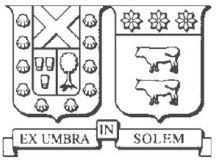
Departamento de Informática
Universidad Técnica Federico Santa María



Entregable III
Análisis y Diseño de Software / Fundamentos de Ingeniería de Software

Integrantes:

Nombre	Email	Teléfono
Andrés Cifuentes	andres.cifuentesv@alumnos.usm.cl	+56 9 9846 2881
Felipe Avaria	felipe.avaria@alumnos.usm.cl	+56 9 4545 8611
Jose Caimapo	jose.caimapo.12@sansano.usm.cl	+56 9 3002 8409

1- Listado de requerimientos

Requerimientos funcionales

ID requerimiento	Requerimiento	Obligatoriedad
FR1	El sistema debe proveer al usuario estudiante la funcionalidad de personalizar(mover contenido a su gusto en su lienzo)	Deseable
FR2	El sistema debe proveer una interfaz diferenciada entre los distintos tipos de perfiles de estudiantes obtenidos a través de la encuesta	Obligatoria
FR3	El sistema debe proveer a los usuarios correspondientes la funcionalidades de crear/modificar/leer/eliminar/calificar los contenidos	Obligatoria
FR4	El sistema debe validar/registrar usuarios.	Obligatoria
FR5	El sistema debe proveer la facilidad de dividir contenidos de un ramo en módulos	Obligatoria
FR6	Los módulos de un ramo se desbloquean por fecha	Obligatoria
FR7	El sistema debe proveer la capacidad de crear diferentes ramos	Deseable
FR8	El sistema facilita al usuario estudiante a mutar de perfil (cambiar de perfil) con ciertas restricciones	Deseable
FR9	El sistema debe poder guardar la última acción del usuario	Deseable
FR10	Solamente tienen perfil diferenciados estudiantes que han respondido la encuesta satisfactoriamente	Obligatoria

Requerimientos no funcionales

ID requerimiento	Requerimiento no funcional	FR asociado
NFR1	El sistema debe estar restringido, rechazar accesos y modificaciones no autorizadas	FR4
NFR2	Todos los campos de la encuesta deben ser llenados	FR10

NFR3	Cada ramo debe tener al menos un profesor como encargado de administrar	FR6
NFR4	La interfaz del sistema debe ser amigable con el usuario, es decir intuitivo y simple	FR1 , FR3
NFR4	El sistema debe ejecutarse sobre node.js en general sobre el stack tecnológico M*EAN	

2- Casos de uso

a.- Sistema:

Nombre	Arándano
Descripción	Plataforma que provee de las funcionalidades de administración y gestión de los contenidos de forma diferenciada además de credenciales.

b.- Actores:

Nombre	Estudiante
Descripción	Usuario que usa la plataforma en modalidad lectura con posibilidades de personalización

Nombre	Profesor
Descripción	Usuario que usa la plataforma como apoyo a su trabajo, tiene poderes de CRUD dentro de un curso

Nombre	Administrador
Descripción	Usuario que tiene control total dentro de la plataforma, gestiona permisos dentro de ella

c.- Casos de usos formato breve:

Ingresar al sistema, el caso de uso empieza el actor se identifica con sus credenciales personales y el sistema pasa a validar estos.

Calificar contenido., el caso de uso empieza con el estudiante pueda dar una calificación al material que ya haya usado (esto cuenta como visto, descargado , etc).

Visualizar contenidos por tipo, el caso de uso empieza con el actor puede ver los contenidos de su lienzo según el tipo de aprendizaje que le corresponde.

Personalizar lienzo, el caso de uso comienza con el actor moviendo los contenidos dentro de su lienzo, para su acomodarlo a su gusto.

Descargar contenido, el caso de uso comienza con el actor seleccionando el contenido que desea descargar, mientras el sistema valida si es posible descargar el contenido y termina con el resultado de la validación.

Responder encuesta, el caso de uso comienza con el registro del estudiante donde responde la encuesta para así asignarle un estilo de aprendizaje.

CRUD contenidos diferentes funciones que tiene el profesor sobre el ramo Create/Read/Update/Delete.
Ver Estadísticas de Ramo, el caso de uso empieza el actor puede ver la estadísticas generales del ramo pudiendo así tener un visión más amplia sobre qué sucede con cada perfil en cuanto a progresos.

Validar Usuarios, El actor controla a los usuarios dentro de la plataforma.
Gestionar ramos, El actor controla a los ramos dentro de la plataforma.

d.- Casos de uso detallados

1)

Nombre:	**Calificar Material**
Autor:	**Jose Caimapo**
Fecha:	**10.9.2016**
Descripción: El estudiante puede dar una calificación al material que ya haya usado (esto cuenta como visto, descargado , etc).	
Actores: Alumno	
Precondiciones: Estar identificado, haber visto el archivo	
Flujo normal: 1) El alumno califica el material dando un valor. 2) El sistema guarda la valorización del alumno. 3) El alumno finaliza (mediante algún botón).	
Flujo alternativo: 1) El alumno califica el material dando un valor. 2) El sistema no logra guardar el valor, levantando una excepción indicando el error. 3) El alumno es redirigido.	
Post-condiciones: volver a la pantalla de inicio correspondiente.	

2)

Nombre:	**Ingresar al sistema**
Autor:	**José Caimapo**
Fecha:	**23/08/2016**
Descripción: El usuario se autentifica para usar el sistema por medio de credenciales privadas.	
Actores:Estudiantes, Profesor, Experto, Administrador.	
Precondiciones: Estar registrado.	
Flujo normal: 1) El usuario rellena el formulario de ingreso con su identificador y contraseña. 2) El sistema comprueba credenciales (mediante api de otro sistema, maybe).	

3) El sistema aprueba credenciales. 4) El usuario está identificado.
Flujo alternativo: 1) El usuario rellena el formulario de ingreso con su identificador y contraseña. 2) El sistema comprueba credenciales (mediante api de otro sistema, maybe). 3) El sistema aprueba credenciales. 4) El usuario está no queda identificado.
Post-condiciones: Ingresar a la pantalla de inicio que le corresponde.

3)

Nombre:	**Responder encuesta**
Autor:	**Felipe Avaria**
Fecha:	**23/08/2016**
Descripción: el estudiante debe responder encuesta, para poder almacenar su tipo de aprendizaje.	
Actores: estudiante	
Precondiciones: estudiante registrado y identificado.	
Flujo normal: 1) El estudiante responde cuestionario 2) El sistema guarda los registros satisfactoriamente 3) El sistema asigna un grupo de aprendizaje y muestra por pantalla el perfil	
Flujo alternativo: no tiene	
1) El estudiante responde cuestionario 2) El sistema no logra guardar los registros, levanta excepción 3) El sistema asigna un grupo por defecto.	

4)

Nombre:	**CRUD Contenidos**
Autor:	**José Caimapo**
Fecha:	**10.9.2016**
Descripción: Diferentes funciones que tiene el profesor sobre el ramo Create/Read/Update/Delete.	
Actores: Profesor, Administrador	

Precondiciones: Tiene que existir ramo,El actor debe estar en el panel correspondiente, Y existir contenido en el caso de Delete,Update,Read
Flujo normal: 1) El actor selecciona el archivo. 2) El actor ejecutar la acción correspondiente sea Create, Read, Update ó Delete. 3) El sistema almacena el cambio. 4) El sistema informa que la acción es satisfactoria.
Flujo alternativo: 1) El actor selecciona el archivo. 2) El actor ejecuta la acción correspondiente sea Create, Read, Update ó Delete. 3) El sistema no almacena el cambio, levanta excepción. 4) EL sistema informa que la acción no fue realizada.
Post-condiciones: El sistema muestra el panel correspondiente

5)

Nombre:	**Ver estadísticas de ramo**
Autor:	**José Caimapo**
Fecha:	**11.9.2016**
Descripción: El actor puede ver la estadísticas generales del ramo pudiendo así tener un visión más amplia sobre qué sucede con cada perfil en cuanto a progresos.	
Actores: Profesor.	
Precondiciones: el usuario debe estar identificado como profesor y estar asociado como encargado de un curso.	
Flujo normal: 1. El actor selecciona la opción de ver estadísticas. 2. El sistema despliega las estadísticas. 3. El actor puede centrarse en diferentes tipos de estadísticas. 4. El actor cierra el despliegue de estadísticas.	
Flujo alternativo: No tiene.	
Post-condiciones: el actor vuelve a la vista inicial.	

6)

Nombre:	**Visualizar contenidos por tipo**
Autor:	**José Caimapo**
Fecha:	**5.11.2016**

Descripción: El actor puede ver los contenidos de su lienzo según el tipo que le corresponde
Actores: Estudiante
Precondiciones: El actor debe estar identificado en la plataforma.
Flujo normal: 1. El actor selecciona el contenido. 2. El sistema despliega despliega el contenido. 3. El actor lee el contenido. 4. El actor cierra el despliegue del contenido.
Flujo alternativo: 1. El actor selecciona el contenido. 2. El sistema no puede mostrar contenido. 3. El sistema levanta una alerta al usuario 4. El actor cierra el contenido.
Post-condiciones: el actor vuelve a la vista inicial.

7)

Nombre:	**Personalizar Lienzo**
Autor:	**José Caimapo**
Fecha:	**11.9.2016**
Descripción: El actor puede mover los contenidos de su lienzo.	
Actores: Estudiante	
Precondiciones: El actor debe estar identificado en la plataforma.	
Flujo normal: 1. El actor selecciona el contenido. 2. El sistema opción de desplegar el contenido. 3. El actor mueve el contenido. 4. El actor cierra el despliegue del contenido.	
Flujo alternativo: 1. El actor selecciona el contenido. 2. El sistema no puede mostrar contenido. 3. El sistema levanta una alerta al usuario 4. El actor cierra el contenido.	
Post-condiciones: el actor vuelve a la vista inicial.	

8)

Nombre:	**Validar usuarios**
Autor:	**José Caimapo**
Fecha:	**11.9.2016**
Descripción: El actor controla a los usuarios dentro de la plataforma.	
Actores: Administrador	
Precondiciones: El actor debe estar identificado en la plataforma como administrador.	
Flujo normal: 1. El actor selecciona al usuario. 2. El sistema muestra al actor los datos del usuario. 3. El actor realiza una acción sobre el usuario validandolo. 4. El sistema guarda los cambios.	
Flujo alternativo: 1. El actor selecciona el contenido. 2. El sistema no puede mostrar contenido. 3. El sistema levanta una alerta al usuario 4. El actor cierra el contenido.	
Post-condiciones: el actor vuelve a la vista inicial.	

9)

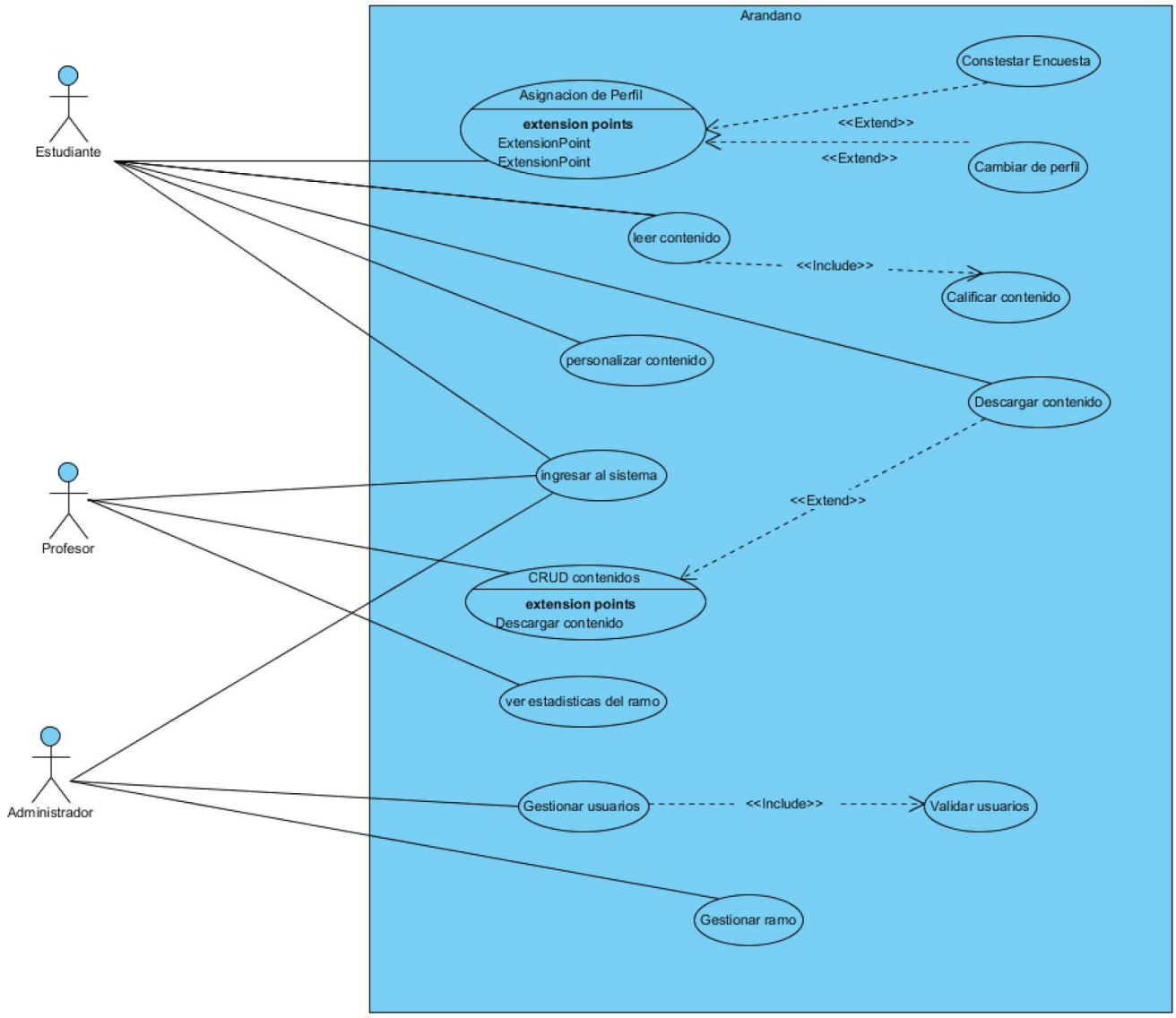
Nombre:	**Gestionar ramos**
Autor:	**José Caimapo**
Fecha:	**11.9.2016**
Descripción: El actor puede crear,modificar,eliminar ramos dentro de la plataforma..	
Actores: Administrador.	
Precondiciones: El actor debe estar identificado en la plataforma con administrador.	
Flujo normal: 1. El actor selecciona la acción de crear ramo. 2. El sistema despliega formulario de creación de ramo. 3. El actor llena el formulario y lo envía. 4. El sistema crea el ramo con los datos proporcionados.	
Flujo alternativo: 1. El actor selecciona la acción de modificar/eliminar el ramo. 2. El sistema despliega confirmación y pide insertar datos del actor .	

3. El actor confirmó acción positiva o negativa. 4. El sistema ejecuta la acción..
Post-condiciones: el actor vuelve a la vista inicial.

10)

Nombre:	**Descargar contenido**
Autor:	**José Caimapo**
Fecha:	**11.9.2016**
Descripción: El actor puede crear,modificar,eliminar ramos dentro de la plataforma..	
Actores: Profesor, Estudiante.	
Precondiciones: El actor debe estar identificado en la plataforma con administrador.	
Flujo normal: 1. El actor selecciona el archivo deseado y ejecuta la acción descargar. 2. El sistema recupera la información e inicia la descarga. 3. El sistema informa la descarga exitosa	
Flujo alternativo: 1. El actor selecciona el archivo deseado y ejecuta la acción descargar. 2. El sistema no encuentra el archivo, levanta excepción. 3. El sistema informa el error de la descarga.	
Post-condiciones: el actor vuelve a la vista inicial.	

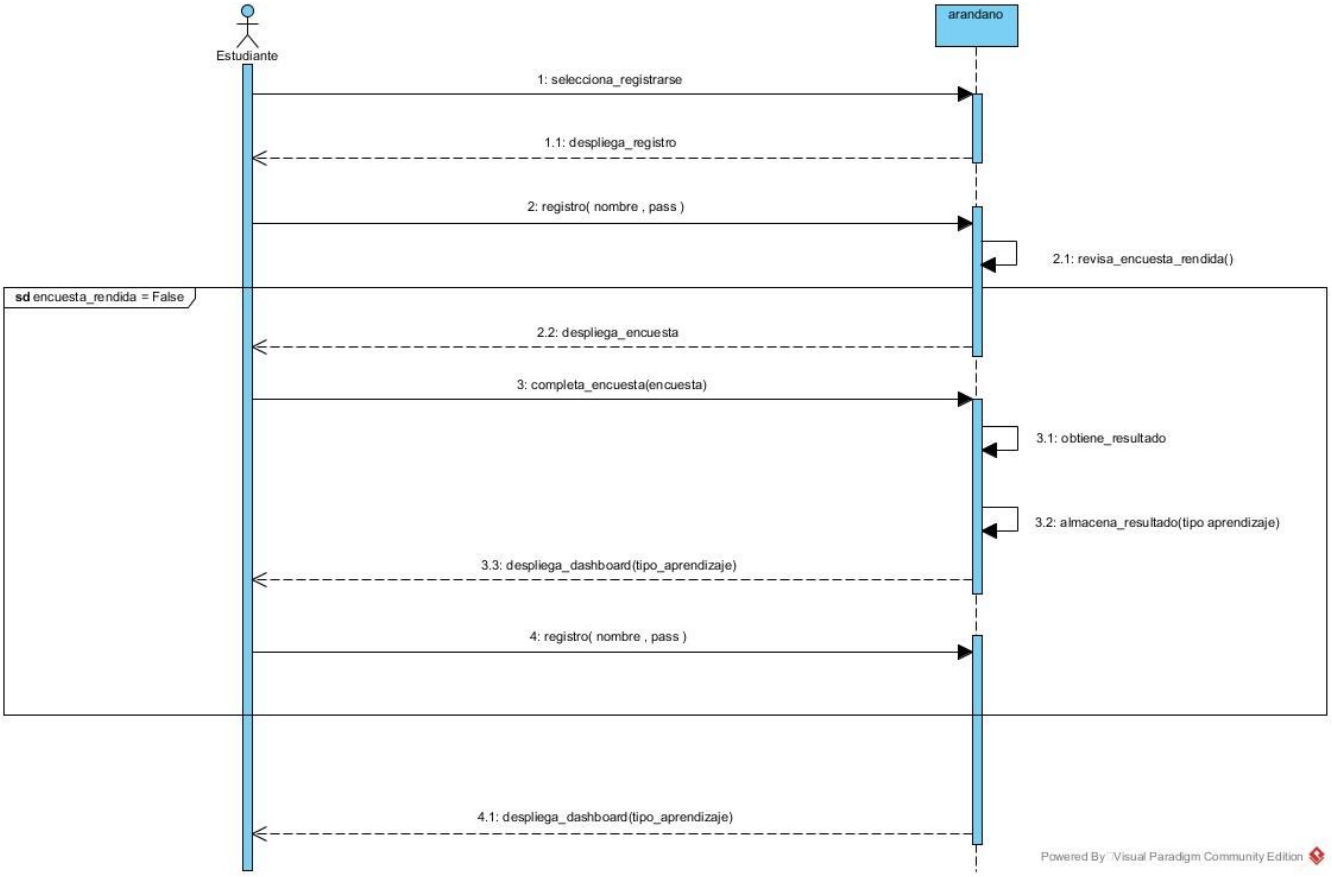
e.- Diagramas de caso de uso general



(ver diagrama completo en ProyectoArandano.vpp)

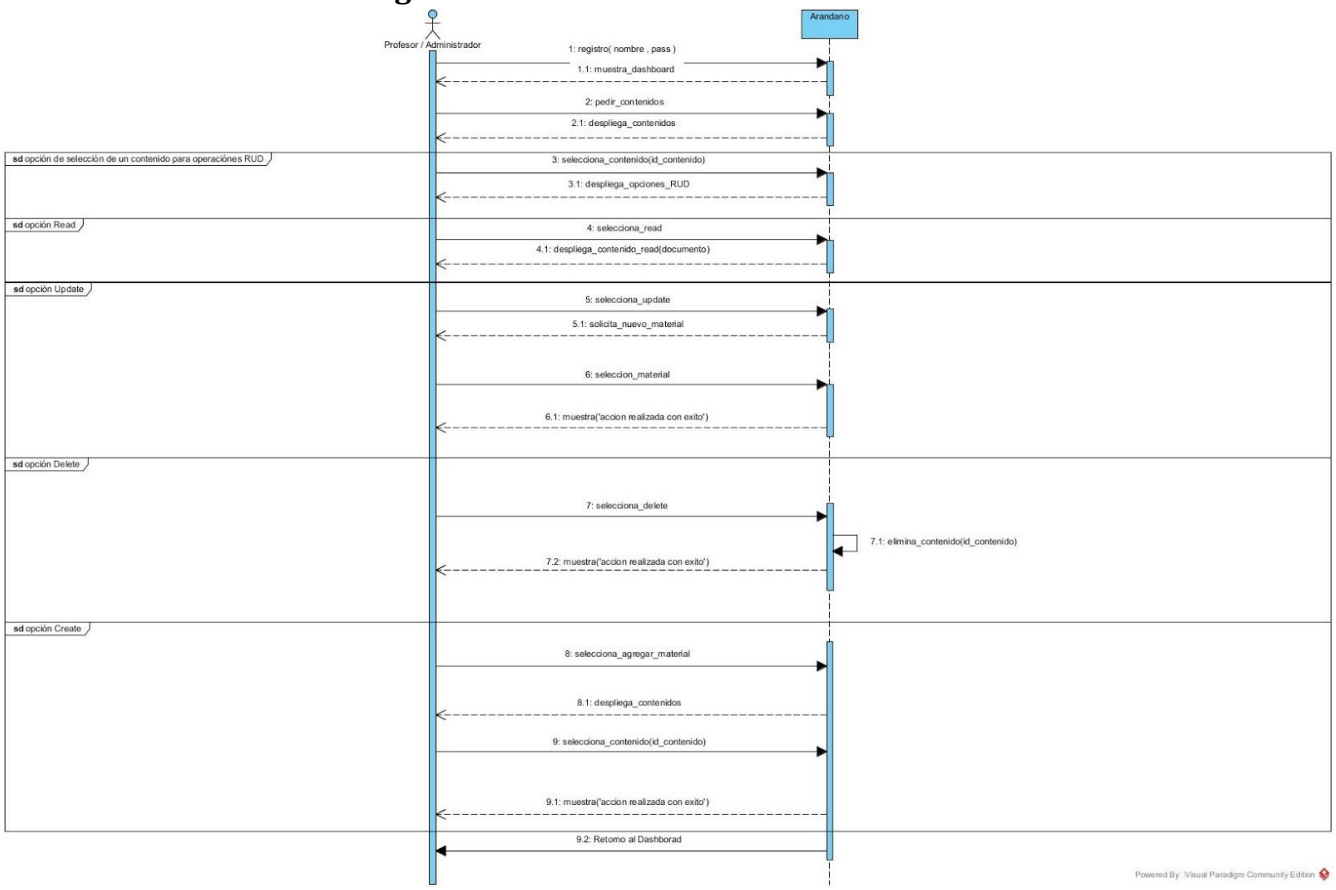
3- Diagramas de secuencia del sistema

Diagrama de secuencia 1 : Responder encuesta



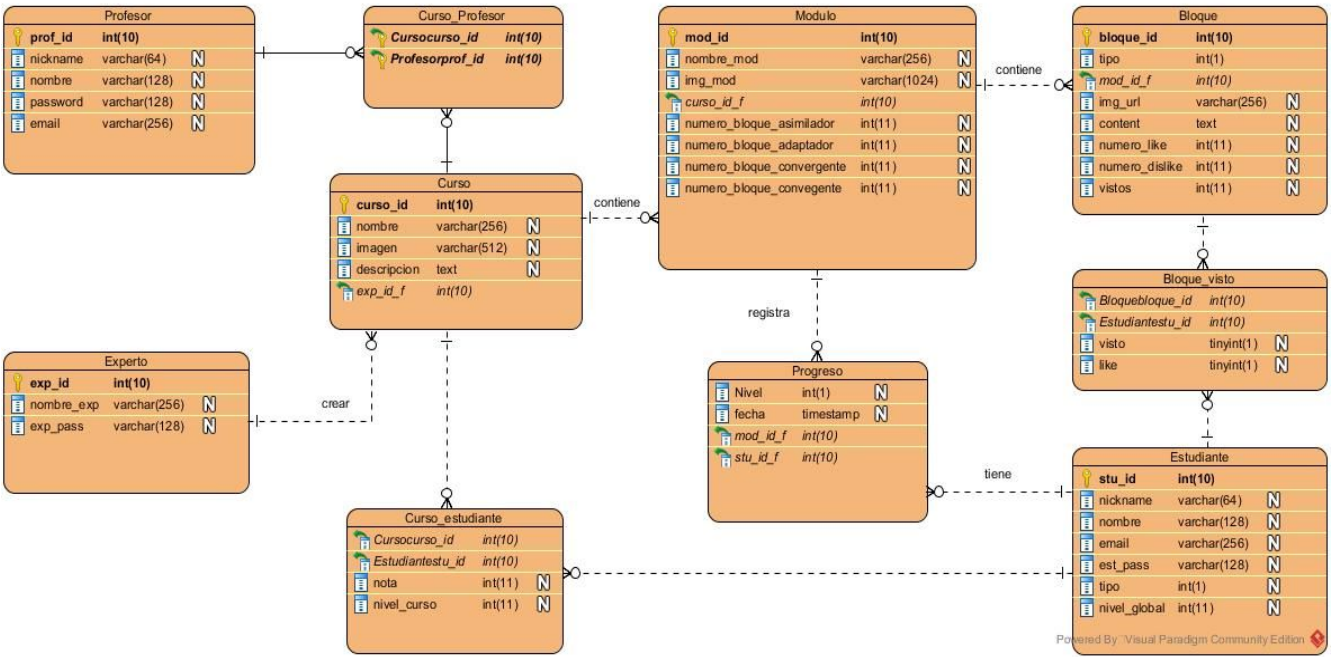
(ver diagrama completo en ProyectoArandano.vpp)

Diagrama de secuencia 2 : CRUD Contenidos



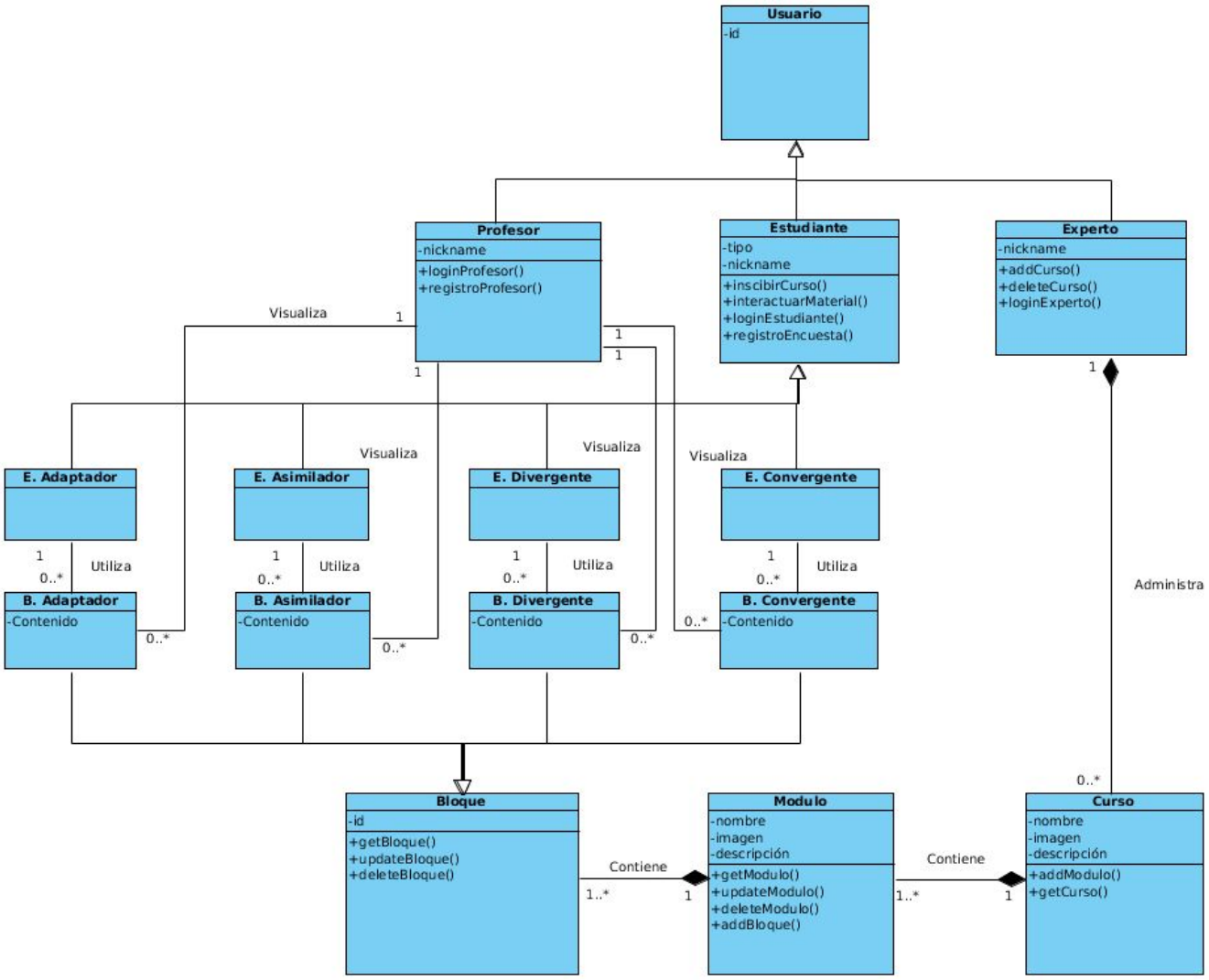
(ver diagrama completo en ProyectoArandano.vpp)

4- Modelo relacional de la base de datos



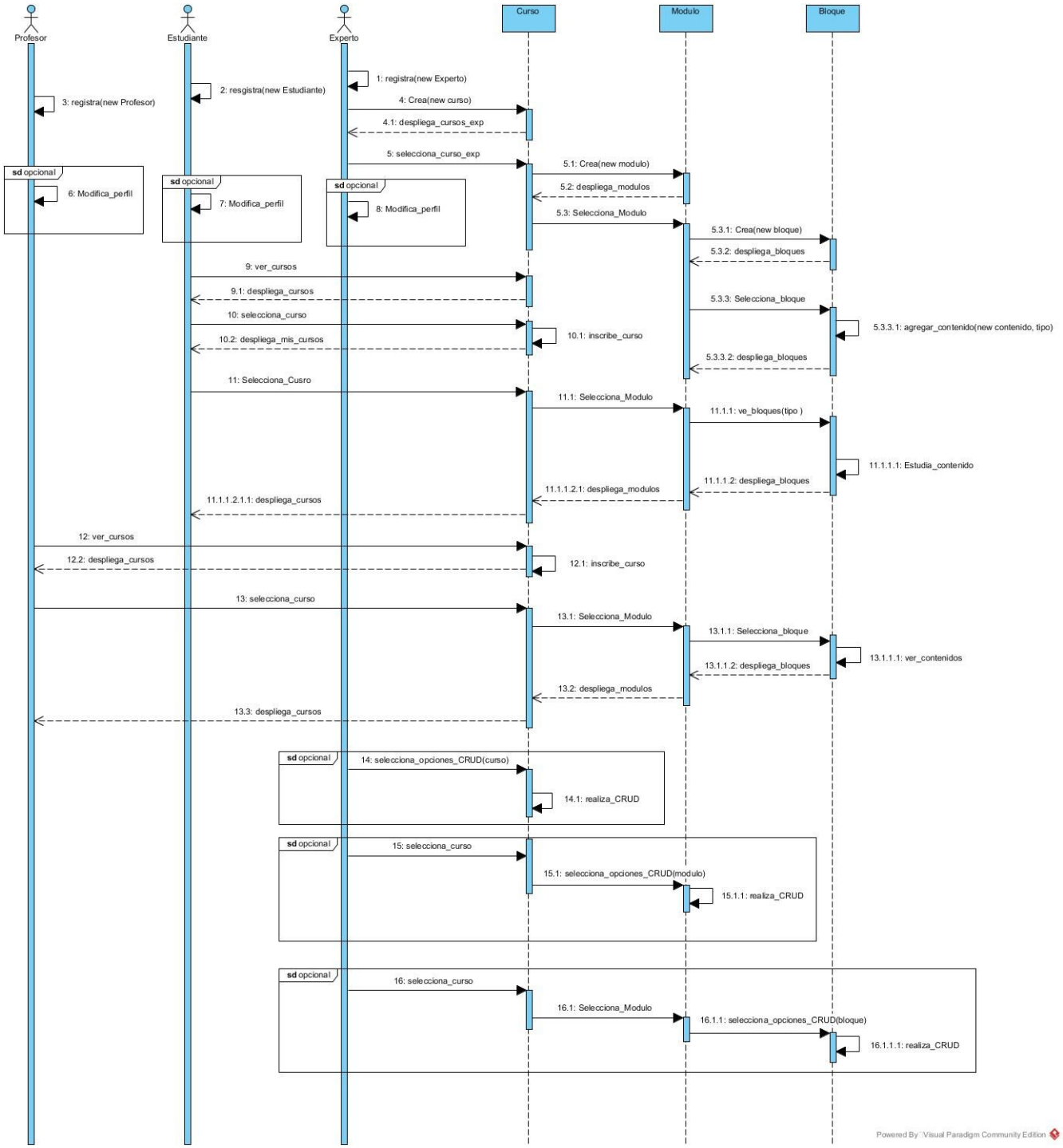
(ver diagrama completo en ProyectoArandano.vpp)

5- Modelo de Clases



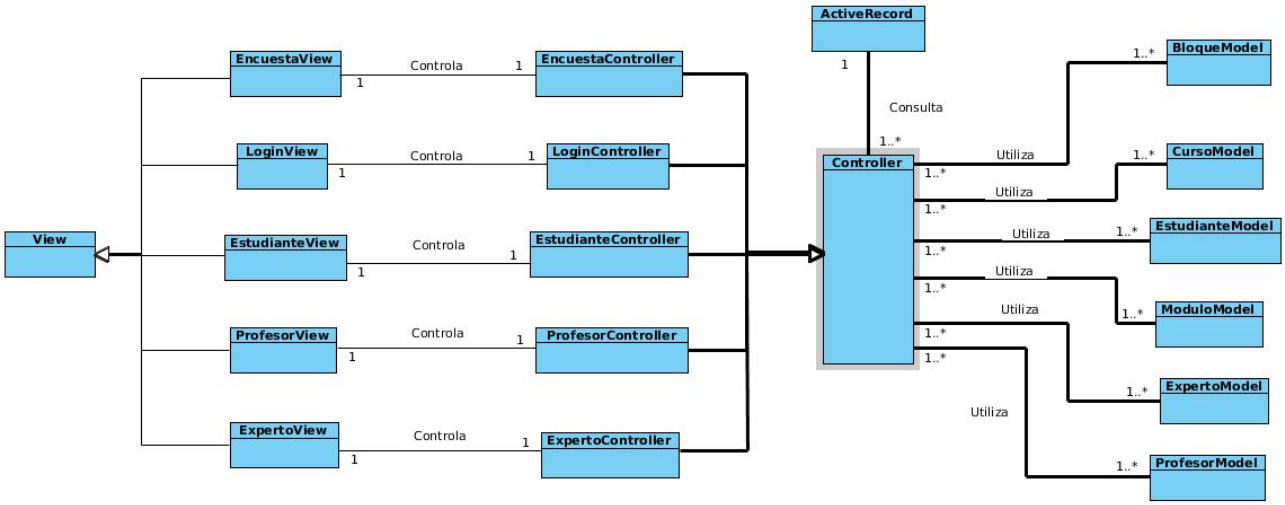
(ver modelo completo en ProyectoArandano.vpp)

6- Diagrama de secuencia de componentes del sistema



(ver diagrama completo en ProyectoArandano.vpp)

7- Bosquejo MVC



(ver modelo completo en ProyectoArandano.vpp)

Explicación:

- Se tiene las relaciones anteriores entre clases de tipo Vista, Modelo y Controlador.
- En general, todos los controladores usan un “Active Record”, el cual se utiliza para hacer llamados a consultas de la base de datos, con métodos que realizan el procedimiento de estas.

8- Lecciones Aprendidas

Como grupo en el proyecto, hemos tenido varias lecciones aprendidas con la experiencia en el desarrollo tanto del software, como el de los informes.

1) Trabajo en Grupo vs Trabajo Personal: El cambio de paradigma de trabajar de una u otra forma es grande. No es para nada lo mismo, ya que en esta parte se requiere de tener en coordinación al grupo en sí. Aun así, los tiempos para juntarse a desarrollar el proyecto nunca han sido perfectos (debido a diferencias en las asignaturas de la universidad), y se ha requerido de poder utilizar las herramientas de trabajo colaborativo

2) Para proyectos grandes, se requiere de dedicación: Como grupo, el proyecto de este semestre del ramo, ha sido el trabajo más grande y complejo que hemos tenido en nuestra carrera. Con esto, el pensar el cómo ir resolviendo el problema ha sido un desafío, ya que a la medida que uno avanza con el proyecto, van surgiendo más cosas de las cuales uno tiene que investigar, y buscar una solución para poder compatibilizarse con las tecnologias que estan utilizando(por ejemplo, si necesito logins, tendré que utilizar cookies. En ello, tenemos que aprender como utilizar un middleware de Node.Js para poder realizar esta parte, o en el peor caso ,tener que implementar

uno mismo la solución). Esto produce que uno como programador tenga que investigar constantemente sobre la mejor solución a acoplar a la plataforma, para poder lograr los requerimientos del cliente.

Otro punto, es que a medida que uno avanza con el proyecto, este va tomando una mayor cantidad de líneas de código, siendo cada vez más difícil de entender, y luego de ir programando o entendiendo de parte de los otros compañeros de trabajo. Un archivo donde nos ocurrió aquello, fue en el de las rutas (routes.js), donde lo usábamos como un patrón active record para poder realizar consultas. Pero más adelante, resultaba más efectivo realizarlas con un modelo.

3) Uso de herramientas Web: Con respecto a las herramientas Web a utilizar, fue interesante el ir desarrollando desde Framework Angular.Js. En un principio, se utilizaba para partes como las rutas, controladores y vistas. Pero a medida que se iba avanzando con el proyecto, se iban descubriendo muchos más usos que pudimos haber dado a Angular desde un principio, los cuales nos hubiesen ayudado a realizar el trabajo de forma más agilizada. Un ejemplo en nuestro caso, ha sido el usar las Factory de Angular, las cuales permiten de mejor forma implementar la parte M del modelo MVC.

4) Uso de herramientas Github y Trello: Con respecto a Github, ha sido una herramienta que uno individualmente puede entender como utilizarla, pero en grupos esta puede ser menos entendible. En los últimos momentos, y a base de experiencia e intentos, hemos entendido más como utilizar la herramienta, tanto con Branch y Merge al Master, pero en un principio ha sido difícil de comprender (motivo del cual, hemos perdido tiempo en aprender, o ver como sincronizarnos utilizando la herramienta). Aun así, hemos logrado entender para el entregable 3 mejores maneras de utilizarla, y nos parece algo muy útil al momento de ir actualizando nuestro código.
Con respecto a Trello, ha sido una herramienta indispensable. Es aquí donde podemos asignarnos que parte del trabajo se realiza por persona, y donde cada uno puede ir avanzando en el tiempo que logre tener.