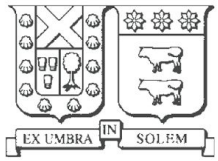
Departamento de Informática
Universidad Técnica Federico Santa María




# Requisitos de Software

# Proyecto: “Machine for learning”

Integrantes:

| Nombres y Apellidos | Email | ROL USM |
|---|---|---|
| Iván Caro León | ivan.caro.12@sansano.usm.cl | 201273545-9 |
| Felipe Flores Valdivia | felipe.floresv@alumnos.usm.cl | 201123518-5 |

## Desarrollo del Prototipo

Se desarrolló el primer prototipo del proyecto el cual se encuentra en el repositorio de github. El prototipo aún presenta fallas y se encuentra en fases de QA tempranas. El Backend se desarrolló utilizando TDD para la mayoría de los controladores y vistas para minimizar el impacto de los errores.

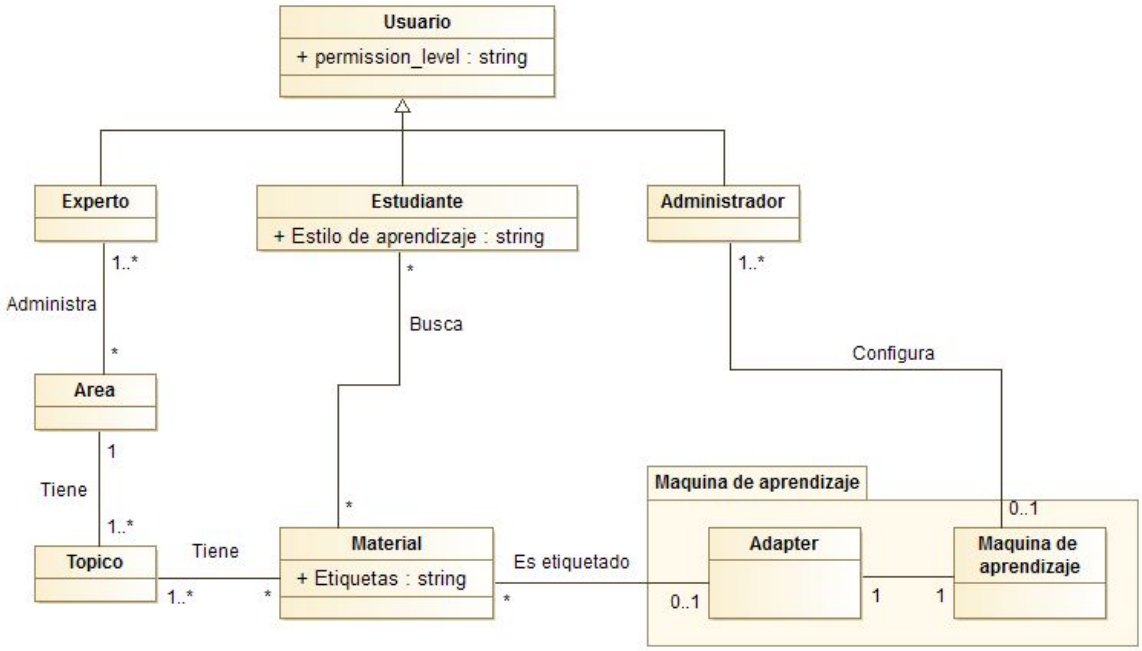
## Selección de Patrones de Diseño

| Intención | Patrón de Diseño | Razonamiento |
|---|---|---|
| Se desea separar los componentes: modelo, vista y controlador para el desarrollo de la aplicación. Para así tener un código más legible, poder reutilizar código | MVC | Este patrón se utiliza ya que debido al tamaño del sistema que se está desarrollando, este puede llegar a ser muy complejo, por lo que dividirlo en partes hace que el desarrollo sea más simple y rápido |
| Se desea tener solo una instancia de la clase máquina de aprendizaje | Singleton | Es necesario tener solo una instancia de la máquina de aprendizaje ya que esta “aprenderá” a medida que clasifica más el material de estudio y se quiere evitar posibles etiquetados distintos realizados por distintas máquinas de aprendizaje que tienen conocimiento diferente. Es para mantener la consistencia en el etiquetado de material. |
| Se desea que la máquina de aprendizaje no tenga problemas al recibir distintos formatos de material de estudio y pueda clasificarlo de manera correcta independientemente del | Adapter | Debido a los distintos formatos en los cuales se encuentra el material de estudio, la máquina de aprendizaje debe realizar un trabajo extra al clasificar todos los distintos tipos de material, por lo que al |

| | | |
|---|---|---|
| formato que este tenga | | agregar una clase que adapte este material a un formato más fácil de entender para la máquina de aprendizaje, esta tendrá menos trabajo al clasificar el material. |
| Se desea poder manejar de forma fácil y eficiente las instancias de cada clase, por lo que al utilizar este patrón facilita el manejo de los datos que usara el sistema | Active Record | Este patrón de diseño hace que el desarrollo sea mucho más simple en el sentido del manejo de los datos provenientes de la base de datos, ya que al ser tratados como objetos se hace mucho más fácil el trabajar con estos. |

## Diagrama de Clases

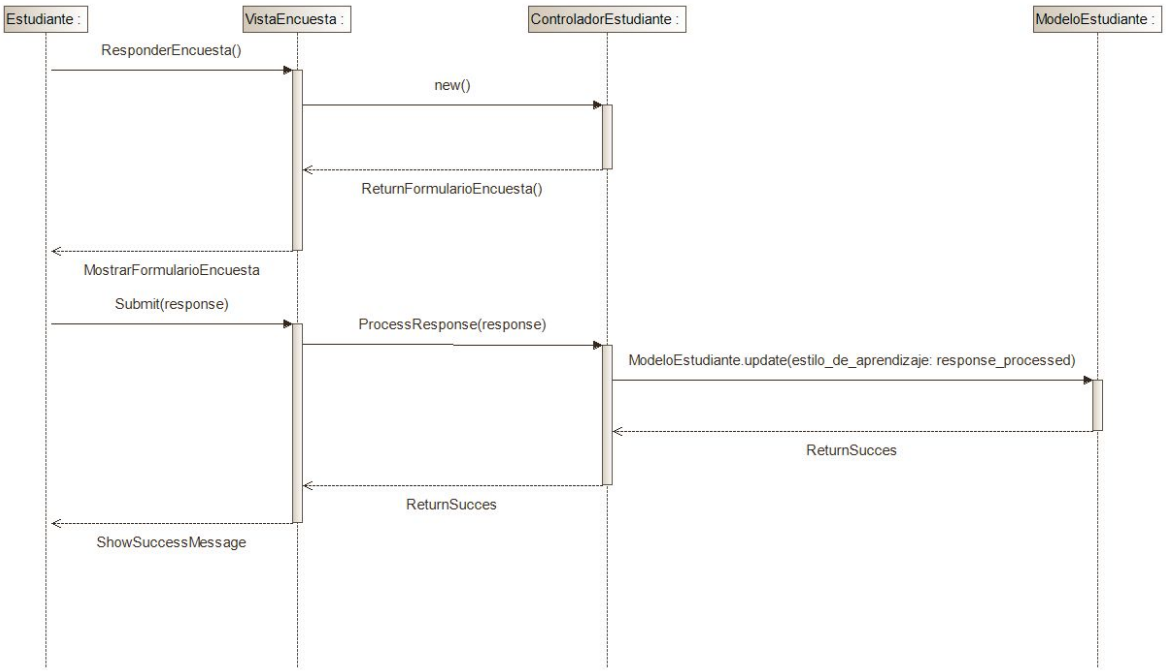
Se modeló un diagrama de clases el cual se muestra a continuación:



## Diagramas de Secuencia

Se generaron diagramas de secuencia para los siguientes casos de uso:

-Clasificar Estudiante:




-Mostrar Material:

## Análisis de Trade-off

Se desea tomar una decisión para agregar una nueva funcionalidad que no se consideró anteriormente, la cual es buscar contenido mediante el uso del etiquetado , para esto se realizó un análisis de trade-off basado en el enfoque QOCT el cual se detalla a continuación:

1) Question:
   - ¿Qué cambios son necesarios hacer para poder realizar búsqueda por etiquetas de contenido?

2) Options:
   - O1: Hacer que la máquina de aprendizaje guarde material de estudio buscado previamente y agregar un nuevo buscador que realice búsquedas usando el material ya procesado y etiquetado por la máquina de aprendizaje para mostrar resultados
   - O2: Agregar un nuevo buscador que realice una nueva búsqueda en la web y filtre los resultados por la etiqueta ingresada en el buscador

3) Criteria:
   - C1: Mantenibilidad
   - C2: Minimización de costos HH
   - C3: Escalabilidad
   - C4: Rendimiento

4) Análisis de Trade-Off:

| Criterio\Opciones | O1 | O2 |
|---|---|---|
| C1 | + | - |
| C2 | - | + |
| C3 | - | + |
| C4 | + | ++ |

A partir del análisis realizado anteriormente se pueden sacar las siguientes conclusiones:

- Opción 1: Si bien esta opción es mucho más fácil de mantener ya que al realizarse una búsqueda dentro del material ya guardado, todo estaría mucho más controlado, Es muy poco escalable ya que se necesita guardar todo el material que fue etiquetado anteriormente y a medida que crece la cantidad de usuarios se gastaría mucha memoria y recursos en almacenar este material.

- Opción 2: Esta opción tiene menos mantenibilidad ya que al realizar búsquedas en la web no se sabe los resultados que pueda entregar esta, lo que hace que sea más complejo manejar estos resultados. Pero a cambio de esto se tiene una mayor escalabilidad, menor costo de implementación y mejor rendimiento que en la opción anterior

En conclusión, la opción 2 es la que el equipo escogió para agregar esta nueva funcionalidad, Se sacrifica la mantenibilidad ya que las búsquedas en la web pueden arrojar a veces resultados que cambien en el tiempo o que sean inesperados, para los cuales habría que agregar alguna forma de manejarlos y así mostrarlos correctamente al usuario. Se gana mucho con respecto a la implementación y escalabilidad ya que no sería necesario agregar una funcionalidad que almacene material, todo este vendría de la web, por lo que no existe el problema de manejo de grandes cantidades de material. También ambas opciones tendrían un buen rendimiento pero es mejor el de la opción 2 ya que no se limita a una cantidad específica de material guardado previamente, al realizar búsqueda en la web se tiene acceso a todo el material disponible en esta incluyendo material nuevo o material que fue modificado recientemente, que en la opción 1 no se tendría o no se tendría actualizado.