blog-index.js* (línea 2 para la variable *paginaId*), del proyecto *blog* (front-end). En el ejemplo de la imagen se puede observar que el blog se trata de infusiones.

Para hacer funcionar y mostrar el contenido como se ve en la sección 1, hay que implementar la API Web: ***/pagina-id/:id***, la cual debe retornar un *json* con la siguiente estructura. Cuando se clickea en "LEER MÁS", también se utiliza la API mencionada.


```
[
        {
                "_id":{
                        "$oid":"59ea5578fad4770f3bb0df1c"
                },
                "titulo":"Sobre las Infusiones, legales... ;)",
                "texto":"Una infusión es una bebida...",
                "autor":"Yo Mismo",
                "fecha":{
                        "$date":"2017-10-20T19:58:48.408Z"
                }
        }
]
```


2. Esta sección presenta los últimos 4 posts publicados en el blog (según la fecha de publicación). Para hacerlo funcionar se debe implementar la API Web: ***/ultimos4posts***, la cual debe retornar un json con la siguiente estructura:


```
[
  {
    "_id":{
      "$oid":"59e7e0b7fad4775bfde9623e"
    },
    "titulo":"Café",
    "resumen":"Sobre el Café solo..."
  },
  {
```

```
    "_id":{
      "$oid":"59e7df6efad4775bb2e9093c"
    },
    "titulo":"Té",
    "resumen":"Sobre el Té solo..."
  },
  {
    "_id":{
      "$oid":"59e7df6cfaw4775bb2e9093c"
    },
    "titulo":"Mate",
    "resumen":"Sobre el Mate..."
  },
  {
    "_id":{
      "$oid":"59e7df6cfad4175bb2e9093c"
    },
    "titulo":"Mate Cocido",
    "resumen":"Sobre el Mate Cocido..."
  }
]
```


Además, cuando se clickea en “LEER POST” de uno de los post, el blog utiliza la API Web: ***/post-id/:id***. La cual debe retornar un json con la siguiente estructura:


```
[
        {
                "_id":{
                        "$oid":"59e7e0b7fad4775bfde9625e"
                },
                "titulo":"Café",
                "resumen":"Sobre el Café solo...",
                "texto":"El texto completo del posts...",
                "tags":[
                        "té",
                        "infusión"
                ],
```

```
                "links-relacionados":[
                        "http://cafenegro.com",
                        "http://cafecito.com"
                ],
                "autor":"Jorge Boles",
                "fecha":{
                        "$date":"2017-10-18T23:16:05.755Z"
                }
        }
]
```

3. En esta sección se muestran todos los autores de posts y entre paréntesis la cantidad que tenga cada autor. En el ejemplo, se puede observar que el autor “Ernesto José Vega”, tiene 2 post, mientras que los otros dos autores solo 1. Para hacer funcionar esto se debe implementar la API Web: ***/byautor***, la cual debe devolver una estructura json con la siguiente forma:

```
[
  {
    "_id":"Jorge Boles",
    "count":2
  },
  ...
]
```

Haciendo click en el nombre de algún autor, el blog te muestra todos los post realizados por ese autor. Para hacer funcionar esto se debe implementar la API Web: ***/posts-autor/:nombreautor***, la cual debe retornar una estructura json con la siguiente forma:

```
[
  {
    "_id":{
      "$oid":"59e7df6cfad4775bb2e9093c"
    },
```

```
    "titulo":"Café",
    "resumen":"Sobre el Café solo...",
    "texto":"El texto completo del post...",
    "tags":[
      "café",
      "infusión"
    ],
    "links-relacionados":[
      "http://cafenegro.com",
      "http://cafecito.com"
    ],
    "autor":"Jorge Boles",
    "fecha":{
      "$date":"2017-10-18T23:10:36.305Z"
    }
  },
  {
    "_id":{
      "$oid":"52r7e0b7fad4775bfde9625e"
    },
    "titulo":"Té",
    "resumen":"Sobre el Té solo...",
    "texto":"El texto completo del posts...",
    "tags":[
      "té",
      "infusión"
    ],
    "links-relacionados":[
      "http://te.com",
      "http://teconleche.com"
    ],
    "autor":"Julio Mark",
    "fecha":{
      "$date":"2017-03-10T12:16:05.755Z"
    }
  },
  ….
]
```

4. Por último, esta sección muestra un cuadro de búsqueda de palabra libre sobre el texto de los posts. Al realizar una búsqueda, se obtiene una lista de resultados con cada post que contenga la palabra buscada en el texto. Para hacer funcionar esto se debe implementar la API Web: ***/search/:text***, la cual debe devolver un json con la siguiente estructura:

```
[
    "_id":{
      "$oid":"59e7df6cfad4775bb2e9093c"
    },
    "titulo":"Café",
    "resumen":"Sobre el Café solo...",
    "autor":"Jorge Boles",
    "fecha":{
      "$date":"2017-10-18T23:10:36.305Z"
    }
  },
  {
    "_id":{
      "$oid":"59e7e0b7fad4775bfde9625e"
    },
    "titulo":"Te con Leche",
    "resumen":"Sobre el Te con Leche...",
    "autor":"Javier Garcia",
    "fecha":{
      "$date":"2017-10-18T23:16:05.755Z"
    }
  }
,
  ...
]
```

Para poder implementar las queries necesarias para cada una de las API Web, necesitará entender como se implementan los siguientes items:

- Creaciones de Bases de datos y colecciones
- Inserción de Documentos

- Filtros (Filters/Read Operations)
- Ordenación (Sort)
- Proyecciones (projections)
- Agregaciones (Agregations)
- Creación de índices para búsqueda de texto libre (Create Indexes)
- búsquedas de texto libre (Text Search)

Utilice el siguiente tutorial oficial: http://mongodb.github.io/mongo-java-driver/3.4/driver/tutorials/