Grado en Ingeniería Informática
Ingeniería del Software

Javier Gómez Palmero
Antonio José Lira Alba

ÍNDICE DE CONTENIDO

1. Introducción 2
2. Práctica 2 2
    2.1 Historias de usuario 2
    2.2 Casos de uso 4
    2.3 Análisis de requisitos 13
3. Práctica 3 15
    3.1 Diagrama de clases 16
    3.2 Diagrama de secuencia 22
4. Práctica 4 26
    4.1 Metodología Scrum 26
    4.2 Matriz requisitos funcionales 28
    4.3 Matriz casos de uso 29

1. INTRODUCCIÓN

Para este trabajo simulamos un caso práctico con un cliente, que nos aportaría su idea sobre un proyecto que nosotros tuvimos que llevar a cabo. En este caso el cliente era el profesor, el cual nos dió la información que necesitábamos para hacer el trabajo. Los datos los tuvimos que obtener realizando preguntas al cliente, simulando así una situación que se podría dar con un cliente que no tuviese conocimientos de informática y no supiese aportar de manera clara la información sobre cómo llevar a cabo su proyecto.

2. PRÁCTICA 2

2.1 HISTORIAS DE USUARIO

De la información aportada por el cliente obtuvimos las siguientes historias de usuario:

-Búsqueda: Como usuario quiero poder buscar cualquier alumno que desee dentro de la base de datos. Esta tiene prioridad 1, y tiene los siguientes requisitos:

- Quiero poder realizar la búsqueda mediante el DNI o los Apellidos.
- Quiero como profesor buscar mediante el número de grupo.
- Quiero que se realice una comprobación de que el alumno existe dentro de la base de datos.

-Borrar: Como usuario quiero poder borrar a uno o varios elementos dentro de la base de datos. Esta tiene prioridad 2, y los siguientes requisitos:

- Se borrará un elemento mediante una búsqueda por DNI o Apellidos.
- En caso de que existan dos personas con el mismo apellido preguntar cual es el que se desea eliminar.

-Cargar: Como usuario quiero poder cargar una base de datos o crear una nueva. Esta tiene prioridad 1, y los siguientes requisitos:

- Si deseo crear una base de datos o cargar una , se me deberá preguntar al inciar el sistema software.

-Guardar: Como usuario quiero poder guardar los datos en la base de datos. Esta tiene prioridad 1, y los siguientes requisitos:

- Deseo poder guardar la base de datos modificada, sobrescribiendo la anteriormente guardada.
- Deseo poder crear una nueva base de datos con los datos que haya modificado o almacenado.

-Modificar: Como usuario quiero poder modificar cualquier elemento de la base de datos. Esta tiene prioridad 2, y los siguientes requisitos:

- Para realizar la modificación del elemento primero quiero buscarlo y posteriormente modificarlo.

-Mostrar: Como usuario quiero poder buscar cualquier alumno que desee, todos los alumnos o un grupo específico dentro de la base de datos. Esta tiene prioridad 2, y los siguientes requisitos:

- Quiero poder seleccionar si deseo mostrar toda la base de datos, solo un alumno específico o a un grupo de trabajo.
- Te permitirá buscar por DNI o Apellidos o en caso de grupo por el número de grupo.
- Al seleccionar la opción de todos los alumnos deberá dejarme ordenarlos por Nombre y Apellido o DNI o Curso más alto cursado de mayor a menor y de forma ascendente y descendente.
- Al seleccionar la opción de un alumno específico en caso de existir dos personas con el mismo apellido mostrar las dos y si una persona es líder del grupo que salga resaltada.
- Al seleccionar Mostrar un grupo se debe introducir el numero del grupo y se mostrarán todas las personas pertenecientes a ese grupo.

-Añadir: Como usuario quiero poder añadir un elemento a la base de datos. Esta tiene prioridad 1, y los siguientes requisitos:

- Quiero que sean necesario todos los datos del alumno excepto si pertenece a algún grupo o si es líder de alguno.
- Quiero que se compruebe si el alumno que se quiere añadir ya pertenece a la base de datos y para ello se comprobará el DNI.
- Si el elemento que se añade, pertenece a un grupo y es líder, se comprobará si dentro de ese grupo ya existe un líder.

2.2 CASOS DE USO

De la información aportada por el cliente obtuvimos los siguientes casos de uso:

-CU 1.-Búsqueda:

- Breve descripción: El sistema permitirá buscar un elemento de la base de datos.
- Actores principales: Usuario.
- Actores secundarios: Base de datos.
- Precondiciones:
  Deberá haber sido creada o cargada dentro del sistema.
- Flujo Principal:
  El caso de uso empieza cuando el usuario desee realizar una búsqueda.
  El sistema enviará la información a una de las funciones que las necesite.
- Postcondiciones:
  La función debe devolver los alumnos que cumplan las características de la búsqueda.
- Flujos alternativos:
  En caso de querer mostrar un alumno esta información será proporcionada a la función de mostrar.

En caso de querer eliminar a un alumno la información será pasada a la función de borrar.
En caso de querer modificar un alumno la información será pasada a la función de modificar.

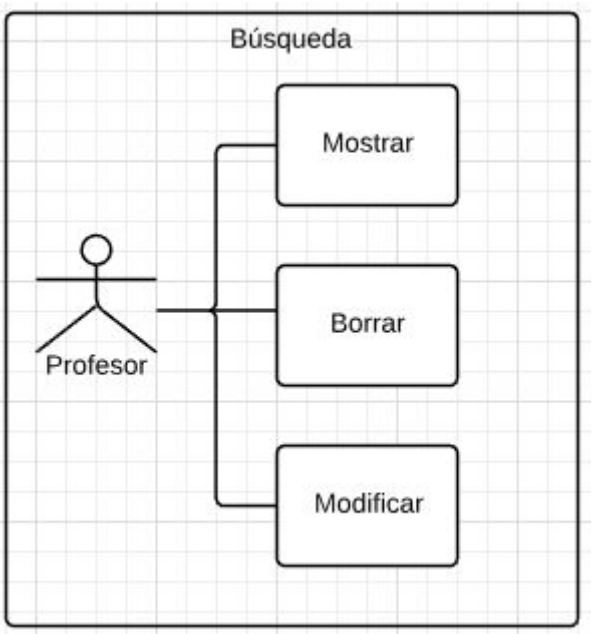


Diagrama de casos de uso de buscar

-CU 1.1-Mostrar:

- Breve descripción: El sistema mostrará un alumno por terminal.
- Actores principales: Usuario.
- Actores secundarios: Alumno.
- Precondiciones:
  El alumno deberá estar incluido dentro de la base de datos.
- Flujo Principal:
  El caso de uso empieza cuando el sistema necesite mostrar a un alumno.
  El sistema muestra todos los datos del alumno.
- Postcondiciones:
  La función mostrará los alumnos por terminal.
- Flujos alternativos:
  Si no existe el Alumno, se deberá mostrar un mensaje de error.

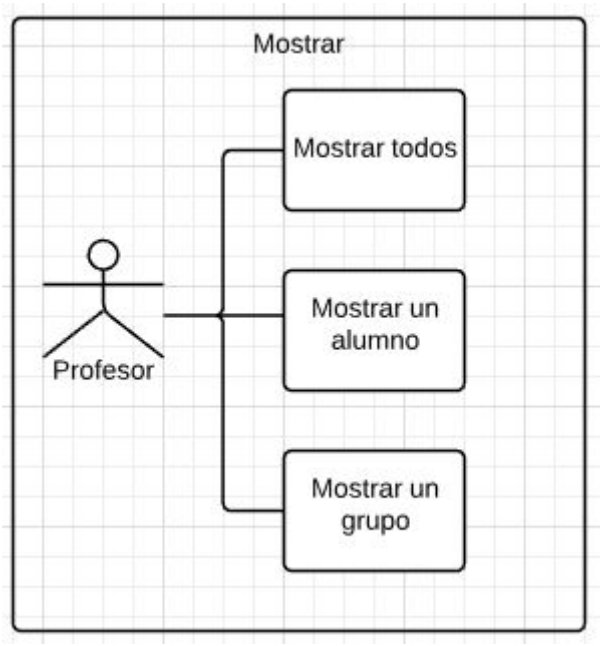


Diagrama de casos de uso de guardar

-CU 1.2-Borrar:

- Breve descripción: El sistema permitirá eliminar un elemento de la base de datos.
- Actores principales: Usuario.
- Actores secundarios: Base de datos.
- Precondiciones:
  Deberá haber sido creada o cargada dentro del sistema y contener al menos un elemento.
  Se habrá tenido que realizar una búsqueda del elemento a borrar mediante la dicha función.
- Flujo Principal:
  El caso de uso empieza cuando el usuario quiera eliminar un elemento.
  El sistema habrá eliminado el elemento en la base de datos.
- Postcondiciones:
  El número de usuarios será el número de usuarios anteriores menos uno.
  Si se ha borrado un alumno líder se deberá nombrar otro en el grupo.

- Flujos alternativos:
  En caso de haber dos personas con el mismo apellido se seleccionara cual es la indicada a borrar.

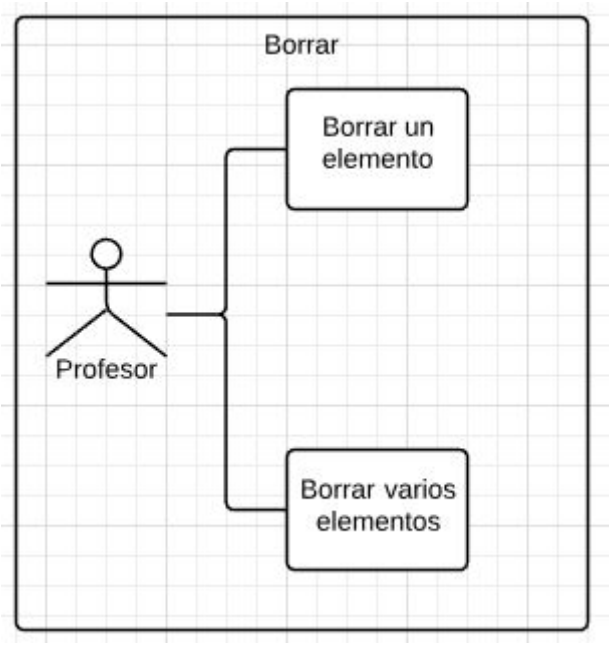


Diagrama de casos de uso de guardar

-CU 1.3-Modificar:

- Breve descripción: El sistema permitirá que se modifiquen los datos de un elemento de la base de datos.
- Actores principales: Usuario.
- Actores secundarios: Alumno.
- Precondiciones:
  Se deberá haber realizado una búsqueda dentro del sistema.
  El elemento a modificar exista dentro de la base de datos.
- Flujo Principal:
  El usuario solicita la modificación de un alumno.
  El usuario elegirá el alumno que desea modificar.
- Postcondiciones:
  El elemento modificado debe ser distinto del original.
- Flujos alternativos:

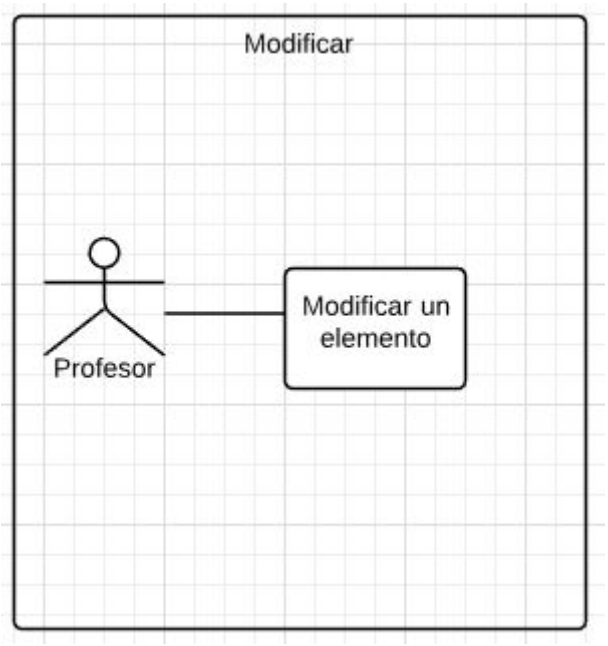


Diagrama de casos de uso de modificar

-CU 2-Cargar:

- Breve descripción:El sistema permitirá cargar o crear una nueva base de datos.
- Actores principales: Usuario.
- Actores secundarios: Sistema.
- Precondiciones:
  En caso de querer cargar la base de datos , se deberá tener el archivo a cargar.
- Flujo Principal:
  El caso de uso empieza cuando se inicie el sistema.
  El usuario elegirá la opción que desee.
- Postcondiciones:
  Los campos de la base de datos han sido cargados.
- Flujos alternativos:
  En caso de elegir crear la base de datos deberá introducir los datos manualmente.
  En caso de elegir cargar la base de datos mediante un archivo, se deberá seleccionar el archivo indicado.

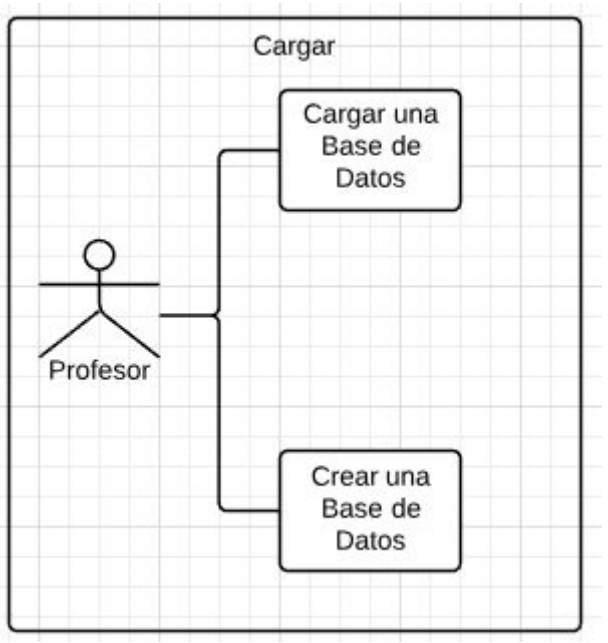


Diagrama de casos de uso de cargar

-CU 3-Añadir:

- Breve descripción: El sistema permitirá la agregación de un nuevo elemento a la base de datos.
- Actores principales: Usuario.
- Actores secundarios: Base de datos.
- Precondiciones:
  Tener toda la información del alumno.
- Flujo Principal:
  El usuario solicita la agregación de un nuevo alumno.
  El usuario rellena todos los campos necesarios para ello.
  El nuevo alumno ha sido añadido.
- Postcondiciones:
  El número de alumnos será igual al número de alumnos anteriores más uno.
- Flujos alternativos:
  En caso de elegir crear la base de datos deberá introducir los datos manualmente.
  En caso de que exista el alumno dentro de la base de datos (Se comparara el DNI) se abortara la acción.

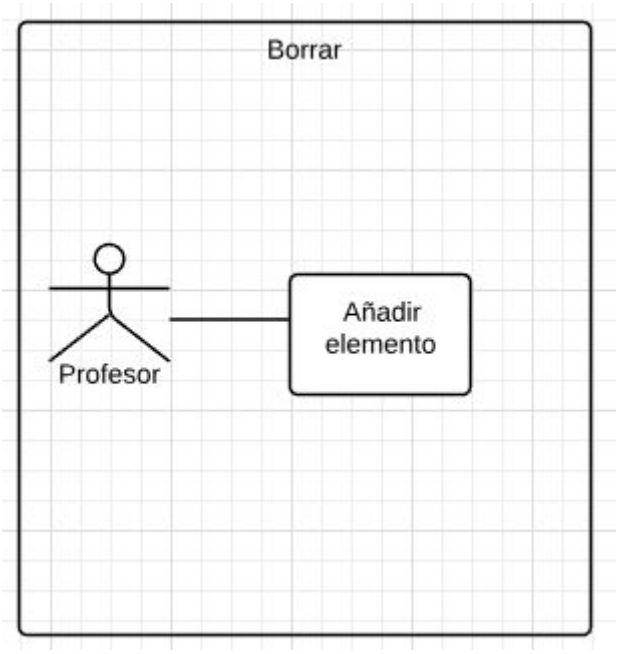


Diagrama de casos de uso de añadir

-CU 4-Guardar:

- Breve descripción: El sistema permitirá guardar los campos modificados o creados.
- Actores principales: Usuario.
- Actores secundarios: Sistema.
- Precondiciones:
  Haber modificado o creado algún campo dentro de la base de datos.
- Flujo Principal:
  El caso de uso empieza cuando se inicie el sistema.
  El usuario deseara guardar la información.
- Postcondiciones:
  El archivo habrá sido sobrescrito o creado.
- Flujos alternativos:
  En caso de elegir crear la base de datos deberá, se creará un nuevo archivo.
  En caso de elegir sobrescribir un archivo, se tendrá que seleccionar el archivo.

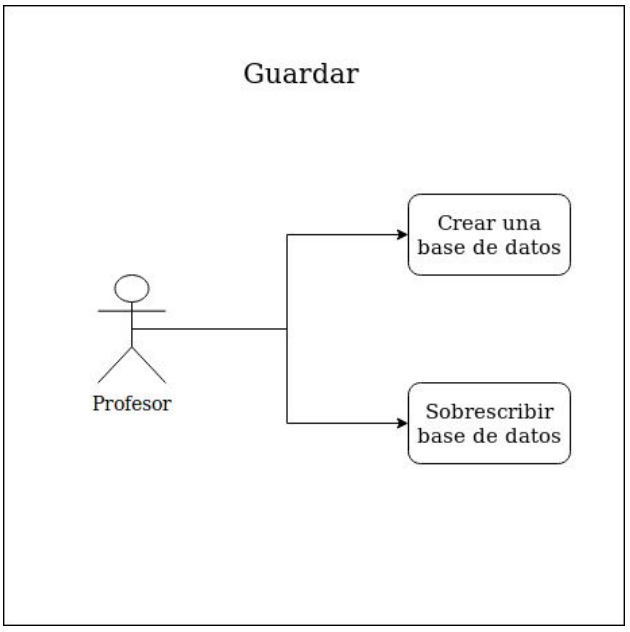


Diagrama de casos de uso de guardar

A partir de esto obtuvimos el diagrama general de casos de uso.

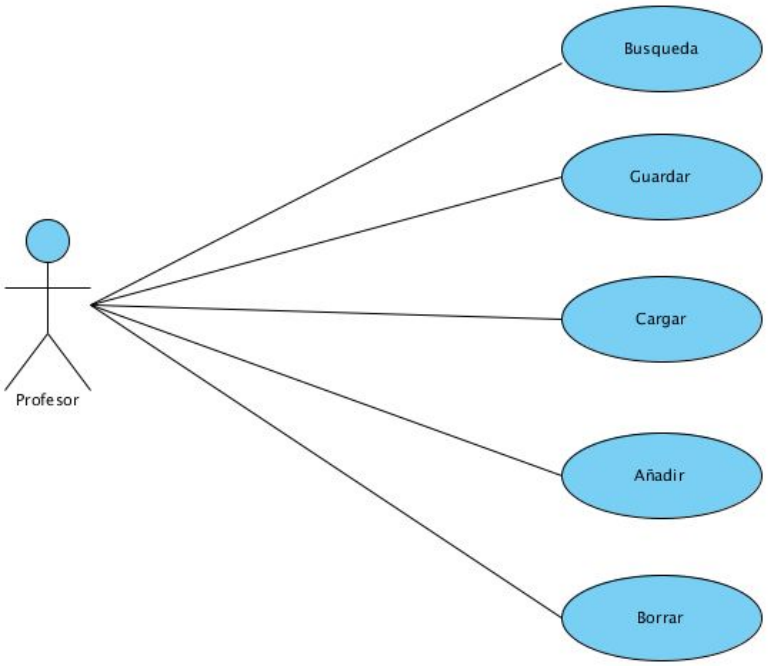


Diagrama general de casos de uso

2.3 ANÁLISIS DE REQUISITOS

De la información aportada por el cliente obtuvimos el siguiente análisis de requisitos:

-Requisitos no funcionales:

Indican cómo debería ser el sistema. Restricciones que afectan al sistema.

- El sistema software deberá almacenar en una base de datos todos los alumnos que estan matriculados en la asignatura.
- El número máximo de alumnos es de 150.
- La base de datos se visualizara por linea de comandos y se pedirá si se desea visualizar por HTML o por Markdown.
- La interfaz será la linea de comandos, aunque también se podrá crear una interfaz gráfica.
- Los alumnos se guardarán en un fichero binario.
- El sistema operativo será Linux.
- A la base de datos solo podrá acceder el profesor.
- El lenguaje de documentación ser Markdown.

-Requisitos funcionales:

Expresa lo que debería hacer el sistema. Las funcionalidades que este debe proporcionar y especifican la manera en que el sistema debe reaccionar antes determinadas entradas y/o el comportamiento del mismo en determinadas situaciones.

- 1.-Búsqueda

  El usuario tiene que poder buscar cualquier alumno que desee dentro de la base de datos.

  Para la búsqueda se usará el DNI, los apellidos (se comprobará que el alumno existe) o por grupo y se transmitirá la información a las demás funciones.
- 2.-Mostrar

  El usuario tiene que poder buscar cualquier alumno que desee, todos los alumnos, o un grupo específico dentro de la base de datos.

Al seleccionar la opción de nombrar se debe especificar si se desea mostrar todo o si se desea mostrar uno en específico.
Si se desean mostrar todos se deben de mostrar ordenados o por nombre, o apellidos (alfabeticamente), o por DNI, o por curso más alto matriculado de mayor a menor (descendente o ascendente).
Si se desea mostrar uno solo se debe mostrar al introducir DNI o apellidos (en caso de tener los mismos apellidos, mostrar todos los que coincidan). En caso de que uno sea líder, este debe de estar resaltado.
Si se desea mostrar por grupo se debe introducir el número del grupo y se mostrarán todos los alumnos que pertenezcan a ese grupo.

- 3.-Borrar x
  Como usuario quiero poder borrar a uno o varios elementos dentro de la base de datos.
  Se borrará un elemento mediante una búsqueda por DNI o apellidos.
  En caso de de que existan dos personas con el mismo apellido preguntar cual es el que se desea eliminar.
- 4.-Modificar
  El usuario tiene que poder modificar cualquier elemento de la base de datos.
  Para realizar la modificación del elemento primero quiero buscarlo y posteriormente modificarlo.
- 5.-Cargar
  El usuario tiene que poder cargar una base de datos o crear una nueva.
  Si se desea crear una base de datos o cargar una , se me deberá preguntar al inciar el sistema software.
- 6.-Añadir
  El usuario tiene que poder añadir un elemento a la base de datos.

Se tiene que comprobar si al alumno que se quiere añadir ya pertenece a la base de datos y para ello se comprobará el DNI.

Si el elemento que se añade, pertenece a un grupo y es líder, se comprobará si dentro de ese grupo ya existe un líder.

- 7.-Guardar

  El usuario tiene que poder guardar los datos modificados de la base de datos.

  Se tiene que sobrescribir la base de datos abierta en caso de que haya una.

  Se deberá crear una base de datos nueva en caso de que no haya ninguna anteriormente abierta.

3. PRÁCTICA 3

Una vez obtenidos los casos de uso, los requisitos y las historias de usuarios, pudimos elaborar el diagrama de secuencia y el diagrama de clases.

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML.

un diagrama de clases en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

3.1 DIAGRAMA DE CLASES

-Agenda

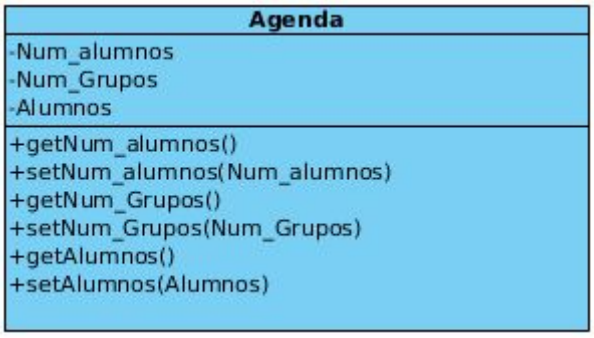
Clase que recoge toda la información de todos los alumnos y la almacena para posteriormente trabajar con ella.

Atributos:

- Nº Alumnos: Número de alumnos dentro de la lista de alumnos.
- Nº_Grupos: Número de grupos formados por alumnos dentro de la lista de alumnos.
- Alumnos: Todos los datos de los alumnos dentro de la lista de alumnos.

Operaciones:

- getNum_alumnos: Función que nos devolverá el número de alumnos que hay en la lista.
- setNum_alumnos: Función que nos permitirá definir el número de alumnos que hay en la lista.
- getNum_grupos: Función que nos devolverá el número de grupos que hay en la lista.
- setNum_grupos: Función que nos permitirá definir el número de grupos que hay en la lista.
- get_Alumnos: Función que nos devolverá el nombre de todos los alumnos de la lista.
- set_Alumnos: Función que nos permitirá definir el nombre de los alumnos de la lista.



clase agenda

-Alumno

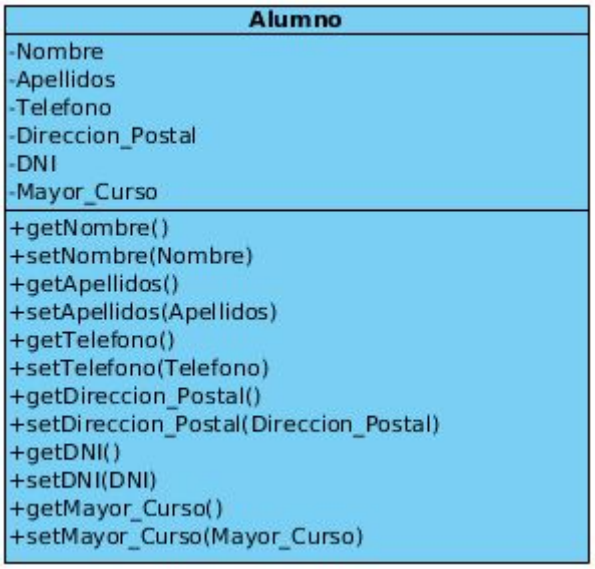
Clase que recoge los datos de cada individuo dentro de la base de datos

Atributos:

- Nombre: Nombre del alumno.
- Apellidos: Apellidos del alumno.
- Teléfono: Número del teléfono de contacto del alumno.
- Direccion_Postal: Dirrección postal de su actual vivienda.
- DNI: Número del documento nacional de identidad del alumno sin tener en cuenta la letra.
- Mayor_Curso: Curso más alto cursado por el alumno este año.

Operaciones:

- getNombre: Función que devuelve el nombre del alumno.
- setNombre: Función que permite establecer el nombre del alumno.
- getApellidos: Función que devuelve los apellidos de un determinado alumno.
- setApellidos: Función que permite establecer los apellidos de un determinado alumno.
- getTelefono: Función que devuelve el número de teléfono de un determinado alumno.
- setTelefono: Función que permite establecer el número de teléfono de un alumno.
- getDireccion_Postal: Función que permite obtener la dirección postal de un alumno.
- setDireccion_Postal: Función que permite establecer la dirección postal de un alumno.
- getDNI: Función que permite obtener el DNI de un alumno.
- setDNI: Función que permite establecer el DNI de un alumno.
- getMayor_Curso: Función que permite obtener el curso más alto cursado por un alumno.
- setMayor_Curso: Función que permite establecer el curso más alto cursado.



clase alumno

-Grupo

Clase que recoge los datos de un conjunto de alumnos que forman parte del mismo grupo.

Atributos:

- Num_alumnos_grupo: Número de alumnos que forman el grupo.
- DNI_Lider: DNI del líder del grupo.
- Num_Grupo: Número identificador del grupo.
- Alumnos_Grupo: Nombre de los alumnos que conforman el grupo.

Operaciones:

- getNum_alumnos_grupo: Función que devuelve el número de alumnos que forman el grupo.
- setNum_alumnos_grupo: Función que nos permite establecer el número de alumnos que hay en un grupo.
- getDNI_lider: Función que permite obtener el DNI del líder del grupo.
- setDNI_lider: Función que permite establecer el DNI del líder del grupo.
- getNum_Grupo: Función que permite obtener el número identificador del grupo.

- setNum_Grupo: Función que permite establecer el número indetificador del grupo.
- getAlumnos_Grupo: Función que permite obtener el número de alumnos que hay en el grupo.
- setAlumnos_Grupo: Función que permite establecer el número de alumnos que hay en el grupo.

clase grupo

-Profesor

Clase que sera la principal dentro de nuestro sistema.Es la encargada de trabajar con la base de datos.

Atributos:

- *Agenda: Puntero a la agenda.
- Rol: Número del documento nacional de identidad del profesor sin tener en cuenta la letra.
- Credenciales: Número de identificación del profesor

Operaciones:

- GetRol: Función que permite eliminar un alumno que ya esté dentro de la base de datos.
- SetRol: Función que permite establecer el rol de un profesor.
- GetCredenciales: Función que permite obtener el credencial.
- SetCredenciales: Función que permite modificar o introducir un credencial.
- CargarBackup: Función que permite cargar una copia de seguridad ya existente.

- GuardarBackup: Función que permite guardar una copia de seguridad.
- Cargar_BDD: Función que permite cargar una base de datos ya existente.
- Crear_BDD: Función que permite crear una nueva base de datos.
- Get_*agenda: Función que devuelve la posición del puntero en la agenda.

clase profesor

A partir de los siguientes diagramas realizamos un diagrama general de clases.

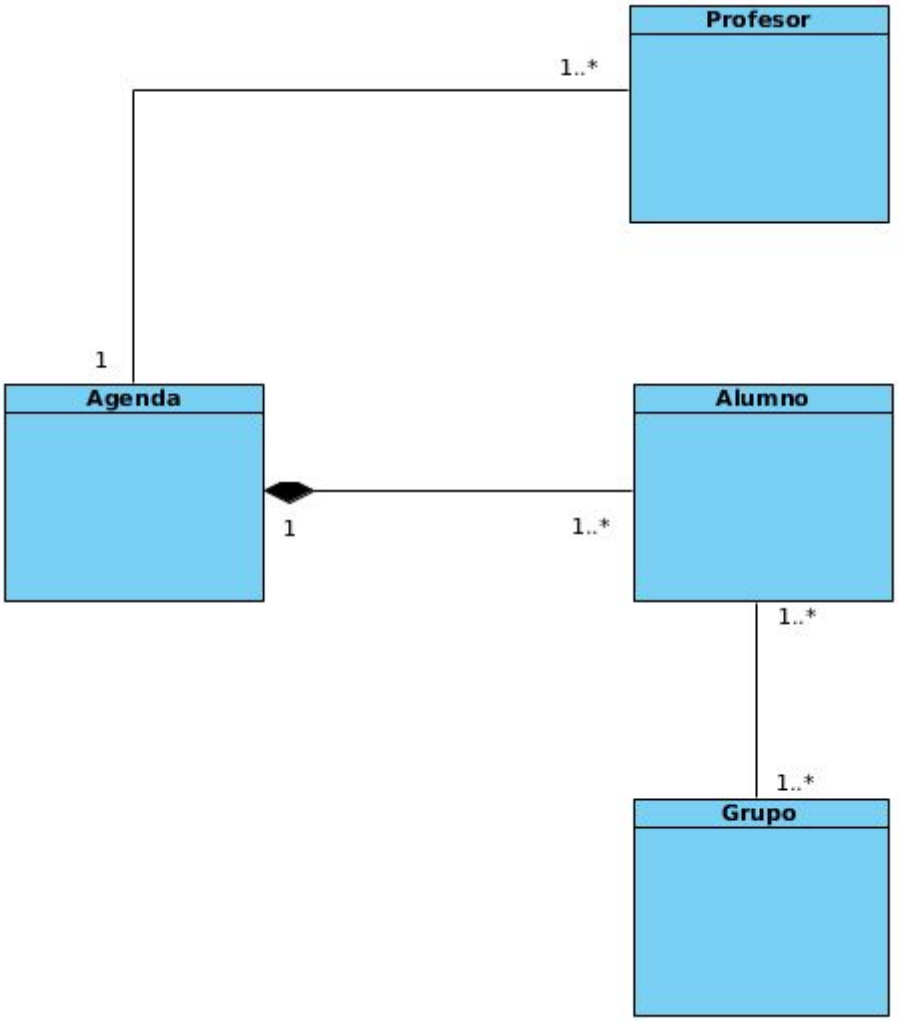


diagrama general de clases

3.2 DIAGRAMA DE SECUENCIA

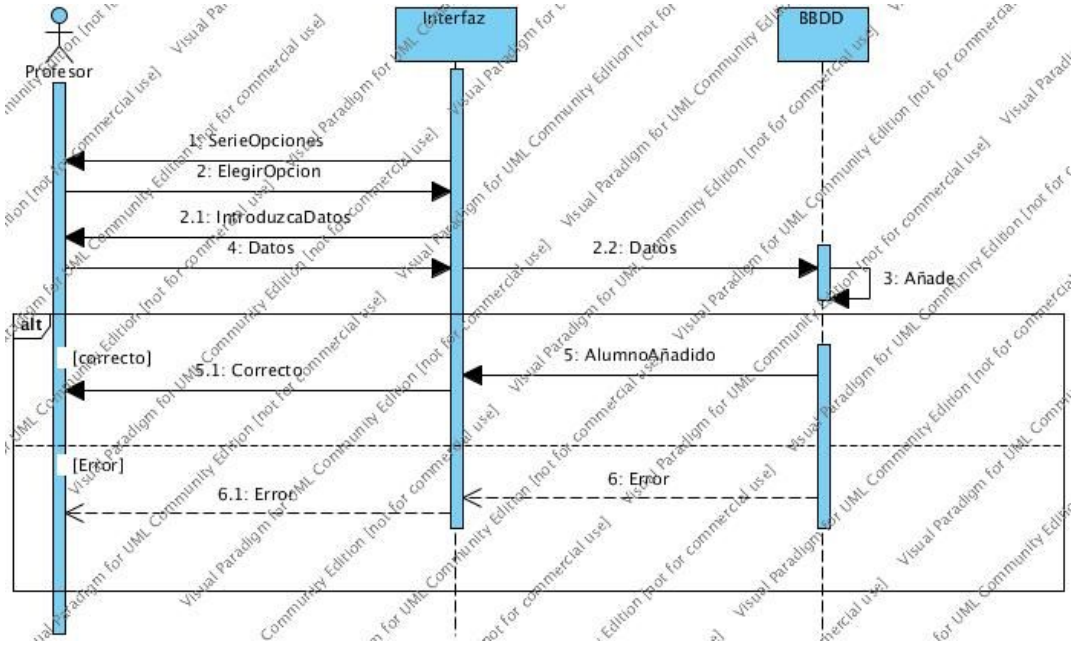


Diagrama de secuencia de Añadir

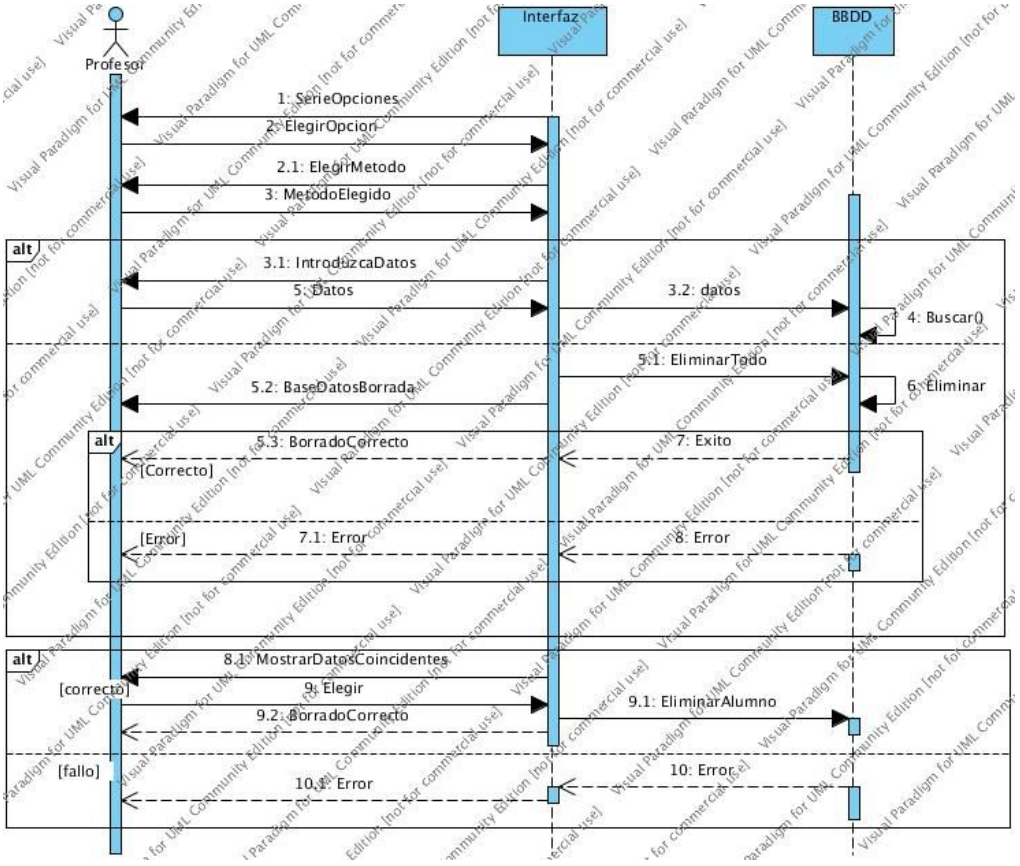


Diagrama de secuencia de Borrar

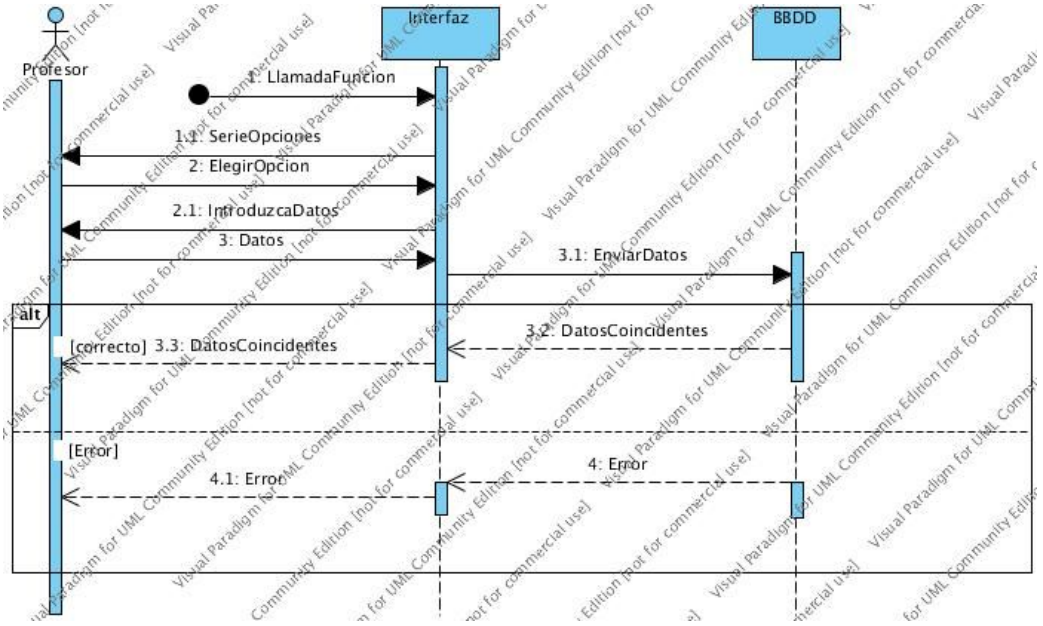


Diagrama de secuencia de Búsqueda

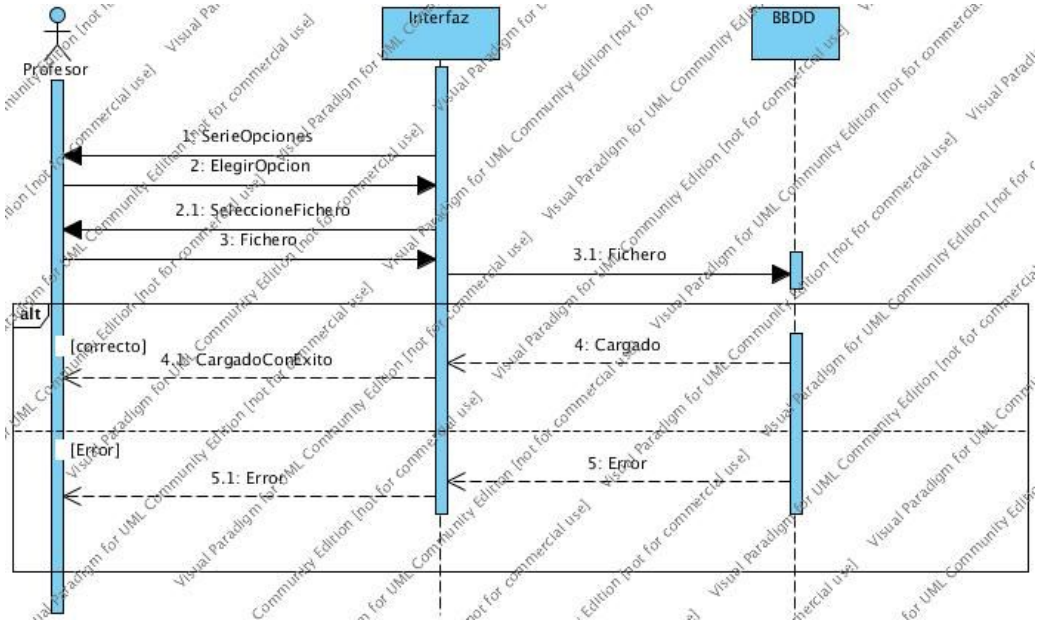


Diagrama de secuencia de Cargar

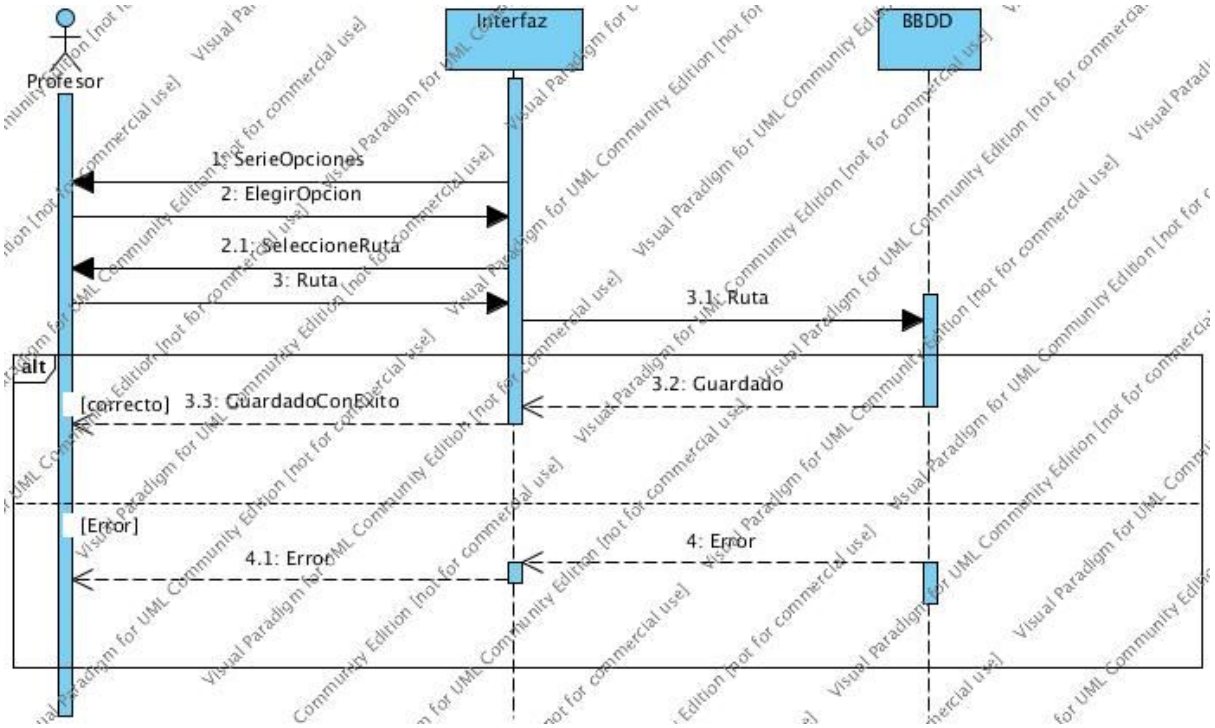


Diagrama de secuencia de Guardar

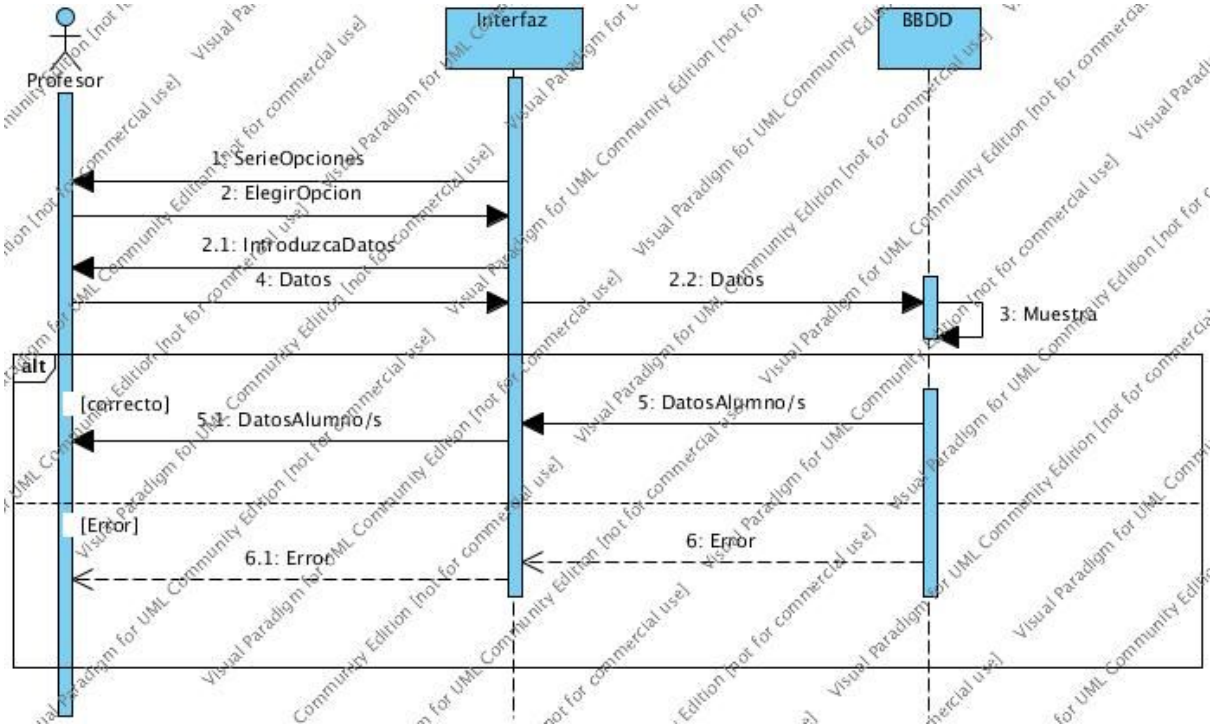


Diagrama de secuencia de Mostrar

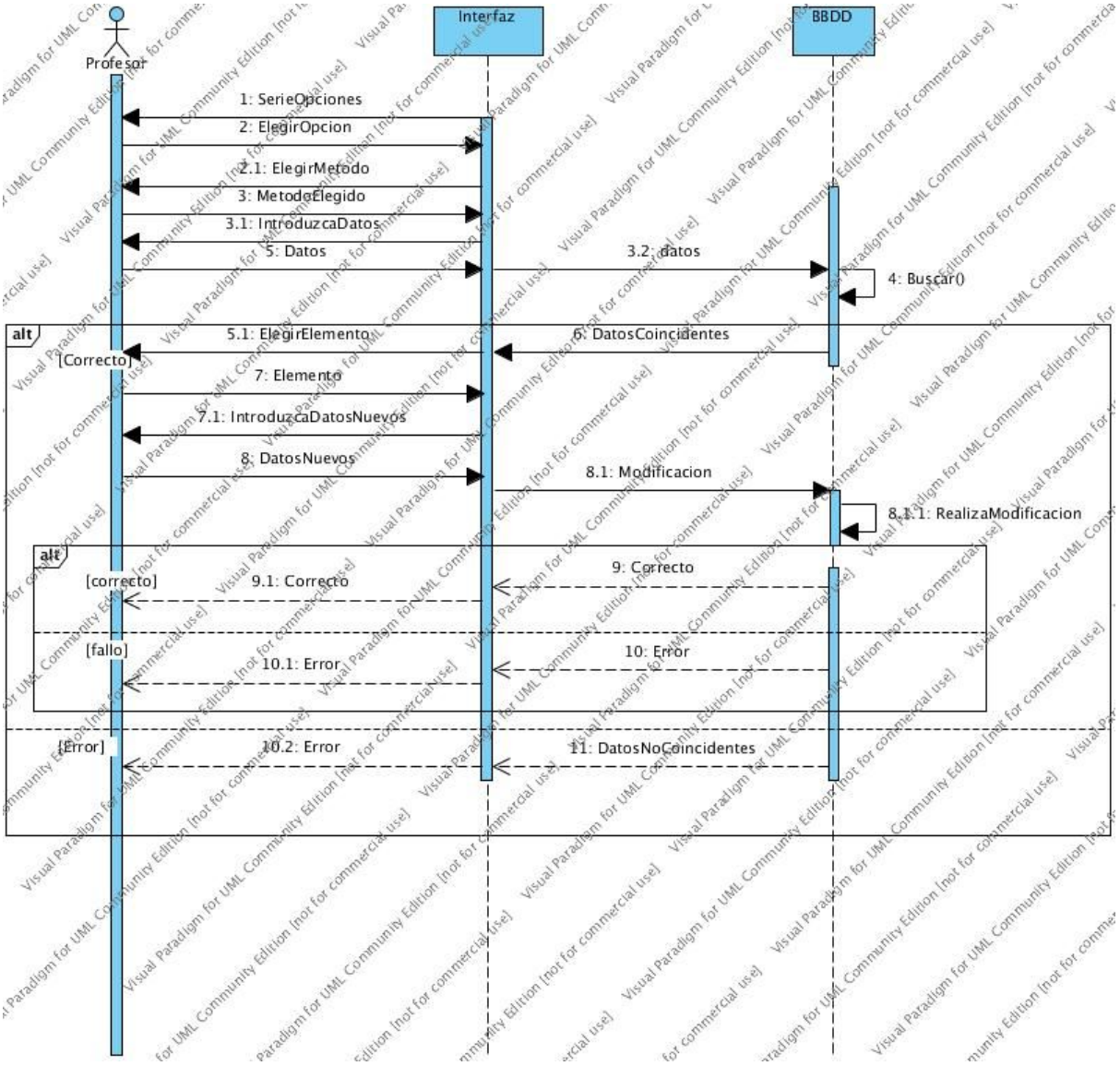


Diagrama de secuencia de Modificar

4. PRÁCTICA 4

4.1 METODOLOGÍA SCRUM

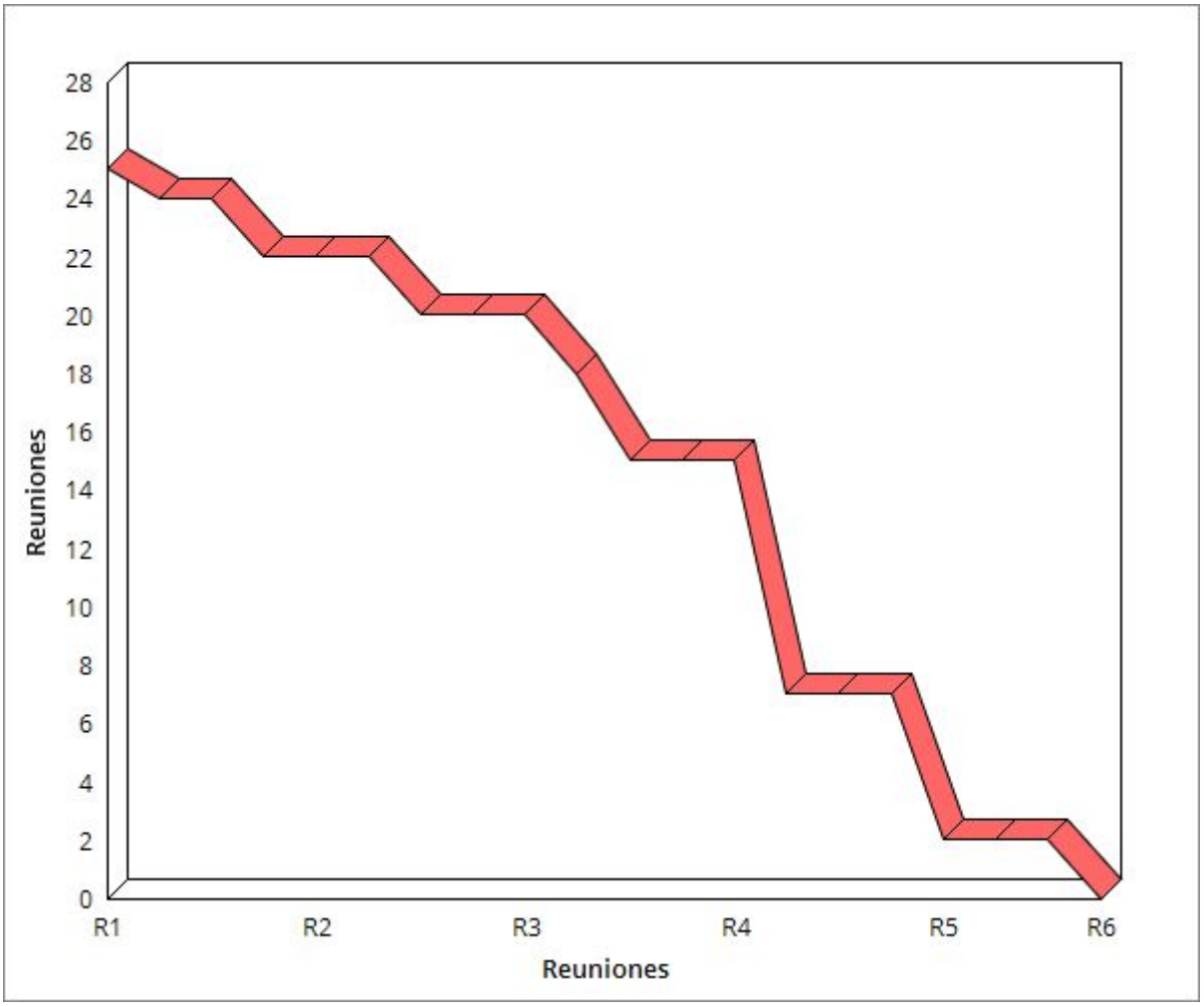
Por último realizamos la implementación del código, pero previamente organizamos el trabajo siguiendo la metodología scrum.
Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.
En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.
En este proceso creamos tres ficheros:

-Sprint Backlog:
Lista de tareas que el equipo elabora en la reunión de planificación de la iteración (Sprint planning) como plan para completar los objetivos/requisitos seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración, en forma de incremento de producto preparado para ser entregado.

-Product Backlog:
Es una lista de Historias de Usuario, ordenadas según el valor de negocio que establece el Dueño del Producto, y que trata de cubrir todas las funcionalidades necesarias. El product backlog se puede ver desde la perspectiva de una iteración o sprint, de una release o de todo el producto.

-Burndown Chart:

Es una representación gráfica del trabajo por hacer en un proyecto en el tiempo. Usualmente el trabajo remanente (o backlog) se muestra en el eje vertical y el tiempo en el eje horizontal. Es decir, el diagrama representa una serie temporal del trabajo pendiente. Este diagrama es útil para predecir cuándo se completará todo el trabajo.



Gráfica Burndown Chart

4.2 MATRIZ REQUISITOS FUNCIONALES

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.

Requisitos Funcionales	Casos de uso
RF1	CU1
RF2	CU1.1
RF3	CU1.2
RF4	CU1.3
RF5	CU2
RF6	CU3
RF7	CU4

4.3 MATRIZ CASOS DE USO

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

Clases	Casos de uso
Agenda	CU2,CU4
Alumno	CU1,CU1.1,CU1.2,CU1.3,CU2, CU3,CU4
Grupo	CU1,CU1.2,CU1.3,CU3
Profesor	CU1,CU1.1,CU1.2,CU1.3,CU2, CU3,CU4