Universidad Técnica Federico Santa María

Ingeniería de Software

Entregable 2

Buscador de Herramientas de Aprendizaje

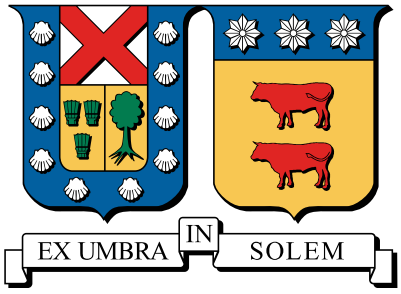
Autores:
Quantum Taco

Francisco Olivares
francisco.olivars.14@sansano.usm.cl
201473575–8

Felipe Vega
felipe.vega.14@sansano.usm.cl
201473511-1

Profesor:
Hernán Astudillo

Primer Semestre 2017



Índice

1. Desarrollo del prototipo

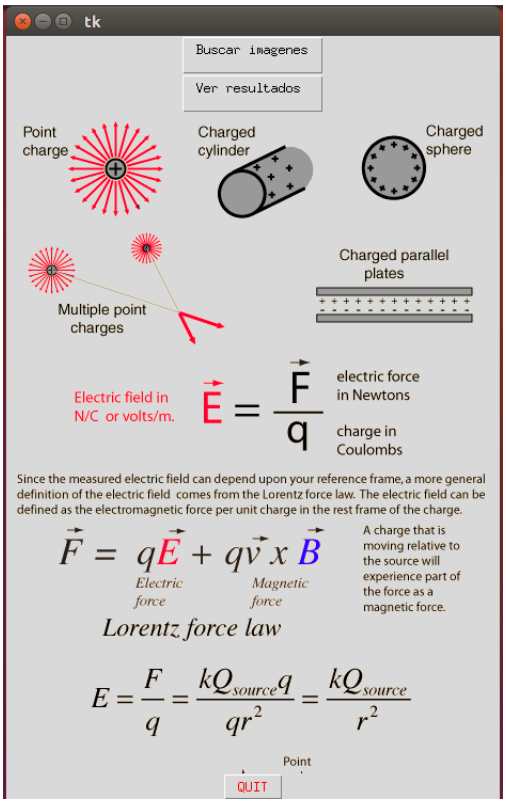


Figura 1: Captura del estado actual de la demo.

Para esta entrega se ha trabajado en lo que es la búsqueda de herramientas de aprendizaje desde sitios web, actualmente la demo funcional puede buscar las imágenes de un tópico fijo (campo eléctrico) en un solo sitio web que se asume de confianza (hyperphysics: http://hyperphysics.phy-astr.gsu.edu), los enlaces de las imágenes encontradas son almacenados en un archivo de texto y mostrados en la sencilla interfaz mostrada en la figura 1.

2. Selección de patrones de diseño

Intención	Patrón de Diseño	Razonamiento
Se desea mostrar en el diagrama de clases una interacción ordenada y separada del sistema	MVC	El patrón MVC satisface dicha necesidad ayudando a separar el sistema en tres componentes principales (modelo, vista y controlador) lo cual facilitará el desarrollo para cumplir todos los requisitos funcionales.
Se desea mostrar en el diagrama de clases la interacción con la base de datos para manejar falsos positivos	Active Record	Se decide utilizar Active Record para gestionar las interacciones con la base de datos.
Se desea mostrar en el diagrama de clases la creación de solo una clase para gestionar la búsqueda de contenido	Singleton	Se decide utilizar Singleton ya que este permite tener solo una instancia de clase que pueda ser utilizada muchas veces.

Tabla 1: Patrones de diseño

3. Creación de diagrama de clases

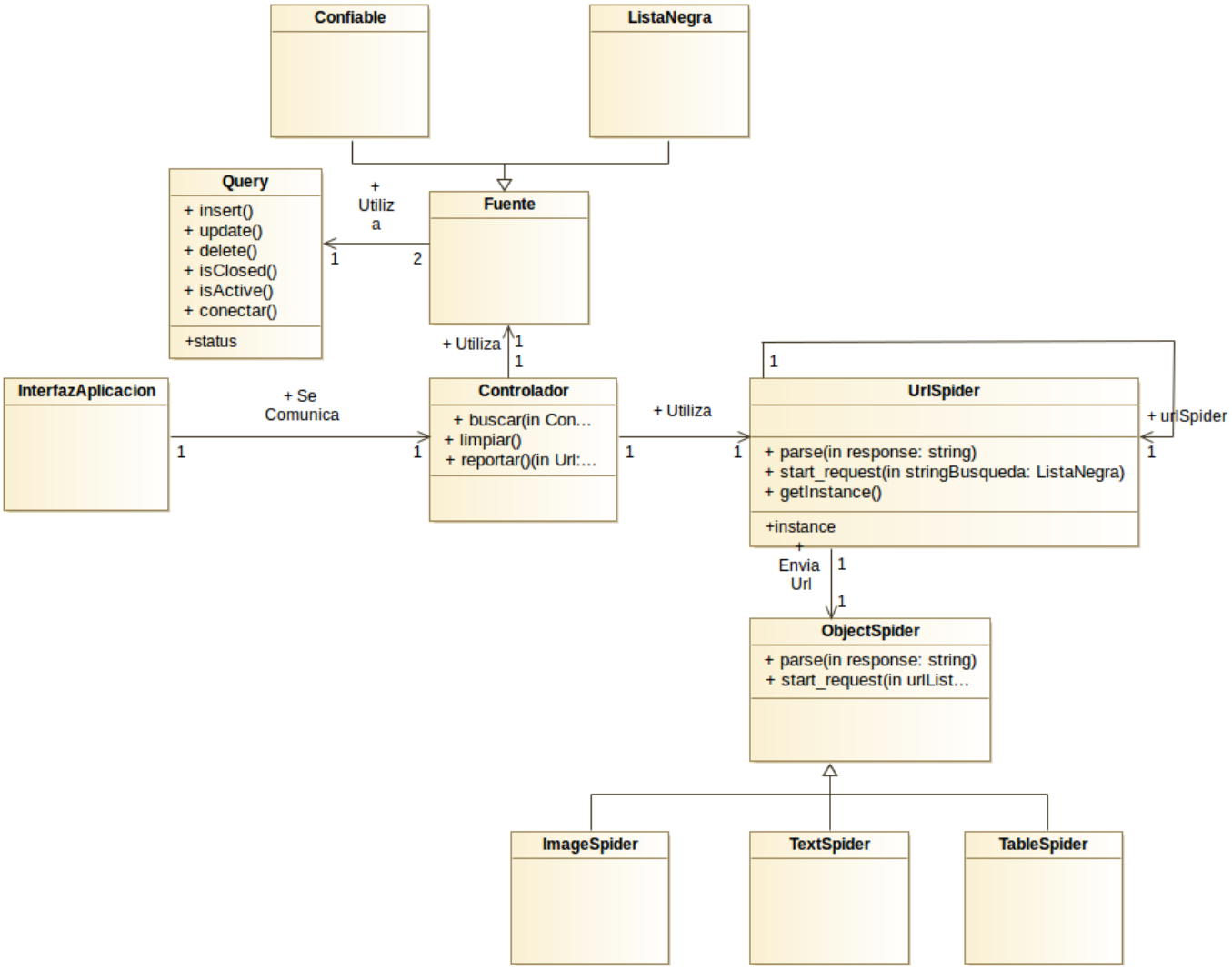


Figura 2: Diagrama de clases versión 1.2 Para ver con más detalle revisar el diagrama de clases en el proyecto de Modelio "Quantum Taco"

Para realizar una búsqueda sobre tópicos es necesario tener una clase que se encargue de encontrar las páginas exactas desde las cuales buscar, es por esta razón que se necesita las clases **UrlSpider** (encarga de recolectar url desde donde se realizará el scrapping) y las fuentes **confiables** (páginas a considerar) y **ListaNegra** (páginas no consideradas), dada la utilización de Singleton se necesita una interacción con si misma y el atributo de instancia a devolver.

Conectada a la clase UrlSpider se necesitan clases encargadas de recolectar los objetos de aprendizaje solicitados, por lo que es natural la aparición de las clases "**Spider**" (encargadas de recolectar los objetos).

Para el manejo de las fuentes se utiliza el patrón Active Record, por lo que se necesita una clase **Query** que interactué con esta para la interacción con la base de datos.

Por último, dada la decisión de utilizar MVC, es natural incluir las clases **controlador** y **InterfazAplicación** (vista).

4. Diagramas de secuencia

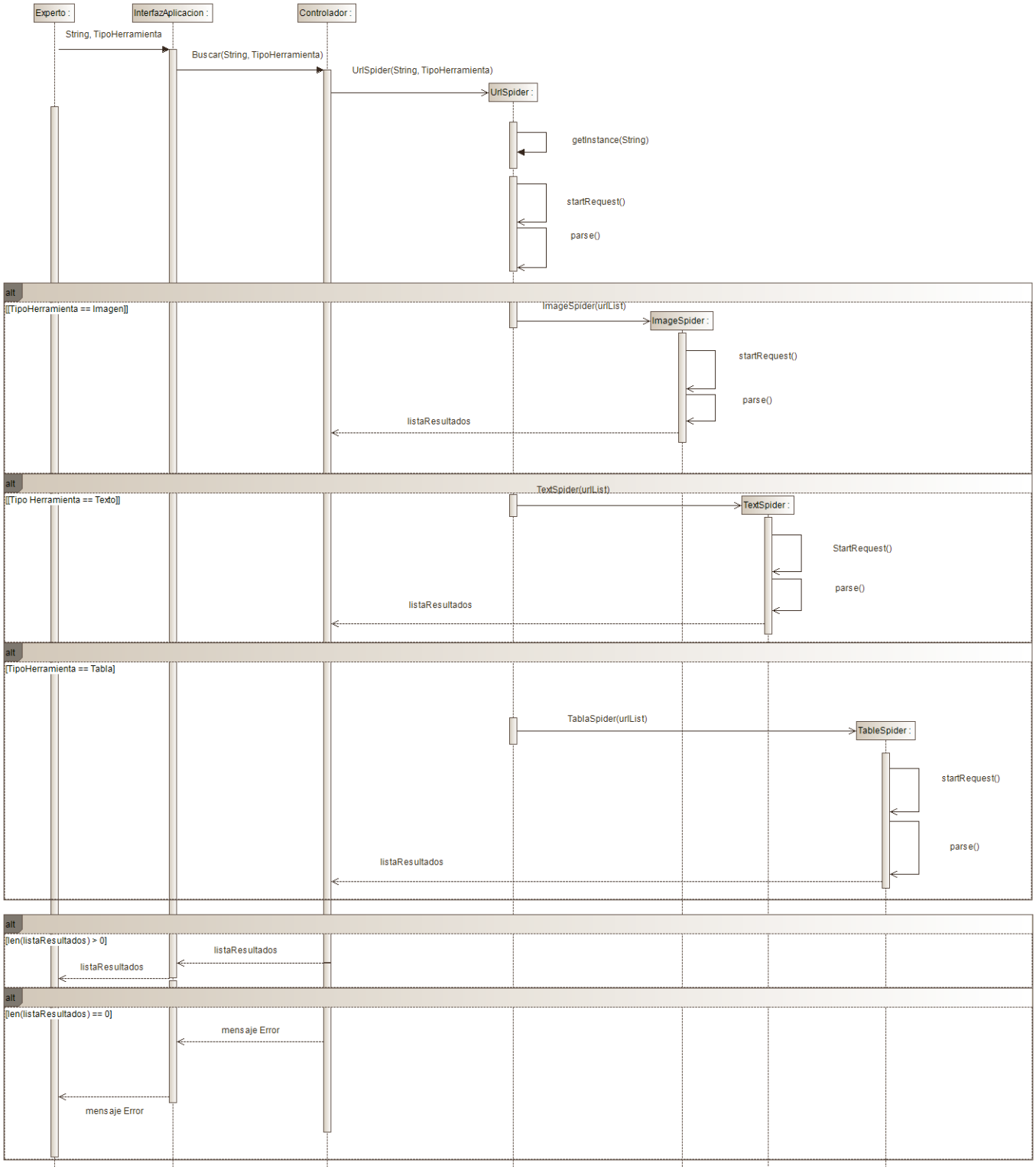


Figura 3: Diagrama de secuencia para el caso de uso "Buscar". Para ver con más detalle revisar el diagrama de secuencia en el proyecto de Modelio "Quantum Taco"

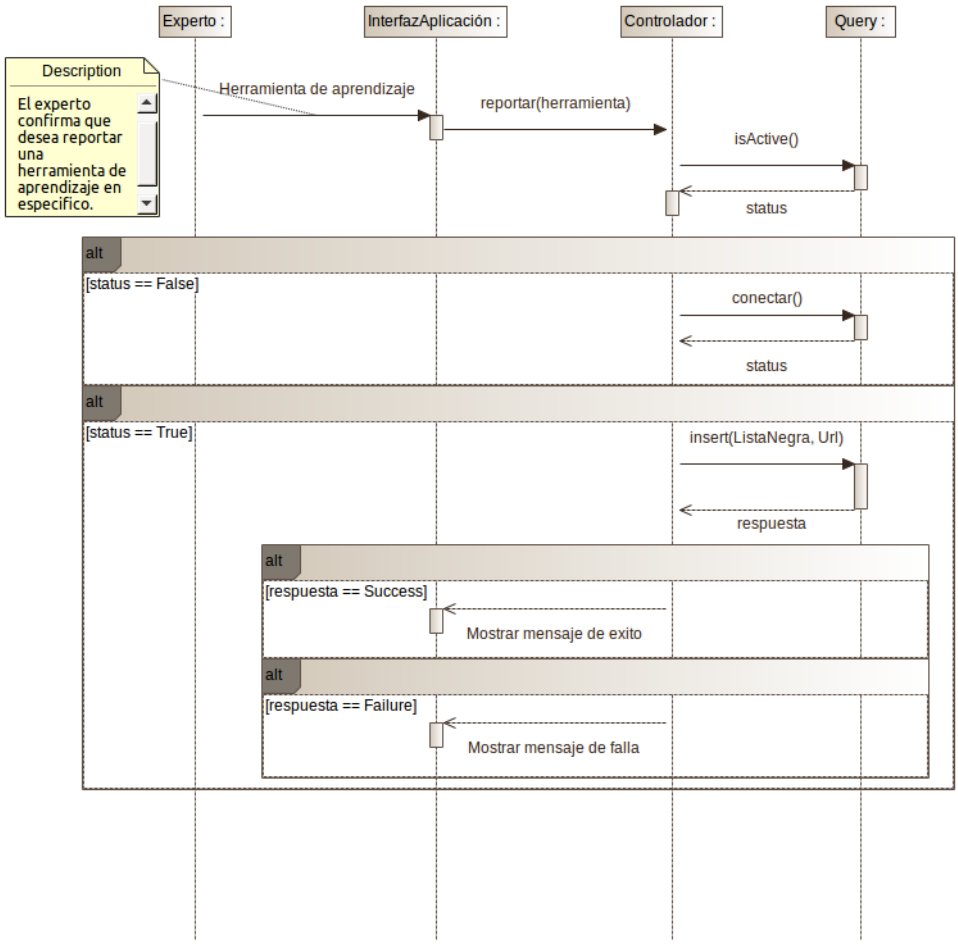


Figura 4: Diagrama de secuencia para el caso de uso "Reportar resultado". Para ver con más detalle revisar el diagrama de secuencia en el proyecto de Modelio "Quantum Taco"

5. Análisis de Trade–off

Dado que el cliente no especificó nada sobre requisitos no–funcionales durante su presentación y no lo mencionó al aprobar el documento inicial, no se consideran NFR. Sin embargo, para realizar este análisis se considerarán (n) criterios comunes en el desarrollo de software.

- **Pregunta**
  ¿Qué cambios se deben realizar en el sistema para que este aprenda de sus errores al realizar búsquedas de objetos de aprendizaje?
- **Opciones**
  - O_1**:** Lista Negra (idea actual)
  - O_2**:** Uso de Machine Learning
- **Criterios**
  - C_1**:** Mantenibilidad
  - C_2**:** Plan de recuperación ante desastres
  - C_3**:** Precio (H.H)
- **Análisis de Trade–Off**

Criterios/Opciones	O_1	O_2
C_1	$++$	-
C_2	$++$	-
C_3	$+$	–

- **Lista Negra**
  Esta opción presenta mucha mantenibilidad y gran recuperación ante desastres y un bajo costo en termino de horas y esfuerzo dedicado al desarrollo.

- **Machine Learning**
  Esta opción es muy desfavorable considerando los criterios utilizados. Es difícil detectar defectos en su codificación y solucionarlos, asimismo el recuperar el sistema ante un desastre mayor. También, en caso de que el usuario de confunda muchas veces, no es trivial hacer que la máquina desaprenda lo que ha aprendido.

  Por otro lado, el esfuerzo y tiempo dedicado para poner en marcha esta implementación es muy alto debido a los conocimientos del equipo de desarrollo.

En conclusión, la opción de desarrollar una lista negra es la opción seleccionada por el equipo, pues favorece los tres criterios seleccionados para realizar el análisis y cumple con la necesidad de aprender de los errores.