DOCUMENTACIÓN TRABAJO INGENIERÍA DEL SOFTWARE

Ingeniería Informática

Ingeniería del Software

Roberto García Castro

Daniel Marín López

Índice

1. Práctica 2

1.1. Definición del problema.

En problema de nuestro de nuestro caso práctico, es que tenemos que almacenar los alumnos de la asignatura de Ingeniería del Software, es decir, tenemos que tener un registro de todos los alumnos matriculados, con todos sus datos, como por ejemplo su DNI, su nombre o su curso más alto al que está matriculado.

1.2. Extracción de Requisitos.

Para la extracción de requisitos, primero vamos a sacar los actores principales que interactúan en el sistema y los datos que necesitamos almacenar en nuestro sistema.

Los actores que interactúan en el sistema son los siguientes:

- Profesor, será el usuario que entre a la aplicación y modifique los datos de los alumnos.

- Alumno, serán los usuarios que estén almacenados en el sistema.

Dichos alumnos, tendrán los siguientes datos, los cuales necesitamos almacenar:

- DNI, será el DNI del alumno que tenemos guardado.

-Nombre, será el nombre del alumno que tenemos guardado.

-Apellido, será el apellido del alumno que tenemos guardado.

-Teléfono, será el teléfono del alumno que tenemos guardado

-Email_UCO, será el correo de la UCO del alumno que tenemos guardado.

-Dirección, será la calle del alumno que tenemos guardado.

-Curso más alto, será el curso más alto al que este matriculado el alumno.

-Fecha de nacimiento, será la fecha de nacimiento del alumno que tenemos guardado.

-Equipo, será el número del equipo al que pertenece el alumno que tenemos guardado.

-Líder o no, será una variable la cual nos dirá si el alumno es líder o no.

Teniendo ya definidos los actores y los datos que vamos a almacenar, vamos a sacar los requisitos del sistema, tanto los funcionales como los no funcionales, a dichos requisitos le vamos a asignar una prioridad en función de la importancia de estos en el sistema.

- Requisitos

  - Requisitos funcionales

    -RF1. El sistema deberá poder buscar un alumno dentro de su base de datos. (Prioridad: 1)

    -RF2. El sistema deberá tener la opción de poder introducir a un nuevo alumno en el sistema. (Prioridad: 2)

    -RF3. El sistema deberá tener la opción de poder modificar a un alumno ya existente en el sistema. (Prioridad: 3)

    -RF4. El sistema deberá tener la opción de poder eliminar a un alumno del sistema. (Prioridad: 3)

    -RF5. El sistema deberá poder guardar datos procedentes de un fichero binario. (Prioridad: 4)

    -RF6. El sistema deberá poder cargar datos procedentes de un fichero binario. (Prioridad: 4)

    -RF7. El sistema deberá tener la opción de realizar una copia de seguridad de todos los datos del sistema. (Prioridad: 5)

-RF8. El sistema deberá tener la opción de cargar una copia de seguridad de todos los datos del sistema. (Prioridad: 5)

-RF9. El sistema deberá tener la opción de imprimir por pantalla todos los alumnos del sistema, ordenados por la variable que el usuario desee, como el DNI, nombre, apellido o curso más alto, de forma ascendente o descendente, a la vez tiene que generar un fichero MarkDown, donde vendrán impresos los datos de los alumnos. (Prioridad: 6)

-RF10. El sistema deberá pedir obligatoriamente todos los datos del alumno, al introducir un nuevo usuario, excepto el equipo y si es líder o no. (Prioridad: 7)

-RF11. El sistema deberá pedir el DNI si al realizar una búsqueda, se encuentra a más de un usuario. (Prioridad: 7)

-RF12. El sistema no deberá tener más de un líder por grupo. (Prioridad: 8)

-RF13. El sistema no deberá de auto rellenar los datos de un nuevo alumno. (Prioridad: 8)

-RF14. El sistema deberá de tener dos tipos de usuarios los cuales pueden acceder al sistema, un coordinador y ayudantes. (Prioridad: 9)

-RF15. El sistema deberá dejar acceso a todas las funciones a ambos usuarios excepto las copias de seguridad, las cuales solo podrán ser realizadas por el coordinador. (Prioridad: 9)

- Requisitos no funcionales

-RNF1. El sistema deberá de estar disponible para Linux. (Prioridad: 1)

-RNF2. El sistema deberá tener capacidad para un máximo de 150 alumnos. (Prioridad: 2)

-RNF3. El sistema deberá guardar toda la información en un fichero binario. (Prioridad: 3)
-RNF4. El sistema deberá tener una interfaz por línea de comandos. (Prioridad: 3)
-RNF5. El sistema solo podrá ser usado por el profesor. (Prioridad: 4)
-RNF6. El sistema no deberá de tener un límite de alumnos por equipo. (Prioridad: 4)
-RNF7. El sistema deberá estar codificado en c++. (Prioridad: 4)

1.3. Historias de usuario

Ahora vamos a realizar las diferentes historias de usuario que va a tener nuestro sistema, las cuales van a estar ordenadas por prioridad.

- Historia 1:
  (ANVERSO)
  ID: 001 Buscar Alumno.
  Como usuario quiero que el sistema busque un alumno al introducirle el DNI, el apellido.
  Prioridad: 1.

  (REVERSO)
  -Quiero poder buscar un alumno cuando comparando el DNI o el apellido introducido con todos los de la base de datos.
  -En caso de que haya más de un usuario al buscar por apellido, se le pedirá el DNI.

- Historia 2:
  (ANVERSO)
  ID: 002 Introducir nuevo alumno.
  Como usuario quiero poder introducir un nuevo alumno.
  Prioridad: 2.

(REVERSO)

-Quiero poder tener la opción de meter un nuevo usuario en el sistema.

-Al seleccionar esta opción, el sistema me tiene que dejar introducir todos los datos obligatoriamente, excepto el número del equipo y si es líder o no.

-Quiero que no se autor rellenen los datos de forma automática.

- Historia 3:

  (ANVERSO)

  ID: 003 Modificar alumno

  Como usuario quiero poder modificar a un alumno ya existente.

  Prioridad: 3.

  (REVERSO)

  -Quiero poder modificar algunos datos de un alumno que ya esté en el sistema.

  -En caso de modificar el número del grupo, no habrá número de máximo de estos.

  -En caso de modificar si es líder o no, en un mismo grupo, solo puede haber un líder como máximo.

- Historia 4:

  (ANVERSO)

  ID: 004 Eliminar usuario

  Como usuario quiero poder eliminar a un usuario ya existente.

  Prioridad: 3.

  (REVERSO)

-Quiero poder eliminar a un usuario del sistema después buscarlo en el sistema.
-Cuando se elimine un usuario, también se tiene que borrar del grupo en el que estaba.

- Historia 5:
  (ANVERSO)
  ID: 005 Cargar fichero
  Como usuario quiero poder cargar todos los datos en un fichero.
  Prioridad: 4.

  (REVERSO)
  -Quiero poder tener la opción de cargar todos los datos de mi sistema en un fichero binario.

- Historia 6:
  (ANVERSO)
  ID: 006 Descargar fichero
  Como usuario quiero poder descargar todos los datos de un fichero.
  Prioridad: 4.

  (REVERSO)
  -Quiero poder tener la opción de descargar todos los datos de un fichero binario en mi sistema.

- Historia 7:
  (ANVERSO)
  ID: 007 Visualizar grupo
  Como usuario quiero poder visualizar todos los integrantes de un grupo.
  Prioridad: 5.

(REVERSO)
-Quiero poder visualizar todos los integrantes de un grupo que haya introducido.
-Quiero que salga marcado de alguna forma quien es el líder de ese grupo.

- Historia 8:
  (ANVERSO)
  ID: 008 Imprimir datos
  Como usuario quiero poder visualizar todos los usuarios ordenados y a la vez quiero que se genere un fichero MarkDown con dichos datos.
  Prioridad: 6.

  (REVERSO)
  -Quiero poder visualizar todos los usuarios ordenados de forma ascendente o descendente, por el DNI, nombre, apellido o curso más alto.
  -Quiero que se genere un documento MarkDown donde van a estar todos los datos de los usuarios del sistema ordenados de forma ascendente o descendente, por el DNI, nombre, apellido o curso más alto.

- Historia 9:
  (ANVERSO)
  ID: 009 Copia de seguridad
  Como usuario quiero poder realizar una copia de seguridad en un fichero.
  Prioridad: 6.

  (REVERSO)
  -Quiero que se realice una copia de seguridad, de todos los usuarios del sistema, en un fichero binario.

-El nombre del fichero copia será introducido por el usuario, junto con la fecha en la que se realizó la copia.

- Historia 10:

  (ANVERSO)

  ID: 010 Cargar copia de seguridad

  Como usuario quiero poder cargar en el sistema todos los datos de un fichero binario.

  Prioridad: 6.

  (REVERSO)

  -Quiero que se cargue una copia de seguridad desde un fichero binario.

  -El nombre del fichero copia será introducido por el usuario.

- Historia 11:

  (ANVERSO)

  ID: 011 Registrar Profesor

  Como usuario quiero poder agregar nuevos profesores que no se encuentren en el sistema e indicar si son coordinador o ayudante.

  Prioridad: 6.

  (REVERSO)

  -Quiero poder agregar nuevos profesores al sistema.

  -Quiero que se pueda indicar si un profesor es coordinador o ayudante.

- Historia 12:

  (ANVERSO)

  ID: 012 Iniciar sesión

  Como usuario quiero poder iniciar sesión en el sistema.

  Prioridad: 6.

(REVERSO)

-Quiero que se pueda iniciar sesión en el sistema.

1.4. Casos de uso

Por ultimo tendiendo ya las historias de usuario vamos a realizar, los casos de uso que va a tener nuestro sistema.

- Caso de uso 1:

  Nombre: Buscar Alumno.

  ID: 01.

  Breve Descripción: El sistema deberá buscar a un usuario en el fichero.

  Actores Principales: Profesor.

  Actores Secundarios: Alumno.

  Precondiciones:

  -

  Flujo Principal:

  1. El sistema le pide al profesor el DNI o el apellido de la persona que está buscando.
  2. El profesor introduce el DNI o el apellido.
  3. Si se ha introducido el apellido y hay más de un alumno con el mismo apellido.
     - 3.1. El sistema pide un DNI.
     - 3.2. El profesor introduce un DNI.
     - 3.3. El sistema comprueba que el DNI está entre los alumnos con el mismo apellido.
     - 3.4. Mientras el DNI no esté entre esos alumnos.
       - 3.4.1. El sistema pide un nuevo DNI.
       - 3.4.2. El profesor introduce un nuevo DNI.
       - 3.4.3. El sistema comprueba si el DNI está entre esos alumnos.

4. El sistema devuelve los datos del alumno que ha encontrado.

Postcondiciones:

-El sistema devuelve los datos del alumno.

Flujos alternativos:

-Si no se ha encontrado un alumno devuelve un mensaje de error.

- Caso de uso 2:

Nombre: Introducir nuevo alumno.

ID: 02.

Breve Descripción: El sistema deberá cargar añadir un nuevo alumno al fichero.

Actores Principales: Profesor.

Actores Secundarios: Alumno.

Precondiciones:

-No hay dentro del sistema más de 150 alumnos.

-El alumno no debe de estar en el sistema.

Flujo Principal:

1. El caso de uso empieza cuando el usuario le pida al sistema la opción de "introducir nuevo alumno".

2. El sistema pide al profesor que introduzca el DNI, nombre, apellido, teléfono, email de la UCO, dirección, curso más alto, fecha de nacimiento, equipo y líder del alumno.

3. El profesor introduce todos los datos obligatoriamente, excepto el grupo y si es líder o no.

4. El sistema guarda todos los datos del nuevo alumno.

Postcondiciones:

-El sistema tendrá los datos del nuevo alumno.

Flujos alternativos:

-El sistema detecta que el DNI ya está en el sistema y lo notifica con un mensaje de error.

- Caso de uso 3:

Nombre: Modificar alumno.

ID: 03.

Breve Descripción: El sistema deberá modificar los datos de un alumno del sistema.

Actores Principales: Profesor.

Actores Secundarios: Alumno.

Precondiciones:

-El usuario tiene que estar en sistema.

Flujo Principal:

1. El caso de empieza cuando el profesor selecciona la opción de "Modificar Alumno".
2. El sistema le pedirá al profesor el DNI o el apellido del alumno que desea modificar.
3. El profesor introducirá el DNI o el apellido del alumno.
4. El sistema buscará al alumno que coincida con el DNI que ha pasado el profesor.
5. El sistema le pedirá al profesor que introduzca los datos que desea modificar.
6. El profesor introduce que datos desea modificar.
7. El sistema le pedirá dichos datos al profesor.
8. El profesor introducirá los datos que va a modificar.
9. El sistema cambiará los datos que tenía antes por los que ha introducido el profesor.

Postcondiciones:

-El sistema tendrá guardados los cambios realizados.

Flujos alternativos:

-Si el profesor decide modificar el dato de líder, y ya hay un líder en el grupo, el sistema informará de un error.

- Caso de uso 4:

Nombre: Eliminar alumno.

ID: 04.

Breve Descripción: El sistema deberá eliminar a un alumno del sistema.

Actores Principales: Profesor.

Actores Secundarios: Alumno.

Precondiciones:

-El usuario deberá estar registrado en el sistema.

Flujo Principal:

1. El caso de empieza cuando el profesor selecciona la opción de "Eliminar alumno".
2. El sistema pedirá al profesor el DNI o el apellido del alumno que desea eliminar.
3. El profesor introduce el DNI o el profesor.
4. El sistema busca dicho alumno y lo elimina del sistema.

Postcondiciones:

-El alumno deberá estar eliminado del sistema.

Flujos alternativos:

-

- Caso de uso 5:

Nombre: Guardar Fichero.

ID: 05.

Breve Descripción: El sistema deberá guardar los datos en un fichero binario que el profesor quiera.

Actores Principales: Profesor.
Actores Secundarios: -

Precondiciones:

-

Flujo Principal:

1. El caso de uso empieza cuando el profesor selecciona la opción de "Guardar fichero".
2. El sistema le preguntará al usuario el fichero binario que desea guardar.
3. El usuario le indicará cual es el fichero que desea guardar.
4. El sistema guarda en el fichero todos los datos de memoria.

Postcondiciones:

-El fichero contendrá todos los datos del sistema.

Flujos alternativos:

-

- Caso de uso 6:

Nombre: Cargar Fichero.
ID: 06.

Breve Descripción: El sistema deberá cargar los datos de un fichero binario que el profesor quiera.

Actores Principales: Profesor.
Actores Secundarios: -

Precondiciones:

- El profesor no deberá tener los datos del fichero binario en el sistema.

Flujo Principal:

1. El caso de uso empieza cuando el profesor selecciona la opción de "Cargar fichero".
2. El sistema le preguntará al usuario el fichero binario donde están los datos que desea cargar.
3. El usuario le indicará cual es el fichero con esos datos.
4. El sistema carga en memoria todos los datos del fichero.

Postcondiciones:

-La memoria tiene que tener todos los datos del fichero.

Flujos alternativos:

-El sistema dará un mensaje de error si no encuentra el fichero deseado.

- Caso de uso 7:

  Nombre: Visualizar grupo.

  ID: 07.

Breve Descripción: El sistema deberá visualizar el grupo que el profesor desee.

Actores Principales: Profesor.

Actores Secundarios: Alumno.

Precondiciones:

-El grupo debe existir, dentro del sistema.

Flujo Principal:

1. El caso de uso empieza cuando el profesor selecciona la opción de "Visualizar grupo".

2. El sistema le preguntará al usuario el grupo que desea visualizar.

3. El usuario le indicará cual es el grupo que desea ver.

Postcondiciones:

-El sistema mostrará el grupo seleccionado.

Flujos alternativos:

-

- Caso de uso 8:

Nombre: Imprimir por Pantalla.
ID: 08.

Breve Descripción: El sistema deberá imprimir todos los datos de los alumnos y generar un fichero MarkDown con ellos.

Actores Principales: Profesor.
Actores Secundarios: Alumno.

Precondiciones:

-

Flujo Principal:

1. El caso de uso empieza cuando el profesor selecciona la opción de "Imprimir por pantalla".

2. El sistema imprimirá por pantalla todos los datos de los alumnos.

3. El sistema creará un fichero Markdown donde estarán todos los datos del sistema.

Postcondiciones:

-Se habrá creado el fichero MarkDown

Flujos alternativos:

-En caso de estar el sistema vacío, se imprimirá un mensaje de error.

- Caso de uso 9:

Nombre: Hacer Copia de Seguridad.

ID: 09.

Breve Descripción: El sistema deberá realizar una copia de seguridad de todos los alumnos del sistema en un fichero binario.

Actores Principales: Profesor.

Actores Secundarios: -

Precondiciones:

-El sistema no puede estar vacío.

Flujo Principal:

1. El caso de uso empieza cuando el profesor selecciona la opción de "hacer copia de seguridad".
2. El sistema genera un fichero binario con todos los datos de los alumnos cuyo nombre será la fecha de ese día.

Postcondiciones:

- El fichero será generado de forma correcta.

Flujos alternativos:

-

- Caso de uso 10:

  Nombre: Cargar Copia de Seguridad

  ID: 10

Breve Descripción: El sistema deberá cargar una copia de seguridad en el sistema.

Actores Principales: Profesor.

Actores Secundarios: -

Precondiciones:

- El sistema tiene que estar vacío.

Flujo Principal:

1. El caso de uso empieza cuando el profesor selecciona la opción de "cargar copia de seguridad".
2. El sistema cargará en la lista todos los datos del fichero binario.

Postcondiciones:

-El sistema contendrá todos los datos del fichero binario.

Flujos alternativos:

-

- Caso de uso 11:

  Nombre: Registrar Profesor.

  ID: 11.

Breve Descripción: El sistema deberá registrar un nuevo profesor e indicar si este es coordinador o ayudante.

Actores Principales: Profesor.
Actores Secundarios: -
Precondiciones:
- El profesor no deberá estar en el sistema.

Flujo Principal:
1. El caso de uso empieza cuando el profesor selecciona la opción de "Registrar".
2. El profesor meterá sus datos en el sistema e indicará si es coordinador o no.
3. El sistema creará un perfil con sus datos y lo dará de alta.

Postcondiciones:
- El profesor quedará registrado en el sistema.

Flujos alternativos:
-

- Caso de uso 12:

Nombre: Iniciar sesión.
ID: 12.

Breve Descripción: El sistema deberá dejar al usuario iniciar sesión.

Actores Principales: Usuario.
Actores Secundarios: -

Precondiciones:
- El usuario no habrá iniciado sesión todavía.

Flujo Principal:
1. El caso de uso empieza cuando el usuario elige la opción "Iniciar sesión".
2. El sistema le preguntará sus datos.

3. El sistema comprobará si los datos son correctos.

4. Si son correctos, este le dará el acceso a su perfil.

Postcondiciones:

- El usuario tendrá acceso al sistema.

Flujos alternativos:

-Si los datos son incorrectos, este le dará un error y tendrá que meter los datos de nuevo.

2. Practica 3

2.1. Diagrama de clases

Ahora teniendo los diferentes requisitos y los casos de uso, vamos a crear las diferentes clases que va a tener nuestro sistema.

La primera clase que vamos a tener, es la clase persona, la cual va a contener todos los datos que sean iguales tanto para los alumnos, como para los profesores, los cuales son los siguientes: DNI, Nombre, Apellido, Teléfono, Email, Dirección y Fecha de nacimiento, y como operaciones tendrá los observadores y los modificadores de dichos atributos.

Después vamos a tener la clase alumno la cual va a heredar de la clase persona todos los atributos y las operaciones, y además de estos va a tener los siguientes atributos: Curso más alto, Equipo y Líder y como operaciones tendrá los observadores y modificadores de los atributos.

Ahora vamos a definir la clase agenda, la cual va a contener como atributo solo un vector de alumnos y tendrá las siguientes operaciones:

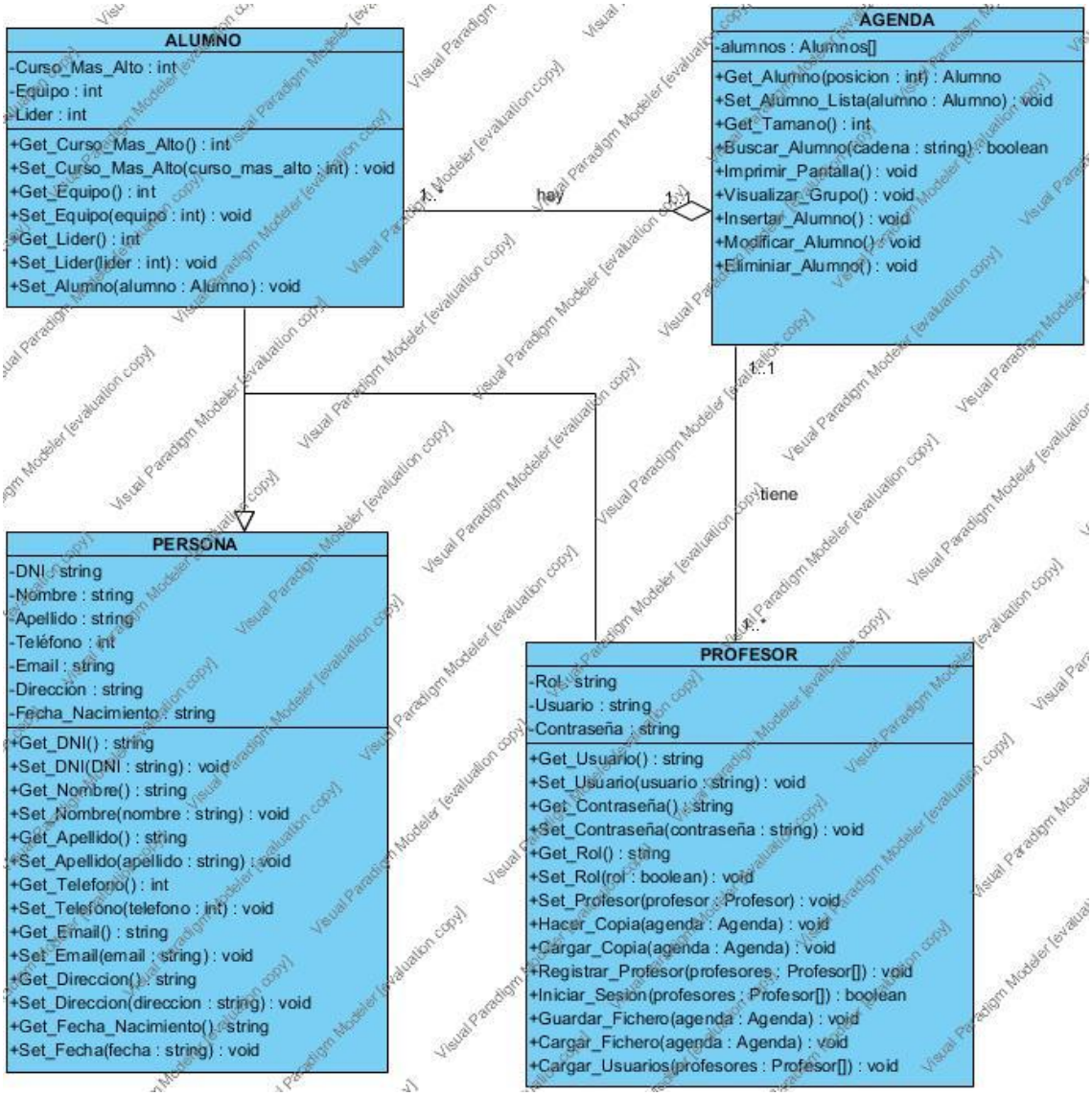
- Get alumno que devuelve el alumno que está en la posición que se le pasa como parámetro.
- Set alumno, la cual mete en el vector el alumno que se ha introducido como parámetro.
- Get tamaño, la cual devuelve el tamaño del vector.
- Buscar alumno, la cual devuelve verdadero si encuentra la cadena introducida en el vector de alumnos o falso en caso contrario.
- Imprimir pantalla, la cual va a imprimir por el terminal todos los alumnos del sistema y va a generar un fichero MarkDown donde también van a estar metidos dichos alumnos.
- Visualizar grupo, la cual va a imprimir por pantalla todos los integrantes de un grupo.
- Insertar alumnos, la cual va a introducir un nuevo alumno en el vector de alumnos.
- Modificar alumno, la cual va a modificar los datos de un alumno.

- Eliminar grupo, la cual va a eliminar un alumno del vector alumnos.

Por ultimo vamos a definir la clase Profesor, la cual también va a heredar de la clase persona y además tiene los siguientes atributos propios: Rol, Usuario y Contraseña y las siguientes operaciones:

- Los modificadores y los observadores de los atributos.
- Hacer copia, la cual va a crear una copia de seguridad de nuestro sistema y va a tener como parámetro la agenda que va a hacer la copia.
- Cargar copia, la cual va a cargar una copia de seguridad en el sistema.
- Registrar profesor, la cual va a meter en el vector que se pasa como parámetro un nuevo profesor, además de dentro del fichero de usuarios.
- Iniciar sesión, la cual va a devolver verdadero si el profesor que hemos introducido está en el vector de profesores que hemos pasado como parámetro o falso en caso contrario.
- Guardar fichero, la cual va a guardar toda la agenda que se le ha pasado por parámetro en un fichero.
- Copiar fichero, la cual va a cargar todos los datos de un fichero en la agenda que tiene como parámetro.
- Cargar usuarios, la cual va a meter en el vector que tiene como parámetro todos los datos del fichero que contiene los usuarios.

Tras especificar las clases del sistema vamos a realizar el diagrama de clases, el cual va a quedar de la siguiente forma.

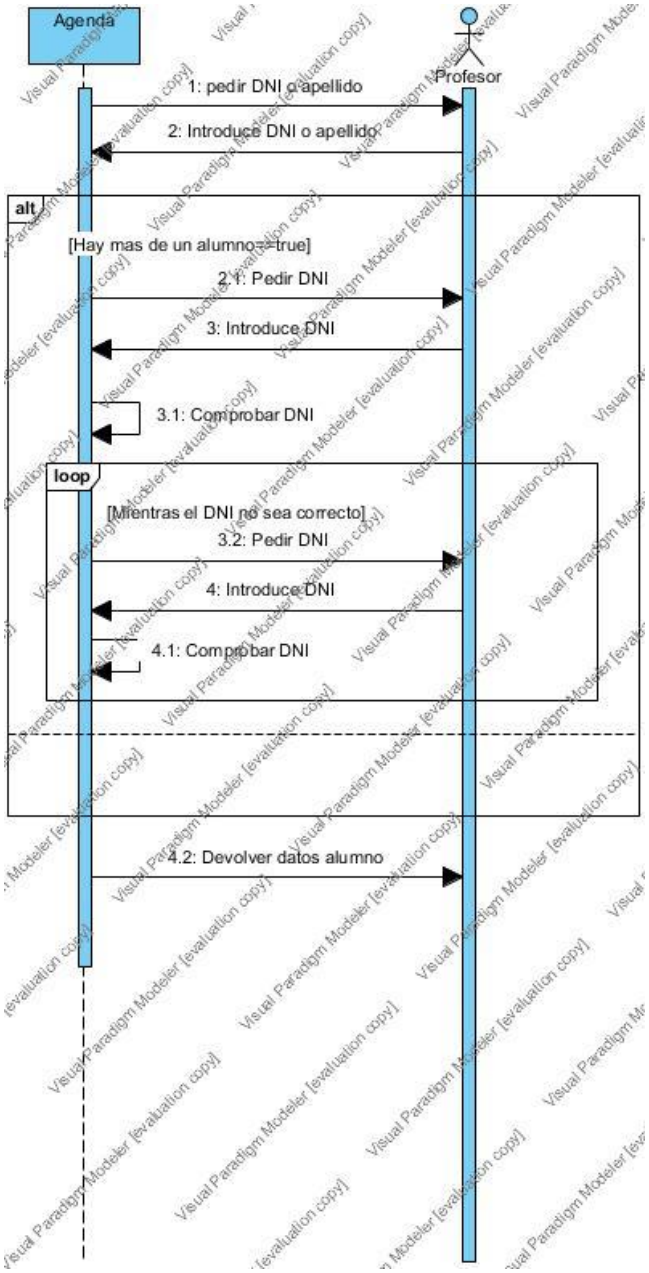


En dicho diagrama vemos que la clase alumno tiene una agregación con la clase agenda, ya que la clase agenda contiene un vector de alumnos. También podemos observar que la clase profesor tiene una relación con la clase agenda de 1..1 1..*, ya que una agenda puede estar controlada por uno o varios profesores, mientras que un profesor tiene solo una agenda.

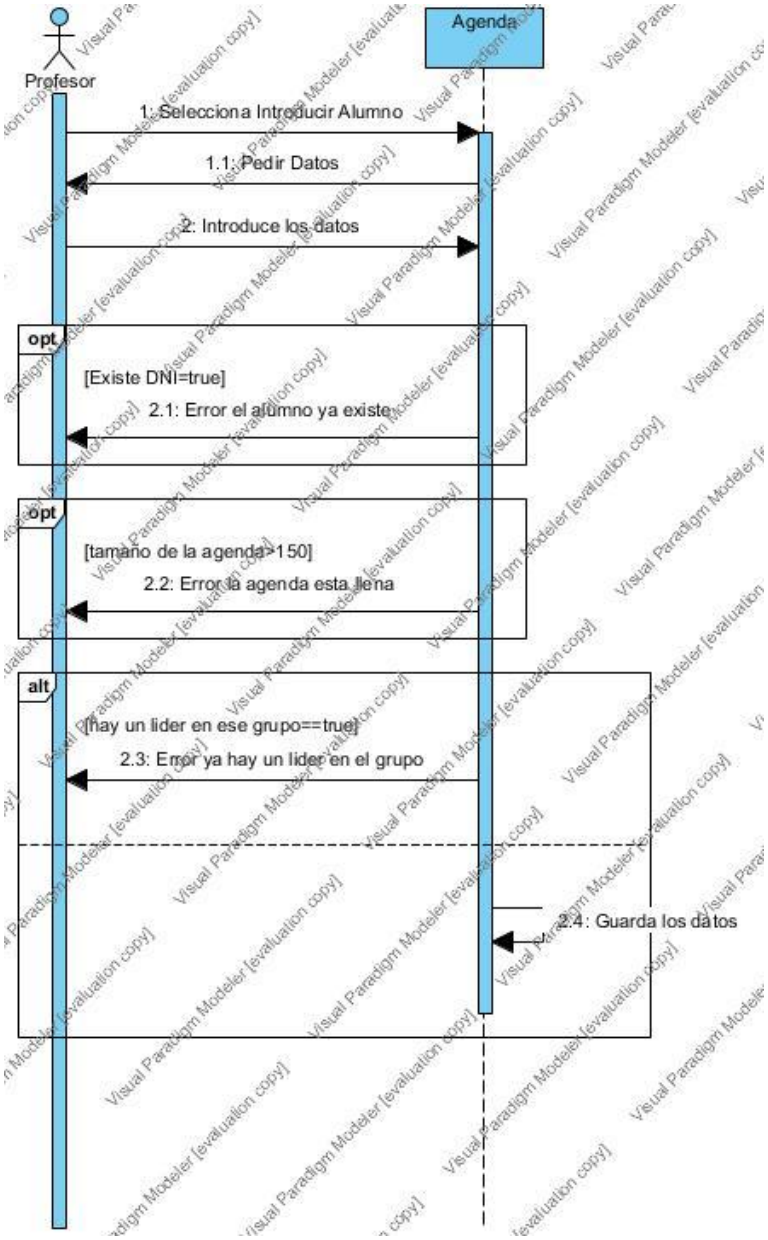
2.2. Diagramas de secuencia

Ahora vamos a realizar los diagramas de secuencia a partir de los casos de uso.

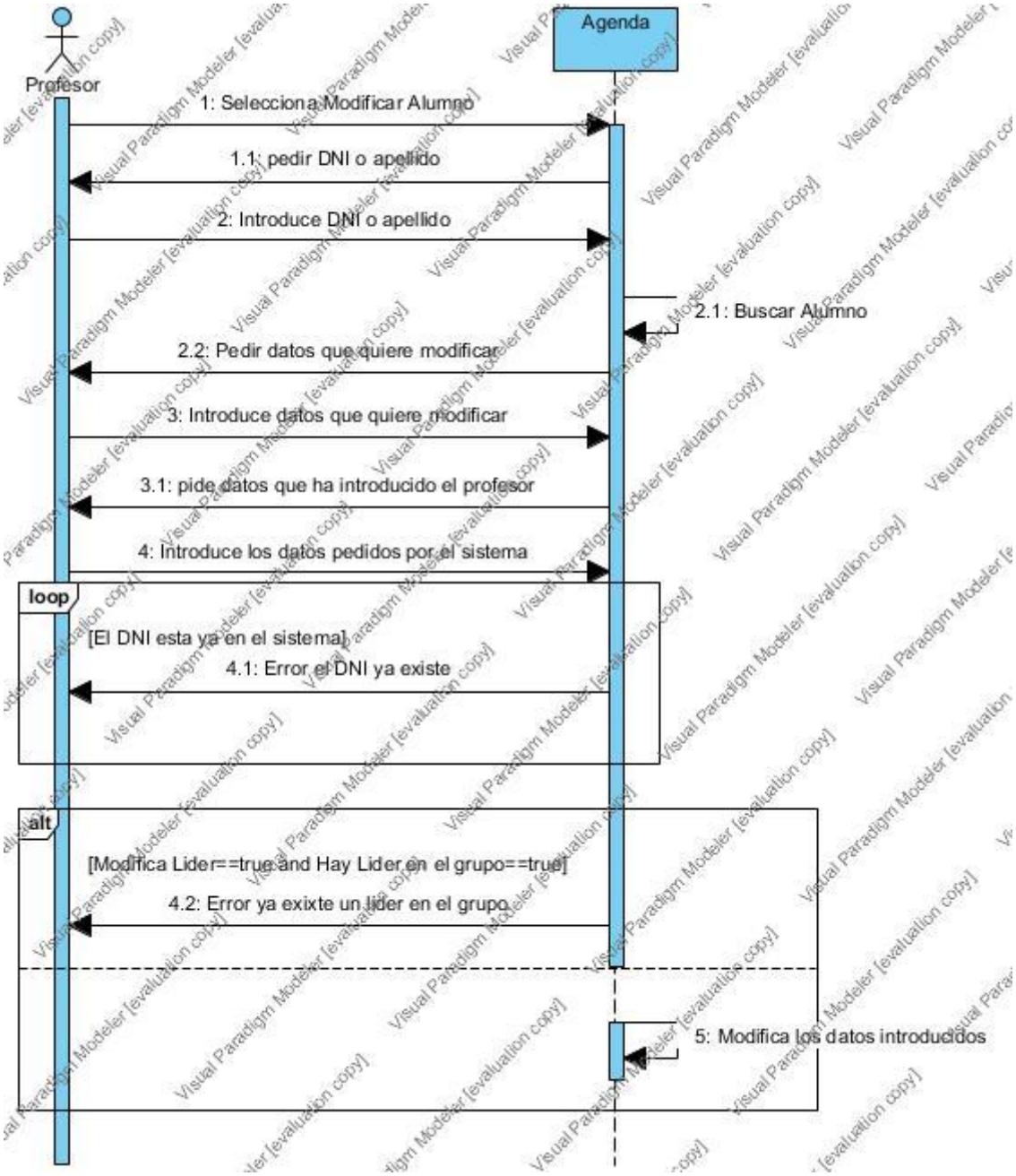
- Caso de uso 1: Buscar alumno



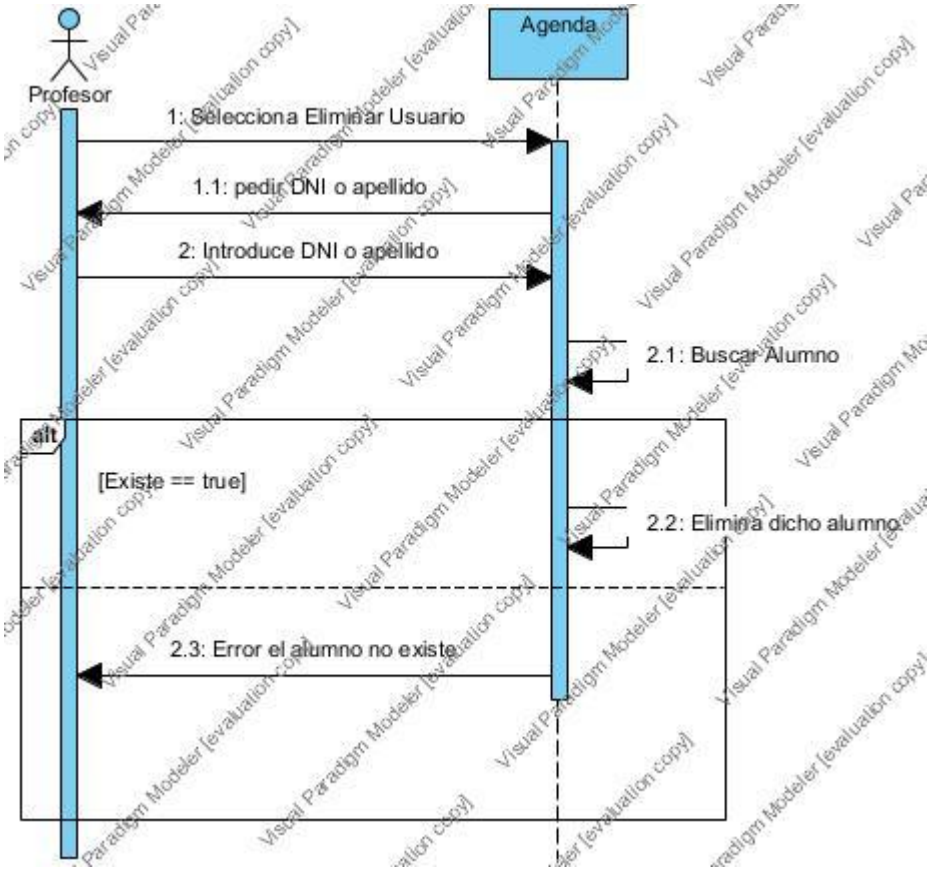
- Caso de uso 2: Introducir nuevo alumno



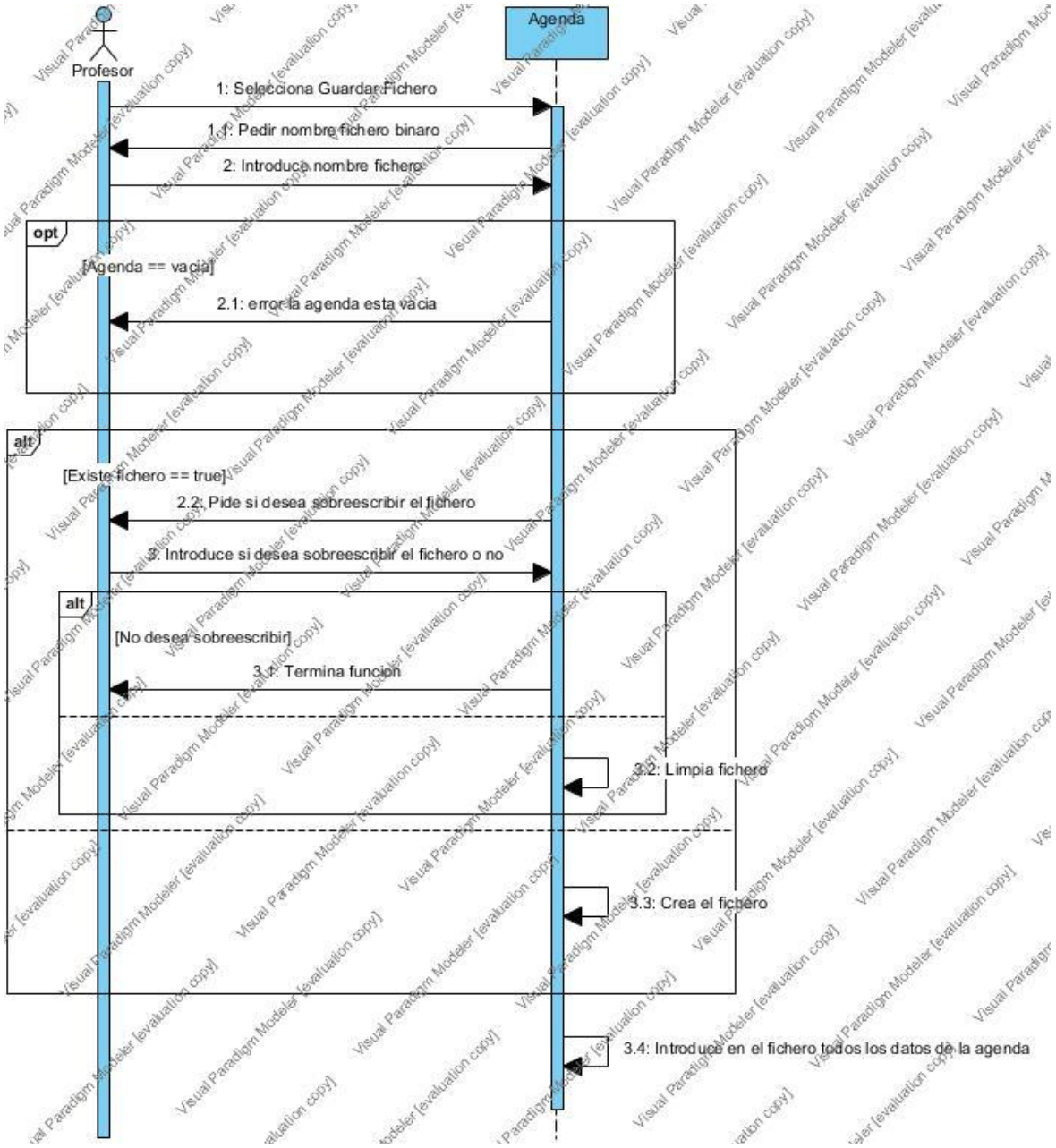
- Caso de uso 3: Modificar alumno



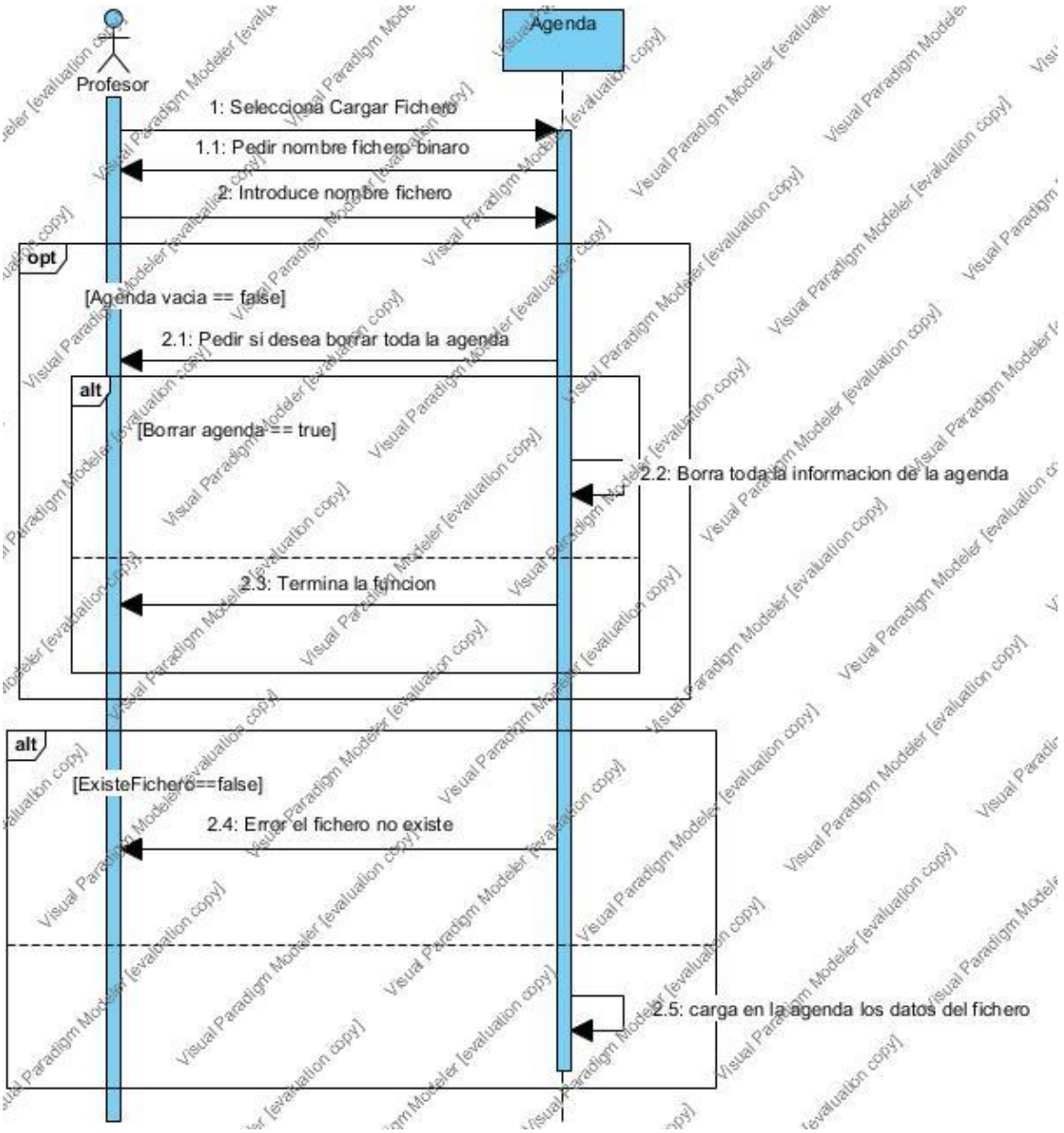
- Caso de uso 4: Eliminar alumno



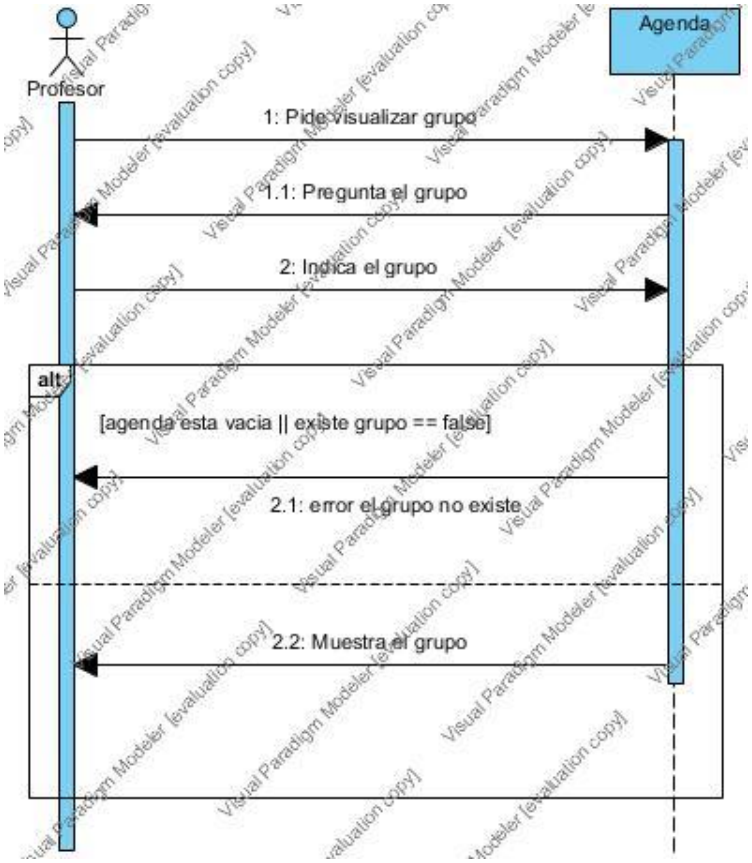
- Caso de uso 5: Guardar fichero



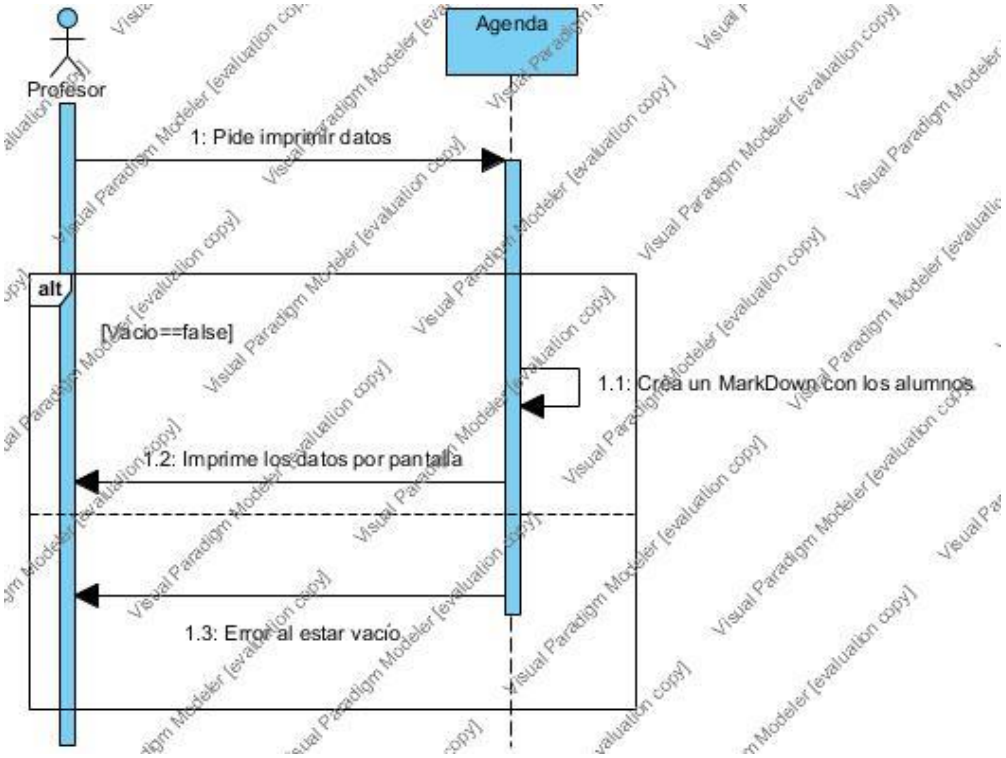
- Caso de uso 6: Cargar fichero



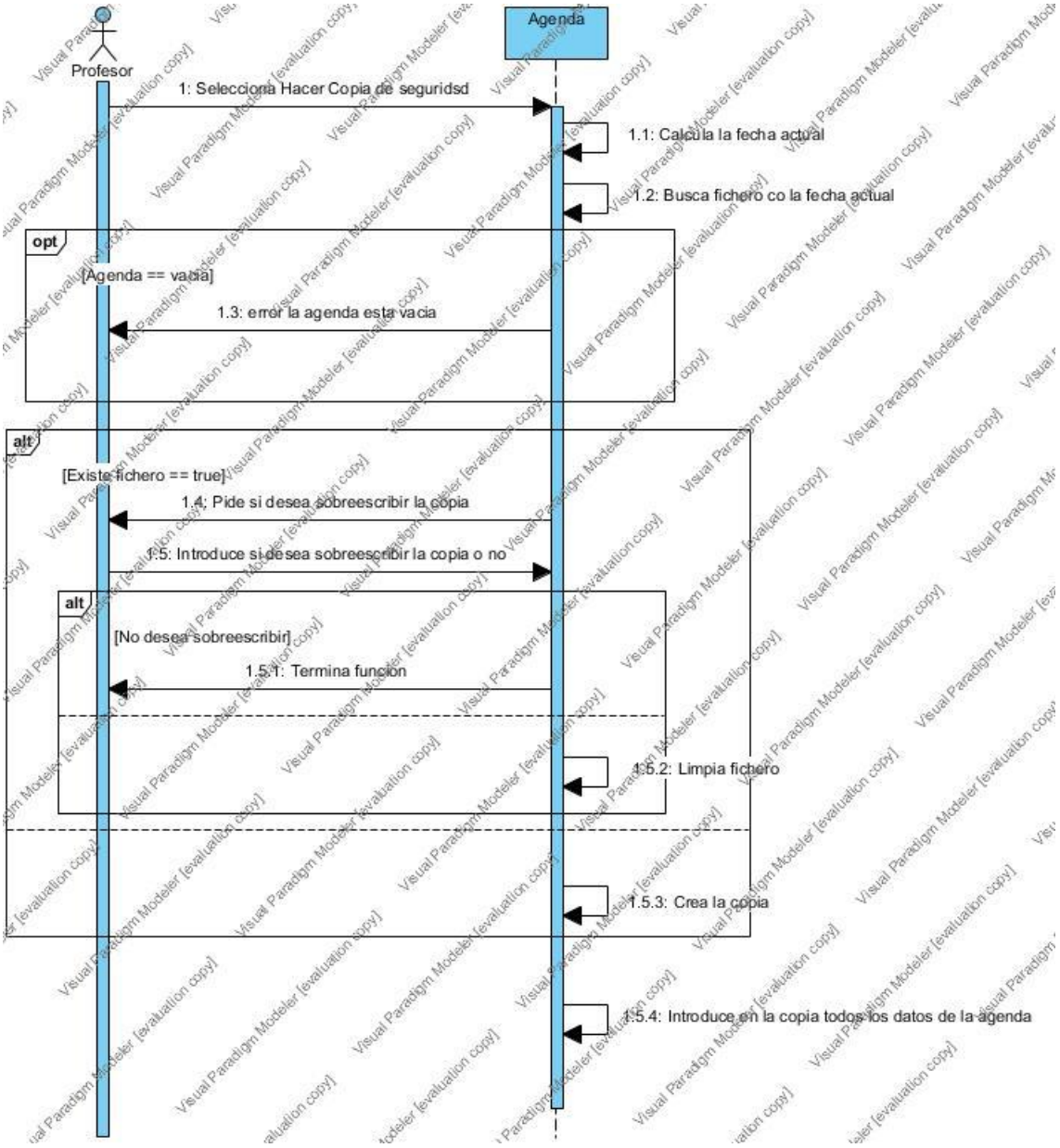
- Caso de uso 7: Visualizar grupo



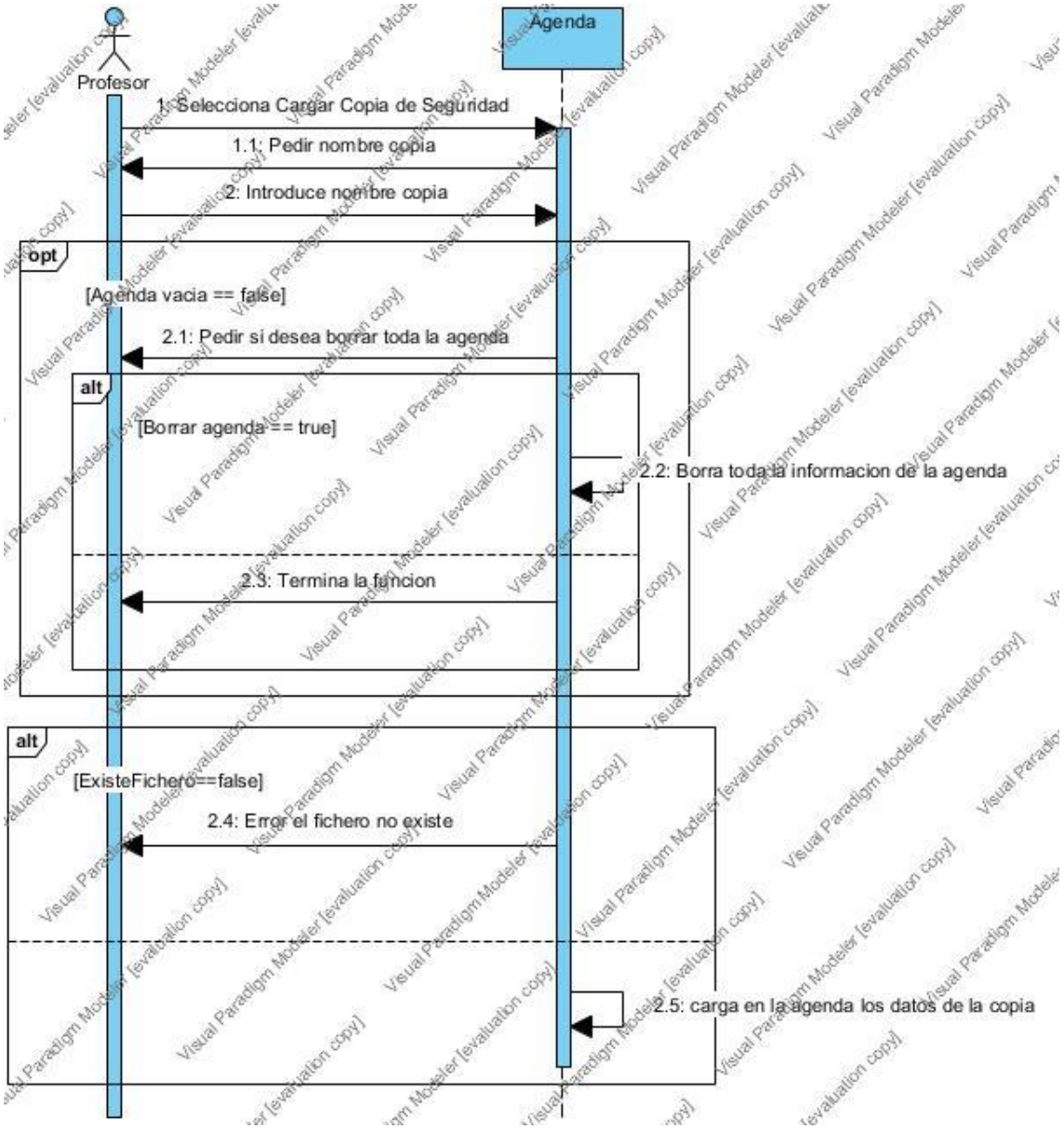
- Caso de uso 8: Imprimir por pantalla



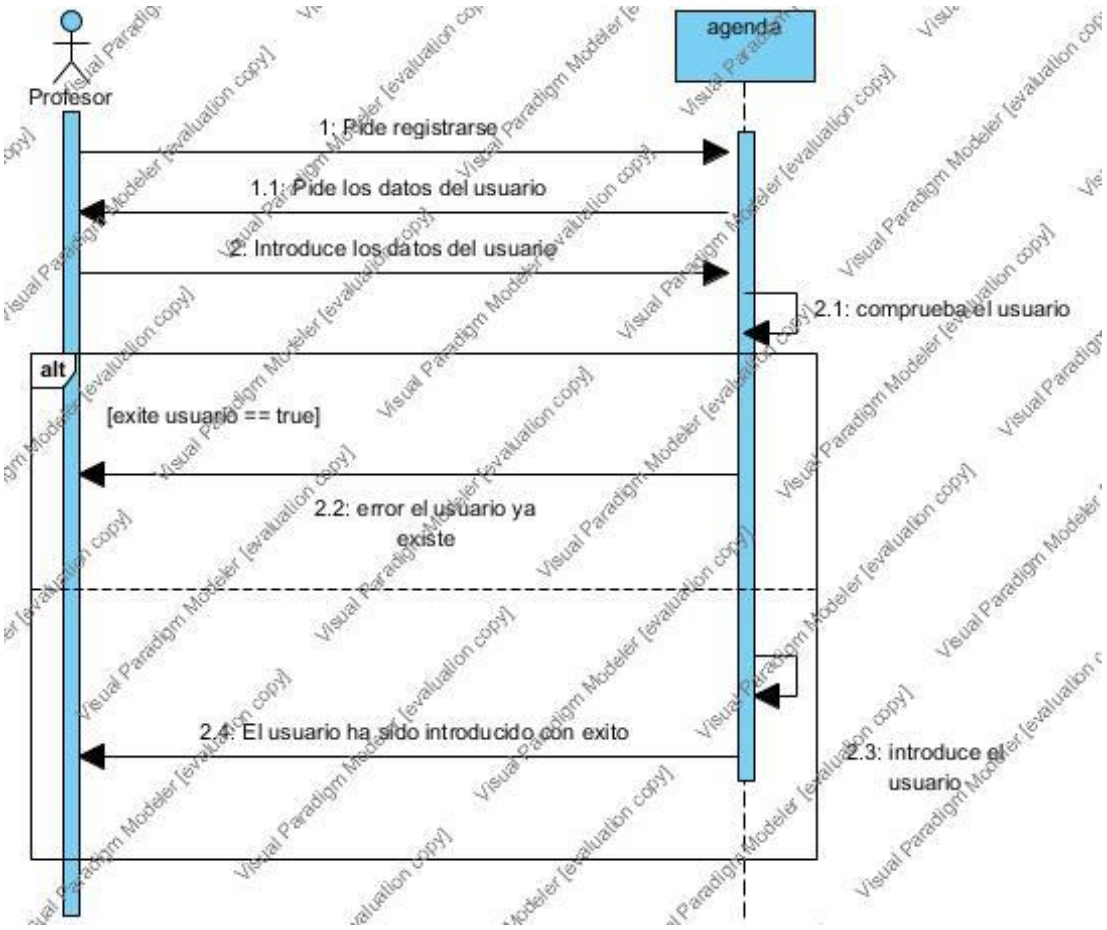
- Caso de uso 9: Hacer copia de seguridad



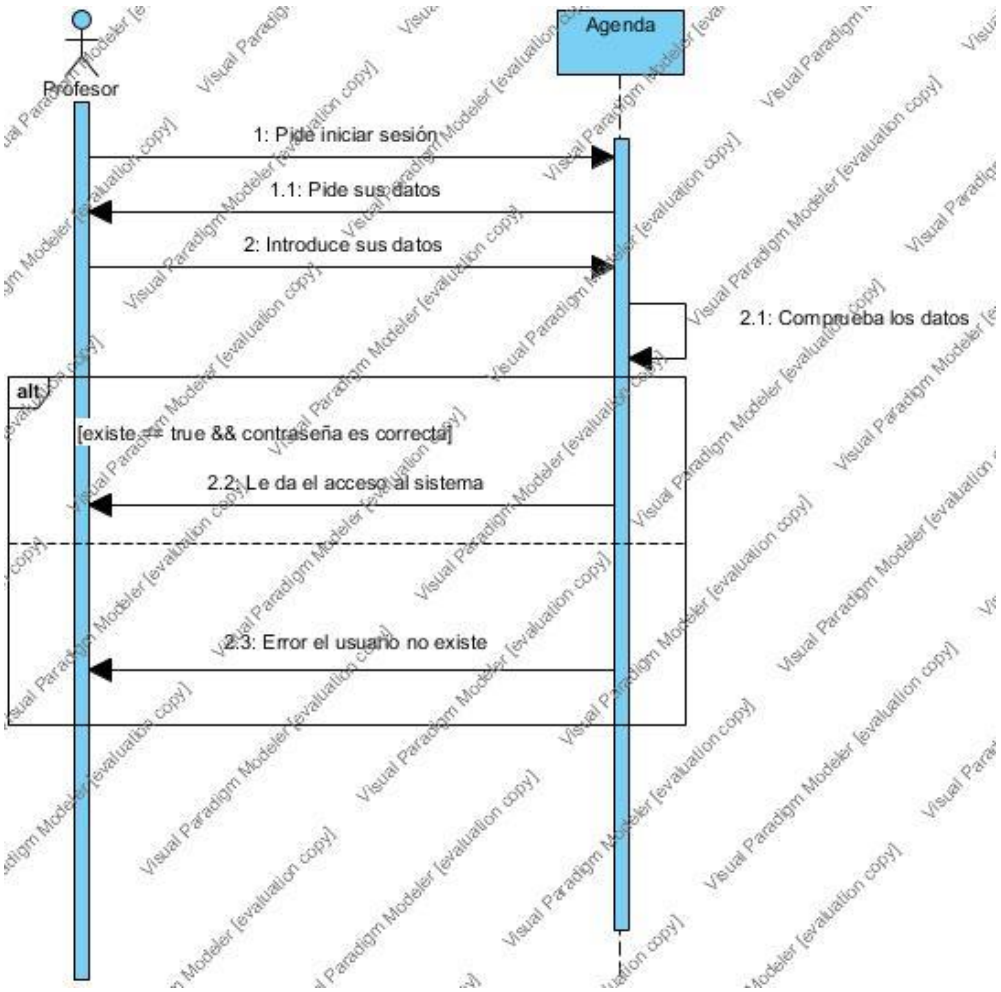
- Caso de uso 10: Cargar copia de seguridad



- Caso de uso 11: Registrar profesor



- Caso de uso 12: Iniciar sesión



3. Práctica 4.

3.1. Metodología SCRUM

Ahora vamos a realizar la metodología SCRUM, donde vamos a realizar el Product Backlog, el Burndown Chart y dos Sprint Backlogs, ya que vamos a realizar dos sprints.

- Product Backlog

Aquí pondremos las funciones que tendremos que realizar, junto con su prioridad y su duración aproximada, en horas, nuestro Product Backlog es así:

Buscar Alumno, prioridad: 0. Duración: 4.
Introducir nuevo alumno, prioridad 1. Duración: 2.
Modificar alumno, prioridad: 2. Duración: 3.
Eliminar usuario, prioridad: 2. Duración: 2.
Cargar fichero, prioridad: 3. Duración: 2.
Guardar fichero, prioridad: 3. Duración: 2.
Visualizar grupo, prioridad: 4. Duración: 2.
Imprimir datos, prioridad: 4. Duración: 4.
Hacer Copia de seguridad, prioridad: 5. Duración: 2.
Cargar copia de seguridad, prioridad: 5. Duración: 2.
Registrar Profesor, prioridad: 6. Duración: 3.
Iniciar sesión, prioridad: 6. Duración: 4.

- Burndown Chart

El Burndown Chart es una gráfica en la que vamos a representar el tiempo que hemos invertido en el proyecto cada cierto intervalo de tiempo. Nuestra grafica tendrá la siguiente forma:

En nuestro caso hemos empezado con un tiempo inicial de 32 horas y hemos realizado 6 intervalos en los que hemos calculado cuantas horas hemos invertido.

- Sprint Backlog

En el Sprint Backlog vamos a poner las funciones que va a realizar cada integrante del grupo en el sprint, en nuestro caso vamos a realizar 2 sprints. Los Sprints Backlogs van a quedar de la siguiente forma:

-Sprint Backlog 1

En el primer sprint vamos a realizar las siguientes historias de usuario:

+Buscar alumno
+Introducir nuevo alumno
+Modificar alumno
+Eliminar alumno
+Cargar fichero
+Guardar fichero

Roberto se encargará de las siguientes historias:
+Cargar fichero
+Guardar fichero

+Modificar alumno

Daniel se encargará de las siguientes historias:
+Buscar alumno
+Introducir nuevo alumno
+Eliminar alumno

-Sprint Backlog 2

En el primer sprint vamos a realizar las siguientes historias de usuario:

+Iniciar Sesión
+Registrar Profesor
+Cargar Copia de Seguridad
+Hacer Copia de Seguridad
+Imprimir por Pantalla
+Visualizar Grupo

Daniel se encargará de las siguientes historias:
+Iniciar Sesión
+Registrar Profesor
+Visualizar Grupo

Roberto se encargará de las siguientes historias:
+Cargar Copia de Seguridad
+Hacer Copia de Seguridad
+Imprimir por Pantalla

3.2. Matrices de validación

Ahora vamos a realizar las matrices de validación, las cuales relacionan los casos de uso con los requisitos funcionales y los casos de uso con las clases.

Primero vamos a realizar la matriz donde vamos a relacionar los casos de uso con los requisitos.

NA	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11	CU12
RF1	X	X	X	X			X	X				
RF2		X										
RF3			X									
RF4				X								
RF5					X							
RF6						X						
RF7									X			
RF8										X		
RF9								X				
RF10		X										
RF11	X											
RF12		X	X				X					
RF13		X										
RF14											X	
RF15												X

Ahora vamos a realizar la matriz donde vamos a relacionar los casos de uso con las clases.

NA	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11	CU12
Agenda	X	X	X	X	X	X	X	X	X	X		
Profe					X	X			X	X	X	X
Alumno	X	X	X	X			X	X				
Pers.	X	X	X	X			X	X				

4. Bibliografía

http://www.cplusplus.com/

http://c.conclase.net/