Proyecto Sistema de Control de Alumnos

Realizado por:

Miguel Ángel Muñoz Martín

Ángel Villegas del Río

ÍNDICE

1. Definición del Problema

Un cliente nos ha pedido un programa para almacenar alumnos. Dicho programa permite al usuario poder almacenar alumnos, modificarlos, borrarlos, entre otras funcionalidades.

Además, dicho cliente nos ha pedido que su programa funcione en el sistema operativo Linux, y que el fichero en el que se almacenen los alumnos sea de tipo binario.

El cliente nos ha indicado que el programa no sólo lo va a manejar él, sino que puede ser que lo utilicen varios profesores, los cuales serán sus ayudantes, por lo que el programa debe ser capaz de diferenciar entre un profesor u otro, y limitar la funcionalidad dependiendo del profesor que sea.

2. Extracción de Requisitos

El cliente nos ha pedido un programa que almacene alumnos, y este alumno tenga varios campos para almacenar datos. Los campos son los siguientes:

- DNI del alumno

- Nombre del alumno

- Apellidos del alumno

- Líder de un grupo o no

- Nº de grupo

- Email

- Edad

Además, nos ha pedido una serie de requisitos, tanto funcionales como no funcionales.

2.1 Requisitos Funcionales

El cliente nos ha pedido que su programa debe contener los siguientes requisitos funcionales:

- Buscar alumno por DNI

- Modificar alumno

- Modificar líder de grupo

- Mostrar alumno

- Borrar alumno

- Mostrar todos los alumnos

- Insertar alumno

- Existe alumno

- Borrar todos los alumnos

- Hacer copia de seguridad

- Cargar copia de seguridad

- Dar de alta a un profesor

- Dar de baja a un profesor

- Entrar al sistema mediante Usuario y Credencial

- Salir del sistema

2.2. Requisitos no funcionales

El cliente nos ha pedido que el programa contenga los siguientes requisitos no funcionales:

- Funcionar en sistema Linux

- El sistema pueda cargar un archivo binario

- Para cargar alumno se necesita mínimo DNI y apellido

- El nº máximo de alumnos es de 150

- Los datos se almacenan en un fichero binario

- Se considera la posibilidad de que haya varios profesores, cada uno con una credencial

- Los profesores estarán divididos en Coordinador, el cual sólo habrá uno, y Ayudantes

3. Historias de Usuario

Conforme a los requisitos funcionales que nos ha solicitado el cliente, se han obtenido las siguientes Historias de Usuario:

```
## (ANVERSO)

**ID**: 001 **Buscar alumno**

Como usuario quiero poder buscar un alumno de la clase.

**Prioridad:** 4

## (REVERSO)

* Quiero poder ver si se ha encontrado el alumno.
* Debe buscarse por DNI o por apellido.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
## (ANVERSO)

**ID**: 004 **Mostrar alumno**

Como usuario quiero poder visualizar el nombre completo de un alumno de la clase.

**Prioridad:** 3

## (REVERSO)

* Quiero poder visualizar todos los datos del alumno.
* Se debe mostrar si el alumno es líder o no.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
## (ANVERSO)

**ID**: 005 **Borrar alumno**

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

**Prioridad:** 2

## (REVERSO)

* Quiero poder borrar todos los datos del alumno.
* En caso de ser líder de un grupo, dicho grupo quedará temporalmente sin líder, hasta que el usuario asigne un nuevo líder.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
## (ANVERSO)

**ID**: 006 **Mostrar todos los alumnos**

Como usuario quiero poder visualizar el nombre completo de todos los alumnos.

**Prioridad:** 2

## (REVERSO)

* Quiero poder visualizar todos los datos de todos los alumnos.
* Se debe mostrar qué alumnos son líderes.
* Se debe preguntar al usuario de que forma quiere ordenar a los alumnos.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
# (ANVERSO)
**ID:** 007 **Insertar alumno**



Como usuario quiero insertar alumnos


**Prioridad:** 3

# (REVERSO)

* Quiero poder insertar alumnos.
* Y como mínimo necesite el DNI, apellidos y nombre.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
# (ANVERSO)
**ID:** 008 **Comprobar la existencia de alumnos**



Como usuario quiero comprobar la existencia de alumnos


**Prioridad:** 4

# (REVERSO)

* Quiero que insertando el DNI del alumno obtenga si existe el alumno o no.
* Como mínimo se necesita para comprobarlo el DNI.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
# (ANVERSO)
**ID:** 009 **Borrado total**



Como usuario quiero borrar los datos de los alumnos.


**Prioridad:** 1

# (REVERSO)

* Quiero que se pueda borrar todos los datos.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
# (ANVERSO)
**ID:** 010 **Hacer copia de seguridad**



Como usuario quiero poder cargar en caso de pérdida de datos una copia de seguridad.


**Prioridad:** 4

# (REVERSO)

* Quiero que insertando el sistema tenga una copia de seguridad para poder cargarla el usuario en cualquier momento.
* La copia de seguridad tiene que estar actualizada con todos los datos.
* Puede acceder sólo el coordinador.
```

```
# (ANVERSO)

**ID:** 011 **Cargar copia de seguridad**

Como usuario quiero poder cargar en caso de pérdida de datos una copia de seguridad.

**Prioridad:** 1

# (REVERSO)

* Quiero que insertando el sistema tenga una copia de seguridad para poder cargarla el usuario en cualquier momento.
* La copia de seguridad tiene que estar actualizada con todos los datos.
* Puede acceder sólo el coordinador.
```

```
# (ANVERSO)

**ID:** 012 **Dar de alta a un profesor**

Como usuario quiero poder dar de alta a un profesor.

**Prioridad:** 3

# (REVERSO)

* Quiero poder insertar un profesor.
* Y como mínimo necesite el DNI, apellidos y nombre.
* Puede acceder sólo el coordinador.
```

```
# (ANVERSO)

**ID:** 013 **Dar de baja a un profesor**

Como usuario quiero poder dar de baja a un profesor.

**Prioridad:** 3

# (REVERSO)

* Quiero poder borrar todos los datos de un profesor.
* Puede acceder sólo el coordinador.
```

```
# (ANVERSO)

**ID:** 014 **Acceder al sistema**

Como usuario quiero poder acceder al sistema mediante la credencial de cada profesor.

**Prioridad:** 1

# (REVERSO)

* Se debe introducir la credencial personal para acceder.
* Pueden acceder tanto el coordinador como los ayudantes.
```

```
# (ANVERSO)

**ID:** 015 **Salir del sistema**

Como usuario quiero poder salir del sistema.

**Prioridad:** 1

# (REVERSO)

* Pueden acceder tanto el coordinador como los ayudantes.
```

4. Casos de Uso

Tras crear las diferentes historias de usuario en base a los requisitos funcionales que nos solicitó el cliente, se crean los siguientes casos de uso:

```
**Buscar alumno**

**ID:** 001

**Breve descripción:** El sistema busca a un alumno.


**Actores principales:** Usuario.

**Actores secundarios:** Alumno.

**Precondiciones:**

  1. El alumno debe existir en el sistema.

**Flujo principal:**

  1. El caso de uso empieza cuando el sistema necesita buscar un alumno.
  2. El sistema comprueba si dicho alumno existe.

**Postcondiciones:**

  * El sistema muestra por pantalla si el alumno se encuentra registrado en el sistema.

**Flujos alternativos:**

  2.a. Si no existe el alumno, el sistema muestra un mensaje de error y finaliza la función.
```

```
**Modificar alumno**

**ID:** 002

**Breve descripción:** El sistema modifica los datos de un alumno.


**Actores principales:** Usuario.

**Actores secundarios:** Alumno.

**Precondiciones:**

  1. El alumno debe existir en el sistema.

**Flujo principal:**

  1. El caso de uso empieza cuando el sistema necesita modificar los datos de un alumno.
  2. El sistema ejecuta la función buscar alumno, para localizar al alumno que se desea modificar los datos.
  3. El sistema recoge los datos del alumno.

**Postcondiciones:**

  * El sistema permite al usuario modificar los datos del alumno.

**Flujos alternativos:**

  2.a. Si no existe el alumno, el sistema muestra un mensaje de error y finaliza la función.
```

```
**Modificar líder de grupo**

**ID:** 003

**Breve descripción:** El sistema modifica al líder de un grupo.


**Actores principales:** Usuario.

**Actores secundarios:** Alumno.

**Precondiciones:**

  1. El grupo debe existir en el sistema.

**Flujo principal:**

  1. El caso de uso empieza cuando el sistema necesita modificar al líder de un grupo.
  2. El sistema recoge los miembros del grupo.

**Postcondiciones:**

  * El sistema muestra por pantalla los miembros del grupo.
  * El sistema permite al usuario modificar al líder del grupo.

**Flujos alternativos:**

  2.a. Si no existe el grupo, el sistema muestra un mensaje de error y finaliza la función.
```

```
**Mostrar alumno**

**ID:** 004

**Breve descripción:** El sistema muestra los datos de un alumno.


**Actores principales:** Usuario.

**Actores secundarios:** Alumno.

**Precondiciones:**

  1. El alumno debe existir en el sistema.

**Flujo principal:**

  1. El caso de uso empieza cuando el sistema necesita mostrar un alumno.
  2. El sistema ejecuta la función buscar alumno, para localizar al alumno que se desea mostrar los datos.
  2. El sistema recoge los datos de un alumno.

**Postcondiciones:**

  * El sistema muestra al alumno por pantalla.

**Flujos alternativos:**

  2.a. Si no existe el alumno, el sistema muestra un mensaje de error y finaliza la función.
```

```
**Borrar alumno**

**ID:** 005

**Breve descripción:** El sistema borra los datos de un alumno.


**Actores principales:** Usuario.

**Actores secundarios:** Alumno.

**Precondiciones:**

  1. El alumno debe existir en el sistema.

**Flujo principal:**

  1. El caso de uso empieza cuando el sistema necesita borrar a un alumno.
  2. El sistema ejecuta la función buscar alumno, para localizar al alumno que se desea borrar los datos.
  3. El sistema recoge los datos de un alumno.

**Postcondiciones:**

  * El sistema borra al alumno del sistema.

**Flujos alternativos:**

  2.a. Si no existe el alumno, el sistema muestra un mensaje de error y finaliza la función.
```

```
**Mostrar todos los alumnos**

**ID:** 006

**Breve descripción:** El sistema muestra a todos los alumnos.


**Actores principales:** Usuario.

**Actores secundarios:** Alumno.

**Precondiciones:**

  1. Debe haber al menos 1 alumno almacenado en el sistema.

**Flujo principal:**

  1. El caso de uso empieza cuando el sistema necesita mostrar a todos los alumnos.
  2. El sistema recoge los datos de todos los alumnos.

**Postcondiciones:**

  * El sistema muestra a todos los alumnos por pantalla.
  * El sistema pregunta por pantalla si se desea un documento txt con los datos de los alumnos.


**Flujos alternativos:**

  2.a. Si no hay ningún alumno, el sistema muestra un mensaje de error y finaliza la función.
  2.b. En el caso de que se desee un documento txt, se crea dicho documento.
```

Insertar alumnos

ID: 007

Breve descripción: El usuario puede insertar alumnos.

Actores principales: Usuario

Actores secundarios: Alumnos

Precondiciones:

1. Para insertar al alumno necesitas mínimo los datos de nombre, apellido y DNI.

Flujo principal

1. El caso de uso empieza cuando el usuario intenta insertar un alumno.
2. El sistema recoge los datos del alumno.

Postcondiciones

1. El sistema guarda en la base de datos, los datos sobre el alumno.

Flujos alternativos

2.a. Si no se inserta alguno de los datos, el sistema dará error y finaliza la función.
2.b. Si se inserta el alumno como líder de algún grupo en el que ya hay un líder, el sistema mostrará un mensaje indicando que ya existe un líder, y se añadirá a dicho grupo sin ser líder.

Comprobar alumno

ID: 008

Breve descripción: El usuario puede comprobar si existe el alumno o no

Actores principales: Usuario

Actores secundarios: Alumnos

Precondiciones:

1. Para comprobar si el alumno existe o no, debemos insertar DNI.

Flujo principal

1. El caso de uso empieza cuando el usuario intenta comprobar la existencia de alumno.
2. El usuario introduce el DNI, y el sistema busca al alumno con el DNI introducido.

Postcondiciones

·El sistema muestra por pantalla los datos del alumno, en caso de que el alumno exista.

Flujos alternativos

· Si no se inserta el DNI del alumno , el sistema dará error al comprobar alumno.

```
**Borrado total**

**ID:** 009

**Breve descripción:** El usuario pueda borrar totalmente los datos

**Actores principales:** Usuario

**Actores secundarios:** Alumnos

**Precondiciones:**

1. Para poder borrar, debe de haber datos de alumnos.

**Flujo principal**

1. El caso de uso empieza cuando el usuario intenta borrar por completo los alumnos.
2. El usuario borra totalmente los datos de los alumnos.

**Postcondiciones**

·El sistema elimina a los alumnos.

**Flujos alternativos**

· Si no hay alumnos en la base de datos, el sistema dará error ya que no hay que borrar.
```

```
**Hacer copia de seguridad**

**ID:** 010

**Breve descripción:** El sistema realiza una copia de seguridad de los datos.

**Actores principales:** Usuario

**Actores secundarios:** Alumnos

**Precondiciones:**

  1. Debe haber mínimo 1 alumno introducido en el sistema.

**Flujo principal**

  1. El caso de uso empieza cuando el sistema necesita hacer la copia de seguridad.

**Postcondiciones**

  * El sistema guarda en un fichero binario todos los datos de los alumnos almacenados.

**Flujos alternativos**

  2.a. Si no hay ningún alumno, el sistema mostrará un mensaje de error y finaliza el programa.
```

```
**Cargar copia de seguridad**

**ID:** 011

**Breve descripción:** El usuario pueda cargar una copia de seguridad de los datos.

**Actores principales:** Usuario

**Actores secundarios:** Alumnos

**Precondiciones:**

1. Para poder cargar la copia de seguridad,debe de haber una copia de seguridad actualizada en el sistema.

**Flujo principal**

1. El caso de uso empieza cuando el usuario intenta cargar la copia de seguridad.
2. La copia de seguridad se actualizará cada vez que el usuario inserte cualquier dato nuevo.

**Postcondiciones**

·El sistema carga los datos de los alumnos desde la copia de seguridad.

**Flujos alternativos**

· Si no hay copia de seguridad, el sistema dará error y finaliza el programa.
```

```
**Dar de alta a un profesor**

**ID:** 012

**Breve descripción:** El usuario puede dar de alta a un profesor.

**Actores principales:** Usuario

**Actores secundarios:** Profesor

**Precondiciones:**

1. El usuario debe ser coordinador.

**Flujo principal**

1. El caso de uso empieza cuando el sistema necesita introducir a un nuevo profesor
2. El sistema recoge los datos del profesor.
3. El sistema pregunta si el profesor es coordinador o ayudante.

**Postcondiciones**

1. El sistema guarda en la base de datos, los datos sobre el profesor.
2. El sistema da permisos a ese profesor dependiendo si es coordinador o ayudante.

**Flujos alternativos**

2.a. Si no se inserta alguno de los datos, el sistema dará error y finaliza la función.
2.b. Si el profesor introducido es ayudante, no podrá tener acceso a algunas funciones del programa.
```

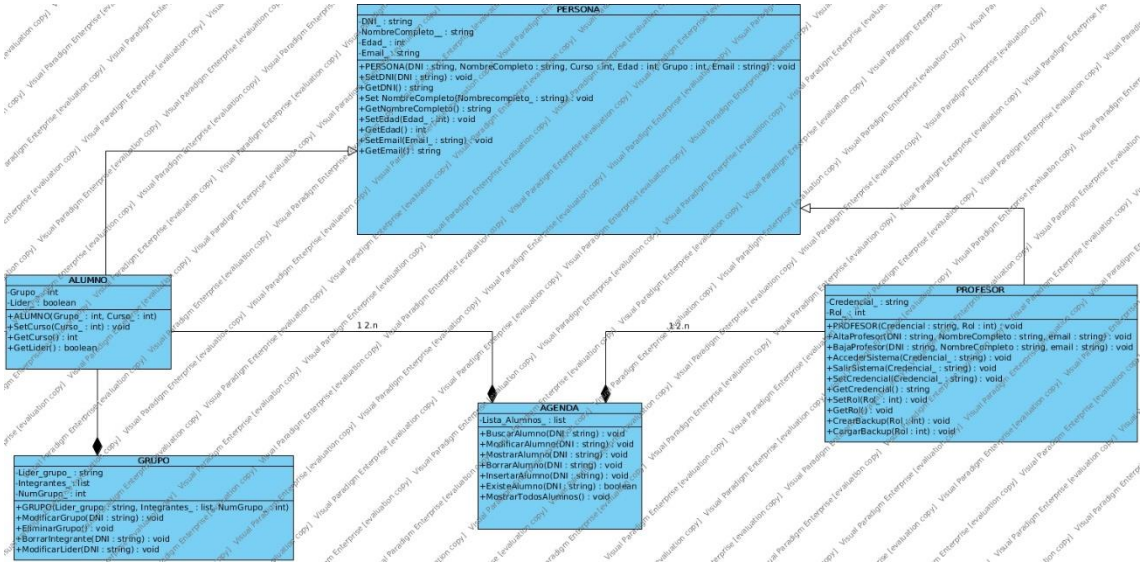
```
**Dar de baja a un profesor**

**ID:** 013

**Breve descripción:** El usuario puede dar de baja a un profesor.

**Actores principales:** Usuario

**Actores secundarios:** Profesor

**Precondiciones:**

1. El usuario debe ser coordinador.

**Flujo principal**

1. El caso de uso empieza cuando el sistema necesita borrar a un profesor.
2. El sistema recoge los datos del profesor.

**Postcondiciones**

1. El sistema borra los datos del profesor.


**Flujos alternativos**

2.a. Si el profesor no existe, el sistema dará error y finaliza la función.
```

```
**Acceder al sistema**

**ID:** 014

**Breve descripción:** El usuario introduce su credencial para acceder al sistema.

**Actores principales:** Usuario

**Precondiciones:**

1. El usuario debe estar dado de alta en el sistema.

**Flujo principal**

1. El caso de uso empieza cuando el usuario quiere acceder al sistema.
2. El sistema pide al usuario una credencial personal y única para cada profesor.
2. El sistema recoge dicha credencial.

**Postcondiciones**

1. El sistema permite acceder al profesor al sistema.


**Flujos alternativos**

2.a. Si el profesor no está dado de alta, el sistema dará error y finaliza la función.
```

```
**Salir del sistema**

**ID:** 015

**Breve descripción:** El usuario sale del sistema.

**Actores principales:** Usuario

**Precondiciones:**

1. El usuario debe estar dado de alta en el sistema.

**Flujo principal**

1. El caso de uso empieza cuando el usuario quiere salir del sistema.
2. El sistema finaliza todas las funciones.

**Postcondiciones**

1. El sistema expulsa al usuario del sistema.


**Flujos alternativos**

2.a. Si hay alguna función abierta, se preguntará al usuario si desea realmente salir del sistema.
```

5. Diagrama de Clases

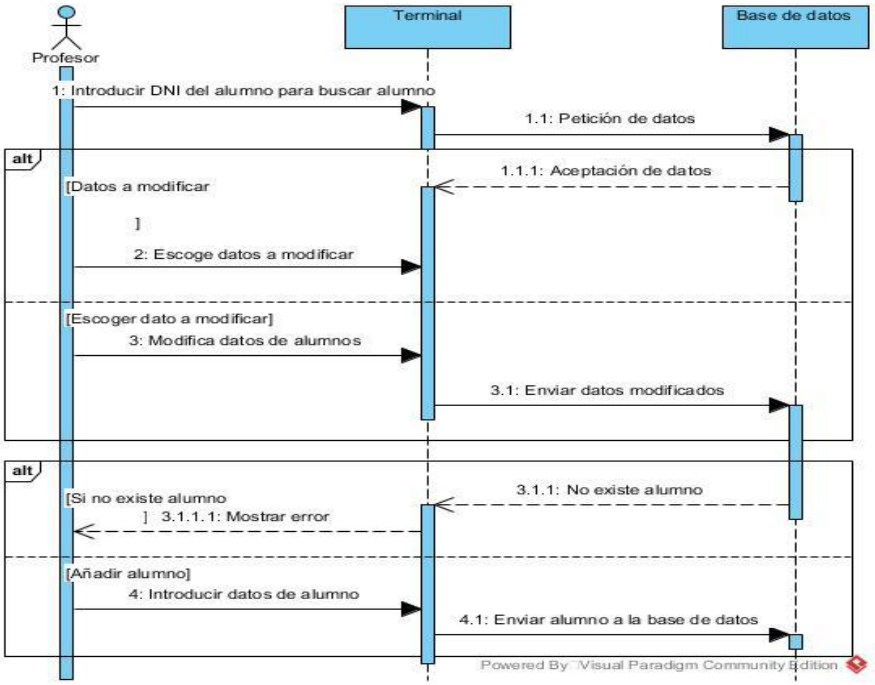
Tras haber sacado las Historias de Usuario y los Casos de Uso, obtenemos, a partir de los Casos de Uso, el Diagrama de Clases. Este diagrama se compone de 5 clases: Persona, Alumno, Profesor, Grupo y Agenda. Dentro de cada clase, encontramos los atributos correspondientes a ella, junto con las funciones que están relacionadas con dicha clase. El diagrama es el siguiente:



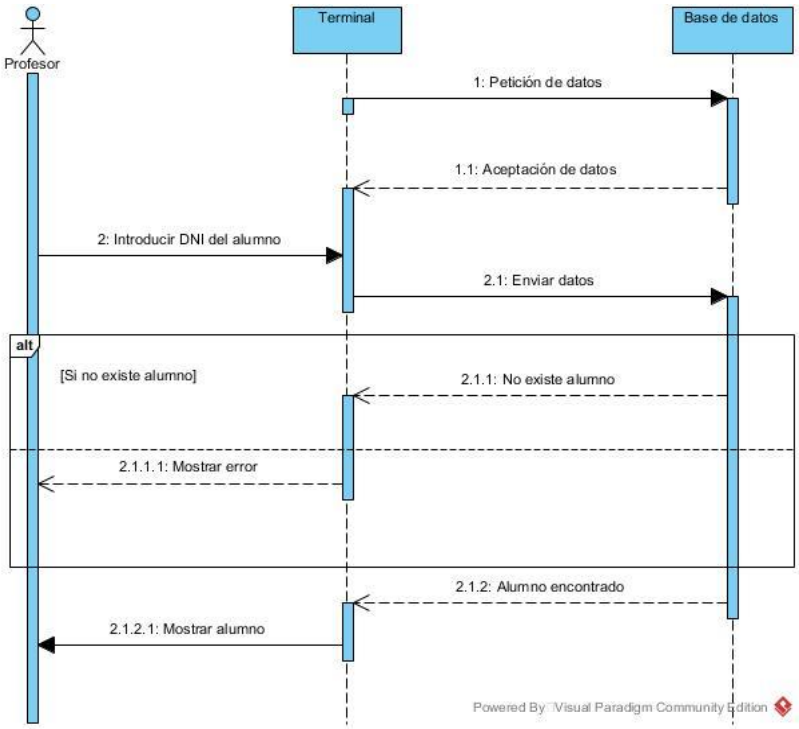
6. Diagramas de Secuencia

Los Diagramas de Secuencia son interpretaciones gráficas de cómo se comportará el sistema para un determinado Caso de Uso. Se han realizado tantos Diagramas de Secuencia como Casos de Uso, y se encuentran ordenados de forma similar a los Casos de Uso. Son los siguientes:

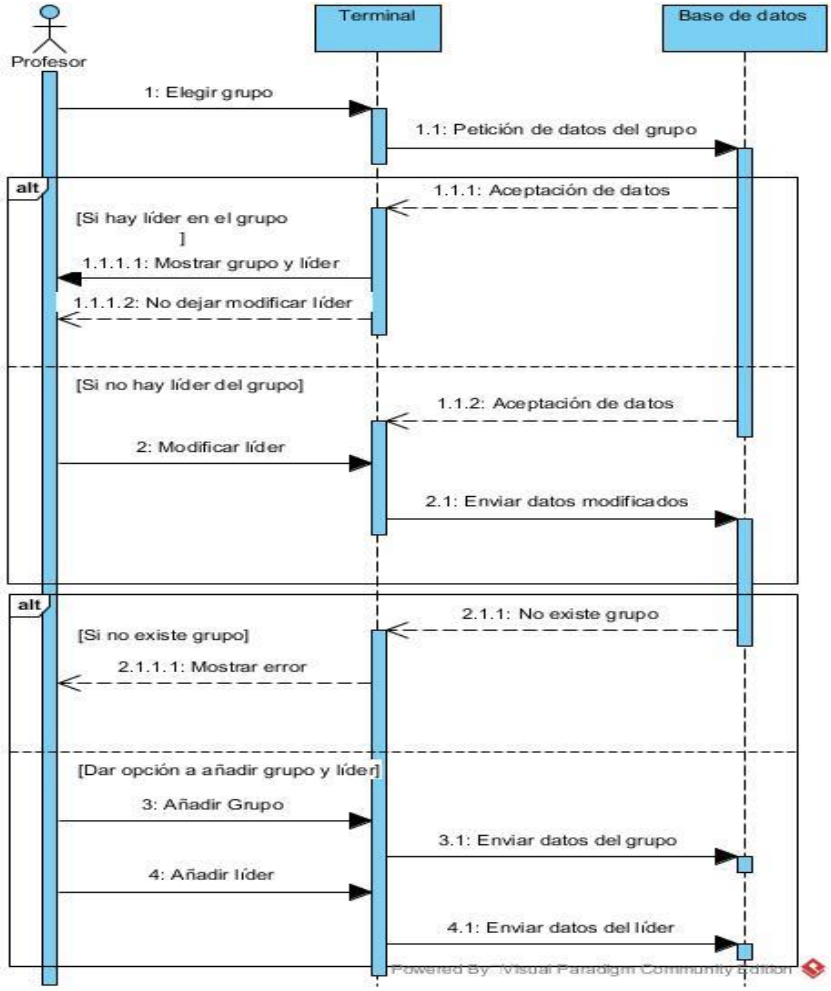
DS01



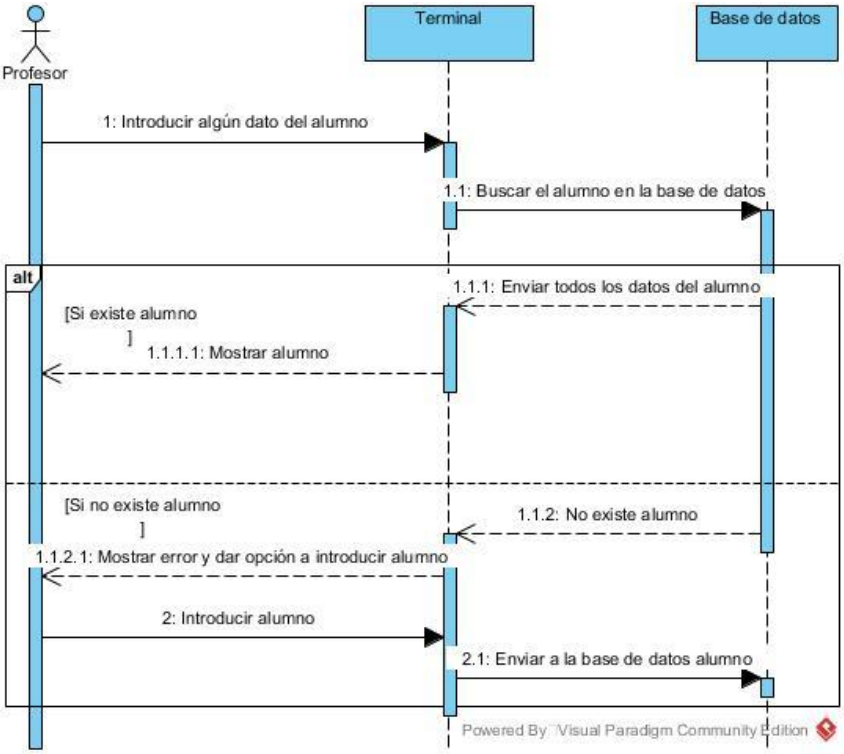
DS02



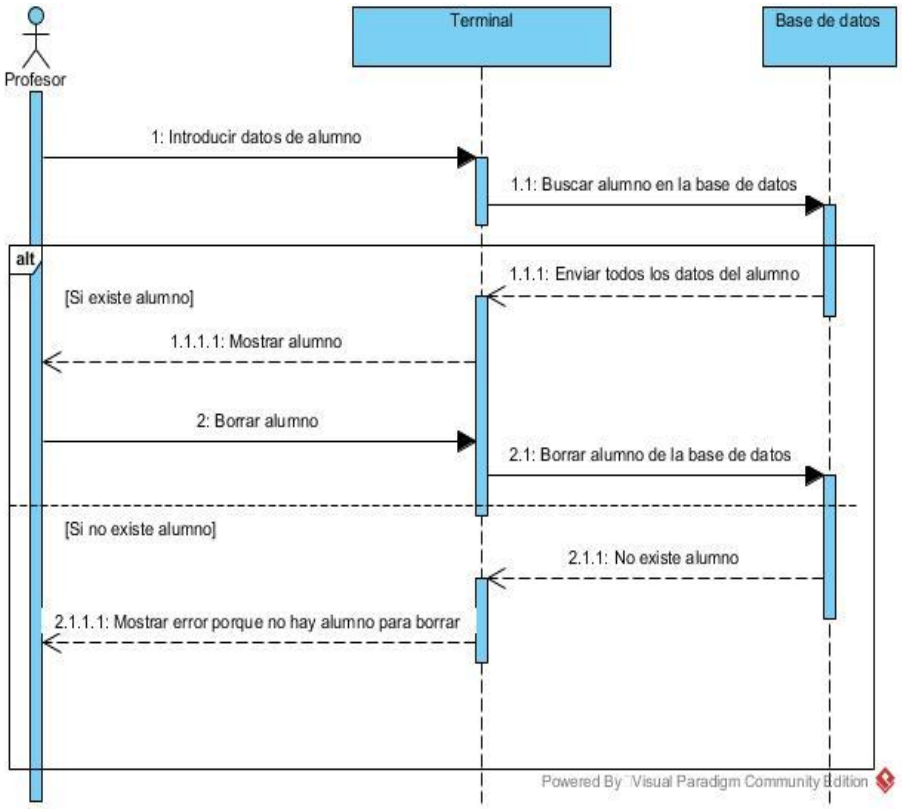
DS03


Profesor
Terminal
Base de datos
1: Elegir grupo
1.1: Petición de datos del grupo
alt
1.1.1: Aceptación de datos
[Si hay líder en el grupo]
1.1.1.1: Mostrar grupo y líder
1.1.1.2: No dejar modificar líder
[Si no hay líder del grupo]
1.1.2: Aceptación de datos
2: Modificar líder
2.1: Enviar datos modificados
alt
2.1.1: No existe grupo
[Si no existe grupo]
2.1.1.1: Mostrar error
[Dar opción a añadir grupo y líder]
3: Añadir Grupo
3.1: Enviar datos del grupo
4: Añadir líder
4.1: Enviar datos del líder
Powered By Visual Paradigm Community Edition

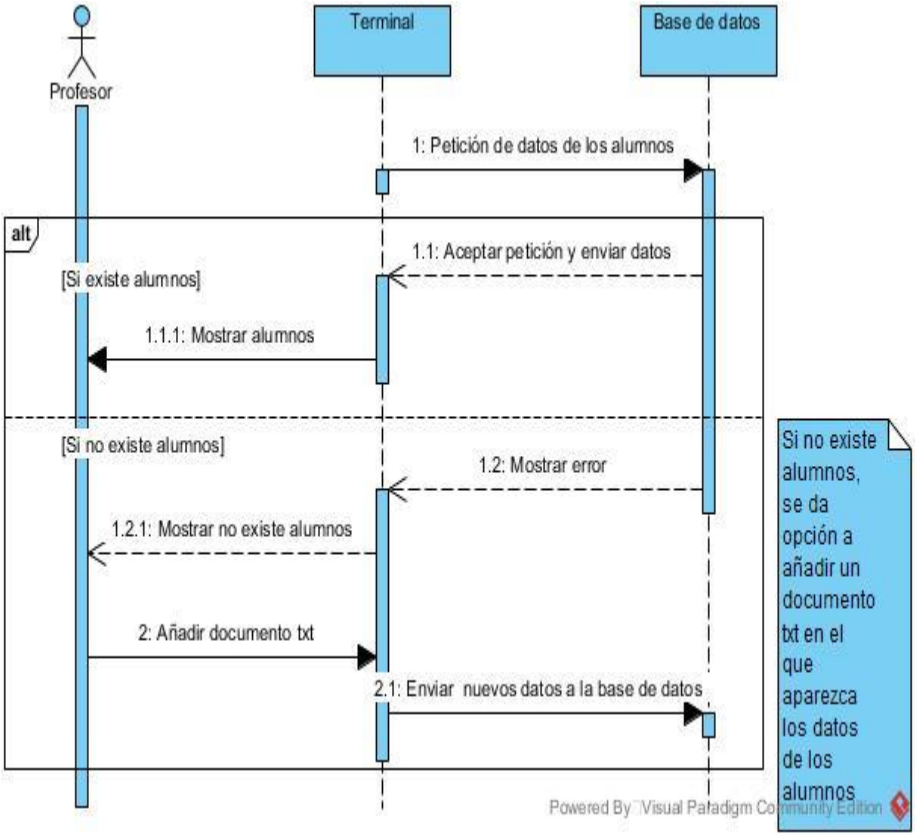
DS04


Profesor
Terminal
Base de datos
1: Introducir algún dato del alumno
1.1: Buscar el alumno en la base de datos
alt
1.1.1: Enviar todos los datos del alumno
[Si existe alumno]
1.1.1.1: Mostrar alumno
[Si no existe alumno]
1.1.2: No existe alumno
1.1.2.1: Mostrar error y dar opción a introducir alumno
2: Introducir alumno
2.1: Enviar a la base de datos alumno
Powered By Visual Paradigm Community Edition

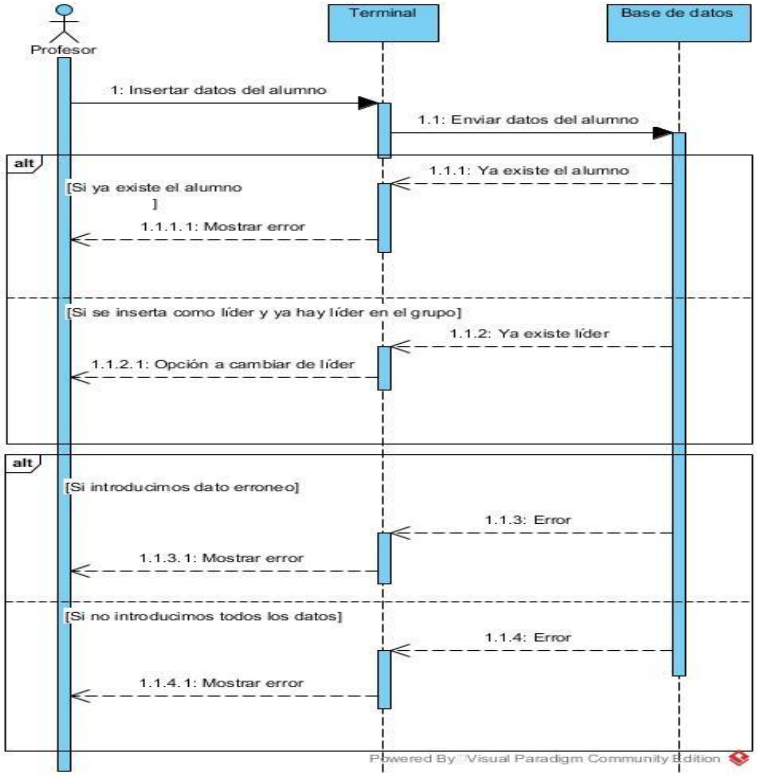
DS05


Profesor
Terminal
Base de datos
1: Introducir datos de alumno
1.1: Buscar alumno en la base de datos
alt
1.1.1: Enviar todos los datos del alumno
[Si existe alumno]
1.1.1.1: Mostrar alumno
2: Borrar alumno
2.1: Borrar alumno de la base de datos
[Si no existe alumno]
2.1.1: No existe alumno
2.1.1.1: Mostrar error porque no hay alumno para borrar
Powered By Visual Paradigm Community Edition

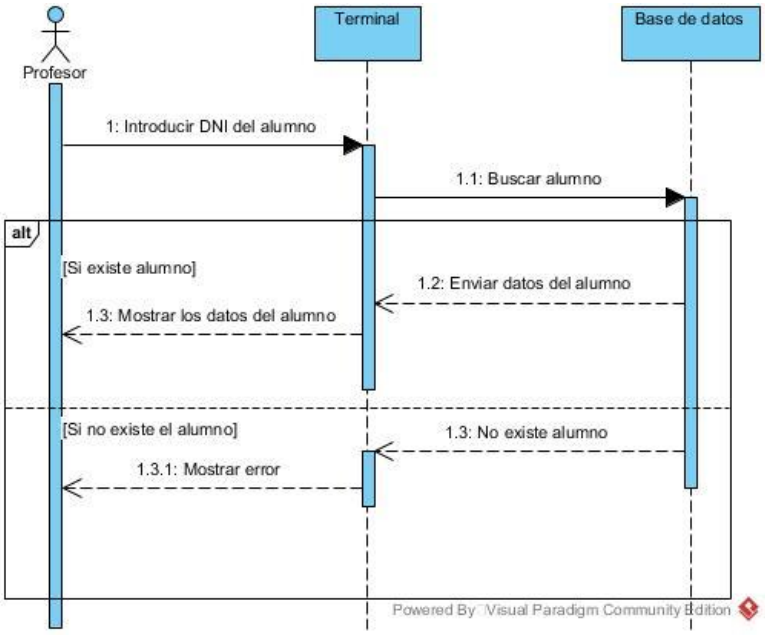
DS06


Profesor
Terminal
Base de datos
1: Petición de datos de los alumnos
alt
1.1: Aceptar petición y enviar datos
[Si existe alumnos]
1.1.1: Mostrar alumnos
[Si no existe alumnos]
1.2: Mostrar error
1.2.1: Mostrar no existe alumnos
2: Añadir documento txt
2.1: Enviar nuevos datos a la base de datos
Si no existe alumnos, se da opción a añadir un documento txt en el que aparezca los datos de los alumnos
Powered By Visual Paradigm Community Edition

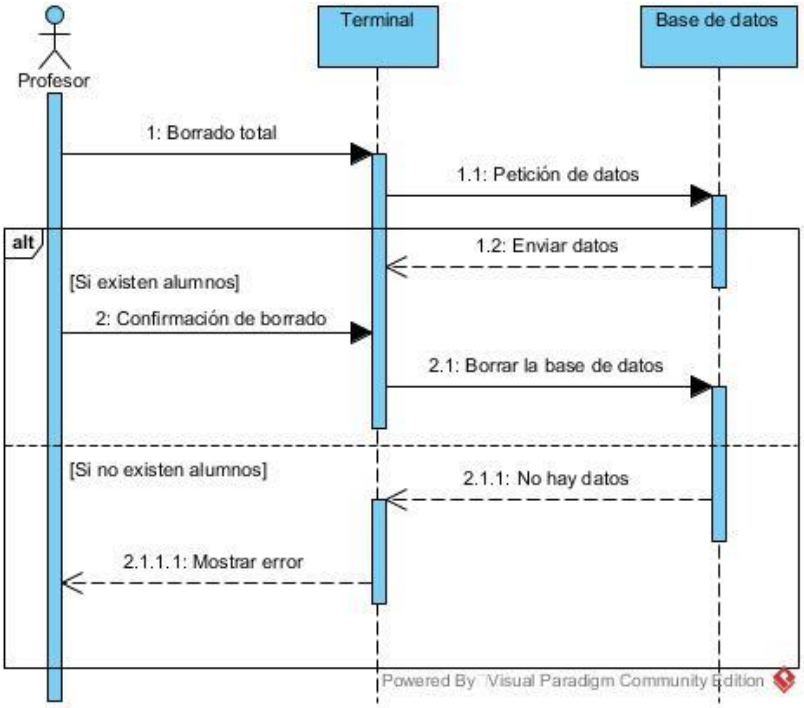
DS07


Profesor
Terminal
Base de datos
1: Insertar datos del alumno
1.1: Enviar datos del alumno
alt
[Si ya existe el alumno
]
1.1.1: Ya existe el alumno
1.1.1.1: Mostrar error
[Si se inserta como líder y ya hay líder en el grupo]
1.1.2: Ya existe líder
1.1.2.1: Opción a cambiar de líder
alt
[Si introducimos dato erroneo]
1.1.3: Error
1.1.3.1: Mostrar error
[Si no introducimos todos los datos]
1.1.4: Error
1.1.4.1: Mostrar error
Powered By Visual Paradigm Community Edition

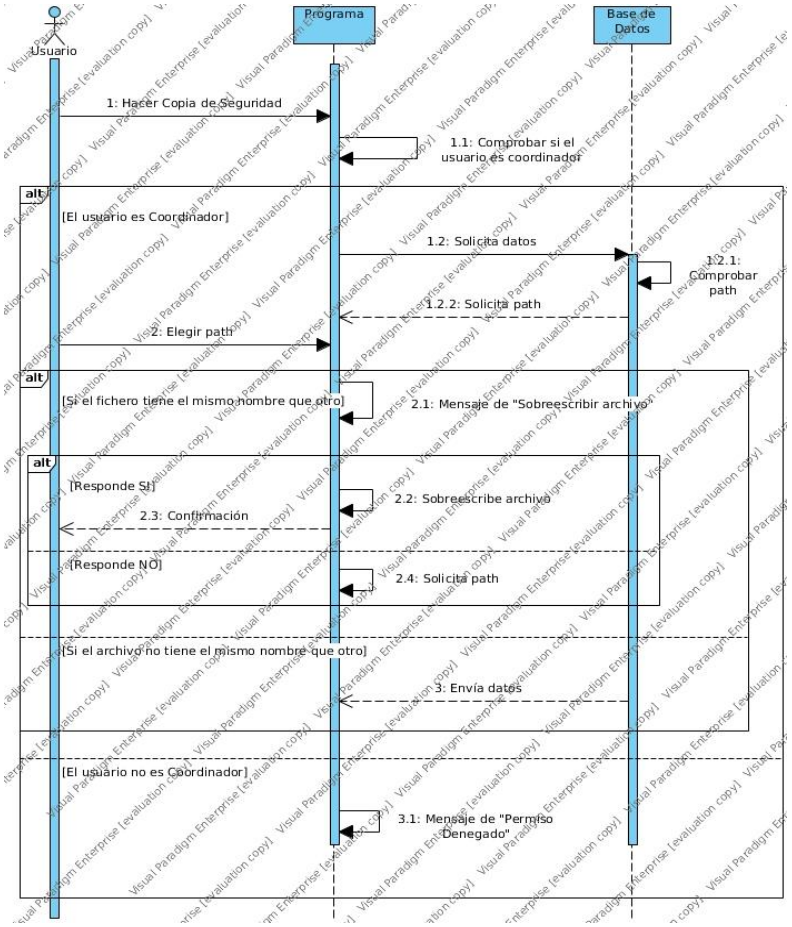
DS08


Profesor
Terminal
Base de datos
1: Introducir DNI del alumno
1.1: Buscar alumno
alt
[Si existe alumno]
1.2: Enviar datos del alumno
1.3: Mostrar los datos del alumno
[Si no existe el alumno]
1.3: No existe alumno
1.3.1: Mostrar error
Powered By Visual Paradigm Community Edition

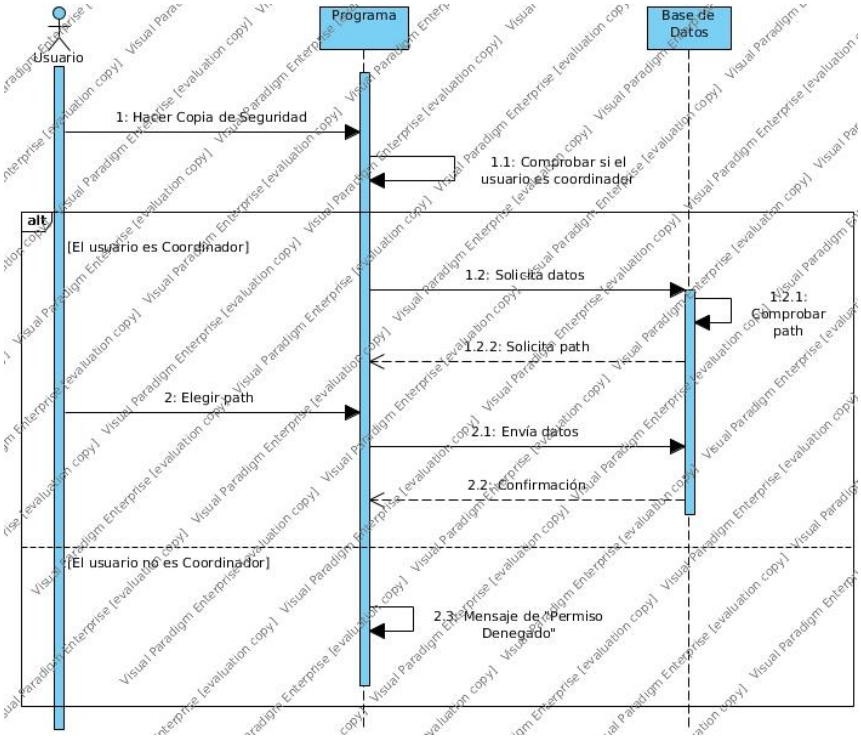
DS09


Profesor
Terminal
Base de datos
1: Borrado total
1.1: Petición de datos
alt
1.2: Enviar datos
[Si existen alumnos]
2: Confirmación de borrado
2.1: Borrar la base de datos
[Si no existen alumnos]
2.1.1: No hay datos
2.1.1.1: Mostrar error
Powered By Visual Paradigm Community Edition

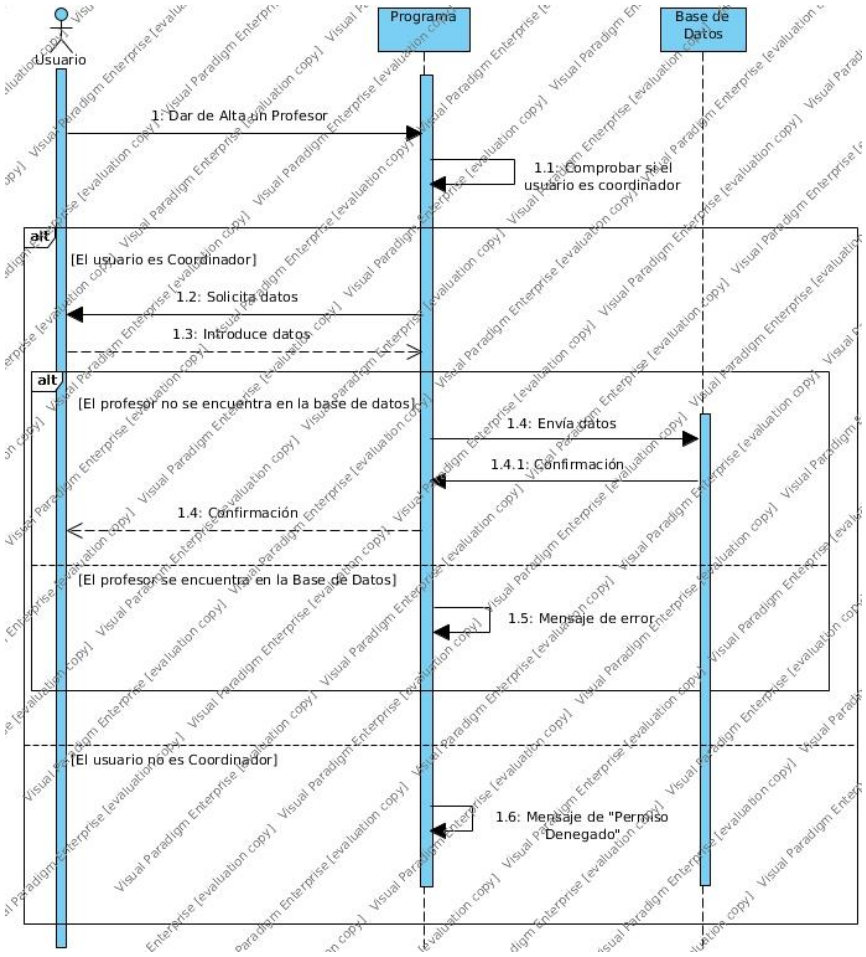
DS10


Usuario
Programa
Base de Datos
1: Hacer Copia de Seguridad
1.1: Comprobar si el usuario es coordinador
alt
[El usuario es Coordinador]
1.2: Solicita datos
1.2.1: Comprobar path
1.2.2: Solicita path
2: Elegir path
alt
[Si el fichero tiene el mismo nombre que otro]
2.1: Mensaje de "Sobreescribir archivo"
alt
[Responde SI]
2.2: Sobreescribe archivo
2.3: Confirmación
[Responde NO]
2.4: Solicita path
[Si el archivo no tiene el mismo nombre que otro]
3: Envía datos
[El usuario no es Coordinador]
3.1: Mensaje de "Permiso Denegado"

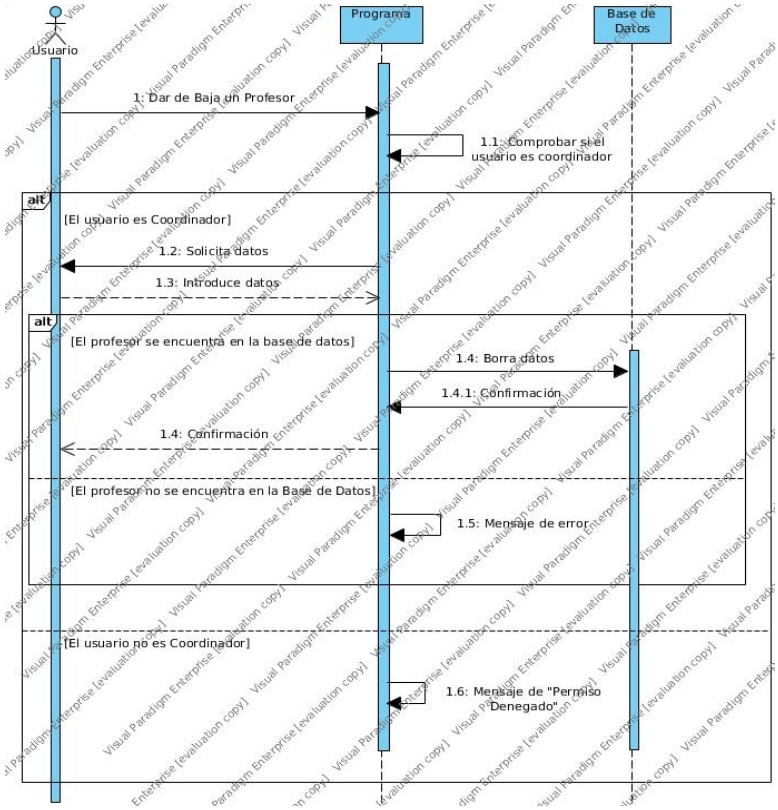
DS11


Usuario
Programa
Base de Datos
1: Hacer Copia de Seguridad
1.1: Comprobar si el usuario es coordinador
alt
[El usuario es Coordinador]
1.2: Solicita datos
1.2.1: Comprobar path
1.2.2: Solicita path
2: Elegir path
2.1: Envía datos
2.2: Confirmación
[El usuario no es Coordinador]
2.3: Mensaje de "Permiso Denegado"

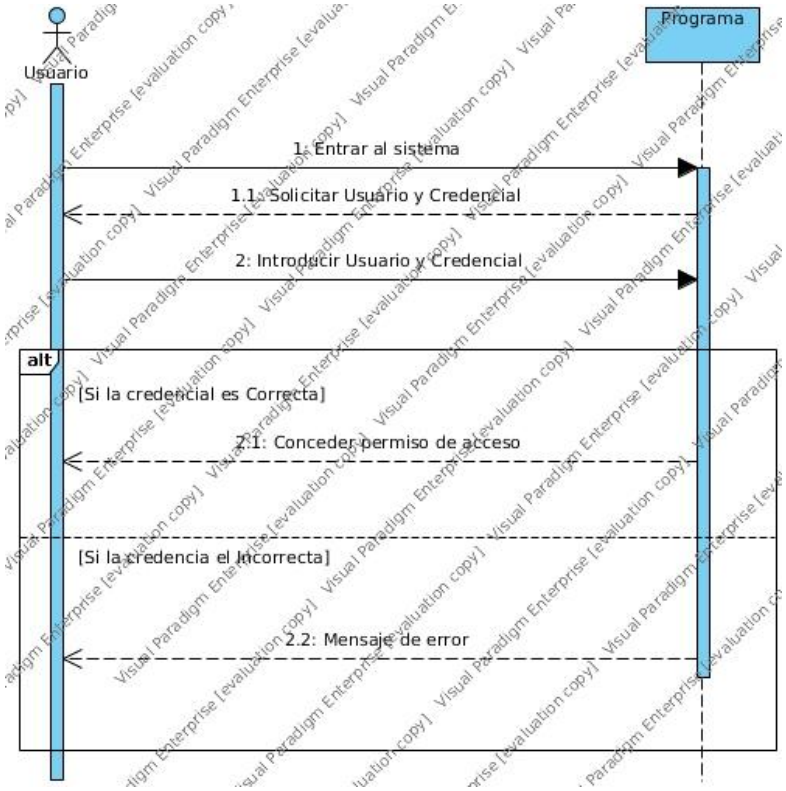
DS12


Usuario
Programa
Base de Datos
1: Dar de Alta un Profesor
1.1: Comprobar si el usuario es coordinador
alt
[El usuario es Coordinador]
1.2: Solicita datos
1.3: Introduce datos
alt
[El profesor no se encuentra en la base de datos]
1.4: Envía datos
1.4.1: Confirmación
1.4: Confirmación
[El profesor se encuentra en la Base de Datos]
1.5: Mensaje de error
[El usuario no es Coordinador]
1.6: Mensaje de "Permiso Denegado"

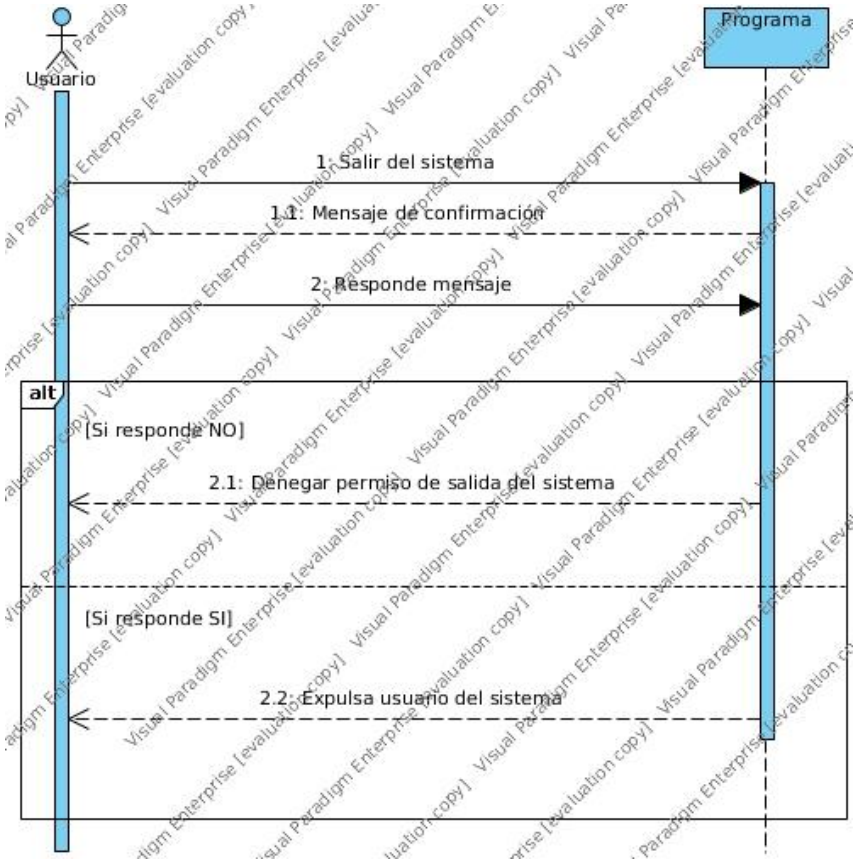
DS13


Usuario
Programa
Base de Datos
1: Dar de Baja un Profesor
1.1: Comprobar si el usuario es coordinador
alt
[El usuario es Coordinador]
1.2: Solicita datos
1.3: Introduce datos
alt
[El profesor se encuentra en la base de datos]
1.4: Borra datos
1.4.1: Confirmación
1.4: Confirmación
[El profesor no se encuentra en la Base de Datos]
1.5: Mensaje de error
[El usuario no es Coordinador]
1.6: Mensaje de "Permiso Denegado"

DS14


Usuario
Programa
1: Entrar al sistema
1.1: Solicitar Usuario y Credencial
2: Introducir Usuario y Credencial
alt
[Si la credencial es Correcta]
2.1: Conceder permiso de acceso
[Si la credencia el Incorrecta]
2.2: Mensaje de error

DS15



7.-Metodología SCRUM

La metodología SCRUM permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Esta metodología está basada en entregas parciales y regulares del producto final en base al valor que ofrecen a los clientes. En nuestro caso, utilizamos para nuestra entrega 3 elementos de la metodología: Product Backlog, Sprint Backlogs y Burndown charts.

7.1.- 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 15 funcionalidades ordenadas por prioridad y en cada funcionalidad su historia de usuario. Además, el tiempo estimado por función.

Las funcionalidades se clasifican en 5 prioridades, que van desde prioridad 0 a prioridad 4. Siendo prioridad 0 lo más esencial del producto.

Las funcionalidades (ordenadas por ID) son las siguientes:

·· Buscar Alumno

·· Modificar Alumno

·· Modificar líder de Grupo

·· Mostrar Alumno

·· Borrar alumno

·· Mostrar todos los alumnos

·· Insertar Alumno

·· Comprobar la existencia de alumnos

·· Borrado total

·· Hacer copia de seguridad

·· Cargar copia de seguridad

·· Dar de alta a un profesor

·· Dar de baja a un profesor

·· Acceder al sistema

·· Salir del sistema

7.2.- Sprint backlogs

Funcionalidad a desarrollar en un sprint determinado.

En nuestro caso, presentamos 3 sprint backlogs. En el primero determinamos a cada integrante del grupo sus funcionalidades a crear. Como en nuestro grupo somos solo 2 integrantes, determinamos a un miembro las funciones de prioridad 2 y 3, mientras que al otro integrante se le determinó las demás funciones y la declaración de clases get y set.

En el segundo determinamos a cada integrante del grupo la tarea de seguir creando sus funcionalidades respectivamente. Pero ya no era necesario la declaración de clases get y set, ya que ya habían sido creadas.

En el tercero y último determinamos a un miembro a acabar las funcionalidades de modificar alumno, modificar líder, dar de alta a un profesor, dar de baja a un profesor y mostrar alumno, además de comprobar las funciones y corregir respectivos fallos. Mientras que al otro miembro solo se le determinó comprobar fallos en sus funciones y constructores.

7.3.- Burndown charts

Gráfico que muestra la cantidad de trabajo hecho.

En nuestro caso, este gráfico muestra 20 horas de estimación para hacer el producto.

Los integrantes del grupo empiezan a desarrollar sus funcionalidades el 4 de Diciembre y para el 7 de Diciembre han acortado 5 horas de trabajo. Mientras que desde esa fecha hasta el 13 de Diciembre se acortan 10 horas de trabajo. Y para acabar el producto se dedican 5 horas desde el 13 de Diciembre hasta el 14 de Diciembre, en la que se corrigen errores en las funcionalidades y constructores.

8.-Matriz de validación

8.1.- Casos de uso-Requisitos

8.1.- Casos de uso-Requisitos

Casos de Uso/Requisitos	CU01	CU02	CU03	CU04	CU05	CU6	CU07	CU08	CU09	CU10	CU11	CU12	CU13	CU14	CU15
Requisito 1	X														
Requisito 2		X													
Requisito 3			X												
Requisito 4				X											
Requisito 5					X										
Requisito 6						X									
Requisito 7							X								
Requisito 8								X							
Requisito 9									X						
Requisito 10										X					
Requisito 11											X				
Requisito 12												X			
Requisito 13													X		
Requisito 14														X	
Requisito15															X

8.2.- Clases-Caso de Uso

Casos de Uso/ Clases	CU01	CU02	CU03	CU04	CU05	CU6	CU07	CU08	CU09	CU10	CU11	CU12	CU13	CU14	CU15
Agenda	X	X		X	X	X	X	X	X						
Grupo			X												
Profesor										X	X	X	X	X	X

9.-Bibliografía

-Material de Moodle:

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

-Material para código y funciones:

http://www.cplusplus.com/

-Material para diagramas:

https://www.youtube.com/watch?v=Q1kH7XKxK5I

-Material de GitHub:

https://moodle.uco.es/m1819/pluginfile.php/110517/mod_resource/content/2/P1_IngenieriaSoftware_Presentacion.pdf