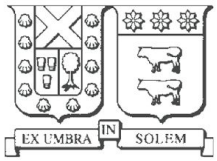
Departamento de Informática
Universidad Técnica Federico Santa María



Requisitos de Software

Proyecto: Scientific Adaptative Searching Platform (SASP)

Integrantes:

Nombres y Apellidos	Email	ROL USM
Pablo Alberto Aguirre Moreau	pablo.aguirre.14@sansano.usm.cl	201473555-3
Sebastian Ignacio Muñoz Godoy	sebastian.munoz.14@sansano.usm.cl	201473503-0
Jorge Ignacio Aliste Ahumada	jorge.aliste.14@sansano.usm.cl	201473538-3

Desarrollo del Prototipo

En la siguiente sección se presentará el proceso de fabricación del primer prototipo, el cual contempla primeras funcionalidades para el desarrollo de la plataforma. Esta considera la búsqueda de un cierto contenido según las palabras claves que ingrese el usuario y la presentación de los resultados obtenidos junto con sus correspondientes enlaces a dicha información. Estos por ahora no contendrán ningún filtro de tipo cognitivo (a implementar en posteriores entregas). Cabe mencionar que en un comienzo se utilizará un conjunto acotado de datos (para este primer prototipo datos generados aleatoriamente) para luego incluir contenido obtenido de la web a través de un "crawler".

Elección de Tecnologías

El desarrollo del prototipo comienza con la elección de las tecnologías a ocupar: el desarrollo será de una plataforma web, por lo que se ha elegido utilizar HTML y CSS para el estilo de la página. Como lenguaje de programación JavaScript junto con Node.js para el server de la plataforma. Finalmente, el buscador elegido para realizar las búsquedas es el motor de búsqueda Elasticsearch, cuya función es principalmente encontrar los contenidos que correspondan con la búsqueda que realice el usuario.

ElasticSearch será nuestro principal gestor de las búsquedas. Este funciona al almacenar todos los datos en los cuales deseamos buscar y luego, con distintas funcionalidades, es capaz de buscar texto en todos los archivos que se encuentren cargados al programa (PDF, HTML, Excel y otros). Con ayuda de estas funcionalidades se espera en un futuro prototipo poder filtrar nuestras búsquedas a partir del tipo cognitivo e implementar el sistema de calificaciones que ayudará a determinar qué contenido le sirve más a cada tipo cognitivo.

Prototipo actual

Para el prototipo de este entregable, solo se consideró un conjunto de datos generados aleatoriamente para determinar si nuestro buscador está cumpliendo su función. Estos datos poseen varios parámetros entre ellos título, contenido y un enlace a donde se encuentra el contenido almacenado en la web.

Nuestro prototipo actualmente funciona en la consola. Para realizar una búsqueda, el usuario debe colocar las palabras claves de su búsqueda y luego la cantidad de resultados que desea obtener. La **Figura 1** muestra esta parte del prototipo:

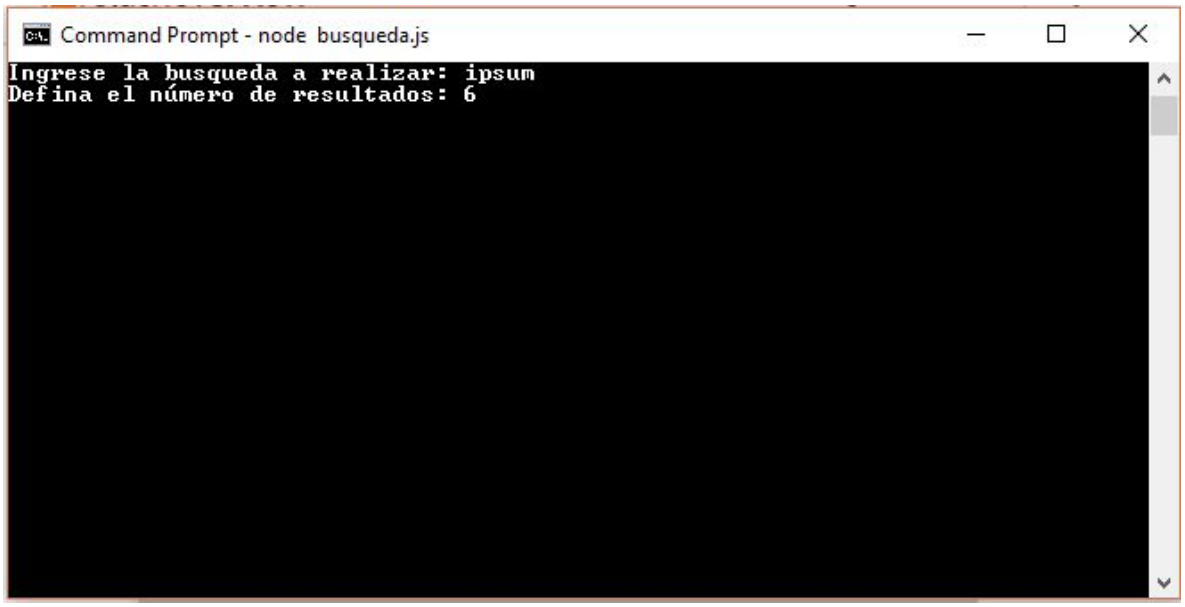


Figura 1: El usuario busca la palabra "ipsum" y define que necesita 6 resultados

Luego que el usuario ingresa su búsqueda, el programa le retornará un listado de los resultados obtenidos. Estos resultados serán mostrados de la siguiente forma: primero se especificará el título de los archivos encontrados seguido de su correspondiente enlace, el cual llevará al usuario a donde se encuentra alojado el archivo buscado. La **Figura 2** muestra cómo los resultados son entregados por el programa:

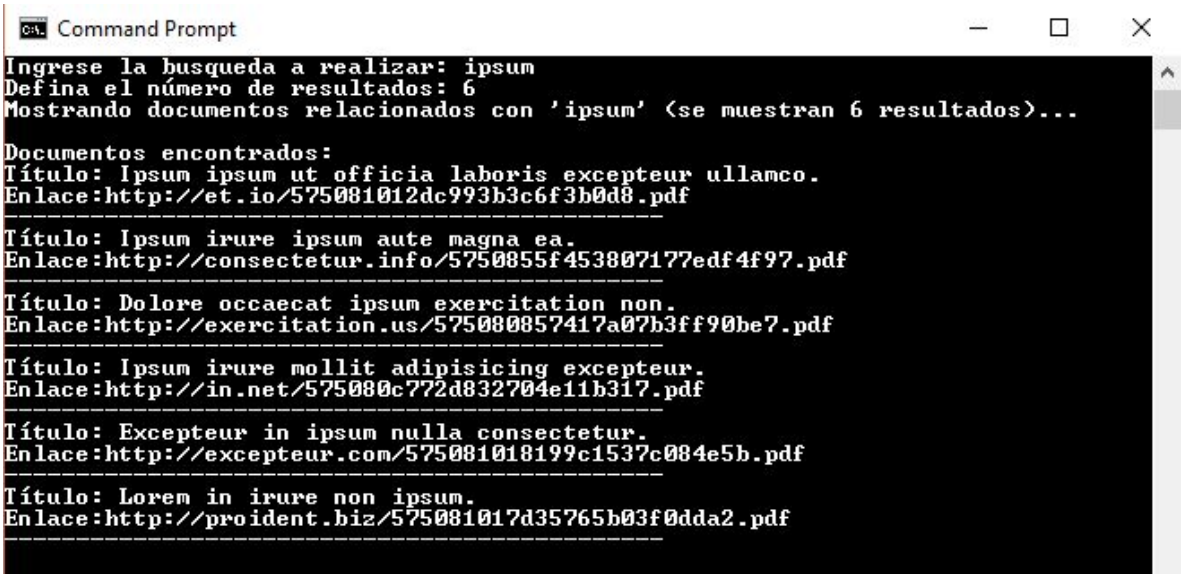


Figura 2: 6 resultados de la búsqueda "ipsum"

El prototipo también incluye una vista parcial de lo que será la interfaz gráfica de la aplicación. Esta consiste en una vista principal en la cual se deberá elegir el tipo cognitivo que filtra los resultados entregados por el buscador, ya que estos variarán dependiendo del tipo elegido. La **Figura 3** muestra específicamente cómo es esta vista principal:

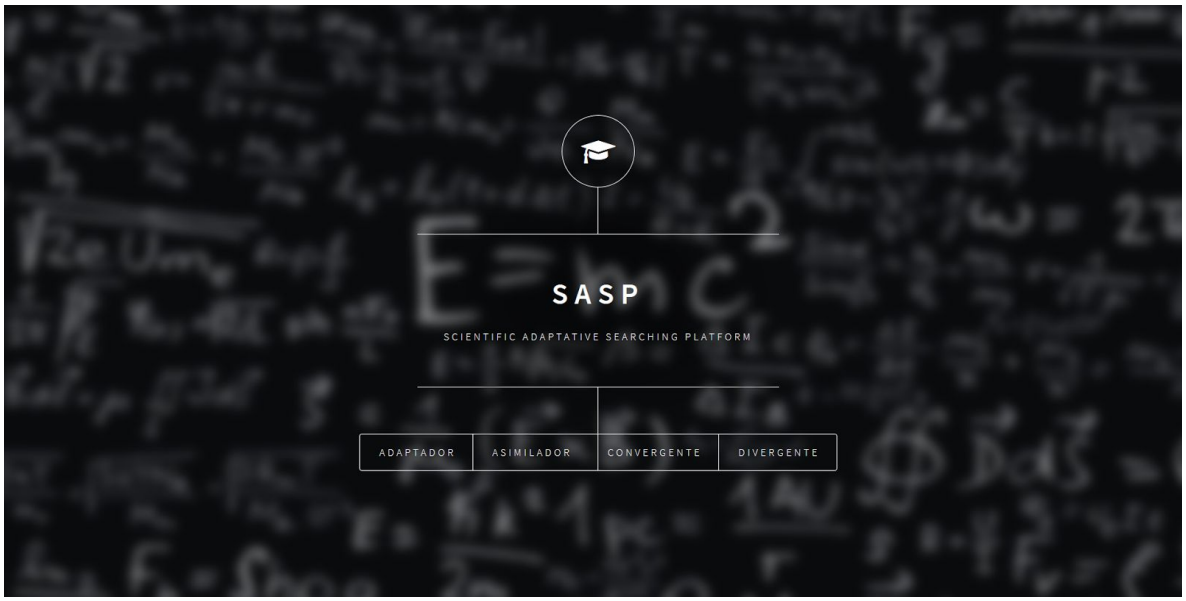


Figura 3: Vista principal de SASP

Cuando el usuario seleccione el tipo cognitivo por el cual filtrar la búsqueda, se desplegará el buscador, indicando para qué tipo cognitivo se está buscando. La **Figura 4** muestra el buscador desplegado para un usuario asimilador:

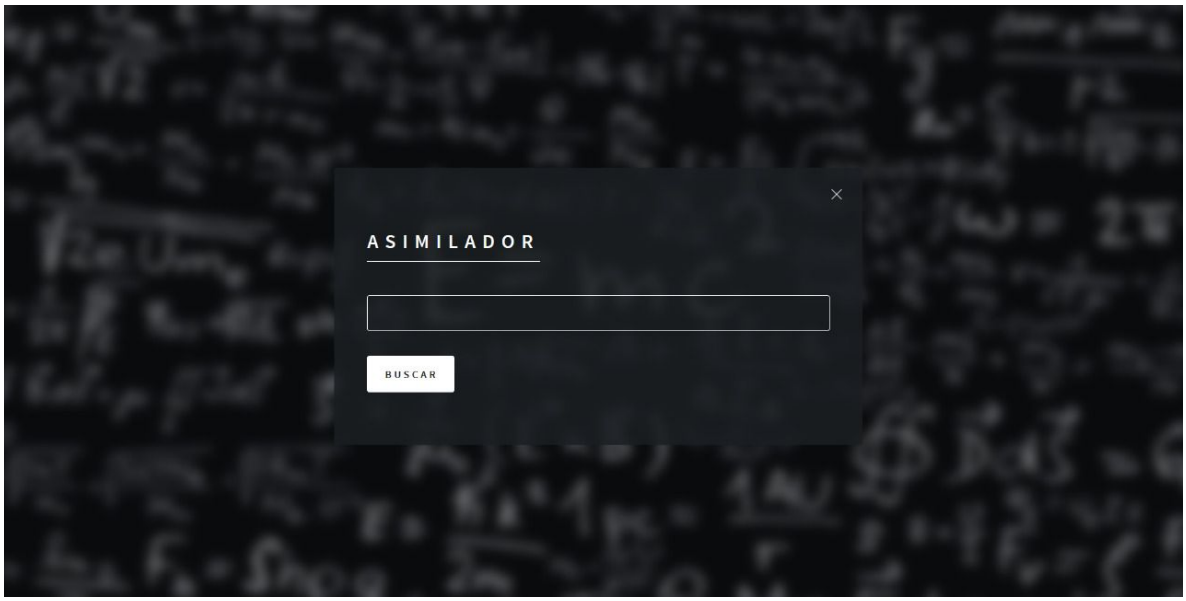


Figura 4: buscador para un usuario asimilador

Una vista con los resultados aún no se encuentra funcional, pero esta consistirá en una lista con los enlaces que se adapten a las palabras claves que el usuario ingresará en el buscador. Cada enlace tendrá calificaciones de lo usuarios, por ejemplo, si varios usuarios asimiladores han votado que el enlace es de mucha utilidad, las búsquedas realizadas por los asimiladores priorizará este enlace, mostrándolo al comienzo de la lista de resultados. Además, cada enlace tendrá un botón en el que los usuarios podrán dejar su calificación y posible comentario, junto con su mail.

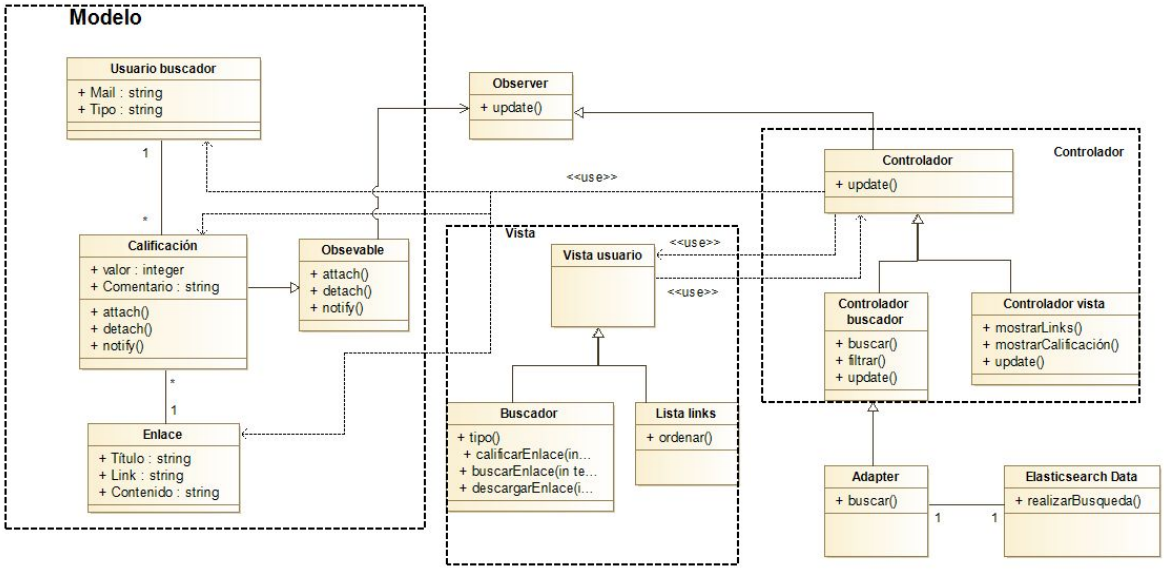
Selección de Patrones de Diseño

Intención	Patrón de Diseño	Razonamiento
En el Diagrama de Clases se desea mostrar el proceso de una búsqueda realizada exitosamente y posterior evaluación del enlace visitado, además de cómo interactúan las distintas partes del sistema.	MVC	Este patrón de diseño nos permite administrar de manera más expedita nuestro código, a la vez que las funciones que ejecuta cada parte de este. Utilizando MVC podemos realizar separaciones visibles, dividiendo el Modelo, parte del sistema que interactúa con la base de datos que almacena y recupera las calificaciones y enlaces, las vistas, código HTML que genera la parte visual con la que interactúa el usuario del portal web, y el controlador, la capa de que sirve de enlace entre ambas, envía comandos al modelo para actualizar su estado, o sea, agregar y/o modificar opiniones y a la vista que corresponda para cambiar su presentación, o sea, reaccionar a las interacciones con usuarios, por ejemplo mostrar la lista de resultados tras una

		búsqueda, o abrir un enlace al ser seleccionado, no obstante no es el encargado de manipular datos o generar una salida. En resumen, la utilización de este patrón de diseño en nuestro caso se justifica con una mejor modularización del trabajo, separar las responsabilidades mejorando las prácticas de desarrollo, abriendo posibilidades a escalabilidad y facilitar el trabajo de los desarrolladores.
Se quiere mostrar la actualización de sugerencias que ocurre en el sistema cuando un usuario califica con 5 estrellas un link o página.	Observer	Para una ideal implementación de una clase observadora que permita crear una sección de sugerencias en las que se muestren de manera preferencial calificaciones de 5 estrellas a material de estudio, se utiliza este patrón. Por lo que la clase calificación debe ser capaz de avisar a la clase usuario que alguien calificó con 5 estrellas un link.
Se desea mostrar en el Diagrama de Clases la adaptación que se necesita hacer para la obtención de datos al buscar.	Adapter	Elasticsearch tiene su propia forma de buscar el contenido, por lo cual es necesaria una forma de adaptar esta forma a las necesidades de nuestro programa. Entonces, el controlador buscador se debe apoyar de un adaptador para comunicarse con los métodos de búsqueda de Elasticsearch.

Creación de Diagrama de Clases

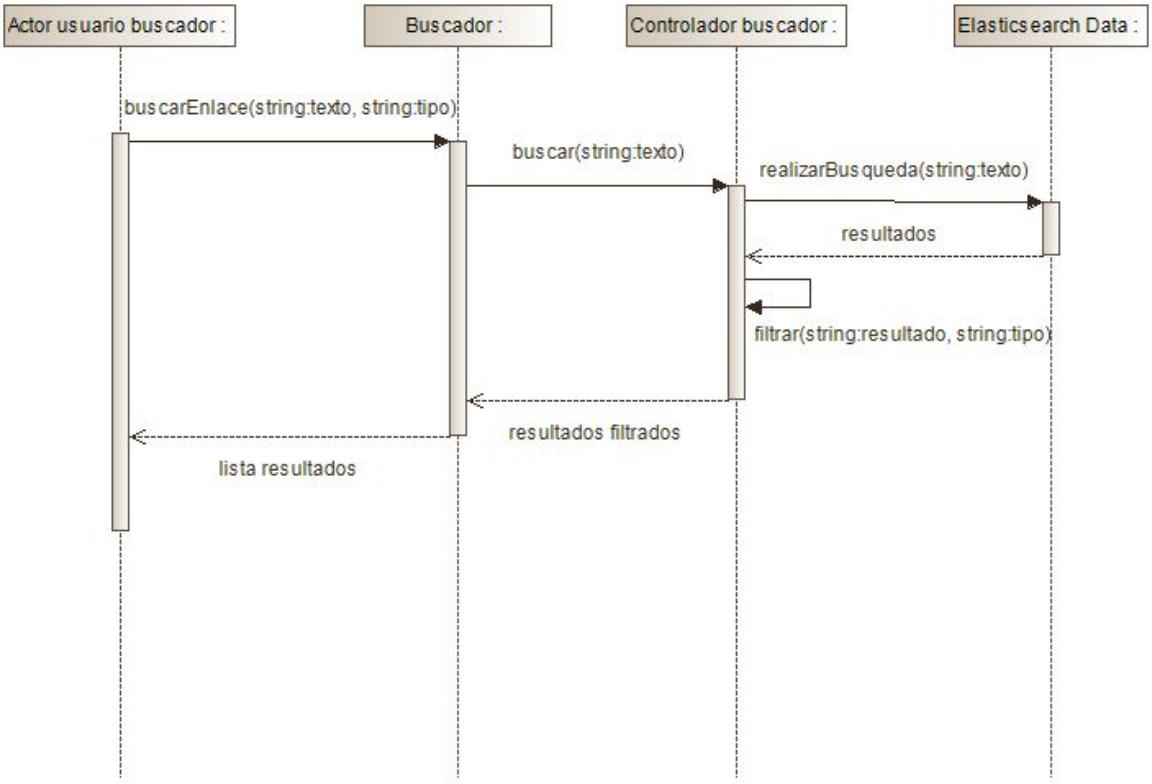
El diagrama de clases contiene el patrón MVC implementado: el modelo corresponde a los datos de los usuarios, las calificacione que se almacenan y los enlaces que se obtienen de la web, la vista se compone de una vista para el buscador que permite elegir el tipo cognitivo al cual buscar y otra vista correspondiente a la lista de resultados, el controlador es el que modificará e interactúa con estos dos componentes y realizará las operaciones de búsquedas. El patrón Observer se encargará de notificar al controlador cuando una calificación sea ingresada y esta tenga cinco estrellas. Finalmente, el patrón adapter ayudará a el controlador a interactuar con la interfaz de elasticsearch, nuestro motor de búsqueda.



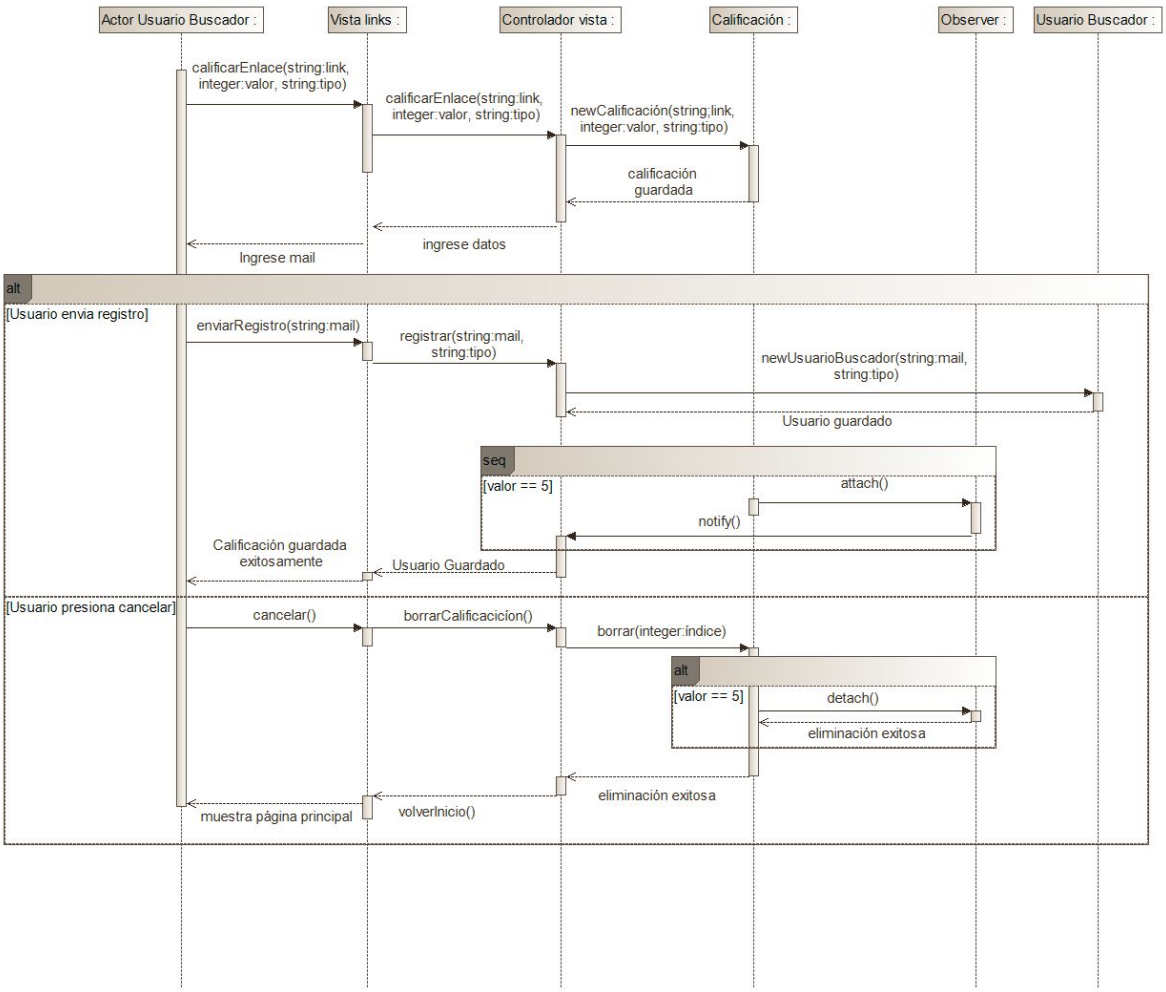
Diagramas de Secuencia

Se eligieron los casos de uso **Buscar contenido** y **Calificar contenido.** Esto ya que la principal funcionalidad requerida por el cliente es la de buscar contenido en la web, por lo que es el caso de uso más importante de nuestro software. Luego, calificar contenido tiene casi la misma importancia ya que gracias a este método se podrá organizar la manera en que se entregan los resultados al usuario, colocando los de más importancia al comienzo de la lista y sugerencias de contenido respecto a las calificaciones anteriores (gracias al patrón Observer).

- Buscar contenido:



- Calificar contenido:



Análisis de Trade-off

La nueva funcionalidad a implementar, pedida por el cliente, es una búsqueda en la web ampliada. Para explicar mejor esto, recordamos que nuestra propuesta inicial fue realizar la búsqueda en sitios web definidos previamente, de tal forma que la búsqueda se realice dentro de estos para mejorar los resultados obtenidos

asegurándose de la confiabilidad de estos y la velocidad de búsqueda. La petición del cliente consiste en que la búsqueda se haga más ampliamente en la web sin que este se vea en la necesidad de ingresar sitios previamente a las búsquedas.

Pregunta:

¿Qué herramientas deben usarse para implementar la búsqueda ampliada descrita previamente?

Opciones:

- O1: Utilizar Elasticsearch en conjunto con un Crawler para realizar la búsqueda ampliada.
- O2: Utilizar Solr para realizar la misma tarea
- O3: Crear un módulo que se encargue de buscar y seleccionar páginas web elegibles para la búsqueda de manera automática, y luego implementar la solución inicial.

Criterios:

- C1: Tiempo de búsqueda
- C2: Almacenamiento
- C3: Implementabilidad (si bien no está incluido en los NFR del entregable 1, también es un criterio utilizado para elegir entre estas opciones).

Análisis (Trade Off):

Criterio\Opciones	O1	O2	O3
C1	+	0	++
C2	0	0	0
C3	+	-	-

- La opción 1 (O1) en general tiene un buen tiempo de búsqueda según la investigación realizada sobre Elasticsearch. En cuanto a implementabilidad, es la más realizable de implementar, y testings hecho por el equipo señalan como viable su utilización.
- La opción 2 (O2), basado en lo que hemos investigado, no aporta un particular desempeño en tiempo de búsqueda. En cuanto a implementabilidad, testings hechos por el equipo han arrojado solo fallos y por tanto concluimos que esta es difícil en esta opción.
- La opción 3 (O3) basado en un análisis teórico del desempeño de los componentes del sistema muestra ser positivo en cuanto a tiempo de búsqueda, ya que buscar en una cantidad limitada de páginas reduce drásticamente el tiempo que toma realizar una búsqueda en general.
- Todas las opciones se muestran neutrales en cuanto a almacenamiento, puesto que no interactúan directamente con esta parte del sistema, sino que están inmersas en el módulo de búsqueda.

En conclusión, la opción 1 es la seleccionada por el equipo, dado que es favorable en todos los aspectos pertinentes, que incluye un buen tiempo de búsqueda, una característica deseable en cualquier tipo de buscador, sin además comprometer ningún criterio, a diferencia de la opción 2 y 3 cuya implementabilidad reduce el valor que estas alternativas entregan. Aún cuando O3 ofrece un tiempo de búsqueda aún mejor que O1, va en contra de los deseos del cliente que la búsqueda sea limitada a ciertas páginas en la medida de lo posible, por ende la opción 1 es la definitiva.