# DOCUMENTO FINAL DEL PROYECTO:

# ÍNDICE:



# INTEGRANTES DEL GRUPO:

1. Ignacio Antonio Ruiz Martin-Romo
2. Alejandro Siles Jiménez
3. Jorge Ramírez Galván

## 1.Definición del problema:

Nuestro cliente nos ha encargado una base de datos en el cual debe estar los datos de los alumnos de la clase de Ingeniería de Software. Y que pueda manipular dicha base de datos pudiendo insertar, buscar, borrar y modificar alumno y lo mismo con los grupos.

## 2. Requisitos:

Los atributos que almacena dicha base de datos son:

- DNI del alumno
- Nombre del alumno
- Apellidos del alumno
- Teléfono
- Email
- Código Postal
- Fecha de nacimiento
- Nº de grupo
- Líder de un grupo o no

*** Requisitos Funcionales:**

Los requisitos funcionales son los siguientes:

- Login
- Buscar alumno por DNI
- Insertar alumno
- Modificar alumno
- Mostrar base de datos
- Borrar alumno
- Borrar grupo
- Mostrar grupo
- Modificar integrantes
- Modificar lider
- Borrar base de datos
- Copia de seguridad

*** Requisitos no funcionales**

Los requisitos no funcionales son:
- Debe funcionar en Linux.
- Se trabaja con ficheros binarios.
- Los profesores pueden ser coordinadores o ayudantes.
- El máximo de alumnos es de 150
- Para poder manipular un alumno se necesita el DNI de dicho alumno.

**3. Historias de Usuario:**

Función: *BuscarAlumno*

**(ANVERSO)**

Apellidos: López Jiménez | *Buscar alumno*.

DNI: 31025106V | *Buscar alumno*.

Como usuario, quiero poder buscar un alumno por sus apellidos o DNI y que el programa me muestre la información del alumno.

**(REVERSO)**

- Quiero poder visualizar todos los datos de un único alumno.
- El programa deberá pedir el DNI si al introducir apellidos se encuentra más de un alumno con los mismos apellidos.
- Si al introducir DNI el alumno no está en la base de datos, se mostrará un mensaje.

Función: InsertarAlumno.

**(ANVERSO)**

Como usuario quiero insertar un nuevo alumno en la base de datos, recogiendo antes todos los datos necesarios.

**(REVERSO)**

- Quiero introducir un nuevo usuario en la base de datos.
- El hará una llamada a la funcion Buscar_alumnos, si el alumno que se desea introducir no existe se pedirán los datos.
- El sistema introduce los datos de nuevo alumno en la base.

Funcion: *ModificarAlumno*

**(ANVERSO)**

Apellidos: López Jiménez | *Modificar_alumno alumno*.

DNI: 31025106V | *Modificar alumno*.

Como usuario, quiero poder buscar un alumno por sus apellidos o DNI y que el programa me permita modificar todos los campos del alumno a excepción del DNI, nombre y apellidos.

**(REVERSO)**

- Quiero poder modificar todos los datos de un único alumno.
- El programa deberá pedir el DNI si al introducir apellidos se encuentra más de un alumno con los mismos apellidos.
- Si al introducir DNI el alumno no está en la base de datos, se mostrará un mensaje.

Funcion: *BorrarAlumno*

**(ANVERSO)**

Apellidos: López Jiménez | *Borrar alumno*.

DNI: 31025106V | *Borrar alumno*.

Como usuario, quiero poder buscar un alumno por sus apellidos o DNI y que el programa borre el alumno de la base de datos

**(REVERSO)**

- Quiero poder eliminar todos los datos de un único alumno.
- El programa deberá pedir el DNI si al introducir apellidos se encuentra más de un alumno con los mismos apellidos.
- Si al introducir DNI el alumno no está en la base de datos, se mostrará un mensaje.

Función: *CambioLider.*

**(ANVERSO)**

Como usuario quiero cambiar el líder de un grupo o asignar uno en caso de que no haya.

**(REVERSO)**

- Quiero cambiar el líder de un equipo.
- El programa primero comprobara si ya existe un líder para cambiar el asignado o asignar uno nuevo.

Función: BorrarBaseDatos

**(ANVERSO)**

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

**(REVERSO)**

- Quiero poder ejecutar el programa y que se limpie toda la base de datos.

Función: *CopiaSeguridad.*

**(ANVERSO)**

Como usuario quiero hacer una copia de todos los datos de la base a un archivo binario.

**(REVERSO)**

- Se crea un nuevo fichero binario y todos los datos de la base se almacenan en este.

Función: *MostrarBaseDatos*

**(ANVERSO)**

Como usuario quiero poder visualizar todos los alumnos almacenados en la base de datos, ordenados según, alfabéticamente, curso más alto matriculado y DNI. De manera tanto ascendente como descendente.

**(REVERSO)**

- Quiero poder visualizar todos los alumnos almacenados en la base de datos ordenandos.
- El programa pedirá los datos según de la forma que se desee ordenar.
- Mostrará los alumnos ordenados de la forma deseada.

Función: *Login*

**(ANVERSO)**

Como usuario quiero identificarme para obtener los distintos tipos de permisos de uso del sistema.

**(REVERSO)**

- Quiero entrar en el menú del sistema.
- Este pide por pantalla un código de identificación.
- Se introduce el código y el sistema accede con los distintos tipos de permisos, si el código no coincide con ningún tipo de permiso el sistema muestra un mensaje de error y pide de nuevo el código.

Función: BorrarGrupo.

**(ANVERSO)**

Como usuario quiero poder eliminar un grupo disolviendo sus integrantes.

**(REVERSO)**

- Quiero eliminar un grupo.
- El programa busca si el grupo indicado existe y lo elimina.

Función: ContarMiembros.

**(ANVERSO)**

Como usuario quiero conocer el número de miembros de un grupo.

**(REVERSO)**

- Quiero conocer el número de miembros de un grupo.
- El programa comprueba que el grupo que se ha introducido existe en la base de datos y cuenta su número de integrantes.
- El sistema devuelve el número de integrantes.

Función: ModificarIntegrantes.

**(ANVERSO)**

Como usuario quiero poder añadir y borrar integrantes de los grupos deseados.

**(REVERSO)**

- Quiero modificar los integrantes de un grupo.
- El programa hace una llamada a BuscarAlumno para comprobar si el alumno que desea modificar existe en la base de datos.
- El sistema modifica el alumno deseado.

Función: MostrarGrupo.

**(ANVERSO)**

Como usuario quiero poder visualizar los alumnos y sus datos de un respectivo grupo.

**(REVERSO)**

- Quiero visualizar los datos de los alumnos que componen determinado grupo.
- El programa busca si el grupo indicado existe y muestra por pantalla los datos de sus integrantes.

**4. Casos de Uso:**

***~~Buscar alumno~~***

*Apellidos:* López Jiménez | *DNI:* 31025106V

*Actores principales:* usuario

*Actores secundarios:* alumno

**Precondiciones:**

- El alumno debe existir en la base de datos.

**Flujo principal:**

1. El caso de uso empieza cuando el sistema necesita mostrar un alumno.
2. Se muestran todos los datos correspondientes al alumno.

**Postcondiciones:**

- El sistema muestra al alumno y todos sus datos por pantalla.

**Flujos alternativos:**

- 2a: Si al introducir los apellidos del alumno se encuentran coincidencias, el el programa pedirá el DNI.
- 2b: Si al introducir DNI el alumno no se encuentra en la base de datos, se mostrará un mensaje de error.

**Insertar alumno**

*Breve descripción:* Serán pedidos todos los requerimientos tanto obligatorios como opcionales que se rellenan a la hora de crear un nuevo alumno en la base por el sistema, creando existe a ese nuevo alumno.

*Actores principales:* usuario

*Actores secundarios:* equipo, alumno

**Precondiciones:**

- Se hace una llamada a BuscarAlumno y empieza el programa.

**Flujo principal:**

1. El caso de uso comienza cuando se desea introducir un nuevo alumno en la base de datos.
2. El programa pide los datos necesarios para crear ese alumno.
3. El nuevo alumno queda registrado en la base de datos.

**Postcondiciones:**

- Carga el menú de programa de nuevo.

**Flujos alternativos:**

2.1 Si alguno de los datos no ha sido correctamente introducido aparecerá un mensaje de error.

***Modificar alumno***

*Apellidos:* López Jiménez | *DNI:* 31025106V

*Actores principales:* usuario

*Actores secundarios:* alumno

**Precondiciones:**

- El alumno debe existir en la base de datos.

**Flujo principal:**

1. El caso de uso empieza cuando el sistema necesita modificar un alumno.
2. Se deberá introducir todos los datos nuevos una vez encontrado el alumno.

**Postcondiciones:**

- El sistema muestra al alumno y pide todos sus datos por pantalla.

**Flujos alternativos:**

- 2a: Si al introducir los apellidos del alumno se encuentran coincidencias, el el programa pedirá el DNI.
- 2b: Si al introducir DNI el alumno no se encuentra en la base de datos, se mostrará un mensaje de error.

***Borrar alumno***

*Apellidos:* López Jiménez | *DNI:* 31025106V

*Actores principales:* usuario

*Actores secundarios:* alumno

**Precondiciones:**

- El alumno debe existir en la base de datos.

**Flujo principal:**

1. El caso de uso empieza cuando el sistema necesita borrar un alumno.
2. Se deberá encontrar al alumno único y luego este será borrado

**Postcondiciones:**

- El sistema debe encontrar a un único alumno.
- El alumno debe existir en la base de datos.

**Flujos alternativos:**

- 2a: Si al introducir los apellidos del alumno se encuentran coincidencias, el programa pedirá el DNI.
- 2b: Si al introducir DNI el alumno no se encuentra en la base de datos, se mostrará un mensaje de error.

**Cambio líder**

*Breve descripción:* El sistema modifica el líder del grupo o asigna uno en el caso de no haberlo.

ID: 3

*Actores principales:* usuario

*Actores secundarios:* grupo, alumno

**Precondiciones:**

- Debe existir un alumno y grupo.

**Flujo principal:**

1. El caso de uso comienza cuando se desea cambiar o asignar el líder de uno de los grupos.
2. Se comprueba si este equipo tenía ya un líder.
3. En caso afirmativo se elige cuál de los otros componentes del grupo será el líder y en caso de no haberlo se asignará uno.

**Postcondiciones:**

- El sistema mostrará una confirmación del cambio de líder.
- Después del programa vuelve al menú.

**Flujos alternativos:**

Esta función no posee flujos alternativos.

**Borrado completo**

*Breve descripción:* El sistema borrará por completo la base de datos creada.

*Actores principales:* usuario

*Actores secundarios:* equipo, alumno

**Precondiciones:**

- Debe existir una base de datos con alumnos.

**Flujo principal:**

1. El caso de uso comienza cuando se desea borrar la base de datos completa.
2. Se confirma la existencia de esta y se procede a su borrado.

**Postcondiciones:**

- El sistema mostrará una confirmación del borrado completo de la base de datos.
- Después del programa vuelve al menú.

**Flujos alternativos:**

No existen flujos alternativos para esta función.

**Copia de seguridad**

*Breve descripción:* El sistema copia todos los datos de la base a un archivo binario que el mismo crea.

*Actores principales:* usuario

*Actores secundarios:* equipo, alumno

**Precondiciones:**

- Debe existir una base de datos de alumnos.

**Flujo principal:**

1. El caso de uso comienza cuando se desea hacer una copia de los datos de la base.
2. Se realiza esta copia en un archivo binario.

**Postcondiciones:**

- El sistema mostrará una confirmación de la copia en archivo binario.
- Después del programa vuelve al menú.

**Flujos alternativos:**

Esta función no posee flujos alternativos.

**Mostrar todos**

*Breve descripción:* El sistema mostrará todos los alumnos de la base de datos, ordenados según el usuario indique.

*Ordenación según:* DNI, alfabeto, curso más alto matriculado, ascendente y descendente

*Actores principales:* usuario

*Actores secundarios:* alumno

**Precondiciones:**

- Debe existir una base de datos con alumnos.

**Flujo principal:**

1. El caso de uso comienza cuando se desean mostrar los alumnos ordenandos.
2. Se deben mostrar estos de manera ordenada alfabéticamente

**Postcondiciones:**

- El sistema muestra los alumnos ordenados según se desean y vuelve al menú del programa

**Flujos alternativos:**

3. Esta función no posee ningún flujo alternativo.

**Login**

*Breve descripción:* El sistema solicitará una contraseña al usuario y este accederá con unos determinados permisos al sistema.

*Actores principales:* usuario

*Actores secundarios:* equipo

**Precondiciones**

- No existen precondiciones en esta función.

**Flujo principal:**

1. El usuario se introduce en el sistema y este pide una identificación.
2. El usuario introduce por teclado una contraseña y según esta el sistema le concederá los permisos requeridos para acceder a las distintas funciones de este.

**Postcondiciones:**

- Carga el menú del programa.

**Flujos alternativos**

- Si la identificación no corresponde a ningún tipo de permiso el sistema mostrará por pantalla un mensaje de error y volviendo de nuevo a la solicitud de identificación.

**Borrar Grupos**

*Breve descripción:* El usuario desea borrar un grupo y disolver sus integrantes.

*Actores principales:* usuario

*Actores secundarios:* grupo, alumno

**Precondiciones:**

- Debe existir un alumno y un grupo

**Flujo principal:**

1. El caso de uso comienza cuando se quiere borrar un grupo.
2. El programa recibe un número de grupo y lo elimina disolviendo sus integrantes.

**Postcondiciones:**

- Carga el menú de programa de nuevo.

**Flujos alternativos:**

- Si el grupo que se desea borrar no existe, se mostrará un mensaje de error por pantalla.

**Contar Miembros**

*Breve descripción:* El usuario introduce un número de grupo y el programa calcula el número de miembros que lo compone.

*Actores principales:* usuario

*Actores secundarios:* grupo, alumno

**Precondiciones:**

- Debe existir un alumno y un grupo.

**Flujo principal:**

1. El caso de uso comienza cuando se desea conocer el número de integrantes de un grupo.
2. El programa solicitará un número de grupo, comprobando que este existe y contando su número de miembros.
3. El programa devuelve el número de miembros del grupo.

**Postcondiciones:**

- Carga el menú de programa de nuevo.

**Flujos alternativos:**

- Si el grupo introducido no existe en la base de datos el programa lo mostrará por pantalla.

**Modificar Integrantes**

*Breve descripción:* El usuario podrá añadir o borrar integrantes del grupo seleccionado.

*Actores principales:* usuario

*Actores secundarios:* grupo, alumno

**Precondiciones:**

- Debe existir un alumno y un grupo.

**Flujo principal:**

1. El caso de uso comienza cuando se desean modificar los integrantes de un grupo.
2. Según el número de integrantes el usuario puede decidir entre añadir o borrar uno, si el grupo completa o no el número de integrantes (3).
3. Se introduce el DNI del alumno haciendo una llamada a BuscarAlumno, para saber si se encuentra en la base de datos.
4. En caso afirmativo se añadirá o borrará del grupo deseado.

**Postcondiciones:**

- Carga el menú de programa de nuevo.

**Flujos alternativos:**

- Si el alumno que se desea introducir o borrar no existe, se mostrará un mensaje de error por pantalla.

**Mostrar Grupos**

*Breve descripción:* El usuario desea mostrar los integrantes de un grupo y sus datos.

*Actores principales:* usuario

*Actores secundarios:* grupo, alumno

**Precondiciones:**

- Debe existir un alumno y un grupo

**Flujo principal:**

1. El caso de uso comienza cuando se quieren ver los datos de los alumnos de un grupo.
2. El programa recibe un número de grupo y muestra los alumnos que lo componen con sus respectivos datos.

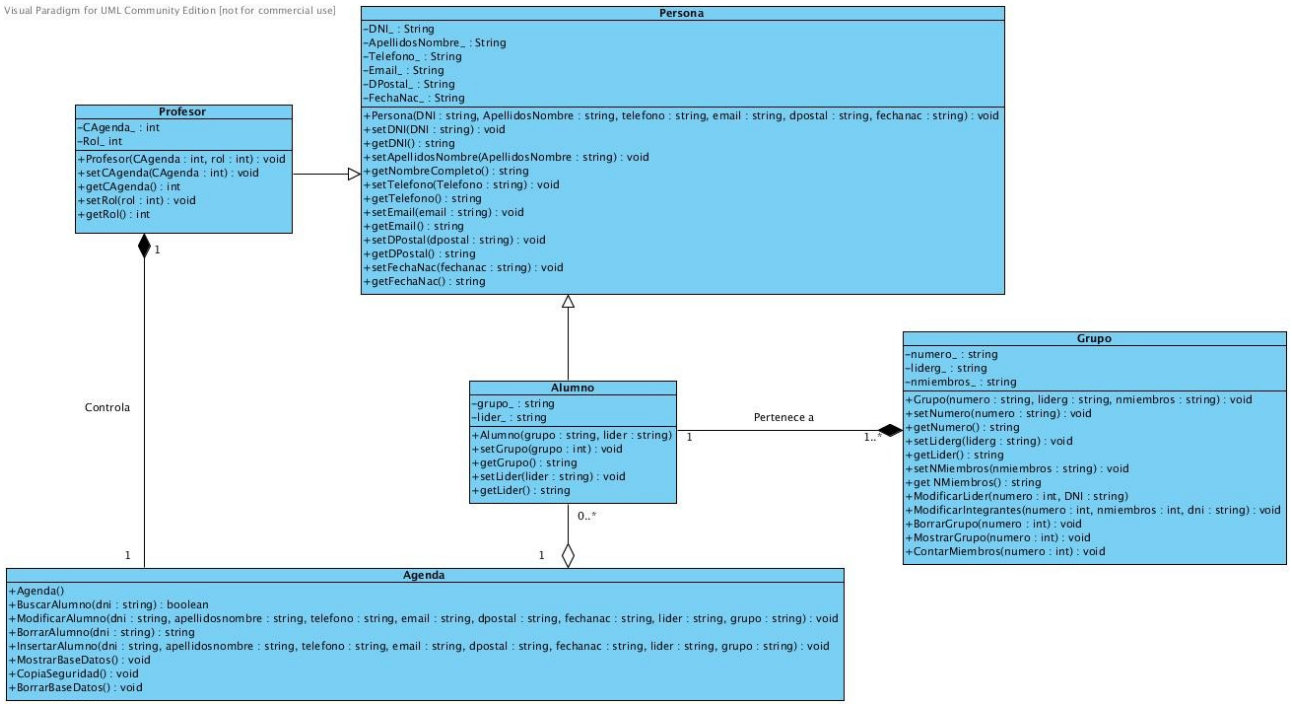
**Postcondiciones:**

- Carga el menú de programa de nuevo.

**Flujos alternativos:**

- Si el grupo que se desea mostrar no existe, se mostrará un mensaje de error por pantalla.

## 5. Diagrama de Clases:



**CLASES PRESENTES**

Cada clase contiene su propia función constructora de la clase y sus funciones 'Get' y 'Set' para cada atributo de la propia clase.

- *Persona:* Es la clase general que distingue entre alumnos y profesores, tiene atributos comunes de ambos pero no llega a tener ninguna operación, sus atributos son:
    - DNI: Distingue entre una persona u otra
    - NombreCompleto: Contiene el nombre completo de la persona.
    - Teléfono: Es el teléfono de contacto de la persona.
    - Email: Es el correo que se crea a cada persona al pertenecer a la base de datos de la universidad.
    - DPostal: Contiene el código postal de cada persona.
    - FechaNac: Contiene el dia, mes y año de nacimiento de la persona con el formato 'dd/mm/aaaa'.
- *Alumnos:* Es una de las clases que aparecen de la distinción de Persona y da lugar a la clase Grupo, tiene como atributos:
    - Grupo: Es el número de grupo al que cada alumno pertenece, apunta a la clase 'Grupo'
    - Líder: Es un dato de 'Si/No' que comprueba si dicho alumno es el líder de su grupo.
- *Grupo:* Contiene información referente a todos los grupos de trabajo de la clase, el grupo no puede existir sin un líder que puede ser modificado por el profesor en cualquier momento, sus atributos son:
    - Número: Es el atributo que distingue los grupos.
    - Líder: Contiene el DNI del alumno líder de su grupo.
    - Integrantes: Es un atributo múltiple que contiene el nombre de todos los integrantes del grupo, líder incluido.
- La clase Grupo también contiene las operaciones:
    - MostrarGrupo(): Muestra a todos los componentes del grupo.
    - ContarMiembros(): Cuenta los miembros que existen dentro de un grupo.
    - ModificarLider(): Permite el cambio del líder del grupo mediante la introducción del DNI del nuevo líder.

- ModificarIntegrantes(): Permite la expulsión o agregación de alumnos al grupo, no se puede expulsar al líder de forma que para echar al alumno líder hay que cambiar el líder previamente con 'Cambiar líder()'.
- EliminarGrupo(): Permite la eliminación total del grupo.

- *Profesor:* Es la otra clase que aparece como distinción de la clase 'Persona' y da lugar a la clase 'Agenda', esta clase hace referencia a la clase 'Alumnos' en algunas funciones, sus atributos son:
  - CAgenda: Contiene el dato que distingue a los profesores
  - Rol: Si contiene un 1, el rol asignado es el de 'Coordinador', si contiene un 0, se le asigna el rol 'Ayudante'. El rol 'Coordinador' tiene acceso a más funciones por ello es necesario hacer esta distinción.

- *Agenda:* Es la clase que recoge el listado de todos los alumnos de la base de datos, no tiene atributos.

  *La clase Agenda contiene las operaciones:*
  - BuscarAlumno(): Busca un alumno mediante el dni y devuelve sus datos, si el alumno no existe aparece un mensaje de error por pantalla.
  - ModificarAlumno(): Recibe el DNI del alumno a modificar y hace una llamada a BuscarAlumno(), muestra todos los datos del alumno almacenado y pide todos los nuevos datos.
  - BorrarAlumno(): Recibe el DNI del alumno a mostrar y hace una llamada a BuscarAlumno(), luego borra al alumno de la base de datos.
  - InsertarAlumno(): Recibe el DNI del nuevo alumno y hace una llamada a BuscarAlumno(), si el alumno no existe en la base de datos pide los datos del nuevo alumno.
  - CopiaSeguridad(): Crea una copia de seguridad de toda la base de datos existente en un fichero binario.
  - BorrarBaseDeDatos(): Imprime un mensaje de alerta y luego pide confirmación para borrar toda la base de datos.
  - MostrarBaseDatos(): Muestra todo el listado de alumnos de la base de datos.

**RELACIONES ENTRE CLASES**

- Un alumno solo puede pertenecer a un grupo mientras que un único grupo se compone de muchos alumnos
- Un alumno puede uno o varios profesores pero uno de ellos debe ser Coordinador mientras que todos los demás serán Ayudantes
- Un profesor controla una única agenda de clase y una agenda de clase la controla un único profesor
- La agenda se compone de cero o muchos alumnos

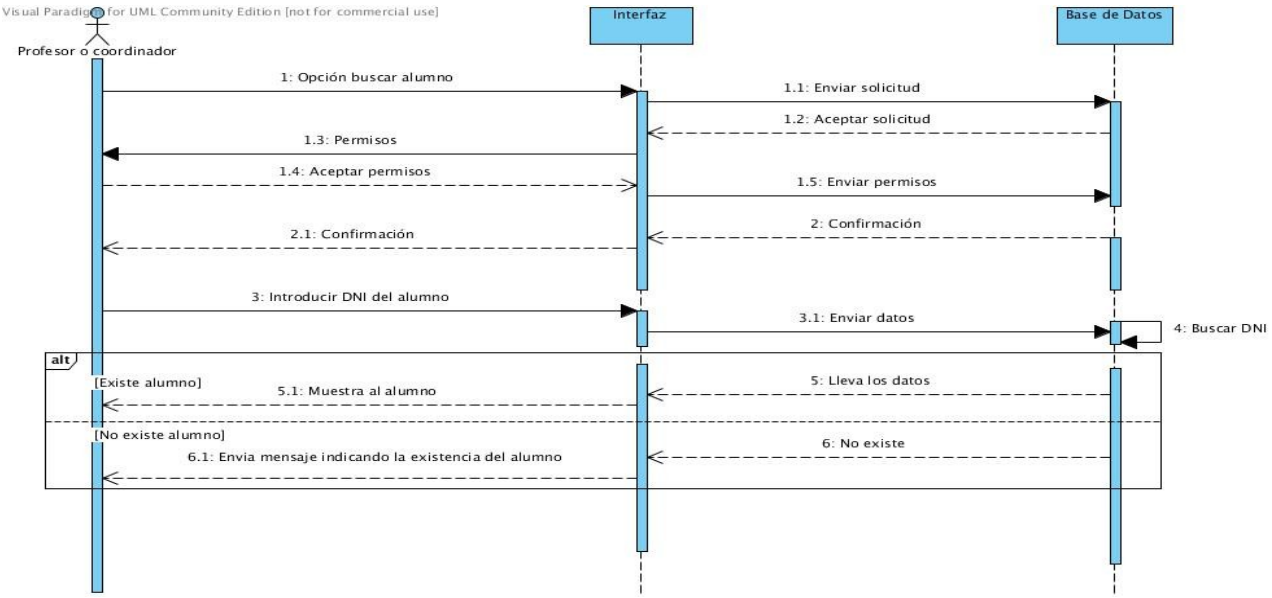
## 6. Diagramas de Secuencia:

### BUSCAR ALUMNO

- En esta función el usuario desea buscar a un alumno por DNI y que lo muestre por pantalla.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el DNI del alumno para encontrar todos sus datos, si ese alumno no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Finalmente, muestra al alumno deseado.

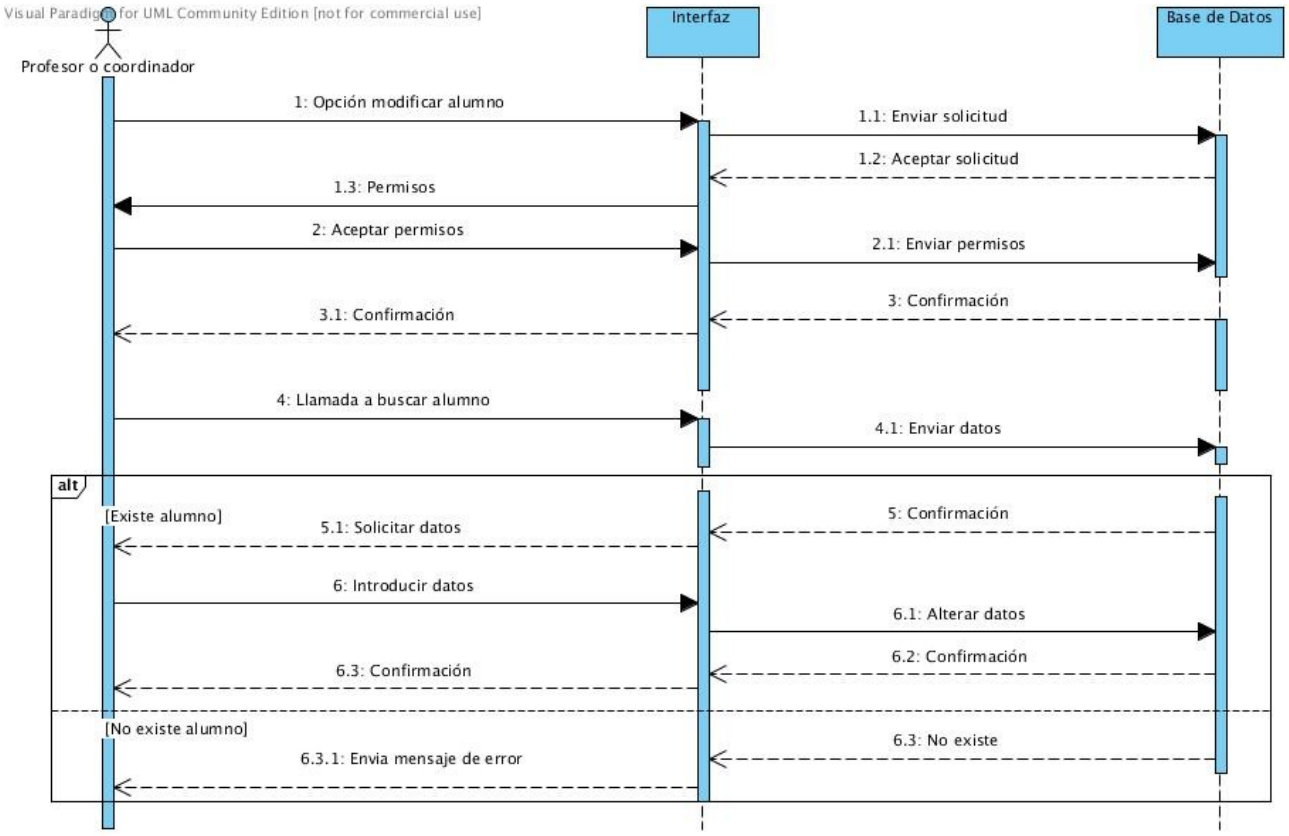



### MODIFICAR ALUMNO

- En esta función el usuario desea modificar los datos de un alumno.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el DNI del alumno para encontrar todos sus datos mediante una llamada a la función Buscar alumno, si ese alumno no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Introducir los datos que se desea modificar al pedirlo el programa.
4. Finalmente, que dichos datos se cambien en la base de datos.

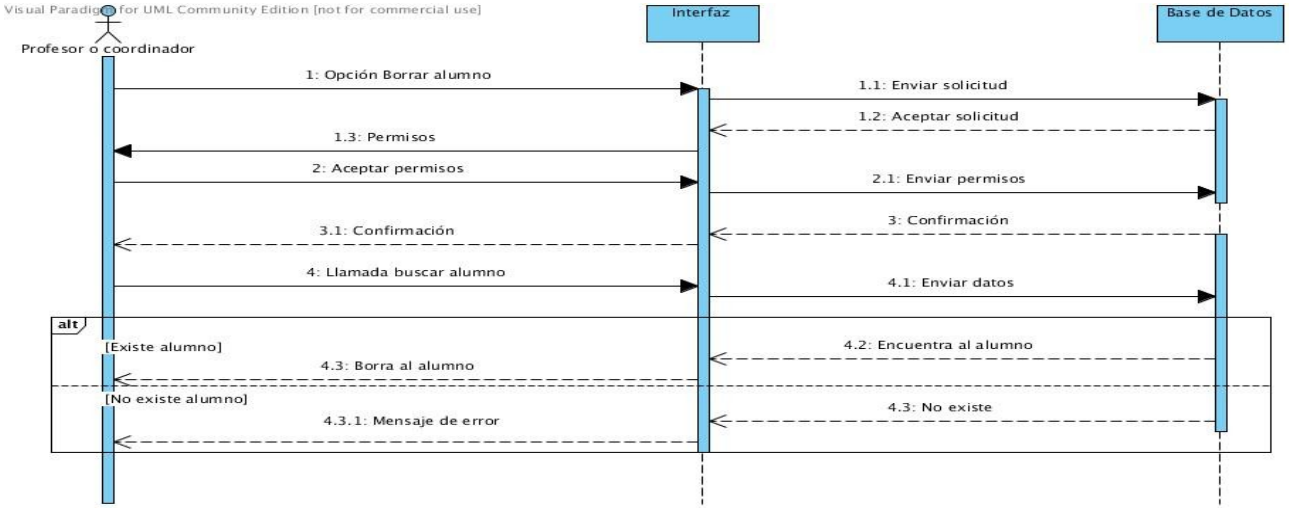



**BORRAR ALUMNO**

- En esta función el usuario desea borrar a un alumno identificado por DNI .

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el DNI del alumno para encontrar todos sus datos mediante una llamada a la función Buscar alumno, si ese alumno no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Finalmente, borra al alumno al cual le pertenezca el DNI introducido.

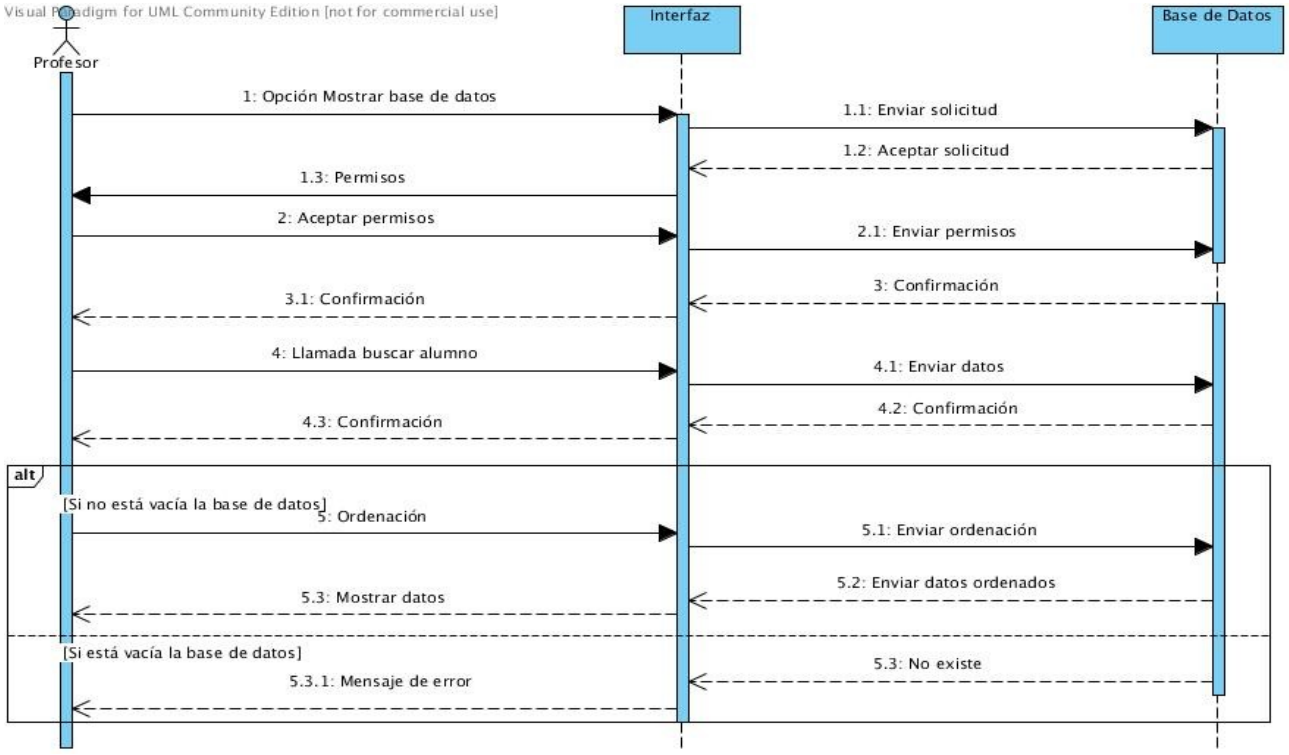



**MOSTRAR BASE DE DATOS**

- En esta función muestra todos los alumnos de la base de datos según la ordenación deseada.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que se introduzca la ordenación deseada el mostrar los alumnos (DNI o alfabéticamente, todo esto de forma tanto ascendente como descendente).
3. Finalmente se mostrarán los datos ordenados de la forma deseada y finalizará el programa.

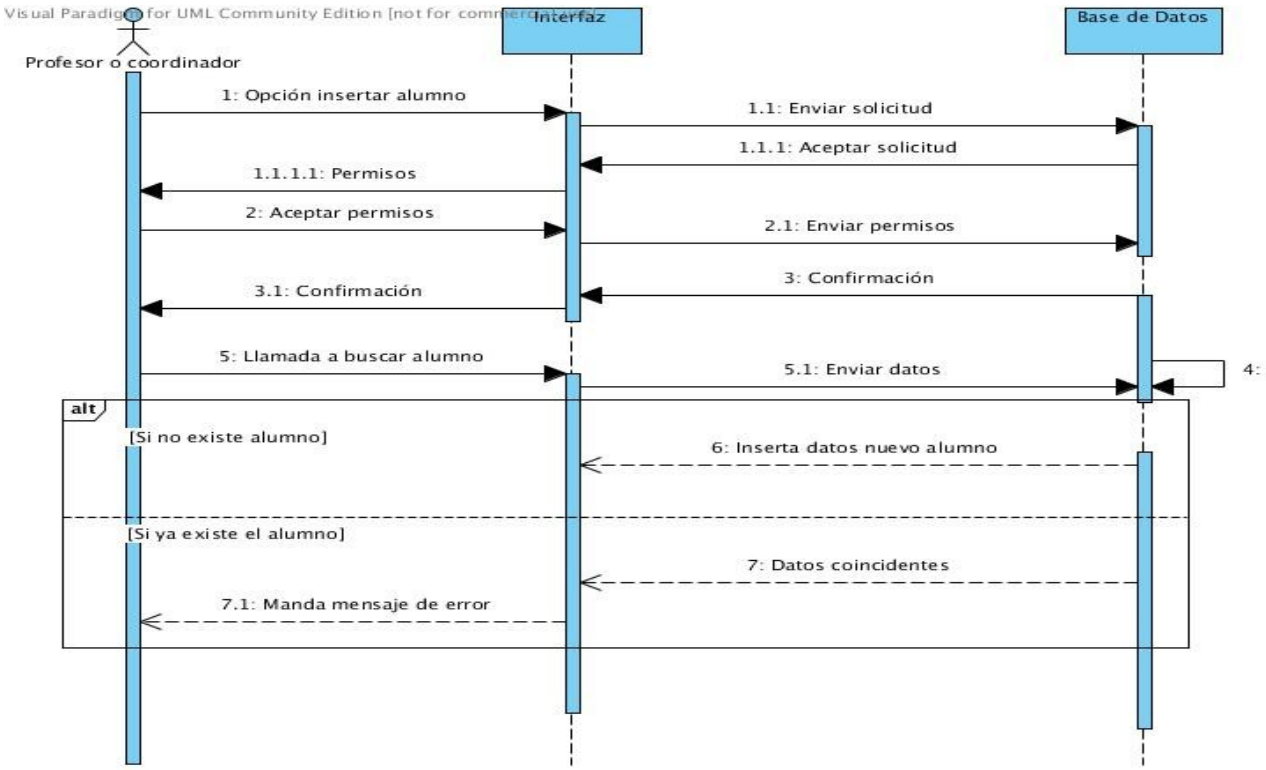



**INSERTAR ALUMNO**

- En esta función se desea insertar los datos de un nuevo alumno en la base de datos.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. El sistema pedirá los datos necesarios para hacer una llamada a BuscarAlumnos.
3. Si estos datos ya están en la base de datos el sistema mostrará un mensaje indicándolo.
4. Si los datos no están registrados, el sistema pedirá el resto de datos necesarios para introducir el nuevo alumno.
5. Se creará el alumno nuevo y finalizará el programa.



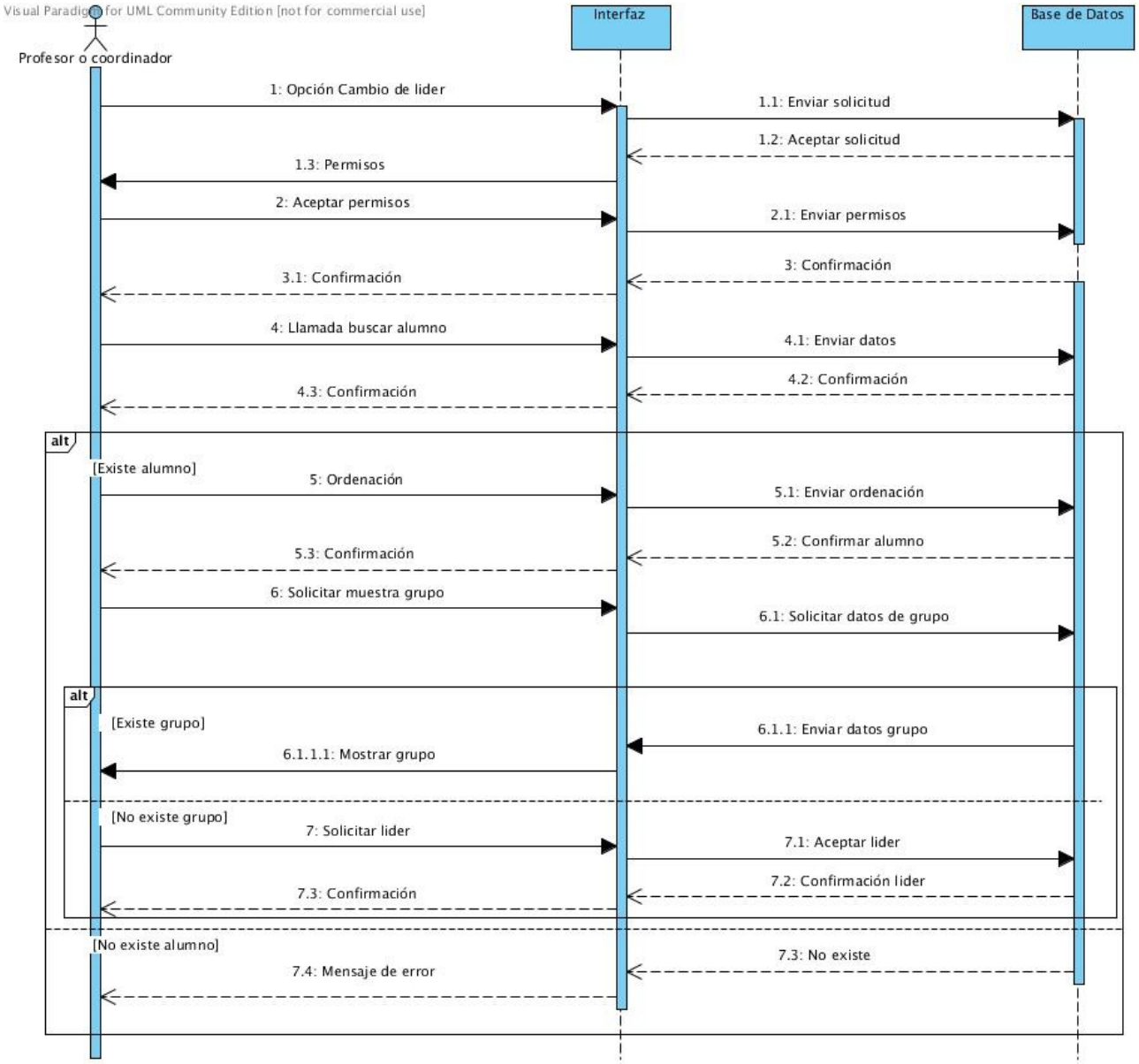

**CAMBIO DE LIDER**

- En esta función el usuario desea cambiar el líder de un grupo y en caso de que no tenga introducirlo.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el DNI del alumno para encontrar todos sus datos mediante una llamada a la función Buscar alumno, si ese alumno no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Confirma que el grupo existe y si existe mostrarlo, en caso de que no exista en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.

4. Finalmente confirma que dicho alumno es líder del equipo.

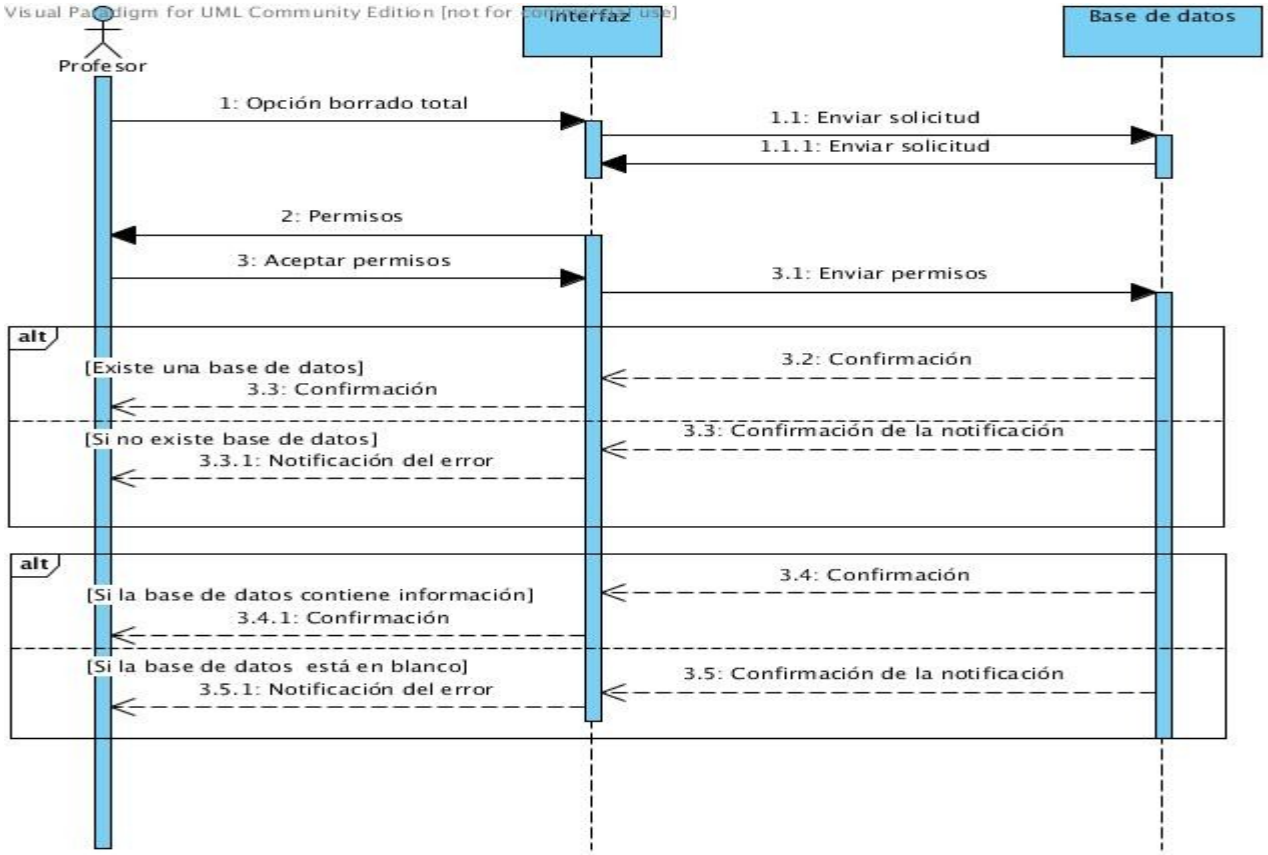



**BORRADO TOTAL**

- En esta función el usuario desea borrar toda la base de datos.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Y que se confirme el borrado de la base de datos.

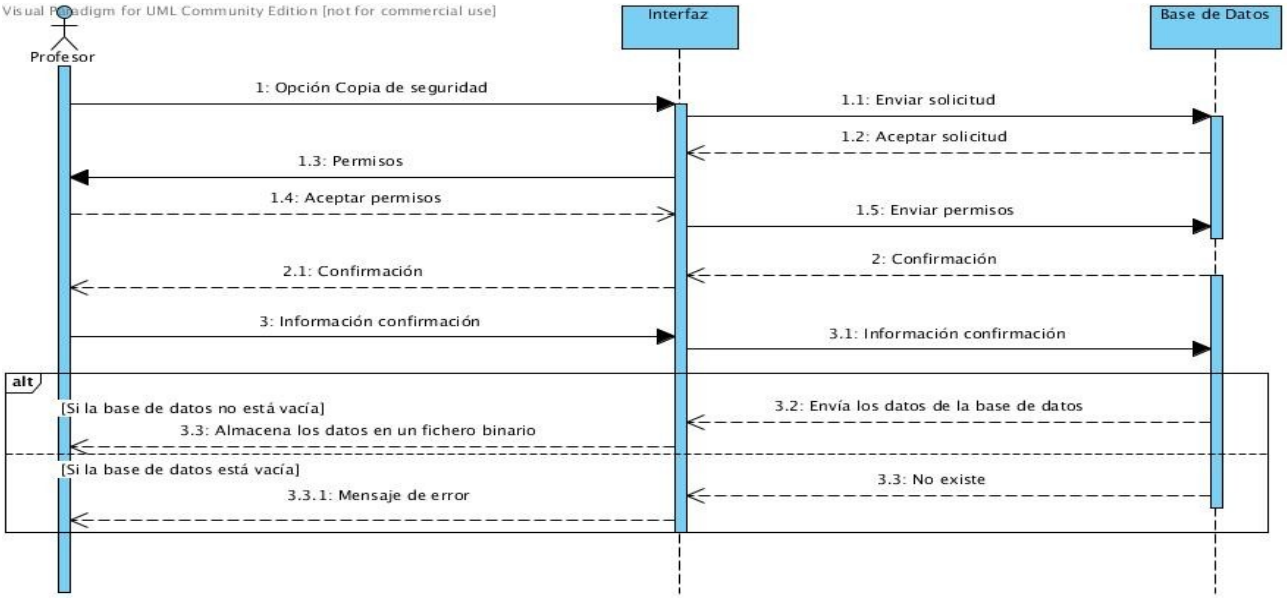



**COPIA SEGURIDAD**

- En esta función se desea realizar una copia de la base de datos completa, almacenandola en un archivo binario.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Se le enviarán al sistema una copia de la base de datos completa y este lo almacenará en un archivo binario, finalmente se cerrará el programa.

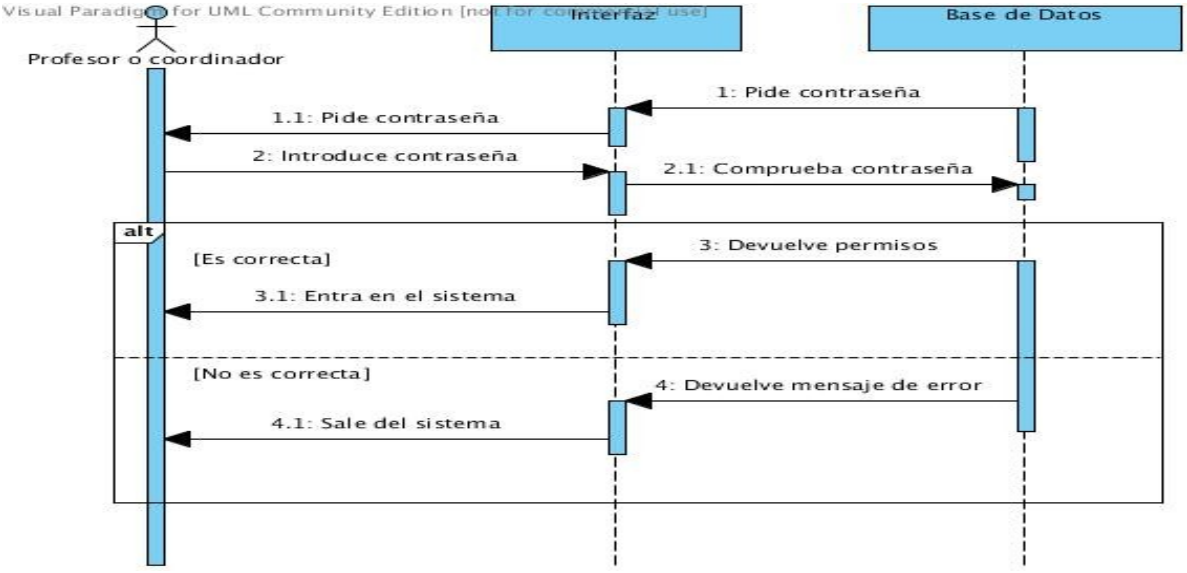



**LOGIN**

- En esta función el usuario se identificará como profesor o coordinador.

- Para ello es necesario:

1. El usuario introducirá la contraseña para entrar en el programa.
2. Si el usuario introduce en el último digito un 0 se identificará como profesor, si introduce un 1 entrará en el programa como coordinador.

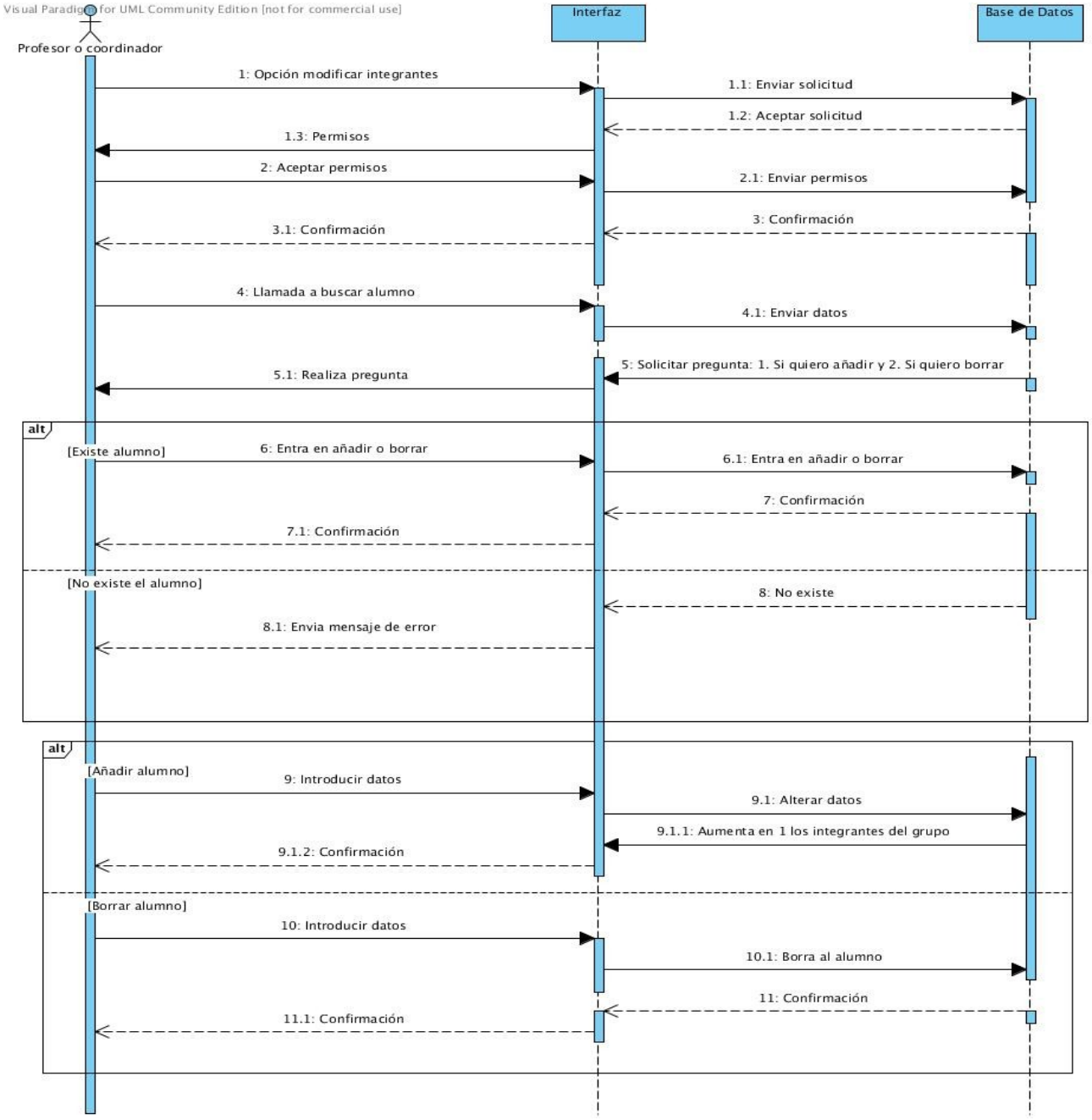


**MODIFICAR INTEGRANTE**

- En esta función el usuario desea modificar el intregrante de un grupo.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el DNI del alumno para encontrar todos sus datos, si ese alumno no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Si se encuentra en la base de datos, te dará la opción de elegir entre añadir y borrar al integrante de un grupo.
4. Al elegir añadir, se introducirá dentro del grupo deseado; Si elige borrar, se borra al alumno de dicho grupo.

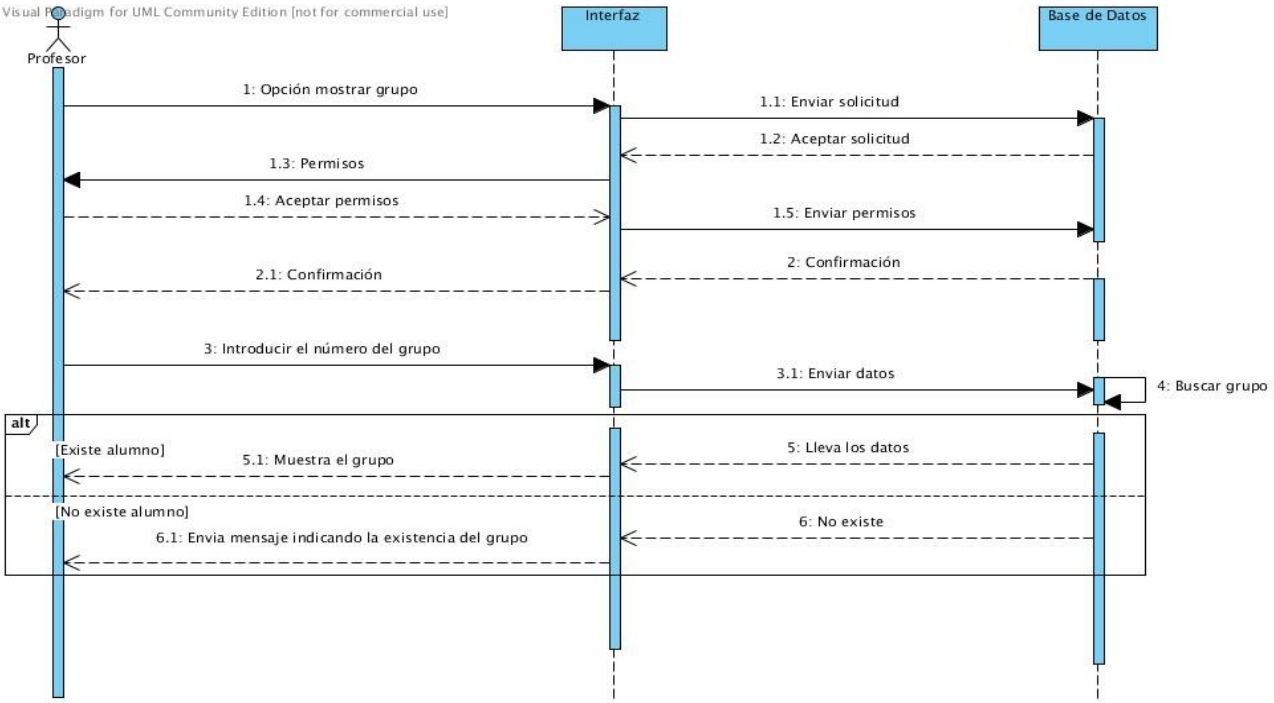


## MOSTRAR GRUPO

- En esta función el usuario desea mostrar por pantalla un grupo.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el número de grupo deseado, si ese grupo no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Finalmente, muestra el grupo deseado.

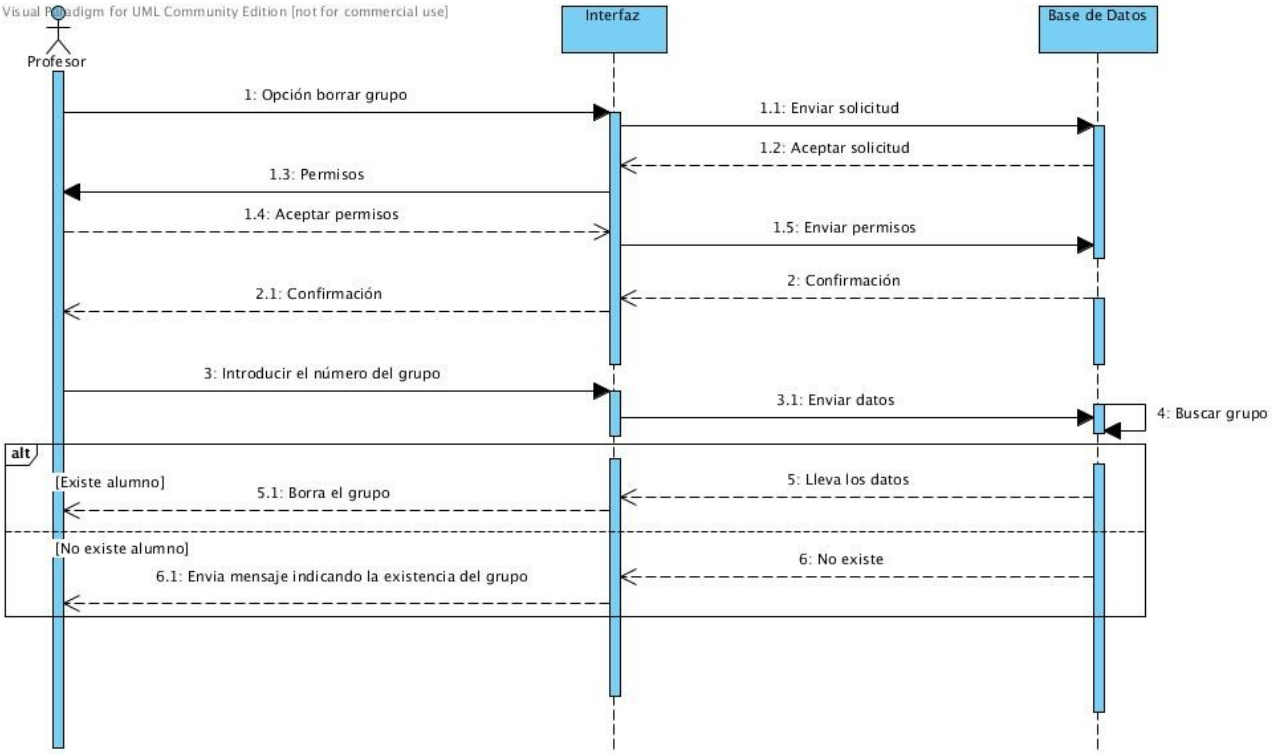



## BORRAR GRUPO

- En esta función el usuario desea borrar un grupo.

- Para ello es necesario:

1. Dicho usuario se identifique para conseguir los permisos necesarios para acceder a esta función.
2. Que introduzca el número de grupo deseado, si ese grupo no se encuentra en la base de datos se mandará un mensaje de error y volveremos a iniciar el programa.
3. Finalmente, borra el grupo deseado.




## 7.-Metodología SCRUM:

La metodología SCRUM es una metodología ágil que se basa en entregas parciales y frecuentes de un producto para obtener resultados con rapidez

*** Product Backlog:**

Contiene todas las funcionalidades ordenadas por prioridades. Es la agrupación de todas las historias de usuario del producto

En nuestro caso, el product backlog contiene 13 funciones ordenadas por prioridad y el tiempo estimado por función.

El orden de nuestras funciones por prioridad es el siguiente:

1.Login

2.BuscarAlumno

3.InsertarAlumno

4.ModificarAlumno

5.Borraralumno:

6.ModificarIntegrantes

7.ModificarLider

8.ContarMiembros:

9.MostrarGrupo:

10.BorrarGrupo:

11.MostrarBaseDatos:

12.CopiaSeguridad:

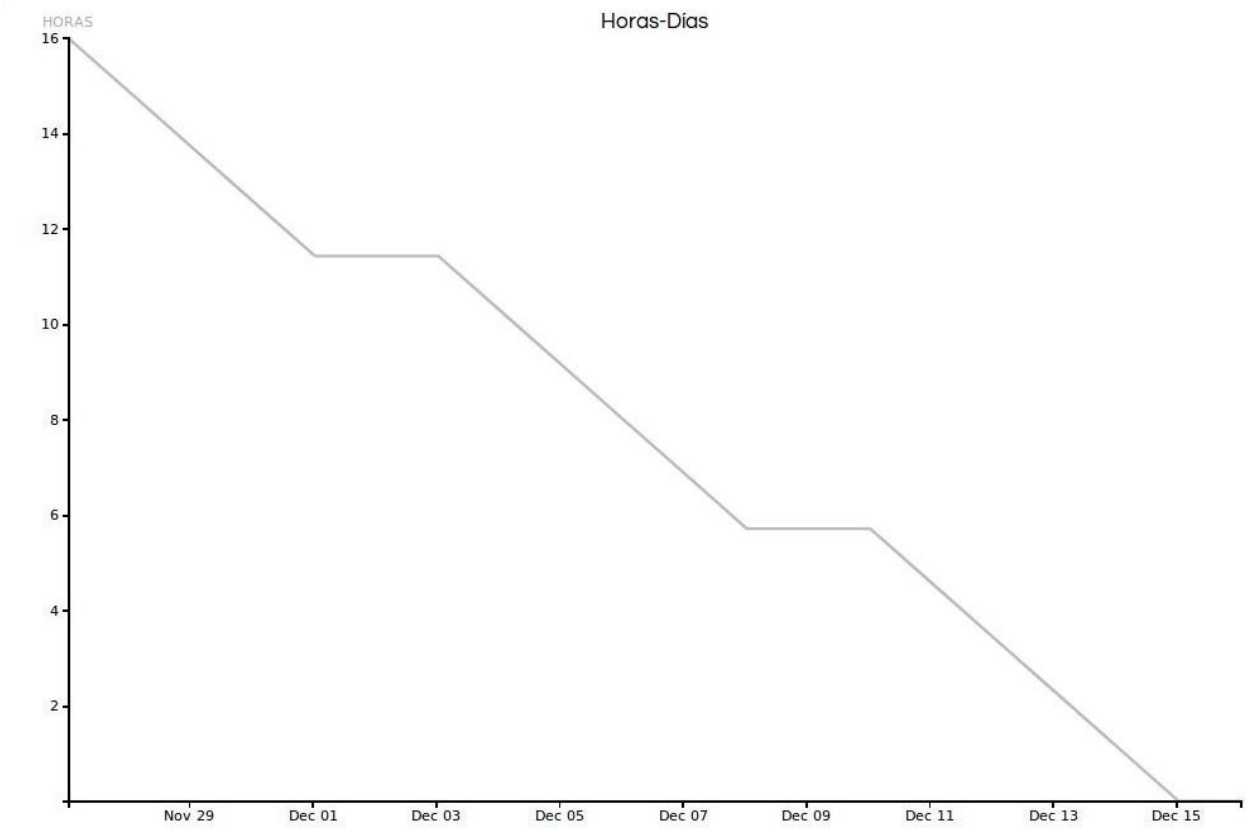
13.BorrarBaseDatos:

***Sprint backlogs:**

Funcionalidad a desarrollar en un sprint determinado.

*** Burndown charts**

Gráfico que muestra la cantidad de trabajo hecho.
En nuestro caso, se representan las 16 horas las cuales hemos trabajado en este proyecto.




Los integrantes del grupo empiezan a desarrollar el proyecto el 27 de noviembre y terminamos el día 15 de diciembre.

8.-Matriz de validación

FICHA TÉCNICA DEL CLIENTE

- Nombre:
- Apellidos:
- DNI:
- Profesión:
- Teléfono:

| **Aspectos** | Excelente 5 | Muy bien 4 | Bien 3 | Mal 2 | Muy mal 1 |
|---|---|---|---|---|---|
| Clases | | | | | |
| Funciones | | | | | |
| Casos de uso | | | | | |
| Historias de usuario | | | | | |
| Diagrama de clases | | | | | |
| Diagramas de secuencias | | | | | |
| Product Backlog | | | | | |
| Sprint backlog | | | | | |
| Burndown charts | | | | | |
| Tiempo estimado | | | | | |
| Calidad final del producto | | | | | |