Inteligencia artificial aplicada a optimizar las adopciones de perros

Carolina Lauriano da Silva
Máster en Ingeniería Informática
Inteligencia Artificial

Samir Kanaan Izquierdo
Carles Ventura Royo

25/12/2016

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Inteligencia artificial aplicada a optimizar las adopciones de perros*
Nombre del autor:	*Carolina Lauriano da Silva*
Nombre del consultor/a:	*Samir Kanaan Izquierdo*
Nombre del PRA:	*Carles Ventura Royo*
Fecha de entrega (mm/aaaa):	12/2016
Titulación:	*Máster en Ingeniería Informática*
Área del Trabajo Final:	*Inteligencia Artificial*
Idioma del trabajo:	*Español*
Palabras clave	*Recomendador, mahout , adopciones*

Resumen del Trabajo (máximo 250 palabras):

En este proyecto final de master se pretende utilizar Inteligencia Artificial y algoritmos de recomendación ya existentes aplicados a la adopción de perros.

Solo en España hay más de 140 mil mascotas abandonadas al año, muchos de estos abandonos se podrían posiblemente ser evitados simplemente realizando un análisis del perfil del adoptante y recomendando una mascotas más afín a la personalidad, necesidades y estilo de vida del adoptante. Con este propósito se ha implementado un recomendador de perros según el perfil del adoptante.

He tenido esta idea inspirada por recomendadores tal como Tinder y Movielens. Dado que actualmente no existe ninguna aplicación que recomiende perros en concreto por afinidad y que la mayoría de paginas existentes son solo recomendadores de razas de perros o filtros directos de perros por sus características (como el matchota), he decidido implementar DogAdopt.

Se ha investigado las librerías más utilizadas actualmente de sistemas recomendadores existentes. He elegido la librería Mahout para implementar una aplicación que dado un posible adoptante de perro (usuario de Dogadopt), se obtiene las características claves principales a partir de un formulario y se recomienda las diez mascotas más similares, utilizando distancia euclidiana.

La suma de los valores de las características que superan el desvío máximo aceptado nunca supera el 21% de los valores recomendados.

Los puntos principales pendientes serian mejorar la aplicación web, para que los usuarios estuvieran mas cómodos utilizándola y conseguir una base de datos de perros reales en adopción para que se pudiera poner el proyecto en marcha.

Abstract (in English, 250 words or less):

In this project, we intend to use Artificial Intelligence and the existing recommender systems applied to the dog adoption problem.

Only in Spain more than 140 thousands pets are abandoned per year, many of those abandonment could be possibly avoided by simply analyzing the adopter's profile and recommending a pet more closely related to the adopter's personality, needs and lifestyle. For this purpose, a dog recommendation system has been implemented, recommending dogs according to the adopter's profile.

I had this idea inspired by recommenders systems such as Tinder and Movielens. And since there is currently no application that recommends dogs in particular by affinity and that the most existing websites are just dog breeds advisors or direct dog's filter by their characteristics (such as matchota), I have decided to implement DogAdopt.

To do so, I researched the most commonly libraries of recommender systems. I have chosen the Mahout library to implement an application that ,given a dog's adopter(a Dogadopt user), after he fills the form , the application extracts the most key features of his profile and recommends the most similar pets to his personality using Euclidean distance algorithm.

The sum of values of the characteristics that exceed the maximum accepted deviation never exceeds 21% of the recommended values.

The main points pending in this project are to improve the web application, so the final users would be more comfortable using it and to get a real dogs in adoption database so that the project could be put in the market.

Índice

Lista de figures

1. Introducción

El objetivo de esta sección es detallar el proyecto y los motivos para la elección de este tema en concreto y como se ha estructurado la realización de este trabajo

1.1 Contexto y justificación del Trabajo

Desde que he cursado la asignatura Inteligencia Artificial avanzada, he visto que aun quedan muchas áreas que podrían ser optimizadas con la ayuda de la inteligencia artificial y que se podría obtener un beneficio social inmenso.

Debido al creciente uso de algoritmos recomendadores para las más distintas áreas, como recomendación de películas, como el Movielens[1], o hasta parejas, como por ejemplo el Tinder, he identificado la necesidad de aplicar estos algoritmos en la adopción de mascotas.

España es el país con mas abandono de mascotas en Europa, con 150000 mascotas abandonadas a cada año [2]. Las protectoras de animales están con superpoblación y lo tienen cada vez mas difícil encontrar buenos adoptantes para las mascotas. El proceso de adopción normalmente es muy largo, donde las protectoras solicitan que los adoptantes rellenen un cuestionario socio cultural, y posteriormente realizan una entrevista.

Aun así, muchas de estas mascotas adoptadas, son posteriormente devueltas a la protectora porque no se encajaban correctamente con el perfil del adoptante. Por mucho que se intente, es muy difícil encontrar una mascota adecuada, porque hay detalles que puede que se escape y también porque quizás la mascota ideal se encuentra en una protectora que no está próxima, y por esto el adoptante termina con una mascota no adecuada para su perfil.

La idea del proyecto final era adaptar un algoritmo de sistema de recomendador para la recomendación de perros en adopción. A partir de una base de datos de perros en adopción (que en el futuro seria una base real de datos de perros en adopción de toda España), agrupada por características físicas (tamaño, color, pelo, etc) y por características de comportamiento (activo, social, tímido, miedoso), se procedería hacer un cuestionario al adoptante, donde se recomendaría las mascotas mas similares a los gustos y perfil del adoptante.

He investigado y no existe nada muy similar en el mercado. Aunque existen algunas páginas que recomiendan razas en concreto como la de pedigree[3] ,estas lo hacen de manera muy genérica, y no me parece muy correcto, ya que recomienda razas y no perros, y los perros de la misma raza pueden ser completamente distintos.

También he encontrado la página matchcota.com[4], un nuevo proyecto que tiene algunas similitudes como el proyecto que estamos desarrollando, pero donde el cuestionario para encontrar la "mascota ideal" es muy directo y simplificado, buscando mascotas solo por características y no por la afinidad con la personalidad y gustos del adoptante.

El diferencial de nuestro proyecto con la aplicación matchota.com, es que en el matchcota el usuario tiene que identificar exactamente las características del perro que quieres (pelaje, comportamiento, sexo, edad,etc) y en nuestro proyecto recomendarías perros basando solamente en el perfil del usuario.

1.2 Objetivos del Trabajo

Los objetivos que se desean conseguir en este proyecto son los siguientes

- Investigar los sistema de recomendación actuales y elegir el algoritmo/tecnología más apropiada
- Adaptar el algoritmo para el objetivo principal (desarrollar un sistema de recomendación para perros en adopción a partir de una base de datos)
- Desarrollar una aplicación web muy sencilla que seria la interfaz entre el usuario (posible adoptante) y nuestro recomendador.
- Crear un cuestionario donde sea posible determinar características fundamentales del usuario (posible adoptante) para posteriormente obtener su recomendación. No existirá un cuestionario directo donde el adoptante elija color/edad/sexo/raza y si elaboraremos preguntas donde intentaremos encontrar características claves de comportamiento del adoptante y sus gustos.
- Crear una base de datos con datos para realizar los experimentos. La idea es que en el futuro se consiga la colaboración de protectoras en distinta partes de España.
- Realizar pruebas con usuarios reales y comprobar la eficiencia del proyecto.
- Realizar toda la documentación necesaria.

1.3 Enfoque y método seguido

La estrategia principal es investigar los actuales sistema de recomendación existentes, elegir el más eficiente y apropiado para adaptarlo a la recomendación de perros en adopción y después desarrollar una aplicación web sencilla para la aplicación de algoritmo.

Posteriormente realizaremos testes unitarios y comprobaremos los resultados, para verificar que los resultados son los esperados.

Existe muchísimos algoritmos implementados en distintas lenguajes, y el objetivo será encontrar el mas eficiente y aplicarlo al nuestro contexto.

Posterior la investigación de todos algoritmos recomendadores existentes, he decidido utilizar el framework Apache Mahout de java para la implementación del sistema recomendador .

Para la creación de la pagina web utilizamos, en el backend java con Spring framework(spring mvc, spring data y spring boot) y en el frontend plantilla thymeleaf , jquery y bootstrap.

1.4 Planificación del Trabajo

Para desarrollar el proyecto, seguiremos las tareas como detalladas abajo.

Este diagrama de Gantt es solo una estimación de la planificación y estará sujeto a modificaciones durante el proyecto.

Name	Begin date	End date	Duration
DogAdoptProject	**9/23/16**	**12/27/16**	**68**
Planificacion	9/23/16	10/4/16	10
Estudiar documentacion TFM	9/23/16	9/26/16	4
Detallar tema TFM	9/23/16	9/25/16	3
Planificar proyecto	9/26/16	9/30/16	3
Redactar entrega 1	9/23/16	10/4/16	10
Desarrollo	10/7/16	10/31/16	19
Configuración de entorno de desarrollo	10/7/16	10/7/16	1
Analisis sistemas recomendadores existentes	10/7/16	10/9/16	3
Investigación de datos relevantes en la adopción de perros	10/10/16	10/14/16	3
Elaboración de cuestionario de adopción	10/15/16	10/17/16	3
Implementación de recomendador especifico	10/18/16	10/24/16	5
Implementación web sencilla para pruebas	10/11/16	10/30/16	14
Publicación de web en servidor publico	10/28/16	10/29/16	2
Redactar 2nda v memoria	10/7/16	10/31/16	19
Pruebas	11/4/16	11/28/16	19
Definición de pruebas	11/4/16	11/7/16	4
Ejecución de pruebas	11/8/16	11/21/16	10
Analisis de resultados	11/22/16	11/28/16	5
Redactar 3ra version memoria	11/4/16	11/28/16	19
Matenimiento de la web y solucionar posibles incidencias	11/4/16	11/28/16	19
Confección de Entrega final	12/2/16	12/24/16	17
Documentacion	12/2/16	12/9/16	6
Preparacion codigo fuente para la entrega	12/9/16	12/12/16	4
Presentacion virtual	12/12/16	12/19/16	6
Redactar version final memoria	12/2/16	12/24/16	17
Final TFM	12/27/16	12/27/16	1

Figura 1-Listado de tareas a realizar actualizado

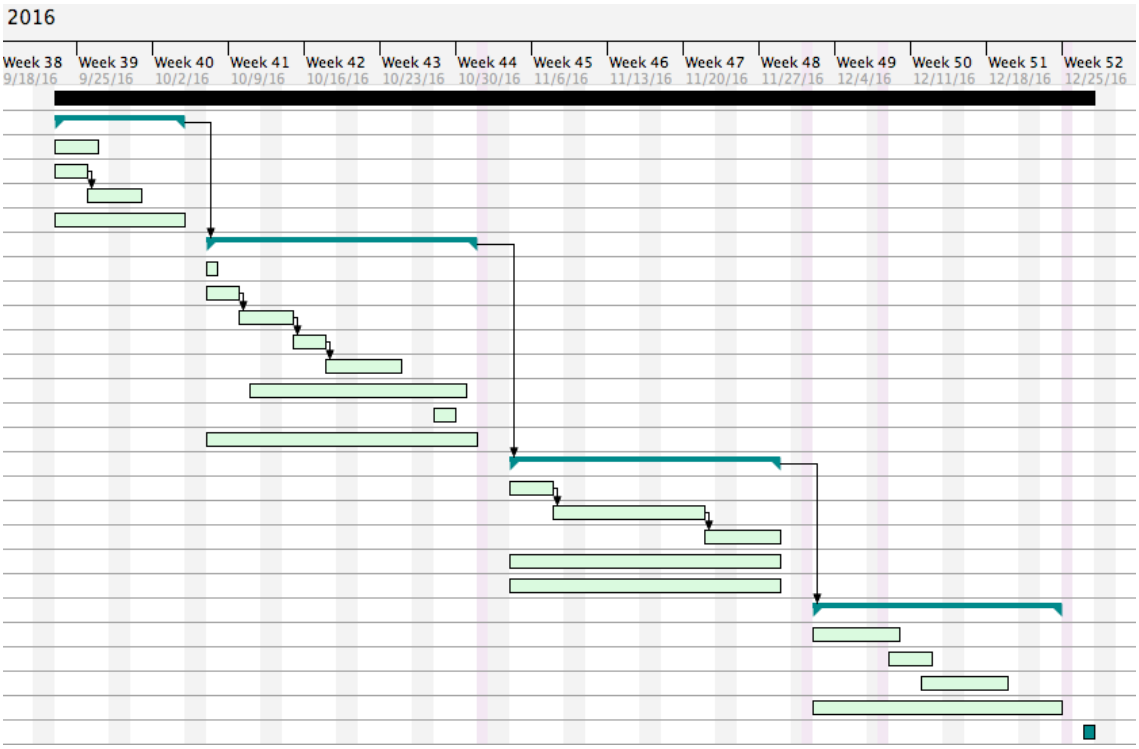
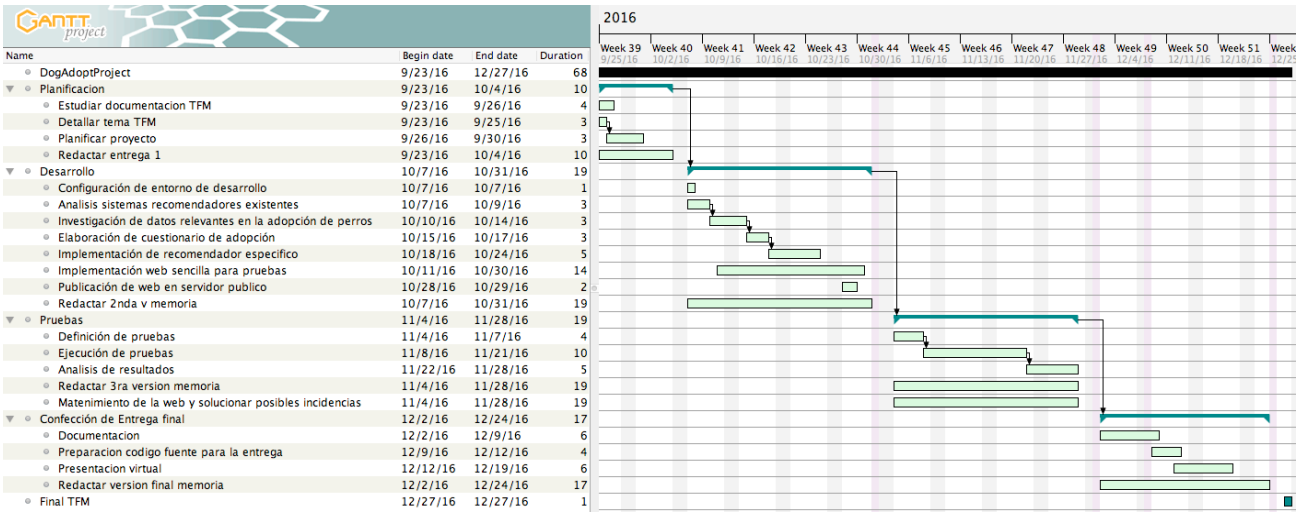


Figura 2-Diagrama Gantt actualizado



Name	Begin date	End date	Duration
DogAdoptProject	9/23/16	12/27/16	68
Planificacion	9/23/16	10/4/16	10
Estudiar documentacion TFM	9/23/16	9/26/16	4
Detallar tema TFM	9/23/16	9/25/16	3
Planificar proyecto	9/26/16	9/30/16	3
Redactar entrega 1	9/23/16	10/4/16	10
Desarrollo	10/7/16	10/31/16	19
Configuración de entorno de desarrollo	10/7/16	10/7/16	1
Analisis sistemas recomendadores existentes	10/7/16	10/9/16	3
Investigación de datos relevantes en la adopción de perros	10/10/16	10/14/16	3
Elaboración de cuestionario de adopción	10/15/16	10/17/16	3
Implementación de recomendador especifico	10/18/16	10/24/16	5
Implementación web sencilla para pruebas	10/11/16	10/30/16	14
Publicación de web en servidor publico	10/28/16	10/29/16	2
Redactar 2nda v memoria	10/7/16	10/31/16	19
Pruebas	11/4/16	11/28/16	19
Definición de pruebas	11/4/16	11/7/16	4
Ejecución de pruebas	11/8/16	11/21/16	10
Analisis de resultados	11/22/16	11/28/16	5
Redactar 3ra version memoria	11/4/16	11/28/16	19
Matenimiento de la web y solucionar posibles incidencias	11/4/16	11/28/16	19
Confección de Entrega final	12/2/16	12/24/16	17
Documentacion	12/2/16	12/9/16	6
Preparacion codigo fuente para la entrega	12/9/16	12/12/16	4
Presentacion virtual	12/12/16	12/19/16	6
Redactar version final memoria	12/2/16	12/24/16	17
Final TFM	12/27/16	12/27/16	1

Figura 3-Listado de tareas y Diagrama Gantt actualizado

1.5 Breve sumario de productos obtenidos

Hemos obtenido una investigación detallada sobre algoritmos recomendadores y una página web simple a ser mejorada el diseño y un backend, que dado un conjunto de perros y su característica, y un posible adoptante, lista los 10 perros más recomendables para este adoptante.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos detallaremos todo el trabajo realizado en este TFM.
Lo estructuraremos del siguiente modo:

- Descripción de la solución, donde detallaremos todos los pasos seguidos e investigaciones realizadas para la elección del algoritmo y desarrollo de la aplicación. Detallaremos también las dudas y como lo hemos solucionado.
  En este capitulo es donde detallamos todo lo que fue necesario para la implementación del proyecto en global.
- Análisis del resultado - analizaremos los resultados obtenidos y comprobaremos si el recomendador creado ha sido efectivo y ha cumplido los objetivos propuestos.
  En este capitulo validaremos si el proyecto realmente nos ha funcionado correctamente.
- Conclusión-Donde haremos un sumario de todos los resultados obtenidos y conclusiones de este TFM.

2. Definición de la solución

En este apartado se presentará los pasos seguidos para desarrollar el proyecto y toda la información colectada para que se pudiera tomar las mejores decisiones de implementación del proyecto. También se detallará los pasos seguidos para llegar a la solución.

2.1 Tareas previas

Antes de empezar el desarrollo del proyecto, ha sido necesario tomar distintas decisiones que serán detalladas en este capítulo. Detallaremos las posibles soluciones y como se ha decidido implementar el proyecto.

2.2 Definiciones del desarrollo

Para el desarrollo del proyecto, ha sido necesario elegir el lenguaje del sistema recomendador, framework de backend , frontend y base de datos.

2.2.1) Lenguaje del sistema recomendador

Para el desarrollo del sistema recomendado, he dudado si utilizar Python, que es el lenguaje por excelencia para Inteligencia Artificial, y era el lenguaje donde habíamos utilizado en la asignatura de Inteligencia Artificial avanzada o si utilizar Java, que es el lenguaje que trabajo diariamente.

Los motivos finales para elegir Java fueron:

- Por trabajar diariamente con Java desde hace 9 años, seria más fácil realizar la implementación del proyecto y debido al corto plazo en el cual se hay que realizar todo el desarrollo, esto ha sido un factor limitante.
- Existen muchas librerías de recomendadores ya implementadas y testeadas para ser utilizadas en java, las cuales serian bastante útiles y se podrían utilizar para la solución de este problema.

2.2.2)Backend

Una vez se había decidido utilizar Java, hemos utilizado Spring framework (spring mvc, spring data y spring boot) , ya que era un framework que había tenido poco contacto pero me parecía bastante interesante y muy practico para crear una aplicación con todas las capas. Como tenía interés en aprender más sobre este framework, he decidido utilizarlo.

2.2.3)Frontend

Para el frontend se ha decidido utilizar thymeleaf[5] como herramienta de *template* java, para mapeo de la parte grafica con el servidor, ya que esta bastante integrada con el Spring Framework .

Se ha utilizado el *framework* bootstrap para desarrollo *responsive* de nuestro proyecto. Como nuestro proyecto tiene la opción de listado y pagina, he adaptado la plantilla SB Admin 2[6] para el proyecto.

2.2.4)Almacenamiento de datos

Con respecto al almacenamiento de datos, habían muchas posibilidades pero la elección dependía en partes del algoritmo de recomendación que vayamos utilizar.

Para el desarrollo del proyecto con los datos ficticios, utilizaremos almacenamiento en memoria. Esta fue el almacenamiento elegido ya que el proyecto será desarrollado en Mahout, el cual es cargar los datos en memoria para posteriormente generar la recomendación.

2.3 Investigación de algoritmos de recomendación

Para desarrollar este proyecto existía la posibilidad de implementar el recomendador desde cero o utilizar alguna de las librerías/framework de recomendadores ya existentes, con lo cual nos permitía tener un algoritmo el más testeado posible y con código más maduro.

Para ello hemos investigado las distintas librerías que se podría utilizar en Java para adaptar al problema.

Las principales librerías encontradas fueron las detalladas abajo:

- **Mahout[7]:** Es un entorno de desarrollo simple y extensible y un framework para construir algoritmos escalables.

  El principal foco del Mahout es en el algoritmo, que es implementado de modo muy modular permitiendo que sea posible cambiar alguna parte fácilmente. Esta la librería de sistema recomendador probablemente la mejor para ser integrada directamente con aplicaciones java.

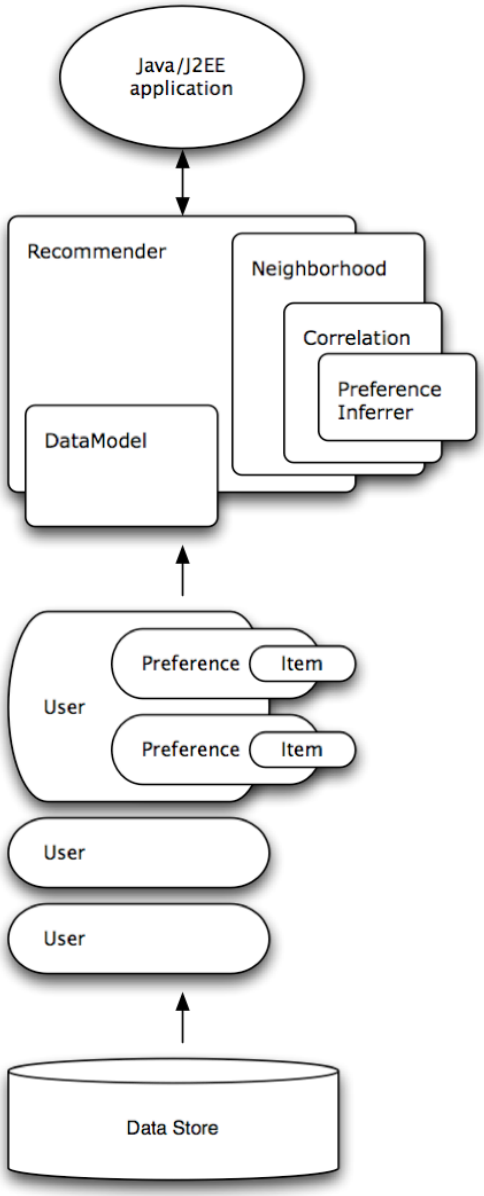


Figura 4-Flujo Mahout Recommender

Pros: su librería es muy fácil de utilizar y tienen prácticamente todos los métodos de similitud implementados.
La parte del sistema recomendador de Mahout tiene código bastante maduro y permite escalar fácilmente. Y es ideal que es basada en Hadoop(framework de software que soporta aplicaciones distribuidas). Como no es un producto cerrado, la documentación es escasa.
Es la librería recomendadora Java mas utilizada.

Cons: El DataModel de entrada tiene un formato especial, y habría que tratar todos los datos.
Se utiliza principalmente para los recomendadores online in-memory (es decir, las acciones del usuario afectan las recomendaciones de inmediato)

- **Librec [8] :**Es una librería java para sistemas recomendadores para solucionar el problema de predicción de valoración y puntuación de ítems.

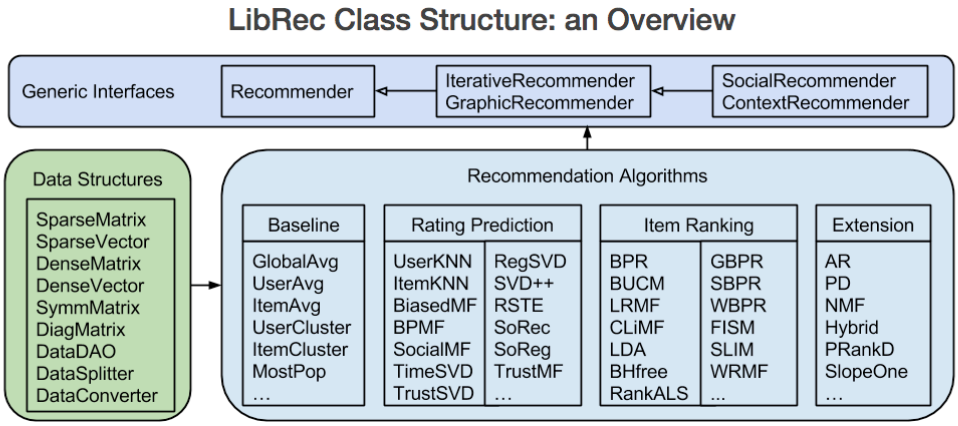


Figura 5-Estructura de clases LibRec

Pros: El LibRec es cross plataform, y puede ser desplegado en MS Windows, Linux y MacOS. Tiene ejecución rápida en un conjunto de datos del mundo real. Además de tener fácil configuración (por el fichero librec.conf).

Esta librería también es de fácil expansión, ya que tiene interfaces de recomendación con fácil expansión para nuevos recomendadores.

Cons: No es tan fácil su integración con Java.

- **Lenskit[9]:** Es una librería de recomendador muy fácil de utilizar , desarrollado por GroupLens.
  Esta librería fue desarrollada para ser utilizada para investigación [10] y académico. El objetivo del proyecto es producir un marco común de recomendación, reutilizable en aplicaciones, y con implementaciones claras y legibles de algoritmos comunes que emplean las mejores prácticas con respecto a estrategias de implementación, ajuste y normalizaciones.

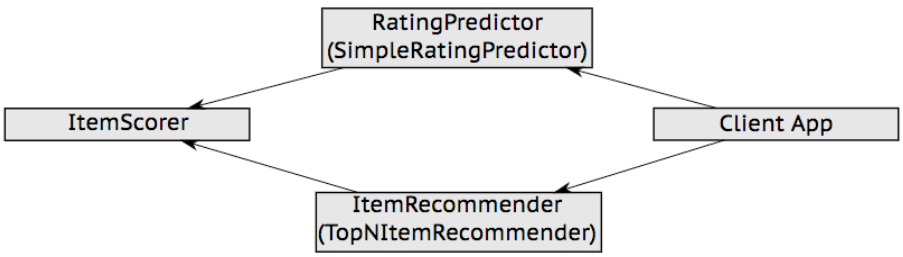


Figura 6-Flujo Lenskit

Pros: Es fácil de configurar con un proyecto Java Maven y bastante fácil de utilizar directamente de un proyecto java. El Lenskit ha sido ampliamente utilizado en el mundo académico.
Cons: No es tan adaptable como Mahout , ya que su estructura es más cerrada.

- **Easyrec[11]:** Es un sistema de recomendación *opensource*, donde puedes añadir a una aplicación un sistema de recomendación utilizando restful.

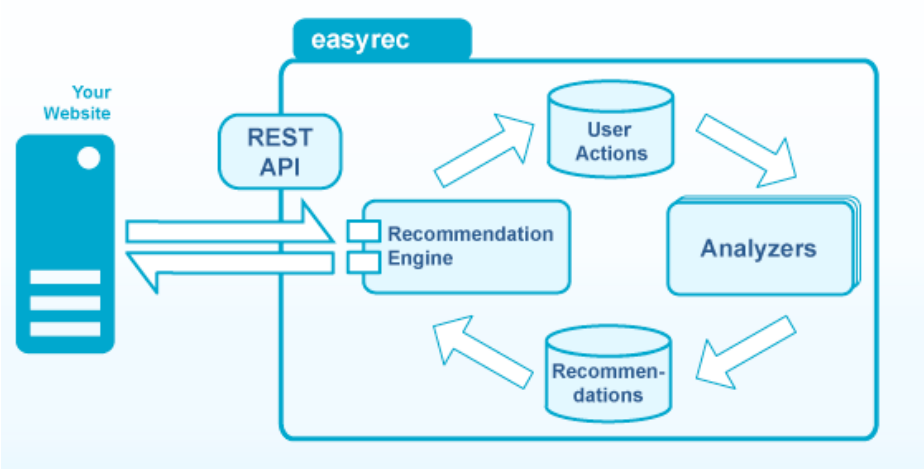


Figura 7-Flujo Easyrec

Pros: Se centra en la facilidad de instalación e integración a través del servicio Web REST.
Viene con una herramienta de administración basada en Web, un instalador, y tiene un sistema de complemento que le permite cargar / descargar *plugins* en tiempo de ejecución.
Es un recomendador *offline* - las recomendaciones se calculan sobre una base regular y son almacenados en la base de datos.

Cons: No es un sistema directamente integrado con Java (aunque que es de fácil acceso).

- **Duine framework[12]:** Es una colección de librería de software para recomendación y predicción.
  Tales predicciones se pueden utilizar para personalizar la información a los usuarios, específicamente en recomendar a los usuarios qué información es y no es de interés para ellos. El recomendador calcula las predicciones ejecutando algoritmos que razonan sobre el usuario actual y el elemento de información, resultando en un valor de predicción. Las técnicas de predicción usan perfiles de usuario e ítems de información como entrada para su cálculo.

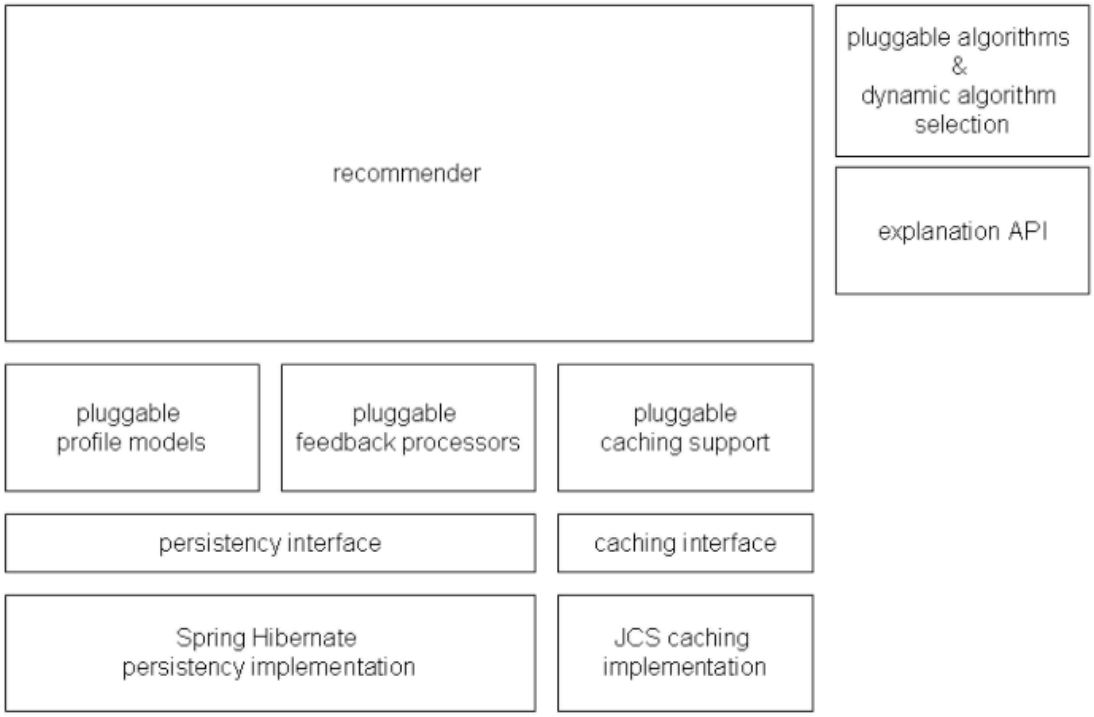


Figura 8-Overview de características de Duine

Pros: El recomendador Duine tiene capacidades de aprendizaje. Cuando un usuario califica un elemento, el recomendador extrae datos de este elemento (por ejemplo, palabras clave o géneros de una descripción de programas de televisión) para determinar los intereses del usuario. Mediante el uso de algoritmos inteligentes de aprendizaje el recomendador adapta ligeramente el perfil de usuario después de cada calificación, sobre la base de estos intereses. Además ofrece extensibilidad, validación y API de explicación (una API que explica para el usuario final como se ha obtenido el resultado de su recomendación).

Cons: No ha sido posible encontrar muchos tutoriales de cómo implementar el recomendador.

2.4 Apache Mahout

Para la implementación del recomendador utilizando la librería Mahout, se ha seguido los siguientes pasos:

- Transformar datos en DataModel de entrada siguiendo el patrón utilizado por Mahout.
- Elegir la métrica de similitud
- Elegir el número de vecinos que recomendaríamos al usuario.
- Obtener recomendación
- Recuperar detalles de cada perro del listado.

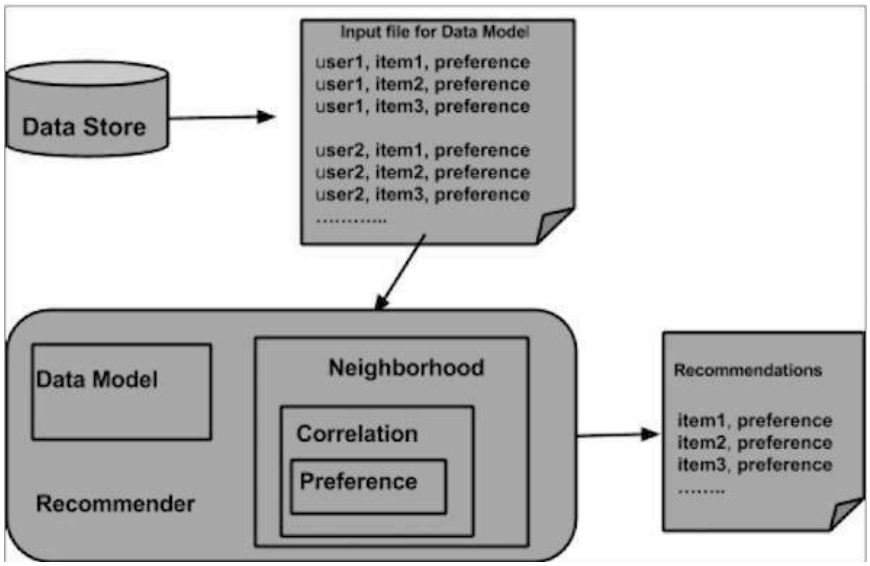


Figura 9-Arquitectura del sistema recomendador de Mahout

2.5 Elección de métricas de similitud

Para implementar un recomendador utilizando Mahout, el primer paso es transformar el DataModel en un UserSimilarity, utilizando una métrica de similitud, para posteriormente calcular los vecinos (UserNeighborhood).

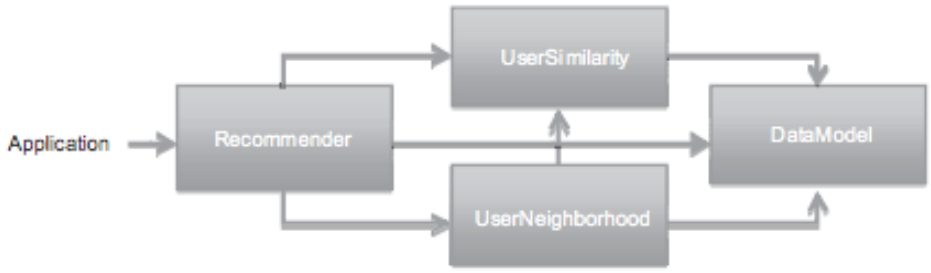


Figura 10-Dependencias en un sistema recomendador User-based y como se refresca la información

Los principales algoritmos métricas de similitud, son los siguientes:

- **Pearson correlation similarity:** es un número entre -1 y 1 que mide la tendencia de dos series de números, combinados uno a uno, seguir juntos. O sea, mide la probabilidad de que un numero de una serie sea relativamente grande cuando el numero de la otra serie sea grande, y viceversa. [13] Esto se puede ser utilizado directamente en nuestro problema, ya que queremos que un adoptante con por ejemplo, con valor activo 4 sea vecino de un perro con valor activo 4 también, y así sucesivamente para las demás características.

  Hemos realizado pruebas y hemos obtenido estos valores en este proyecto:

Características x Perros recomendados	Adoptante	RYJAP	BNAZAGQQZ	LNEWYKAH	MSXLLE	Luke
Similitud	100%	82%	76%	76%	75%	70%
Tamaño	**Pequeño-2**	**3**	**2**	**2**	**1**	**2**
Pelo	**Corto-1**	3	3	**1**	**2**	**1**
Color	**Claro-1**	**1**	**1**	3	**1**	**1**
Activo	**4**	**4**	**3**	**3**	**4**	**4**
Independiente	**1**	**1**	**1**	**2**	3	**1**
Sociable	**3**	1	**2**	**3**	**3**	**4**
Tolerancia al ruido	**4**	**4**	**3**	**4**	**3**	**4**
Necesidades Especiales	**4**	**4**	**4**	**4**	**3**	1
Cariñoso	**4**	**3**	**3**	**4**	**4**	**4**
Juguetón	**4**	**4**	**3**	**4**	**4**	**4**
Gastos	**1**	**1**	**2**	3	**2**	**1**

Figura 11-Tabla resultado de Características x Perros recomendados con UserSimilarity PearsonCorelation

En negrito se destaca los valores exactamente iguales o que varían en una unidad (lo que sería el equivalente que una variación de 25% del valor introducido por el usuario), y como podemos comprobar, el resultado obtenido es bastante satisfactorio, ya que prácticamente todos los valores de los perros recomendados se encajan en estos criterios, con excepción de 6 valores.

- **Euclidean distance similarity:** es una implementación basada en la distancia entre usuarios. Esta métrica calcula la distancia euclidiana entre dos puntos de usuarios. Este valor por si solo, no constituye una métrica valida de similitud, ya que los valores grandes significarían más distantes y menos similares (o sea, seria dos usuarios con distancia euclidiana 1 serian los más distantes).
  Para el cálculo de la similitud, la implementación devuelve 1/1+distanciaEuclidiana.

  Hemos vuelto a repetir las pruebas, introduciendo los mismos datos en el formulario, y hemos obtenido estos valores:

Características x Perros recomendados	Adoptante	HKCIAJH	VKVJHIB	WBUHHH	VECWNUPTZ	NZKUB
Similitud	100%	59%	55%	52%	51%	48%
Tamaño	**Pequeño-2**	**3**	**2**	**2**	4	**1**
Pelo	**Corto-1**	**1**	**1**	**2**	**2**	**2**
Color	**Claro-1**	**2**	**2**	3	**1**	**2**
Activo	**4**	**3**	**3**	2	**3**	**4**
Independiente	**1**	**1**	**1**	**2**	**1**	**1**
Sociable	**3**	**2**	**2**	1	**3**	**4**
Tolerancia al ruido	**4**	**3**	2	**4**	**3**	**4**
Necesidades Especiales	**4**	**3**	**4**	**4**	**3**	2
Cariñoso	**4**	**4**	**4**	**3**	**3**	**4**
Juguetón	**4**	**4**	**4**	**3**	**4**	**4**
Gastos	**1**	**1**	**2**	**1**	**1**	**2**

Figura 12-Tabla resultado de Características x Perros recomendados con UserSimilarity EuclideanDistanceSimilarity

Como podemos comprobar, los dos resultados son muy satisfactorios, ya que los perros recomendables son muy similares al adoptante.

Destacando en negrito los valores que son exactamente iguales que el del adoptante ideal o que varían solamente una unidad, podemos comprobar que el resultado es tan bueno como el obtenido con PearsonCorrelation.

Con el algoritmo Pearson el primer perro recomendado tiene siete valores exactamente iguales al perro ideal para el adoptante, aunque los otros cuatro valores tiene una pequeña diferencia.

Ya el algoritmo distancia Euclidiana, el primer perro recomendado tiene solo cinco valores exactamente iguales, pero los otros valores son mas similares con el adoptante (aunque no son exactamente iguales) que el Pearson correlation. Por este motivo hemos decidido utilizar el algoritmo Pearson correlation para que el primer perro recomendado sea mas similar posible al adoptante.

2.6 Diseño de la aplicación

Para el desarrollo del proyecto, se ha creado cuatro módulos:

- *Dogadopt-web-app-data:* modulo para controlar los datos, esencial para cuando se almacene los datos en base de datos.

- *Dogadopt-web-app-core:* donde hay toda la lógica del sistema recomendador y envío de email.

- *Dogadopt-web-app-api* : donde esta el *controller*.

- *Dogadopt-web-app-war:* modulo donde esta toda la parte gráfica (templates) y funciones comunes a toda la aplicación.

Se ha decidido desarrollar en distintos módulos para que fuera más independiente y más fácil de escalar en el futuro, si fuera el caso.

He estructurado la aplicación como el diagrama abajo. Todos estos módulos forman parte del proyecto dogadopt-web-app.

La aplicación tiene el siguiente flujo:

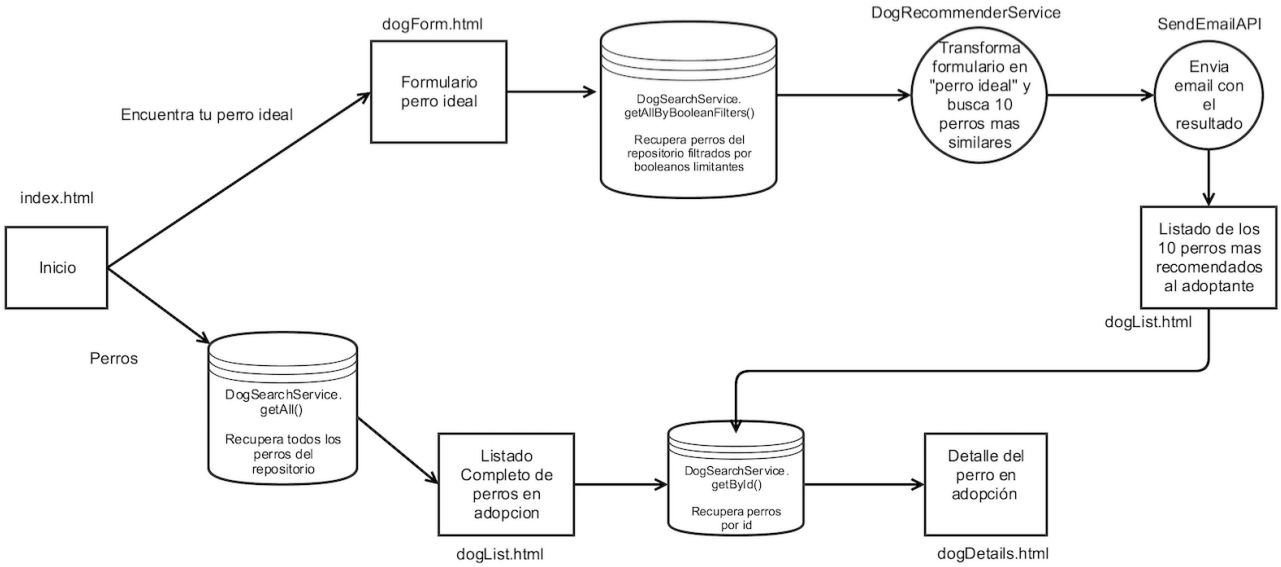


Figura 13-Flujo de la aplicación DogAdopt

He desarrollado así porque consideramos este el flujo mas claro para el usuario.

Los componentes y funcionalidades a destacar:

- Index.html: pantalla donde se inicia la aplicación.
- DogSearchService: clase donde se ha implementado todos los Servicios, como búsqueda de todos los perros (getAll), búsqueda por id(getById) y búsqueda filtrando con las características limitantes (getAllByBooleanFilters).
- dogList.html: listado completo de perros en adopción, donde se puede buscar por nombre o id, y ordenar. También permite clicar en el nombre del perro y acceder la pantalla dogDetails.html
- dogForm.html: esta pantalla esta detallada en la sección 2.7 y 2.8
- DogRecommenderService: clase donde hemos implementado el recomendador, la detallaré mejor en la sección 2.9.

- dogDetails.html: detalle del perro en adopción incluyendo nombre, foto,etc.
- SendEmailAPI: clase de envío de email para el posible adoptante y para projectdogadopt@gmail.com con finalidad de obtener los resultados y datos estadísticos.

2.7 Cuestionario

Para empezar a realizar el recomendador, estuve investigando varias páginas web de recomendación de razas, y he seleccionado las que me parecieron mas fiables [14],[3] para observar cuales criterio tenían en cuenta al recomendar una raza de perro.

Existen muchos recomendadores de razas, pero las que me han parecido más correctas [14][3] tenían en cuenta, las siguientes características:

- El nivel de energía y la cuantidad de actividad el adoptante tiene pensado en compartir con su mascota.
- El tamaño.
- Que tipo de compañero un adoptante busca.
- El nivel de inteligencia buscado en un perro.
- El nivel de paciencia del adoptante.
- La energía que transmite el adoptante.
- La importancia que el adoptante da a la suciedad generada por la caída de pelo y por la saliva de la mascota.
- El tiempo que el perro estaría solo en casa.
- La cantidad de dinero el adoptante tiene pensado en gastar en su mascota.
- Características limitantes: si hay niños, otros perros o alérgicos.

He observado también que muchas de estas páginas tenían como un criterio el tamaño del piso y de las áreas exteriores del piso/casa del adoptante, pero como he investigado, el tamaño del jardín no debería ser limitante, ya que todo perro debe realizar paseos diarios [18] y esto es mas importante que tener un gran jardín/piso [19].

También he investigado los principales motivos para abandonos de animales de compañía [15][16][17].
Los motivos comunes en todas las páginas que hemos encontrado han sido los siguientes:

- Los propietarios piensan que el perro ocasiona mucho trabajo y no tienen tiempo suficiente para entrenarlo.
- Los propietarios no pueden mantener financieramente la mascota.
- Los propietarios tiene que estar todo el día fuera de su piso por motivos adversos.
- El propietario tiene un hijo y no quiere o no tiene más tiempo para cuidar de su mascota.

- Alguien de la familia es alérgico y prefieren abandonar la mascota, en vez de tratar la alergia.
- El propietario no quiere hacer cargo de su mascota cuando se pone enfermo o se hace mayor.
- El propietario se cambia a un piso el cual no permite mascotas.
- El propietario muere o se pone enfermo.

Como los primeros seis primeros motivos se pueden detectar previamente antes de que se haga efectiva una adopción de un perro, creo que lo correcto seria enfocar en ellos para el suceso de este proyecto.

Los últimos dos motivos de abandono son factores que no se pueden controlar, y que pueden pasar a cualquiera, no lo tendremos en cuenta en este proyecto.

Basando en todo lo que hemos encontrado, hemos seleccionado las características principales de un perro y las hemos elegido las siguientes características.

Características físicas:
Estas características serán las que harán con que el perro en adopción sea "atractivo" visualmente por su propietario.

1) Pelo: corto, indiferente o largo (valor de 1 a 3)
2) Color: claro, indiferente o oscuro (valor de 1 a 3)
3) Tamaño : mini, pequeño, mediano, grande o maxi (valor de 1 a 5)

Características carácter(escala de 1-4 , donde 1 es nada y 4 es mucho):
Con estas características intentaremos detectar la mejor afinidad de personalidad entre el adoptante y su futura mascota
1) juguetón
2) sociable
3) independiente
4) activo
5) cariñoso
6) necesidades especiales
7) tolerancia a ruido
8) gastos

Características limitantes de convivencia:
Estas características son las que he investigado y que limitan la adopción de una mascota, y en vez de utilizarlas para el cálculo de similitud, las utilizaremos como filtro directo.
Si el valor para alguno de estos campos es "si", solo recomendaremos perros que tengan la misma característica con "si".
En el caso que el valor sea "no", lo tendremos en cuenta todos los perros, tanto que tengan "si" y los que tengan "no".
1) si hay niños : si o no

2) si hay otros perros : si o no
3) si hay gatos : si o no
4) si hay algún alérgico en la familia: si o no

Teniendo en cuenta estas categorías, hemos creado una función que nos genera perros aleatoriamente, simulando los datos que serian introducidos por las protectoras de animales en las cuales colaboraríamos en un futuro.

Hemos elaborado un cuestionario para intentar identificar de manera más honesta posible y sin influenciar, estas preferencias de los posibles adoptantes.

2.8 Mapping cuestionario a características del perro en adopción

He decidido mapear las características físicas y las características limitantes directamente con el formulario web, ya que lo más probable es que un posible adoptante ya conozca sus características físicas favoritas en los perros y también sepa en cuales factores limitantes hay que tener en cuenta.

Las características de carácter he considerado más correcto mapear indirectamente para que así las personas no se sientan influenciadas a poner un valor no real, de como gustarían ser en vez de como realmente son y que solo tengan que contestar preguntas objetivas.

Detallare abajo las preguntas elegidas y como la mapearemos para determinar las características de carácter de un adoptante (y por consecuencia, las de su mascota ideal).

Pregunta 1:
*En los fines de semana, me gusta: **
Mirar la tele y estar en casa (1)
Estar en la naturaleza (4)
Pasear por la ciudad (3)
Salir de fiesta (2)

Esta pregunta la mapearemos con la característica activo, con los valores entre paréntesis ().

Pregunta 2:
*Durante los días laborales, estoy fuera de casa: **
Trabajo en casa
Hasta 6 horas
Hasta 8 horas
Hasta 12 horas

Mapearemos este factor al tema de la dependencia del perro, ya que perros más independientes pueden estar más tiempo solos y tranquilos y no tienen tanta ansiedad por separación.

Pregunta 2.1:
*Tengo otros familiares que estarán en casa mientras estoy trabajando?: * Si/No*

He considerado que si la respuesta de esta pregunta es afirmativa, seria lo equivalente a "Trabajo en casa" en la pregunta anterior.

Pregunta 4:
*Tengo pensado gastar (en comida, peluquería y veterinarios): **
Hasta 20€ por mes aproximadamente
Hasta 60€ por mes
Hasta 80€ por mes
Hasta 100€ por mes
No me preocupan los temas financieros

Esta pregunta nos ayuda a determinar las limitaciones financieras del posible adoptante. Si el adoptante elige la opción "No me preocupa los temas financieros", atribuimos el valor 0 a esta opción y no la tendremos en cuenta para calcular la similitud (o sea, obtendremos tanto los perros que generan pocos gastos cuanto perros que generan muchos gastos)

Pregunta 5:
*Vivo en un piso/casa donde podría haber: **
Ningún ruido Mucho ruido

Con esta pregunta mapearemos el valor resultante en la de tolerancia a ruido que tiene el futuro adoptante.

Pregunta 7:
Me gusta jugar con perros: *
Nada Mucho

Esta pregunta la mapeamos con la característica “juguetón”.

Pregunta 6:
*Tengo tolerancia que mi perro me rompa algo en casa: **
Ninguna Mucha

Pregunta 8:
*Tengo mucha paciencia y tiempo para educar y cuidar de las necesidades especiales de mi perro: **
Nada Mucho

Realizando la media de los valores obtenidos en estas dos preguntas (pregunta 6 y 8) obtenemos la cantidad de atención a las necesidades especiales del perro el adoptante tiene pensado en tener.

Pregunta 9:
*Normalmente soy muy afectuoso con perros: **
Nada Mucho

Existen perros que necesitan constantemente afecto con humanos y otros que no tanto, y esta pregunta se mapea con la característica cariñoso.

Pregunta 3:
*Tenía pensado en ir al parque con mi perro: **
Hasta 3 horas aproximadamente por semana
Hasta 5 horas aproximadamente por semana
Hasta 7 horas aproximadamente por semana
Más de 7 horas por semana si fuera necesario

Pregunta 10:
*Diariamente estaríamos en contacto con: **
Ningún perro Muchos perros

Realizando la media de los valores obtenidos en estas dos preguntas (3 y 10), obtenemos el tan sociable debe ser un perro recomendado para este adoptante.

2.9 Implementación recomendador

He creado la clase `DogRecommenderServiceImpl.java` para recibir los datos del formulario y generar la recomendación.

Para esto he implementado un método utilitario "transformFormInDog" que dado los datos introducidos en el formulario, lo transforma en un perro ideal para el adoptante, mapeando las preguntas del formulario, como se comenta en la sección 2.8 .

También existe la variable result que es el resultado de todos los perros existentes, filtrados por los booleanos limitantes del formulario (tiene niños, perros, gatos y alergia).

Estos booleanos solo se tendrán en cuenta en el caso que sea afirmativo, como por ejemplo, una persona con alergia (tieneAlergia true) solo puede adoptar un perro apto para alérgicos, mientras una persona sin alergia puede adoptar un apto para alérgico o no.

He creado otra función utilitaria "transformDogsInAModel" que transforma el result (perros filtrados por los booleanos limitantes) en un GenericDataModel para ser posteriormente utilizado en Mahout. El perro ideal para el usuario generado en el método anterior será introducido en la posición 0 con ID 0 en este modelo.

Cada perro será el equivalente a un usuario de recomendador. Donde cada característica de un perro, por ejemplo Playful (juguetón) es como si fuera una película en el movielens, y hay una valoración de 1-4 para esta característica.

Entonces cada perro almacenado será un conjunto de valoración de cada característica, por ejemplo, Hair 1, Color 2, Playful 2, etc. Lo que sería el equivalente al movielens que un usuario haya valorado cada película con una nota.

	Película 1	Película 2	Película 3	…	Película n
Usuario 1	2	3	¿?	…	6
Usuario 2	¿?	4	3	…	¿?
Usuario 3	3	2	¿?	…	3
…	…	…	…	…	…
Usuario n	1	¿?	6	…	¿?

Figura 14-Tabla resultado valoración de películas de una web de recomendación

	Característica 1	Característica 2	Característica 3	…	Característica n
Usuario 1	4	3	2	…	1
Perro 1	3	2	2	…	3
Perro 2	4	2	1	…	1
…	…	…	…	…	…
Perro n	2	1	1		4

Figura 15-Tabla de resultados de características de DogAdopt

La tabla de resultado de la valoración de películas de una web recomendador , por ejemplo movielens (Figura 14) seria lo equivalente a (Figura 15) el DogAdopt.

El perro generado a partir del usuario que contesta el formulario también será transformado según lo que comentamos anteriormente.

Una vez tenemos todo el modelo creado, utilizamos el algoritmo PearsonCorrelation, como ya hemos comentado anteriormente en la sección 2.5, para agrupar los perros y obtener los vecinos.

Para cada elemento de la lista, recuperamos toda su información a partir de este ID, y así se obtiene los 10 perros más próximos al perro ideal para el usuario, y lo ordeno por similitud (afinidad), que seria el porcentaje de afinidad que el futuro adoptante tiene con las mascotas en adopción.

Este es el código obtenido:

```java
package com.lauriano.dogadopt.core.service.dog.impl;

import java.util.List;

@Service("dogRecommenderService")
public class DogRecommenderServiceImpl implements DogRecommenderService {
    private static final Logger log = LoggerFactory.getLogger(DogRecommenderServiceImpl.class);
    @Autowired
    @Qualifier("dogSearchService")
    protected DogSearchService dogSearchService;

    @Autowired
    @Qualifier("dogAdoptEmail")
    protected SendEmailAPI sendEmailAPI;

    @Override
    public List<DogContentItem> generateRecomendation(DogFormContentItem dogForm) {
        //  A partir del custionario, obtener un listado de perros recomndados
        //  Filtrar por valores discriminatorios
        final Boolean childrens = dogForm.isChildrens();
        final Boolean otherDogs = dogForm.isOtherDogs();
        final Boolean anyCats = dogForm.isAnyCats();
        final Boolean alergies = dogForm.isAlergies();
        final List<DogContentItem> result = dogSearchService.getAllByBooleanFilters(childrens, otherDogs, anyCats, alergies);

        DogContentItem user = transformFormInDog(dogForm);

        try {
            DataModel model = transformDogsInAModel(user,result);
            UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(model);
            UserNeighborhood neighborhood =
                      new NearestNUserNeighborhood(10, userSimilarity, model);
            long[] vecinos=neighborhood.getUserNeighborhood(0L);
            if (vecinos.length>0){
                final List<DogContentItem> resultFiltered = dogSearchService.getByIds(vecinos);
                for (DogContentItem dogItem : resultFiltered){
                    Long dogId = dogItem.getId();
                    Long elVecino = null;
                    for(Long vecino : vecinos) {
                        if(vecino.equals(dogId)) {
                            elVecino = vecino;
                            break;
                        }
                    }
                    if(elVecino!=null) {
                        dogItem.setSimilarity((int)( (userSimilarity.userSimilarity(0L, elVecino))*100));
                    }
                }
                sendMail(dogForm, user,resultFiltered);
                return resultFiltered;
            }
        } catch (TasteException e) {
            log.info(e.toString());
        }
        return result;
    }
```

Figura 16-Código fuente DogRecommenderServiceImpl

3. Análisis del resultado

En esta sección se analizara los resultados obtenidos y validaremos la calidad del recomendador.

3.1 Datos de pruebas

Para que fuera posible comprobar si el sistema recomendador funcionaba correctamente, he introducido datos de 11 perros reales en la aplicación y he solicitado que sus amos/propietarios realizaran el test.

Se ha solicitado que 10 personas probaran la página y contestaran el cuestionario. Un correo fue enviado a cada uno de este posible adoptante con los 10 perros más recomendados (los resultados obtenidos en nuestro recomendador).

Para averiguar los resultados obtenidos, también he implementado para que se envíe un correo al projectdogadopt@gmail.com con los mismos datos enviados al posible adoptante y además otras informaciones como: el perfil del adoptante (características que el adoptante ha introducido en el formulario) y el perfil de los 10 perros más recomendados para cada persona (características que los perros recomendados tenían).

3.2 Criterios de pruebas

Hemos solicitado 10 personas para que realizaran el test. Se ha intentado elegir personas con perfiles bastante distintos, para que fuera posible encontrar posibles errores.

3.3 Criterios para validación de calidad de los resultados

Para comprobar la calidad del recomendador, he analizado los resultados obtenidos para 10 usuarios reales y los resultados recomendado que tienen desvío de más de 25% de los valores del usuario.

Como la mayor parte de las características tienen valor entre 1-4 (juguetón, sociable, independiente, activo, cariñoso, necesidades especiales, tolerancia a ruido y gastos), teniendo algunas de 1-3 (pelo y color) y una de 1-5(tamaño), hemos utilizado las siguientes métricas:

Para los valores que varían de 1-4, cuando el valor obtenido era exactamente igual que el introducido por el usuario, el desvío es 0%.
Si el valor variaba en una unidad, el desvío es 1 dividido por 4, o sea, 25%, donde esta variación podría ser positiva o negativa. Como por ejemplo si el usuario ha introducido “activo=3”, se considera desvío de mas de 25% si el valor del perro obtenido por el recomendador es diferente de 3 (desvío 0%), 4 o 2 (desvío de 25%).

Para las características que tienen varían de 1-5 y 1-3, el desvío de más de 1 unidad seria el equivalente a 20% o 33% de desvío respectivamente.

Como los valores son enteros, tendremos en cuenta el desvío de 20% para las características que varían de 1-5, 25% para características de 1-4 y 33% para las características de 1-3.

Para el valor gastos, si el valor introducido para el usuario es 0, no tendremos en consideración los valores de los perros recomendados en estas características.

3.4 Resultados

3.4.1 Usuario 1

Características limitantes: otros perros.

Resultados con desvío superior al determinado en [3.3]: 23 de 110 equivalentes a 21%

Resultado obtenidos:

67-IWCBVYM
65-ETDIKSOA
63-VAQKGIZ
63-DFBPM
63-COWWCJ
55-QKFHZQWAU
50-MTHJTSXG
49-FXTOCVKUQ
48-KEQAOBZL
47-Luke

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	67%	65%	63%	63%	63%	55%	50%	49%	48%	47%
Pelo	3	3	2	2	2	2	2	3	2	2	**1**
Color	2	1	2	2	2	1	1	3	2	2	1
Tamaño	3	**1**	2	**5**	**1**	2	3	**1**	3	4	2
Juguetón	4	4	3	3	**2**	4	3	4	4	4	4
Sociable	2	3	1	1	2	1	2	3	1	2	**4**
Independiente	1	1	**3**	2	1	2	1	1	1	1	1
Activo	4	4	4	4	3	4	**2**	3	3	4	4
Cariñoso	4	3	4	4	4	3	4	3	**2**	**2**	4
Necesidades Especiales	3	**1**	3	**1**	4	2	**1**	**1**	2	**1**	**1**
Tolerancia al ruido	4	3	4	4	4	**2**	**1**	3	**1**	**2**	4
Gastos	4	3	4	4	3	**2**	4	3	**2**	**2**	**1**

Figura 17-Tabla resultado Usuario 1

3.4.2 Usuario 2

Características limitantes: otros perros y gatos.
Resultados con desvío superior al determinado en [3.3]: 14 de 110 equivalentes a 13%

Resultado obtenidos:

78-HJMITPA
76-HLYADK
70-RJMCMVC
65-YLOJLBJV
65-AXXEB
63-POFKNIHGB

59-Luke
58-GXNQIBEN
56-TUFRI
54-ZZIKI

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	78%	76%	70%	65%	65%	63%	59%	58%	56%	54%
Pelo	2	2	1	3	1	3	3	1	1	3	2
Color	2	1	3	2	3	3	2	1	2	2	1
Tamaño	4	5	5	5	4	4	4	**2**	**1**	4	3
Juguetón	3	2	3	3	3	**1**	3	4	**1**	**1**	**1**
Sociable	2	**4**	1	2	2	3	2	**4**	1	3	1
Independiente	2	2	2	3	1	2	**4**	1	2	3	2
Activo	4	4	3	3	**2**	4	3	4	4	4	3
Cariñoso	4	4	4	4	3	4	4	4	3	4	4
Necesidades Especiales	3	3	2	2	4	2	**1**	**1**	2	3	**1**
Tolerancia al ruido	4	4	3	4	3	3	4	4	3	3	**1**
Gastos	0	1	1	2	1	1	1	1	1	2	1

Figura 18-Tabla resultado Usuario 2

3.4.3 Usuario 3

Características limitantes: ninguno.
Resultados con desvío superior al determinado en [3.3]: 11 de 110 equivalentes a 10%

Resultado obtenidos:

86-JAFIV
79-Bruno
76-Tuco
75-TPTMCTN
71-WUWRZUC
70-YLIWXUMU
70-HSCQLGNVV
66-Nami
62-WYGZX
61-LKYQPNQCM

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	86%	79%	76%	75%	71%	70%	70%	66%	62%	61%
Pelo	2	2	1	2	2	3	2	3	1	3	3

Color	2	3	1	3	1	2	1	1	1	1	1
Tamaño	2	**4**	2	2	2	2	3	**4**	2	**4**	1
Juguetón	4	3	4	3	3	3	3	4	4	4	4
Sociable	3	3	**4**	3	3	2	3	2	**4**	3	2
Independiente	3	3	3	2	4	3	2	3	2	4	**1**
Activo	4	4	4	4	**2**	3	4	4	4	4	4
Cariñoso	4	4	4	4	4	4	**2**	4	4	3	4
Necesidades Especiales	4	**1**	**2**	3	4	3	4	4	**1**	4	3
Tolerancia al ruido	2	1	2	2	1	3	2	3	2	1	3
Gastos	1	2	1	2	1	1	1	1	1	2	2

Figura 19-Tabla resultado Usuario 3

3.4.4 Usuario 4

Características limitantes: ninguno.
Resultados con desvío superior al determinado en [3.3]: 21 de 110 equivalentes a 19%

Resultado obtenidos:

76-XOZGGHR
76-HSCQLGNVV
75-Luke
71-HLYADK
71-DFBPM
70-DDGSKVX
65-LKYQPNQCM
64-WUWRZUC
64-RJMCMVC
62-HJMITPA

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	76%	76%	75%	71%	71%	70%	65%	64%	64%	62%
Pelo	1	1	**3**	1	1	2	1	**3**	**3**	**3**	2
Color	1	**3**	1	1	**3**	1	2	1	2	2	1
Tamaño	3	3	4	2	**5**	2	3	**1**	2	**5**	**5**
Juguetón	4	4	4	4	3	4	**2**	4	3	3	**2**
Sociable	1	1	2	**4**	1	1	2	2	2	2	**4**
Independiente	1	1	**3**	1	2	2	**3**	1	**3**	**3**	2
Activo	4	4	4	4	3	4	3	4	3	3	4
Cariñoso	4	3	4	4	4	3	4	4	4	4	4
Necesidades	2	2	**4**	1	2	2	3	3	3	2	3

Especiales											
Tolerancia al ruido	4	**2**	3	4	3	**2**	4	3	3	4	4
Gastos	0	1	1	1	1	2	1	2	1	2	1

Figura 20-Tabla resultado Usuario 4

3.4.5 Usuario 5

Características limitantes: otros perros y niños.

Resultados con desvío superior al determinado en [3.3]: 24 de 110 equivalentes a 21%

Resultado obtenidos:

84-Luke
69-Jacob
61-WQUTOFAQ
61-HJMITPA
58-OJYFAJMJ
58-Bruno
52-VAQKGIZ
52-QDBFTKGVA
50-DFBPM
48-Duna

	User	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Similitud	100%	84%	69%	61%	61%	58%	58%	52%	52%	50%	48%
Pelo	2	1	1	1	2	2	1	2	2	2	2
Color	1	1	1	2	1	2	1	2	2	1	1
Tamaño	3	2	3	2	**5**	**1**	2	**1**	**1**	2	4
Juguetón	4	4	4	3	**2**	4	4	**2**	4	4	4
Sociable	4	4	4	3	4	4	4	**2**	3	**1**	4
Independiente	1	1	2	2	2	1	**3**	1	1	2	4
Activo	4	4	4	2	4	2	4	**3**	4	4	**3**
Cariñoso	4	4	3	3	4	4	4	4	**2**	3	4
Necesidades Especiales	3	**1**	**1**	3	3	**1**	2	4	4	2	**1**
Tolerancia al ruido	4	4	**2**	4	4	3	**2**	4	**2**	**2**	4
Gastos	3	**1**	2	3	**1**	4	**1**	3	3	**1**	3

Figura 21-Tabla resultado Usuario 5

3.4.6 Usuario 6

Características limitantes: ninguna.

Resultados con desvío superior al determinado en [3.3]: 7 de 110 equivalentes a 6%

Resultado obtenidos:

82-RDUYT
82-LKGXXM
78-JRHZEX
75-JQSFZF
73-Tuco
70-JWRUIAX
69-Luke
68-WCEDSFHS
66-BCBUTX
65-GPCWQU

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	82%	82%	78%	75%	73%	70%	69%	68%	66%	65%
Pelo	1	1	1	2	2	2	2	1	1	**3**	2
Color	2	1	2	1	3	3	2	1	1	2	2
Tamaño	2	2	2	1	3	2	1	2	**4**	2	1
Juguetón	3	4	2	3	**1**	3	3	4	4	4	3
Sociable	2	2	3	2	1	3	2	**4**	2	3	1
Independiente	1	2	2	1	**4**	2	2	1	**3**	1	2
Activo	3	4	3	4	3	4	3	4	3	2	3
Cariñoso	4	4	4	4	3	4	4	4	4	4	3
Necesidades Especiales	3	3	3	3	4	3	4	**1**	3	3	2
Tolerancia al ruido	3	4	4	3	2	2	2	4	4	4	4
Gastos	1	2	1	2	2	2	2	1	1	2	1

Figura 22-Tabla resultado Usuario 6

3.4.7 Usuario 7

Características limitantes: gatos.

Resultados con desvío superior al determinado en [3.3]: 11 de 110 equivalentes a 10%

Resultado obtenidos:

72-Luke
65-Nami
64-YFEMPEFT
64-DTRNGPXLT
59-Tuco
59-DWPWAUMZZ
56-Jacob
52-EWUPQ
49-COIWV
49-BXUQBBVVO

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	72%	65%	64%	64%	59%	59%	56%	52%	49%	49%
Pelo	2	1	1	3	3	2	1	1	3	3	1
Color	2	1	1	2	2	3	2	1	2	1	3
Tamaño	3	2	2	3	3	2	**5**	3	**5**	2	**5**
Juguetón	3	4	4	4	4	3	**1**	4	3	**1**	3
Sociable	4	4	4	4	4	3	4	4	**1**	3	**2**
Independiente	1	1	2	2	2	2	2	2	1	**3**	2
Activo	4	4	4	3	3	4	3	4	4	4	4
Cariñoso	3	4	4	2	2	4	2	3	4	4	2
Necesidades Especiales	3	**1**	**1**	4	4	3	3	**1**	3	4	2
Tolerancia al ruido	3	4	2	3	3	2	3	2	3	3	2
Gastos	0	1	1	2	2	2	1	2	1	1	1

Figura 23-Tabla resultado Usuario 7

3.4.8 Usuario 8

Características limitantes: otros perros y niños.

Resultados con desvío superior al determinado en [3.3]: 13 de 110 equivalente a 11%

Resultado obtenidos:

82-Tuco
66-CSHEELRPK
63-Jacob
61-GUUZLYVU
59-Bruno

56-WZROH
52-EQRTAIMT
51-Luke
51-Berri
49-YKBVXGF

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	82%	66%	63%	61%	59%	56%	52%	51%	51%	49%
Pelo	1	2	1	1	1	1	1	2	1	2	1
Color	2	3	3	1	3	1	3	1	1	3	2
Tamaño	2	2	2	3	3	2	**4**	**4**	2	**4**	3
Juguetón	2	3	1	**4**	1	**4**	**4**	1	**4**	**4**	2
Sociable	3	3	4	4	4	4	4	4	4	4	3
Independiente	1	2	1	2	1	**3**	1	1	1	**3**	1
Activo	3	4	4	4	3	4	3	2	4	4	**1**
Cariñoso	3	4	2	3	2	4	2	3	4	3	4
Necesidades Especiales	2	3	1	1	1	2	**4**	1	1	1	1
Tolerancia al ruido	1	2	2	2	2	2	1	1	**4**	2	2
Gastos	2	2	3	2	2	1	1	1	1	3	3

Figura 24-Tabla resultado Usuario 8

3.4.9 Usuario 9

Características limitantes: ninguna.

Resultados con desvío superior al determinado en [3.3]: 16 de 110 equivalente a 14%

Resultado obtenidos:

72-Tuco
70-CSLLX
60-Luke
57-PDKHZMOCO
53-AKZWX
52-ZIWTEG
51-Nami
51-Bruno
50-MUANTTDAG
50-AFEANF

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	72%	70%	60%	57%	53%	52%	51%	51%	50%	50%
Pelo	1	2	2	1	1	1	1	1	1	1	2
Color	2	3	3	1	2	2	2	1	1	3	2
Tamaño	2	2	3	2	**4**	**5**	1	2	2	2	2
Juguetón	3	3	4	4	3	2	2	4	4	**1**	2
Sociable	1	**3**	1	**4**	1	1	2	**4**	**4**	2	1
Independiente	1	2	**4**	1	2	**3**	1	2	**3**	1	**4**
Activo	4	4	3	4	3	4	3	4	4	**2**	4
Cariñoso	4	4	4	4	**2**	3	3	4	4	4	4
Necesidades Especiales	3	3	4	**1**	4	4	4	**1**	2	3	3
Tolerancia al ruido	3	2	4	4	2	4	**1**	2	2	4	2
Gastos	0	2	1	1	1	2	2	1	1	2	2

Figura 25-Tabla resultado para Usuario 9

3.4.10 Usuario 10

Características limitantes: ninguna.

Resultados con desvío superior al determinado en [3.3]: 14 de 110 equivalente a 13%

Resultado obtenidos:

80-CHXCR
76-SLUIX
75-ANNETCMA
71-OJUWUC
71-NQGXGVKP
68-KVHBUGEN
67-JFFLE
66-DNJUPE
66-AUDBT
64-ICIMYDY

	User	1	2	3	4	5	6	7	8	9	10
Similitud	100%	80%	76%	75%	71%	71%	68%	67%	66%	66%	64%
Pelo	1	1	1	2	1	**3**	**3**	2	2	**3**	1
Color	2	2	2	1	2	2	2	1	2	2	2
Tamaño	1	1	1	2	1	2	1	1	1	**3**	2
Juguetón	4	3	4	4	3	4	4	4	4	4	4
Sociable	3	4	4	3	4	4	3	4	4	4	3

Independiente	1	1	**3**	1	**3**	2	**3**	1	**3**	2	1
Activo	1	2	1	1	2	**3**	1	**3**	2	1	1
Cariñoso	3	3	4	4	4	4	3	3	3	4	4
Necesidades Especiales	3	4	3	2	4	4	4	3	2	4	**1**
Tolerancia al ruido	2	1	1	2	3	**4**	2	1	3	1	**4**
Gastos	2	2	1	3	3	**4**	2	1	2	2	3

Figura 26-Tabla resultado para Usuario 10

3.5 Pruebas de rendimiento

He ejecutado la web DogAdopt, introduciendo los mismos datos en el formulario para distintos tamaños de conjunto de perros en adopción.

Con la función abajo compramos la eficiencia de la obtención de resultados en conjuntos de datos de 50, 100, 300, 500, 1000 y 1500 perros en adopción.

```
time_start = System.currentTimeMillis();
CODIGO DE GENERACION DE DATAMODEL, RECOMENDADOR y ENVIO DE EMAIL
time_end= System.currentTimeMillis();
log.info("the task has take" + (time_end - time_start ) + "milliseconds");
```

Figura 27-Función utilizada para calcular el tiempo

Los resultados obtenidos fueron muy aceptables, no tardando más de 4005 milisegundos para hasta 1500 perros.

Numero total de perros	Tiempo en milisegundos
50	2439
100	2459
300	3145
500	4005
1000	3679
1500	3807

Figura 28-Tabla resultado de pruebas de rendimiento

Como podemos comprobar, el tiempo de respuesta con hasta 1500 perros es bastante aceptable.

Estas pruebas han sido ejecutado en mi ordenador , que es un iMac con OSX El Capitán, Memory 8GB y procesador 2,7GHz Intel Core i5.

3.6 Estadísticas y consideraciones finales

Teniendo en cuenta las pruebas con usuario, los resultados obtenidos han sido bastante satisfactorios.

Figura 29-Gráfico Usuario x desvío superior al % determinado

Comparando con lo actualmente existe en el mercado, que seria el matchota.com , que es una web donde hay filtros directos para seleccionar mascota, nuestro proyecto ha obtenido mejores resultados ya que la búsqueda no ha sido directa y ha encontrado resultados más similares.

Me he dado cuenta que la calidad del resultado es afectada por el número de condicionantes limitantes (si tiene perro, alergia, hijos o gatos) se ha contestado con afirmativo, ya que el recomendador solo tienen en cuenta los perros después de limitarlos por los condicionantes limitantes.

La calidad también es afectada si el valor introducido por el usuario es el máximo o el mínimo (ejemplo: 1 o 4), ya que es más fácil obtener un desvío superior al determinado en [3.3.

También se puede comprobar que cuantos más perros hayan en la BBDD, mejores serán los resultados obtenidos, ya que hay más posibilidad que algún perro se asemeje más a las características del usuario.

4. Conclusiones

En este apartado se detallará todas las conclusiones finales del proyecto.

4.1 Conclusiones generales

Con este proyecto he podido desarrollar un sistema recomendador y una web de interface con el usuario, lo que me ha permitido tener una visión de implementar un proyecto en todas sus fases.

Los resultados han sido bastante satisfactorios ya que la suma de los valores de las características que superan el desvío máximo aceptado nunca supera el 21% de los valores recomendados.

4.2 Análisis del seguimiento de planificación y metodología

Ha sido posible seguir la metodología y la planificación en casi todas las fases del proyecto.

He tenido mayor dificultad en seguir la planificación en la fase de Desarrollo, donde claramente no había tenido en cuenta la investigación de los recomendadores existentes ni de los datos relevantes en la adopción/abandono de perros.

También inicialmente no he tenido en cuenta que tendría dificultades en solucionar los pequeños errores en la web e ir mejorando, a medida que fuera desarrollando. El tiempo invertido en desarrollar la web y aplicar el algoritmo recomendador ha sido mayor que el esperado.

Esto ha ocasionado que la fase de prueba se retrasará un poco, porque fue necesario tener las funcionalidades básicas implementadas y sin errores, para que los usuarios pudieran probar y por esto el tiempo inicialmente previsto no ha sido el real.

La redacción de la memoria ha sido otro factor en el cual se ha tenido que dedicar más tiempo que el previsto. Decidir como presentar la información ha sido una dificultad que no se había planeado.

He actualizado la planificación inicial del apartado 1.4 para las modificaciones que han sido realizadas durante el proyecto.

4.3 Análisis del cumplimento del objetivo

Analizando cada objetivo propuesto y el grado de cumplimento, llegamos a las conclusiones abajo:

- *“Investigar los sistemas recomendadores actuales y elegir el algoritmo/tecnología mas apropiada”*

Este objetivo fue cumplido, para elegir como desarrollar el proyecto, hemos investigado distintos algoritmos y sistemas recomendadores existentes.

- *"Adaptar el algoritmo para el objetivo principal (desarrollar un sistema recomendador para perros en adopción a partir de una base de datos) "*

Este objetivo fue cumplido, ya que he utilizado el Mahout para desarrollar e implementar un recomendador para el objetivo principal de este proyecto

- *"Desarrollar una aplicación web muy sencilla que sería la interfaz entre el usuario (posible adoptante) y nuestro recomendador."*

Este objetivo fue satisfactoriamente cumplido, ya que he implementado una web muy sencilla para que usuarios finales pudieran probar la aplicación .

- *"Crear un cuestionario donde sea posible determinar características fundamentales del usuario (posible adoptante) para posteriormente obtener su recomendación. No existirá un cuestionario directo donde el adoptante elija color/edad/sexo/raza y si elaboraremos preguntas donde intentaremos encontrar características claves de comportamiento del adoptante y sus gustos."*

También he podido cumplir este objetivo, elaborando un cuestionario que no mapeará directamente las características y si buscara encontrar características claves de los adoptantes.

- *"Crear una base de datos con datos para realizar los experimentos. La idea es que en el futuro se consiga la colaboración de protectoras en distinta partes de España."*

Se ha cumplido este objetivo creando una base de datos de perros ficticios in memoria para que fuera posible realizar las pruebas.

- *"Realizar pruebas con usuarios reales y comprobar la eficiencia del proyecto."*

Este objetivo se cumplió ya que fue posible que varias personas reales probaran la aplicación y analizar sus resultados.

- *"Realizar toda la documentación necesaria."*

Este objetivo también ha sido cumplido, y se ha generado toda la documentación necesaria para este TFM.

Con el análisis detallado del cumplimento de los objetivos, considero que se ha cumplido totalmente los objetivos propuestos en el inicio del proyecto, ya que todos los puntos han sido concluidos con éxito.

Ha sido necesario que la web implementada fuera lo mas simple posible y sin datos reales, ya que esto ocasionaría un coste que no se podría asumir en este proyecto.

4.4 Futuras mejoras

Existen varios puntos de mejora que han quedado fuera del alcance del proyecto, los de más alta prioridad considero que son los siguientes:

- Añadir los datos de los perros disponibles en una BBDD.
- Mejorar la parte grafica de la web para que el usuario esté más predispuesto a utilizarla.
- Modificar la pagina web para que al consultar los detalles del perro, muestre gráficos del porque este perro ha sido recomendado para el usuario, además de los datos de este perro.
- Tener una intranet donde las asociaciones y protectoras de animales puedan introducir los datos de sus perros en adopción.
- Traducir la web en varios idiomas.

4.5 Lecciones aprendidas

Con este proyecto he aprendido que es muy importante tener en cuenta el tiempo necesario para solucionar pequeñas incidencias.

También es muy importante que usuarios reales utilicen el producto el cuanto antes, aunque solamente realicen pruebas básicas, para que haya tiempo suficiente para solventar los pequeños errores antes de las pruebas con usuarios reales que se utilizaran para estadísticas.

He aprendido que es muy importante empezar a redactar toda la información encontrada en la memoria desde el inicio del proyecto, porque así es más fácil tomar las decisiones y analizar las investigaciones realizadas.

5. Glosario

Recomendadores, sistemas recomendadores o sistemas de recomendación: según Wikipedia.org forman parte de un sistema de filtrado de información, los cuales presentan distintos tipos de temas o ítems de información que son del interés de un usuario en particular. Generalmente, un sistema recomendador compara el perfil del usuario con algunas características de referencia de los temas, y busca predecir el "ranking" o ponderación que el usuario le daría a un ítem que aún el sistema no ha considerado. Estas características pueden basarse en la relación o acercamiento del usuario con el tema o en el ambiente social del mismo usuario.

6. Bibliografía

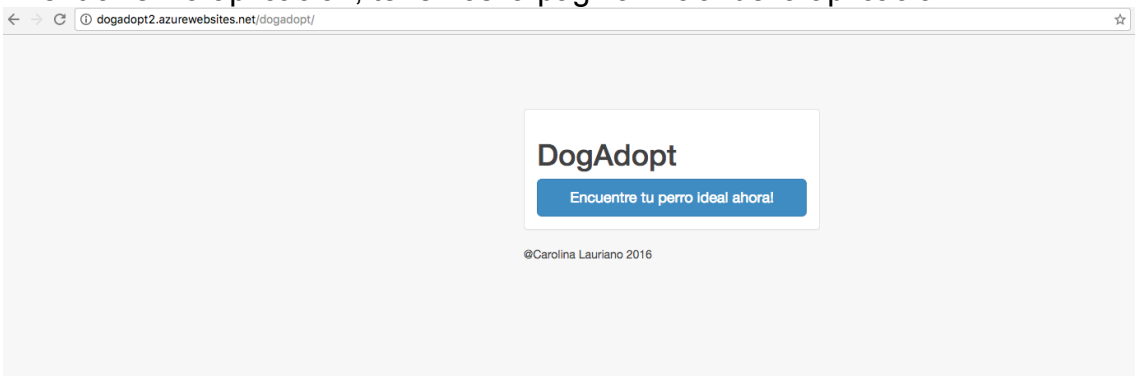
- [1]https://www.cnet.com/news/top-10-movie-recommendation-engines 30/09/2016
- [2]http://www.lavanguardia.com/vida/20140710/54410959519/espana-europa-abandona-mascotas.html 30/09/2016
- [3]http://www.pedigree.com/all-things-dog/select-a-dog/ 2/10/2016
- [4] http://matchcota.com/ 2/10/2016
- [5] http://www.thymeleaf.org/
- [6]https://startbootstrap.com/template-overviews/sb-admin-2/
- [7] https://mahout.apache.org/
- [8] http://www.librec.net/
- [9] http://lenskit.org/
- [10] http://recsyswiki.com/wiki/LensKit
- [11] http://easyrec.org/
- [12] http://www.duineframework.org/
- [13]OWEN Sean, ANIL Robin, DUNNING Ted, FRIEDMAN Ellen, “Mahout in Action”, Manning Publications Co, Shelter Island NY, 2012
- [14] http://dogtime.com/quiz/dog-breed-selector
- [15] http://prime.peta.org/2010/01/why-people-abandon-animals
- [16] https://www.cesarsway.com/get-involved/rescue/reasons-dogs-end-up-in-shelters-rescue-series-pt1
- [17] http://www.animal-rights-action.com/pet-abandonment.html
- [18] http://www.mundoanimalia.com/articulo/pasear_perros
- [19] http://www.dogeduca.es/index.php/extensions/129-la-importancia-del-paseo
- [20] https://www.tutorialspoint.com/mahout/mahout_recommendation.htm
- [21] https://es.wikipedia.org/wiki/Sistema_de_recomendaci%C3%B3n

7. Anexos

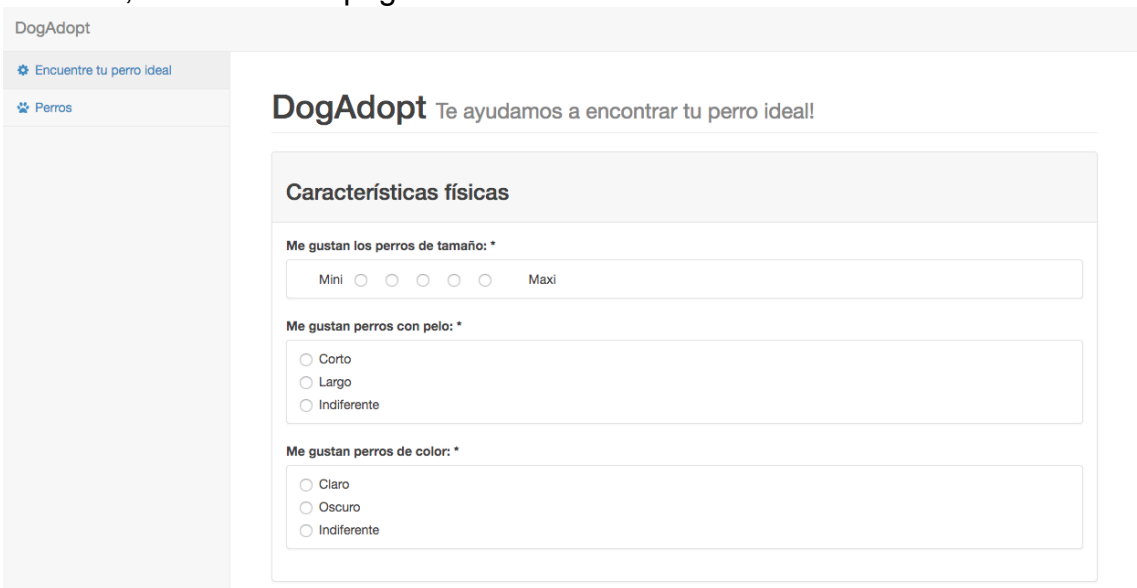
7.1 Manual de usuario

Para acceder la aplicación, tenemos que entrar en:
http://dogadopt.azurewebsites.net/dogadopt/ (servidor en USA)
http://dogadopt1.azurewebsites.net/dogadopt/ (servidor en Europa)
http://dogadopt2.azurewebsites.net/dogadopt/ (servidor en Europa)

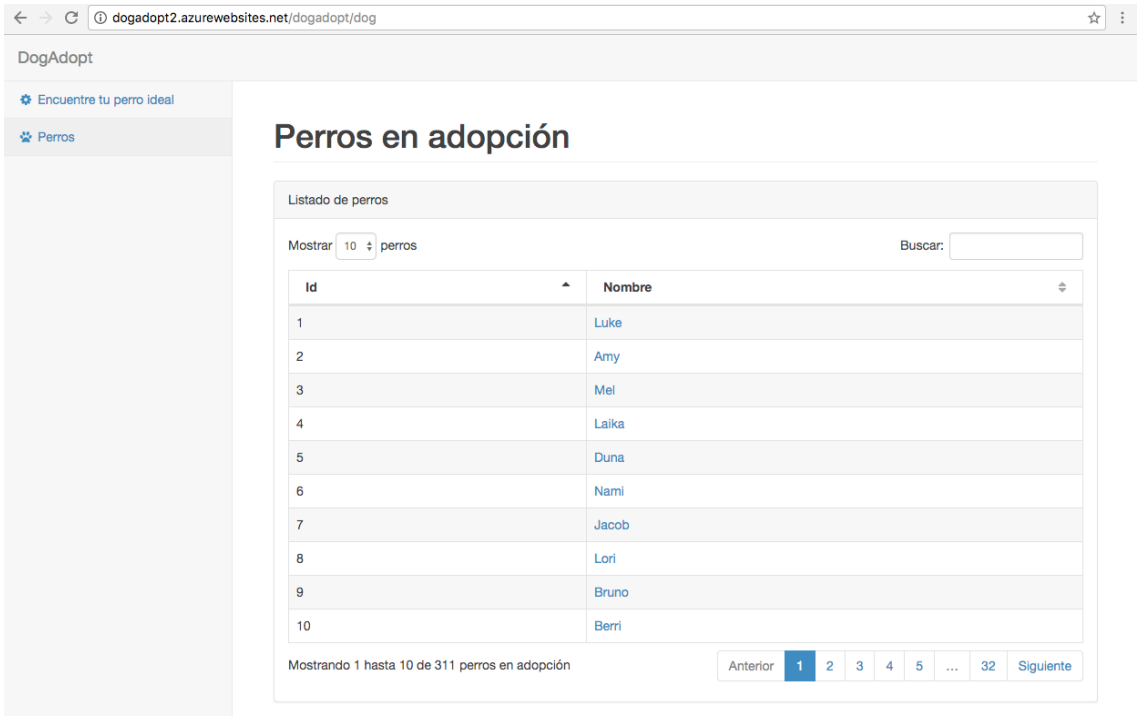
Al entrar en la aplicación, tenemos la pagina inicial de la aplicación:



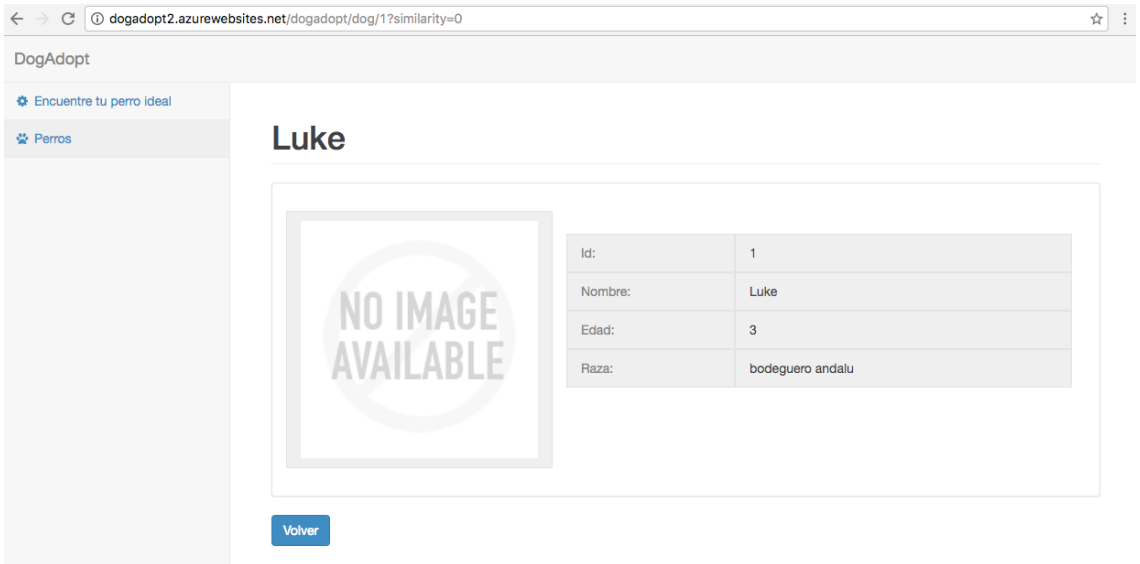
Al entrar, tenemos esta página:



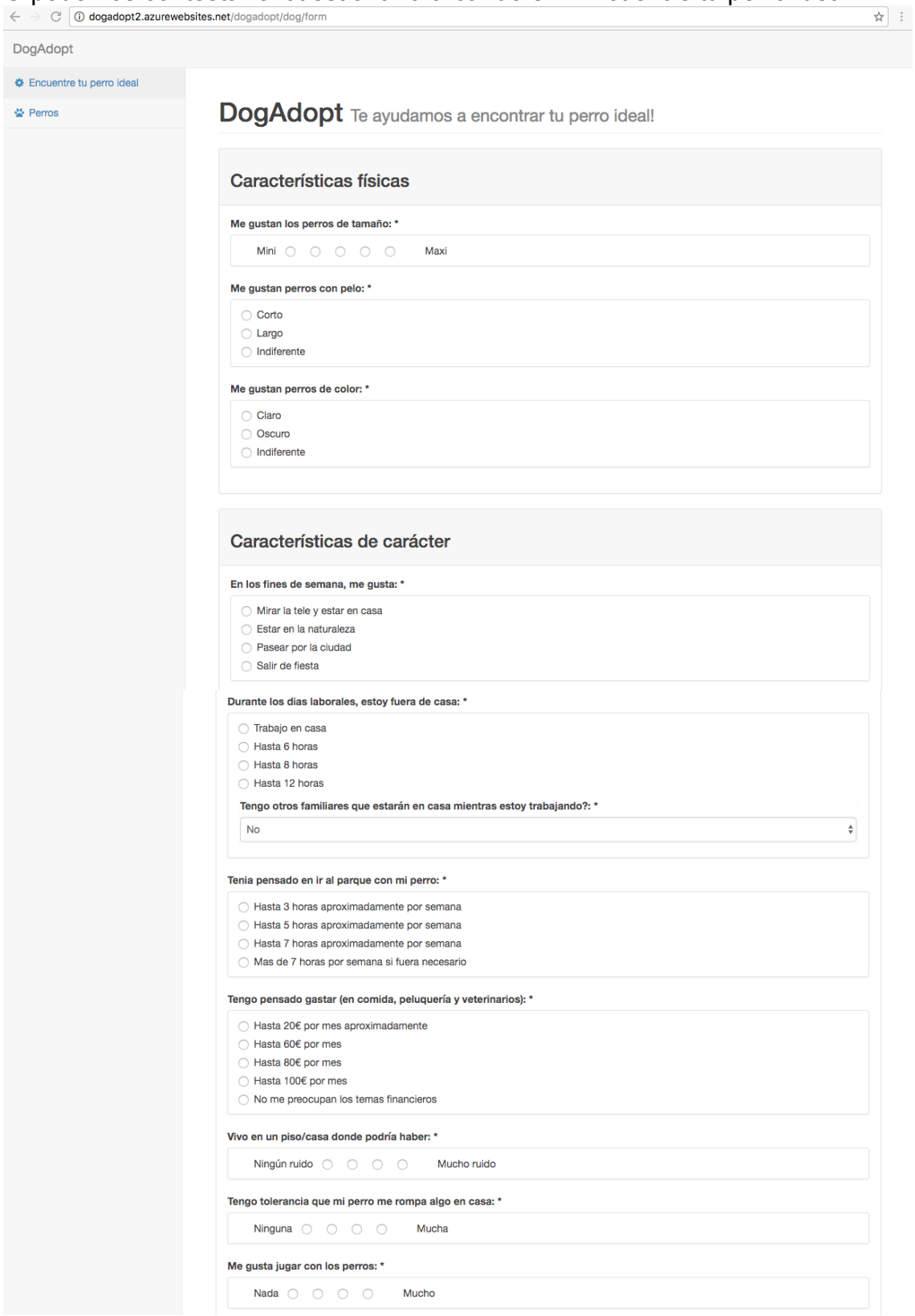
Donde podemos elegir ver todo el listado de los perros existentes clicando en “Perros”:

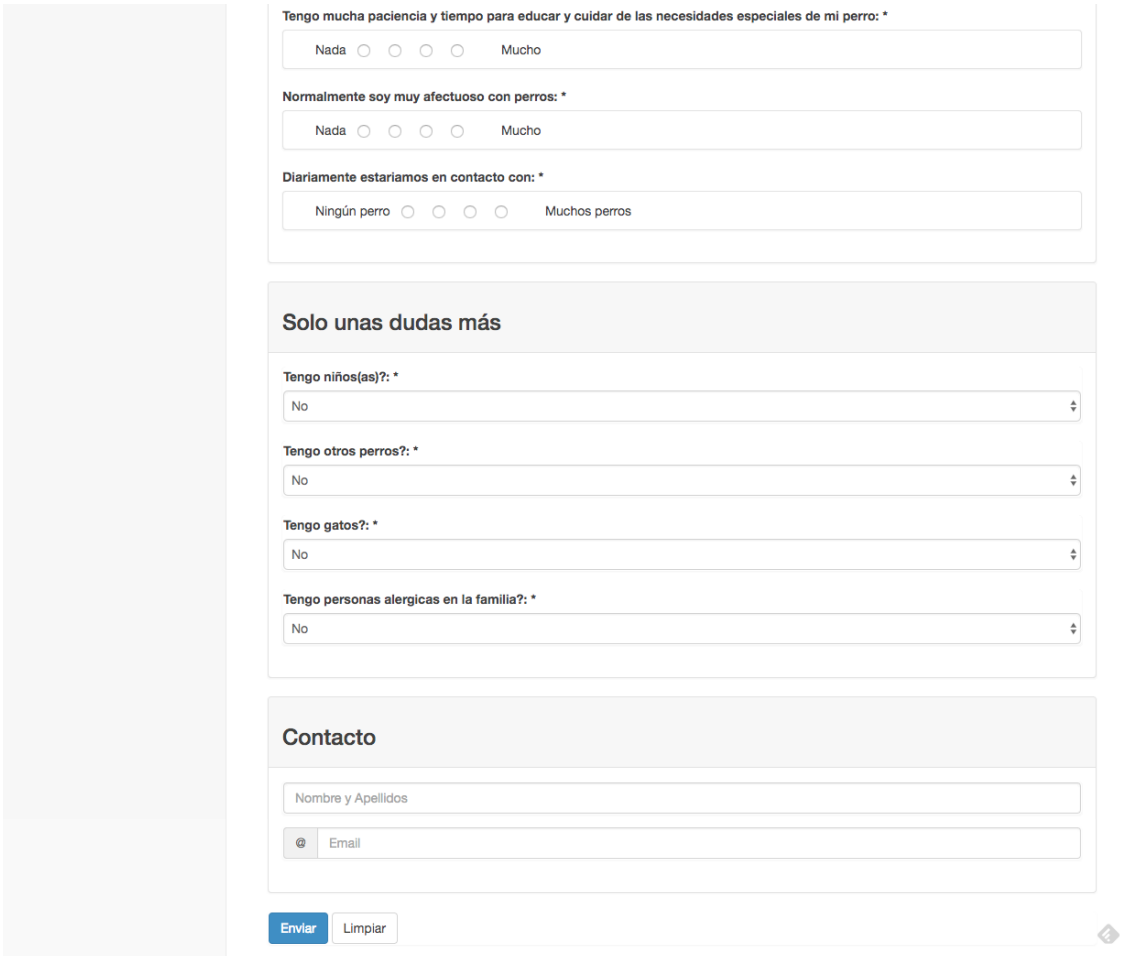


Y clicando en uno vemos sus detalles:

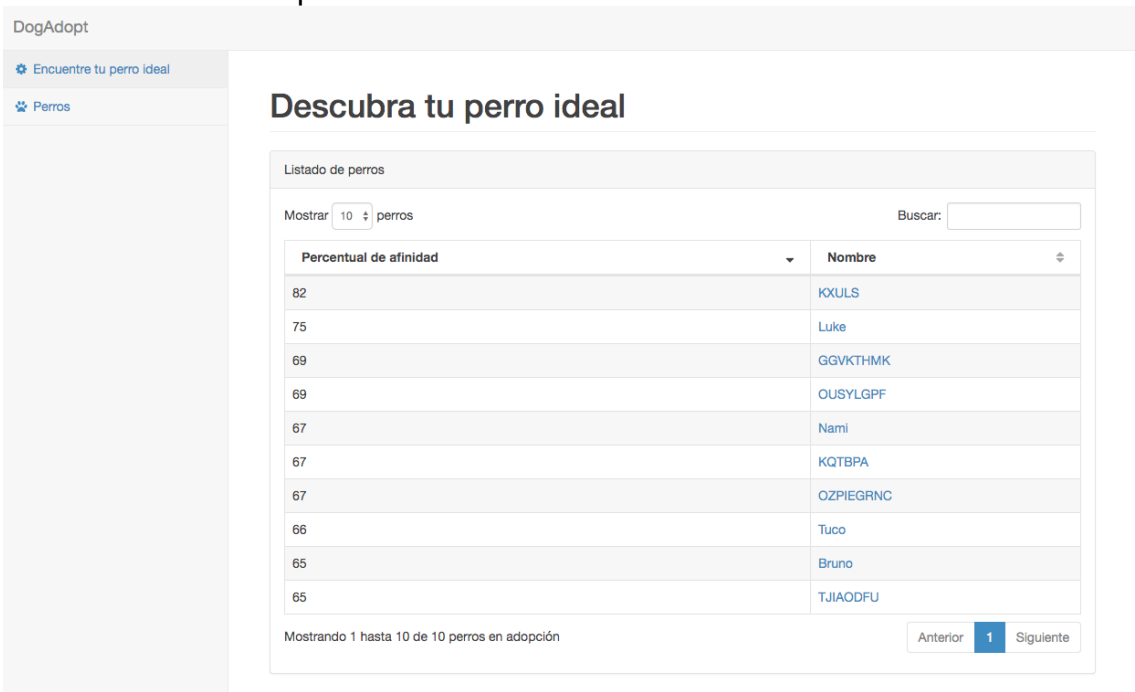


O podemos contestar el cuestionario clicando en “Encuentre tu perro ideal”:

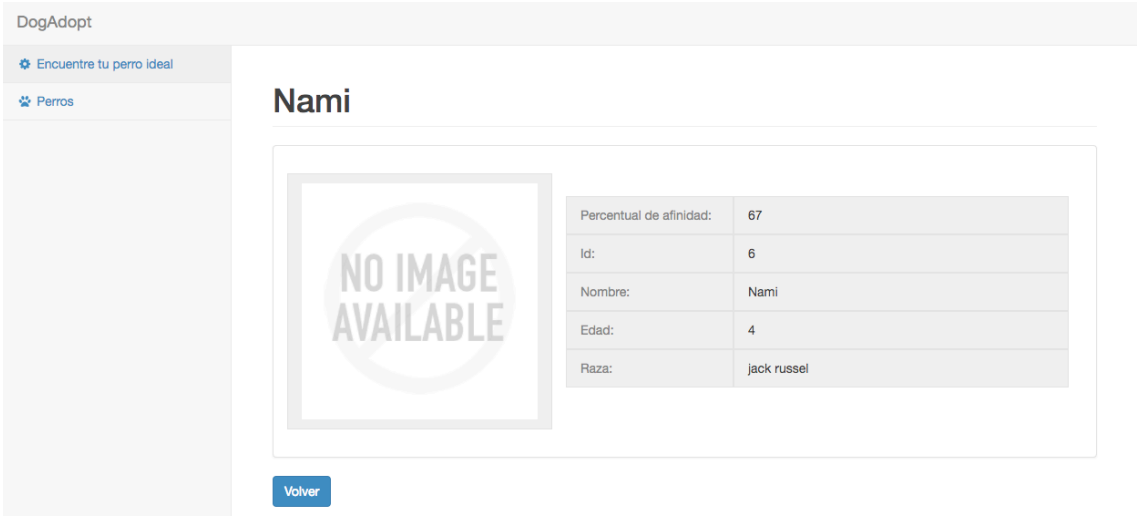




Al clicar en enviar, obtendremos una página con los 10 primeros perros mas similares al usuario que ha contestado el cuestionario:



Y clicando en cada uno vemos sus detalles:



7.2 Estructura de la entrega

En este proyecto final entregamos los siguientes ficheros:

- presentacionTFMCarolinaLauriano.mov: Presentación del proyecto con voz sobrepuesta.

- presentacionTFMCarolinaLauriano.pdf: Dispositivas con la presentación.

- src/dogadoptTFM-1.0.zip: Código fuente con la implementación del proyecto.
  Este código fuente puede ser ejecutado gracias a Spring-boot y Maven de la siguiente forma:
  Desde la carpeta raíz :
    - `mvn clean install –U`

  Desde la carpeta dogadopt-web-app-war:
    - `mvn spring-boot:run`

  Consultar aplicación desde http://localhost:8080/

- TFM_LaurianoDaSilvaCarolina.pdf : Memoria final.

- war/dogadopt.war: Ejecutable del software entregado (lo mismo que se encuentra en http://dogadopt1.azurewebsites.net/dogadopt/), que puede ser ejecutado en cualquier servidor de aplicaciones, por ejemplo, Tomcat.