UNIVERSIDAD DE CÓRDOBA

Ingeniería del Software, Conocimiento y Bases de Datos

GRADO DE INGENIERÍA INFORMÁTICA

INGENIERÍA DEL SOFTWARE

ENTREGA FINAL DE PRÁCTICAS

En este documento se va a redactar y documentar, de forma formal y profesional, todo el proceso de análisis, diseño y desarrollo del producto de programación que hemos ido trabajando durante todo el cuatrimestre.

Autor/es: Mérida Palacios, Felipe
Mifsut Castilla, Ricardo
Santos Ramírez, Ángel Fco.

Fecha: **23/12/2018**

ÍNDICE

Práctica 4:

1 DEFINICIÓN DEL PROBLEMA

El profesorado de la asignatura de Ingeniería del Software quiere informatizar los datos de contacto de los alumnos en un programa informático que guarde esta información.

El profesorado vino diciendo de incorporar nuevos elementos a nuestro problema:

- Quiero considerar la posibilidad de que entren diferentes profesores (y que usen credenciales – fichero binario).
- Me gustaría crear dos roles de profesor: coordinador y ayudantes.
- Las tareas son las mismas para ambos roles, a excepción del trabajo con las copias de seguridad, que únicamente las haría el coordinador.

2 Extracción de requisitos

Primeramente, se va a explicar la extracción de requisitos que se realiza en su documento correspondiente. Para llevarlo a cabo, este documento consiste en desarrollar la extracción de los requisitos de un producto software como la primera etapa para crearlo.

En él aparecerán las partes interesadas, los datos que debe almacenar la aplicación, los requisitos funcionales y no funcionales, y finalmente, la priorización de los mismos.

A continuación se muestra y se desarrolla el documento de Extracción de Requisitos:

Documento de Extracción de Requisitos

Partes Interesadas:

Las partes interesadas de nuestro sistema serán los profesores coordinadores y ayudantes que utilicen el programa para el almacenamiento de datos de los alumnos.

Datos a almacenar:

Archivo de Credenciales:

Los datos que introduce el profesor para registrarse/identificarse en/con el archivo de credenciales son los siguientes:

1. DNI.

Base de Datos de los Alumnos:

Los datos que introduce el profesor, que se guardan en un fichero binario, que hará de base de datos de los alumnos, serán los siguientes:

1. DNI.
2. Nombre.
3. Apellidos.
4. Teléfono.
5. Email.
6. Dirección postal.
7. Curso más alto matriculado.
8. Fecha de nacimiento.
9. Número de grupo.
10. Si es líder de su grupo o no.

Requisitos funcionales:

- Identificar profesor.
- Insertar alumno.
- Buscar alumno.
- Mostrar alumno.
- Cargar datos de un fichero binario preestablecido.
- Cargar datos de una copia de seguridad binaria.
- Guardar datos en un fichero binario preestablecido.
- Guardar datos en una copia de seguridad binaria.
- Eliminar alumno.
- Modificar alumno.
- Eliminar a todos los alumnos de la base de datos.

Requisitos no funcionales:

- Lenguaje de programación: C++.
- Documentación en Markdown.
- Uso de Commits.
- Se utilizará Git y como repositorio remoto GitHub.
- La aplicación se utilizará en Linux.
- Se podrán almacenar como máximo 150 alumnos.
- Solamente habrá 1 líder por grupo.
- Se podrá almacenar en el archivo de credenciales como máximo un profesor coordinador y 5 profesores ayudantes.
- Como criterio de ordenación y búsqueda se utilizará DNI, Apellido, Curso más alto en el que el alumno está matriculado.
- Se ordenará alfabéticamente de forma ascendente o descendente.
- Se permite borrar y asignar al líder de un grupo.

Priorización de Requisitos:

N.º de Prioridad: Requisito

Prioridad 1: Identificar profesor.

Prioridad 1: Insertar alumno.

Prioridad 2: Cargar fichero binario preestablecido.

Prioridad 2: Cargar copia de seguridad de un fichero binario.

Prioridad 2: Guardar copia de seguridad en un fichero binario.

Prioridad 2: Cargar en fichero binario preestablecido.

Prioridad 3: Buscar alumno.

Prioridad 4: Mostrar alumno.

Prioridad 4: Eliminar alumno.

Prioridad 4: Modificar alumno.

Prioridad 5: Eliminar base de datos de los alumnos.

3 HISTORIAS DE USUARIO

Las historias de usuario consisten en descripciones, siempre muy cortas y esquemáticas, que resumen la necesidad concreta de un usuario al utilizar un producto o servicio, así como la solución que la satisface.

Su función principal es identificar problemas percibidos, proponer soluciones y estimar el esfuerzo que requieren implementar las ideas propuestas.

A continuación se desarrollan todas las historias de usuario de los distintos requisitos funcionales:

3.1. H.U. de Insertar Alumno

(ANVERSO)

ID: 001 Insertar Alumno

Como usuario quiero poder insertar los datos de un alumno en la base de datos. .

Prioridad: 1

(REVERSO)

- Quiero poder insertar todos los datos de los alumnos.
- Si el DNI introducido ya existe, quiero poder volver a introducirlo o salir y no insertar al alumno.
- Si el correo introducido ya existe, quiero poder volver a introducirlo o salir y no insertar al alumno.

3.2. H.U. de Modificar Alumno

(ANVERSO)

ID: 002 Modificar alumnos

Como usuario quiero poder modificar los datos de un alumno.

Prioridad: 4

(REVERSO)

- Quiero poder modificar los datos del alumno.
- A la hora de asignar el líder del grupo, solamente puede haber un líder por grupo.

3.3. H.U. de Buscar Alumno

(ANVERSO)

ID: 003 Buscar Alumno

Quiero que el sistema pueda buscar un alumno de la base de datos.

Prioridad: 3

(REVERSO)

- La búsqueda se realizará por apellidos y si hay dos iguales se realizará la búsqueda por DNI.

3.4. H.U. de Mostrar Alumno

(ANVERSO)

ID: 004 Mostrar Alumnos

Como usuario quiero poder ver los datos de los alumnos.

Prioridad: 4

(REVERSO)

- Quiero mostrar todos los datos de los alumnos (DNI, Nombre, Apellidos, teléfono, correo UCO, Dirección postal, Curso más alto que estás matriculado, fecha de nacimiento, número del equipo del alumno y si es líder o no lo es).
- Los alumnos serán mostrados en orden alfabético por apellidos o por el curso más alto, de ambas formas ascendente y descendentemente.
- Quiero mostrar a todos los alumnos o un alumno en concreto.
- En el caso de un alumno en concreto quiero que me pregunte por DNI o Apellidos.
- Si hay varios alumnos con el apellido que he introducido quiero que se muestre un mensaje de error.

3.5. H.U. de Borrar Alumno

(ANVERSO)

ID: 005 Borrar Alumno

Como usuario quiero poder borrar los datos de un alumno de la base de datos.

Prioridad: 4

(REVERSO)

- Quiero poder borrar todos los datos de un alumno.
- Quiero poder buscar el alumno a eliminar por los apellidos.
- Si existen varios alumnos con los mismos apellidos, quiero poder buscar por el DNI el alumno a borrar.

3.6. H.U. de Eliminar la base de datos de los alumnos

(ANVERSO)

ID: 006 Eliminar la base de datos de los alumnos

Como usuario quiero poder borrar a todos los alumnos de la base de datos.

Quiero que el sistema me pregunte que si estoy seguro de hacerlo.

Prioridad: 5

(REVERSO)

- Quiero poder borrar a todos los alumnos de la base de datos.

3.7. H.U. de Cargar datos de un fichero binario preestablecido

(ANVERSO)

ID: 007 Cargar datos de un fichero binario preestablecido

Como usuario quiero poder cargar datos de los alumnos de un fichero binario preestablecido actuando como una base de datos del mismo.

Prioridad: 2

(REVERSO)

- Quiero poder cargar los datos de los alumnos de un fichero binario.
- En el fichero binario sólo podrán existir como máximo 150 alumnos.
- Quiero que me muestre un mensaje de error si el archivo binario no existe.

3.8. H.U. de Guardar datos en un fichero binario preestablecido

(ANVERSO)

ID: 008 Guardar datos en un fichero binario preestablecido

Como usuario quiero poder hacer una copia de seguridad de los datos de los alumnos en un fichero binario preestablecido para así dejarlos registrados en el mismo.

Prioridad: 2

(REVERSO)

- Quiero poder hacer una copia de seguridad de los datos de los alumnos en un fichero binario.

3.9. H.U. de Identificar al profesor como coordinador o ayudante al iniciar

(ANVERSO)

ID: 009 Identificar al profesor como coordinador o ayudante al iniciar.

Como usuario quiero poder identificarme como profesor coordinador o ayudante.

Al identificarme como Coordinador, quiero poder cambiar mi DNI, eliminar un profesor coordinador y ver todos los profesores registrados y tener disponible todas las funcionalidades.

Al identificarme como Ayudante, quiero tener disponible todas las funcionalidades excepto 2: cargar y guardar copia de seguridad binaria.

Prioridad: 1

(REVERSO)

- Quiero poder identificarme como profesor coordinador.
- Quiero poder identificarme o registrarme como profesor ayudante.
- Para identificarme\registrarme quiero que me pregunte mi DNI.

3.10. H.U. de Cargar datos de una copia de seguridad binaria

(ANVERSO)

ID: 010 Cargar datos de una copia de seguridad binaria

Como usuario quiero poder cargar datos de los alumnos de una copia de seguridad binaria actuando como una base de datos del mismo.

Prioridad: 2

(REVERSO)

- Quiero poder cargar los datos de los alumnos de un fichero binario.
- En el fichero binario sólo podrán existir como máximo 150 alumnos.
- Quiero que me muestre un mensaje de error si el archivo binario no existe.

3.11. H.U. de Guardar datos en una copia de seguridad binaria

(ANVERSO)

ID: 011 Guardar datos en una copia de seguridad binaria

Como usuario quiero poder hacer una copia de seguridad de los datos de los alumnos en un fichero binario para así dejarlos registrados en el mismo.

Prioridad: 2

(REVERSO)

- Quiero poder hacer una copia de seguridad de los datos de los alumnos en un fichero binario.
- Quiero poder introducir el nombre del fichero binario a crear/sobrescribir.
- Si el nombre introducido no es válido, quiero que me permita volver a introducir el nombre del fichero binario a crear/sobrescribir.

4 Casos de uso

Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Es una herramienta valiosa dado que es una técnica de aciertos y errores para obtener los requerimientos del sistema, justamente desde el punto de vista del usuario.

A continuación se desarrollan todos los casos de uso de los distintos requisitos funcionales:

4.1. C.U. de Insertar Alumno

Insertar Alumno

ID: 001

Breve Descripción: El profesor introducirá, en la base de datos, los datos de un alumno.

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. El alumno no debe existir.
2. La base de datos no debe tener 150 alumnos.

Flujo principal:

1. El caso de uso empieza cuando el profesor quiere introducir un alumno a la base de datos.
2. El sistema pedirá los datos del alumno al profesor.
3. El sistema mostrará un mensaje diciendo que el alumno ha sido introducido correctamente.

Postcondiciones:

- Si no existe el alumno se inserta en la base de datos.

Flujos alternativos:

1.a. Si la base de datos tiene 150 alumnos, se mostrará un mensaje diciendo que la base de datos está llena.

2.a. Si el DNI o el e-mail introducidos son iguales a otro ya en la base de datos, se da la opción de volver a introducir el dato o de parar de introducir el alumno y no introducirlo.

2.a.a. Si ha habido un dato no válido (Ej.: Fecha de nacimiento: 34/12/1997), se volverá a pedir el dato no válido.

3.a. En caso de haber un error, se mostrará un mensaje diciendo que no se ha introducido correctamente.

4.2. C.U. de Modificar Alumno

Modificar alumno

ID: 002

Breve descripción: El sistema permite la modificación de los datos de un alumno.

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. El alumno debe existir en la base de datos.

Flujo principal:

1. El caso de uso empieza cuando el profesor quiere modificar los datos de un alumno.
2. El sistema pedirá que el profesor introduzca el DNI del alumno a modificar.
3. Una vez se encuentre al alumno a modificar el sistema mostrará un mensaje que dice que seleccione un atributo a modificar o seleccione la opción salir, y mostrará un menú con todos los atributos del alumno y la opción salir.
4. Al seleccionar la opción de salir, el sistema se sale y no modifica nada.

Postcondiciones:

- El sistema machaca los datos a modificar por los datos nuevos.

Flujos alternativos:

2.a. Si el alumno no existe, el sistema muestra un mensaje de error.

4.a. Al seleccionar la opción de cualquier atributo, el sistema pedirá el valor nuevo del atributo seleccionado.

4.b. El valor del alumno será modificado en la base de datos.

4.c. El flujo vuelve al punto 3 del flujo principal.

4.a.a. Al no seleccionar ninguna opción, el sistema muestra un mensaje de error.

4.a.b. El flujo vuelve al punto 3 del flujo principal.

4.b.a. Si el valor pasado no es compatible (Ej.: Dia de Nacimiento del Alumno: 34 de Diciembre), el sistema vuelve a pedir el valor hasta que sea compatible.

4.b.b. El flujo vuelve al punto 4.b. del flujo alternativo.

4.3. C.U. de Buscar Alumno

Buscar Alumno

ID: 003

Breve descripción: El sistema buscará un alumno en la base de datos.

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. El alumno debe existir

Flujo principal:

1. El caso de uso empieza cuando el sistema quiere buscar un alumno de la base de datos.
2. El sistema pide al profesor introducir los apellidos del alumno a buscar.
3. El sistema busca a ese alumno.
4. Si el alumno se encuentra, se muestra su nombre y apellidos por pantalla.

Postcondiciones:

- Si se encuentra, se muestra su nombre y apellidos por pantalla.

Flujos alternativos:

2.a. Si al introducir los apellidos hay varios alumnos con los mismos se requerirá buscar por DNI.

2.a.a. Si el DNI introducido no está en la base de datos se muestra un mensaje de error.

4.a. Si no existe el alumno se muestra un mensaje de error.

4.4. C.U. de Mostrar Alumno

Mostrar alumno

ID: 004

Breve descripción: El sistema muestra los datos de un alumno, de todos los alumnos o de los alumnos de un grupo.

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. El alumno a buscar, en el caso de elegir un solo alumno debe estar en la base de datos.

Flujo principal:

1. El caso de uso empieza cuando el profesor quiera ver los datos de un alumno o de todos los alumnos.
2. El sistema mostrará un menú en el que muestra las opciones de muestra mostrar datos de un alumno o todos los alumnos.
3. En el caso de elegir mostrar todos los alumnos, el sistema muestra otro menú en el que el profesor selecciona si se muestra por curso más alto matriculado o por apellido y si se muestra por ordenación ascendente o descendente.
4. El sistema muestra los datos de los alumnos de la forma elegida.

Postcondiciones:

- Si no hay ningún alumno, no mostrará nada.

Flujos alternativos:

2.a. Si el profesor selecciona mostrar datos de un alumno, el sistema pedirá buscar por el apellido o por el DNI del alumno.

2.b. Si ha seleccionado la opcion por apellido, el sistema pedirá al profesor el apellido del alumno.

2.c. Si el alumno ha sido encontrado, se muestran sus datos por pantalla.

2.c.1.a. Si existen varios alumnos con el mismo apellido se muestra un mensaje de error.

2.c.2.a. Si no existe ningún alumno con el apellido introducido, se muestra un mensaje de error.

2.b.a. Si ha seleccionado la opción por DNI, el sistema pedirá al profesor el DNI del alumno.

2.b.b. Si el alumno ha sido encontrado, se muestran sus datos por pantalla.

2.b.b.a. Si no existe ningún alumno con el DNI introducido, se muestra un mensaje de error.

4.5. C.U. de Borrar Alumno

Borrar Alumno

ID: 005

Breve Descripción: El sistema borra un alumno de la base de datos.

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. El alumno debe existir.

Flujo principal:

1. El caso de uso empieza cuando el profesor quiere borrar los datos de un alumno de la base de datos.
2. El profesor introduce los apellidos del alumno a borrar.
3. El sistema busca a ese alumno.
4. El alumno encontrado es borrado del sistema.

Postcondiciones:

- Si se encuentra al alumno, lo borra de la base de datos.

Flujos alternativos:

2.a. Si al introducir los apellidos hay varios alumnos con los mismos apellidos, se requerirá buscar por DNI.

2.b. El flujo continuará por el punto 3 del flujo principal.

3.a. Si no existe el alumno se muestra un mensaje de error diciendo que el alumno no existe.

4.6. C.U. de Borrar a todos los alumnos de la base de datos

Borrar a todos los alumnos de la base de datos

ID: 006

Breve descripción: El sistema permite borrar a todos los alumnos de la base de datos.

Actores principales: Profesor Coordinador.

Actores secundarios: Alumnos.

Precondiciones:

1. Debe de haber alumnos en la base de datos.

Flujo principal:

1. El caso de uso empieza cuando el profesor quiera borrar a todos los alumnos de la base de datos.
2. El sistema muestra un menú con dos opciones (sí o no) y me pregunta si estoy seguro de ello.
3. Al seleccionar la opción de sí, el sistema borra a todos los alumnos de la base de datos.

Postcondiciones:

- La base de datos de los alumnos queda vacía.

Flujos alternativos:

3.a. Al seleccionar la opción de sí y al no existir alumnos en la base de datos, el sistema mostrará un mensaje diciendo que la base de alumnos está vacía.

3.1.a. Al seleccionar la opción de no, el sistema no hace nada.

3.2.a. Al no seleccionar ninguna opción el sistema muestra un mensaje de error diciendo que elijamos una opción válida.

3.2.b. El flujo vuelve al punto 2 del flujo principal.

4.7. C.U. de Cargar datos de un fichero binario preestablecido

Cargar datos de un fichero binario preestablecido

ID: 007

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

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. El fichero a cargar tiene que ser binario.
2. El fichero binario puede tener como máximo 150 alumnos.
3. El fichero binario a cargar debe existir en el sistema.

Flujo principal:

1. El caso de uso empieza cuando el profesor necesita cargar los datos de los alumnos de un fichero binario.
2. El sistema carga los datos del fichero binario preestablecido en el sistema.

Postcondiciones:

- Una vez cargado el fichero, el sistema tiene la base de datos de los alumnos del fichero binario cargado.

Flujos alternativos:

2.a. Si el fichero binario preestablecido a cargar no existe, muestra un mensaje de error.

3.a. Si el fichero binario tiene más de 150 alumnos, muestra un mensaje de error.

4.8. C.U. de Guardar datos en un fichero binario preestablecido

Guardar datos en un fichero binario preestablecido

ID: 008

Breve descripción: El sistema guarda la base de datos de los alumnos en un fichero binario preestablecido.

Actores principales: Profesor Coordinador o Ayudante.

Actores secundarios: Alumnos.

Precondiciones:

1. Si la base de datos está vacía, el archivo binario donde se guardarán los datos quedará vacío.

Flujo principal:

1. El caso de uso empieza cuando el profesor necesita guardar los datos de los alumnos en un fichero binario.
2. El sistema crea/sobrescribe el fichero preestablecido y guarda la base de datos de los alumnos.

Postcondiciones:

- El fichero binario tiene todos los datos de los alumnos de la base de datos en ese instante al ser guardado.

Flujos alternativos:

2.a. Si hay un error a la hora de crearse/sobrescribirse el fichero, se muestra un mensaje de error.

4.9. C.U. de Identificar Profesor Coordinador o Ayudante

Identificar Profesor Coordinador o Ayudante

ID: 009

Breve descripción: El sistema identifica como coordinador o ayudante al profesor gracias a unas credenciales de un fichero binario.

Actores principales: Usuario No Identificado.

Actores secundarios: No existen actores secundarios en este caso.

Precondiciones:

1. El archivo de credenciales tiene que existir.
2. Como máximo, en el archivo de credenciales solo habrá un profesor coordinador.
3. El número máximo de profesores ayudantes en el archivo de credenciales será de 5.

Flujo principal:

1. El caso de uso empieza cuando el sistema necesita identificar al profesor al iniciar.
2. El sistema pregunta al usuario no identificado si es profesor Coordinador o Ayudante.
3. En el caso de seleccionar Coordinador, el sistema pedirá el DNI del profesor coordinador.
4. El sistema comprueba si el DNI introducido coincide con el DNI del profesor Coordinador del archivo de credenciales.
5. Si los DNIs coinciden, el sistema mostrará un menú con 4 opciones: Iniciar, Cambiar DNI, Eliminar Profesor Ayudante y Mostrar todos los Profesores Registrados.
6. En el caso de seleccionar Iniciar, el sistema inicia el programa con las funcionalidades de profesor Coordinador.

Postcondiciones:

- Al identificarse como profesor coordinador tendrá acceso a todas las tareas del sistema.
- Al identificarse como profesor ayudante tendrá accesos a todas las tareas del sistema excepto aquellas tareas que trabajan con las copias de seguridad, las cuales son Guardar/Cargar la base de datos de los alumnos en/de una copia de seguridad binaria.

Flujos alternativos:

1.a. Si el archivo de credenciales no existe, el sistema muestra un mensaje diciendo que el archivo de credenciales no existe.

2.a. En el caso de seleccionar Ayudante, el sistema mostrará un menú con dos opciones: Registrarse o Identificarse.

2.b. En el caso de seleccionar Registrarse, el sistema pedirá el DNI a registrar como profesor Ayudante.

2.c. El sistema guarda el DNI en el archivo de credenciales como profesor Ayudante.

2.d. El sistema inicia el programa con las funcionalidades de profesor Ayudante.

2.b.1.a. En el caso de no seleccionar ninguna opción, el sistema muestra un mensaje de error diciendo que se ha elegido una opción incorrecta.

2.b.1.b. El flujo vuelve al punto 2 del flujo principal.

2.b.2.a. En el caso de seleccionar Registrarse y existir 5 profesores ayudantes en el archivo de credenciales, el sistema muestra un mensaje diciendo que ya hay 5 profesores ayudantes registrados.

2.b.2.b. El flujo vuelve al punto 2 del flujo principal.

2.b.3.a. En el caso de seleccionar Identificarse, el sistema pedirá el DNI para identificarlo.

2.b.3.b. El sistema comprueba si el DNI introducido coincide con alguno de los profesores ayudantes del archivo de credenciales.

2.b.3.c. En el caso de coincidir, el sistema inicia el programa con las funcionalidades de profesor Ayudante.

2.b.3.c.a. En el caso de no coincidir, el sistema muestra un mensaje diciendo que no se ha podido identificar.

2.b.3.c.b. El flujo vuelve al punto 2 del flujo principal.

3.a. En el caso de no seleccionar ninguna opción, el sistema muestra un mensaje de error diciendo que se ha elegido una opción incorrecta.

3.b. El flujo vuelve al punto 2 del flujo principal.

5.a. Si el DNI introducido no concuerda con el del profesor Coordinador de las credenciales, el sistema muestra un mensaje de error diciendo que no se ha podido identificar correctamente.

5.b. El flujo vuelve al punto 2 del flujo principal.

6.a.1.a. En el caso de seleccionar Cambiar DNI, el sistema pedirá al profesor el nuevo DNI que desea.

6.a.1.b. El sistema comprueba que el DNI introducido no coincide con alguno de los profesores ayudantes de las credenciales.

6.a.1.c. Si no coincide, el sistema cambia el DNI del profesor coordinador por el DNI insertado en las credenciales.

6.a.1.d. El sistema muestra un mensaje diciendo que el cambio se ha realizado correctamente.

6.a.1.e. El flujo vuelve al punto 5 del flujo principal.

6.a.1.c.a. Si coincide con alguno, el sistema muestra un mensaje de error diciendo que el DNI introducido coincide con el DNI de un profesor ayudante.

6.a.1.c.b. El flujo vuelve al punto 5 del flujo principal.

6.a.2.a. En el caso de seleccionar Eliminar Profesor Ayudante, el sistema pedirá al profesor el DNI del profesor Ayudante a Eliminar

6.a.2.b. El sistema comprueba que el DNI introducido coincide con alguno de los profesores ayudantes de las credenciales.

6.a.2.c. Si coincide, el sistema elimina el profesor ayudante encontrado de las credenciales.

6.a.2.d. El sistema muestra un mensaje diciendo que el profesor ayudante se ha eliminado correctamente.

6.a.2.e. El flujo vuelve al punto 5 del flujo principal.

6.a.2.c.a. Si no coincide con ninguno, el sistema muestra un mensaje de error diciendo que no se encuentra ningún profesor ayudante con ese DNI.

6.a.2.c.b. El flujo vuelve al punto 5 del flujo principal.

6.a.3.a. En el caso de seleccionar Mostrar todos los Profesores Registrados, el sistema recopilará del archivo de credenciales todos los datos de los profesores.

6.a.3.b. El sistema mostrará por pantalla el número de profesores registrados y, a continuación, escribirá sus roles (Coordinador o Ayudante) y sus DNIs correspondientes.

6.a.3.c. El flujo vuelve al punto 5 del flujo principal.

4.10. C.U. de Cargar datos de una copia de seguridad binaria

Cargar datos de una copia de seguridad binaria

ID: 010

Breve descripción: El sistema carga los datos de los alumnos de una copia de seguridad binaria.

Actores principales: Profesor Coordinador.

Actores secundarios: Alumnos.

Precondiciones:

1. El fichero a cargar tiene que ser binario.
2. El fichero binario puede tener como máximo 150 alumnos.
3. El fichero binario a cargar debe existir en el sistema.

Flujo principal:

1. El caso de uso empieza cuando el profesor necesita cargar los datos de los alumnos de un fichero binario.
2. El sistema pregunta al profesor el nombre del fichero binario a cargar.
3. El sistema carga los datos del fichero binario en el sistema.

Postcondiciones:

- Una vez cargado el fichero, el sistema tiene la base de datos de los alumnos del fichero binario cargado.

Flujos alternativos:

2.a. Si el fichero binario a cargar no existe, muestra un mensaje de error.

3.a. Si el fichero binario tiene más de 150 alumnos, muestra un mensaje de error.

4.11. C.U. de Guardar datos en una copia de seguridad binaria

Guardar datos en una copia de seguridad binaria

ID: 011

Breve descripción: El sistema guarda la base de datos de los alumnos en una copia de seguridad.

Actores principales: Profesor Coordinador.

Actores secundarios: Alumnos.

Precondiciones:

1. Si la base de datos está vacía, el archivo binario donde se guardarán los datos quedará vacío.

Flujo principal:

1. El caso de uso empieza cuando el profesor necesita guardar los datos de los alumnos en un fichero binario.
2. El sistema pregunta al profesor el nombre del fichero binario a crear/sobrescribir para guardar los datos.
3. El sistema crea/sobrescribe el fichero y guarda la base de datos de los alumnos.

Postcondiciones:

- El fichero binario tiene todos los datos de los alumnos de la base de datos en ese instante al ser guardado.

Flujos alternativos:

2.a. Si el nombre introducido no es válido, es decir, tenga el mismo nombre que de alguno de importancia(credenciales, cuerpo, etc), se volverá a pedir por teclado el nombre del fichero binario a guardar hasta que sea un nombre válido.

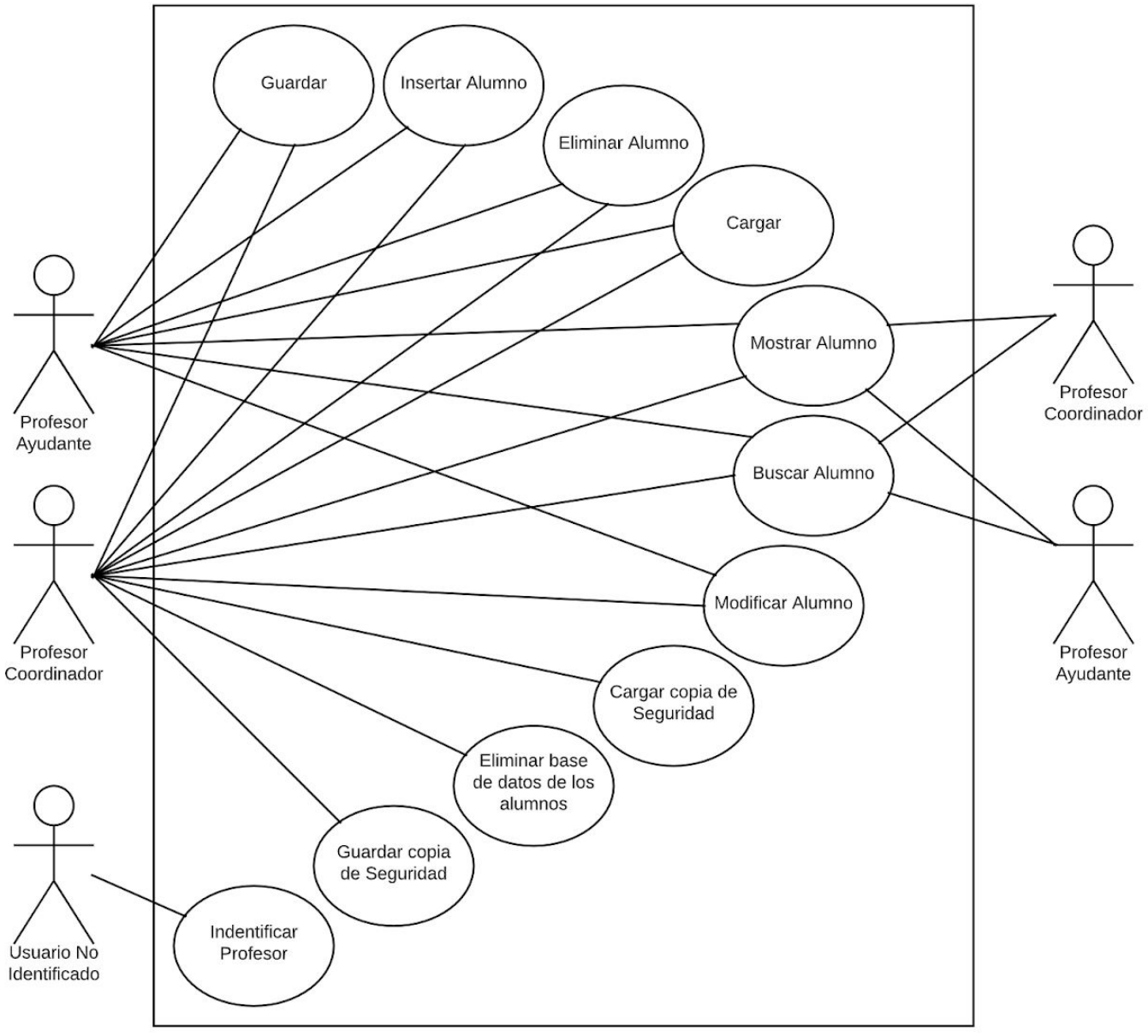
3.a. Si hay un error a la hora de crearse/sobrescribirse el fichero, se muestra un mensaje de error.

4.12. Diagrama de Casos de Uso

Los diagramas de caso de uso modelan la funcionalidad del sistema usando actores y los casos de uso. Los casos de uso anteriormente descritos son los servicios o funciones provistas por el sistema para sus usuarios.

El diagrama de caso de uso representa a un sistema o subsistema como un conjunto de interacciones que se desarrollarán entre casos de uso y entre estos y sus actores en respuesta a un evento en concreto.

A continuación se muestra nuestro Diagrama de Casos de Uso:



5 DIAGRAMA DE CLASES

El diagrama de clases recoge las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos, pero no muestra información temporal.

A continuación redacto y explico el Diagrama de Clases y su composición:

Explicación del Diagrama de Clase

El Diagrama de Clase consiste en una estructura estática que muestra las **clases, el sistema, atributos, métodos y relaciones entre ellos.**

Como podremos observar más tarde, nuestro diagrama de clase está compuesto por **3 clases** diferentes, estas son:

1. **Alumno**
2. **Base de Datos**
3. **Profesor**

También está formado por **2 relaciones** diferentes, estas son:

1. **Relación Alumno - Base de Datos**
2. **Relación Base de Datos - Profesor**

Clases

A continuación, se van a ir **desarrollando cada una de las clases** que contiene nuestro diagrama:

Alumno

La **clase Alumno** representa al alumno y todos sus datos.

Atributos

A continuación, se describirán todos los atributos de la clase Alumno.

Cabe resaltar que todos sus atributos son privados, estos son:

1. **Nombre:** Representa el nombre del alumno. Es un atributo de tipo *string*.
2. **Apellidos:** Representa los apellidos del alumno. Es un atributo de tipo *string*.
3. **DNI:** Representa el DNI del alumno. Es un atributo de tipo *string*.
4. **Correo:** Representa el correo del alumno. Es un atributo de tipo *string*.
5. **Dirección:** Representa la dirección del alumno. Es un atributo de tipo string.
6. **Teléfono:** Representa el número de teléfono del alumno. Es un atributo de tipo *string*, ya que podían existir números de telefono mayores a lo que un *int* y un *double* podían representar.
7. **Dirección:** Representa la dirección del alumno. Es un atributo de tipo *string*.
8. **Curso más alto matriculado:** Representa el curso más alto matriculado del alumno. Es un atributo de tipo *int*.
9. **Fecha nacimiento:** Representa la fecha de nacimiento del alumno. Es un atributo de tipo *string*.
10. **¿Líder?:** Representa si el alumno es lider o no de un grupo. Es un atributo de tipo *boolean*.
11. **Número del Grupo:** Representa el número del grupo del alumno. Es un atributo de tipo *int*.

Operaciones

A continuación, se describirán todas las operaciones de la clase Alumno.

Cabe resaltar que **todas sus operaciones son públicas**, estas se pueden catalogar en dos tipos:

1. **Modificadores (set):** Son operaciones que, con el valor pasado, **modifican el atributo** correspondiente. Todos estos tipos de operaciones son de **tipo *void***. Por ejemplo, la operación **setDNI()** modificará el DNI del alumno en cuestión con el valor pasado. Algunos ejemplos más son: **setNombre(), setApellidos(), etc.**
2. **Observadores (get):** Son operaciones que **devuelven el atributo** correspondiente. Cada operación de este tipo devuelve un atributo del **tipo correspondiente a su atributo**. Por ejemplo, la operación **getDNI()** devolverá el DNI del alumno en cuestión, es decir, devolverá un atributo de tipo *string*. Algunos ejemplos más son: **getNombre(), getApellidos(),** etc.

Base de Datos

La **clase Base de Datos** representa la base de datos de todos los alumnos y las operaciones que puede llevar a cabo esta.

Atributos

Su único atributo es **Lista Alumnos** el cual representa la **lista** de todos los alumnos. Es una lista de tipo *Alumno*, es decir, es una lista de elementos de la **clase Alumno**. El atributo es de tipo **privado**, ya que si no lo fuera se puede vulnerar la seguridad de la base de datos.

Operaciones

A continuación, se describirán todas las operaciones de la clase Base de datos:

1. **buscarAlumno():** Se encarga de **buscar un alumno** en específico en la *Lista Alumnos*. Es de tipo *bool* y es una operación **pública.**
2. **insertarAlumno():** Se encarga de **insertar un nuevo alumno** en la *Lista Alumnos*. Es de tipo *bool* y es una operación **pública.**
3. **eliminarAlumno():** Se encarga de **eliminar un alumno** en específico de la *Lista Alumnos*. Es de tipo *void* y es una operación **pública**.
4. **mostrarAlumno():** Se encarga de **mostrar un alumno o todos los alumnos** de la *Lista Alumnos*de distintas formas (ascendente por apellido o descendente por DNI, etc). Es de tipo *void* y es una operación **pública.**
5. **modificarAlumno()**: Se encarga de **modificar un alumno** en específico de la *Lista Alumnos*. Es de tipo *void* y es una operación **pública.**
6. **guardar():** Se encarga de **guardar los datos en un fichero preestablecido** de la *Lista Alumnos*. Es de tipo *void* y es una operación **pública.**
7. **guardarCopia():** Se encarga de **guardar los datos en una copia de seguridad binaria** de la *Lista Alumnos*. Es de tipo *void*y es una operación **pública.**
8. **cargar():** Se encarga de **cargar los datos de un fichero preestablecido** a la *Lista Alumnos*. Es de tipo *void* y es una operación **pública.**
9. **cargarCopia():** Se encarga de **cargar los datos de una copia de seguridad binaria** a la *Lista Alumnos*. Es de tipo *void* y es una operación **pública.**
10. **getListaAlumno():** Se encarga de **devolver el atributo *Lista Alumnos*.** Es de tipo *list* y es una operación **pública.**

11. **setListaAlumno():** Se encarga de **modificar el atributo *Lista Alumnos*** con el valor pasado. Es de tipo *void* y es una operación **pública.**

Profesor

La **clase Profesor** representa a un profesor de la clase de los alumnos.

Atributos

A continuación, se describirán todos los atributos de la clase Profesor.

1. **DNI:** Representa el DNI del profesor. Es un atributo de tipo *string*. Este atributo es **privado.**
2. **Rol:** Representa el tipo del profesor, es decir, si es Profesor Coordinador o Profesor Ayudante. Es un atributo de tipo *string*. Este atributo es **privado.**
3. **Base:** Representa la base de alumnos con la que interactúa el profesor. Este atributo es de tipo *BaseAlumnos*. Este atributo es **público.**

Operaciones

A continuación, se describirán todas las operaciones de la clase Profesor.

Cabe resaltar que **todas sus operaciones son públicas**, estas son:

1. **eliminarBaseAlumno():** Se encarga de **eliminar la base de datos** de los alumnos completamente. Es de tipo *void*.
2. **identificaProfesor():** Se encarga de **registrar o identificar al profesor** que accede al programa. Los datos utilizados para registrarse o identificarse se guardarán en la *Base de datos de los Profesores*. Es de tipo *void*
3. **setDni():** Se encarga de **modificar el DNI** del profesor en cuestión con el valor pasado. Es de tipo *void*.
4. **getDni():** Se encarga de **devolver el DNI** del profesor en cuestión. Es de tipo *string*.
5. **setRol():** Se encarga de **modificar el tipo (coordinador o ayudante)** del profesor en cuestión con el valor pasado. Es de tipo *void*.
6. **getRol():** Se encarga de **devolver el tipo** del profesor en cuestión. Es de tipo *string*.

Relaciones entre las Clases

A continuación, se van a ir **desarrollando cada una de las relaciones entre clases** que contiene nuestro diagrama:

Relación Alumno - Base de Datos

Como se puede observar, la relación indica que **un alumno está en una sola base de datos y una base de datos está compuesta por 0 o muchos alumnos.**

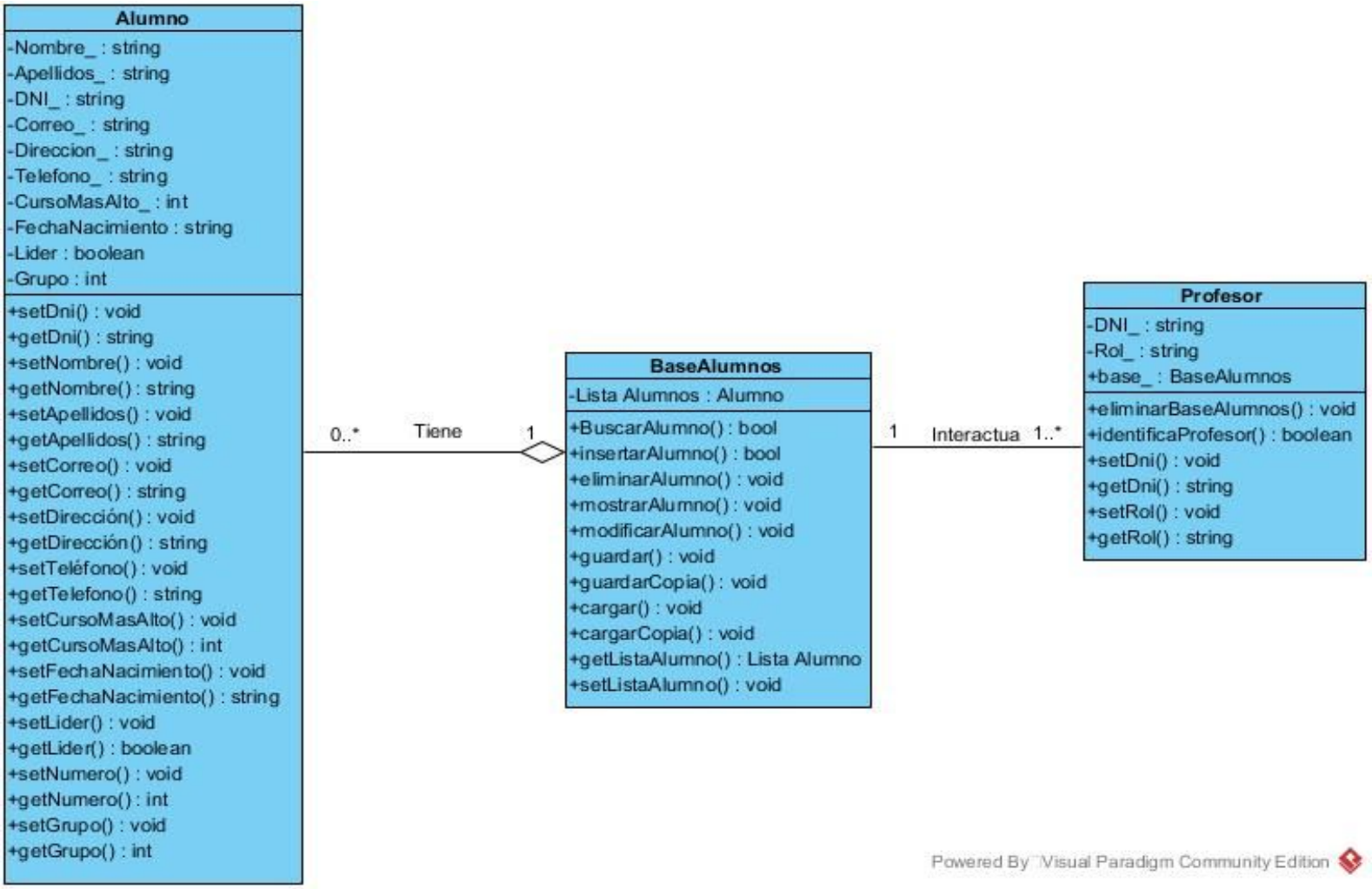
Esta relación es una relación de **Agregación**, es decir, la base de datos está compuesta por alumnos.

Relación Base de Datos - Profesor

Esta relación indica que **un profesor interactúa con una base de datos de los alumnos, y una base de datos de los alumnos es utilizada por uno o varios profesores.**

Esta relación es una relación de **Asociación**, es decir, es una relación estática entre ambas clases.

A continuación se muestra nuestro **Diagrama de Clases**:



6 Diagrama de secuencia

Explicación de los diagramas de secuencia

Como podemos observar nuestro programa tiene 11 métodos diferentes. Los diferentes diagramas de secuencia de los métodos especificarán cómo se desarrollarán las ejecuciones de los diferentes métodos, así como las diferentes ejecuciones alternativas de estos.

Los diagramas de secuencia que corresponden a nuestro sistema son:

1. **Secuencia Borrar:** Diagrama de secuencia de la función *eliminarAlumno*.

2. **Secuencia Borrar Base de Datos:** Diagrama de secuencia de la función *eliminarBaseAlumnos*.

3. **Secuencia Buscar:** Diagrama de secuencia de la función *buscarAlumno*.

4. **Secuencia Cargar:** Diagrama de secuencia de la función *cargar*.

5. **Secuencia Cargar Copia de Seguridad:** Diagrama de secuencia de la función *cargarCopia*.

6. **Secuencia Guardar:** Diagrama de secuencia de la función *guardar*.

7. **Secuencia Guardar Copia de Seguridad:** Diagrama de secuencia de la función *guardarCopia.*

8. **Secuencia Identificar Profesor:** Diagrama de secuencia de la función *identificaProfesor.*

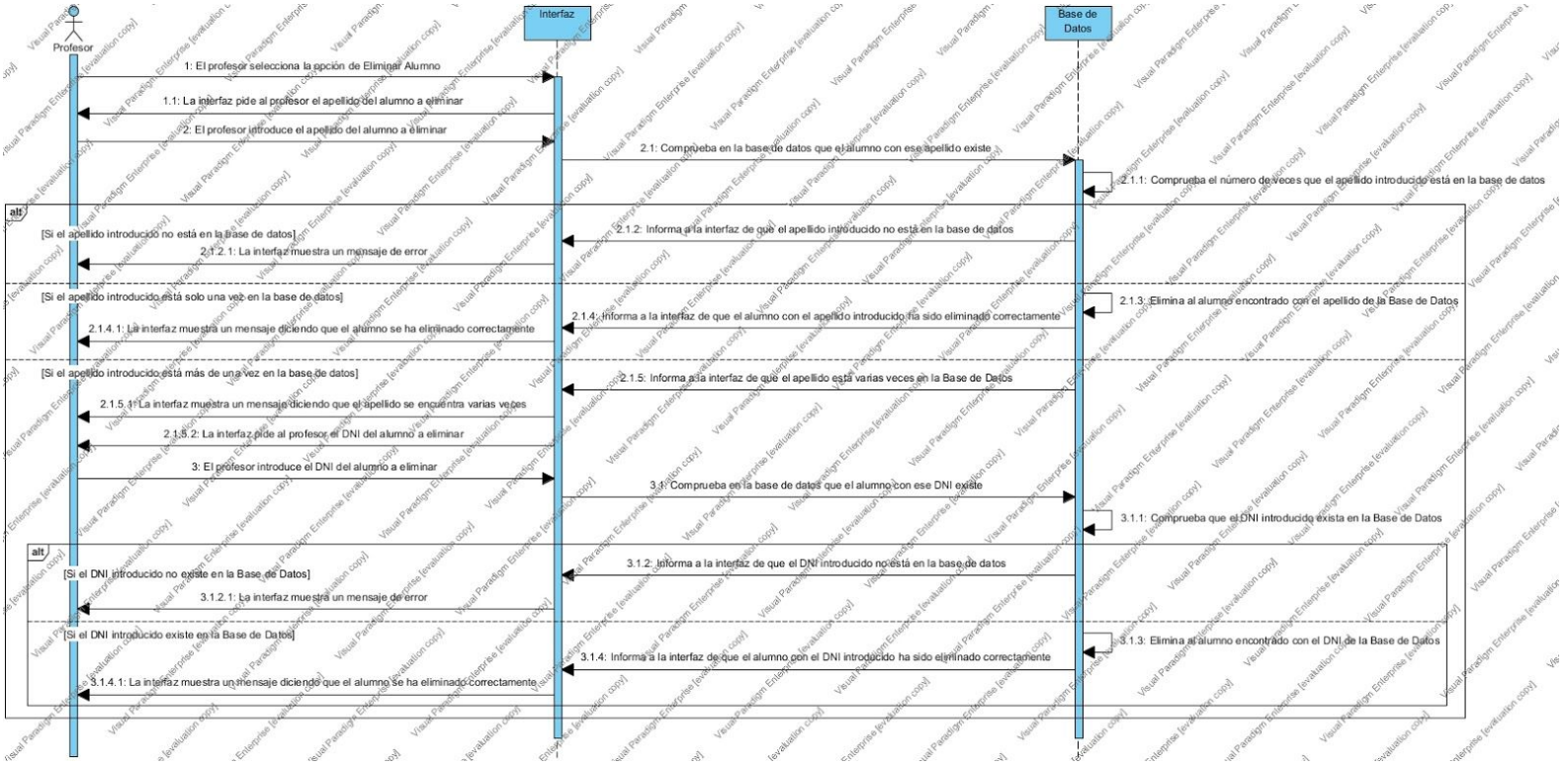
9. **Secuencia Insertar:** Diagrama de secuencia de la función *insertarAlumno*.

10. **Secuencia Modificar:** Diagrama de secuencia de la función *modificarAlumno*.

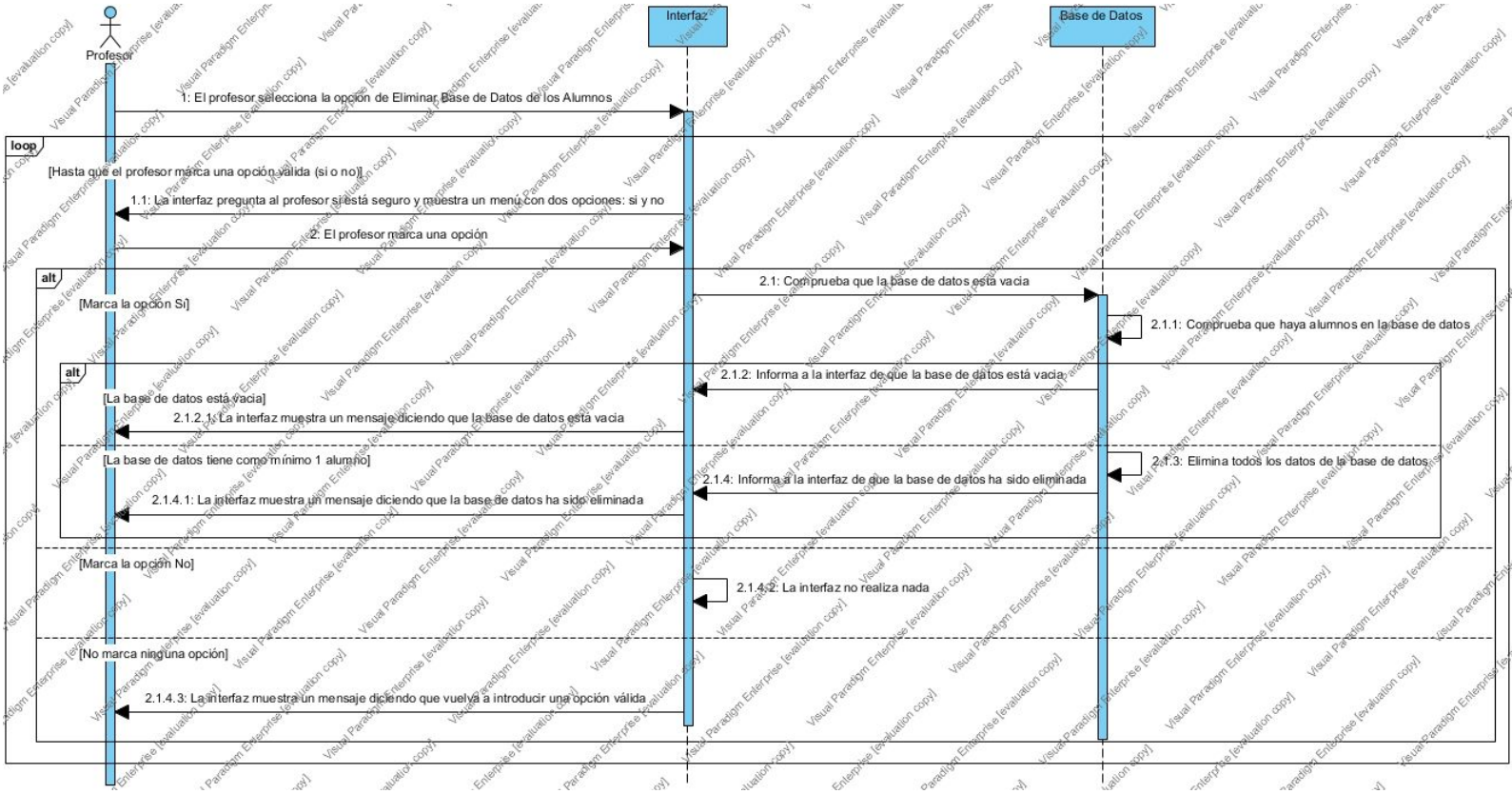
11. **Secuencia Mostrar:** Diagrama de secuencia de la función *mostrarAlumno*.

A continuación se muestran cada uno de los diagramas de secuencia de nuestro sistema:

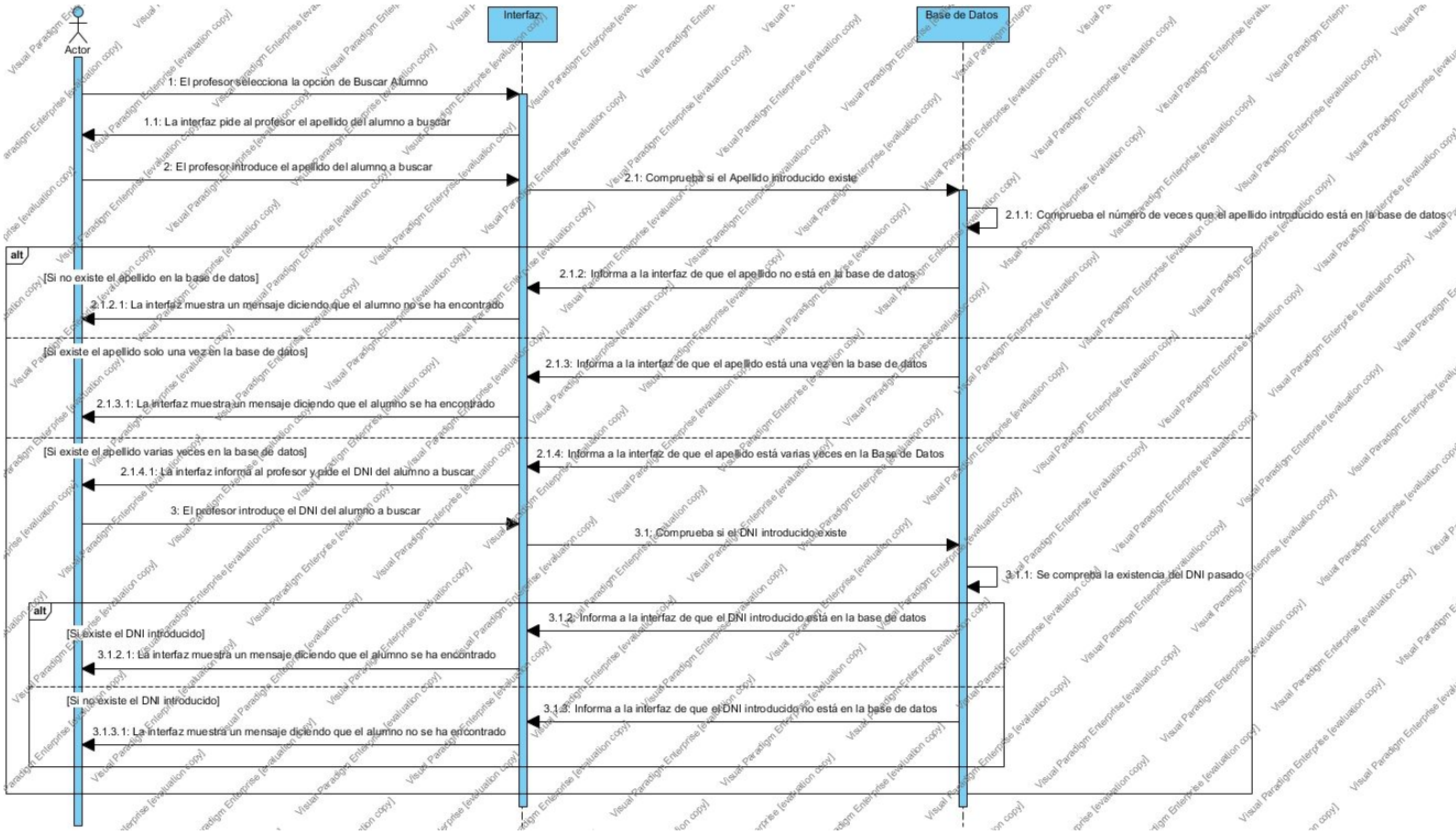
6.1. Secuencia Borrar



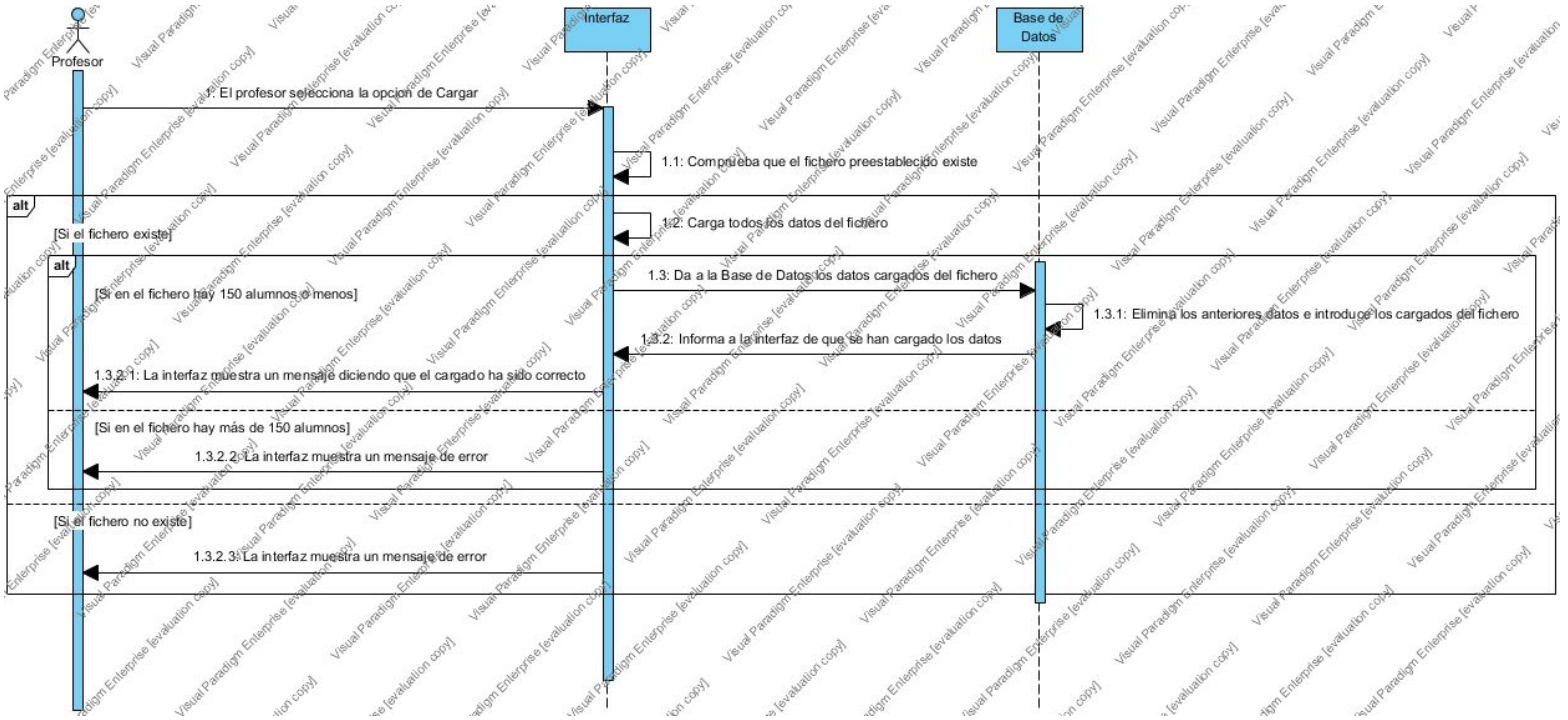
6.2. Secuencia Borrar Base de Datos



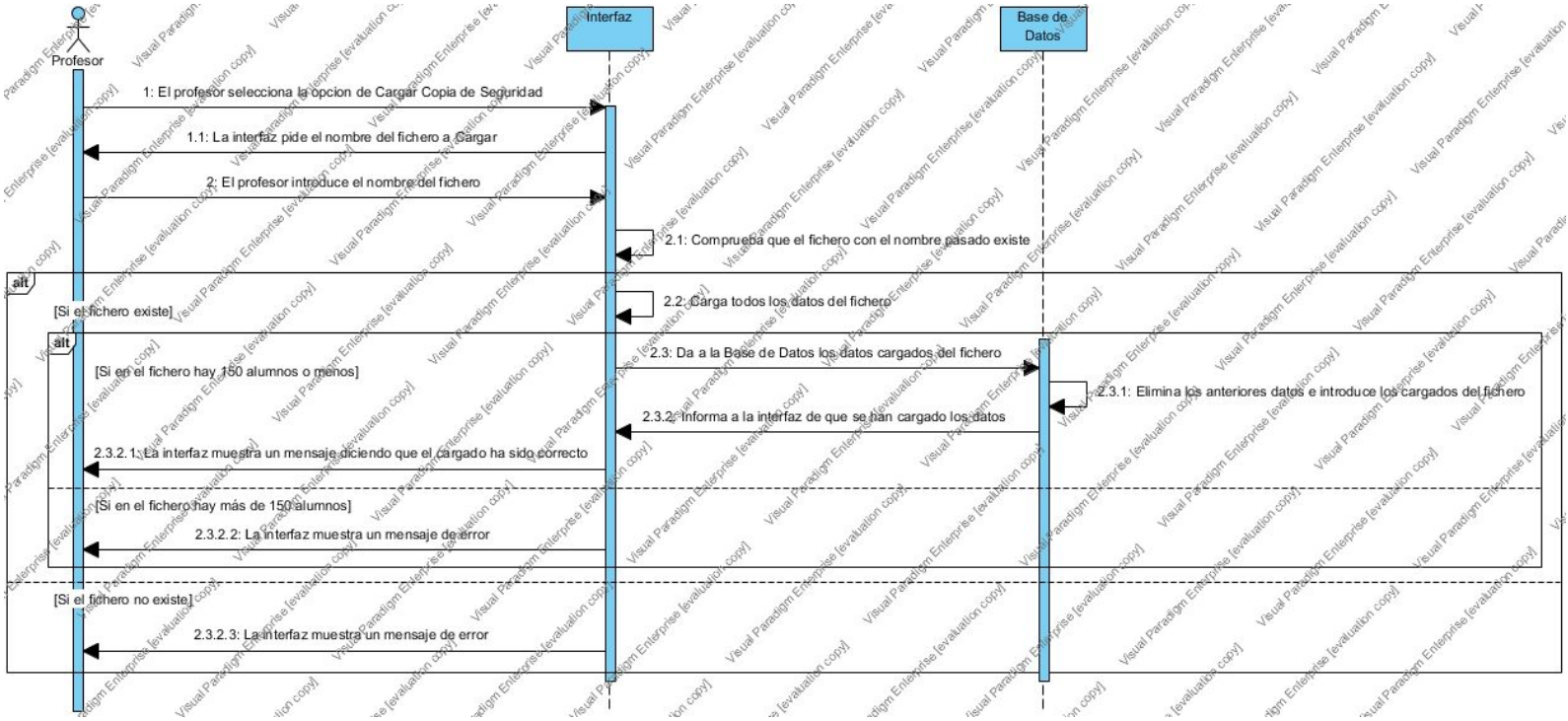
6.3. Secuencia Buscar



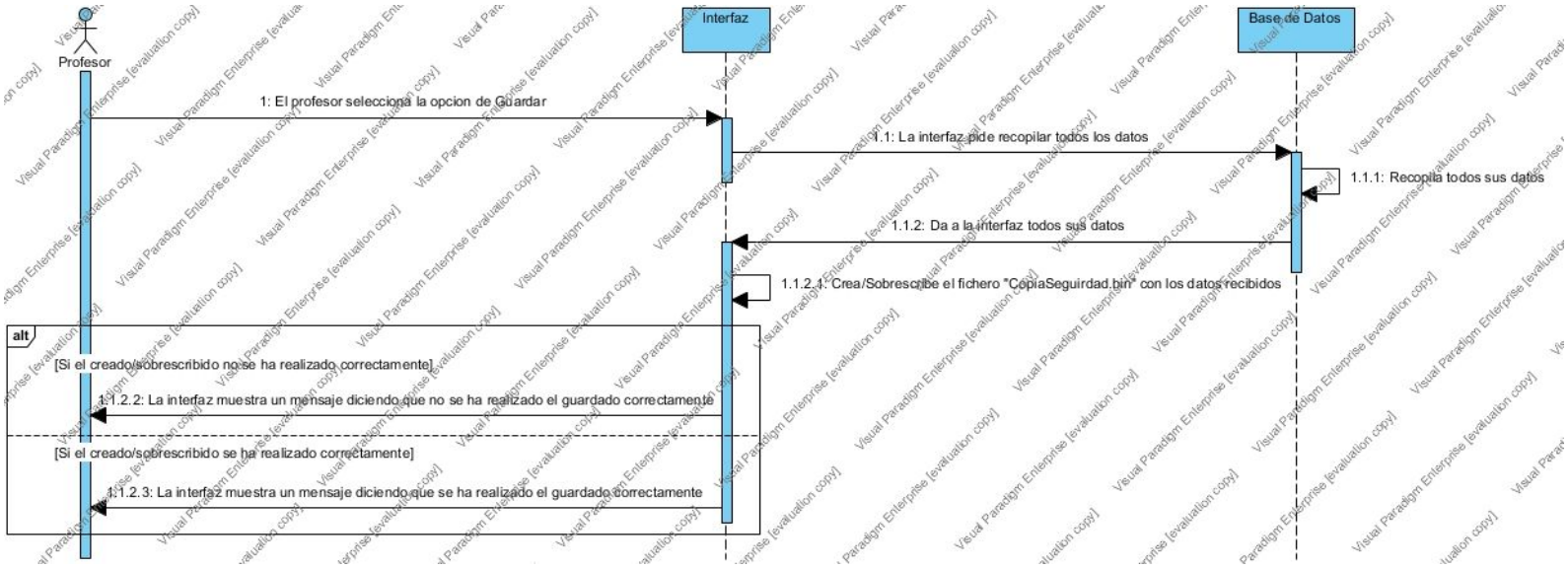
6.4. Secuencia Cargar



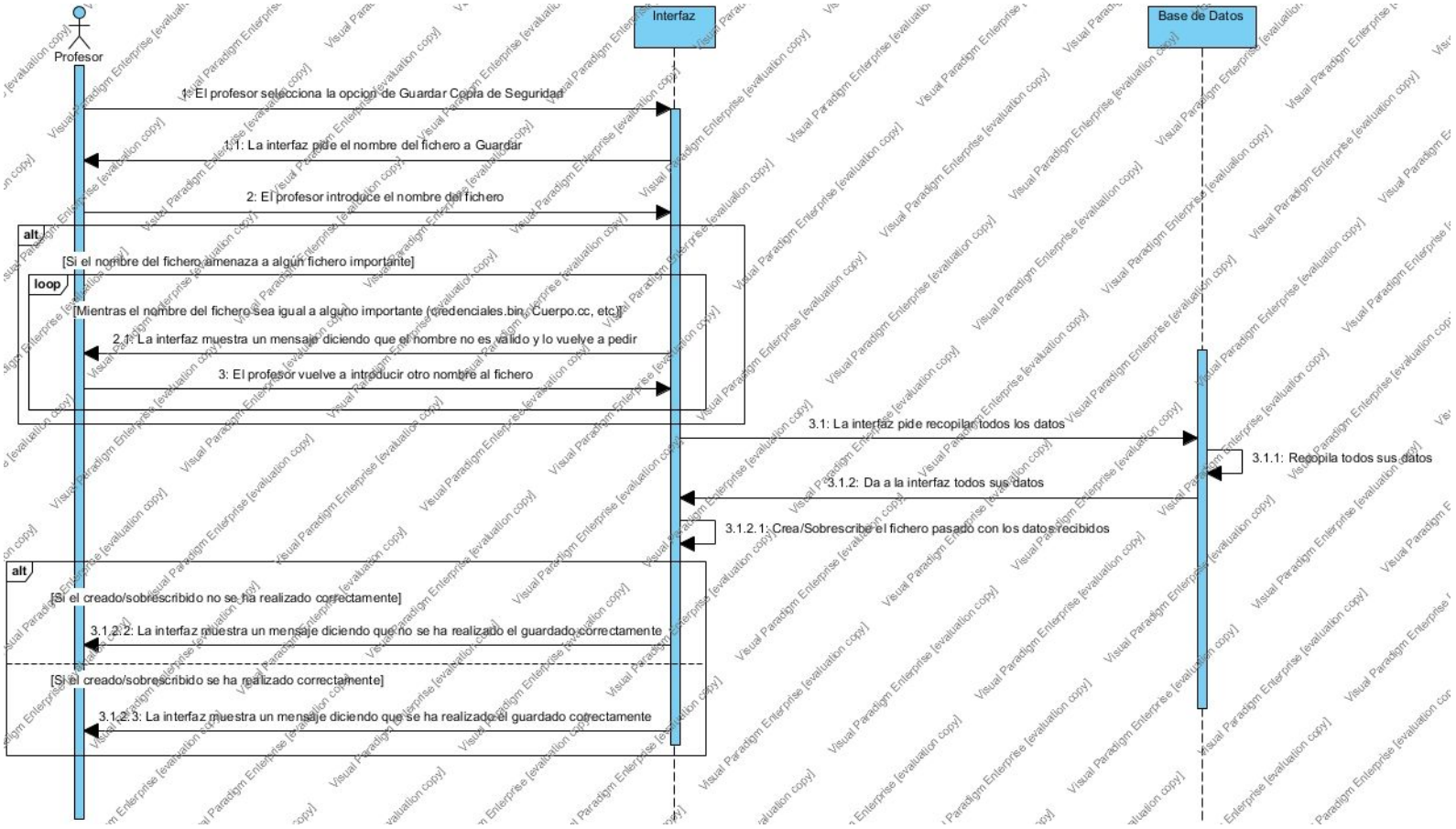
6.5. Secuencia Cargar Copia Seguridad



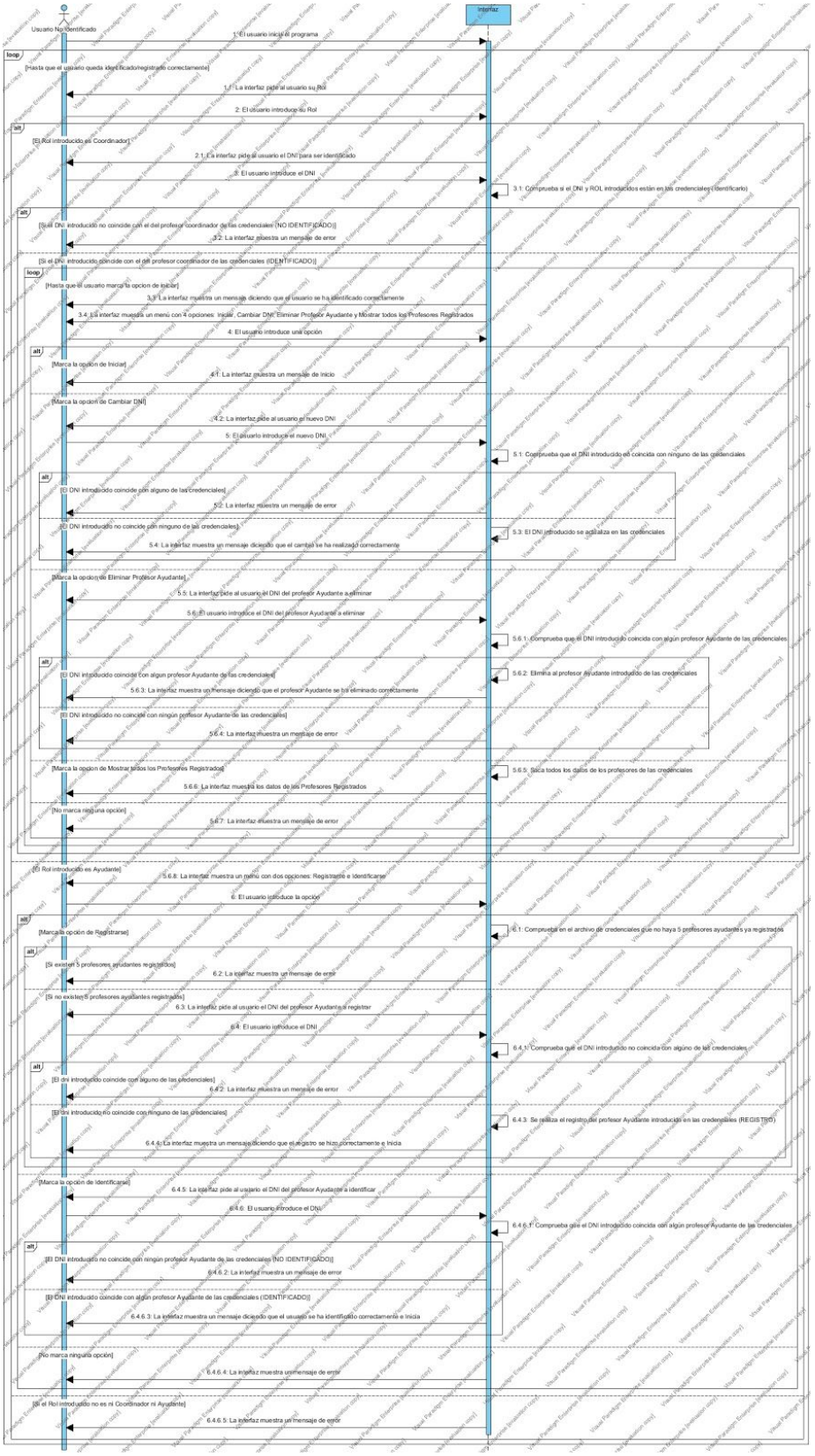
6.6. Secuencia Guardar



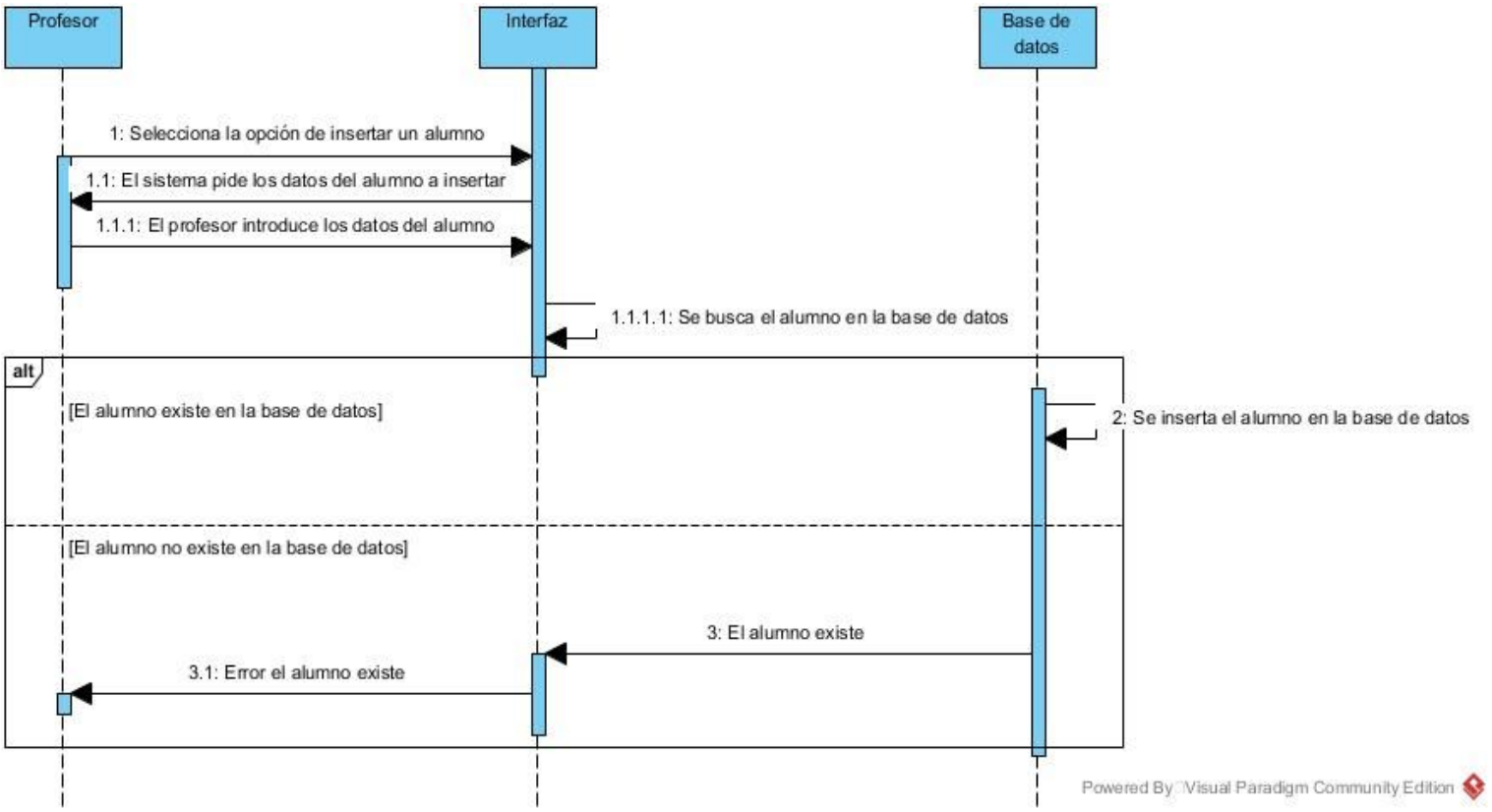
6.7. Secuencia Guardar Copia Seguridad



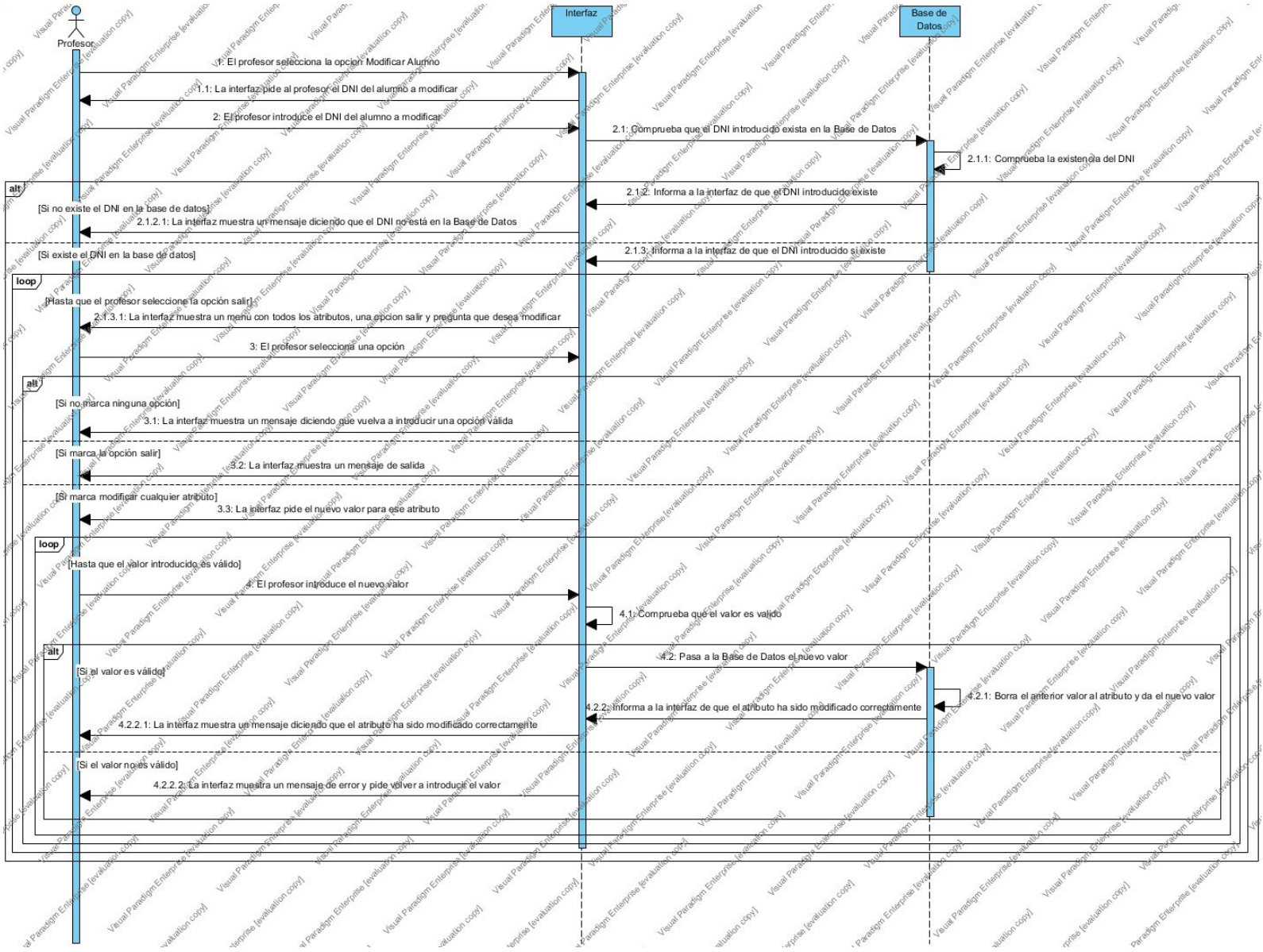
6.8. Secuencia Identificar Profesor



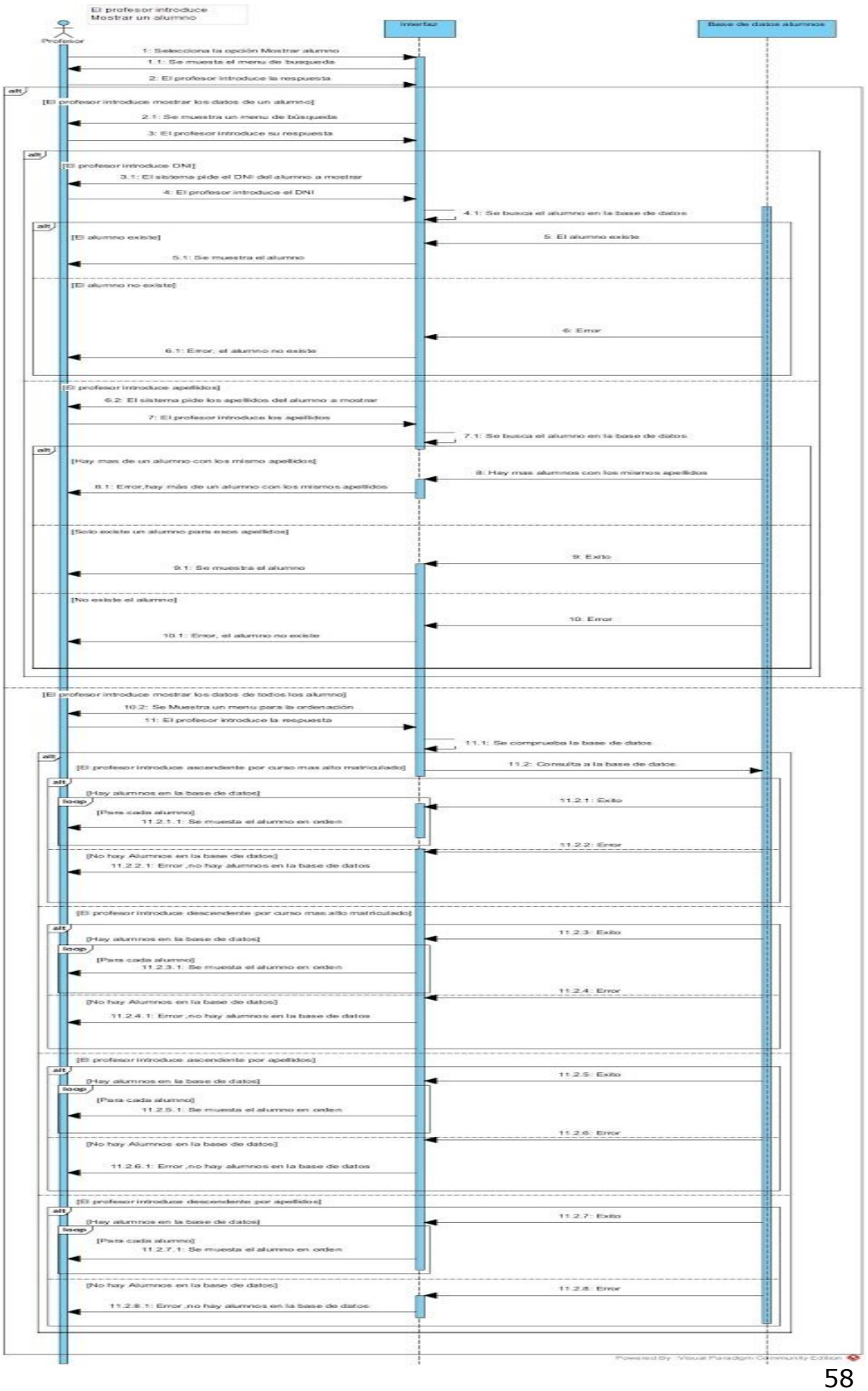
6.9. Secuencia Insertar



6.10. Secuencia Modificar



6.11. Secuencia Mostrar



7 Metodología SCRUM (Product backlog, Sprint backlogs y Burndown charts)

7.1. Product Backlog

El product backlog consiste en un archivo que contiene todas las funcionalidades ordenadas por prioridades. Es la agrupación de todas las historias de usuario del producto.

Este se realiza para tener una perspectiva de todo lo que se quiere hacer y tener claras las prioridades del cliente.

Ayuda a que el equipo sea más autodisciplinado y respete las prioridades del cliente.

A continuación se muestra el contenido de nuestro archivo Product Backlog:

Product Backlog

Este archivo es el ***Product Backlog***, el cual se encarga de nombrar todas las **historias de usuario** detectadas, su **prioridades** otorgadas y los **tiempos estimados** de implementación.

Historia de Usuario - Prioridad - Tiempo estimado

- **Identificar Profesores** - Prioridad **1** - Tiempo estimado: **8 horas**.
- **Insertar Alumno** - Prioridad **1** - Tiempo estimado: **5 horas.**
- **Cargar** - Prioridad **2** - Tiempo estimado: **1 hora.**
- **Guardar** - Prioridad **2** - Tiempo estimado: **1 hora.**
- **Cargar copia de seguridad de un fichero binario** - Prioridad **2** - Tiempo estimado: **1 hora.**
- **Guardar copia de seguridad en un fichero binario** - Prioridad **2** - Tiempo estimado: **1 hora.**
- **Buscar Alumno** - Prioridad **3** - Tiempo estimado: **2 horas.**
- **Mostrar Alumno** - Prioridad **4** - Tiempo estimado: **5 horas.**
- **Eliminar Alumno** - Prioridad **4** - Tiempo estimado: **2 horas.**
- **Modificar Alumno** - Prioridad **4** - Tiempo estimado: **3 horas.**
- **Eliminar base de datos de los alumnos** - Prioridad **5** - Tiempo estimado: **1 hora**.

Como se puede observar, la suma total del tiempo estimado es de **30 horas**.

7.2. Sprint Backlogs

Sprint Backlog es el listado de tareas en el que subdivide las historias de usuario que describen las funcionalidades que componen nuestro proyecto.

Este listado se define y estima en la reunión de Planificación del Sprint al inicio de la iteración.

Este listado se va actualizando a medida que avanza el proyecto ya que las tareas se van realizando progresivamente. Es por ello que tenemos 4 versiones de Sprint Backlogs diferentes.

Seguidamente, se van a mostrar cada una de esas versiones:

7.2.1. Sprint Backlog Ver. 1

Sprint Backlog

Este archivo es el ***Sprint Backlog***, el cual se encarga de **distribuir** los distintos *sprint backlogs*. Dicho de otro modo, en este documento se hará **la distribución de las distintas tareas del proyecto a desarrollar**.

Este archivo es desarrollado por el ***Scrum Master***, el cual, en nuestro caso se trata del líder/administrador del grupo, es decir, Felipe Mérida Palacios.

El programador número 3, es decir, Ángel Santos Ramírez se encargará de 2 funciones más, ya que Cargar/Guardar Copia de Seguridad es idéntica a Cargar y Guardar, excepto por la variación de poder introducir en nombre del fichero.

Programadores

Los programadores del grupo son los siguientes:

- Programador 1 y Scrum Master: **Felipe Mérida Palacios**
- Programador 2: **Ricardo Mifsut Castilla**
- Programador 3: **Ángel Santos Ramírez**

Tareas

A continuación se van a nombrar las tareas para cada programador:

Programador 1 - Felipe Mérida Palacios

- Identificar Profesor
- Eliminar Alumno
- Eliminar base de datos de los alumnos

Programador 2 - Ricardo Mifsut Castilla

- Modificar Alumno
- Insertar Alumno
- Buscar Alumno

Programador 3 - Ángel Santos Ramírez

- Mostrar Alumno
- Cargar
- Guardar
- Cargar Copia de Seguridad
- Guardar Copia de Seguridad

7.2.2. Sprint Backlog Ver. 2

Sprint Backlog

Este archivo es el ***Sprint Backlog***, el cual se encarga de **distribuir** los distintos *sprint backlogs*. Dicho de otro modo, en este documento se hará **la distribución de las distintas tareas del proyecto a desarrollar.**

Este archivo es desarrollado por el ***Scrum Master***, el cual, en nuestro caso se trata del líder/administrador del grupo, es decir, Felipe Mérida Palacios.

Programadores

Los programadores del grupo son los siguientes:

- Programador 1 y Scrum Master: **Felipe Mérida Palacios**
- Programador 2: **Ricardo Mifsut Castilla**
- Programador 3: **Ángel Santos Ramírez**

Tareas

A continuación se van a nombrar las tareas para cada programador:

Programador 1 - Felipe Mérida Palacios

Programador 2 - Ricardo Mifsut Castilla

- Modificar Alumno
- Buscar Alumno

Programador 3 - Ángel Santos Ramírez

- Cargar en un fichero binario
- Guardar en un fichero binario
- Cargar
- Guardar

7.2.3. Sprint Backlog Ver. 3

Sprint Backlog

Este archivo es el ***Sprint Backlog***, el cual se encarga de **distribuir** los distintos *sprint backlogs*. Dicho de otro modo, en este documento se hará **la distribución de las distintas tareas del proyecto a desarrollar.**

Este archivo es desarrollado por el ***Scrum Master***, el cual, en nuestro caso se trata del líder/administrador del grupo, es decir, Felipe Mérida Palacios.

Programadores

Los programadores del grupo son los siguientes:

- Programador 1 y Scrum Master: **Felipe Mérida Palacios**
- Programador 2: **Ricardo Mifsut Castilla**
- Programador 3: **Ángel Santos Ramírez**

Tareas

A continuación se van a nombrar las tareas para cada programador:

Programador 1 - Felipe Mérida Palacios

Programador 2 - Ricardo Mifsut Castilla

Programador 3 - Ángel Santos Ramírez

- Cargar

7.2.4. Sprint Backlog Ver. Final

Sprint Backlog

Este archivo es el ***Sprint Backlog***, el cual se encarga de **distribuir** los distintos *sprint backlogs*. Dicho de otro modo, en este documento se hará **la distribución de las distintas tareas del proyecto a desarrollar.**

Este archivo es desarrollado por el ***Scrum Master***, el cual, en nuestro caso se trata del líder/administrador del grupo, es decir, Felipe Mérida Palacios.

Programadores

Los programadores del grupo son los siguientes:

- Programador 1 y Scrum Master: **Felipe Mérida Palacios**
- Programador 2: **Ricardo Mifsut Castilla**
- Programador 3: **Ángel Santos Ramírez**

Tareas

A continuación se van a nombrar las tareas para cada programador:

Programador 1 - Felipe Mérida Palacios

Programador 2 - Ricardo Mifsut Castilla

Programador 3 - Ángel Santos Ramírez

7.3. Burndown Charts

El Burndown Chart consiste en un gráfico que muestra cantidad de trabajo realizado hasta el momento actual.

Como la línea del tiempo es progresiva, hemos realizado dos Burndown Charts diferentes.

El eje vertical de nuestro Burndown Charts muestra la suma del trabajo acumulado a realizar. Este se mide en horas. El total son 30 horas ya como hemos indicado anteriormente en el Product Backlog.

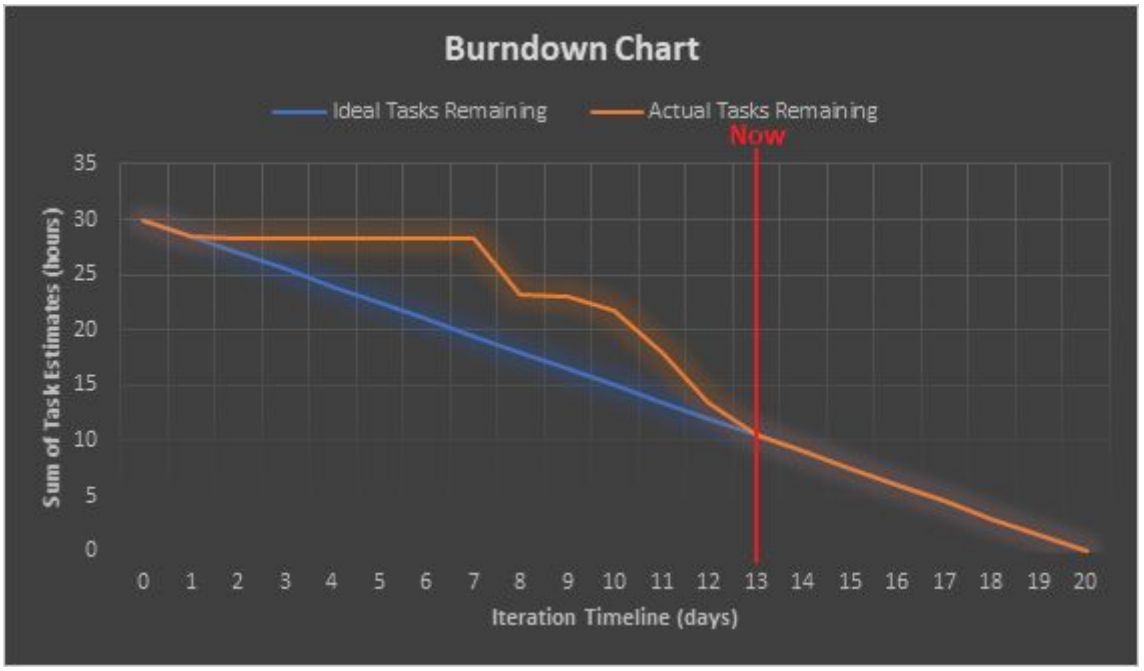
El eje horizontal representa la línea de tiempo. El tiempo máximo que tuvimos fue de 20 días.

La línea diagonal azul representa la cantidad ideal de tarea restante que quedaría por realizar en un momento determinado y la línea naranja representa la cantidad real de tarea restante que queda por realizar en ese mismo instante.

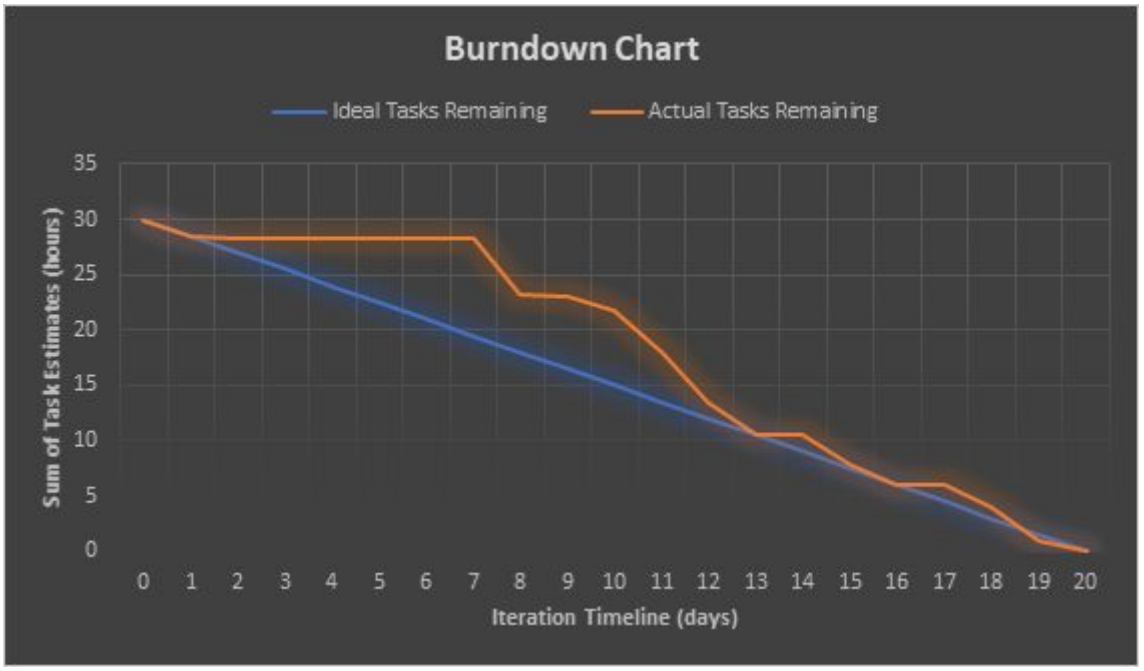
A continuación se muestran los dos Burndown Charts realizados:

7.3.1. Burndown Chart Ver. 1

En este caso, la línea roja vertical representa el momento del tiempo actual en ese instante.



7.3.2. Burndown Chart Ver. Final



8 MATRICES DE VALIDACIÓN

A continuación se mostraran las matrices de validación, en nuestro caso, hemos utilizado como técnicas de validación 2 matrices de trazabilidad:

- Matriz requisitos funcionales (RF) - casos de uso (CU).
- Matriz casos de uso (CU) - clases.

En primer lugar, le mostramos la Matriz requisitos funcionales (RF) - casos de uso (CU).

En esta matriz cada RF debe quedar cubierto por al menos un CU. Con esto nos aseguramos que todas las funcionalidades requeridas son tenidas en cuenta.

MATRIZ REQUISITO FUNCIONALES(RF) - CASOS DE USO(CU)											
REQ. FUNCIO-NALES	**CASOS DE USO**										
	Insertar alumno	Modificar alumno	Buscar alumno	Mostrar alumno	Borrar alumno	Borrar base de datos	Cargar	Guardar	Ident. prof.	Cargar copia	Guardar copia
Ident. prof.									✔		
Insertar alumno	✔										
Buscar Alumno			✔								
Mostrar alumno				✔							
Cargar							✔				
Cargar copia										✔	
Guardar								✔			
Guardar copia											✔
Eliminar alumno					✔						
Modificar alumno		✔									
Eliminar base de datos						✔					

Seguidamente, le mostraremos la Matriz casos de uso (CU) - clases.

En esta matriz cada caso de uso debe tener asignado una clase al menos, en caso contrario, nos faltaría mejorar la definición de la clase, o la creación de otra.

MATRIZ CASOS DE USO(CU) - CLASES											
CLASES	**CASOS DE USO**										
	Insertar alumno	Modificar alumno	Buscar alumno	Mostrar alumno	Borrar alumno	Borrar base de datos	Cargar	Guardar	Ident. prof.	Cargar copia	Guardar copia
Alumno	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔
Base Alumnos	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔
Profesor						✔			✔		

9 Bibliografía y Referencias Web

https://github.com/

https://www.lucidchart.com/

https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso

https://es.wikipedia.org/wiki/Caso_de_uso

https://es.wikipedia.org/wiki/Historias_de_usuario

https://www.genbeta.com/desarrollo/historias-de-usuario-una-forma-natural-de-analisis-funcional

https://solvingadhoc.com/las-historias-usuario-funcion-agilidad/

http://www.cplusplus.com/reference/

https://www.youtube.com/watch?v=aYSJUct2PvU

http://c.conclase.net/curso/?cap=039

https://www.dlsi.ua.es/assignatures/p2/downloads/1314/teoria/es/t3-files-es-ver.pdf

http://www.chuidiang.org/clinux/ficheros/fichero-binario.php

http://www.programacionenc.net/index.php?option=com_content&view=article&id=69:manejo-de-archivos-en-c&catid=37:programacion-cc&Itemid=55

https://www.visual-paradigm.com/support/documents/vpuserguide/94_umlmodeling.html

https://moodle.uco.es/m1819/course/view.php?id=2230

https://es.slideshare.net/iscrquinter/03-administracion-de-requisitos