

UCO

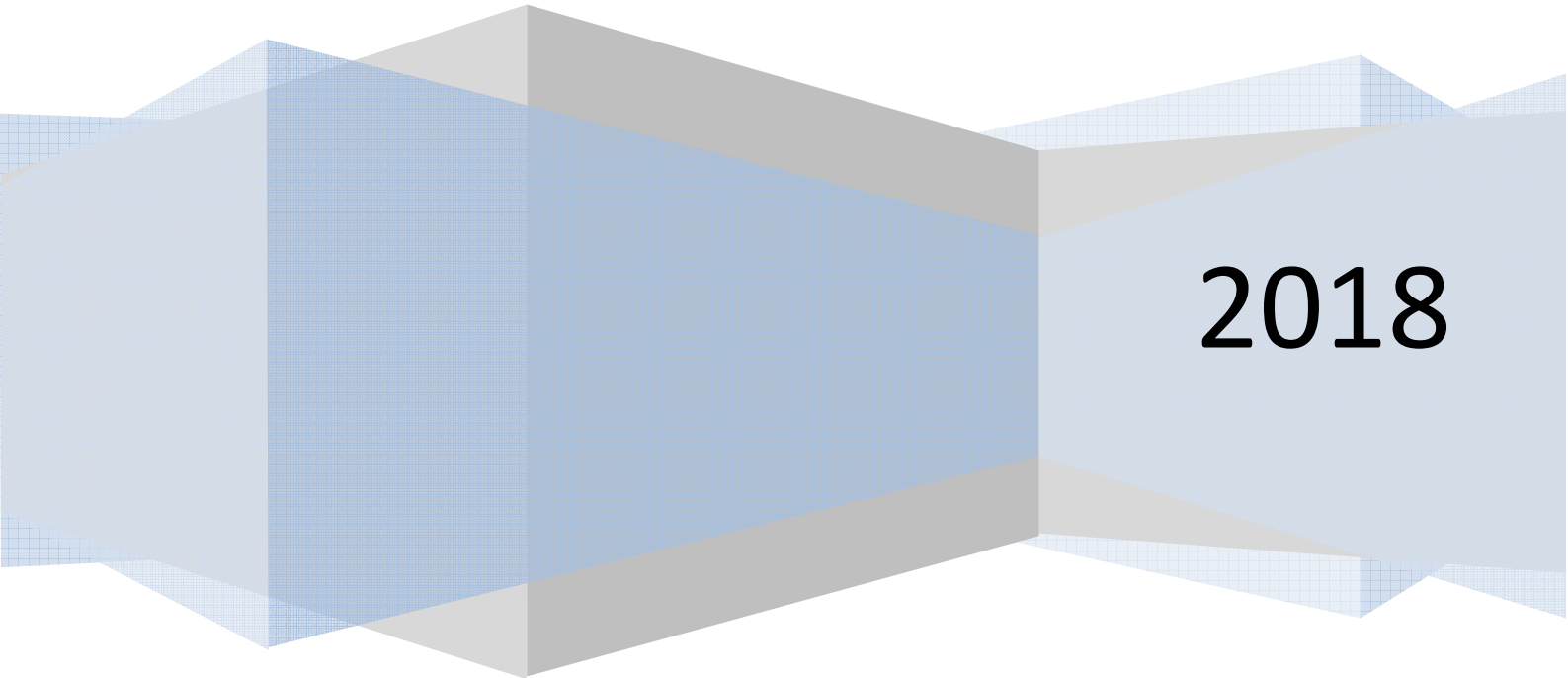
Práctica 4

Ingeniería del Software

Tomas J. Bolaños Campos

Juan A. Serrano Del Río

Lourdes Jiménez Bravo



2018

INDICE

- 1.Introduccion
- 2.Historias de Uso
- 3.Casos de Uso
- 4.Diagrama de Clases
- 5.Diagramas de Secuencia
- 6.Matrices de Validación
- 7.Procedimiento Scrum

INTRODUCCION

A continuación, expondremos la documentación sobre el trabajo final de las prácticas realizadas en Ingeniería del Software.

El objetivo de este trabajo era la realización de un programa para satisfacer las necesidades de nuestro cliente. Dicho cliente nos proporcionó una serie de requerimientos de los cuales sacamos unos requisitos, que expondremos a continuación.

HISTORIAS DE USUARIO

(ANVERSO)

ID:001 Guardar datos

Como usuario quiero guardar la información referente a un alumno en un fichero (DNI, nombre, apellidos, teléfono, correo, domicilio, curso más alto en el que está matriculado, fecha de nacimiento, equipo de clase, líder o no).

Prioridad : 4.

Depende : -

(REVERSO)

- Queremos guardar todos los datos proporcionados del alumno en un fichero binario.

(ANVERSO)

ID:002 Buscar Alumno

Como usuario quiero poder buscar y localizar a un alumno según su dni, apellido o email.

Prioridad : 2.

Depende : -

(REVERSO)

- Queremos localizar a un alumno a través de su dni, apellido o email.
- Se debe tratar el caso de que coincidan los apellidos.

(ANVERSO)

ID:003 Borrar Alumno

Como usuario quiero poder borrar toda la información de un alumno.

Prioridad : 3

Depende : 002

(REVERSO)

- Tengo que borrar todos los datos referente a un alumno.

(ANVERSO)

ID:004 Mostrar todos

Como usuario quiero poder ver la lista de alumnos que hay en la base de datos.

Prioridad : 4

Depende : -

(REVERSO)

- Quiero poder visualizar todos los datos de todos los alumnos.

(ANVERSO)

ID:005 Añadir alumno

Como usuario quiero poder añadir un alumno con sus datos correspondientes.

Prioridad : 2.

Depende : -

(REVERSO)

- Queremos añadir un alumno.
- Tener en cuenta que si un alumno ya existe no podemos volver a añadirlo.
- Tratar el caso de que haya líder o no.
- Se debe tener en cuenta que no puede haber DNIs ni correos iguales.
- Se debe tener en cuenta también los datos que son obligatorios y los que no.

(ANVERSO)

ID:006 Cargar fichero de alumnos

Como usuario quiero poder cargar un fichero binario de alumnos.

Prioridad : 4.

Depende : 005

(REVERSO)

- Leer desde un fichero binario con alumnos y añadirlos al sistema.

(ANVERSO)

ID:007 Modificar alumno

Como usuario quiero ser capaz de modificar la información de un alumno en concreto.

Prioridad : 3

Depende : 002

(REVERSO)

- Modificar la información de un alumno.

(ANVERSO)

ID:008 Mostrar 1 alumno

Como usuario quiero poder ser capaz de ver la información de un alumno.

Prioridad : 4

Depende : 002

(REVERSO)

- Buscar y mostrar un alumno de forma clara.

(ANVERSO)

ID:009 Cargar Copia de Seguridad

Como coordinador quiero poder descomprimir y cargar un fichero binario de alumnos.

Prioridad : 4.

Depende : 005

(REVERSO)

- Descomprimir el fichero con los datos de los alumnos.
- Leer el fichero binario con alumnos y añadirlos al sistema.

(ANVERSO)

ID:010 Guardar Datos Copia Seguridad

Como coordinador quiero guardar la información referente a un alumno en un fichero (DNI, nombre, apellidos, teléfono, correo, domicilio, curso más alto en el que está matriculado, fecha de nacimiento, equipo de clase, líder o no) en una copia de seguridad.

Prioridad : 4.

Depende : -

(REVERSO)

- Queremos guardar todos los datos proporcionados del alumno en un fichero binario.
- Comprimir dicho fichero.

(ANVERSO)

ID:011 Loguear Profesor

Como profesor deseo poder loguearme y así la aplicación sirva para varios profesores.

Prioridad : 4.

Depende : -

(REVERSO)

- Queremos guardar todas credenciales del profesor en un fichero binario.

CASOS DE USO

Guardar datos

ID : 001

Breve descripción : El sistema guarda los datos en un fichero binario.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. Deben existir datos de alumnos que guardar.

Flujo principal :

1. El sistema guarda en un fichero binario los alumnos y sus datos correspondientes.
2. Se le pide al usuario un nombre para el fichero de alumnos.
3. En caso de nombres duplicados en el fichero, se sobrescribe.

Postcondiciones :

1. El fichero queda guardado.

Flujo alternativo :

1. El sistema muestra un mensaje de error si no se han guardado correctamente los datos en el fichero binario.

Buscar Alumno

ID : 002

Breve descripción : El sistema busca un alumno.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. El alumno debe existir en el sistema.

Flujo principal :

1. El caso de uso empieza cuando el sistema necesita buscar un alumno.
2. El sistema busca el alumno según el DNI, apellido o correo especificado por el usuario.

Postcondiciones :

1. No hay.

Flujo alternativo :

1. Si el alumno no existe el sistema muestra un mensaje de error.
2. Si los apellidos de varios alumnos coinciden se muestra un mensaje por pantalla con las coincidencias y se pide una nueva búsqueda por DNI o correo.

Borrar Alumno

ID : 003

Breve descripción : El sistema borra a un alumno.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. El alumno debe existir.

Flujo principal :

1. Borra el alumno según el DNI o apellido especificado por el usuario.

Postcondiciones :

1. El alumno borrado no debe existir.
2. El número total de alumnos debe quedar decrementado en 1.

Flujo alternativo :

1. Si no existe el DNI y apellido del alumno que se desea borrar se muestra un mensaje de error.
2. Si coinciden varios apellidos se busca por DNI.

Mostrar Todos

ID : 004

Breve descripción : El sistema muestra todos los alumnos que hay en la base de datos.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. Deben existir alumnos en el sistema.

Flujo principal :

1. El caso de uso comienza cuando el sistema necesita mostrar todos los alumnos.
2. El sistema recoge los datos de los alumnos.
3. El sistema genera un fichero HTML.

Postcondiciones :

1. EL sistema muestra a los alumnos por pantalla.

Flujo alternativo :

1. Si no existen alumnos, el sistema muestra un mensaje por pantalla.

Añadir Alumno

ID : 005

Breve descripción : El sistema añade un nuevo alumno con sus datos correspondientes al fichero binario.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. El alumno no debe existir.
2. El sistema no puede tener más de 150 alumnos.

Flujo principal :

1. El sistema guarda un nuevo alumno.
2. Se comprueba que los datos obligatorios no estén vacíos.
3. Si añades un alumno a un grupo se comprueba que en el grupo no exista un líder, en caso de existir, dicho alumno no puede ser líder.
4. No se podrá añadir alumno si hay coincidencias de DNI o correo con otro alumno.

Postcondiciones :

1. El número total de alumnos aumenta en 1.

Flujo alternativo :

1. El sistema muestra un mensaje de error si no se puede añadir el alumno.
2. El sistema muestra un mensaje de error si se supera los 150 alumnos.

Cargar Fichero de Alumnos

ID : 006

Breve descripción : El sistema carga un fichero binario con los datos de los alumnos.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. No debe haber alumnos repetidos.
2. El fichero debe existir.

Flujo principal :

1. El caso de uso empieza cuando el sistema necesita cargar el fichero con los datos de los alumnos.
2. El sistema lee los datos de los alumnos recogidos en el fichero binario.
3. El sistema va leyendo un solo alumno y lo inserta en el sistema.

Postcondiciones :

1. El sistema guarda los datos en memoria.

Flujo alternativo :

1. Si no existe el fichero el sistema muestra un mensaje de error.
2. Si existen datos repetidos se muestra un mensaje de error.
3. Si existen alumnos en el sistema, se sobrescriben con los nuevos.

Modificar Alumno

ID : 007

Breve descripción : El sistema modifica la información de un alumno en concreto.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. El alumno a modificar debe existir.

Flujo principal :

1. El sistema te permite modificar los campos un alumno.
2. Se comprueba que los datos obligatorios no quedan vacíos después de modificar y que no haya coincidencia de DNI ni correo con otro alumno.

Postcondiciones :

1. El alumno se guarda con los datos que se modifiquen.

Flujo alternativo :

1. El sistema muestra un mensaje de error si no se ha podido modificar el alumno correctamente.

Mostrar Alumno o Grupo

ID : 008

Breve descripción : El sistema muestra un alumno de la base de datos, o un grupo.

Actores principales : Usuario.

Actores secundarios : Alumno.

Precondiciones :

1. El alumno debe existir.

Flujo principal :

1. El caso de uso comienza cuando el sistema necesita mostrar un alumno.
2. El sistema recoge los datos del alumno.
3. El sistema muestra los alumnos del grupo del alumno mostrado.
4. El sistema muestra los integrantes de un grupo.

Postcondiciones :

1. El sistema muestra el alumno por pantalla.

Flujo alternativo :

1. Si el alumno no existe, el sistema muestra un mensaje de error.
2. Si el grupo no existe, el sistema da un mensaje de error.

Cargar Copia de Seguridad

ID : 009

Breve descripción : El sistema descomprime y carga un fichero binario con los datos de los alumnos.

Actores principales : Administrador.

Actores secundarios : Alumno.

Precondiciones :

1. No debe haber alumnos repetidos.
2. El fichero debe existir.

Flujo principal :

1. El caso de uso empieza cuando el sistema necesita cargar una copia de seguridad del fichero con los datos de los alumnos.
2. El sistema descomprime el fichero.
3. El sistema lee, uno a uno, los datos de los alumnos recogidos en el fichero binario y los inserta en el sistema.

Postcondiciones :

1. El sistema guarda los datos en memoria.

Flujo alternativo :

1. Si no existe el fichero el sistema muestra un mensaje de error.
2. Si existen datos repetidos se muestra un mensaje de error.

Guardar Copia de Seguridad

ID : 010.

Breve descripción : El sistema guarda los datos de los alumnos en un fichero comprimido.

Actores principales : Administrador.

Actores secundarios : Alumno.

Precondiciones :

1. Deben existir datos de alumnos que guardar.

Flujo principal :

1. El sistema guarda en un fichero binario los alumnos y sus datos correspondientes.
2. Se le pide al usuario un nombre para el fichero de alumnos.
3. En caso de nombres duplicados en el fichero, se sobrescriben.
4. El sistema comprime el fichero.

Postcondiciones :

1. El fichero queda guardado y comprimido.

Flujo alternativo :

1. El sistema muestra un mensaje de error si no se han guardado correctamente los datos en el fichero binario.

🔑 Loguear profesor

ID : 011.

Breve descripción : El sistema loguea a uno de los profesores registrados.

Actores principales : Profesor.

Actores secundarios : -

Precondiciones :

1. Deben existir datos del profesor.

Flujo principal :

1. El profesor manda su nombre de usuario
2. El sistema comprueba las credenciales, leyendo un fichero binario.

Postcondiciones :

1. El profesor encontrado en el fichero es el mismo que el dado por el profesor.

Flujo alternativo :

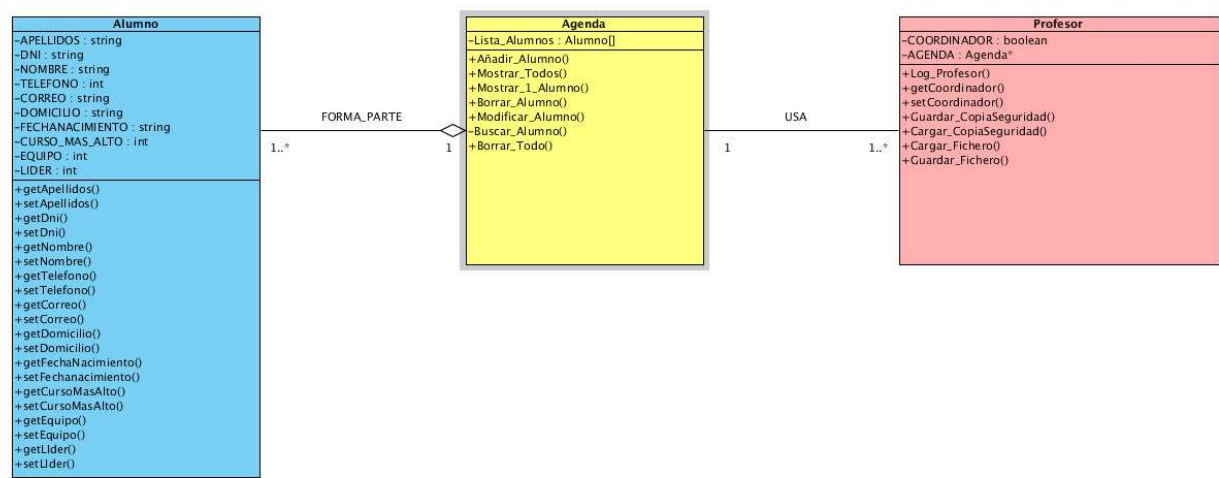
1. El sistema muestra un mensaje de error si no se encuentra al profesor.
2. El sistema da la opción a registrarse si no se encuentra al profesor.

ANALISIS DE REQUISITOS

REQUISITOS

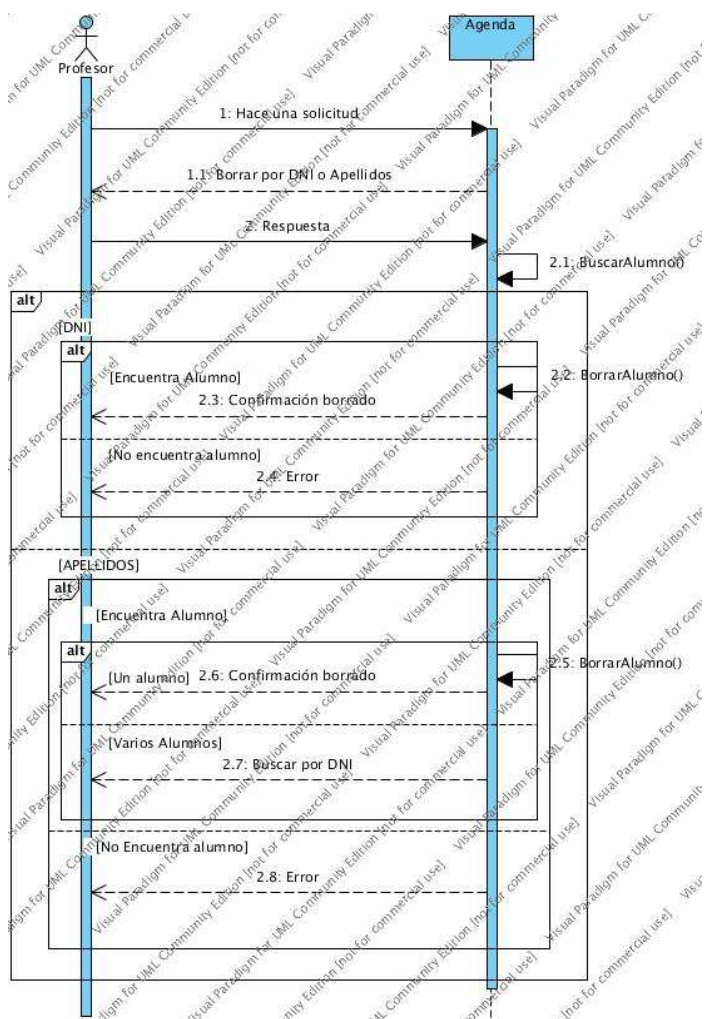
1. Datos a guardar sobre los alumnos --> DNI, nombre, apellidos, teléfono, correo, domicilio, curso más alto, fecha de nacimiento, equipo dentro de la clase y si es líder o no.
2. Mostrar alfabéticamente tanto por nombre como por apellido, dni(sin letra) y curso más alto.
3. Capacidad máxima 150.
4. Borrar alumno por dni o apellido.
5. El programa debe permitir introducir usuarios a mano o cargarlos desde una base de datos.
6. Modificar alumno.
7. Búsqueda de usuario por dni o apellido.
8. Todos los campos son obligatorios menos equipo y si es líder o no.
9. Visualización por línea de comandos y una a elegir entre generar fichero html o por markdown.
10. Interfaz mínima línea de comandos.
11. Tiene que ser capaz de mostrar todos los usuarios como si buscas 1, al mostrar todos, se puede mostrar usuarios dentro de un grupo.
12. Debe funcionar obligatoriamente en linux.
13. Ficheros binarios donde guardes la información de los alumnos.
14. No es obligatorio tener líder, pero si hay líder solo puede haber 1.
15. Mostrar el nombre del líder de forma distinta al resto.
16. Posibilidad de que entren varios profesores y que usen credenciales.
17. Dos roles de profesor, ayudante y coordinador ambos iguales salvo que coordinador puede guardar y cargar copias de seguridad.

DIAGRAMA DE CLASES

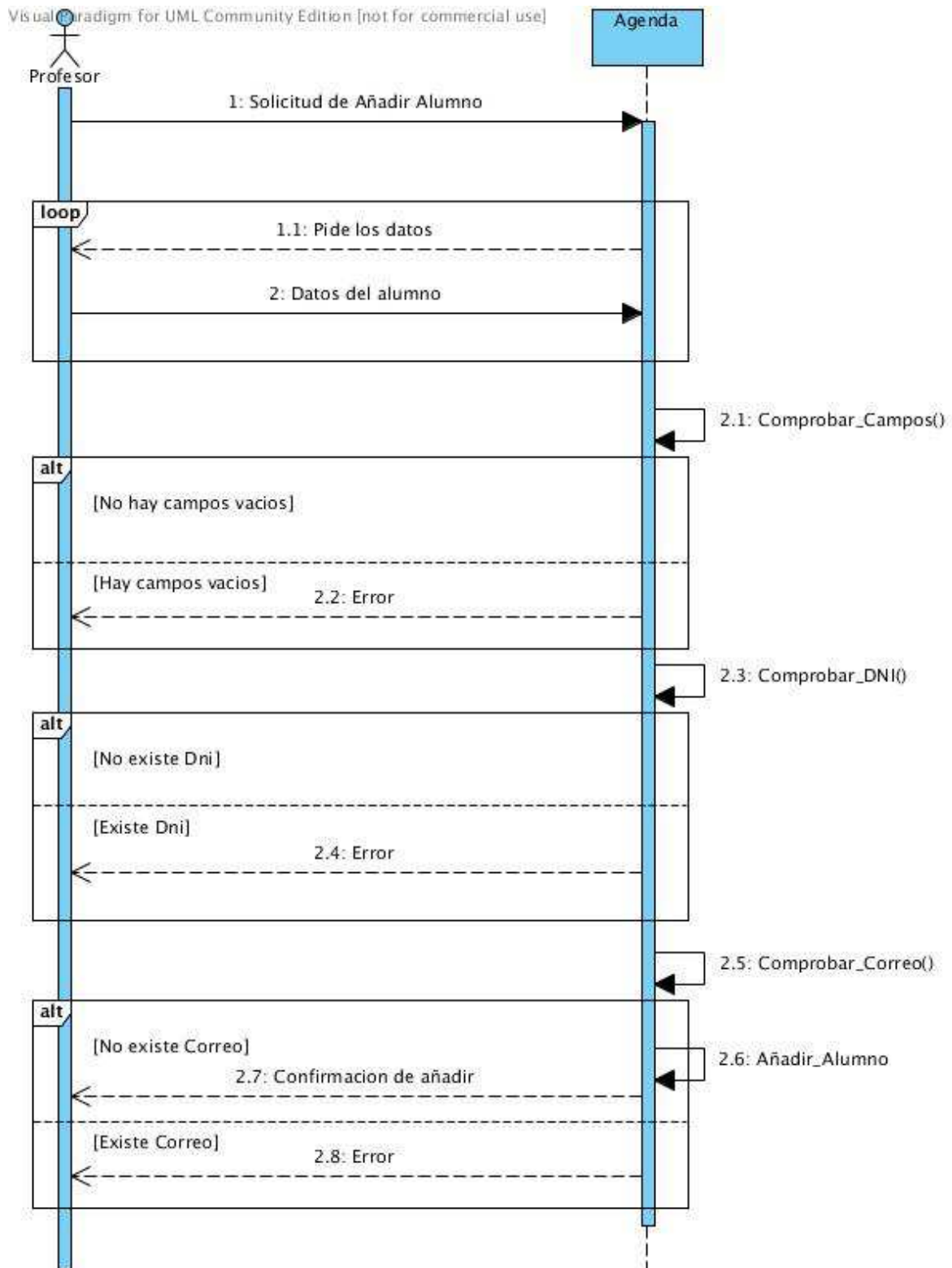


DIAGRAMAS DE SECUENCIA

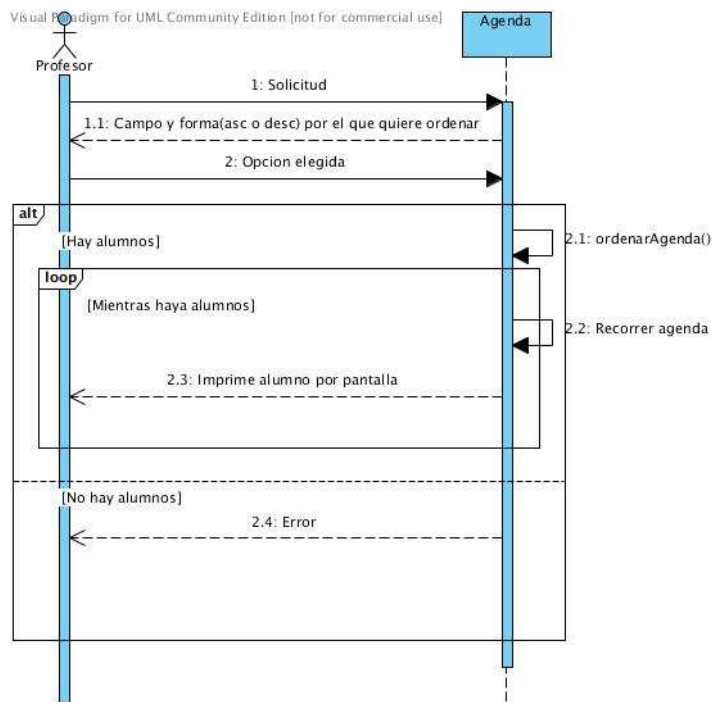
-Función para borrar un alumno.



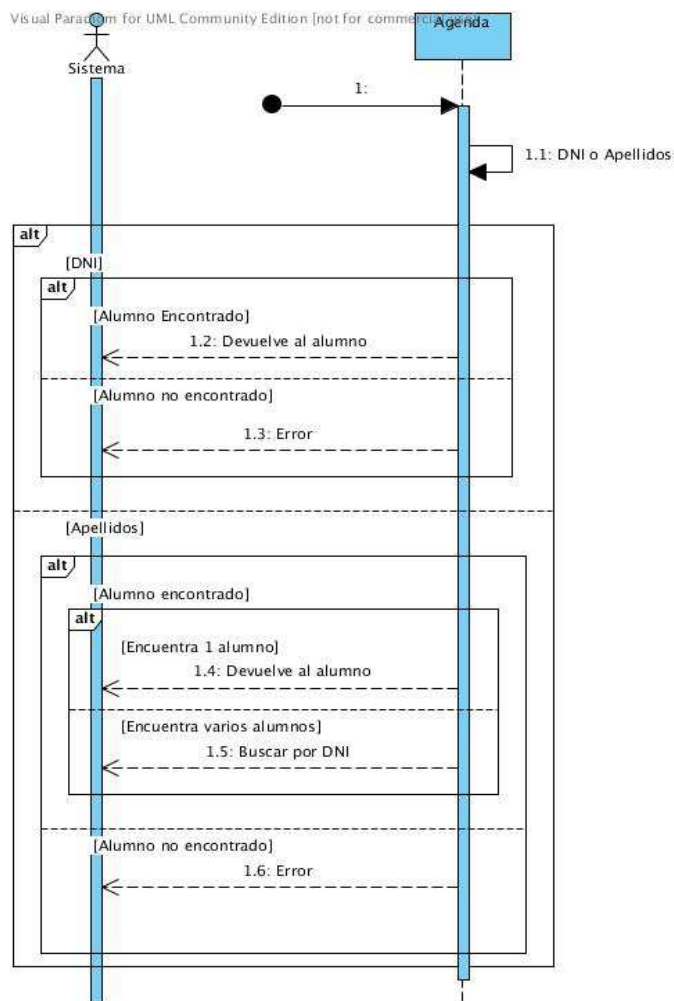
-Función de añadir un alumno.



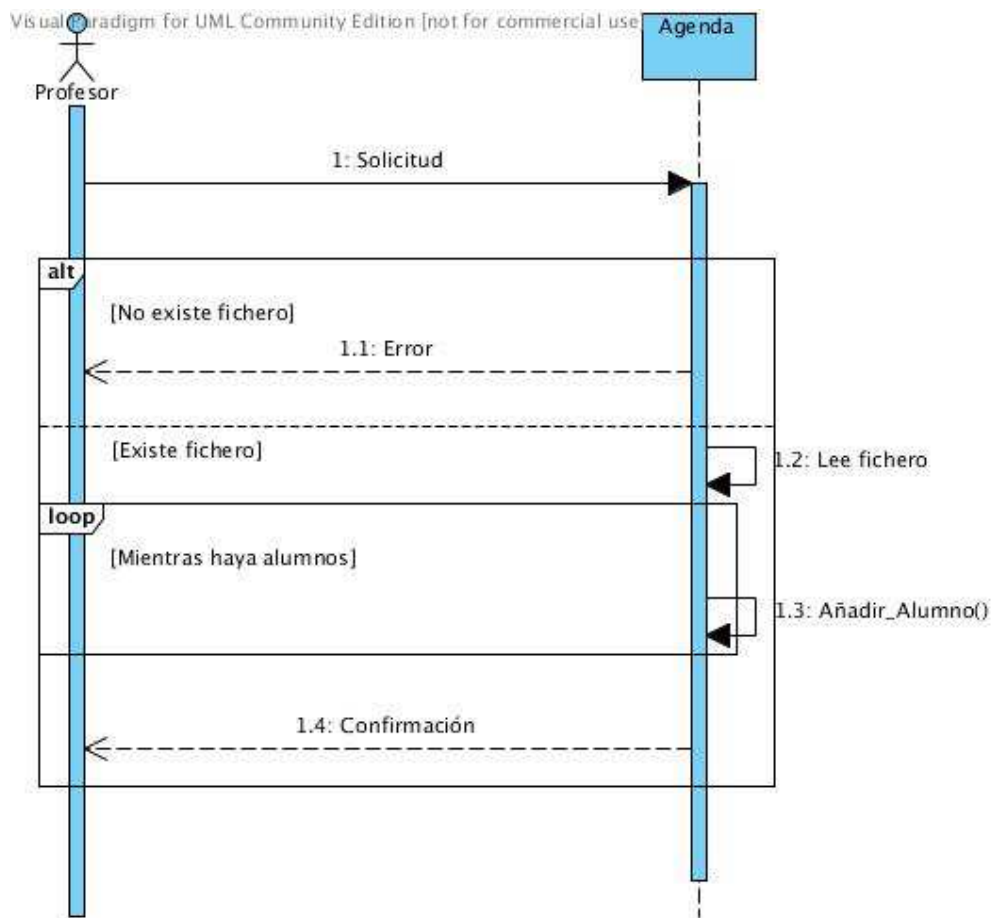
-Función de mostrar todos los alumnos de la agenda.



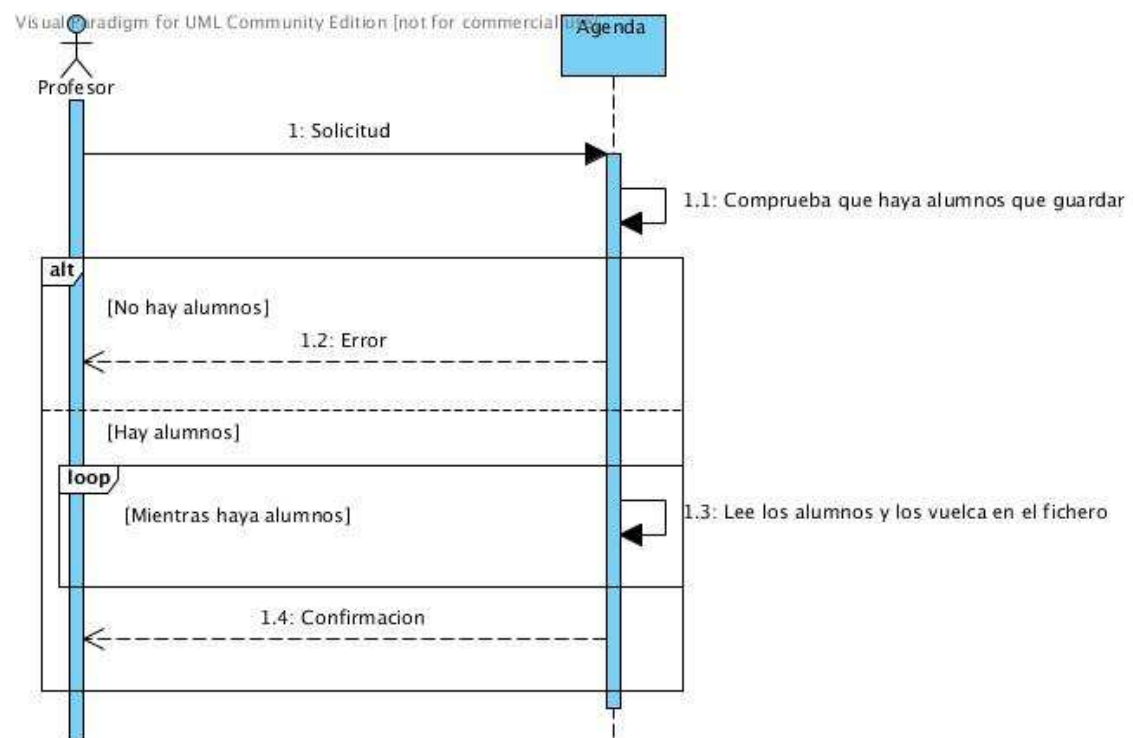
-Función interna para buscar un alumno.



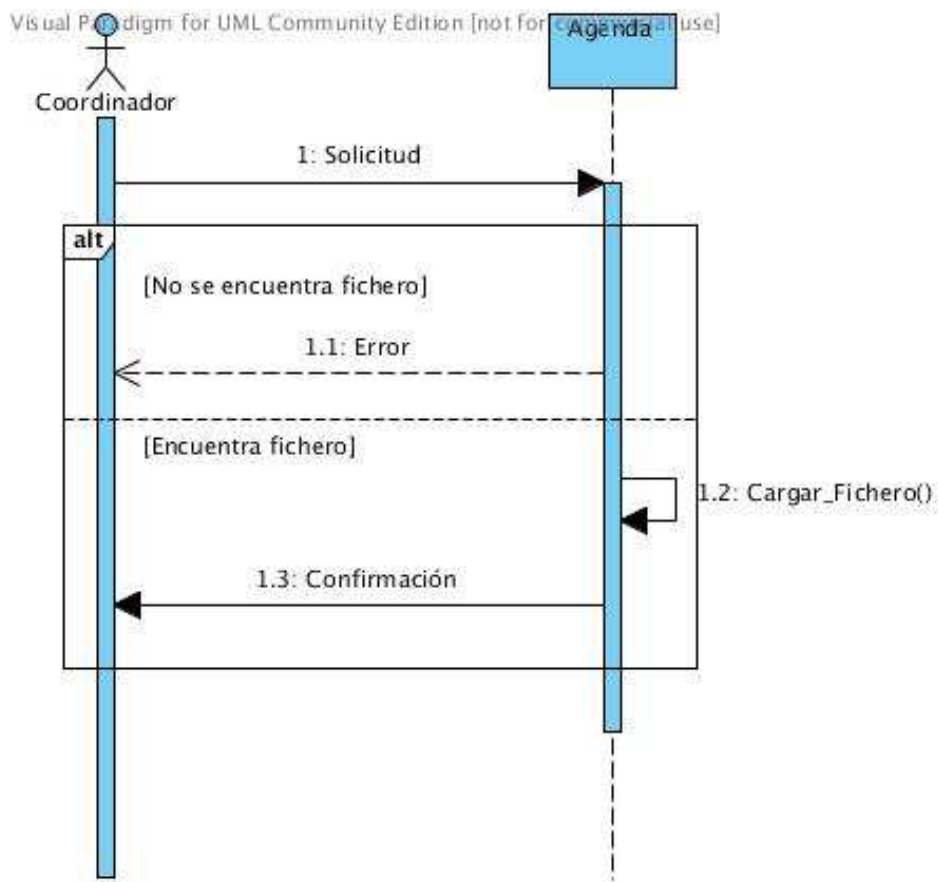
-Función para cargar un fichero de alumnos.



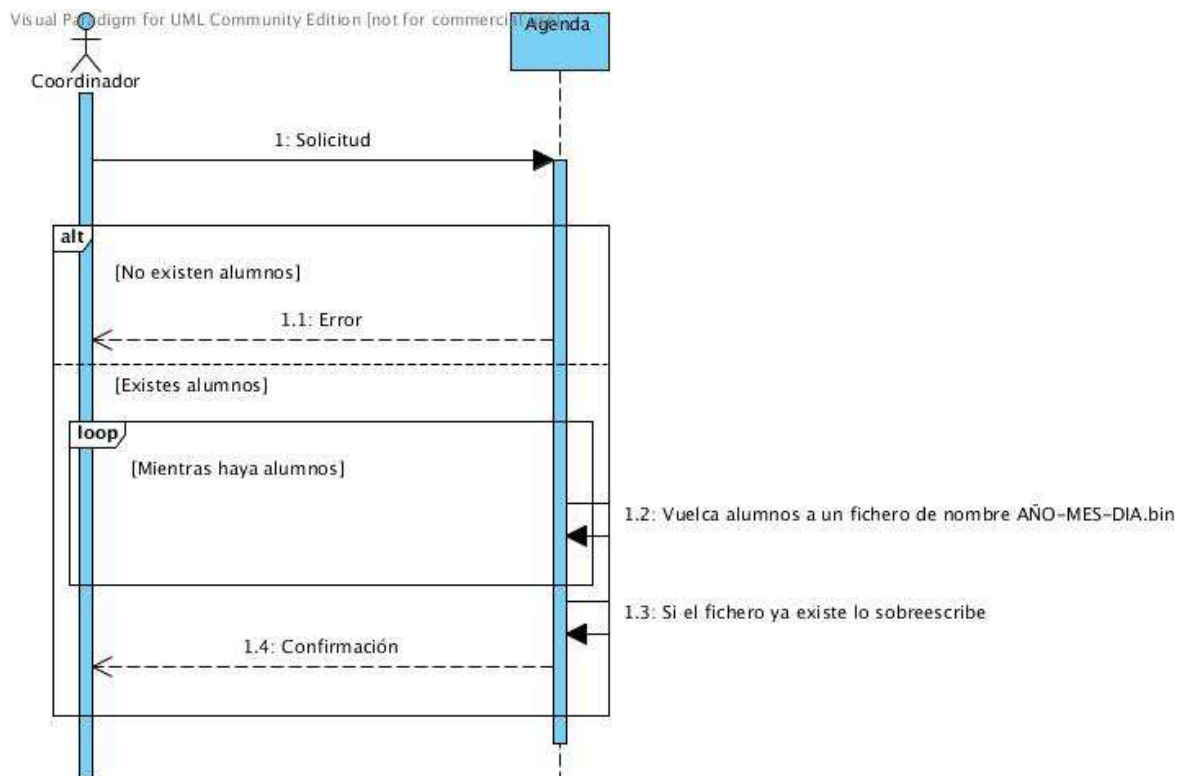
-Función para guardar la agenda en un fichero de alumno.



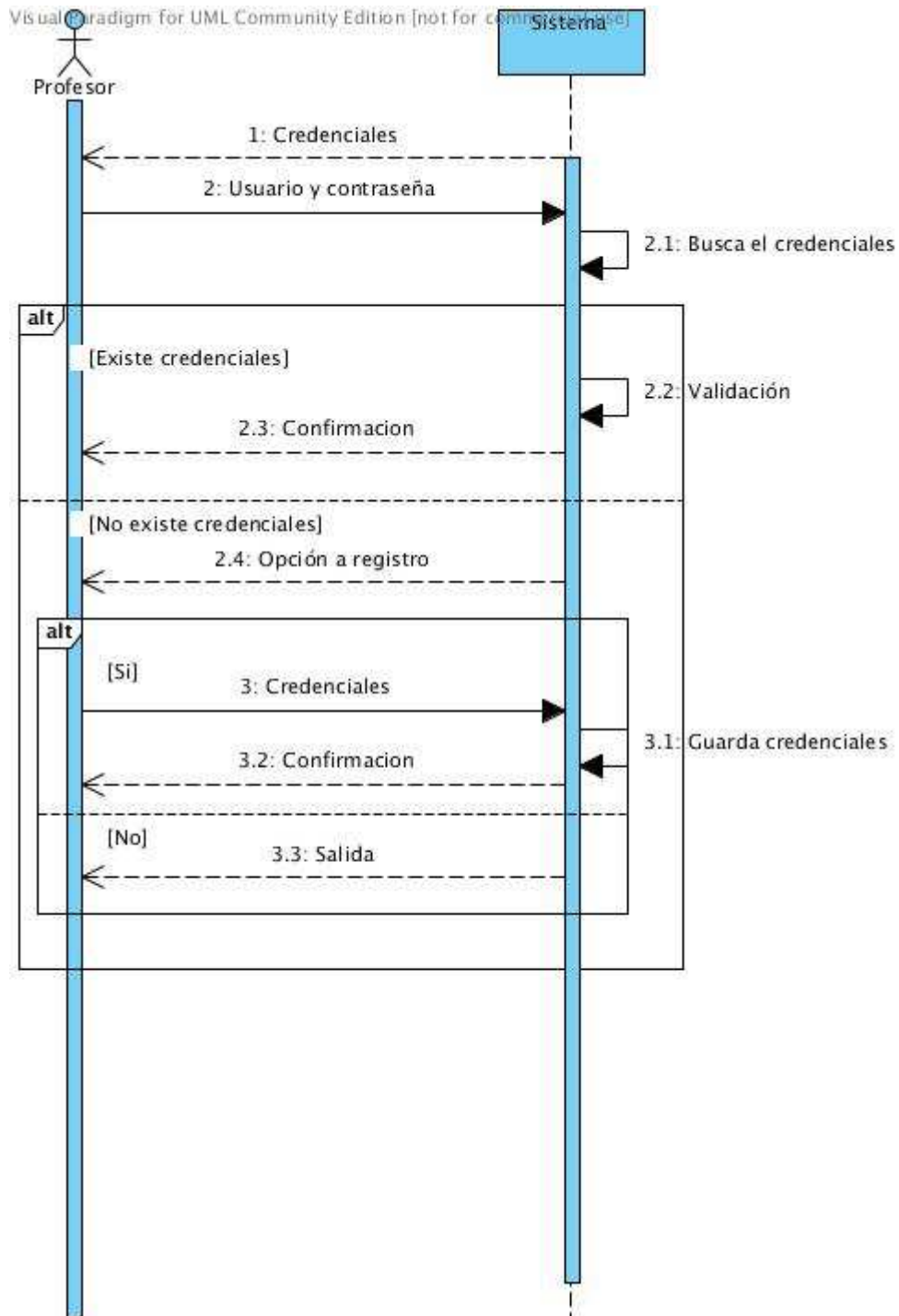
-Función para cargar una copia de seguridad.



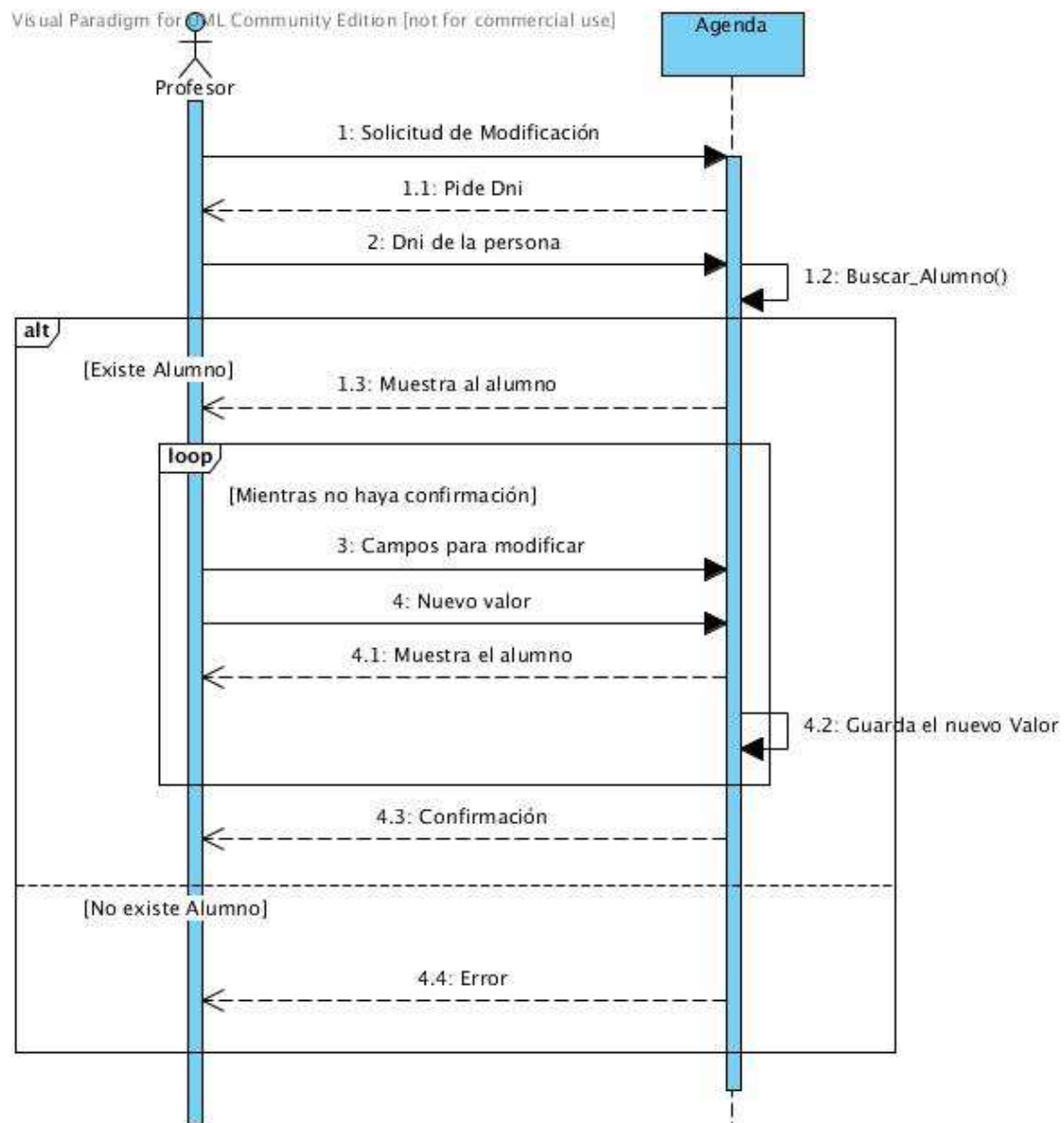
-Función para hacer una copia de seguridad de la agenda.



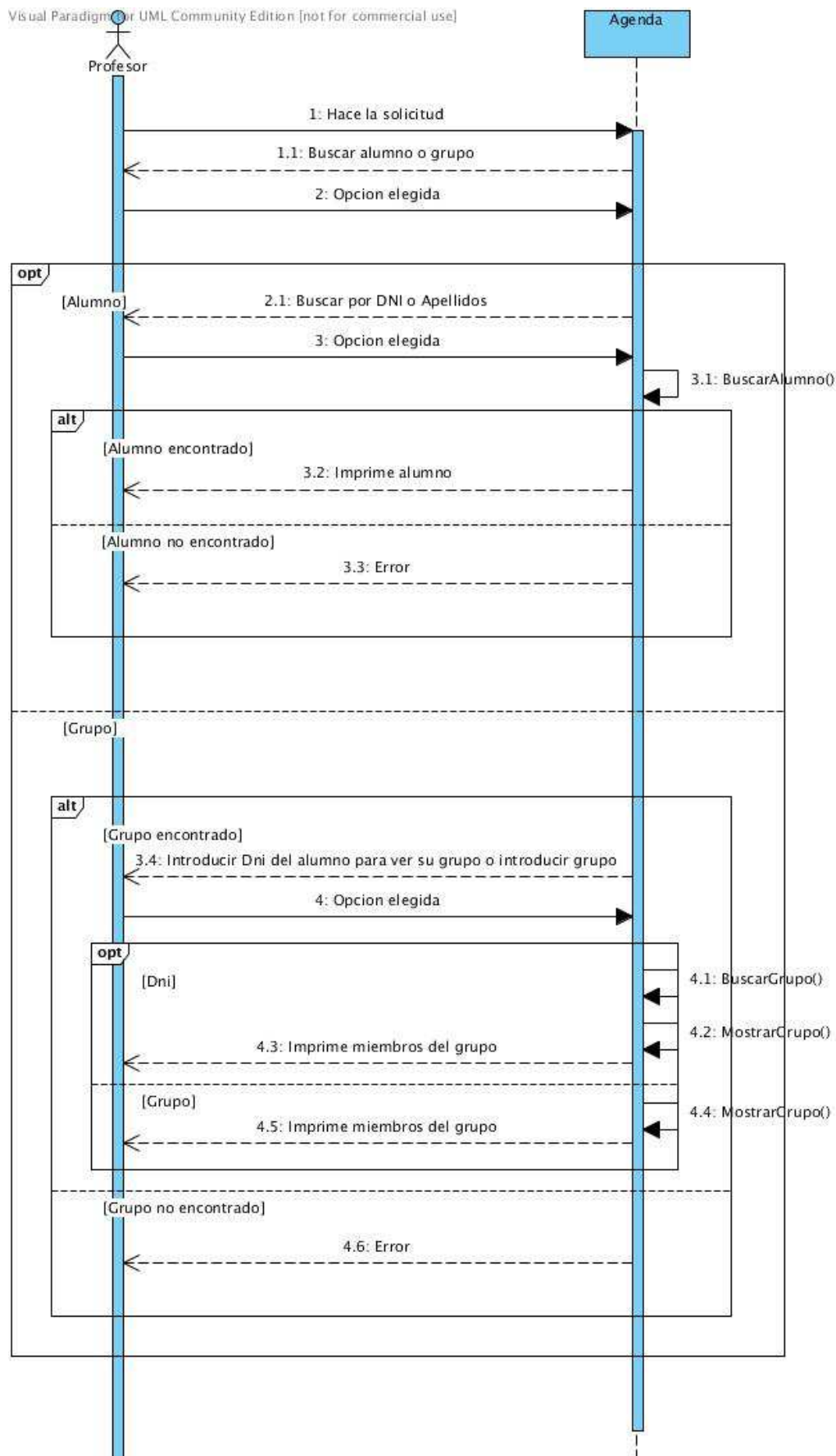
-Función para loguear y registrar al profesorado.



-Función para modificar un alumno de la agenda.



-Función de mostrar a un alumno o a un grupo.



MATRICES DE VALIDACION

·Requisitos/Casos De Uso

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1										X		X	X				
2										X		X					
3				X			X			X		X					
4		X							X	X		X			X		
5	X		X		X			X		X		X		X			
6					X					X		X					
7						X				X		X					
8							X			X	X	X			X		
9										X		X				X	
10										X		X					X
11										X		X					X

·Clases/Casos De Uso

	1	2	3	4	5	6	7	8	9	10	11
Alumno					X						
Agenda		X	X	X	X		X	X			
Profesor	X					X			X	X	X

PROCEDIMIENTO SCRUM

PRODUCT BACKLOG

1. Creacion de clases Prioridad: MAX N_horas: 5 1.1 BuscarAlumno ID:002 Prioridad: 0 N_horas: 2
2. AñadirAlumno ID:005 Prioridad: 1 N_horas: 3
3. MostrarTodos ID:004 Prioridad: 2 N_horas: 2
4. ModificarAlumno ID:007 Prioridad: 2 N_horas: 2
5. BorrarAlumno ID:003 Prioridad: 2 N_horas: 2
6. Mostrar1Alumno ID:008 Prioridad: 2 N_horas: 4
7. CargarFichero ID:006 Prioridad: 3 N_horas: 2
8. GuardarFichero ID:001 Prioridad: 3 N_horas: 2
9. LoguearProfesor ID:011 Prioridad: 3 N_horas: 3
10. CargarCopiaSeguridad ID:009 Prioridad: 4 N_horas: 4
11. GuardarCopiaSeguridad ID:010 Prioridad: 4 N_horas: 4

SPRINT BACKLOG

Tomas Va a realizar BuscarAlumno() *Numero total de horas: 7*

Juan Antonio Va a realizar MostrarTodos() y Mostrar1Alumno() *Numero total de horas: 6*

Lourdes Va a realizar AñadirAlumno() y ModificarAlumno() *Numero total de horas: 5*

SPRINT BACKLOG 2

Tomas Va a realizar BorrarAlumno() y GuardarCopiaSeguridad() *Numero total de horas: 6*

Juan Antonio Va a realizar CargarFichero() y CargarCopiaSeguridad() *Numero total de horas: 6*

Lourdes Va a realizar LoguearProfe() y GuardarFichero() *Numero total de horas: 5*

BURNDOWN CHART

