Departamento de Informática
Universidad Técnica Federico Santa María

Entregable IV

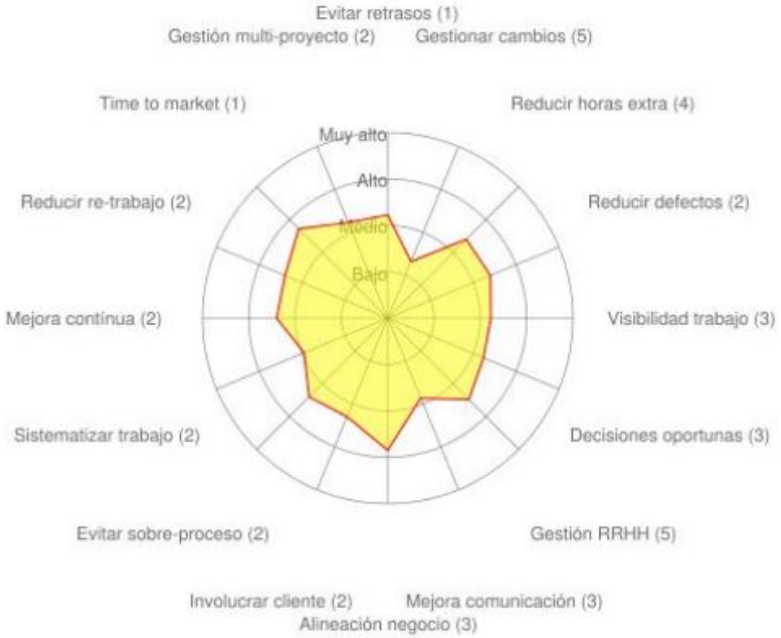
Proyecto: Classer

Integrantes:

Nombres y Apellidos	Email	ROL USM
Francisco García M	francisco.garciam.13@sansano.usm.cl	201373540-1
Rodrigo Valenzuela M	rodrigo.valenzuem.13@sansano.usm.cl	201373602-5
Simón Contreras M	simon.contreras.13@sansano.usm.cl	201323503-4

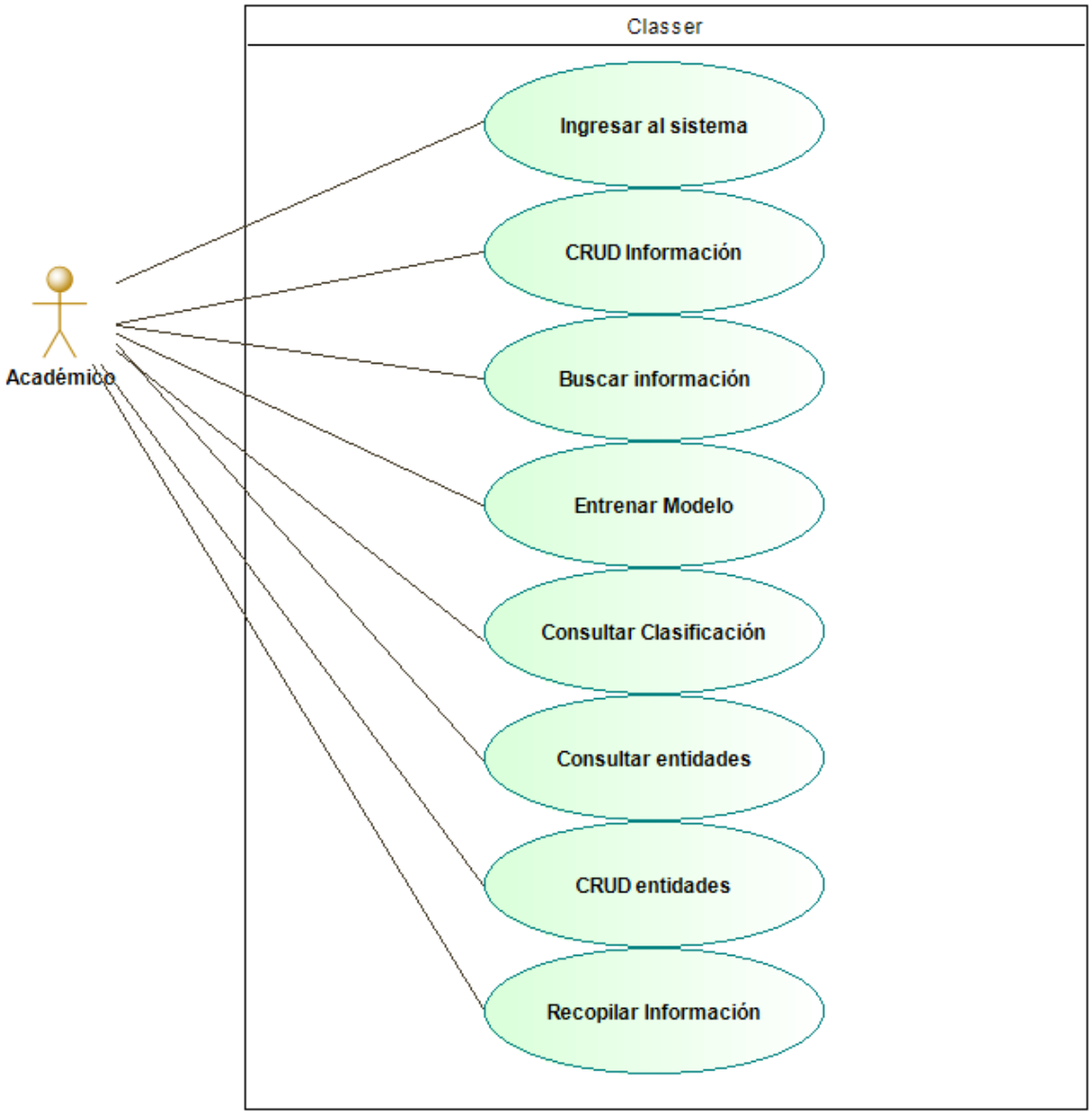
Post-Mortem Metodológico

De los resultados nos damos cuenta que en áreas como la relación con el cliente y liderazgo demostramos un mayor nivel de aplicación de las prácticas, por otro lado, en áreas como espacio trabajado, reuniones y proceso notamos un bajo nivel de aplicación de prácticas. Dado esto, en caso de seguir trabajando juntos, debiéramos darle más prioridad a la definición y mejora del proceso de desarrollo, así como también generar más instancias de reunión entre los miembros del equipo.



Por otro lado, el nivel de agilísimo en los objetivos es de nivel medio en general, con cierto grado de fortaleza en reducir: re-trabajo, evitar retrasos, reducir defectos, decisiones oportunas, alineación del negocio y evitar sobre proceso. Mientras que, en objetivos como mejora de comunicación, sistematizar trabajo y gestionar cambios se observa un cierto déficit en el nivel de agilísimo, que es necesario mejorar, por ejemplo, por medio de reuniones entre los miembros para mejorar en comunicación o como ya fue expuesto en el análisis anterior: un mejor enfoque en el proceso, así como en su sistematización.

Diagrama de Casos de Uso (final)

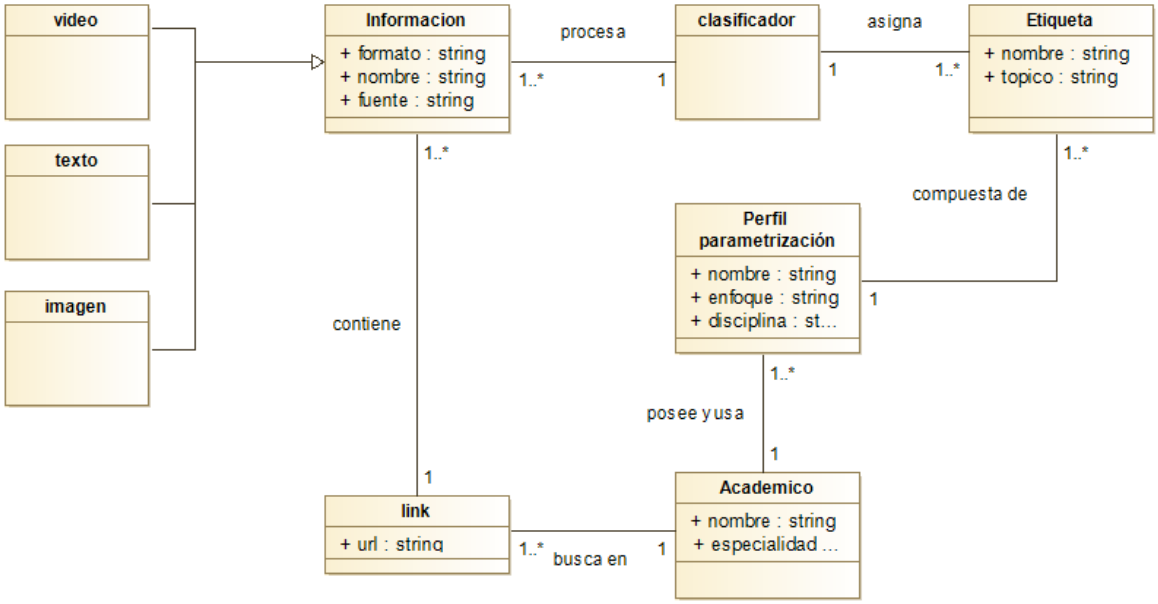


Los requerimientos principales del cliente, eran poder recopilar información de forma automática desde la Internet y que esta información fuera clasificada bajo distintos estándares, en este caso se separó la información en dos categorías principales: Definición y Ejercicio, además el texto y las imágenes recopiladas fueron analizadas para encontrar tópicos relevantes y poder asignar etiquetas a la información, con lo cual se pueden realizar búsquedas.

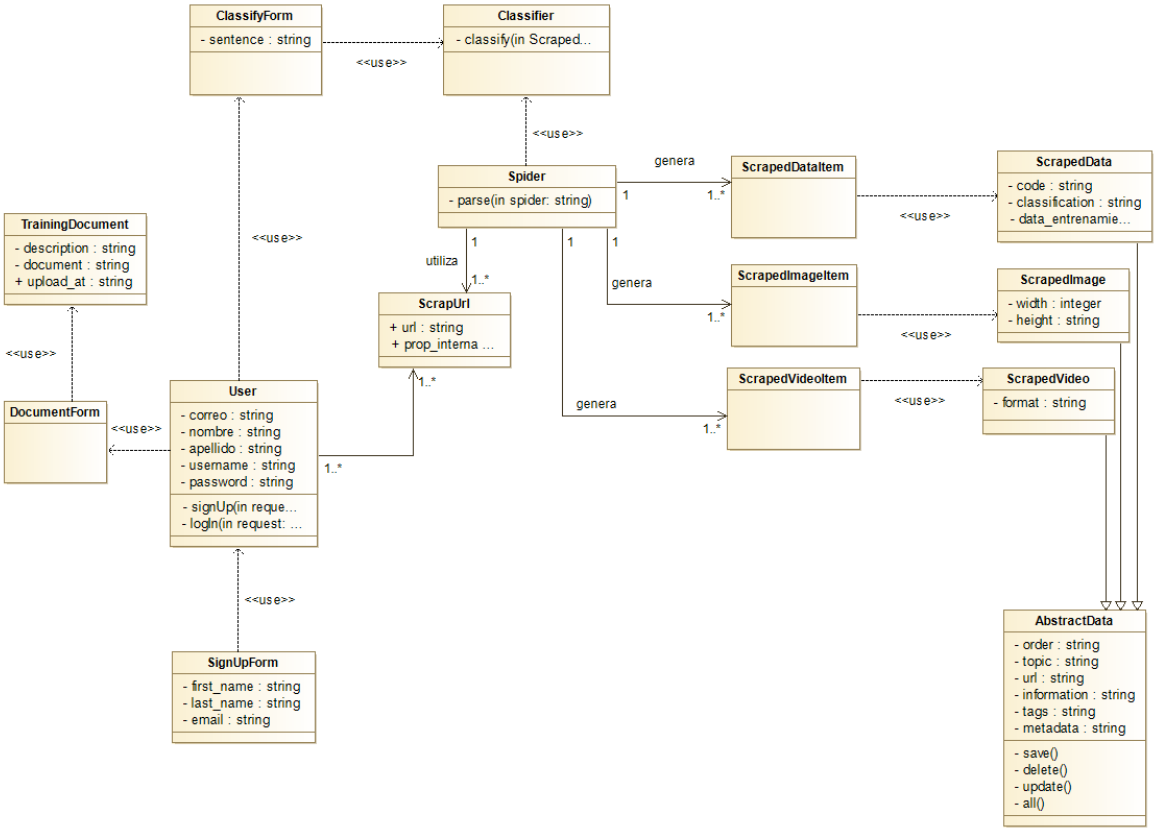
Patrones de diseño y Frameworks (final)

Intención	Patrón de Diseño o Framework	Razonamiento
Simplificar el desarrollo de software, utilizando una estructura definida, con módulos ya implementados.	Django	Se decidió utilizar un Framework como Django debido a que contiene módulos y patrones de diseños incorporados que permiten aumentar la rapidez en el desarrollo. Además dado que posee todas las herramientas que nos permitieron desarrollar las distintas funcionalidades solicitadas.
Separar los componentes principales, de manera de agilizar el proceso de desarrollo de nuestro sistema.	MVC	Patrón de diseño en el cual se basa Django, que nos permitió generar una división clara de componentes principales como son vistas, modelos y controlador.
Dar un enfoque simple y estructurado de acceso a la base de datos, que permita reducir la complejidad del código.	Active Record	Patrón de diseño que nos permite relacionar clases con tablas en la base de datos, de tal forma de no mezclar código con sentencias SQL, reduciendo la complejidad del código.
Se desea controlar de forma eficiente y precisa el flujo entre las diferentes secciones del sistema, para proveer una correcta experiencia de usuario.	Page Controller	Page Controller provee un sistema de asignación de rutas intuitivo y eficiente para las diferentes vistas de nuestro sistema.

Modelo de Dominio y Diagrama de Clases (final)



En el modelo de dominio final, se incluyen nuevos artefactos que contribuyen a una mejor comprensión del entorno que rodea al problema abordado, en donde la inclusión de los links, como del clasificador aportan a la completitud del dominio en donde se desenvuelven los requerimientos del cliente y de como nosotros planteamos una solución a estos.



El diagrama de clases final, varió en gran medida sobre el inicial debido a la restructuración de los componentes del proyecto y la forma de manipular la información entre los módulos internos y externos de Django. Los patrones de diseño mencionados forman parte de la estructura intrínseca del framework Django, ya que este está basado en una arquitectura MVC. Además, Django posee métodos y módulos propios que facilitan el desarrollo ya que implementan directamente patrones como active record, mediante el uso de una api del back-end la cual funciona como interfaz dependiendo del motor de base de datos a utilizar. En el caso del patrón page controller, este viene implementado mediante el uso de directivas y del método url(), el cual forma parte del módulo django.conf, el cual posee los métodos esenciales para poder desarrollar en base a capas (MVC).

Sobre el diagrama se destaca el uso de clases “intermediarias”, las cuales conectan los modulos de Scrapy (clases Spider, ScrapedDataItem, ScrapedImageItem, ScrapedVideoitem), directamente con el modelo de Django, lo cual permite una recopilación de información mucho más expedita y bajo el mismo estándar.

Por último, es destacable señalar que Django potencia el desarrollo mediante el uso de middlewares y de templates, los cuales aportan con esquemas a requerimientos comunes en el desarrollo de software.

Nota: para visualizar de mejor manera los modelos, referirse al proyecto de Modelio en GitHub.

Pruebas de Software (actualización)

No hubo tiempo para que los QA realizaran un nuevo testing sobre nuestro proyecto.