UNIVERSIDAD DE CÓRDOBA

Ingeniería del Software

GRADO DE INGENIERÍA INFORMÁTICA

INGENIERÁ DEL SOFTWARE

BASE DE DATOS DE ALUMNOS

Autor/es:

- **José Manuel Alcalde Llergo**
- **Francisco Bérchez Moreno**
- **Javier Herrero Porras**

Fecha: 23/12/2018

ÍNDICE DE CONTENIDOS

1 DEFINICIÓN DEL PROBLLEMA 5

1.1 DESCRIPCIÓN GENERAL 5

1.2 SOLUCIÓN PROPUESTA 6

2 EXTRACCIÓN DE REQUISITOS 7

2.1 PARTES INTERESADAS 8

2.2 DATOS DE PERSONA 8

2.3 DATOS DE PROFESOR 8

2.4 DATOS DE ALUMNO 8

2.5 REQUERIMIENTOS FUNCIONALES 9

2.6 REQUERIMIENTOS NO FUNCIONALES 9

2.7 PRIORIDADES 10

3 HISTORIAS DE USUARIO 11

3.1 HISTORIA DE USUARIO 01 12

3.2 HISTORIA DE USUARIO 02 12

3.3 HISTORIA DE USUARIO 03 13

3.4 HISTORIA DE USUARIO 04 13

3.5 HISTORIA DE USUARIO 05 14

3.6 HISTORIA DE USUARIO 06 14

3.7 HISTORIA DE USUARIO 07 15
3.8 HISTORIA DE USUARIO 08 15
3.9 HISTORIA DE USUARIO 09 16
3.10 HISTORIA DE USUARIO 10 16
3.11 HISTORIA DE USUARIO 11 17
3.12 HISTORIA DE USUARIO 12 17
3.13 HISTORIA DE USUARIO 13 18
3.14 HISTORIA DE USUARIO 14 18

4 CASOS DE USO 19

TABLA 1: CU-01 20
TABLA 2: CU-02 22
TABLA 3: CU-03 23
TABLA 4: CU-04 24
TABLA 5: CU-05 26
TABLA 6: CU-06 28
TABLA 7: CU-07 29
TABLA 8: CU-08 30
TABLA 9: CU-09 32
TABLA 10 CU-10 33
TABLA 11: CU-11 34
TABLA 12: CU-12 35

TABLA 13: CU-13 36
TABLA 14: CU-14 37

5 DIAGRAMA DE CLASES 38
5.1 DIAGRAMA 38
5.2 EXPLICACIÓN DE LAS CLASES 39
5.3 EXPLICACIÓN DE LAS RELACIONES 42

6 DIAGRAMAS DE SECUENCIA 44
6.1 INTRODUCCIÓN 44
6.2 DIAGRAMAS 45
INSERTAR ALUMNO 45
BUSCAR ALUMNO 46
MOSTRAR ALUMNO 47
MOSTRAR TODOS 48
MODIFICAR ALUMNO 49
BORRAR ALUMNO 50
BORRAR TODOS 51
GUARDAR AGENDA 52
CARGAR AGENDA 53
AUTENTIFICAR 54
AÑADIR PROFESOR 55
BORRAR PROFESOR 56

CREAR COPIA DE SEGURIDAD 57

RESTAURAR COPIA DE SEGURIDAD 58

7 METODOLOGÍA SCRUM 59

8 MATRICES DE VALIDACIÓN 63

9 BIBLIOGRAFÍA 65

1 DEFINICION DEL PROBLEMA

1.1 DESCRIPCIÓN GENERAL

El problema planteado consiste en la creación de una base de datos para los alumnos de la universidad, la cual podrá albergar un total de 150 alumnos.

Además, la base de datos deberá tener una serie de funcionalidades para actuar sobre los alumnos de dicha base y sobre los profesores que operan con ella.

Dichas funcionalidades deben ser: insertar un nuevo alumno en la base de datos, mostrar un alumno, mostrar todos los alumnos de dicha base de datos, buscar un alumno, modificar un alumno, borrar un alumno, borrar toda la base de datos y guardar y cargar la base de datos. Añadiendo cualquier tipo de funcionalidad extra que se considere oportuna a la hora de desarrollar dicha base de datos.

1.2 Solución Propuesta

Una vez comenzado el trabajo de desarrollo de la base de datos, consideramos necesario establecer una referencia a aquellos profesores que puedan acceder a esta, por lo que se consideró necesario discutirlo con el cliente y en respuesta a nuestra pregunta el susodicho decidió que se deberían crear dos tipos de profesores, uno el profesor ayudante y otro el profesor coordinador.

El profesor ayudante podría utilizar todas las funciones de la base de datos excepto, las funciones de insertar o borrar un profesor ayudante y las de hacer y restaurar una copia de seguridad de la base de datos, las cuales según el cliente deberán quedar reservadas para el profesor coordinador.

Por lo que a la hora de diseñar la base de datos se decidió atribuirle a cada profesor un usuario y contraseña, con el objetivo de poder limitar el uso de la base de datos solo para aquellos profesores autorizados y así poder limitar las acciones de los profesores ayudantes con respecto a la del profesor coordinador.

En cuanto al resto de peticiones del cliente, en relación con las funcionalidades de la base de datos, se especificarán en el siguiente punto con mucho más detalle, incluyendo las restricciones para cada una designadas por el cliente.

2 EXTRACCIÓN DE REQUISITOS

En este punto trataremos todos aquellos requisitos que se han discutido con el cliente, en los que primero especificaremos cuales son las partes interesadas en este proyecto, y a continuación, se presentarán una serie de atributos que serán necesarios para definir cuáles serán los campos que se tratarán en dicha base de datos cuando utilicemos sus distintas funcionalidades.

Una vez terminada esta parte inicial comenzaremos primero especificando cada una de las funcionalidades requeridas por el cliente para este proyecto, las cuales serán identificadas como requerimientos funcionales. Como continuación a esto se especifican una serie de restricciones necesarias para las funciones designadas por el cliente las cuales se conocerán como requerimientos no funcionales.

Por último, encontramos el apartado de prioridades, en el cual se le asigna una prioridad en relación con la importancia y necesidad de unas funciones sobre otras en la base de datos.

2.1 Partes Interesadas:

1. Profesores de la asignatura de IS.
2. Alumnos de la asignatura de IS.

2.2 Datos de Persona:

1. DNI (identificador)
2. Nombre
3. Apellido 1
4. Apellido 2
5. Teléfono
6. E-mail corporativo
7. Dirección postal
8. Fecha de nacimiento

2.3 Datos de Profesor:

Hereda los datos de persona y además contiene:

1. Nombre Usuario
2. Coordinador

2.4 Datos de Alumno:

Hereda los datos de persona y además contiene:

1. Curso más alto en el que está matriculado
2. Equipo
3. Líder o no líder

2.5 REQUERIMIENTOS FUNCIONALES (RF):

1. Insertar un alumno, en el caso de que el DNI o e-mail corporativo del alumno no esté ya en la base de datos. Se pueden introducir alumnos con el resto de los datos repetidos.
2. Mostrar un alumno existente en la base de datos, con todos sus campos.
3. Mostrar el listado de los alumnos ordenado por apellidos, nombre, DNI, por curso más alto que esté matriculado y en orden ascendente o descendente. Si la base de datos está vacía se indicará al cliente. El listado mostrará todos los campos. Se podrá filtrar el listado para que solo aparezcan alumnos con características comunes. Estas características podrán ser equipo, líder o curso más alto matriculado, y serán especificadas por el usuario a la hora de mostrarlos.
4. Buscar alumno por apellido 1 o ambos, equipo o DNI.
5. Modificar los datos de un alumno.
6. Borrar un alumno, en caso de que el alumno borrado sea el líder de un equipo indicar una advertencia.
7. Borrar toda la base de datos, enviando previamente una advertencia de que no se podrán recuperar los datos.
8. Guardar la nueva información y la modificada.
9. Cargar la información ya existente.
10. Para la identificación, será necesario aportar el nombre de usuario y la contraseña.
11. Añadir un profesor ayudante.
12. Borrar un profesor ayudante.
13. Hacer copia de seguridad.
14. Restaurar la copia de seguridad.

2.6 REQUERIMIENTOS NO FUNCIONALES:

1. El tipo de formato de los ficheros será binario, tanto para el de las credenciales como el de los alumnos.
2. La salida será mostrada por pantalla.
3. El número máximo de alumnos será de 150.
4. El cliente preestablece quién será el profesor coordinador, el cual estará ya registrado en la base de datos.

2.7 PRIORIDADES:

1. **Prioridad 1:** Inserción alumno
2. **Prioridad 1:** Inserción profesor
3. **Prioridad 2:** Autentificar
4. **Prioridad 2:** Cargar
5. **Prioridad 2:** Guardar
6. **Prioridad 2:** Buscar
7. **Prioridad 3:** Hacer copia seguridad
8. **Prioridad 3:** Restaurar copia seguridad
9. **Prioridad 3:** Mostrar un alumno
10. **Prioridad 4:** Mostrar todos los alumnos
11. **Prioridad 5:** Modificar
12. **Prioridad 5:** Borrar alumno
13. **Prioridad 5:** Borrar profesor
14. **Prioridad 6:** Borrar toda la base de datos

3 HISTORIAS DE USUARIO

En nuestro proyecto de creación de una base de datos de 150 alumnos para la universidad, se han creado 14 historias de usuario las cuales la utilizaremos a modo de una breve descripción de cada una de las funcionalidades del software que es, en esencia, lo que el cliente quiere.

Cada historia de usuario vendrá distribuida de la siguiente forma: un anverso en el que vendrá un identificador de dicha historia de usuario, una breve descripción de su funcionalidad software y la prioridad de esta; y una segunda parte llamada reverso en la que vendrá una especificación de la funcionalidad software para que sirva como orientación para la persona encargada de programar dicha funcionalidad.

3.1 Historia de Usuario 01

ANVERSO

ID: 01 **Insertar alumno**

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

Prioridad: 1

REVERSO

1. Quiero poder introducir un nuevo alumno.
2. Todos los campos son obligatorios excepto el equipo y el líder.
3. No podemos introducir alumnos cuyo DNI o e-mail corporativo esté ya en el programa. Esto implica que no existen dos alumnos iguales en la base de datos.

3.2 Historia de Usuario 02

ANVERSO

ID: 02 **Buscar alumno**

Como usuario quiero buscar un alumno determinado.

Prioridad: 2

REVERSO

1. Quiero poder buscar un alumno determinado dentro de la base de datos.
2. Podremos buscar a dicho alumno por apellidos, DNI o incluso el equipo del cual forma parte. Si los apellidos se repiten, entonces se pedirá el DNI.

3.3 Historia de Usuario 03

ANVERSO

ID: 03 **Mostrar alumno**

Como usuario, quiero poder visualizar los datos de un alumno que exista en la base de datos.

Prioridad: 3

REVERSO

1. Quiero poder mostrar un alumno con todos sus datos.

3.4 Historia de Usuario 04

ANVERSO

ID: 04 **Mostrar todos los alumnos**

Como usuario quiero poder mostrar todos los alumnos ordenados, o un subconjunto de ellos.

Prioridad: 4

REVERSO

1. La aplicación podrá mostrar el listado de todos los alumnos. Podrán ser ordenados alfabéticamente (apellidos o nombre), por número de DNI, o por el curso más alto de matriculación en orden ascendente o descendente.
2. La aplicación también podrá mostrar ciertos alumnos con características comunes. Estas características pueden ser equipo, líder o curso más alto matriculado.

3.5 Historia de Usuario 05

ANVERSO

ID: 05 **Modificar alumno**

Como usuario, quiero poder modificar un alumno existente en la base de datos.

Prioridad: 5

REVERSO

1. Quiero poder modificar un alumno que exista en la base de datos.

3.6 Historia de Usuario 06

ANVERSO

ID: 06 **Borrar un alumno**

Como usuario quiero poder borrar un determinado alumno.

Prioridad: 5

REVERSO

1. Si el alumno que borramos es el líder de un equipo, deberemos mostrar un mensaje de advertencia.

3.7 Historia de Usuario 07

ANVERSO

ID: 07 **Borrar todos los alumnos**

Como usuario quiero poder vaciar toda la base de datos, es decir, borrar todos los alumnos.

Prioridad: 5

REVERSO

1. La función eliminará todos los datos guardados en el fichero binario.
2. La función dará un aviso al usuario antes de proceder al borrado.

3.8 Historia de Usuario 08

ANVERSO

ID: 08 **Guardar**

Como usuario quiero poder guardar las modificaciones hechas en el programa.

Prioridad: 2

REVERSO

1. Quiero poder almacenar los datos introducidos en el programa dentro de un fichero binario.
2. El fichero se guardará en el mismo directorio que el programa.

3.9 Historia de Usuario 09

ANVERSO

ID: 09 **Cargar**

Como usuario quiero poder cargar los datos que he almacenado en la base de datos.

Prioridad: 2

REVERSO

1. Quiero poder recuperar los datos que he almacenado sobre los alumnos dentro del fichero binario.

3.10 Historia de Usuario 09

ANVERSO

ID: 10 **Autentificar**

Como usuario quiero identificarme para entrar a la base de datos.

Prioridad: 2

REVERSO

1. La función pedirá usuario y contraseña.

3.11 Historia de Usuario 11

ANVERSO

ID: 11 **Añadir profesor**

Como profesor quiero añadir otros profesores.

Prioridad: 1

REVERSO

1. La función le permitirá añadir al coordinador un nuevo profesor.

3.12 Historia de Usuario 12

ANVERSO

ID: 12 **Borrar profesor**

Como profesor quiero borrar otros profesores.

Prioridad: 5

REVERSO

1. La función le permitirá al coordinador borrar otro profesor

3.13 Historia de Usuario 13

ANVERSO

ID: 13 **Hacer copia de seguridad**

Como profesor quiero hacer una copia de seguridad de la agenda de alumnos

Prioridad: 3

REVERSO

1. La función le permitirá al coordinador realizar una copia de seguridad de la agenda de alumnos que haya en ese momento.

3.14 Historia de Usuario 14

ANVERSO

ID: 14 **Restaurar copia de seguridad**

Como profesor quiero restaurar una copia de seguridad de la agenda de alumnos.

Prioridad: 3

REVERSO

1. La función le permitirá al coordinador restaurar una copia de seguridad realizada sobre la agenda de alumnos.

4 CASOS DE USO

Para poder hacer el modelado del comportamiento que se desee del sistema se utilizarán los casos de uso. Primero describiremos los actores que interactúan con el sistema.

Después se analizará cada módulo del sistema mediante el caso de uso correspondiente.

Identificación de Actores en el Sistema

Se describirá con detalle cada uno de aquellos actores que forman parte de la interacción con el sistema. Entre ellos se pueden identificar a los siguientes:

- **Profesor Coordinador:** Profesor encargado de organizar la asignatura. Tiene privilegios en el sistema
- **Profesor Ayudante:** Profesor que es añadido por el Coordinador. Solo tiene acceso a una parte del sistema.
- **Usuario no Registrado:** Persona que todavía no se ha autentificado en el sistema.

NOMBRE	**Insertar alumno**
IDENTIFICADOR	CU-01
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario introduce un nuevo alumno a la Base de Datos.
PRECONDICIONES	1. El DNI introducido debe ser correcto. 2. El total de alumnos no debe exceder 150.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario necesita introducir un alumno. 2. Se introducen los campos con un determinado orden: DNI, nombre, apellido, teléfono, e-mail corporativo, dirección postal, curso más alto matriculado, equipo del que forma parte, líder/ no líder.
POSTCONDICIONES	1. Los datos se escriben en la base de datos. 2. Se muestra por pantalla un mensaje informando del proceso correcto. 3. Los únicos campos que se pueden quedar en blanco son equipo y líder.

FLUJO ALTERNATIVO	1. Si el DNI no es correcto, se muestra un mensaje de error, y se vuelve a pedir el DNI. 2. Si el DNI o el e-mail se repiten con los de otro alumno, se para el proceso y se muestra un mensaje de error, volviendo al menú de opciones. 3. En el caso de que ya haya 150 alumnos, se mandará un mensaje de error y se volverá al menú de opciones.

Tabla 1: CU-01

NOMBRE	**Buscar alumno**
IDENTIFICADOR	CU-02
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario busca un alumno dentro de la Base de Datos.
PRECONDICIONES	
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario necesita buscar un alumno de la base de datos. 2. El alumno será buscado por DNI, apellidos o equipo al que pertenece.
POSTCONDICIONES	El Alumno será pasado a la Función correspondiente.
FLUJO ALTERNATIVO	1. Si el DNI no es correcto, se muestra un mensaje de error, y se vuelve a pedir el DNI. 2. Si el parámetro usado para buscar no existe en la base de datos se volverá a pedir. 3. Si los apellidos se repiten, entonces se pedirá el DNI del alumno.

Tabla 2: CU-02

NOMBRE	**Mostrar alumno**
IDENTIFICADOR	CU-03
ACTORES	Profesor Coordinador/ Ayudante
DESCRIPCIÓN	El sistema muestra un Alumno.
PRECONDICIONES	El Alumno para mostrar deberá existir en la Base De Datos.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario necesita mostrar un alumno. 2. Se recogen los datos del alumno buscado.
POSTCONDICIONES	Los datos del Alumno se muestran por pantalla.
FLUJO ALTERNATIVO	1. Si los campos introducidos no coinciden con la base de datos, se muestra un mensaje de error y vuelve al menú de opciones.

Tabla 3: CU-03

NOMBRE	**Mostrar Todos los Alumnos**
IDENTIFICADOR	CU-04
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El sistema muestra todos los Alumnos existentes en la Base de Datos.
PRECONDICIONES	Debe haber al menos un Alumno dentro de la Base de Datos.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario quiere visualizar todos los alumnos de la base de datos. 2. El programa pregunta al usuario si quiere visualizar algún subconjunto de alumnos (equipo, líder) o quiere visualizar todos. 3. Se mostrarán todos los alumnos: ordenados por apellidos y nombre, DNI o curso más alto matriculado. Todas las opciones pueden ser ascendente o descendentemente y mostraran: DNI, nombre, apellido, telefono, e-mail corporativo, dirección postal, curso más alto matriculado, equipo del que forma parte, líder/ no líder.
POSTCONDICIONES	Los datos de los Alumnos se muestran por pantalla. Adicionalmente, se genera un archivo Mark Down con la consulta.

FLUJO ALTERNATIVO	En caso de que no exista ningún alumno en la base de datos, se mostrará un mensaje de error y se volverá al menú.

Tabla 4: CU-04

NOMBRE	**Modificar alumno**
IDENTIFICADOR	CU-05
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario modifica un Alumno existente en la Base de Datos.
PRECONDICIONES	El Alumno para modificar debe existir dentro de la Base de Datos.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario quiere cambiar los datos de un alumno. 2. Se debe buscar al alumno antes de modificar sus datos. 3. Se pueden cambiar los datos del alumno: DNI, nombre, apellido, teléfono, e-mail corporativo, dirección postal, curso más alto matriculado, equipo del que forma parte, líder/ no líder.
POSTCONDICIONES	1. Los datos modificados se reescriben en la base de datos 2. Se muestra por pantalla un mensaje informando del proceso correcto 3. El usuario no puede dejar campos en blanco, excepto el líder y el equipo.

FLUJO ALTERNATIVO	1. Si el DNI no es correcto, se muestra un mensaje de error y se vuelve a pedir el DNI hasta que sea correcto. 2. Si el DNI o apellidos coinciden con los de otro alumno, se para el proceso y se muestra un mensaje de error. Se repite el proceso. 3. Si el usuario pone como líder a un alumno que este dentro de un grupo que ya cuente con un líder, se mostrará un mensaje de error y este campo del alumno se pondrá como falso.

Tabla 5: CU-05

NOMBRE	**Borrar Alumno**
IDENTIFICADOR	CU-06
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario borra el Alumno que desee de la Base de Datos
PRECONDICIONES	El alumno que el usuario desee borrar debe existir
FLUJO PRINCIPAL	1. El alumno que se quiera borrar se buscará dentro de la base de datos. 2. Se eliminará el alumno del programa 3. En caso de que el alumno borrado sea el líder de un equipo se mostrará una advertencia.
POSTCONDICIONES	
FLUJO ALTERNATIVO	1. En caso de que el alumno no exista dentro de la base de datos, se mostrará un mensaje de error y volverá al menú.

Tabla 6: CU-06

NOMBRE	**Borrar todos los Alumnos**
IDENTIFICADOR	CU-07
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario borra todos los alumnos de la base de datos
PRECONDICIONES	Debe haber al menos un alumno en la base de datos.
FLUJO PRINCIPAL	1. El programa muestra una advertencia de que no se podrán recuperar los datos. 2. El usuario confirma o no la acción 3. Se borra el contenido del fichero binario en el que están contenidos los alumnos.
POSTCONDICIONES	Se muestra un mensaje al usuario de que los datos han sido borrados
FLUJO ALTERNATIVO	En caso de que no haya alumnos en la base de datos, se mostrará un mensaje de error.

Tabla 7: CU-07

NOMBRE	**Guardar la Base de Datos**
IDENTIFICADOR	CU-08
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario quiere volcar los cambios sobre un fichero binario
PRECONDICIONES	La Base de Datos debe haber sido cargada antes de realizar algún cambio.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario hace alguna operación de modificación o adición, entonces se volcará el contenido nuevo en la base de datos. 2. Se pregunta al usuario si quiere guardar los cambios sobre el mismo fichero o no. 3. Se pide el nombre del fichero al usuario. 4. Se crea el fichero con el nombre especificado
POSTCONDICIONES	

FLUJO ALTERNATIVO	1. En caso de que no se pueda guardar el contenido, se mostrará un mensaje de error y se finalizará el programa. 2. Si el nombre del fichero que pide el usuario coincide con algún fichero en el directorio, se muestra un mensaje al usuario. Puede sobrescribirlo o cambiar el nombre.

Tabla 8: CU-08

NOMBRE	**Cargar la Base de Datos**
IDENTIFICADOR	CU-09
ACTORES	Profesor Coordinador/Ayudante
DESCRIPCIÓN	El usuario quiere volcar los cambios sobre un fichero binario
PRECONDICIONES	1. Existe un fichero binario sobre el que cargaremos la base de datos.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario accede al programa. 2. Se cargará el fichero para su uso por el usuario.
POSTCONDICIONES	
FLUJO ALTERNATIVO	1. En caso de que no se pueda abrir el fichero, se mostrará un mensaje de error, y se terminará el programa. 2. Si no existe fichero para cargar la base de datos, se mostrará un mensaje de error y se terminará el programa.

Tabla 9: CU-09

NOMBRE	**Autentificar**
IDENTIFICADOR	CU-10
ACTORES	Usuario no Registrado
DESCRIPCIÓN	Comprueba si el usuario tiene acceso al sistema
PRECONDICIONES	Debe existir un fichero binario en el directorio con los usuarios y las contraseñas de los profesores registrados.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario inicia el programa. 2. El programa pide al usuario su identificador y contraseña. 3. El programa comprueba si el usuario está en el fichero de credenciales. 4. El programa comprueba si la contraseña coincide con el usuario introducido 5. El sistema manda un mensaje de bienvenida al usuario.
POSTCONDICIONES	Se abre el menú de la base de datos.
FLUJO ALTERNATIVO	1. Si el usuario no se encuentra en el fichero de credenciales, no podrá acceder al menú y se le mandará un mensaje de error. 2. Si el usuario se encuentra en el fichero de credenciales, pero la contraseña no coincide, se le mandará un mensaje de error y no podrá acceder al sistema.

Tabla 10: CU-10

NOMBRE	**Insertar Profesor**
IDENTIFICADOR	CU-11
ACTORES	Profesor Coordinador
DESCRIPCIÓN	El profesor coordinador añade un profesor ayudante al sistema
PRECONDICIONES	1. Debe existir un fichero binario en el directorio con los usuarios y las contraseñas de los profesores registrados. 2. El usuario debe estar registrado como coordinador para ejecutar esta funcionalidad.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el coordinador necesita añadir un nuevo profesor que pueda acceder a la base de datos. 2. El programa pide al usuario el nombre de usuario que quiere añadir junto con la contraseña. 3. El programa comprueba si el usuario está ya en el fichero de credenciales. 4. Se introduce el nombre de usuario y su contraseña en el fichero de credenciales.
POSTCONDICIONES	En el fichero de credenciales queda modificado con el nuevo usuario y su contraseña.
FLUJO ALTERNATIVO	Si el nuevo usuario ya existía en la base de datos, no se añadirá y se mandará un mensaje de error.

Tabla 11: CU-11

NOMBRE	**Borrar Profesor**
IDENTIFICADOR	CU-12
ACTORES	Profesor Coordinador
DESCRIPCIÓN	Elimina un profesor de los que pueden acceder a la base de datos.
PRECONDICIONES	1. Debe existir un fichero binario en el directorio con los usuarios y las contraseñas de los profesores registrados. 2. El usuario debe estar registrado como coordinador para ejecutar esta funcionalidad.
FLUJO PRINCIPAL	3. El caso de uso empieza cuando el usuario quiere dejar de permitir el acceso al sistema a un profesor registrado. 4. El programa pide al coordinador el nombre de usuario que quiere borrar. 5. El programa comprueba si el usuario está en el fichero de credenciales. 6. El sistema elimina el profesor del fichero de credenciales.
POSTCONDICIONES	Se modifica el fichero de credenciales, dejándolo igual, pero sin incluir el profesor eliminado y la contraseña de este.
FLUJO ALTERNATIVO	1. Si el nombre de usuario a eliminar pertenece al coordinador, se mandará un mensaje de error y no se eliminará. 2. Si el usuario no se encuentra en el fichero de credenciales, este no se verá modificado.

Tabla 12: CU-12

NOMBRE	**Hacer Copia de Seguridad**
IDENTIFICADOR	CU-13
ACTORES	Profesor Coordinador
DESCRIPCIÓN	Hacer una copia de seguridad del fichero de alumnos.
PRECONDICIONES	Existe un fichero binario sobre el que realizaremos la copia de seguridad.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el usuario quiere realizar una copia de seguridad del fichero de alumnos. 2. Se realizará una copia de seguridad en el mismo directorio que el del programa
POSTCONDICIONES	
FLUJO ALTERNATIVO	1. Si ya existe un archivo con el mismo nombre, se pedirá al usuario un nuevo nombre para la copia de seguridad.

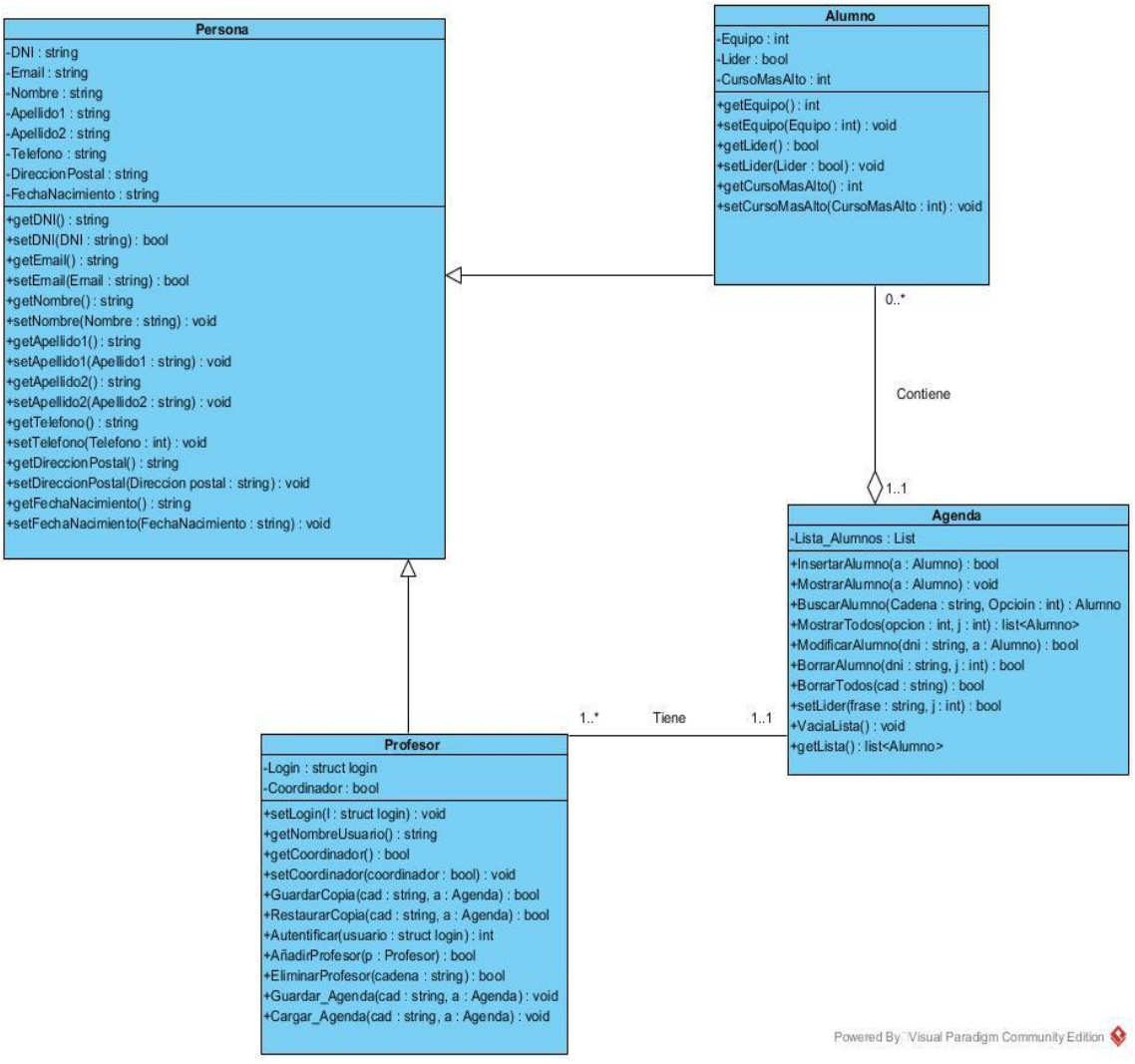
Tabla 13: CU-13

NOMBRE	**Restaurar Copia de Seguridad**
IDENTIFICADOR	CU-14
ACTORES	Profesor Coordinador
DESCRIPCIÓN	Restaura una copia de seguridad realizada con anterioridad.
PRECONDICIONES	Debe existir una copia de seguridad para que podamos restaurarla.
FLUJO PRINCIPAL	1. El caso de uso empieza cuando el coordinador necesita restaurar una copia de seguridad que haya realizado sobre la agenda de alumnos con anterioridad. 2. Se pide el nombre del fichero por el que restauraremos la base de datos. 3. Se restaura la copia seleccionada.
POSTCONDICIONES	La agenda de alumnos queda restaurada según una copia de seguridad previa.
FLUJO ALTERNATIVO	1. En caso de no existir una copia de seguridad previa se mostrará un mensaje de error al coordinador. 2. Si el nombre no coincide con ningún archivo, se muestra un mensaje de error y se vuelve al menú.

Tabla 14: CU-14

5 DIAGRAMA DE CLASES

5.1 DIAGRAMA



5.2 EXPLICACIÓN DE LAS CLASES

En nuestro diagrama de clases hemos considerado 4 clases distintas:

1. Persona
2. Alumno
3. Profesor
4. Agenda

Persona

En la clase Persona hemos definido una serie de atributos que sirven para hacer referencia a unas características generales tanto de los profesores como de los alumnos.

Dichos atributos son:

1. DNI
2. Email
3. Nombre
4. Apellido1
5. Apellido2
6. Teléfono
7. Dirección Postal
8. Fecha de nacimiento

En esta clase hemos definido una serie de funciones (get y set) para poder obtener y modificar cada uno de los atributos relacionados tanto con los alumnos como con los profesores.

Alumno

Definimos la clase Alumno como una especificación de la clase persona, por lo que a parte de los atributos típicos de los objetos de la clase persona, incluiremos otros propios de los alumnos.

Dichos atributos son:

1. Equipo
2. Líder
3. Curso más alto

En esta clase hemos implementado una serie de funciones (get y set) con las cuales podemos obtener y modificar los atributos propios de los objetos de la clase alumno. Además, al ser una especificación de la clase Persona podremos hacer uso de las funcionalidades de esta, ya que nos harán falta para poder visualizar el resto de los atributos del alumno.

Profesor

Definimos la clase Profesor como una especificación de la clase persona, por lo que a parte de los atributos típicos de los objetos de la clase persona, incluiremos otros propios de los profesores que accederán a nuestra base de datos.

Dichos atributos son:

1. Login
2. Coordinador
3. Puntero Agenda

En esta clase al igual que en las demás, se emplearán funciones get y set para obtener los distintos atributos del profesor que acceda a la base de datos.

Además, incluiremos una función de autentificación para que solo los usuarios registrados por el coordinador puedan tener acceso a la base de datos. También tendremos cuatro funciones tan solo ejecutables por el profesor coordinador, las cuales serán, realizar una copia de seguridad de la base de datos, restaurar dicha copia, añadir un nuevo profesor autorizado para emplear la base de datos y eliminar uno de estos profesores autorizados.

Por último, añadiremos una función de guardar los cambios y cargar datos de una agenda de alumnos, las cuales podrán ser realizadas por cualquier profesor. Además, al ser una especificación de la clase Persona podremos hacer uso de las funcionalidades de esta, ya que nos harán falta para poder visualizar el resto de los atributos del alumno.

AGENDA

En la clase Agenda hemos definido únicamente un solo atributo que será nuestra lista de alumnos, el cual será una lista en la cual almacenaremos los alumnos que tengamos en nuestra base de datos y sobre la que trabajaremos antes de realizar los cambios en la agenda.

En esta clase hemos implementado una función getLista, para poder obtener la lista de alumnos y trabajar con ella, y otra función para identificar si nuestra lista está vacía.

También hemos añadido otras funciones que son: introducir un alumno en la agenda, borrar un alumno de nuestro fichero, buscar un

alumno, modificarlo, mostrar uno o mostrarlos todos (incluyendo mostrar por subconjuntos de líderes, grupos o cursos) y borrar toda la agenda de alumnos. Además, definimos una agregación entre esta clase y la clase alumno, ya que la clase Alumno forma parte de la clase Agenda.

5.3 EXPLICACIÓN DE LAS RELACIONES

Persona-Alumno

Indican la relación entre una persona y uno de los alumnos cuyos datos vamos a almacenar en la base de datos.

Entre estas clases hay una relación de herencia, ya que todo alumno **ES** una persona, y por tanto Alumno heredará todos los atributos de la clase Persona, al igual que sus funcionalidades.

Al ser la clase Alumno una especificación de la clase Persona, cada alumno y cada alumno **ES** una única persona (1,1), aunque al ser una relación de herencia, no indicamos explícitamente en el diagrama esta multiplicidad.

Persona-Profesor

Indican la relación entre una persona y un profesor que va a acceder a la base de datos.

Entre estas clases hay una relación de herencia, ya que todo profesor **ES** una persona, y por tanto Profesor heredará todos los atributos de la clase Persona, al igual que sus funcionalidades.

Al ser la clase Profesor una especificación de la clase Persona, cada profesor **ES** una única persona (1,1), aunque al ser una relación

de herencia, no indicamos explícitamente en el diagrama esta multiplicidad.

Profesor-Agenda

Indican la relación entre un profesor que va a tener acceso a la base de datos y la agenda donde almacenamos los datos.

Entre estas clases hay una relación de asociación ya que la clase Profesor **ESTÁ ASOCIADA A** la clase Agenda, porque un profesor va a acceder a dicha agenda para trabajar con la información que necesite de ella y poder modificarla.

Cada profesor tiene asignada una única agenda (1,1), mientras que cada agenda puede pertenecer a uno o a varios profesores (1,*).

Alumno-Agenda

Indica la relación entre un alumno cuyos datos queremos almacenar y la agenda en la que se almacenarán dichos datos.

Entre estas clases hay una relación de agregación, ya que la clase Alumno **FORMA PARTE DE** la clase agenda, ya que nuestra agenda estará compuesta por alumnos.

Cada alumno pertenece a una única agenda (1,1) y en cada una de las agendas puede haber contenidos desde ningún alumno hasta varios (0,*).

6 DIAGRAMAS DE SECUENCIA

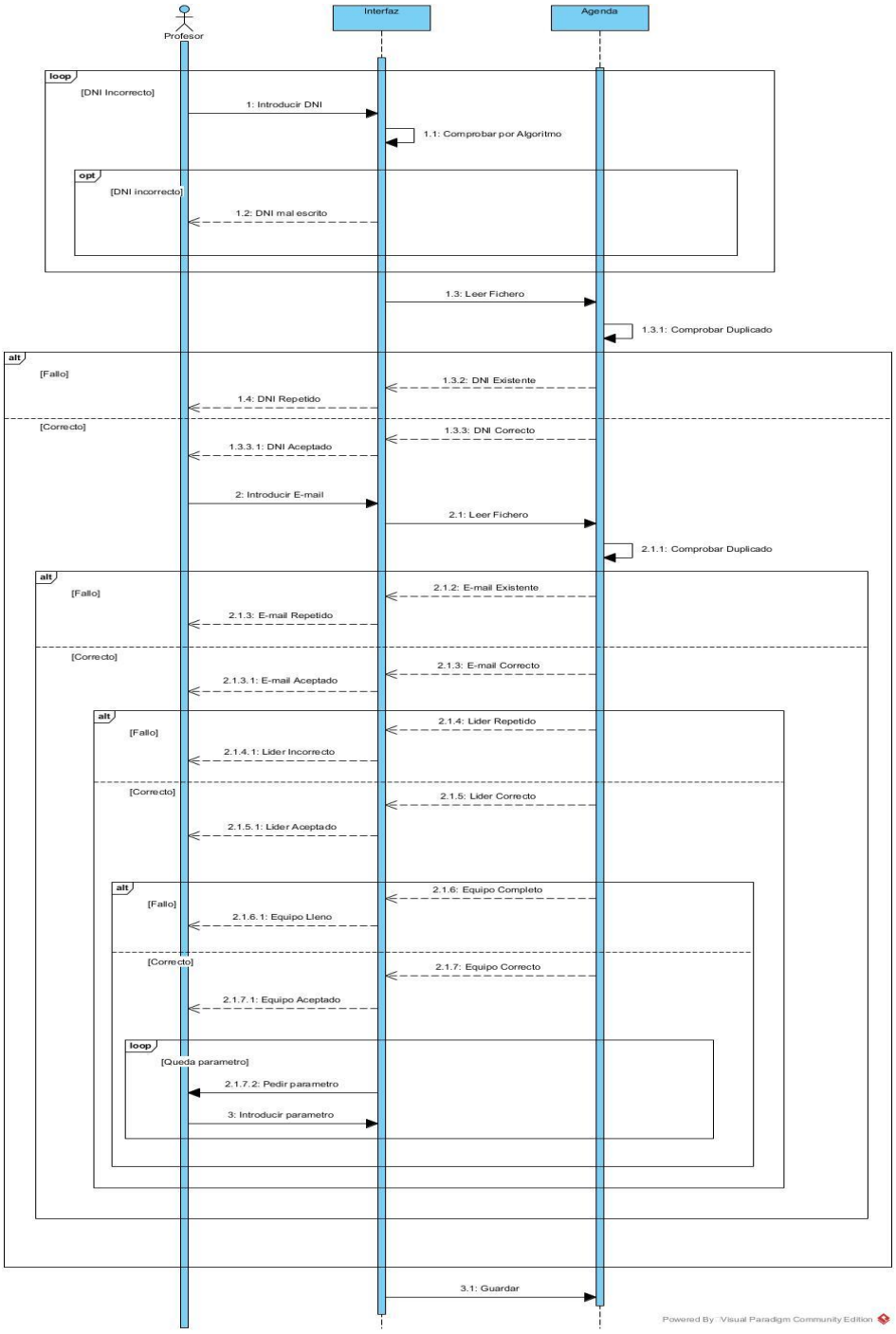
6.1 INTRODUCCIÓN

Para poder ver cómo se comunican los distintos objetos de nuestro sistema durante las interacciones, hemos creado una serie de diagramas de secuencias, acordes a los casos de uso detallados anteriormente. Estos serán:

- Insertar Alumno
- Buscar Alumno
- Mostrar Alumno
- Mostrar Todos
- Modificar Alumno
- Borrar Alumno
- Borrar Todos
- Guardar Agenda
- Cargar Agenda
- Autentificar
- Insertar Profesor
- Borrar Profesor
- Crear Copia de Seguridad
- Restaurar Copia de Seguridad

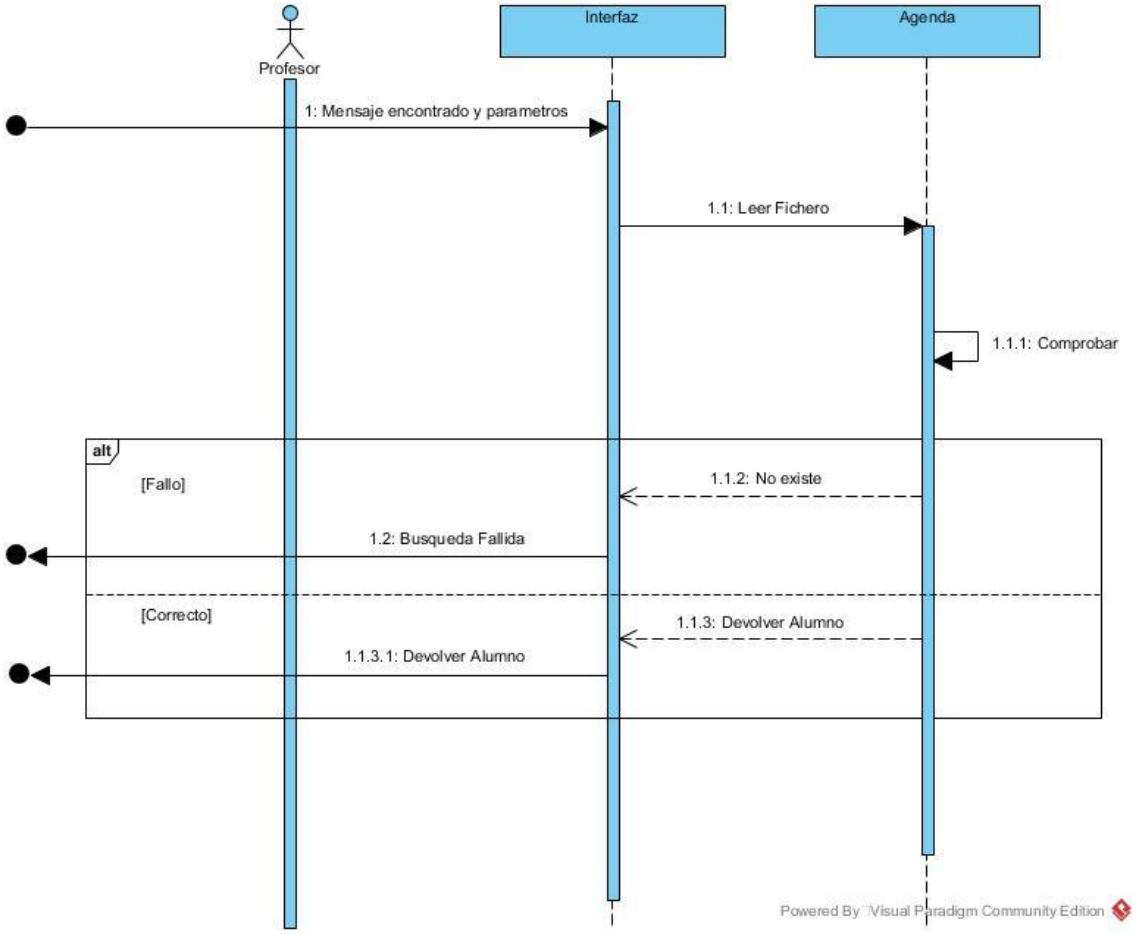
6.2 DIAGRAMAS

Insertar Alumno



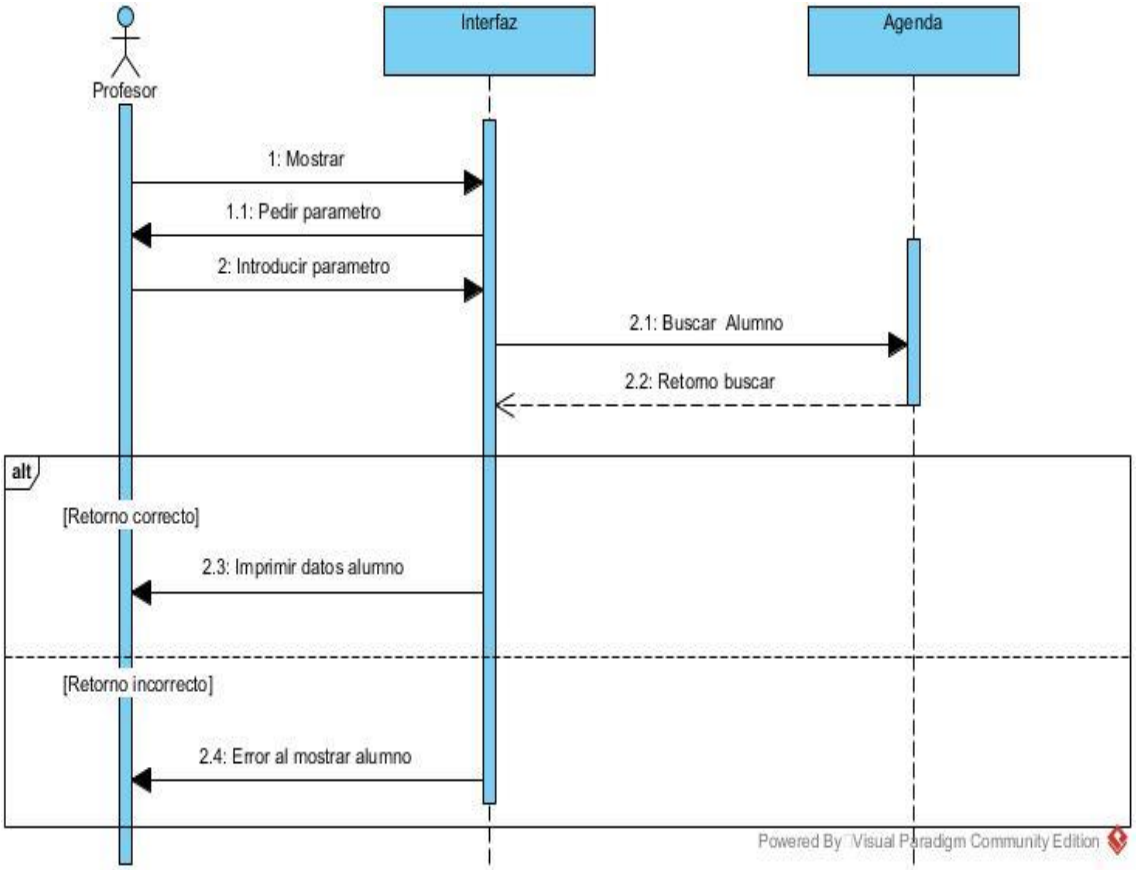
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de alumnos.

Buscar Alumno



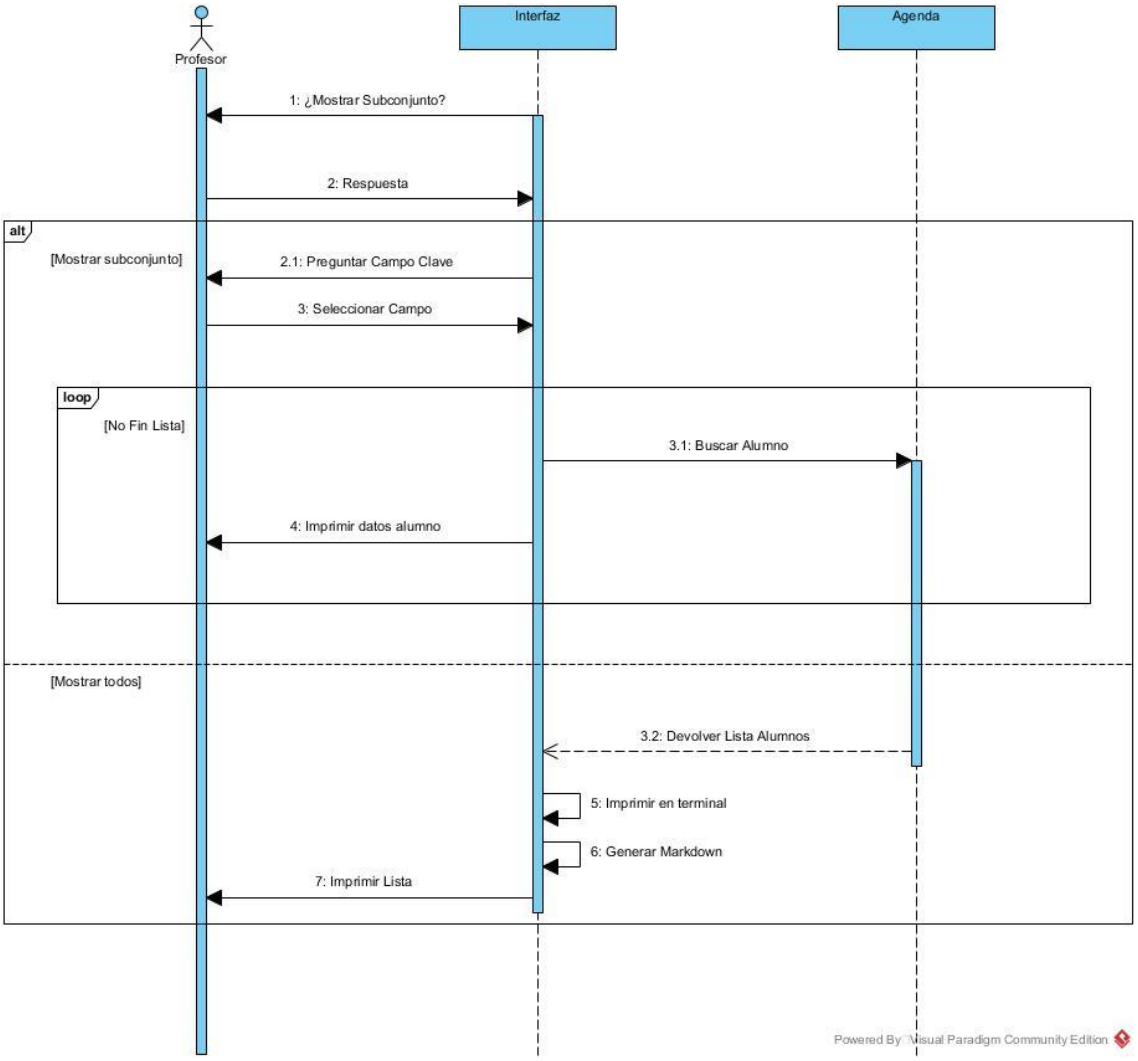
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Mostrar Alumno



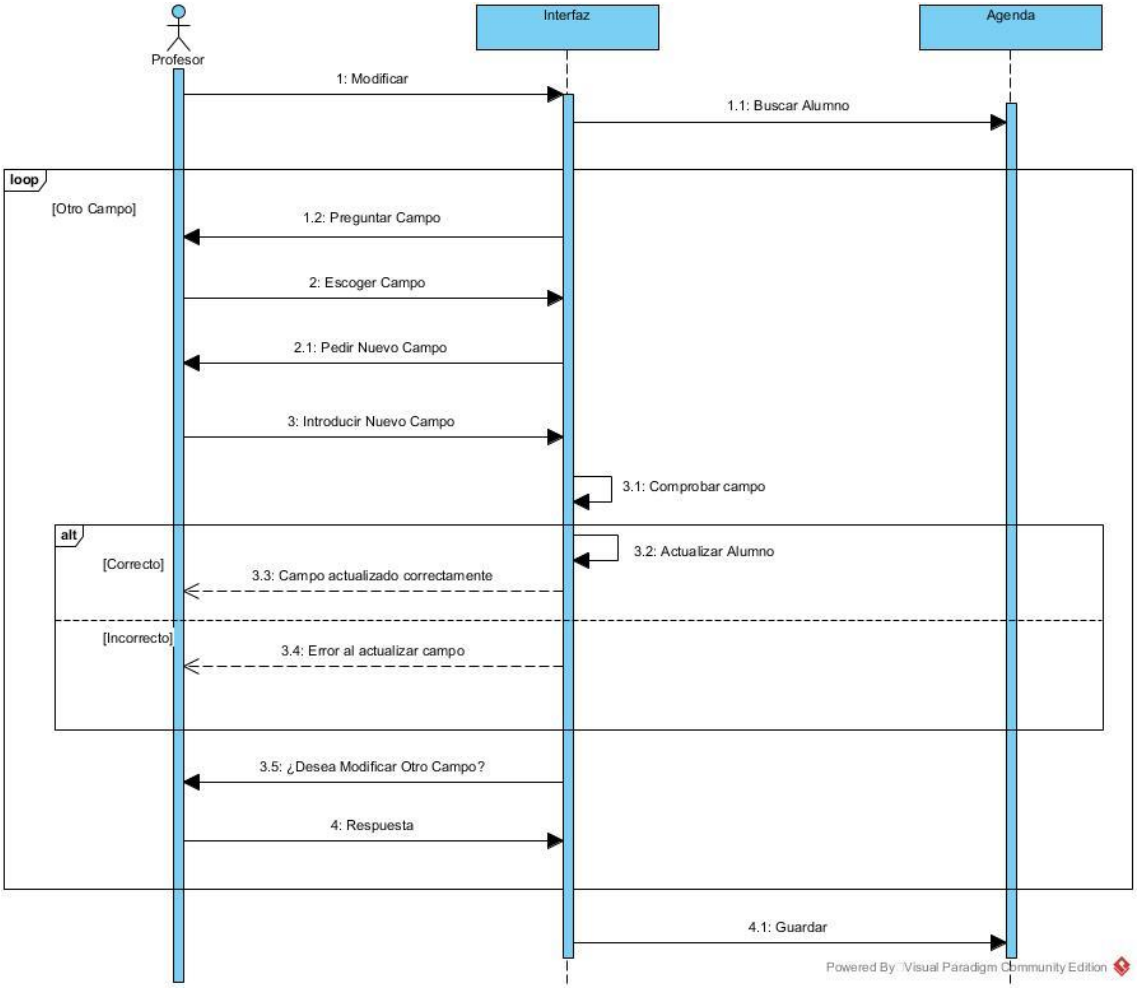
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Mostrar Todos



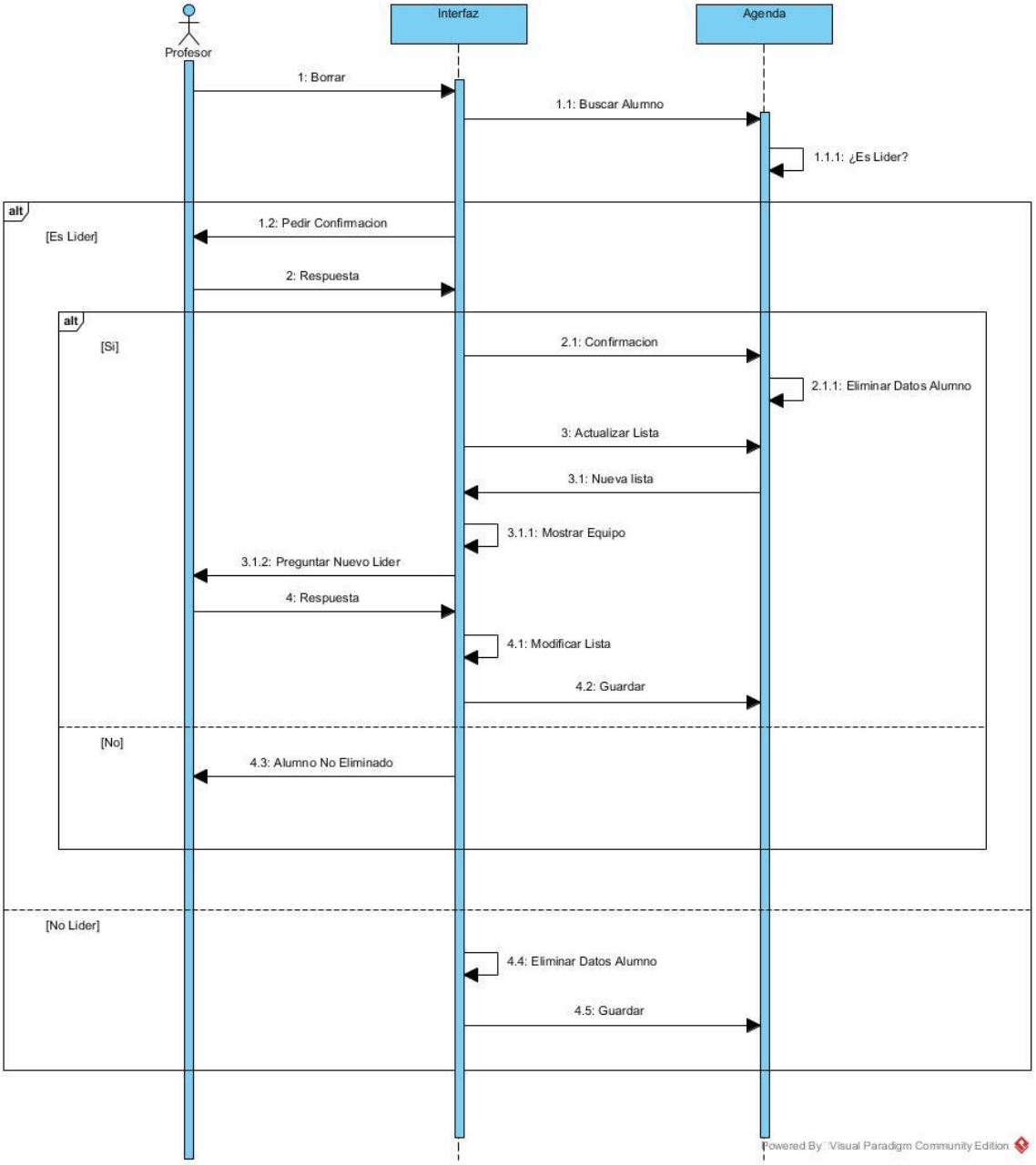
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Modificar Alumno



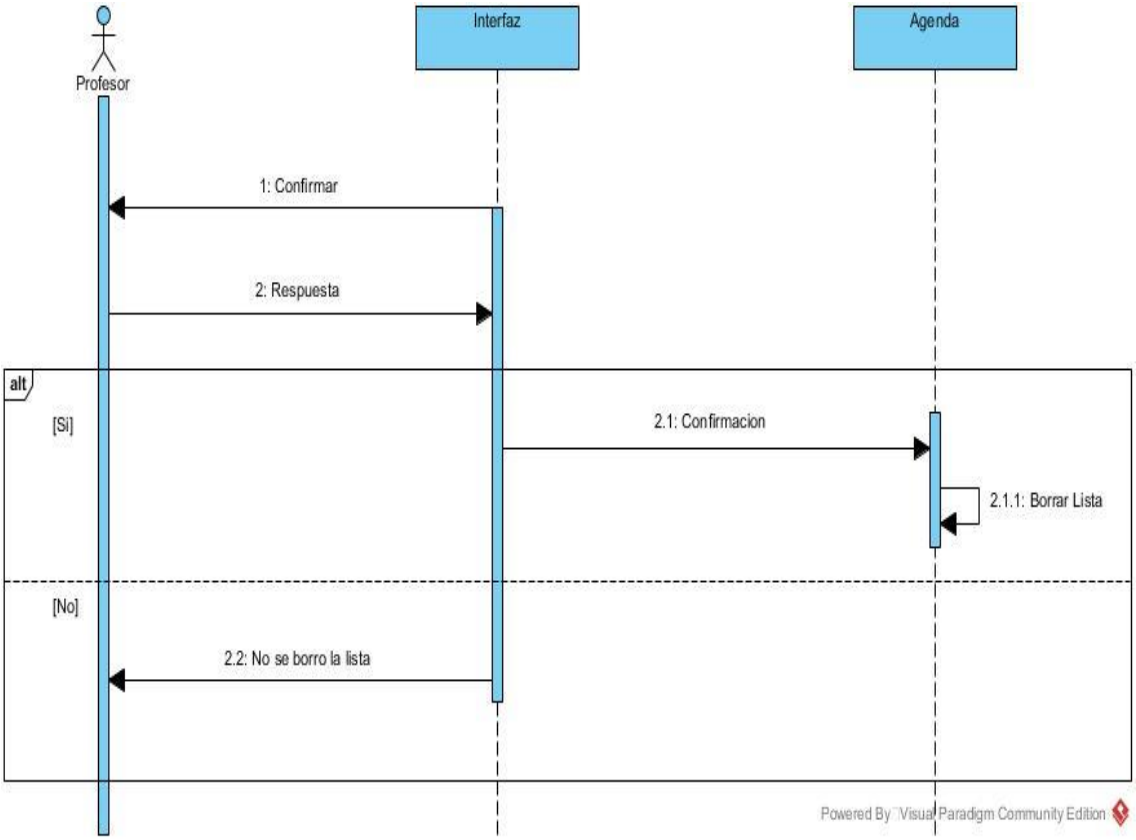
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Borrar Alumno



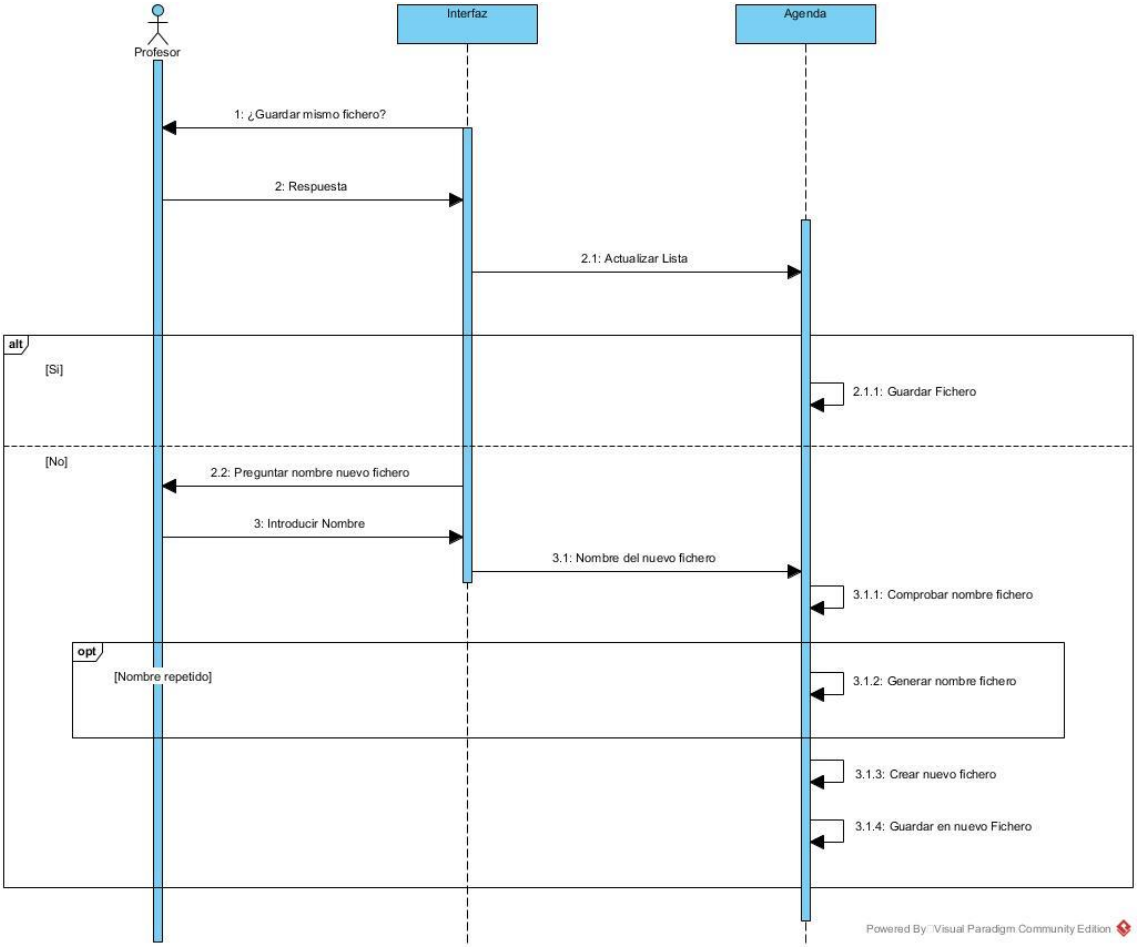
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Borrar Todos



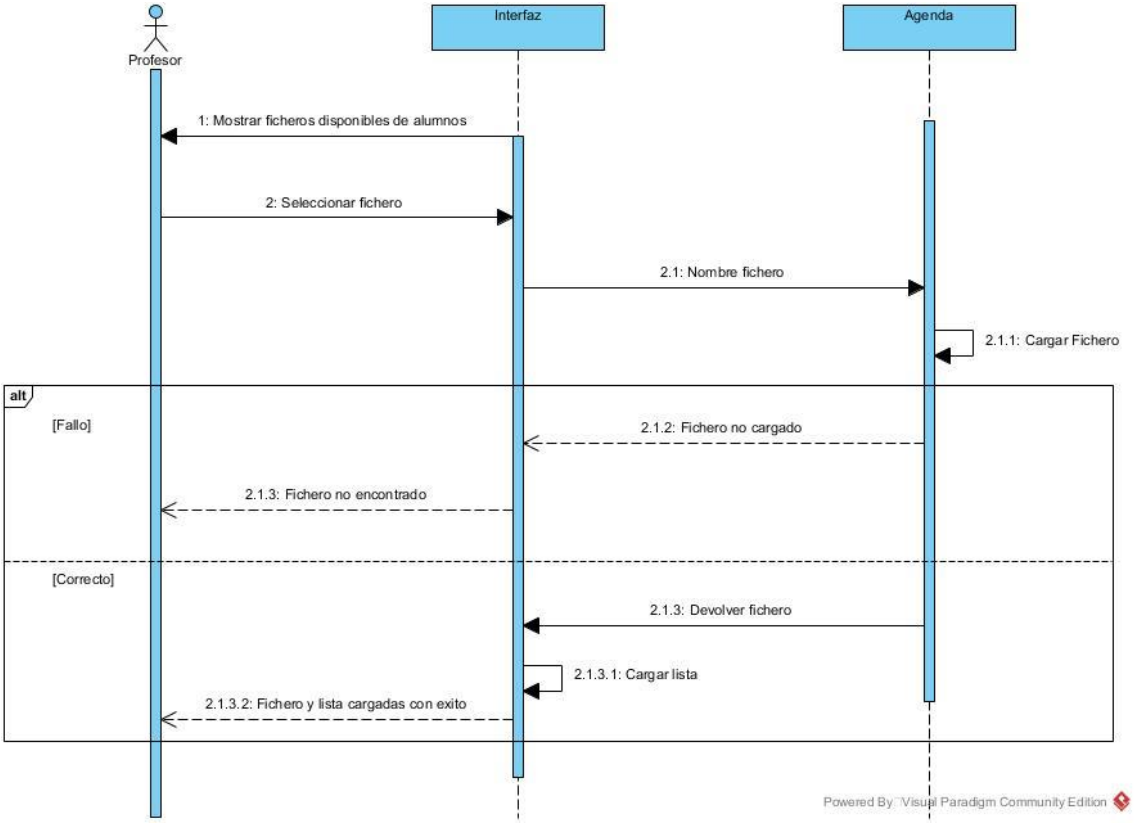
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Guardar Agenda



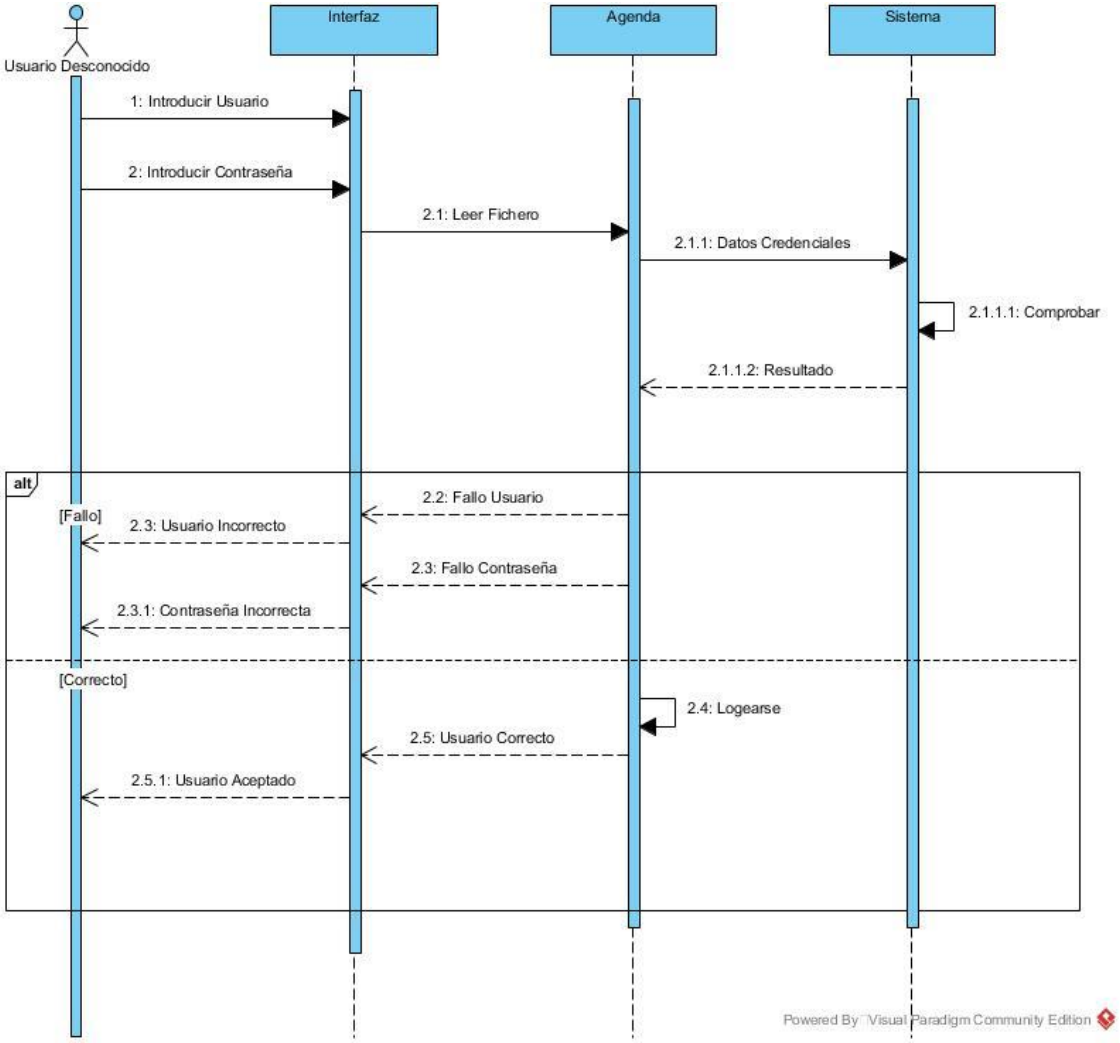
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Cargar Agenda



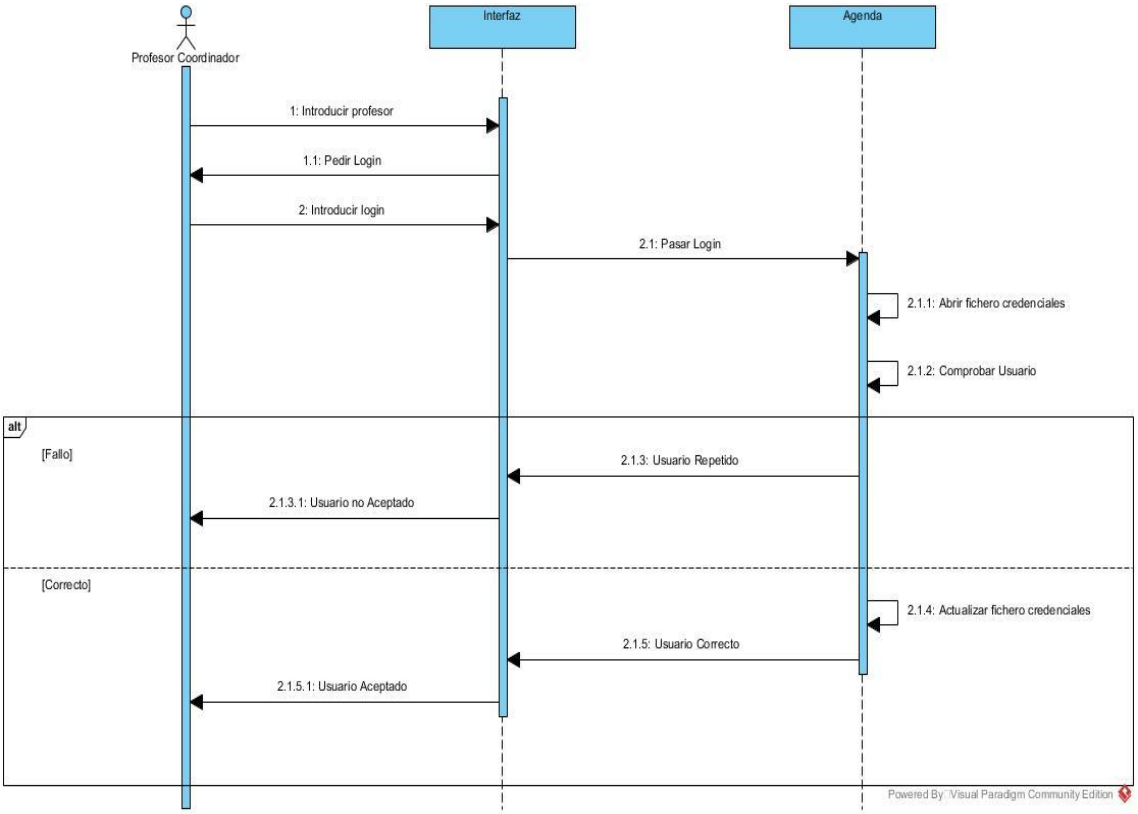
El actor es el profesor que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Autentificar



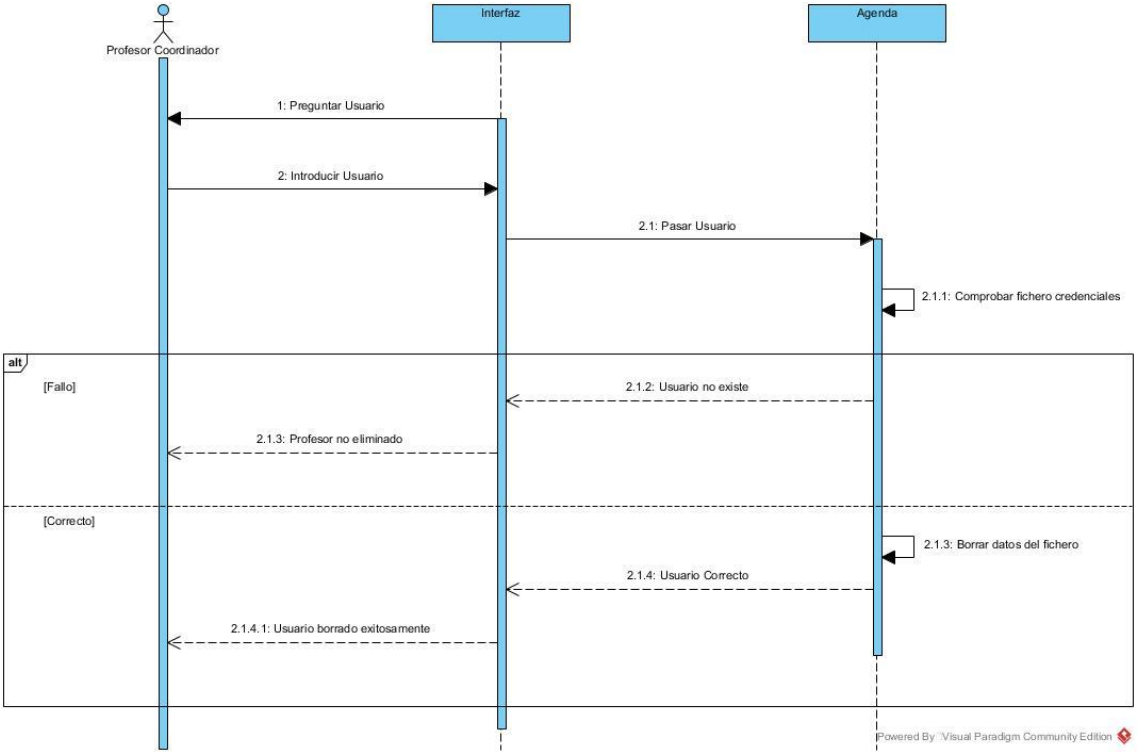
El actor es un usuario sin identificar que trata de acceder a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, la agenda donde almacenaremos los datos y el sistema donde almacenamos los usuarios autorizados para emplear la base de datos de los alumnos.

Añadir Profesor



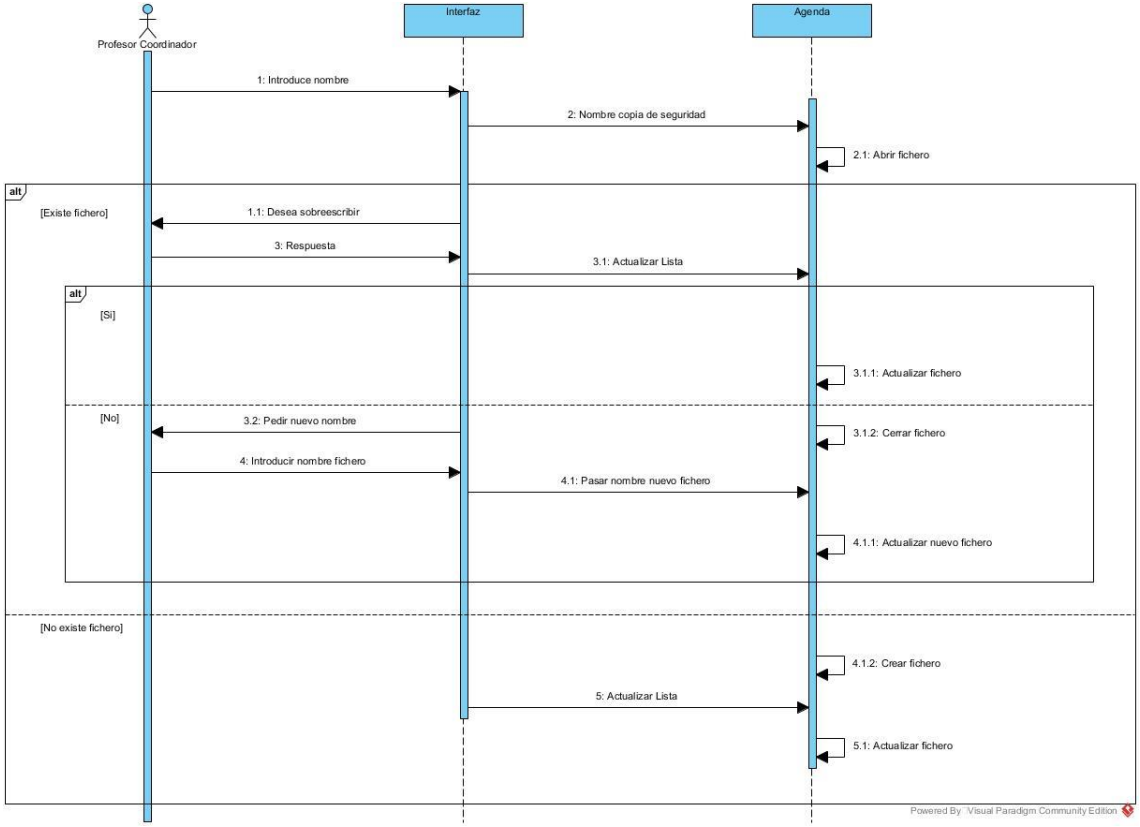
El actor es un profesor (que necesariamente debe ser coordinador en este caso) que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Borrar Profesor



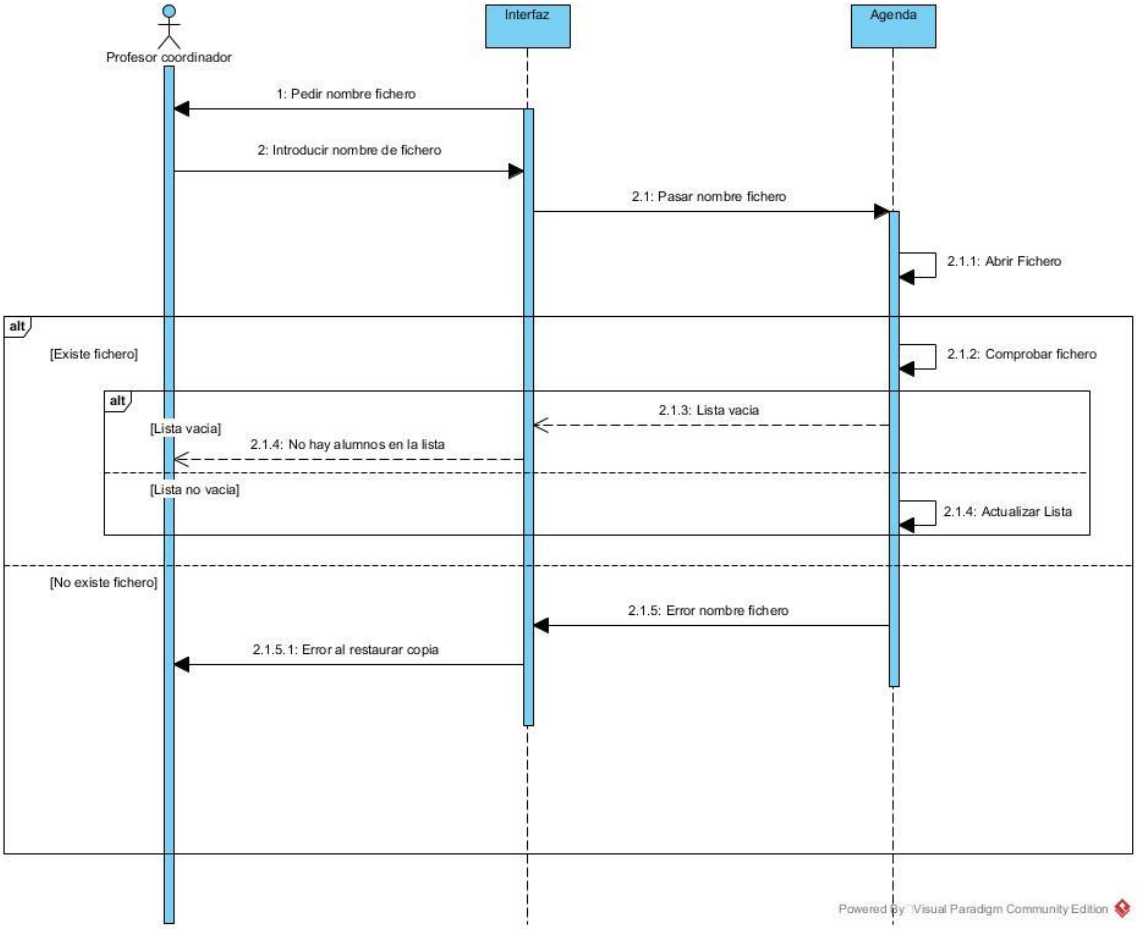
El actor es un profesor (que necesariamente debe ser coordinador en este caso) que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará el profesor, y la agenda donde almacenaremos los datos de los alumnos.

Crear Copia de Seguridad



El actor es un profesor (que necesariamente debe ser coordinador en este caso) que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará este profesor coordinador, y la agenda donde almacenaremos los datos.

Restaurar Copia de Seguridad



El actor es un profesor (que necesariamente debe ser coordinador en este caso) que accede a la base de datos, mientras que los objetos participantes son la interfaz con la que interactuará este profesor coordinador, y la agenda donde almacenaremos los datos.

7 METODOLOGÍA SCRUM

Para el desarrollo del sistema, se ha seguido la metodología SCRUM. SCRUM es un proceso ágil que nos permite centrarnos en ofrecer el más alto valor de negocio en el menor tiempo.

En SCRUM existen varios roles o participantes: "Product Owner", en este caso los profesores de la asignatura, que encargan el sistema. Actuarían como el cliente. "SCRUM Master", el coordinador del proyecto, que en este caso es el representante del grupo (José Manuel Alcalde Llergo). "Team", o equipo de desarrollo, que en este caso se compone de los 3 integrantes del grupo.

En SCRUM, los proyectos avanzan en una serie de Sprints, de dos semanas a un mes, aunque en nuestro contexto, la duración del Sprint ha sido de una semana.

Al comienzo de cada iteración, se tiene una lista de los objetivos/requisitos priorizados. Este documento se conoce como Product Backlog.

En el Product Backlog se recoge una lista de objetivos (en este caso las funcionalidades) priorizados, y con un tiempo estimado para su realización.

Funcionalidad	Prioridad/Tiempo Estimado	Funcionalidad	Prioridad/Tiempo estimado
Ficheros cabecera	Prioridad 0/2 Horas	Función principal	Prioridad 0/ 6 Horas
Añadir profesor	Prioridad 1/3 Horas	Insertar Alumno	Prioridad 1/1 Hora
Guardar	Prioridad 2/1 Hora	Buscar Alumno	Prioridad 2/2 Horas
Autentificar	Prioridad 2/3 Horas	Cargar	Prioridad 2/3 Horas
Hacer copia seguridad	Prioridad 3/3 Horas	Cargar copia seguridad	Prioridad 3/1 Hora
Mostrar Alumno	Prioridad 3/0.5 Horas		
Mostrar todos	Prioridad 4/2 Horas		
Borrar alumno	Prioridad 5/1 Hora	Modificar alumno	Prioridad 5/3 Horas
Borrar todos	Prioridad 6/1.5 Horas	Borrar profesor	Prioridad 6/ 1.5 Horas

Product Backlog del sistema

En nuestro sistema, el tiempo total estimado para el alcance de los objetivos es de 35 horas.

Una vez que se han recogido y especificado los requisitos, se hace una lista de tareas correspondiente a cada Sprint. Este documento se conoce como Sprint Backlog, en el que se asigna cada tarea a un equipo de desarrollo.

En nuestro caso, cada miembro del grupo compone un equipo de desarrollo. Como el sistema se ha desarrollado en 2 Sprints, tenemos 2 Sprint Backlog:

José Manuel Alcalde Llergo	Durante esta primera semana ha realizado las funciones BuscarAlumno, RestaurarCopia y GuardarCopia.
Francisco Bérchez Moreno	Durante esta primera semana ha realizado las funciones InsertarAlumno, AñadirProfesor y MostrarAlumno.
Javier Herrero Porras	Durante esta primera semana ha realizado las funciones Autentificar, Guardar y Cargar.

Sprint Backlog Semana 1

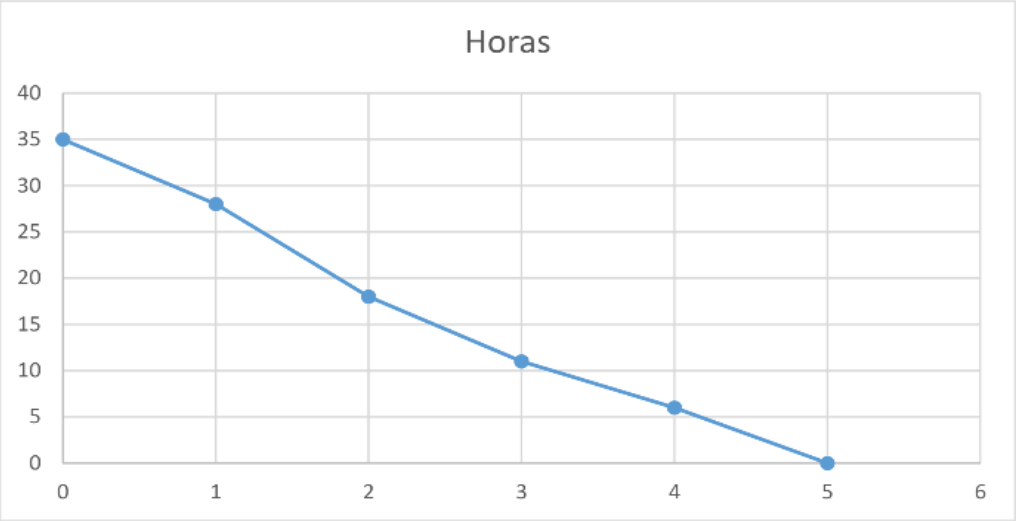
Observación: Durante la primera reunión desarrollamos en conjunto los ficheros de cabecera y las clases necesarias para el sistema.

José Manuel Alcalde Llergo	Durante la segunda semana ha realizado la función principal, y ha corregido los fallos en sus anteriores funciones.
Francisco Bérchez Moreno	Durante la segunda semana ha realizado las funciones ModificarAlumno, BorrarTodo y BorrarProfesor, y ha corregido los fallos en sus anteriores funciones.
Javier Herrero Porras	Durante la segunda semana ha realizado las funciones MostrarTodos y BorrarAlumno, y ha corregido los fallos en sus anteriores funciones.

Sprint Backlog Semana 2

Durante los Sprints se llevan a cabo varios tipos de reuniones: Sprint planning, Sprint review, Sprint retrospective y Daily SCRUM meeting.

En estas reuniones se realiza el llamado Burndown Chart, o gráfico de tareas completadas. En dicho gráfico se puede observar la carga de trabajo completada, y la restante:



Burdown Chart

8 MATRICES DE VALIDACIÓN

Matriz Requisitos Funcionales- Casos de Uso

	CU-01	CU-02	CU-03	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14
RF-01	X														
RF-02		X	X												
RF-03				X											
RF-04		X													
RF-05		X				X									
RF-06		X					X								
RF-07								X							
RF-08									X						
RF-09										X					
RF-10											X				
RF-11												X			
RF-12													X		
RF-13														X	
RF-14															X

Matriz Clases- Casos de Uso

	CU-01	CU-02	CU-03	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13	CU-14
Agenda	X	X	X	X	X	X	X	X	X	X					
Alumno	X	X	X	X	X	X	X	X							
Profesor									X	X	X	X	X	X	X
Persona															

9 BIBLIOGRAFÍA

1. https://moodle.uco.es/m1819/pluginfile.php/23352/mod_resource/content/0/2014-2015/IS-tema05-Resumen_diagramas_UML.pdf
2. http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf
3. http://www.didierperez.com/2012/02/diagrama-de-clases-uml-asociacion/
4. http://www.didierperez.com/2012/02/diagrama-de-clases-uml-asociacion/
5. https://proyectosagiles.org/