Universidad de Córdoba

Escuela Politécnica Superior

Grado en Ingeniería Informática

Ingeniería del Software

Práctica final:

Aplicación para la gestión de alumnos

Antonio Tomás Romero Cortés - p32rocoa@uco.es

Francisco Sáenz Ruiz - p32saruf@uco.es

Curso académico 2018/19

Índice

Análisis de requisitos

1. Definición del problema

El problema principal consiste en realizar un programa que pudiera ser usado como agenda por un departamento o un conjunto de profesores, mediante la cual, una serie de alumnos pudieran ser almacenados en el sistema de una forma determinada.

De esta información se puede extraer que los actores implicados en el programa serán Alumnos y Profesores. Dentro de este último grupo se distinguen dos tipos: profesor coordinador y profesor ayudante. Cualquier profesor puede tener acceso a los alumnos, puede añadir nuevos alumnos, editarlos, borrarlos y modificarlos; además de poder guardar y cargar la base de datos que usa el programa. El coordinador, además de gozar de los privilegios anteriores, podrá cargar y guardar copias de seguridad, y borrar las cuentas de otros profesores. Por lo tanto también hará falta que cada uno de ellos tenga una cuenta de acceso al programa que los distinga por privilegios. Cada uno tendrá además un correo único asignado y una contraseña de acceso.

Los alumnos deberán ser almacenados con un DNI que los identifique y un correo electrónico de contacto, únicos para cada uno, además de su correspondiente nombre y apellidos, teléfono de contacto, domicilio, fecha de nacimiento, curso más alto matriculado, y el grupo del que forma parte. Si este alumno es líder o no del grupo, también deberá ser guardado. El grupo y el rol dentro de este, no serán obligatorios, al contrario que el resto.

Los alumnos podrán ser mostrados, modificados o borrados siendo identificados por DNI o apellidos o, por el contrario, se podrá mostrar un conjunto o la totalidad de ellos. Si se muestra la lista entera de alumnos, se podrá decidir el orden en el que se muestra (alfabético por apellidos, por curso o por el DNI). De la misma forma, se podrá decidir mostrar a los alumnos que componen un grupo concreto, mostrando a su líder. Todo esto en un archivo Markdown o HTML.

El máximo de alumnos permitidos en el programa será de 150.

El programa debe funcionar en el sistema operativo GNU/Linux, las bases de datos deben ser guardadas en archivos binarios para evitar un fácil acceso desde el exterior del programa, y estos deben tener la posibilidad de guardarse de forma automática o manual.

2. Extracción de requisitos

A continuación, se indicarán las partes interesadas o actores del sistema, los datos que se necesitan guardar (en este caso, alumnos y profesores), y tanto los requisitos funcionales como los no funcionales. Además, en el caso de los requisitos funcionales, se señalará la prioridad de estos.

Partes interesadas:

- Profesores

Datos a almacenar de los alumnos:

- DNI
- E-mail corporativo
- Nombre
- Apellidos
- Fecha de nacimiento
- Domicilio
- Teléfono
- Curso más alto matriculado
- Grupo al que pertenece
- Rol dentro del grupo (Representante de este o no)

Datos a almacenar de los profesores:

- E-mail corporativo
- Contraseña dentro de la aplicación
- Rol o tipo de profesor (Coordinador o ayudante)

Requisitos funcionales:

- **RF-1:** La aplicación permitirá almacenar alumnos.
- **RF-2:** Se podrán insertar nuevos alumnos.
    - **RF-2.1:** No se permitirá la inserción de un alumno con un DNI o e-mail ya existente.
- **RF-3:** Será posible buscar a un alumno mediante su DNI o apellidos.
    - **RF-3.1:** Si existe más de un alumno con esos apellidos, se pedirá el DNI.
- **RF-4:** Se permitirá la modificación de los datos de los alumnos, especificando su respectivo DNI o apellidos previamente.
    - **RF-4.1:** Si existe más de un alumno con esos apellidos, se pedirá el DNI.
- **RF-5:** Será posible el borrado de alumnos mediante su DNI o apellidos.
    - **RF-5.1:** Si existe más de un alumno con esos apellidos, se pedirá el DNI.

- **RF-6:** Se podrá listar a todos los alumnos.
    - **RF-6.1:** Será posible listarlos ordenados alfabéticamente por nombre y apellidos. (Orden ascendente o descendente)
    - **RF-6.2:** Será posible listarlos ordenados numéricamente por su DNI. (Orden ascendente o descendente)
    - **RF-6.3:** Podrán ser listados ordenados por el curso más alto en el que estén matriculados. (Orden ascendente o descendente)
- **RF-7:** Se podrá mostrar a un único alumno mediante sus apellidos o DNI.
    - **RF-7.1:** Si existe más de un alumno con esos apellidos, se pedirá el DNI.
- **RF-8:** Será posible la búsqueda de un conjunto de alumnos mediante su grupo.
- **RF-9:** Se podrá mostrar a todos los alumnos pertenecientes a un grupo.
- **RF-10:** Habrá una opción que permita borrar a todos los alumnos almacenados.
- **RF-11:** Los profesores podrán guardar cambios o cargar la base de datos.
    - **RF-11.1:** Sólo los coordinadores podrán guardar copias de seguridad de la base de datos.
- **RF-12:** Los profesores podrán registrarse como usuarios.
    - **RF-12.1:** No podrá registrarse un profesor que ya esté registrado.
- **RF-13:** El coordinador podrá eliminar profesores de la base de datos.
- **RF-14:** Los profesores registrados podrán iniciar o cerrar sesión en la aplicación.

Requisitos no funcionales:

- **RNF-1:** La aplicación permitirá almacenar un máximo de 150 alumnos.
- **RNF-2:** Todos los campos a introducir son obligatorios excepto el grupo y el rol en este.
- **RNF-3:** Los datos podrán ser recuperados en el caso de que algo interrumpa la ejecución de la aplicación.
- **RNF-4:** Los datos se guardarán en un fichero binario.
- **RNF-5**: El guardado de los datos en el fichero podrá ser manual o automático.
- **RNF-6:** Los grupos formados por el alumnado deben tener, estrictamente, un líder.
    - **RNF-6.1:** Si alguna acción o función hace que un grupo se quede sin líder, otro alumno tomará dicho rol antes de que esta termine.
- **RNF-7:** La aplicación funcionará bajo una interfaz por línea de comandos u órdenes (CLI).
- **RNF-8:** Las impresiones se realizarán en archivos Markdown o HTML.
- **RNF-9:** La aplicación debe funcionar en el sistema operativo GNU/Linux.
- **RNF-10:** Las cuentas de los profesores serán guardadas en un fichero binario.

Priorización de los requisitos funcionales:

- Prioridad 1:
    - **RF-1:** La aplicación permitirá almacenar alumnos.
    - **RF-2:** Se podrán insertar nuevos alumnos.
    - **RF-10:** Habrá una opción que permita borrar a todos los alumnos almacenados.
- Prioridad 2:
    - **RF-3:** Será posible buscar a un alumno mediante su DNI o apellidos.
    - **RF-6:** Se podrá listar a todos los alumnos.
- Prioridad 3:
    - **RF-5:** Será posible el borrado de alumnos mediante su DNI o apellidos.
    - **RF-7:** Se podrá mostrar a un único alumno mediante sus apellidos o DNI.
- Prioridad 4:
    - **RF-4:** Se permitirá la modificación de los datos de los alumnos, especificando su respectivo DNI o apellidos previamente.
    - **RF-8:** Será posible la búsqueda de un conjunto de alumnos mediante su grupo.
- Prioridad 5:
    - **RF-9:** Se podrá mostrar a todos los alumnos pertenecientes a un grupo.
    - **RF-12:** Los profesores podrán registrarse como usuarios.
    - **RF-14:** Los profesores registrados podrán iniciar o cerrar sesión en la aplicación.
- Prioridad 6:
    - **RF-11:** Los profesores podrán guardar cambios o cargar la base de datos.
    - **RF-13:** El coordinador podrá eliminar profesores de la base de datos.

3. Historias de usuario

Las siguientes historias de usuario han sido realizadas mediante la recogida de información y directrices recibidas por parte del cliente interesado. A continuación, se procederá a exponerlas.

Mostrar alumno

(ANVERSO)
ID: 001 Mostrar alumno **Prioridad:** 2
Quiero que el programa me muestre los datos de un alumno.

(REVERSO)
- Quiero poder visualizar todos los datos del alumno.
- Quiero poder buscarlo por DNI o Apellidos.
- Los detalles del alumno serán impresos en un archivo con formato Markdown.

Modificar alumno

(ANVERSO)
ID: 002 Modificar alumno **Prioridad:** 4
Quiero poder modificar los datos de un alumno.

(REVERSO)
- Quiero poder cambiar los campos de información de un alumno.
- Quiero poder concretar el alumno al que modificar aportando el DNI o los apellidos.

Listar alumnos

(ANVERSO)
ID: 003 Listar alumnos **Prioridad:** 2
Quiero poder ver a todos los alumnos por orden.

(REVERSO)
- Quiero poder ver a los alumnos por orden alfabético de su nombre o sus apellidos.
- Quiero poder ordenarlos por su DNI.
- Quiero poder ordenarlos por curso más alto matriculado.

Borrar alumno

(ANVERSO)
ID: 004 Borrar alumno **Prioridad:** 3
Quiero poder borrar a un alumno de la lista.

(REVERSO)
- Quiero poder borrar a un alumno conociendo su DNI.
- Quiero poder borrar a un alumno por sus apellidos, si son únicos en la lista.
- Si al borrar al alumno por apellidos, estos no son únicos, resolver el conflicto pidiendo el DNI.

Insertar alumno

(ANVERSO)
ID: 005 Insertar alumno **Prioridad:** 1
Quiero poder añadir a nuevos alumnos a la lista.

(REVERSO)
- Quiero poder insertar alumnos que no estuvieran previamente.
- Los atributos DNI y email deben ser únicos para cada uno de ellos, por lo que un alumno nuevo no debe tener un DNI ni un email ya introducido en la base de datos.
- Todos los campos deben ser obligatorios excepto el grupo al que pertenece y su rol en este.
- La cantidad de alumnos no puede ser mayor a 150.

Mostrar grupo

(ANVERSO)
ID: 006 Mostrar grupo **Prioridad:** 5
Quiero poder ver a todos los alumnos de un grupo.

(REVERSO)
- Quiero poder listar los miembros de un grupo aportando el número de este.
- Quiero poder ver quién es el líder.

Guardar base de datos

(ANVERSO)
ID: 007 Guardar BD **Prioridad:** 3
Quiero poder cerrar el programa y no perder los cambios realizados.

(REVERSO)
- Quiero poder guardar los cambios de forma manual.
- La información se almacenará en un fichero binario.

Reiniciar lista de alumnos

(ANVERSO)
ID: 008 Reiniciar alumnos **Prioridad:** 1
Quiero poder borrar a todos los alumnos de una vez.

(REVERSO)
- Quiero poder eliminar todos los datos almacenados con una sola acción.

Cargar base de datos

(ANVERSO)
ID: 009 Cargar BD **Prioridad:** 3
Quiero poder cargar la base de datos en el programa.

(REVERSO)
- Quiero poder cargar el archivo de BD en el programa para usar los datos.

Guardar copia de seguridad

(ANVERSO)
ID: 010 Guardar copia de seguridad **Prioridad:** 6
Quiero poder crear una copia de seguridad de la base de datos del programa.

(REVERSO)
- Quiero que se puedan guardar copias de seguridad de los datos, aparte del fichero de la base de datos.
- Quiero que solo pueda hacerlo un profesor coordinador.

Cargar copia de seguridad

(ANVERSO)
ID: 011 Cargar copia de seguridad **Prioridad:** 6
Quiero poder cargar, en el programa, una copia de seguridad de la base de datos.

(REVERSO)
- Quiero poder cargar un archivo de copia de seguridad de la base de datos, previamente guardado.
- Quiero que solo pueda hacerlo un profesor coordinador.

Registrar profesor

(ANVERSO)
ID: 012 Registrar profesor **Prioridad:** 5
Quiero poder registrar cuentas de profesores.

(REVERSO)
- Quiero poder crear nuevas cuentas de profesores.
- Quiero que solo se puedan registrar profesores ayudantes cuando ya haya un coordinador registrado.

Borrar profesor

(ANVERSO)
ID: 013 Borrar profesor **Prioridad:** 6
Quiero poder borrar cuentas de profesor.

(REVERSO)
- Quiero poder borrar cuentas registradas en el sistema.
- Quiero que solo pueda realizar esta acción el coordinador.

Iniciar sesión

(ANVERSO)
ID: 014 Iniciar sesión **Prioridad:** 5
Quiero poder iniciar sesión con una cuenta registrada.

(REVERSO)
- Quiero poder iniciar sesión desde el programa para acceder a este.
- Quiero que no sea accesible sin iniciar sesión.

Cerrar sesión

(ANVERSO)
ID: 015 Cerrar sesión **Prioridad:** 5
Quiero poder cerrar sesión con una cuenta iniciada.

(REVERSO)
- Quiero poder cerrar sesión desde el programa.
- Quiero que los datos no sean accesibles una vez cerrada sesión.

4. Casos de uso

Estudiando y analizando tanto las historias de usuario como los requisitos funcionales y no funcionales, obtenidos anteriormente, se consigue llegar a la conclusión de que existirán los siguientes casos de uso, que se expondrán a continuación, para el proyecto. Finalmente, se mostrará un diagrama con todos los casos de uso mencionados. De esta forma, se podrán observar todos los casos de uso de una manera más globalizada y se percibirán mejor las dependencias que existen entre ellos.

Mostrar alumno

ID: 001
Breve descripción: Muestra a un alumno de la lista.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:

- El alumno debe existir en el sistema.
- El profesor debe conocer, al menos, uno de los datos identificativos del alumno (DNI, apellidos).

Flujo principal:

1. El caso de uso comienza cuando el sistema necesita mostrar un alumno.
2. El sistema recoge los datos que el profesor introduce para encontrar al alumno.

Postcondiciones:

- Se encuentra al alumno y se imprimen los datos en un archivo Markdown o HTML.

Flujos alternativos:

2.a. Si el alumno no existe, se muestra un mensaje de error.
2.b. Si los apellidos buscados coinciden entre varios alumnos, se pedirá el DNI.

Modificar alumno

ID: 002
Breve descripción: Permite modificar los datos de un alumno.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:
- El alumno debe existir en el sistema.
- El profesor debe conocer, al menos, uno de los datos identificativos del alumno (DNI, apellidos).

Flujo principal:
1. El caso de uso comienza cuando el sistema necesita modificar los datos de un alumno.
2. El sistema recoge los datos que el profesor introduce para encontrar al alumno.
3. El sistema muestra los datos que pueden ser modificados y los dispone al cambio.

Postcondiciones:
- El sistema guarda los nuevos datos del alumno.

Flujos alternativos:

2.a. Si el alumno no existe, se muestra un mensaje de error.
2.b. Si los apellidos buscados coinciden entre varios alumnos, se pedirá el DNI.
3.a. Cuando la modificación intenta realizar algo no permitido, se muestra un mensaje de error.

Listar alumnos

ID: 003
Breve descripción: Muestra toda la lista de alumnos.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:
- Debe existir al menos un alumno en el sistema.

Flujo principal:
1. El caso comienza cuando el sistema necesita mostrar la lista de alumnos.
2. El sistema pide al profesor en qué orden los quiere mostrar.

Postcondiciones:
- Se ordenan los alumnos y se muestran en orden mediante un archivo Markdown o HTML.

Flujos alternativos:

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

Borrar alumno

ID: 004
Breve descripción: Borra a un alumno de la lista.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:
- El alumno debe existir en el sistema.
- El profesor debe conocer uno de los datos identificativos del alumno (DNI, apellidos).

Flujo principal:
3. El caso de uso comienza cuando el sistema necesita borrar al alumno que el profesor especifique.
4. El sistema recoge los datos que el profesor introduce para encontrar al alumno y borrarlo.

Postcondiciones:
- Se encuentra al alumno y se borra del sistema.

Flujos alternativos:

2.a. Si el alumno no existe, se muestra un mensaje de error.
2.b. Si los apellidos buscados coinciden entre varios alumnos, se pedirá el DNI.

Insertar alumno

ID: 005
Breve descripción: Inserta a un alumno nuevo en la lista.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:
- El alumno no debe existir en el sistema.
- El profesor debe conocer los datos obligatorios para insertar un alumno nuevo.

Flujo principal:
1. El caso comienza cuando el sistema necesita insertar un nuevo alumno.
2. El sistema recoge los datos que el profesor introduce en una nueva entrada.

Postcondiciones:
- Se inserta el nuevo alumno en la base de datos del programa.

Flujos alternativos:

1.a. Si el alumno ya se encuentra registrado o incumple alguna de las restricciones, se muestra un mensaje de error.

Mostrar grupo

ID: 006
Breve descripción: Muestra todos los datos de un grupo.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:

- Dicho grupo debe existir.
- Debe existir al menos un alumno dentro del grupo a mostrar.

Flujo principal:

1. El caso de uso comienza cuando el sistema necesita mostrar los datos de un grupo.
2. El profesor elige qué grupo mostrar.

Postcondiciones:

- Se muestran los alumnos del grupo y quién es el líder mediante un archivo Markdown o HTML.

Flujos alternativos:

2.a. Si el sistema no encuentra el grupo indicado, se muestra un mensaje de error.

Guardar BD

ID: 007
Breve descripción: Guarda la lista de alumnos en la base de datos.

Actores principales: Profesor

Precondiciones:

- Debe existir algún cambio en la base de datos.

Flujo principal:

1. El caso de uso comienza cuando el profesor activa la acción de guardar.

Postcondiciones:

- Los cambios se guardan en la base de datos.

Flujos alternativos:

1.a. Si el sistema encuentra algún error relativo al fichero de la base de datos, se muestra un mensaje de error.

Reiniciar alumnos

ID: 008
Breve descripción: Elimina a todos los alumnos de la base de datos del programa.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:
- Debe existir algún alumno en la lista.

Flujo principal:
1. El caso de uso comienza cuando el profesor activa la acción de borrar a todos los alumnos de la lista.
2. Se pide confirmación de la acción.

Postcondiciones:
- La lista de alumnos queda vacía.

Flujos alternativos:

2.a. El profesor ha denegado la confirmación y se aborta la operación.

Cargar BD

ID: 009
Breve descripción: Carga la base de datos del programa.

Actores principales: Profesor

Precondiciones:
- Debe existir el fichero de base de datos.

Flujo principal:
1. El caso de uso comienza cuando se pide cargar la base de datos mediante la interfaz.

Postcondiciones:
- Los datos se cargan en el programa y son accesibles por el usuario.

Flujos alternativos:

1.a. Si no existe fichero de base de datos, el programa lanza un mensaje de error.

Guardar copia de seguridad

ID: 010
Breve descripción: Guarda una copia de seguridad de la lista de alumnos.

Actores principales: Coordinador

Precondiciones:
- La lista de alumnos no debería estar vacía.
- La acción la debe activar un profesor coordinador y no cualquier profesor.

Flujo principal:
1. El caso de uso comienza cuando se pide crear una copia de seguridad de la lista de alumnos.
2. Crea un fichero con el nombre, el cual no exista previamente, y ruta elegida por el coordinador.

Postcondiciones:
- Los datos de la lista de alumnos se guardan en dicho fichero.

Flujos alternativos:

2.a. Si ya existe un fichero con ese nombre, el programa lanza un mensaje de error.

Cargar copia de seguridad

ID: 011
Breve descripción: Carga una copia de seguridad de la lista de alumnos.

Actores principales: Coordinador

Precondiciones:
- Debe existir una copia de seguridad que cargar.
- La acción la debe activar un profesor coordinador y no cualquier profesor.

Flujo principal:
1. El caso de uso comienza cuando se pide cargar una copia de seguridad de la lista de alumnos.
2. El coordinador introduce el nombre del fichero que quiere cargar.

Postcondiciones:
- Los datos se cargan en el programa y son accesibles por el usuario.

Flujos alternativos:

2.a. Si no existe un fichero con ese nombre, el programa lanza un mensaje de error.

Registrar profesor

ID: 012
Breve descripción: Registra una nueva cuenta de profesor en el sistema.

Actores principales: Usuario

Precondiciones:
- No debe existir un usuario con el mismo nombre de cuenta.

Flujo principal:
1. El caso de uso comienza cuando se llama a la acción de registrarse.
2. El usuario elige su usuario y contraseña.

Postcondiciones:
- El nuevo usuario se guarda en el fichero de registro.

Flujos alternativos:

2.a. Si el nombre de usuario ya existe, se muestra un mensaje de error.

Borrar profesor

ID: 013
Breve descripción: Borra una cuenta de profesor del fichero de registro.

Actores principales: Coordinador
Actores secundarios: Profesor

Precondiciones:
- Solo el coordinador puede borrar profesores.

Flujo principal:
1. El caso de uso comienza cuando se llama a la acción de borrar profesor.
2. El coordinador elige la cuenta a borrar.

Postcondiciones:
- La cuenta es eliminada del fichero de registro.

Flujos alternativos:

2.a. Si la cuenta indicada no existe, se muestra un mensaje de error.

Iniciar sesión

ID: 014
Breve descripción: Inicia la sesión de un profesor en el sistema.

Actores principales: Profesor

Precondiciones:
- Debe existir una cuenta con la que poder iniciar sesión.
- La contraseña debe coincidir con la del fichero de registro.

Flujo principal:
1. El caso de uso comienza cuando se llama a la acción de iniciar sesión.
2. El usuario introduce su nombre de cuenta y su contraseña.

Postcondiciones:
- El usuario obtiene acceso al sistema.

Flujos alternativos:

2.a. Si la cuenta indicada no existe, se muestra un mensaje de error.

Cerrar sesión

ID: 015
Breve descripción: Cierra la sesión actual de la aplicación.

Actores principales: Profesor

Precondiciones:
- Debe existir una cuenta iniciada.

Flujo principal:
1. El caso de uso comienza cuando se llama a la acción de cerrar sesión.

Postcondiciones:
- El usuario sale del sistema y pierde el acceso al programa hasta que inicie sesión de nuevo.

Flujos alternativos:

1.a. Si no hay una cuenta logueada, se muestra un mensaje de error.

Buscar alumno

ID: 016
Breve descripción: Busca a un alumno en la lista.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:
- Debe existir un alumno que buscar.

Flujo principal:
1. El caso de uso comienza cuando es necesario buscar a un alumno en concreto.
2. El profesor introduce el DNI o los apellidos del alumno a buscar.

Postcondiciones:
- El método devuelve un alumno de la lista, en caso de encontrarlo.

Flujos alternativos:

2.a. Si el alumno no existe, se muestra un mensaje de error.
2.b. Si los apellidos buscados coinciden entre varios alumnos, se pedirá el DNI.

Buscar profesor

ID: 017
Breve descripción: Busca a un profesor en el sistema.

Actores principales: Profesor
Actores secundarios: Profesor

Precondiciones:
- Debe existir, al menos, un profesor registrado.

Flujo principal:
1. El caso de uso comienza cuando es necesario buscar a un profesor en concreto.
2. Se busca al profesor por el correo electrónico indicado.

Postcondiciones:
- El método devuelve un profesor, en caso de encontrarlo.

Flujos alternativos:

2.a. Si el profesor no existe, se muestra un mensaje de error.

Buscar grupo

ID: 018
Breve descripción: Busca a los integrantes de un grupo de alumnos.

Actores principales: Profesor
Actores secundarios: Alumno

Precondiciones:

- Debe existir, al menos, un grupo de alumnos.

Flujo principal:

1. El caso de uso comienza cuando es necesario buscar a un grupo de alumnos.
2. Se busca al grupo por su número.

Postcondiciones:

- El método devuelve una lista de alumnos, en caso de encontrar al grupo.

Flujos alternativos:

2.a. Si el grupo no existe, se muestra un mensaje de error.

Diagrama de casos de uso

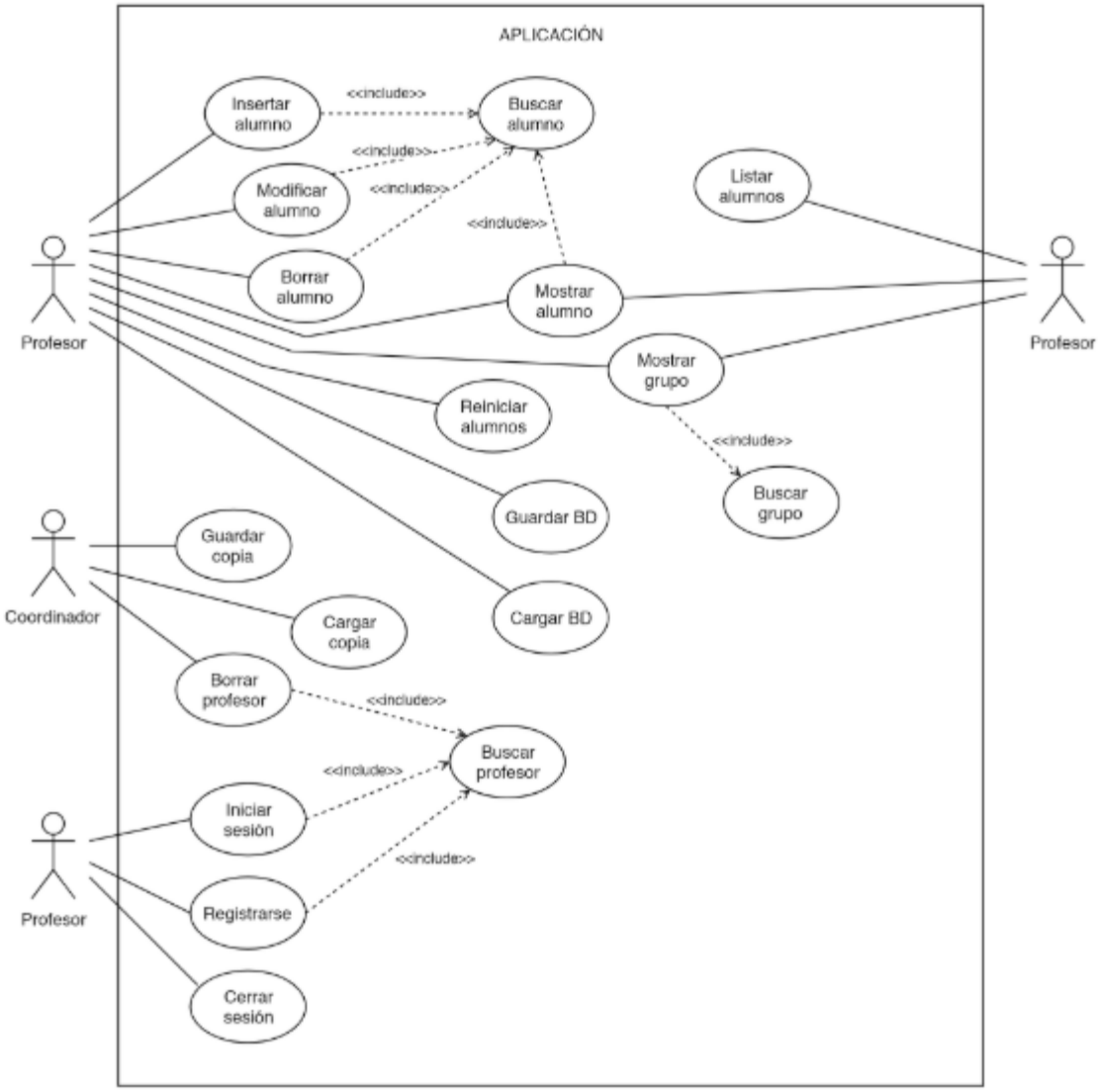

Figura 1. *Diagrama de casos de uso*.

En el diagrama anterior, se puede observar la conexión que tiene cada caso de uso con otro, o bien, con un actor. De esta forma, se aprecia mejor y de manera más globalizada, la estructura que tendrá la aplicación y las funcionalidades del sistema.

Los actores "*Profesor*" se refiere tanto a los profesores coordinadores como a los ayudantes. Es decir, los casos de uso conectados a un actor "*Profesor*" podrán ser utilizados por todos los profesores. Por el contrario, los casos de uso conectados a actores "*Coordinador*", solo podrán ser utilizados por un profesor coordinador.

Los métodos de búsqueda, al ser funciones internas, solo serán llamados por otros métodos del sistema.

Diseño del sistema

5. Diagrama de clases

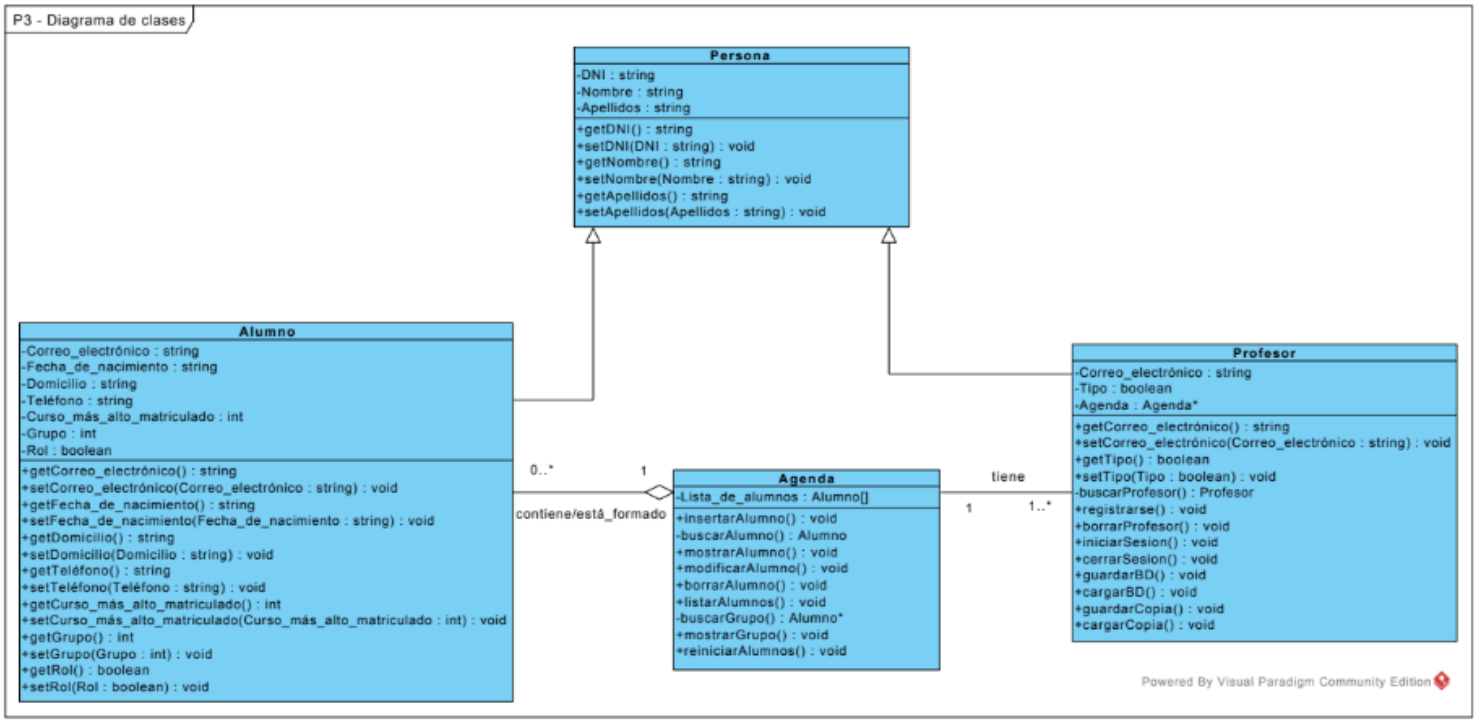

Figura 2. *Diagrama de clases.*

Existen cuatro clases diferentes en el diagrama:

- **Persona:**
  Engloba a las clases *Alumno* y *Profesor*. Además contiene la información básica de cada persona.

- **Alumno:**
  Es una especialización de la clase *Persona* y representa a un alumno de la asignatura.

- **Profesores:**
  Es una especialización de la clase *Persona* y representa a los profesores de la asignatura y, a su vez, a los usuarios de la aplicación.

- **Agenda:**
  Esta clase representa a la lista de alumnos que será gestionada por la aplicación.

6. Diagramas de secuencia

Registrar profesor:

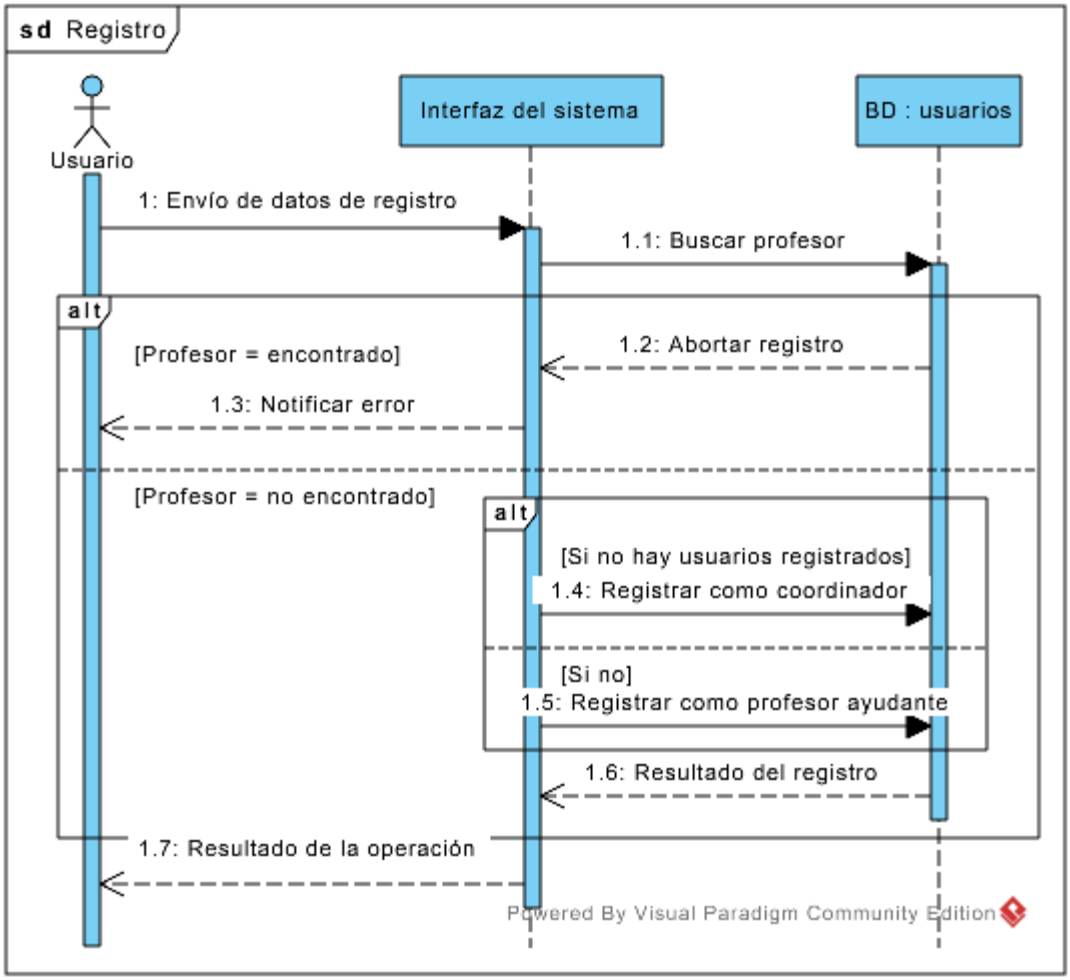

Figura 3. *Diagrama de registrar profesor.*

El usuario introduce los credenciales de la cuenta. Se comprueba que no exista ya esa cuenta. En caso de que exista, se aborta la operación y se comunica el error. Si el fichero de registro está vacío, el primer registro será como profesor coordinador. En cambio, si ya existen usuarios registrados, el registro será como profesor ayudante.

Iniciar sesión:

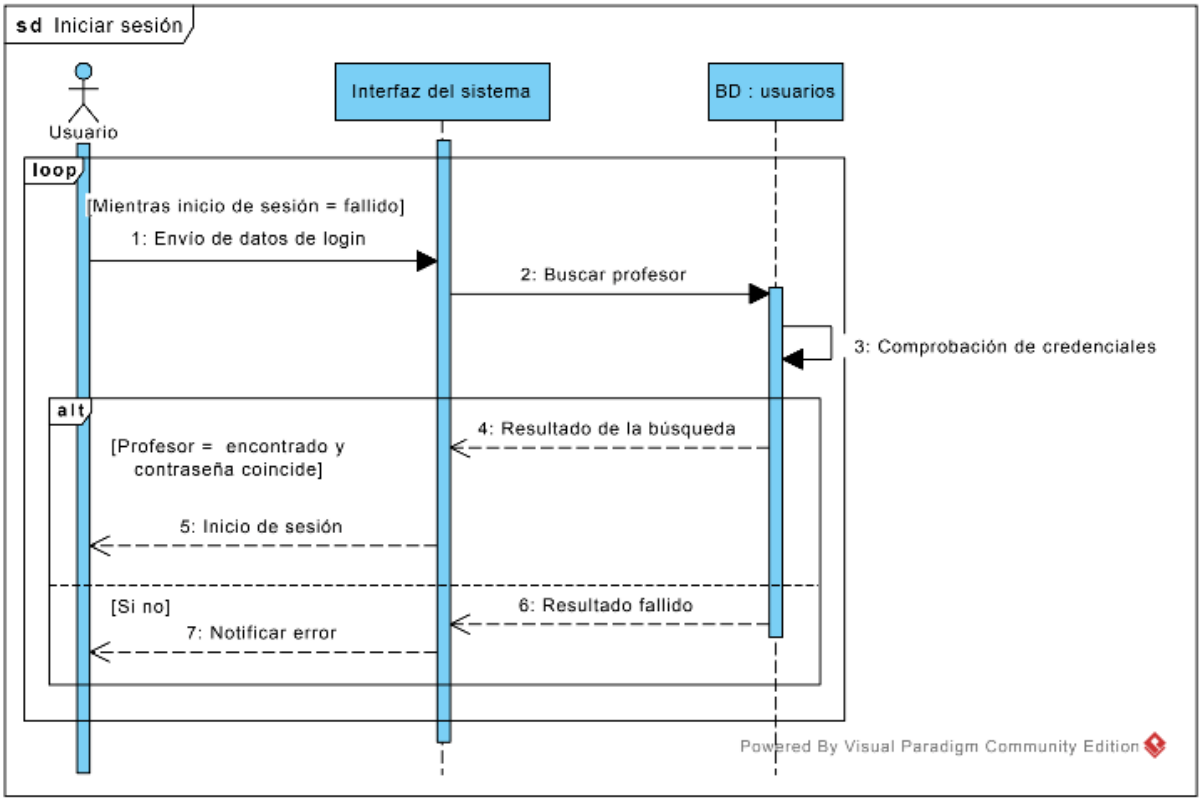

Figura 4. *Diagrama de iniciar sesión.*

El usuario introduce el correo electrónico y la clave de su cuenta. El sistema busca esa cuenta en el registro de usuarios y comprueba que los credenciales coincidan. En caso de que todo esté correcto, el sistema iniciará la sesión de esa cuenta y le permitirá acceder al menú principal de la aplicación. Sin embargo, si los datos introducidos no coinciden con ninguna cuenta del registro, se repetirá la operación hasta que se introduzcan unos datos válidos.

Cerrar sesión:

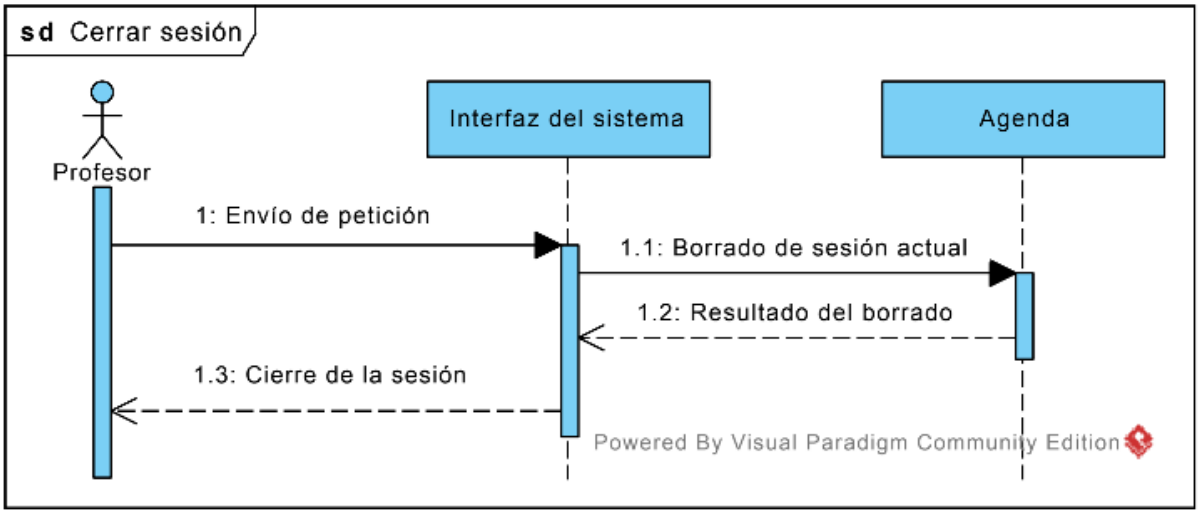

Figura 5. *Diagrama de cerrar sesión.*

En este caso, el usuario envía la petición de cerrar sesión, el sistema borra los datos actuales que no hayan sido guardados, cierra la sesión actual y se vuelve al menú de *Login*.

Una posible mejora que se podría llevar a cabo, sería preguntar si el usuario quiere guardar los cambios realizados en la lista de alumnos.

Borrar profesor:

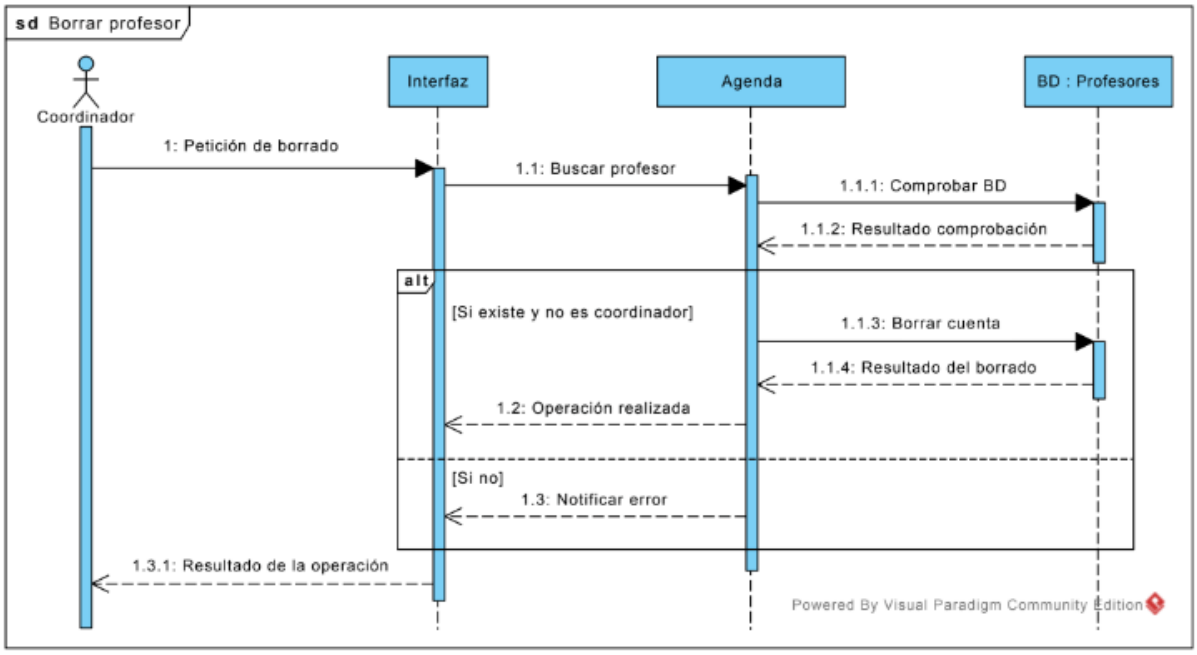

Figura 6. *Diagrama de borrar profesor.*

La función solo puede ser llamada por un profesor coordinador, el cual introduce el correo del profesor al que quiere borrar del fichero de registro de cuentas de usuario. Se comprueba la cuenta indicada exista. Si no existe o es la cuenta de un profesor coordinador, se abortará la operación y se notificará el error. Por otro lado, si la cuenta existe y pertenece a un profesor ayudante, se procederá al borrado de esa cuenta.

Insertar alumno:

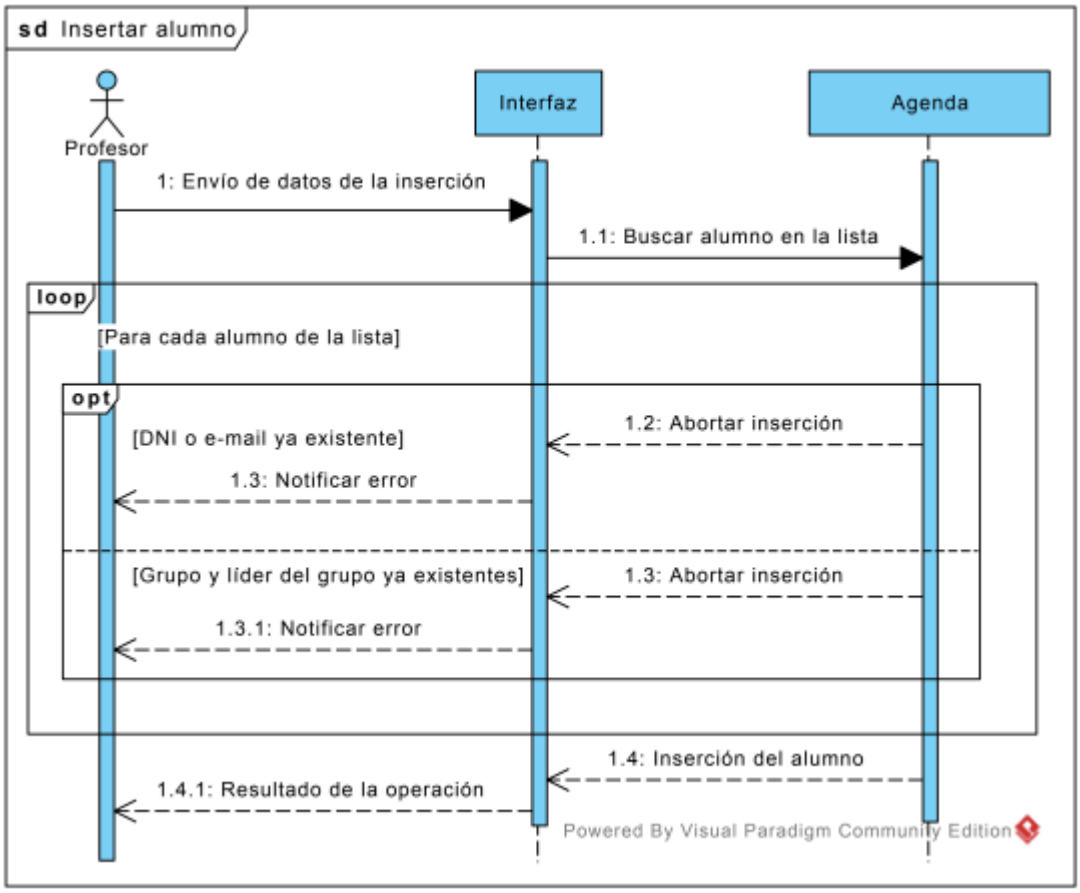

Figura 7. *Diagrama de insertar alumno.*

El sistema pedirá al usuario los datos del alumno nuevo, el cual irá introduciéndolos poco a poco. Una vez se hayan introducido todo los datos, el sistema comprueba que el DNI o correo electrónico introducidos, no existan previamente. En caso de que alguno de los dos exista, se abortará la operación de inserción y se notificará al usuario.

En caso de que se intente insertar un líder de grupo, se comprueba que el grupo indicado no tenga ya un líder. Si ya existe un líder, se abortará la operación y se notificará al usuario.

Por último, si todo va bien, el nuevo alumno se añadirá a la lista de alumnos de la aplicación. Aunque en el diagrama no se ve reflejado (intentado hacer el método algo más genérico), la aplicación no permitirá insertar más de 150 alumnos en la lista.

Modificar alumno:

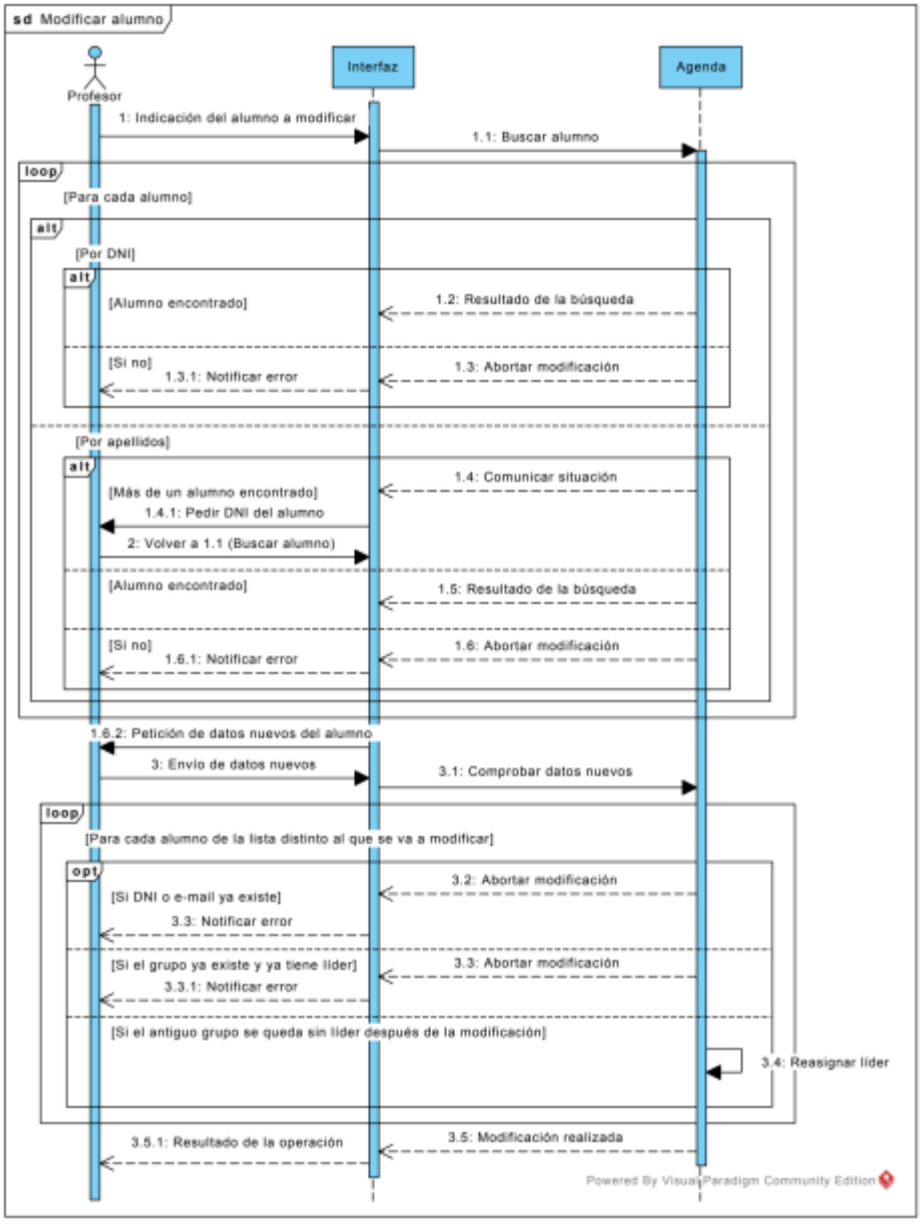

Figura 8. *Diagrama de modificar alumno.*

El usuario introduce el DNI o apellidos del alumno a modificar. Se comprueban los datos y, si no existen, se aborta la operación de modificación. En caso de buscar por apellidos y exista más de un alumno con esos apellidos, se le pedirá al usuario que introduzca el DNI del alumno.

En caso de que todo vaya bien, el sistema pedirá los nuevos datos del alumno a modificar. Se hacen las mismas comprobaciones que en apartado de *insertar alumno* y, además, se comprueba que no se deje un grupo sin líder, después de la modificación. En caso de que ocurra esto último, se asignará un nuevo líder.

Si se pasan estas comprobaciones sin problemas, el alumno indicado será guardado en la lista con los nuevos datos.

Borrar alumno:

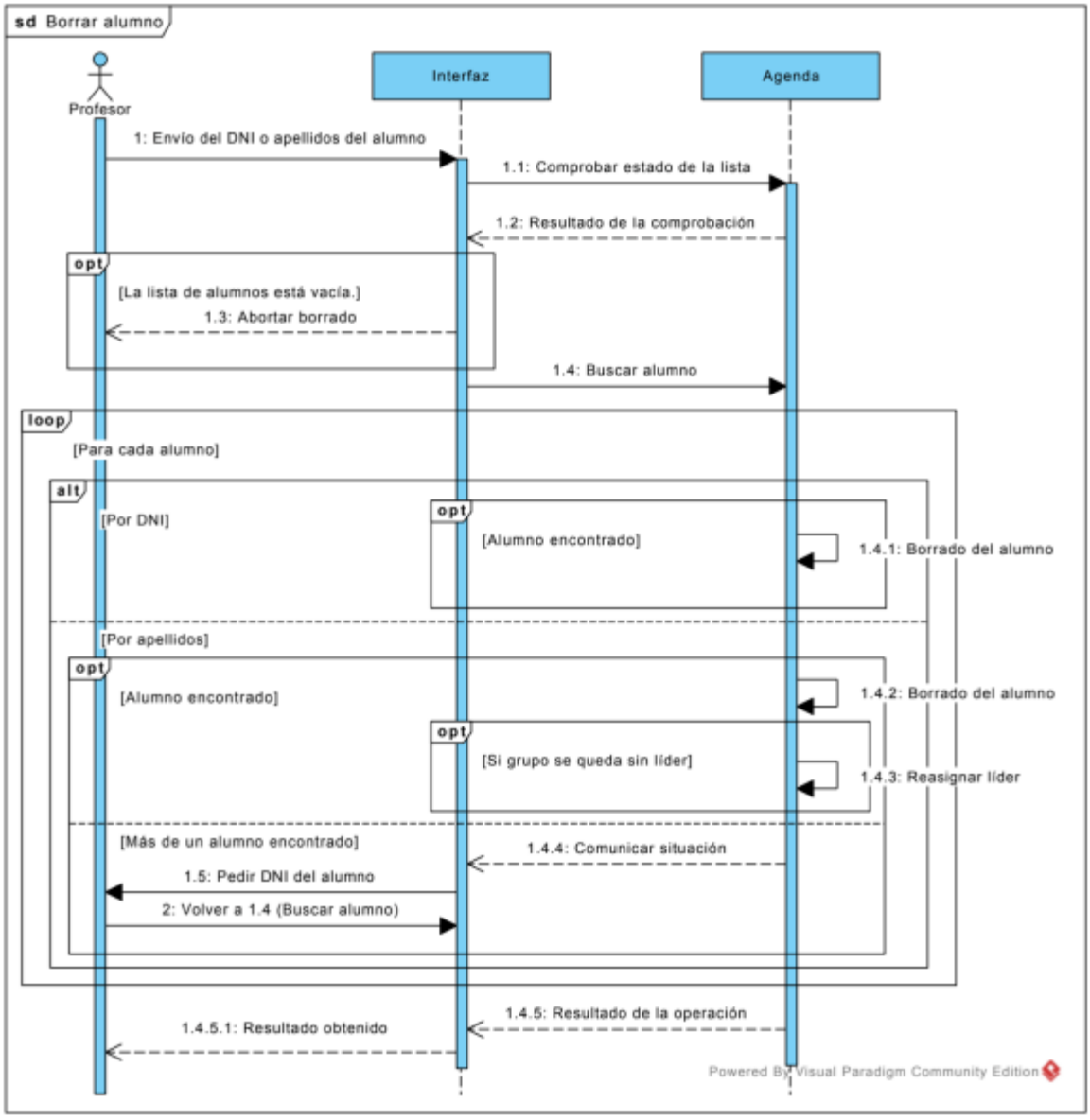

Figura 9. *Diagrama de borrar alumno.*

El usuario introduce el DNI o apellidos del alumno objetivo de ser borrado de la lista. Se comprueba que la lista no esté vacía. En caso de ser así, se abortará la operación de borrado y se notificará al usuario. Si no, se seguirá comprobando que el alumno indicado exista, y se aplicarán las mismas comprobaciones que se han visto anteriormente en el caso de *modificar alumno*.

Si todo ocurre normalmente o bien, de manera óptima, se pasarán todas las comprobaciones sin complicaciones y se borrará de la lista al alumno indicado.

Mostrar alumno:

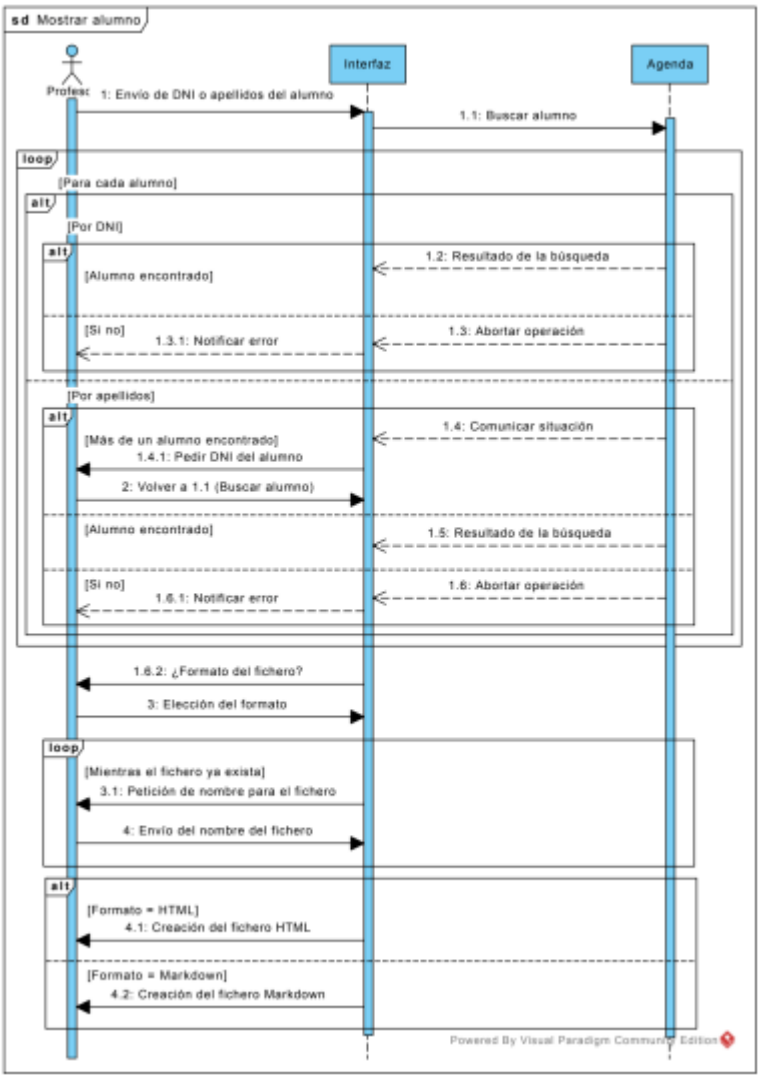

Figura 10. *Diagrama de mostrar alumno.*

El usuario introduce el DNI o apellidos del alumno que se quiere mostrar. Se hacen las comprobaciones para DNI y apellidos ya mencionadas anteriormente. En caso de que se pasen esas comprobaciones sin ningún problema, se pedirá el formato y el nombre del fichero en el que se va a volcar los datos del alumno elegido. Si ya existe ese fichero, se pedirá otro nombre hasta que no exista un fichero con ese nombre. Por último se creará el fichero, del formato elegido, con los datos del alumno indicado.

Mostrar grupo:

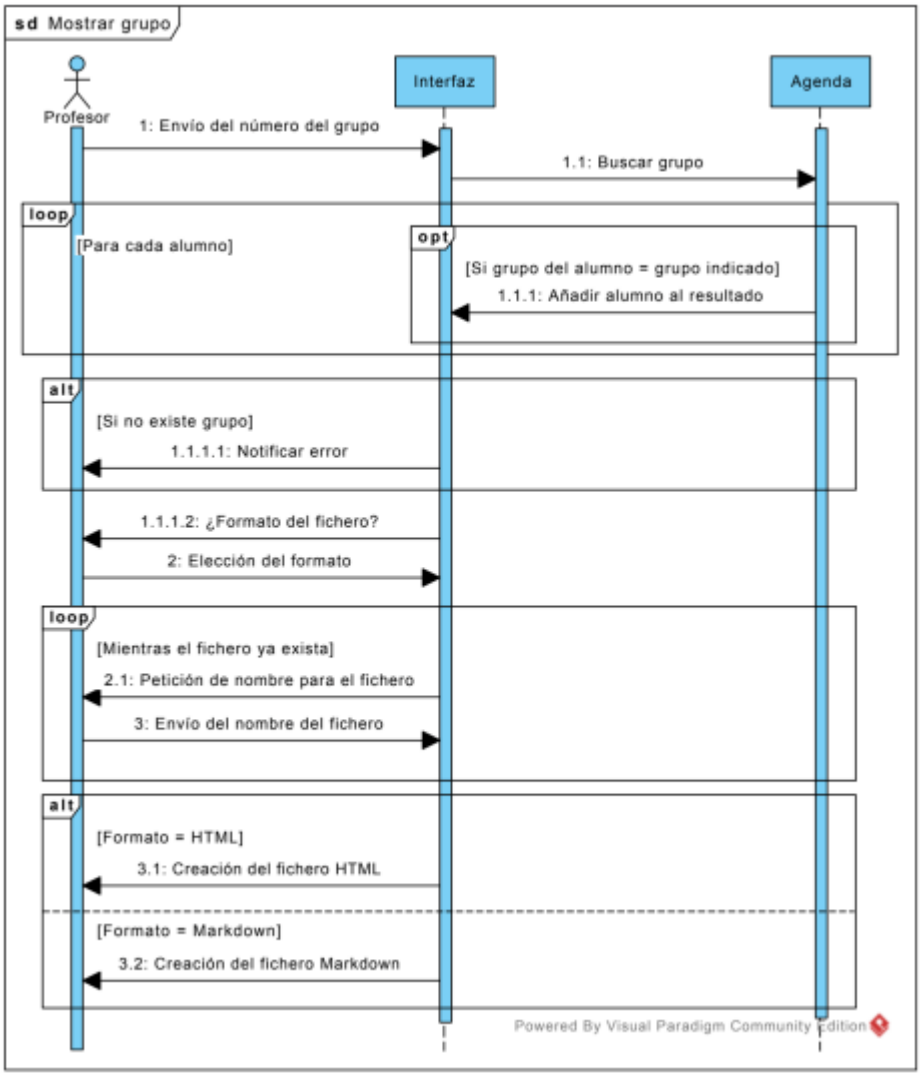

Figura 11. *Diagrama de mostrar grupo.*

El usuario indicará el número del grupo que quiere mostrar. A continuación, se seleccionará a cada alumno que tenga esté en ese grupo. Si no existe ningún alumno con ese grupo, se abortará la operación y se notificará al usuario.

Una vez se tenga a todos los alumnos pertenecientes al grupo, se pedirá el formato y el nombre del fichero. Se harán las comprobaciones que se han visto anteriormente para el fichero, y se creará el fichero, con el formato elegido, con los datos del grupo.

Listar alumnos:

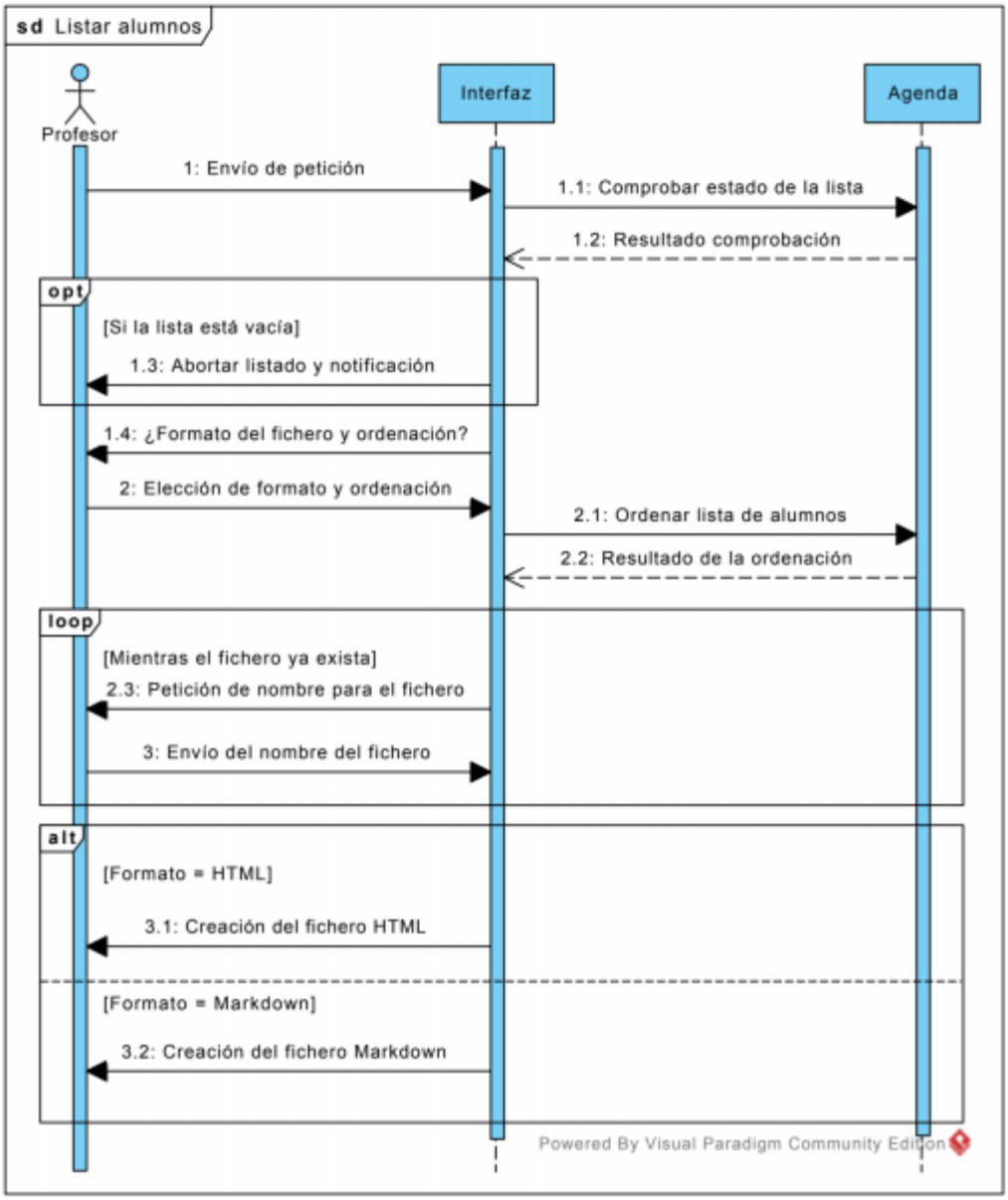

Figura 12. *Diagrama de listar alumnos.*

El usuario realiza la petición del listado de alumnos. El sistema comprueba que la lista no esté vacía. En caso negativo, le pedirá al usuario que indique el formato del fichero y el tipo de ordenación de la lista (Puede ser ordenada por DNI, apellidos y curso más alto matriculado, tanto ascendente como descendentemente).

Se ordena la lista y se comprueba que el fichero no exista ya. En caso de que exista se volverá a pedir de nuevo otro nombre.

Finalmente, se creará el fichero, con el formato indicado, con los datos de la lista de alumnos.

Reiniciar alumnos:

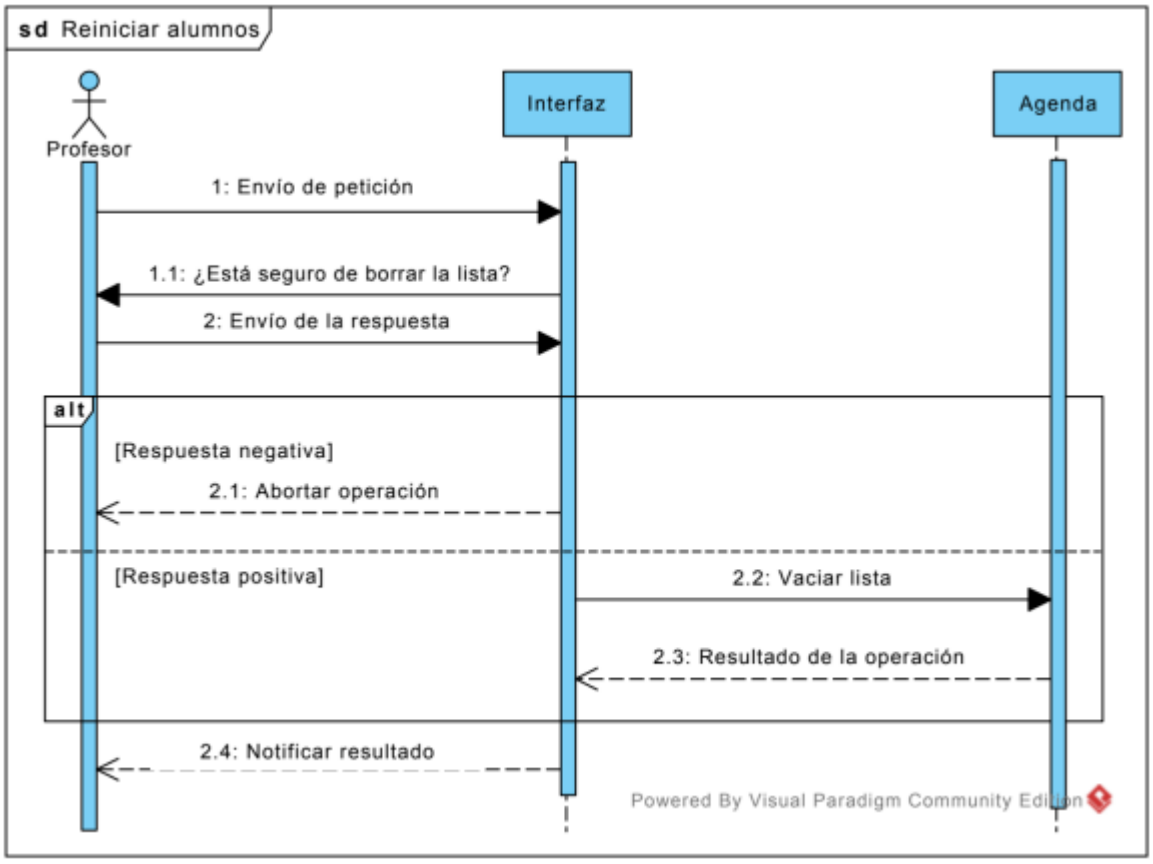

Figura 13. *Diagrama de reiniciar alumnos.*

El usuario envía la petición de borrar la lista de alumnos. El sistema pedirá que confirme que desea realizar esa operación. En caso de que el usuario niegue la confirmación, se abortará la operación de borrado. En caso de que se confirme, se vaciará la lista de alumnos y se notificará el resultado de la operación al usuario.

Una posible mejora que se podría añadir, sería: preguntar si el usuario quisiera cargar una lista de alumnos desde algún fichero, después del borrado de la lista.

Guardar base de datos:

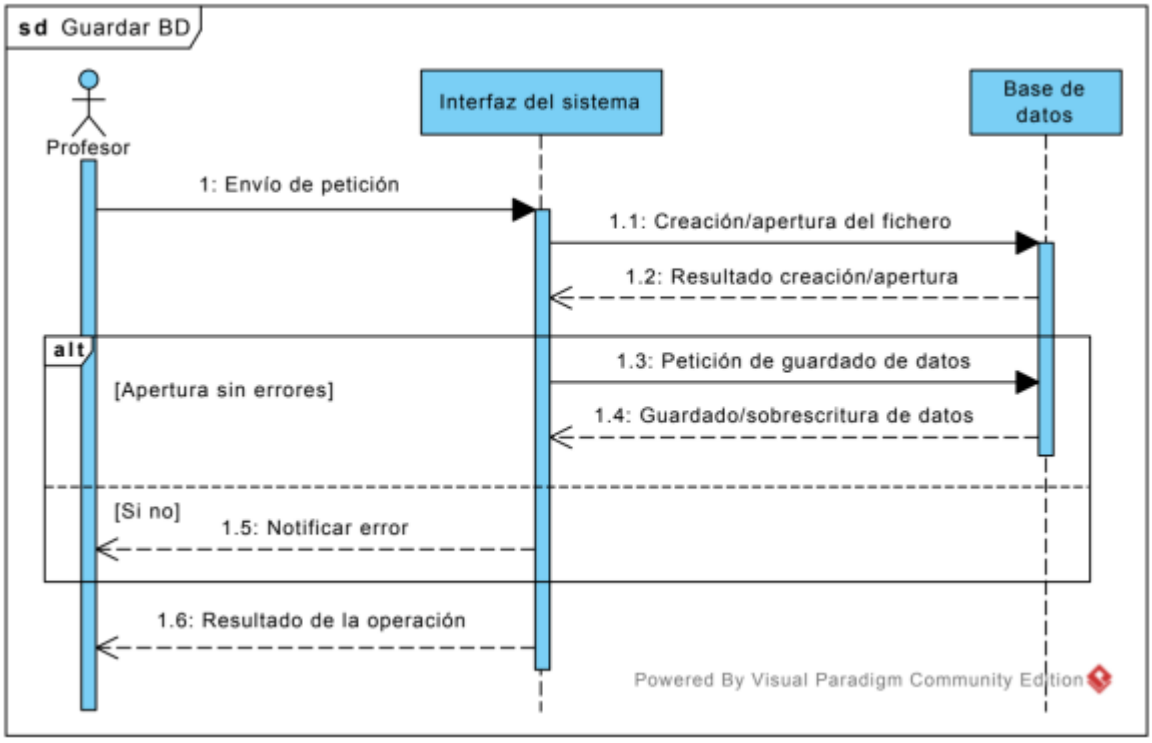

Figura 14. *Diagrama de guardar base de datos.*

El usuario hará la petición de guardado y el sistema creará o abrirá, dependiendo si ya está creado o no, el fichero de base de datos. Este fichero tendrá un nombre predefinido, por lo tanto, no se le pedirá nada más al usuario. En caso de que no haya errores al manipular el fichero, la lista de alumnos se guardará en la base de datos.

El caso del guardado de una copia de seguridad, que se verá más adelante, es similar a este, sin embargo, esa operación solo la podrá pedir un profesor coordinador y, también, tendrá que darle nombre a dicho fichero de copia.

Cargar base de datos:

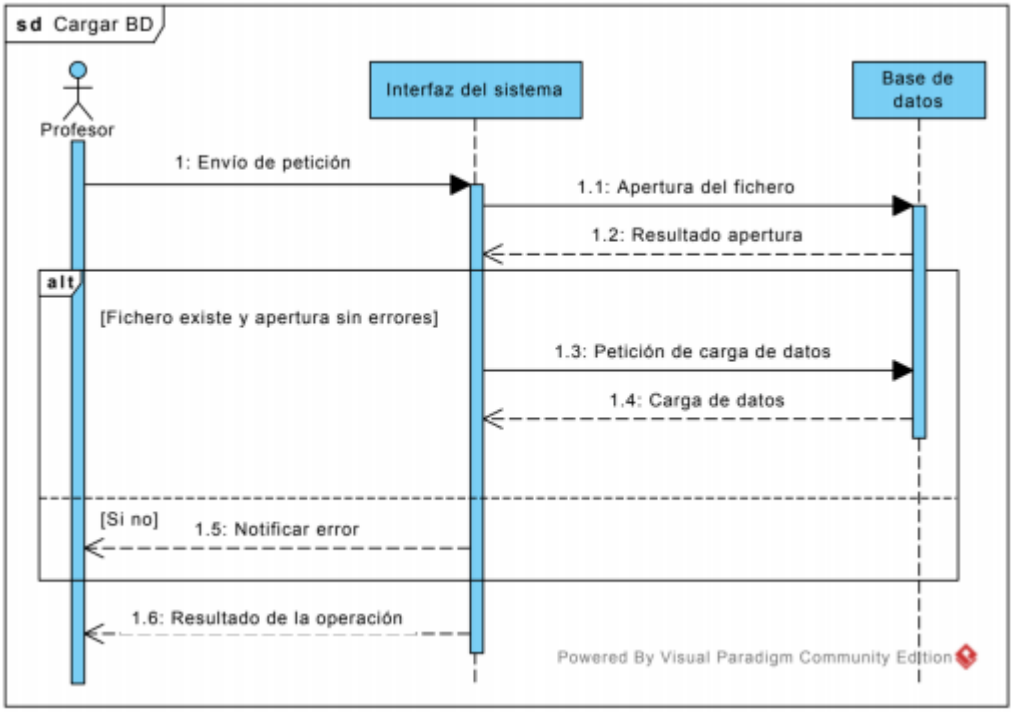

Figura 15. *Diagrama de cargar base de datos.*

El usuario hará la petición de cargar datos y el sistema abrirá, en el caso de que exista, el fichero de base de datos. Este fichero tendrá un nombre predefinido, por lo tanto, no se le pedirá nada más al usuario. En caso de que no haya errores al manipular el fichero, la lista de alumnos de la base de datos se cargará en la aplicación.

El caso de cargar una copia de seguridad, que se verá más adelante, es similar a este, sin embargo, esa operación solo la podrá pedir un profesor coordinador y, también, tendrá que indicar el nombre de dicho fichero de copia.

Guardar copia de seguridad:

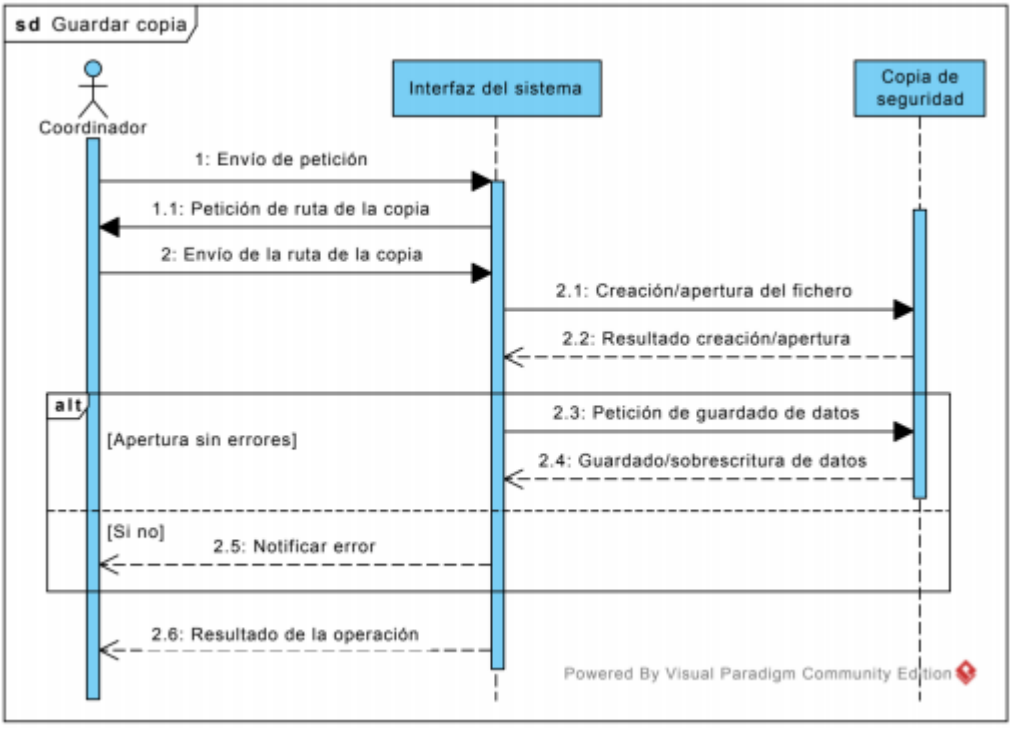

Figura 16. *Diagrama de guardar copia de seguridad.*

Cargar copia de seguridad:

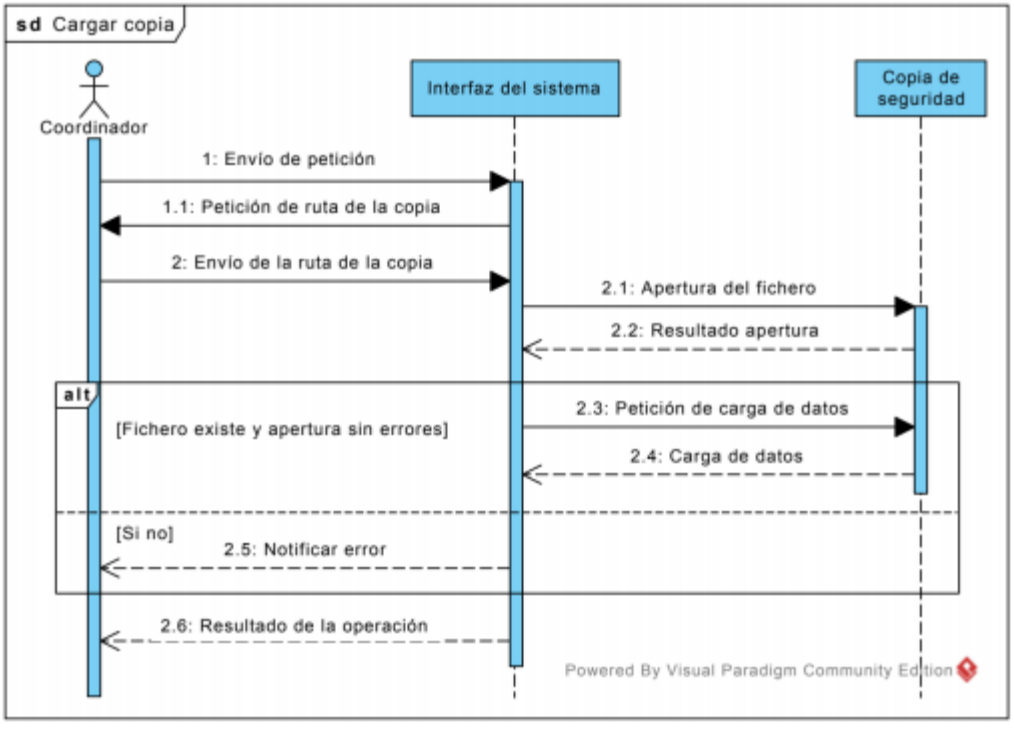

Figura 17. *Diagrama de cargar copia de seguridad.*

Implementación y pruebas

7. Metodología SCRUM

Para la planificación e implementación de esta aplicación se ha utilizado esta metodología ágil. Todo el proceso se ha llevado a cabo en dos *sprints.* A continuación, se expondrá el *Product Backlog*, que es el documento donde se recogen todas las funcionalidades a realizar y sus respectivos tiempos estimados y prioridad. Después, se mostrarán los dos *Sprint Backlogs*, que son documentos con las funcionalidades que se realizarán durante ese sprint. Por último, el Burndown Chart, que es el gráfico donde se recoge la evolución del proyecto durante los *sprints* y las diferentes reuniones del equipo de desarrollo.

Product Backlog

Historias de usuario:

ID: 008 | Reiniciar alumnos | Tiempo estimado: 1 | Prioridad: 0
ID: 005 | Insertar alumno | Tiempo estimado: 2 | Prioridad: 1
ID: 003 | Listar alumnos | Tiempo estimado: 3 | Prioridad: 1
ID: 001 | Mostrar alumno | Tiempo estimado: 2 | Prioridad: 2
ID: 002 | Modificar alumno | Tiempo estimado: 2 | Prioridad: 2
ID: 004 | Borrar alumno | Tiempo estimado: 1 | Prioridad: 2
ID: 006 | Mostrar grupo | Tiempo estimado: 2 | Prioridad: 4
ID: 007 | Guardar bd | Tiempo estimado: 3 | Prioridad: 5
ID: 009 | Cargar bd | Tiempo estimado: 3 | Prioridad: 5
ID: 010 | Guardar copia de seguridad | Tiempo estimado: 3 | Prioridad: 6
ID: 011 | Cargar copia de seguridad | Tiempo estimado: 3 | Prioridad: 6
ID: 012 | Registrar profesor | Tiempo estimado: 2 | Prioridad: 7
ID: 013 | Borrar profesor | Tiempo estimado: 1 | Prioridad: 8
ID: 014 | Iniciar sesión | Tiempo estimado: 1 | Prioridad: 9
ID: 015 | Cerrar sesión | Tiempo estimado: 1 | Prioridad: 9

Funciones internas auxiliares:

Buscar alumno | Tiempo estimado: 3 | Prioridad: 0
Buscar grupo | Tiempo estimado: 3 | Prioridad: 3
Buscar profesor | Tiempo estimado: 2 | Prioridad: 7

Primer Sprint Backlog

Francisco:

- Buscar alumno | Tiempo estimado: 3 | Prioridad: 0
- Insertar alumno | Tiempo estimado: 2 | Prioridad: 1
- Borrar alumno | Tiempo estimado: 1 | Prioridad: 2
- Modificar alumno | Tiempo estimado: 2 | Prioridad: 2
- Mostrar alumno | Tiempo estimado: 2 | Prioridad: 2

Antonio:

- Reiniciar alumnos | Tiempo estimado: 1 | Prioridad: 0
- Listar alumnos | Tiempo estimado: 3 | Prioridad: 1
- Guardar bd | Tiempo estimado: 3 | Prioridad: 5
- Cargar bd | Tiempo estimado: 3 | Prioridad: 5

Segundo Sprint Backlog

Francisco:

- Buscar grupo | Tiempo estimado: 3 | Prioridad: 3
- Mostrar grupo | Tiempo estimado: 2 | Prioridad: 4
- Buscar profesor | Tiempo estimado: 2 | Prioridad: 7
- Registrar profesor | Tiempo estimado: 2 | Prioridad: 7
- Borrar profesor | Tiempo estimado: 1 | Prioridad: 8

Antonio:

- Guardar copia de seguridad | Tiempo estimado: 3 | Prioridad: 6
- Cargar copia de seguridad | Tiempo estimado: 3 | Prioridad: 6
- Iniciar sesión | Tiempo estimado: 1 | Prioridad: 9
- Cerrar sesión | Tiempo estimado: 1 | Prioridad: 9

Burndown Backlog

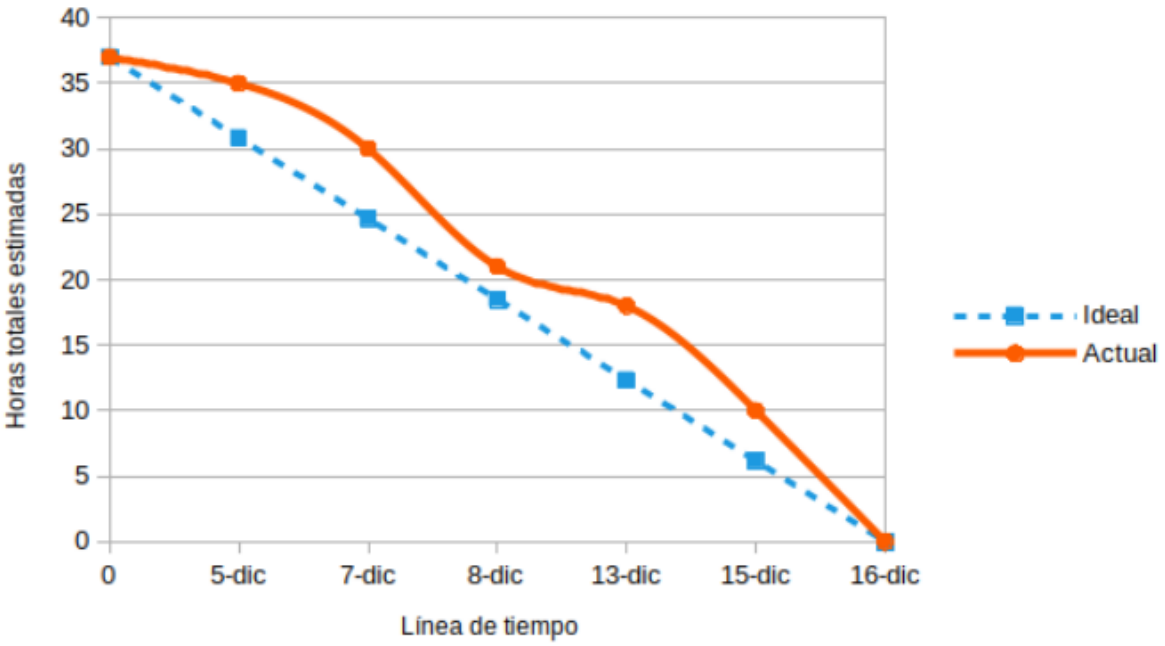

Figura 18. *Burndown chart del proyecto.*

Sin contar el punto de inicio del desarrollo, hay seis puntos de control, de los cuales, tres son del primer *sprint* y los otros tres del segundo.

Como se puede observar en el gráfico, durante el desarrollo de la aplicación, no se ha igualado o superado la línea ideal, exceptuando el punto de control o reunión final. El 8 de diciembre ha sido el momento donde la línea de desarrollo se ha acercado más a la ideal, antes de la finalización del proyecto.

Otra observación que se puede apreciar es que, en ambos *sprints*, se ha invertido más tiempo de esfuerzo o trabajo en la segunda mitad de cada *sprint*, siendo la del último, la pendiente más pronunciada.

8. Matrices de validación

Para validar los requisitos, casos de uso, clases y demás recursos de la planificación realizada anteriormente, se ha procedido a la creación de dos matrices de validación.

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

CU\RF	RF-1	RF-2	RF-3	RF-4	RF-5	RF-6	RF-7	RF-8	RF-9	RF-10	RF-11	RF-12	RF-13	RF-14
Insertar alumno	X	X												
Buscar alumno		X	X	X	X		X							
Mostrar alumno							X							
Modificar alumno				X										
Borrar alumno					X									
Listar alumnos						X								
Buscar grupo								X						
Mostrar grupo									X					
Reiniciar alumnos										X				
Guardar BD	X										X			
Cargar BD											X			
Guardar copia	X										X			
Cargar copia											X			
Registrar profesor												X		
Buscar profesor												X		X
Borrar profesor													X	
Iniciar sesión														X
Cerrar sesión														X

Tabla 1. *Matriz RF-CU.*

Al tener, cada requisito funcional, al menos un caso de uso, se podría decir que no hay ningún fallo en esta parte de la planificación del proyecto. En caso de querer revisar los requisitos funcionales, se recomienda volver al apartado de análisis y extracción de requisitos.

Matriz de casos de uso (CU) - Clases

CU\Clase	Persona	Alumno	Profesor	Agenda
Insertar alumno				X
Buscar alumno				X
Mostrar alumno				X
Modificar alumno				X
Borrar alumno				X
Listar alumnos				X
Buscar grupo				X
Mostrar grupo				X
Reiniciar alumnos				X
Guardar BD			X	
Cargar BD			X	
Guardar copia			X	
Cargar copia			X	
Registrar profesor			X	
Buscar profesor			X	
Borrar profesor			X	
Iniciar sesión			X	
Cerrar sesión			X	

Tabla 1. *Matriz CU-Clases.*

Como se puede observar, todos los casos de uso tiene asignada una clase, por lo tanto, la validación tendría un resultado positivo. Las clases *Persona* y *Alumno* no tienen asignado ningún caso de uso. Esto es debido a que estas clases solo guardan información y sus métodos son observadores y modificadores de esa información.

La clase *Persona* podría ser prescindible en este problema, si sus datos fuesen guardados por las clases *Alumno* y *Profesor*. Sin embargo, se ha preferido mantener esta clase para tener una estructura jerarquizada, además de evitar complicaciones si en un futuro se quisiera añadir otro tipo de persona al sistema de la aplicación.

Bibliografía

- Learning Guides - Visual Paradigm. En línea: https://www.visual-paradigm.com/guide/

- Documentación - LibreOffice. En línea: https://wiki.documentfoundation.org/Documentation/es#Gu.C3.ADas_de_usuario