|  |  |
| --- | --- |
| logo | Universidad de Córdoba |
|  |  |
|  |  |
|  |  |
| GRADO DE INGENIERÍA INFORMÁTICA | |
| INGENIERÍA DEL SOFTWARE | |
|  | |

|  |
| --- |
|  |
| DOCUMENTACIÓN PROCESO SCRUM DESARROLLO Appgestionalumnos g-36 |
|  |
|  |
| El objeto del presente documento es presentar la documentación del proceso de desarrollo basado en la metodología ágil SCRUM de la aplicación AppGestionAlumnos encargado por el profesorado de las prácticas de la asignatura Ingeniería del Software de 2º curso del Grado de Informática de la Universidad de Córdoba. |
|  |

|  |
| --- |
|  |
| Cantero Alén, Rafael  Freire Caballero, Carlos  Herrera Poch, Fernando |
| *Fecha: 24/12/2018* |
|  |

Índice de contenidos

[Índice de contenidos - 2 -](#_Toc533425889)

[1 Análisis y diseño del problema 4](#_Toc533425890)

[1.1 Definición del Problema 4](#_Toc533425891)

[1.2 Extracción de Requisitos 4](#_Toc533425892)

[1.2.1 Metodología de desarrollo y de extracción de requisitos 4](#_Toc533425893)

[1.2.2 Personal implicado. 5](#_Toc533425894)

[1.2.3 Datos almacenados por la aplicación 5](#_Toc533425895)

[1.2.4 Requisitos del sistema 6](#_Toc533425896)

[1.2.5 Historias de Usuario 8](#_Toc533425897)

[1.2.6 Casos de Uso 15](#_Toc533425898)

[1.3 Diagrama de Clases 30](#_Toc533425947)

[1.4 Diagramas de Secuencia 32](#_Toc533425948)

[2- Desarrollo de la aplicación – metodología scrum 45](#_Toc533425949)

[2.1 Introducción 45](#_Toc533425950)

[2.2 Product BackLog 45](#_Toc533425951)

[2.3 Sprints Review meeting 47](#_Toc533425952)

[2.3.1 Sprints BackLogs 47](#_Toc533425953)

[2.3.1.1 Sprint BackLog 1 48](#_Toc533425954)

[2.3.1.2 Sprint BackLog 2 48](#_Toc533425955)

[2.3.1.3 Sprint BackLog 3 49](#_Toc533425956)

[2.3.2 Sprints Meetings 50](#_Toc533425957)

[2.3.2.1 Sprint 1 Meeting 1 50](#_Toc533425958)

[2.3.2.2 Sprint 1 Meeting 2 51](#_Toc533425959)

[2.3.2.3 Sprint 1 Meeting 3 52](#_Toc533425960)

[2.3.2.4 Sprint 2 Meeting 1 53](#_Toc533425961)

[2.3.2.5 Sprint 2 Meeting 2 55](#_Toc533425962)

[2.3.2.6 Sprint 2 Meeting 3 56](#_Toc533425963)

[2.3.2.7 Sprint 3 Meeting 1 57](#_Toc533425964)

[2.3.2.8 Sprint 3 Meeting 2 59](#_Toc533425965)

[2.3.2.9 Sprint 3 Meeting 3 60](#_Toc533425966)

[2.3.3 Burndown Chart 61](#_Toc533425967)

[2.3.4 Sprint Retrospective Meeting 61](#_Toc533425968)

[3-VERIFICACIÓN DE REQUISITOS 65](#_Toc533425970)

[3.1.1 Matrices de Verificación. 65](#_Toc533425971)

[Bibliografía y referencias Web 67](#_Toc533425972)

|  |  |
| --- | --- |
|  |  |
|  | análisis y diseño del problema |
|  |  |

# Definición del Problema

Los profesores de la asignatura de prácticas nos plantean la necesidad de disponer de un sistema de gestión de los alumnos que cursan la asignatura. Con carácter general nos informan que les interesa mantener información de los grupos que se forman de prácticas indicando qué alumno es el representante o líder de la misma.

# Extracción de Requisitos

## Metodología de desarrollo y de extracción de requisitos

El desarrollo del sistema software se realiza de acuerdo a la Metodología Ágil Scrum.

La extracción de los requisitos se realizó en una primera reunión con todos los equipos de desarrollo en la que pudimos realizar preguntas para una adecuada comprensión de las necesidades del sistema a desarrollar. A lo largo de las sesiones prácticas y por correo electrónico se pudo afinar los detalles de los requisitos y tras el análisis de la información facilitada obtuvimos los requisitos funcionales y no funcionales de la aplicación. Los requisitos finales de la aplicación se obtuvieron de forma escalonada, de acuerdo a la realidad del desarrollo Ágil. Al inicio de la sesión práctica 3 se incorporaron nuevos requisitos funcionales de la aplicación.

Analizando los requisitos de usuario, las personas involucradas y los datos que se almacenarían de la aplicación pudimos obtener los requisitos del sistema. Realizamos una especificación de los mismos estableciendo su prioridad y creando sus correspondientes Historias de Usuario.

## Personal implicado.

**Profesores:** Gestionan y mantienen la base de datos de los alumnos. Formado por el coordinador y los ayudantes. Las funciones que realizan son las mismas, salvo que el coordinador puede realizar copias de seguridad externas.

**Alumnos:** Cursan la asignatura impartida por el profesor.

## Datos almacenados por la aplicación

El sistema almacenará los siguientes parámetros correspondientes a un alumno:

*DNI.*

Nombre.

*Apellidos.*

Fecha de nacimiento.

*Teléfono.*

Email Corporativo.

*Domicilio.*

Curso más alto en el que se ha matriculado.

*Nota.*

**Equipo al que pertenece.**

**Líder (Sí/No).**

Los indicados en negrita son datos no obligatorios.

## Requisitos del sistema

En primer lugar, se presentan los requisitos funcionales extraídos para la aplicación:

**0 Acceso a la aplicación:** Los profesores podrán acceder a la aplicación mediante sus credenciales de la Universidad.

**1 Introducir un alumno:** Los profesores podrán introducir un nuevo alumno.

**2 Buscar un alumno:** El sistema deberá buscar un alumno o varios de acuerdo a unas directrices.

**3 Modificar un alumno:** Los profesores tendrán la opción de modificar los datos de un alumno.

**4 Gestión de líderes:** El sistema dará la posibilidad de gestionar los líderes de los grupos de acuerdo a las restricciones impuestas por el cliente (no pudiendo haber más de un líder por grupo).

**5 Mostrar alumnos:** El sistema podrá mostrar uno, varios o todos los alumnos.

**6 Eliminar un alumno:** El sistema permitirá borrar un alumno de la base de datos.

**7 Guardar base de datos:** El sistema dará la opción de guardar los datos introducidos en la aplicación en un fichero binario.

**8 Cargar base de datos:** El sistema permite cargar el fichero binario con los datos introducidos.

**9 Guardar copia de seguridad externa:** El sistema dará la opción de realizar una copia de seguridad en un servidor externo con los datos de la aplicación.

**10 Cargar copia de seguridad externa:** El sistema también dará la opción de cargar la copia de seguridad externa.

**11 Salir de la aplicación:** El sistema consultará si se desea guardar la base de datos o bien realizar la copia de seguridad externa, sólo para el caso del Coordinador.

Los requisitos funcionales 9 y 10 fueron incorporados por los profesores de la asignatura en la sesión 3 de prácticas.

Las prioridades establecidas para cada Requisito funcional fueron las que se indican en la siguiente tabla:



**Tabla 1 Requisitos funcionales**

Los requisitos no funcionales de la aplicación son

**1.** Sistema operativo Linux obligatorio, multiplataforma opcional.

**2.** Lenguaje de Programación C++.

**3.** Lenguaje de documentación Markdown.

**4.** Utilización de ficheros binarios para almacenar la información, tanto la de los alumnos como las credenciales de los profesores.

**5.** Utilización de Drive para almacenar la información en un servidor externo.

**6.** Máximo 150 alumnos.

**7.** Interfaz obligatoria línea de comandos, opcional gráfica.

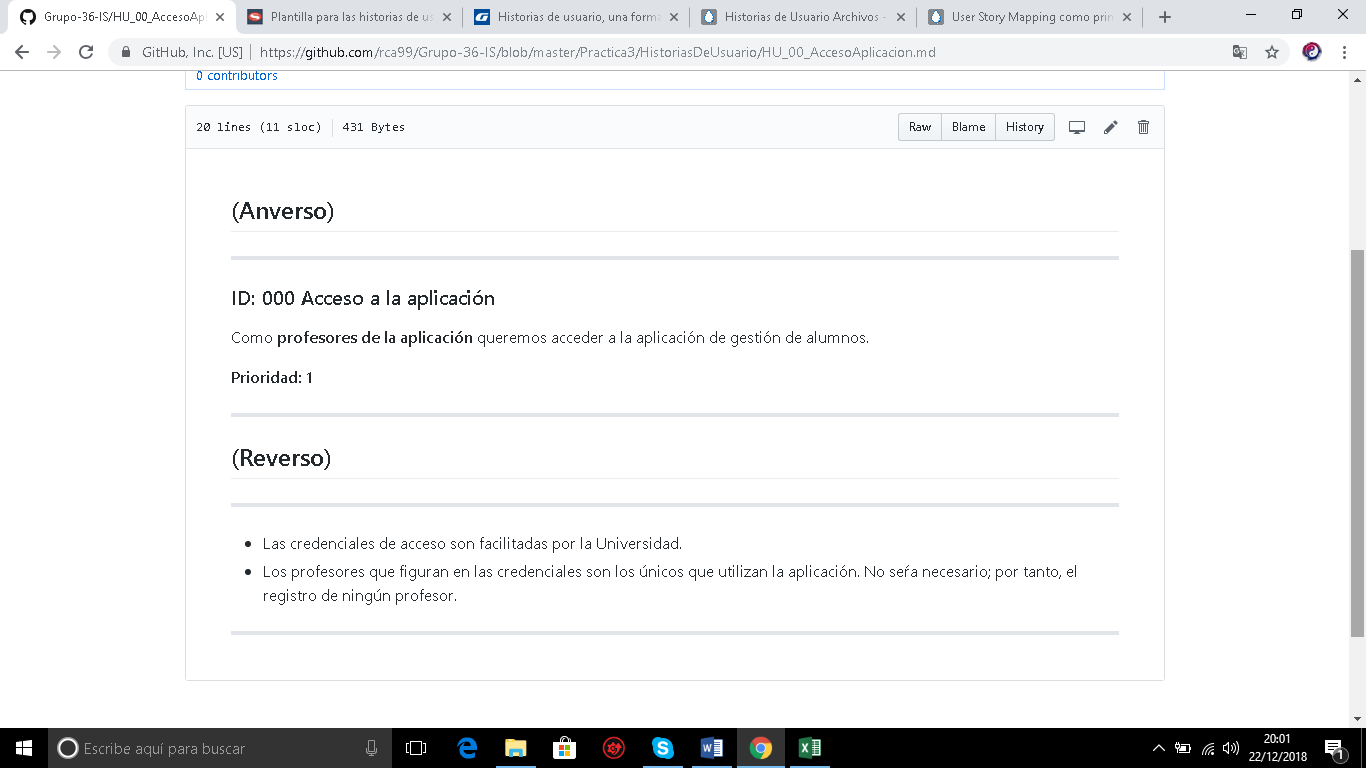
**8.** Únicamente sirve para el año actual.

**9.** Tendrán acceso solo aquellos profesores que figuren en las credenciales cedidas por la Universidad que incluyen usuario, contraseña y rol.

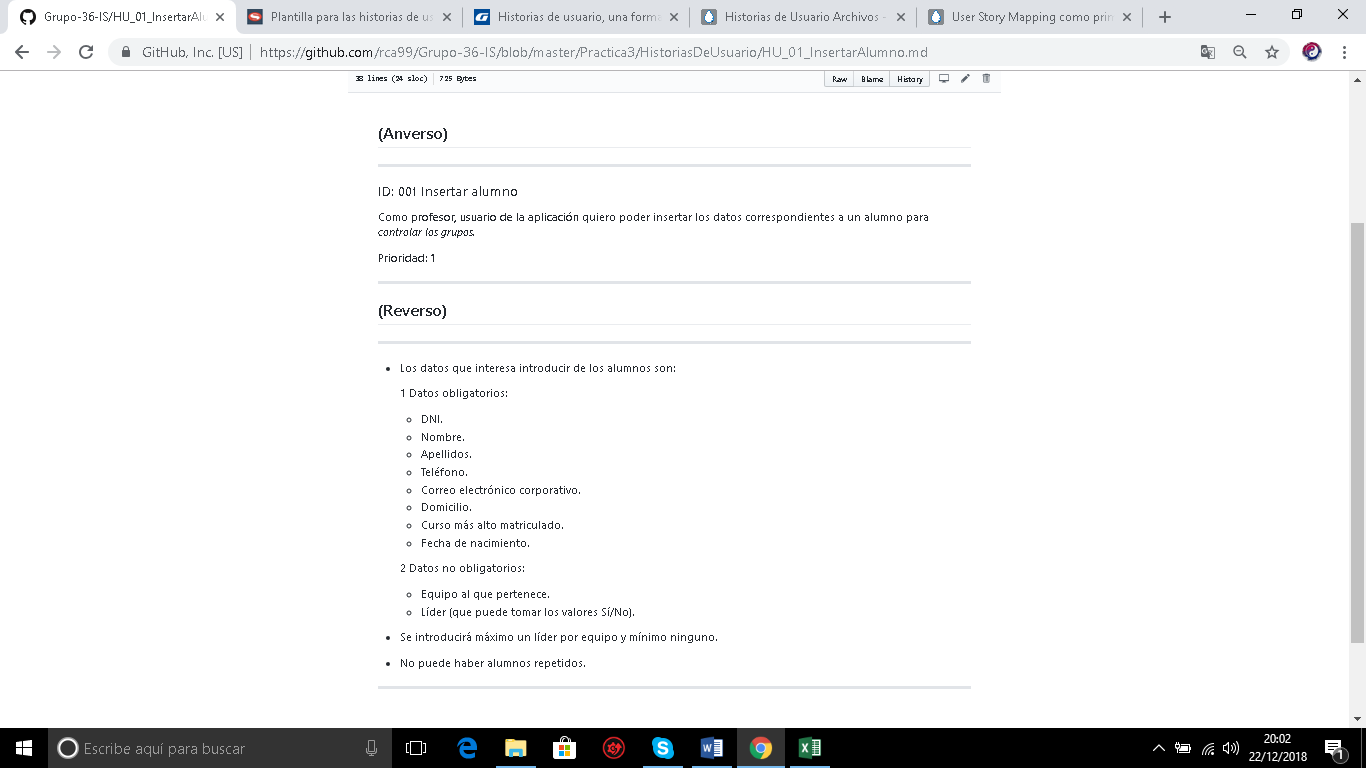
## Historias de Usuario

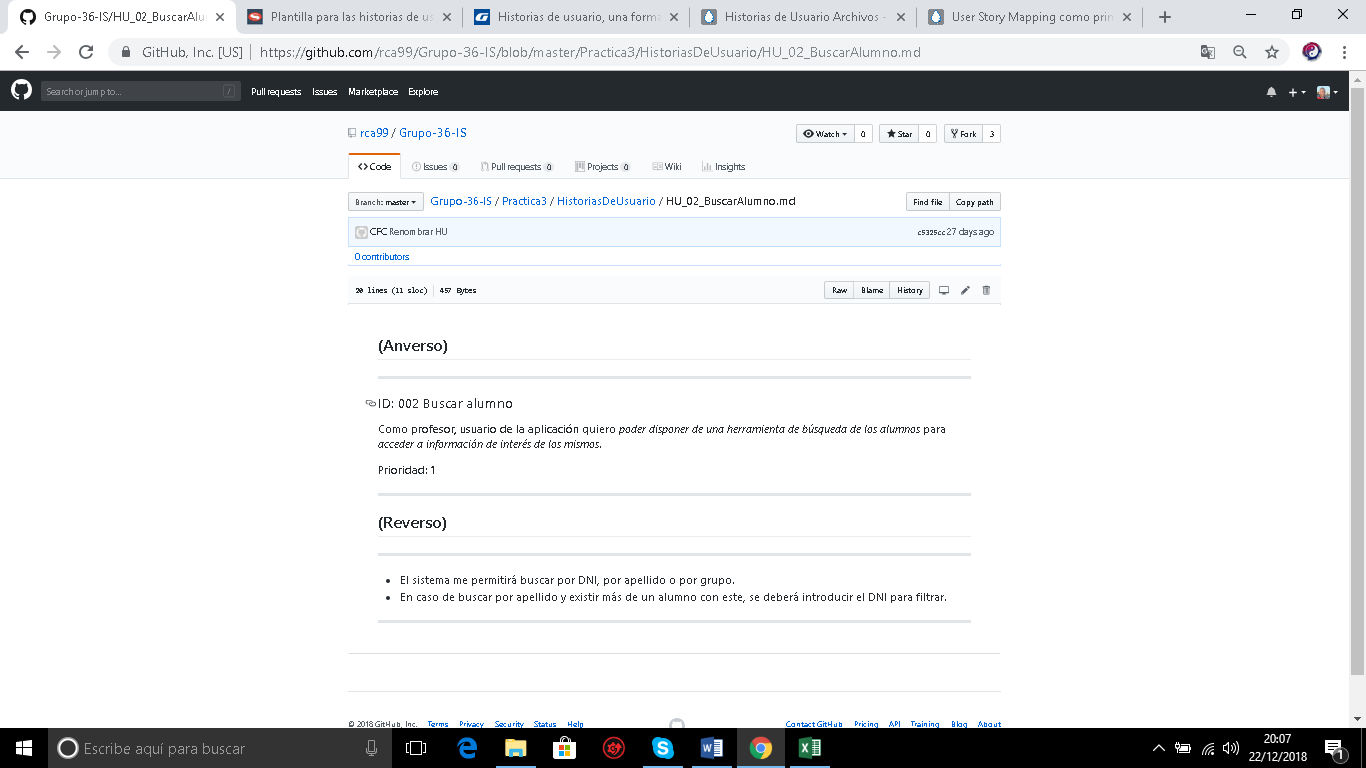
Una vez definidos los requisitos funcionales pasamos a definir las

Historias de usuario como elemento básico para aplicar la metología Scrum. En las mismas describiremos qué requisito dará cumplimiento y los criterios de validación de los mismos.

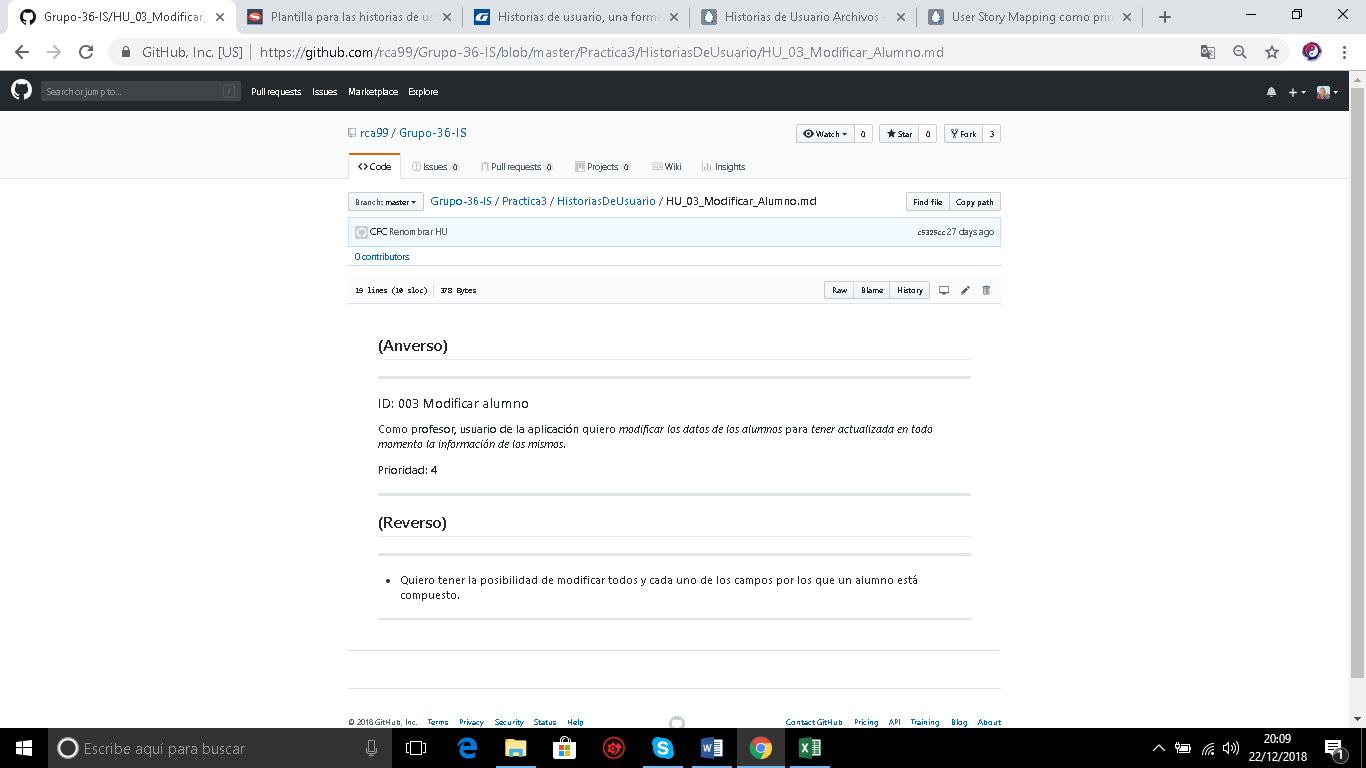
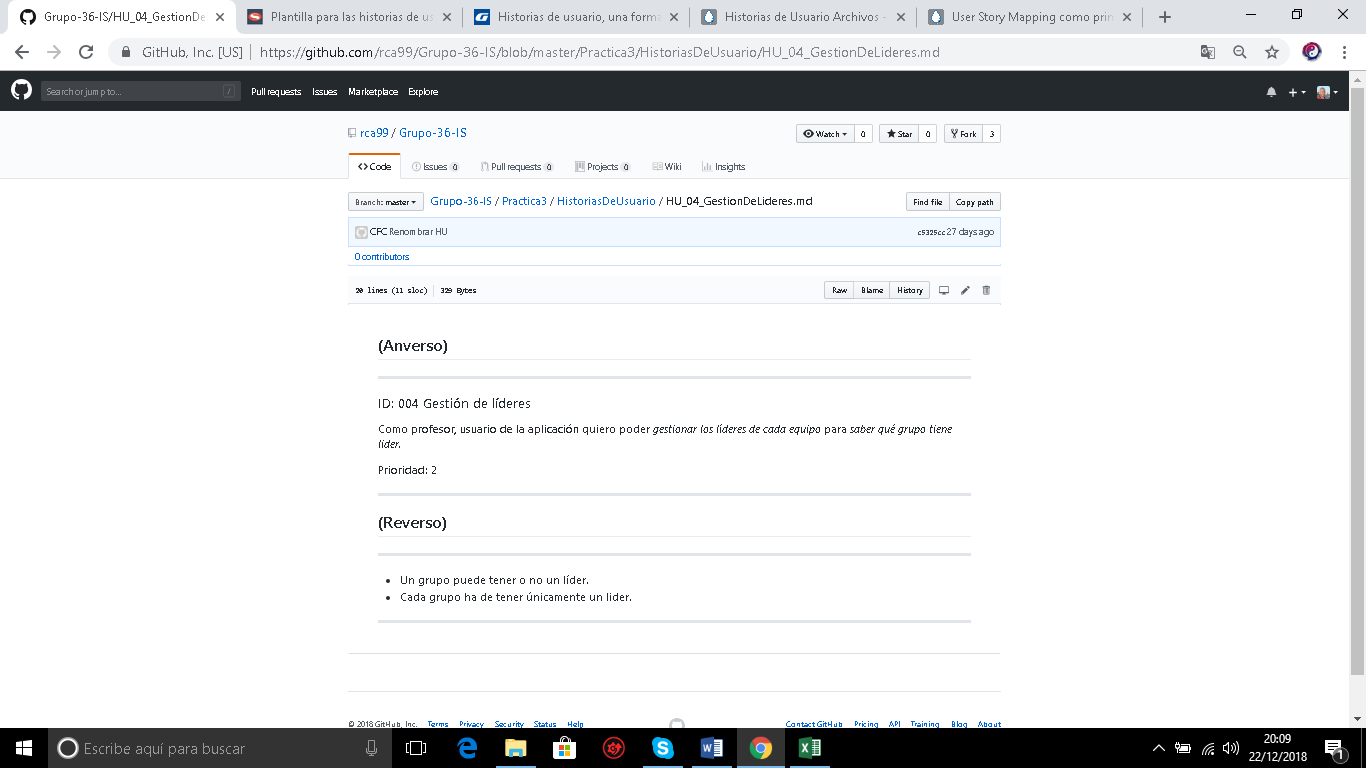


**Ilustración 1 Historia Uso 000 Acceso a la aplicación**



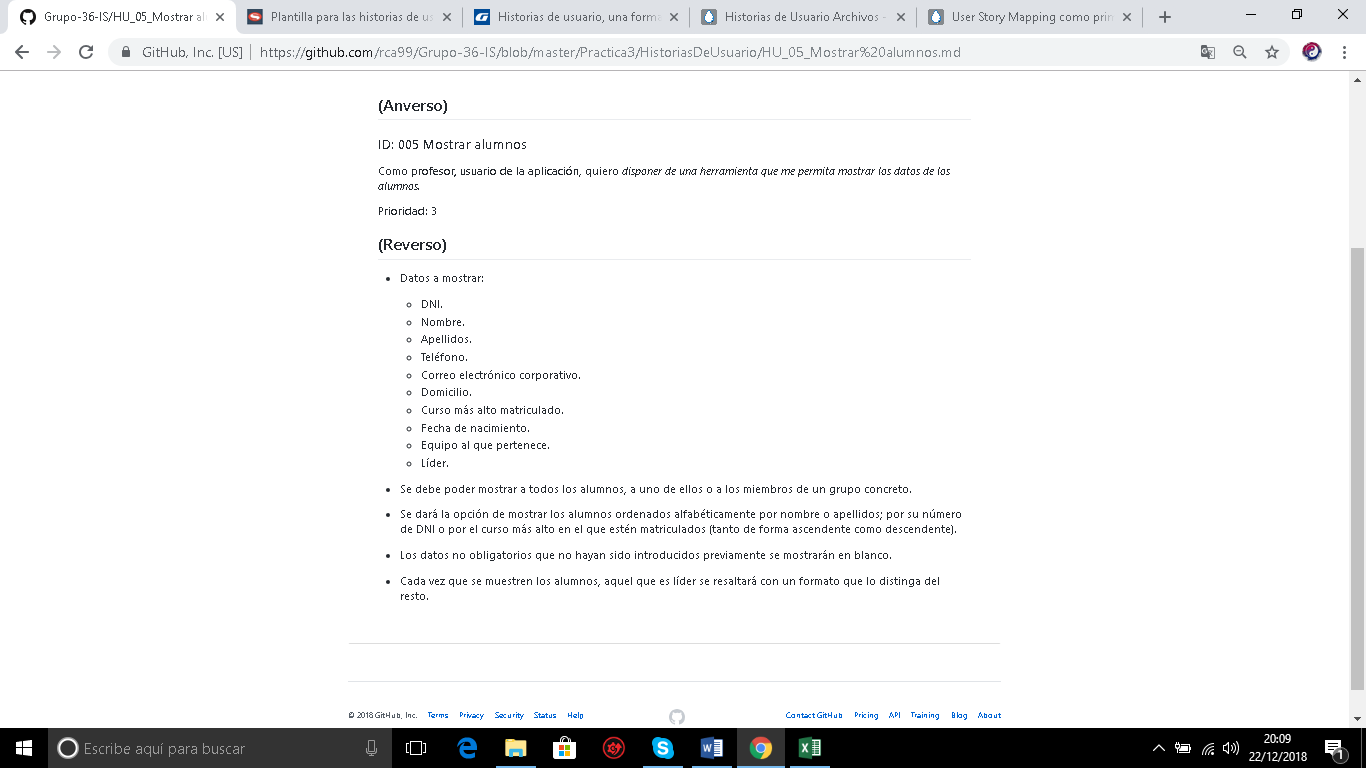
**Ilustración 2 Historia Uso 001 Insertar alumno**

**Ilustración 3 Historia Uso 002 Buscar alumno**

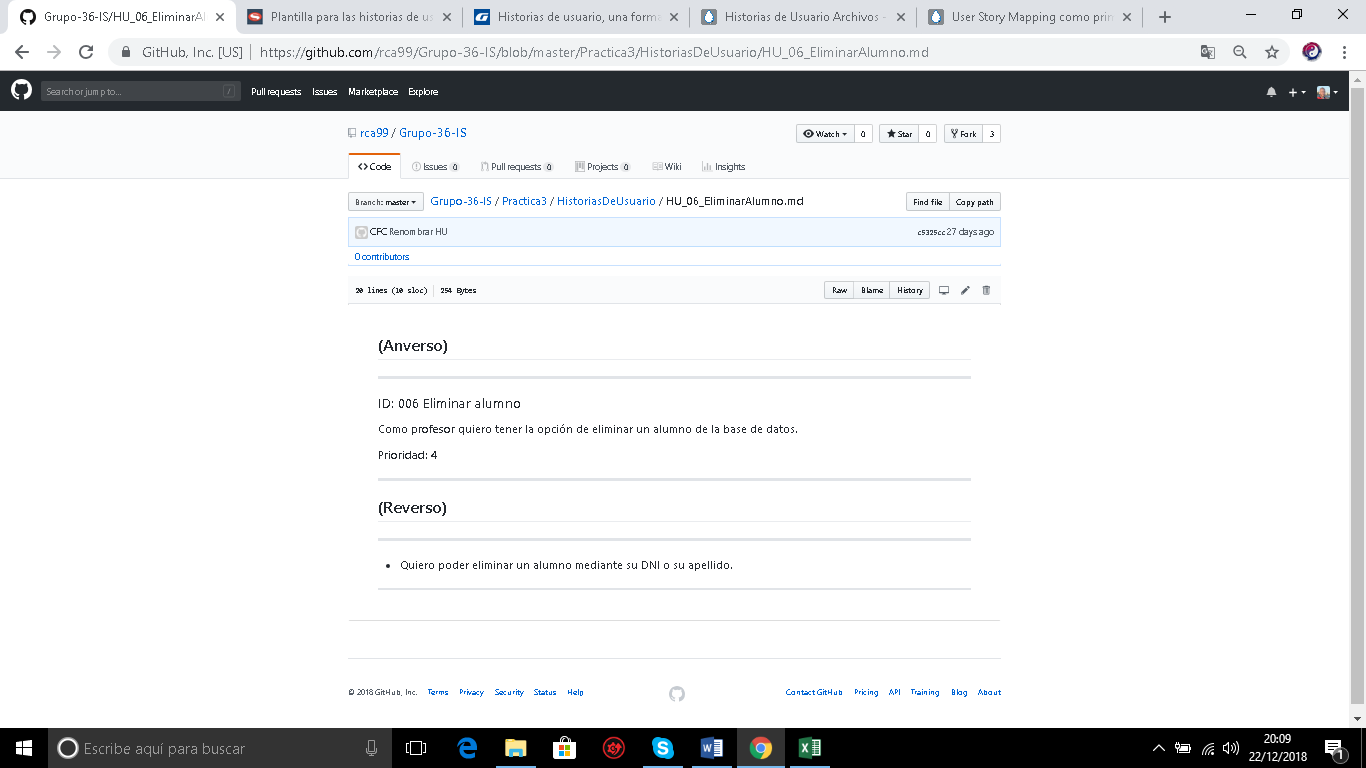


**Ilustración 5 Historia Uso 004 Gestión de Líderes**

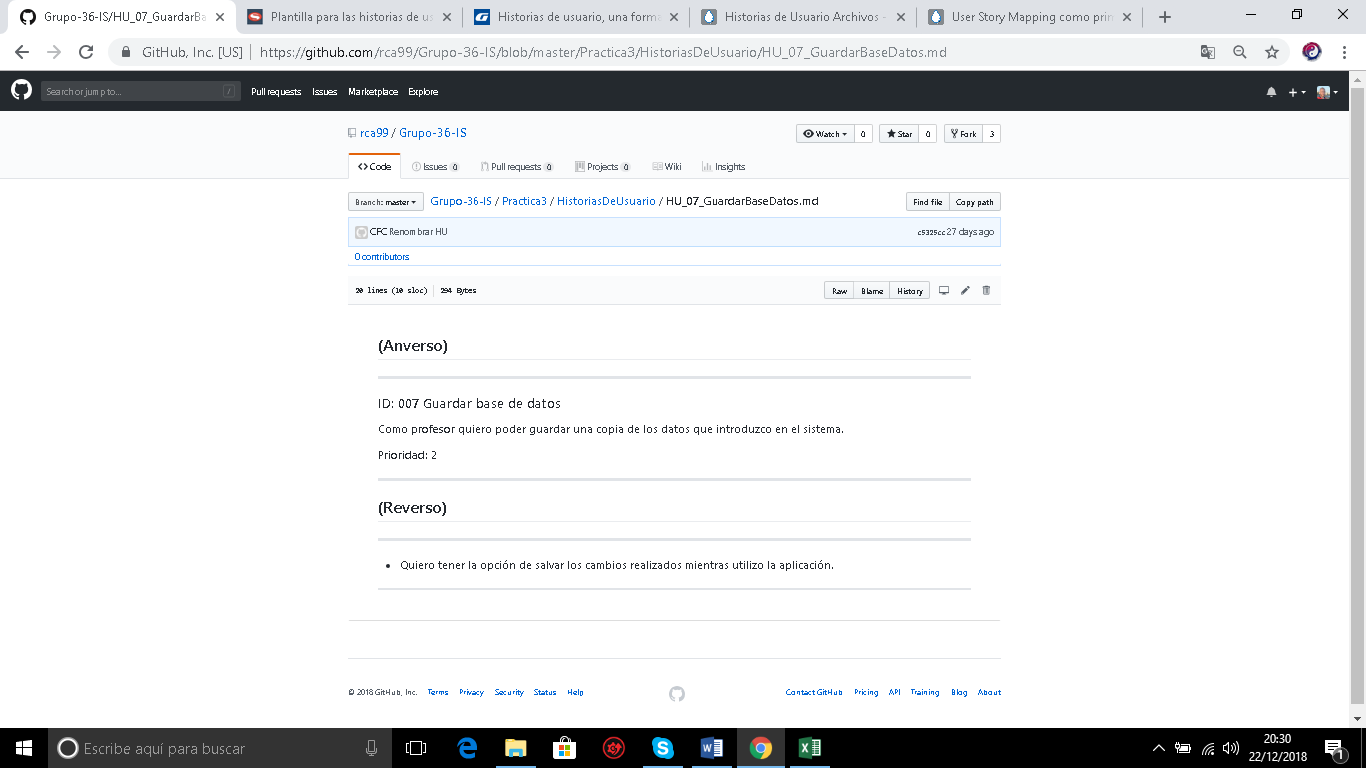
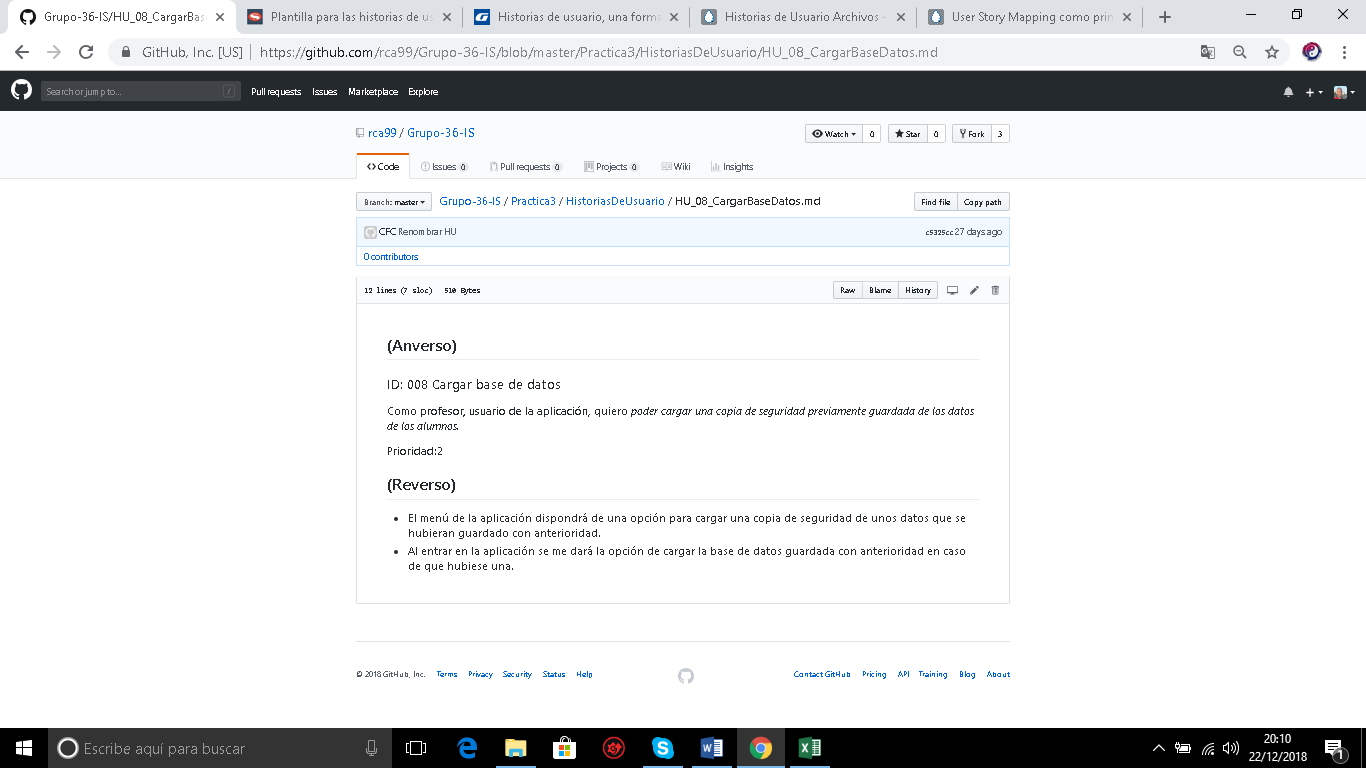
**Ilustración 4 Historia Uso 003 Modificar alumno**



**Ilustración 6 Historia Uso 005 Mostrar alumno**

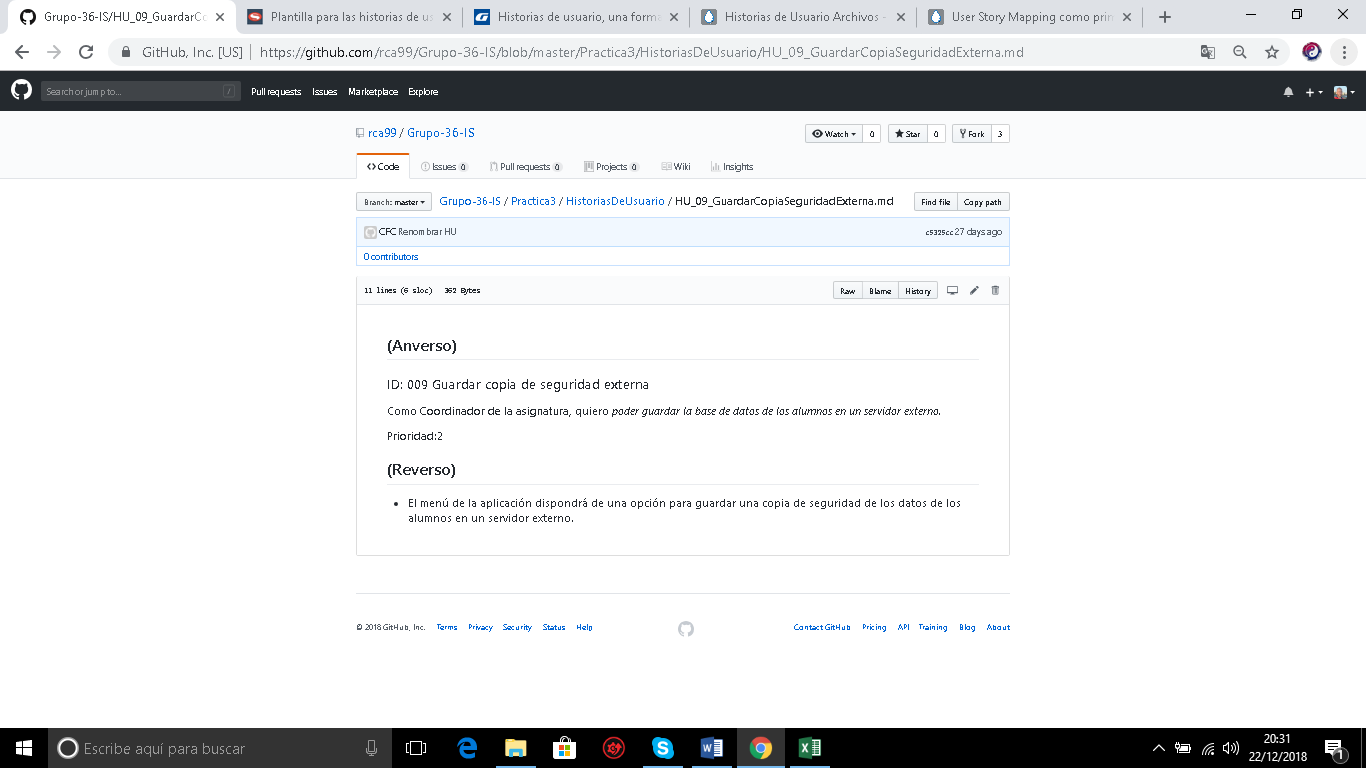
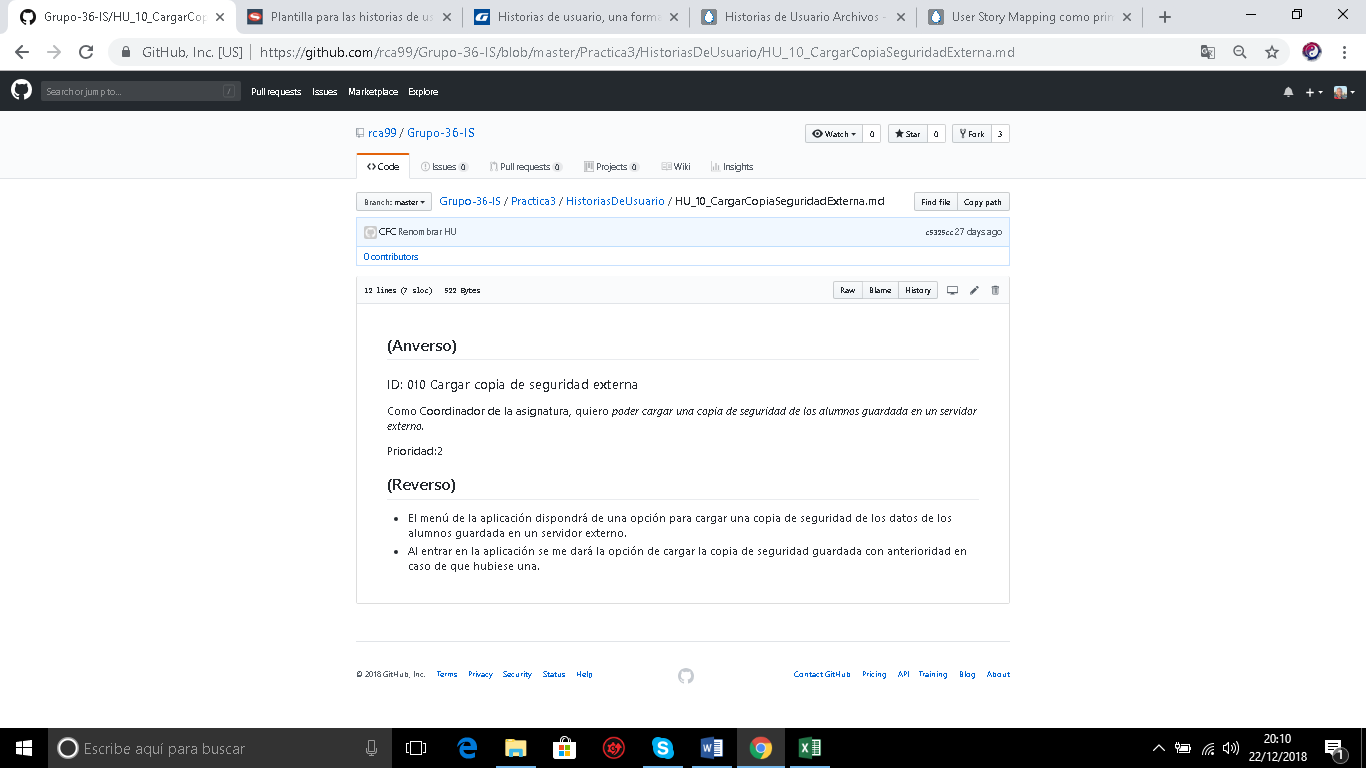


**Ilustración 7 Historia Uso 006 Eliminar alumno**



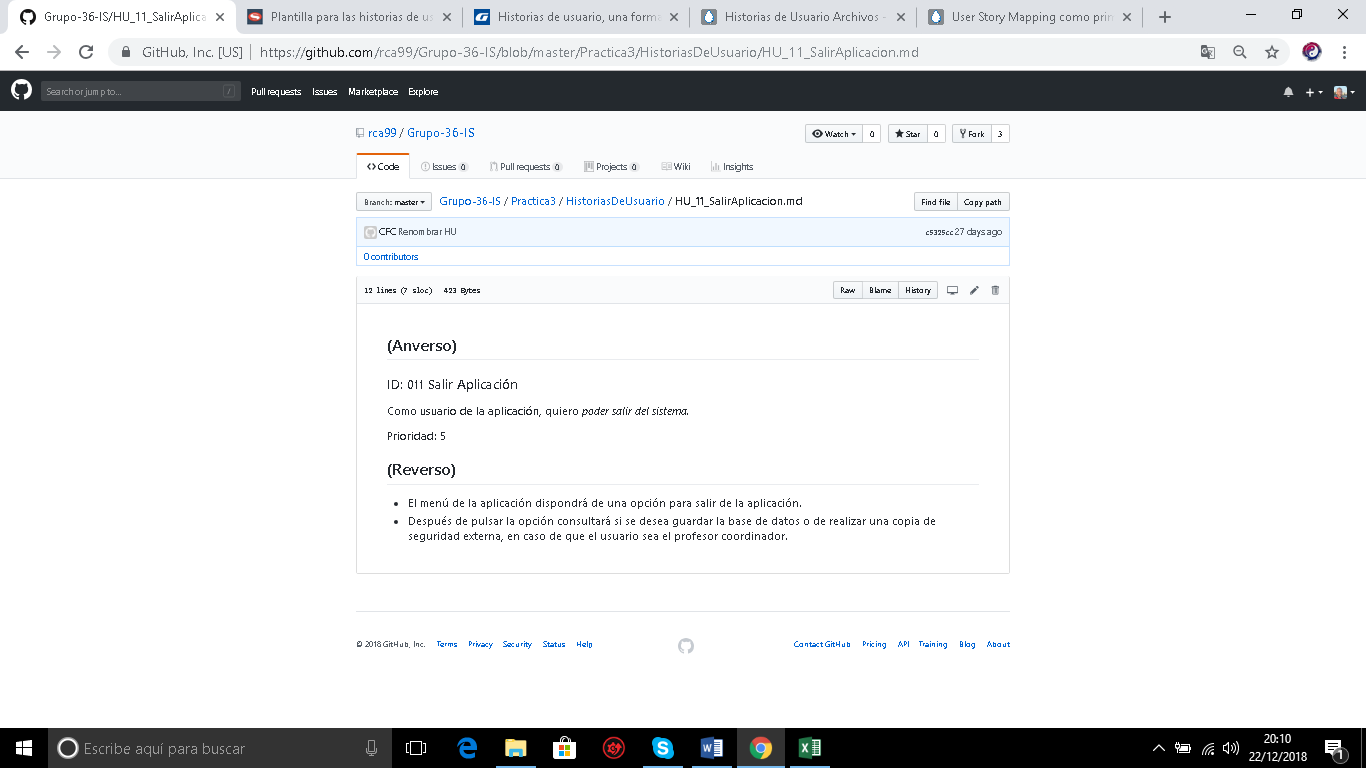
**Ilustración 8 Historia Uso 007 Guardar base de datos**

**Ilustración 9 Historia Uso 008 Cargar base de datos**



**Ilustración 10 Historia Uso 009 Guardar copia de seguridad externa**

**Ilustración 11 Historia Uso 010 Cargar copia de seguridad externa**

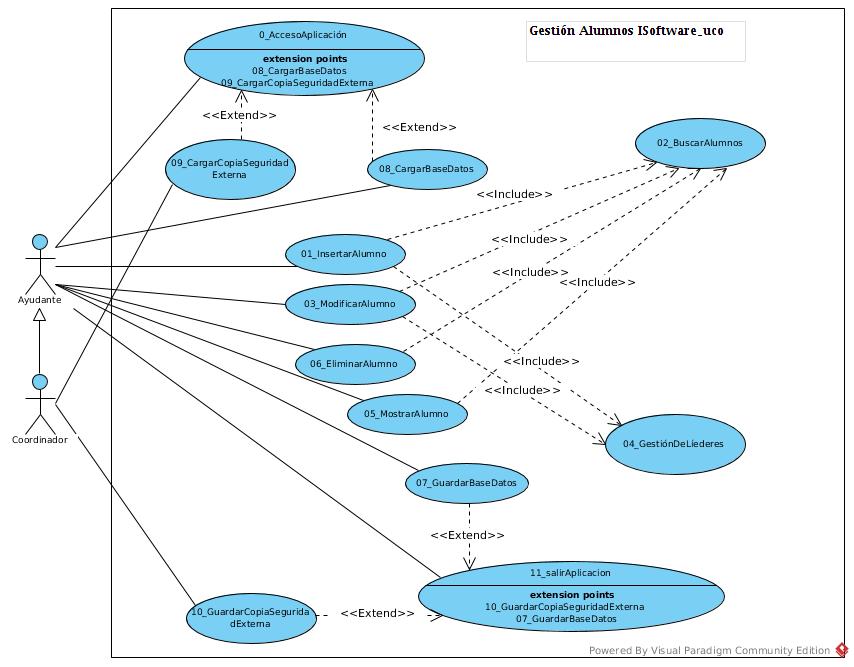


**Ilustración 12 Historia Uso 011 Salir Aplicación**

## Casos de Uso

Aunque desde la metodología Ágil, podríamos partir de las Historias de Usuario para desarrollar nuestro sistema, pasamos a describir los casos de Uso para servirnos de apoyo en la implementación del mismo.

Del análisis del Diagrama de Casos de Uso que se presenta a continuación podemos observar como en nuestro sistema encontramos a los ayudantes y coordinador de la asignatura relacionados mediante una relación de herencia y en el que el coordinador de la asignatura hereda los atributos y funciones propias del ayudante y además extiende sus funcionalidades a las propias definidas en los requisitos de guardar y cargar la base de datos externa. Para cada historia de usuario definimos un caso de uso y en ellos se pueden establecer relaciones de inclusión y extensión entre algunos de ellos. Relaciones de inclusión se dan entre los casos de uso Insertar, Modificar, Eliminar y Mostrar alumno con el caso de uso Buscar Alumno y también entre Insertar alumno y Modificar alumno con el caso de uso Gestión de Líderes. Las relaciones de extensión se dan en el caso de uso de Acceso a la aplicación en el que opcionalmente se puede bien cargar la base de datos o cargar la base de datos externa y en el caso de uso de salir de la aplicación el que opcionalmente se puede bien guardar la base de datos o la guardar la base de datos externa.



**Ilustración 12 Diagrama de Casos de Uso**

Se pasa a continuación a definir cada uno de los Casos de Uso a desarrollar en el sistema.

Acceso a la aplicación **ID: 00**

**Breve descripción: Los profesores podrán acceder a la aplicación mediante sus credenciales de la Universidad.**

**Actores principales: Profesor coordinador y profesores ayudantes**

**Actores secundarios: -**

### Precondiciones

* Se dispone de acceso a las credenciales de los profesores (usuario, contraseña, asignaturas que cursan y papel que desempeñan en la misma, coordinador o ayudante).

### Flujo principal

1 El caso de uso comienza cuando el usuario necesita acceder a la aplicación.

2 El profesor introducirá su usuario y contraseña habitual de moodle.

3 El sistema comprobará si existe el usuario y que la contraseña introducida sea correcta.

4 Opcionalmente, el sistema consultará si se desea cargar la base de datos de alumnos en caso de existir.

4.1 Además, si es coordinador también dará la posibilidad de cargar la copia de seguridad externa en caso de existir.

### Postcondiciones

A Se accede al menú principal de la aplicación.

B Muestra un mensaje de bienvenida al profesor indicando su nombre por pantalla.

C En caso de cargar la base de datos o copia de seguridad muestra el número de alumnos actuales que se hayan registrados.

## Flujos alternativos

3.1 Si no se introducen las credenciales correctas el sistema informa al usuario y da tres oportunidades de acceso. En caso de no introducir las credenciales correctas en estas tres ocasiones, se sale de la aplicación.

4.1 En caso de no haber sido posible cargar la base de datos mostrará un mensaje informando de la causa.

4.1.1 No existe base de datos o copia de seguridad que cargar.

4.1.2 Ha habido un error a la hora de procesar el archivo de base de datos o copia de seguridad.

Insertar Alumno **ID: 1**

**Breve descripción: El profesor podrá introducir los datos de un alumno.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* El alumno a insertar no puede existir en el sistema.
* El número de alumnos del sistema será menor a 150.

### Flujo principal

1 El caso de uso comienza cuando el usuario necesita introducir un alumno.

2 El sistema comprueba el número de alumnos que existen en el sistema.

3 El sistema recoge los datos de dni, nombre y apellidos para comprobar si el alumno existe en el sistema

4 Se comprobará que el alumno no existe ya en el sistema.

4.1 Si el alumno no existe en el sistema se introducen los datos obligatorios y aquellos opcionales que se desean indicar.

5 El sistema comprueba si el grupo que ha elegido el alumno tiene ya lider.

5.1 Si el grupo que ha elegido el alumno no tiene líder permite asignar este alumno como líder del grupo

6 El sistema introduce el alumno.

7 El sistema comprueba la inserción del alumno en el sistema.

### Postcondiciones

**A** El sistema se actualiza con la introducción del alumno.

**B** El sistema muestra un mensaje indicando que la inserción ha sido correcta mostrando el nombre y dni del alumno introducido.

## Flujos alternativos

2.1 Si en el sistema hay 150 alumnos dados de alta, se muestra un error indicando que no se puede introducir el alumno.

4.1 Si no introduce los datos obligatorios muestra un mensaje de error, indicando los datos que faltan por introducir.

4.2 Por defecto, el campo líder valdrá NO.

4.3 Si el usuario introduce que un alumno es líder entonces deberá indicar obligatoriamente el grupo al que pertenece.

5.1 El sistema no permite la inserción de un alumno que ya existe en el sistema y muestra un mensaje de error.

6.1 El sistema no permite indicar que un alumno es líder de un grupo que ya tenga líder.

7.1 Si el sistema no ha podido realizar la inserción del alumno indica que esta no ha podido ser realizada y que vuelva a intentarlo.

Buscar Alumno **ID: 2**

**Breve descripción: El profesor podrá buscar los alumnos introducidos en la base de datos por DNI, apellidos o grupo.**

**Actores principales: Profesor**

**Actores segundarios: Alumno**

### Precondiciones

* La aplicación debe tener alumnos dados de alta.

### Flujo principal

1 El caso de uso comienza cuando se necesita buscar un alumno.

2 El sistema comprueba si hay alumnos dados de alta en el sistema.

3 El sistema recoge los datos necesarios (DNI, apellidos o grupo) para buscar.

4 El sistema busca el alumno por DNI, apellidos o grupo.

### Postcondiciones

**A** El sistema devuelve el alumno o alumnos encontrados.

**B** En caso de devolver los alumnos del grupo, el sistema indicará si el grupo tiene líder o no lo tiene.

## Flujos alternativos

2.1 Si no hay alumnos en la base de datos, el sistema consultará al profesor si ha cargado la base de datos.

2.1.1 Si ha cargado la base de datos, indicará que no hay alumnos y que proceda a insertar alumnos antes de buscarlos.

2.1.2 Si no ha cargado la base de datos, se le indica que la cargue.

3.1 Se comprueba que los datos introducidos son del tipo y formato correcto, si no lo son se indica que se introduzcan adecuadamente.

4.1 En caso de no existir el alumno muestra un mensaje indicando que el alumno no existe en la base de datos y se regresa al paso 1.

4.2 En caso de que el grupo no tenga alumnos asociados, se muestra un mensaje indicando de que el grupo está vacío.

4.3 En caso de que exista más de un alumno con el mismo apellido, el sistema indica al profesor que introduzca el DNI para filtrar.

Modificar Alumno **ID: 3**

**Breve descripción: El profesor podrá modificar los datos de un alumno.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* El alumno a modificar debe estar almacenado en el sistema.

### Flujo principal

1 El caso de uso da comienzo cuando el profesor necesita modificar un alumno.

2 El sistema pregunta si se desea buscar por DNI o por apellido.

3 El sistema busca al alumno en cuestión por DNI o por apellido.

4 El sistema muestra los datos del alumno, dando la opción de modificar aquellos que se deseen.

4.1 Si se mofica el campo de líder, el sistema comprobará que el grupo al que pertenece el alumno tenga líder o no.

### Postcondiciones

**A** El sistema se actualiza con la nueva información del alumno.

**B** El sistema muestra un mensaje indicando que la modificación ha sido correcta.

## Flujos alternativos

3.1 El sistema no permite la modificación de un alumno que no se encuentra almacenado en la aplicación.

4.1 Si se desean dejar en blanco alguno de los campos obligatorios, el sistema no permitirá guardar los cambios volviendo al punto 4 del proceso.

4.1.1 Si se desea modificar el campo líder, indicando que es líder de un grupo que ya lo tiene, muestra un mensaje de error.

Gestión de Líderes **ID: 4**

**Breve descripción: El sistema verifica si un alumno puede ser o no líder del grupo.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* El alumno estará identificado en el sistema con su DNI, nombre, apellidos y grupo al que pertenece.

### Flujo principal

1 El caso de uso comienza cuando el profesor desea indicar si un alumno es líder de un grupo.

2 El sistema recibe el grupo con el que se quiere interactuar.

3 El sistema busca los alumnos que pertenecen a dicho grupo.

4 El sistema comprueba si algún alumno del grupo es líder.

### Postcondiciones

**A** El sistema indica si un grupo tiene líder o no lo tiene.

## Flujos alternativos

Mostrar Alumno **ID: 5**

**Breve descripción: El profesor podrá mostrar los datos de un alumno.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* El alumno a mostrar debe existir en el sistema.

### Flujo principal

1 El caso de uso comienza cuando el usuario necesita mostrar un alumno.

2 El sistema da la opción de mostrar a todos los alumnos dados de alta en el sistema, a uno en particular, o a los pertenecientes a un grupo.

3 El sistema dará la opción de ordenar a los alumnos mostrados por orden alfabético (nombre o apellidos), por su DNI, o por el curso más alto en el que estén matriculados; tanto de forma ascendente como descendente.

4 El sistema muestra los datos del alumno o alumnos según los criterios seleccionados.

### Postcondiciones

**A** El sistema debe mostrar en blanco los datos del alumno no obligatorios que no hayan sido introducidos previamente.

## Flujos alternativos

2.1 Si el sistema no encuentra a un alumno al que se ha pedido mostrar al buscarlo en el sistema, dará un mensaje de error.

3.1 Si hay dos alumnos que posean la misma información en el campo según el que mostrar a los alumnos ordenados (por ejemplo, el curso más alto en el que se esté matriculado), se mostrarán ordenados según otro de los criterios disponibles, bien por apellido o por DNI en última instancia.

Eliminar Alumno **ID: 6**

**Breve descripción: El profesor podrá eliminar un alumno de la base de datos.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* El alumno que será eliminado debe existir en el sistema.

### Flujo principal

1 El caso de uso se iniciará en el momento que el profesor desee eliminar un alumno.

2 El sistema pregunta si se desea eliminar por DNI o por apellido.

3 El sistema busca al alumno en cuestión por DNI o por apellido.

4 Se elimina al alumno del sistema.

### Postcondiciones

**A** El sistema actualiza la base de datos eliminando al alumno.

**B** Se muestra un mensaje de éxito, indicando la correcta eliminación del alumno.

## Flujos alternativos

* 1. El sistema no permite la eliminación de un alumno que no existe en la aplicación.

Guardar Copia de Seguridad **ID: 7**

**Breve descripción: El profesor podrá realizar una copia de seguridad de los alumnos registrados en el sistema.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* La aplicación debe tener alumnos dados de alta.
* Los datos se guardarán en un archivo binario.

### Flujo principal

1 El caso de uso comienza cuando el usuario requiere realizar copia de seguridad.

2 El sistema comprueba si hay alumnos dados de alta.

3 El sistema actualiza los datos de los alumnos en el fichero binario que almacena la base de datos.

4 El sistema comprueba que la base de datos se ha guardado correctamente.

### Postcondiciones

**A** El sistema muestra un mensaje indicando que los datos se han guardado correctamente.

**B** En caso de no existir alumnos el sistema informa que no es necesaria la copia de seguridad.

## Flujos alternativos

2.1 Si no hay alumnos en la base de datos, el sistema consulta al profesor si ha cargado la base de datos.

2.1.1 Si ha cargado la base de datos, indicará que no hay alumnos y que proceda a insertar alumnos antes de guardar la base de datos.

2.1.2 Si no ha cargado la base de datos, se le indica que la cargue.

3.1 Si no existe el fichero binario se procederá a crearlo.

4.1. Si no se insertan correctamente los datos da un mensaje de error, indicando que vuelva a guardar la base de datos.

Cargar Copia de Seguridad **ID: 8**

**Breve descripción: El profesor podrá cargar una copia de seguridad de los datos de los alumnos registrados.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* La copia de seguridad a cargar debe haber sido previamente guardada.

### Flujo principal

1 El caso de uso comienza cuando el usuario necesita cargar una copia de seguridad.

2 El sistema comprueba si existe el fichero de base de datos.

3 El sistema carga la base de datos.

4 El sistema comprueba que la base de datos se ha cargado correctamente.

### Postcondiciones

**A** El sistema muestra un mensaje de confirmación de la carga y de cuál de las copias de seguridad existentes ha sido cargada.

**B** El sistema permite al usuario sobrescribir los datos de esa copia de seguridad.

## Flujos alternativos

2.1 Si no existe fichero de base de datos, el sistema pedirá al usuario que debe guardar antes la base de datos.

4.1 Si el sistema no es capaz de cargar la base de datos, dará un mensaje de error.

Guardar Copia de Seguridad Externa **ID: 09**

**Breve descripción: El Coordinador podrá guardar la copia de seguridad externa en un servidor remoto.**

**Actores principales: Coordinador.**

**Actores segundarios: Alumno.**

### Precondiciones

* Debe existir previamente una base de datos de la cual crear la copia de seguridad externa.
* Debe haber una conexión a internet activa para acceder al repositorio remoto.

### Flujo principal

1 El caso de uso comienza cuando el usuario requiere guardar la copia de seguridad externa

2 El sistema comprueba si existe el fichero de base de datos que guardar.

3 El sistema conecta con el repositorio remoto.

4 El sistema envía el fichero de base de datos al fichero remoto y guarda la copia de seguridad externa.

### Postcondiciones

**A** El sistema muestra un mensaje de confirmación de realización de la copia de seguridad externa.

## Flujos alternativos

2.1 Si no hay fichero de base de datos, el sistema indicará al usuario que cree una copia de seguridad antes de realizar la copia de seguridad externa.

3.1 Si el sistema no es capaz de conectar con el repositorio remoto informa con un mensaje de error.

4.1 Si el sistema no realiza la copia de seguridad externa da un mensaje de error.

Cargar Copia de Seguridad Externa **ID: 10**

**Breve descripción: El Coordinador de la asignatura podrá cargar una copia de seguridad externa de los datos de los alumnos registrados.**

**Actores principales: Coordinador de la asignatura.**

**Actores segundarios: Alumno.**

### Precondiciones

* La copia de seguridad externa a cargar debe haber sido previamente guardada.
* Debe haber una conexión a internet activa para acceder al repositorio remoto.

### Flujo principal

1 El caso de uso comienza cuando el usuario necesita cargar una copia de seguridad externa

2 El sistema carga la copia de seguridad externa.

3 El sistema comprueba que la base de datos se ha cargado correctamente.

### Postcondiciones

**A** El sistema muestra un mensaje de confirmación de la carga y de cuál de las copias de seguridad existentes ha sido cargada.

**B** El sistema permite al usuario sobrescribir los datos de esa copia de seguridad.

## Flujos alternativos

2.1 Si no hay copias de seguridad externas guardadas, el sistema pedirá al usuario que cree una copia de seguridad.

3.1 Si el sistema no es capaz de cargar la copia de seguridad da un mensaje de error, indicando que no ha sido posible cargar esa copia de seguridad.

Salir Aplicación **ID: 11**

**Breve descripción: El profesor podrá salir de la aplicación.**

**Actores principales: Profesor.**

**Actores segundarios: Alumno.**

### Precondiciones

* .

### Flujo principal

1 El caso de uso comienza cuando el usuario necesita salir de la aplicación.

2 El sistema solicita confirmación para salir de la aplicación

3 El sistema consulta si se quiere guardar la base de datos y/o de realizar la copia de seguridad externa en caso de ser el Coordinador.

3 La aplicación se cierra.

### Postcondiciones

**A** El sistema muestra un mensaje de salida al profesor y se cierra la aplicación.

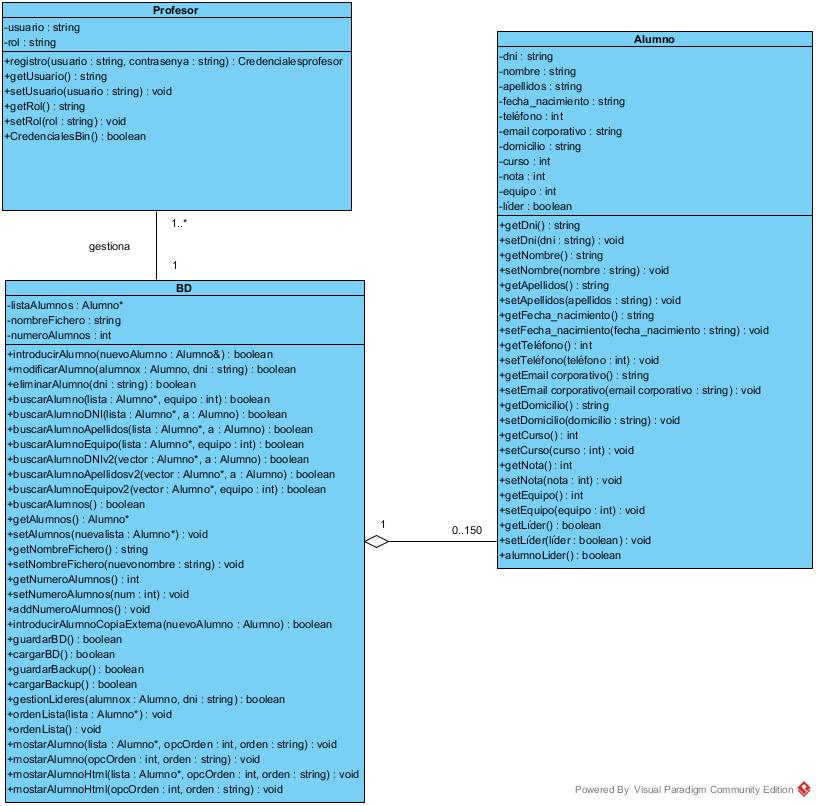
## Flujos alternativos

2.1 Si se indica que no se quiere salir de la aplicación vuelve al menú principal de la aplicación.

3.1 Si no es posible cerrar la aplicación en ese momento se muestra un mensaje de error.

# Diagrama de Clases

Habiendo definido las historias de usuario y los casos de uso, comenzamos a definir las clases de las que se va a componer nuestra aplicación, que serán aquellas en las que se sustenten posteriormente nuestro desarrollo del software de la aplicación, avanzando de esta manera en la preparación para el posterior uso de la metodología Scrum. Esto queda representado en el diagrama de clases que se expone a continuación.



Del análisis del Diagrama de Clases podemos observar que existen tres clases que son el profesor, los alumnos y la base de datos, que se relacionan entre sí, de forma que uno o varios profesores gestionan la base de datos, mientras que la clase alumno es una agregación de la clase BD, habiendo una sola base de datos que gestiona información de uno hasta 150 alumnos.

En cada clase se definen unos datos y unas operaciones propias, a destacar entre ellas:

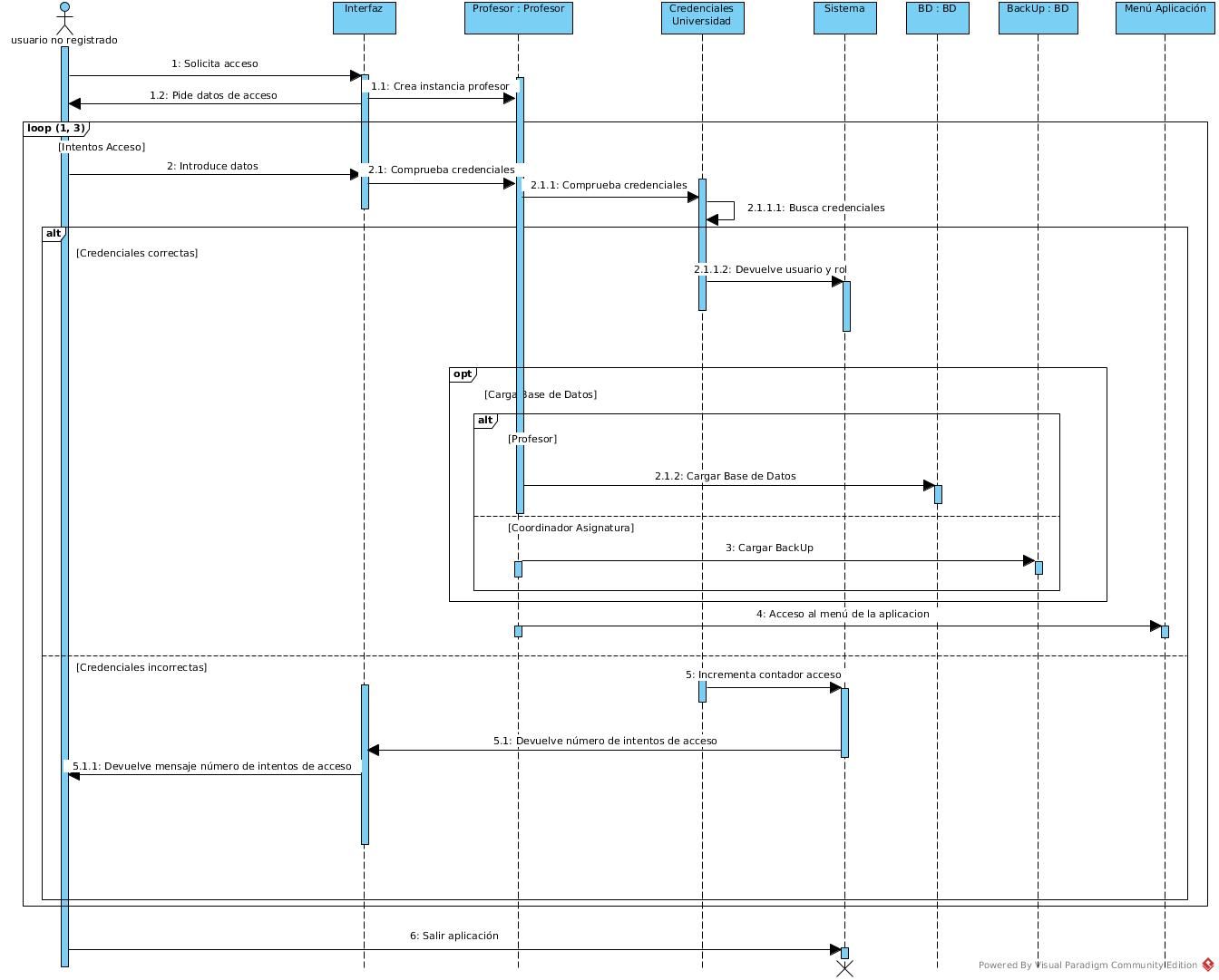
* En la clase profesor se define el usuario que entra a la aplicación y su rol, y se establecen sus credenciales.
* En la clase BD, se incluyen gran parte de los requisitos funcionales que hemos establecido, así como otras funciones que nos sean de ayuda a la hora de desarrollar el software, como pueden ser las distintas funciones de buscar alumno, según la opción de búsqueda, o el uso de lista o vector, o las distintas funciones para mostrar los alumnos, incluyendo el uso de html, entre otras.
* En la clase alumno se definen los datos que debe almacenar la base de datos sobre estos, con los observadores y modificadores oportunos.

# Diagramas de Secuencia

Una vez realizado el Diagrama de Clases, se procede a la elaboración de los Diagramas de Secuencia, diagramas funcionales que muestran al cliente la mecánica de la interacción para cada Caso de Uso y que permiten modelar la interacción entre objetos (clases) en un sistema (AppGestionAlumnos).

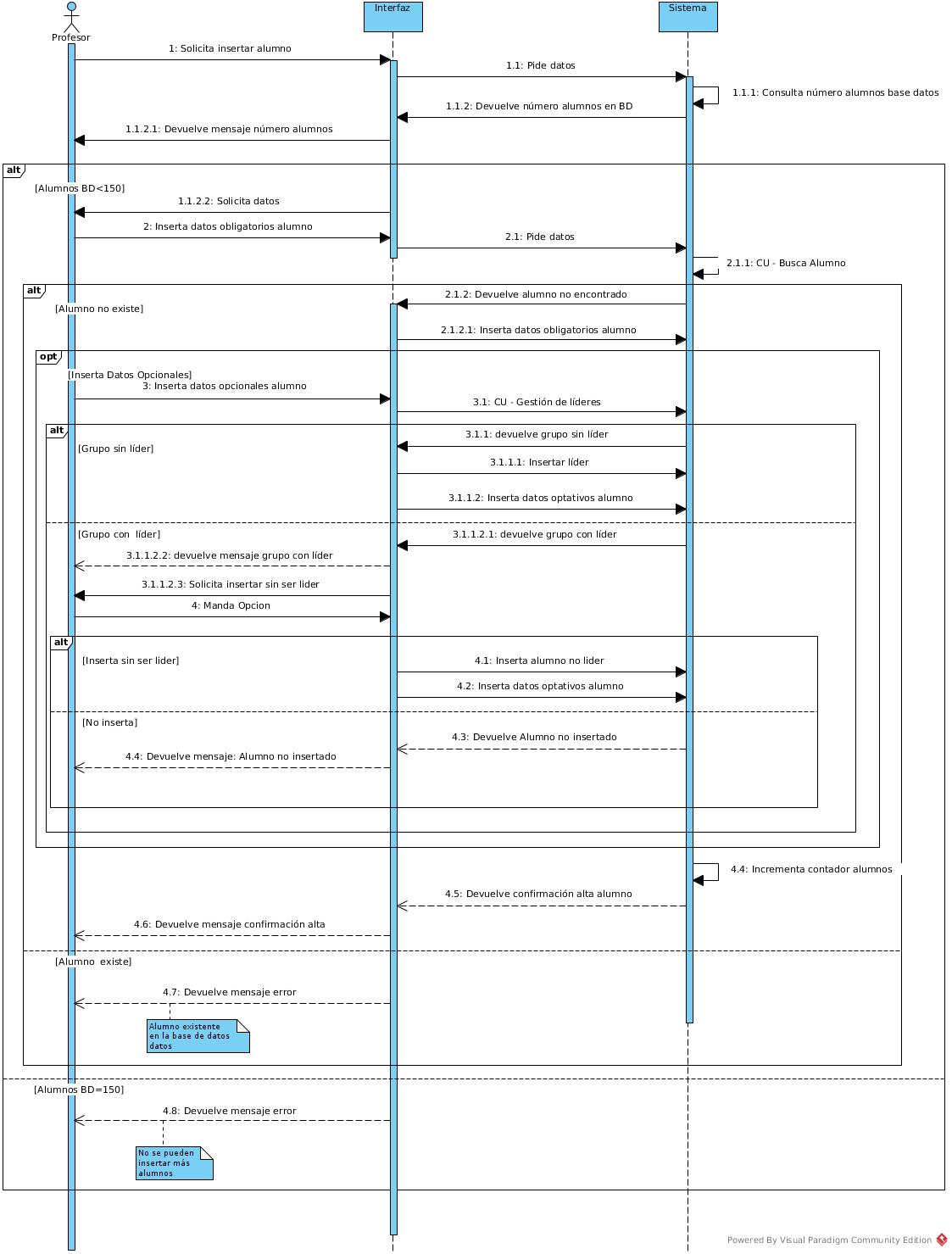
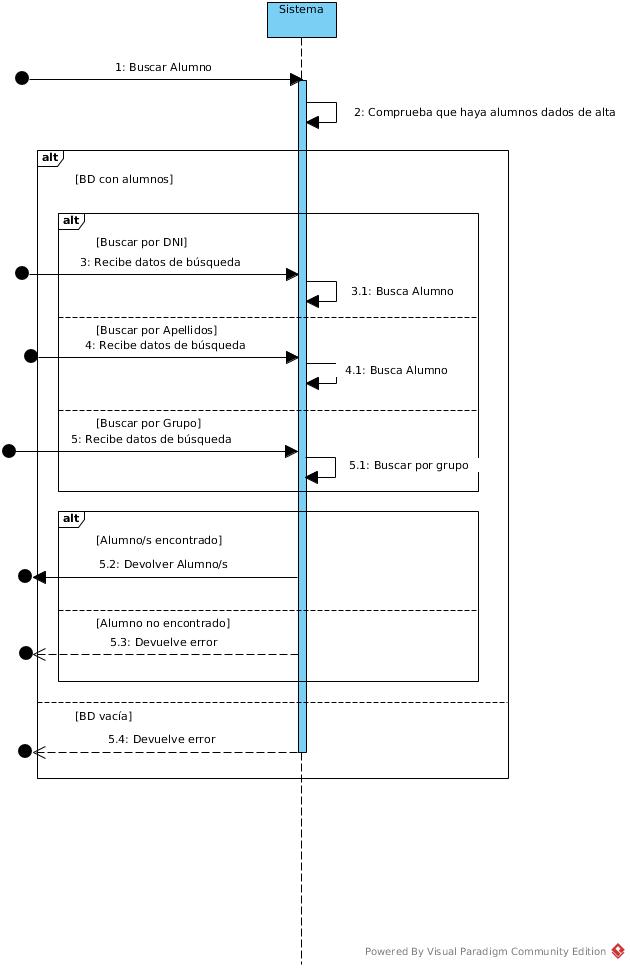
Dichos objetos se representan con un encabezado rectangular que muestra el nombre del objeto en su interior (puede incluir también el nombre de la clase). Bajo el rectángulo, una línea vertical y discontinua, llamada *línea de vida*, indica la presencia del objeto. Cuando el objeto entra en acción, es decir, cuando realiza una tarea, la línea de vida entra en *activación*, el tiempo en que el objeto realiza la tarea. Finalmente, estos objetos que van activándose y realizando sus respectivas tareas, se comunican mediante *mensajes* los cuales pueden servir para indicar a un objeto si tiene que realizar una tarea o que su papel en el diagrama ha terminado.

Tras explicar qué son los Diagramas de Secuencia, se procede ahora mostrar los realizados para el sistema de gestión de alumnos.

* **SD 00: Acceso a la Aplicación**

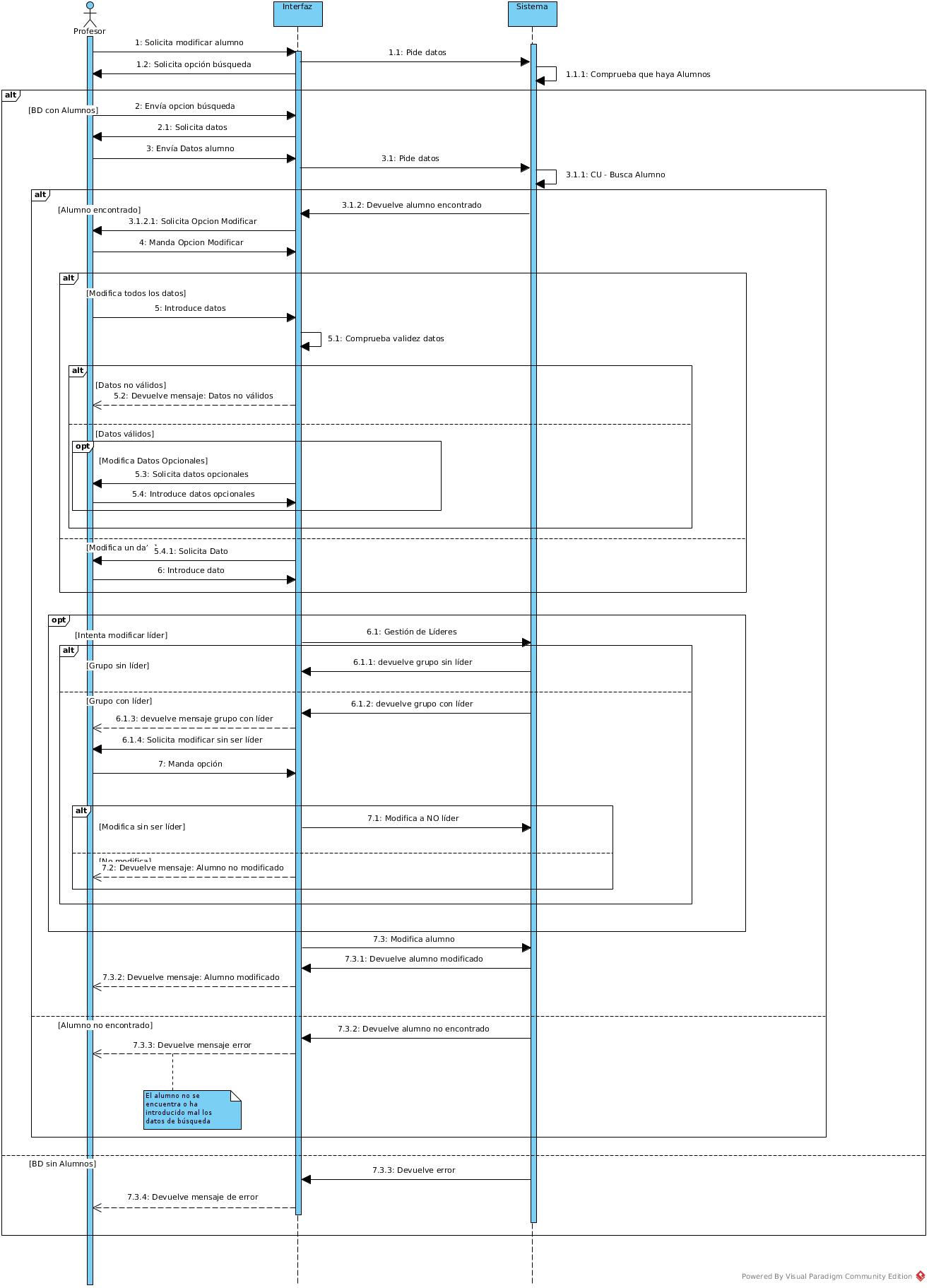
Determina los pasos a seguir para entrar correctamente a la aplicación: El profesor ayudante o coordinador deberá verificarse con las credenciales proporcionadas por la Universidad con hasta 3 intentos de acceso.

Además, en caso de existir una copia de los datos guardada, dependiendo si el profesor es coordinador o ayudante, se permitirá cargar la copia de seguridad externa o local, respectivamente.

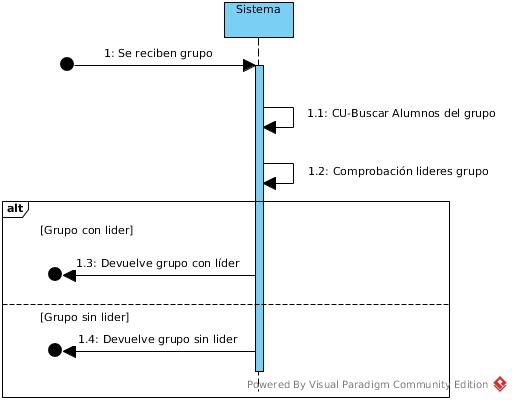
* **SD 01: Insertar Alumno**
* **SD 02: Buscar Alumno**

Este diagrama de secuencia se encuentra entre los más importantes del proyecto, pues para realizar cualquier operación de introducción, modificado o borrado se necesita de él.

* **SD 03: Modificar Alumno**

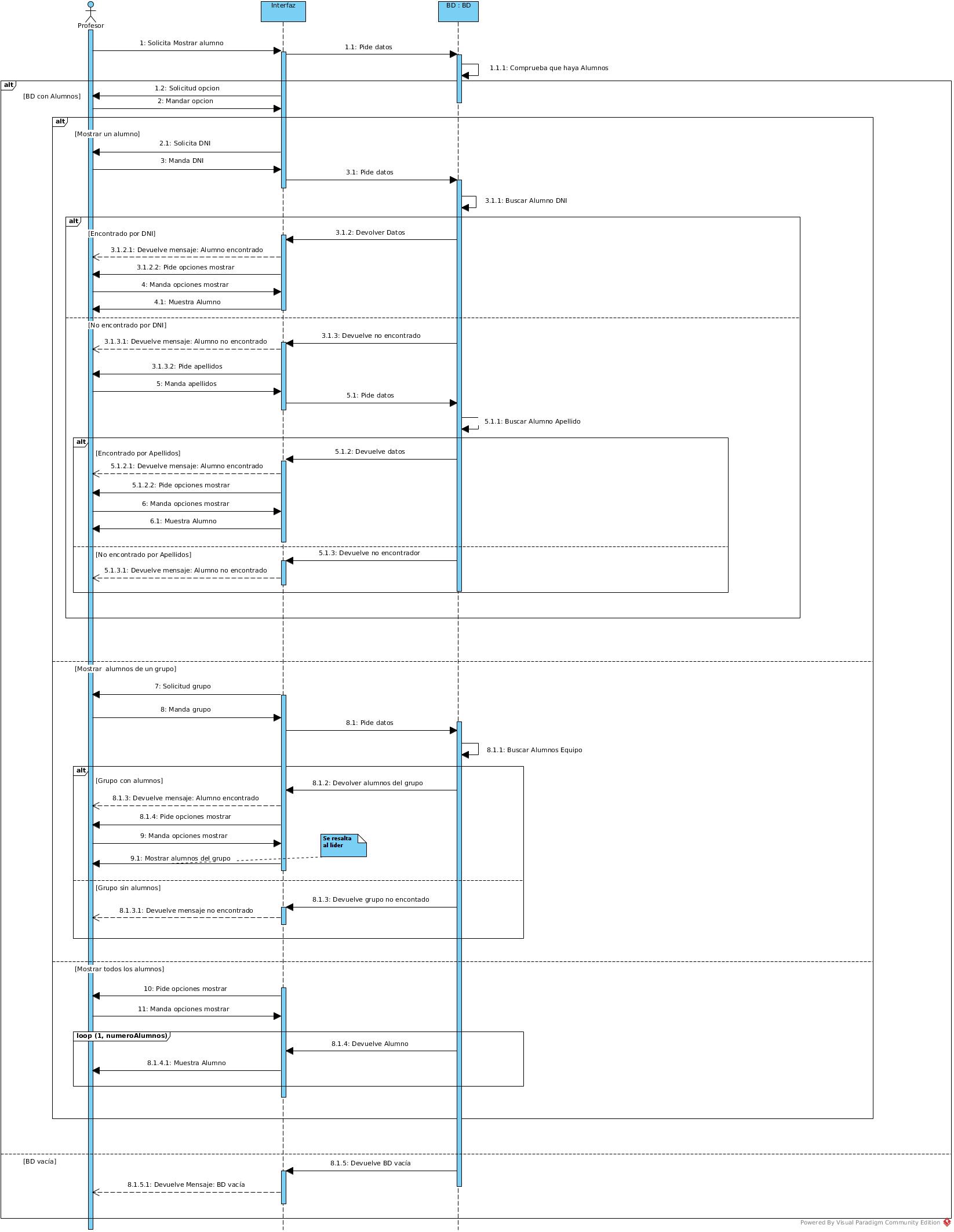


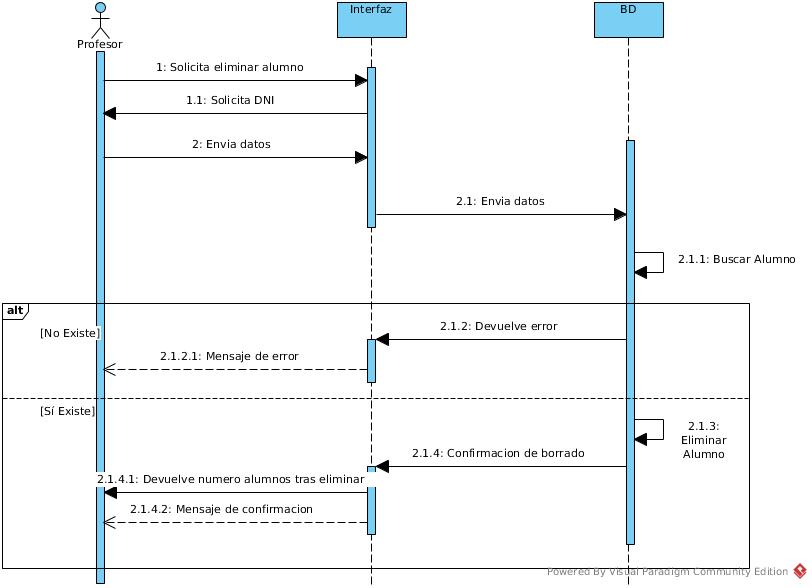
* **SD 04: Gestión de Líderes**



Al igual que Buscar Alumno, (aunque no tanto) este diagrama tiene una gran importancia, pues permite que se cumpla el requisito impuesto por el cliente: debe haber 1 único líder por grupo.

Verifica que ninguno de los alumnos del grupo sea ya líder, si no hay líder, permite introducir un alumno para este grupo como líder, o bien se puede modificar otro alumno de otro grupo para que sea líder de este. Si ya hay un líder en dicho grupo de deniegan estas operaciones. Permitiendo introducir o modificar, pero sin ser líder.

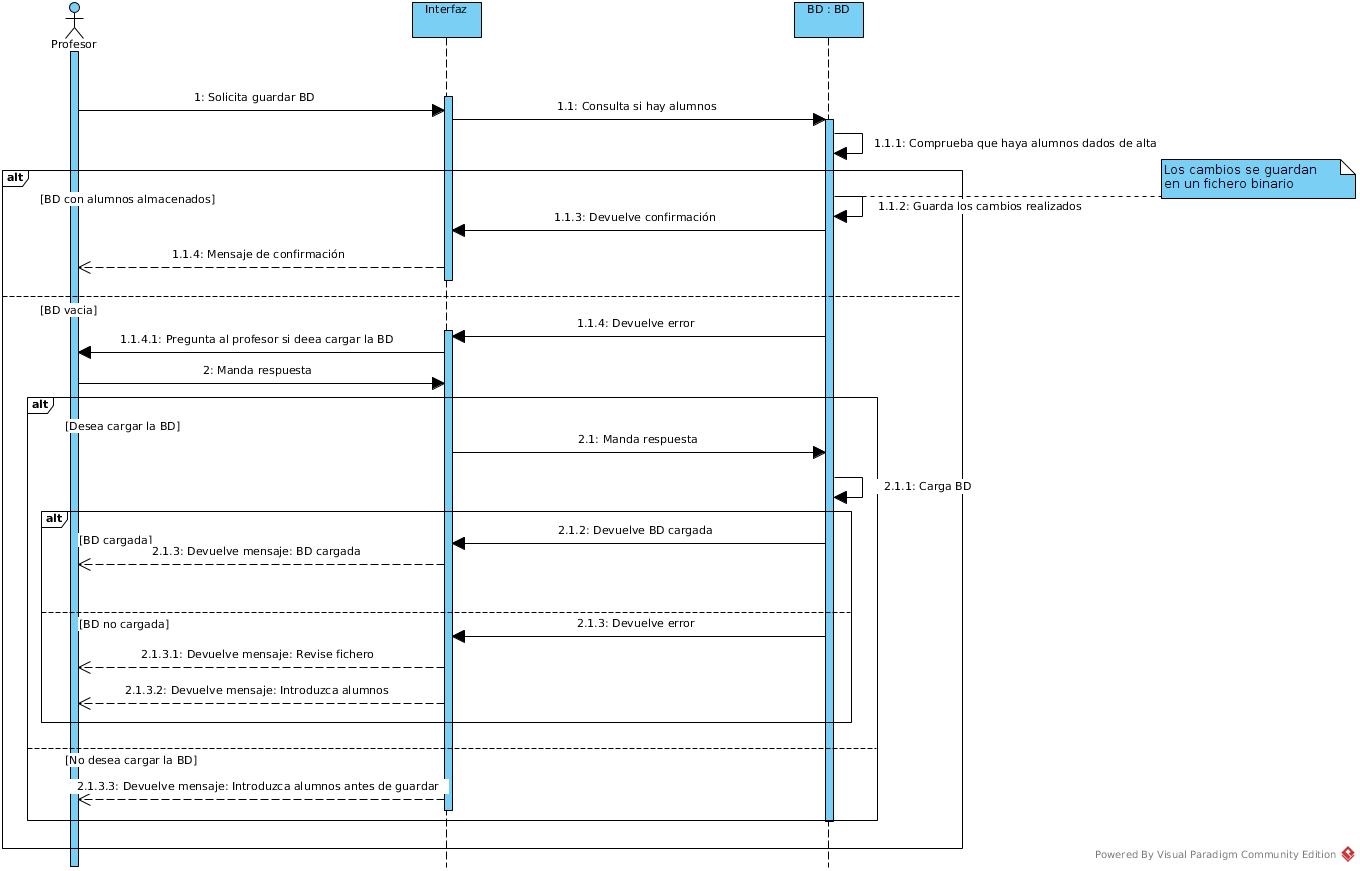
* **SD 05: Mostrar Alumnos**
* **SD 06: Eliminar Alumno**



La función que se lleva a cabo en este diagrama es sencilla. Para eliminar a un alumno, se pide su DNI al profesor y procede a buscar en los datos del sistema mediante DNI.

Si lo encuentra, lo borra; si no, devuelve un mensaje indicando que el alumno introducido no fue encontrado en la base de datos actual.

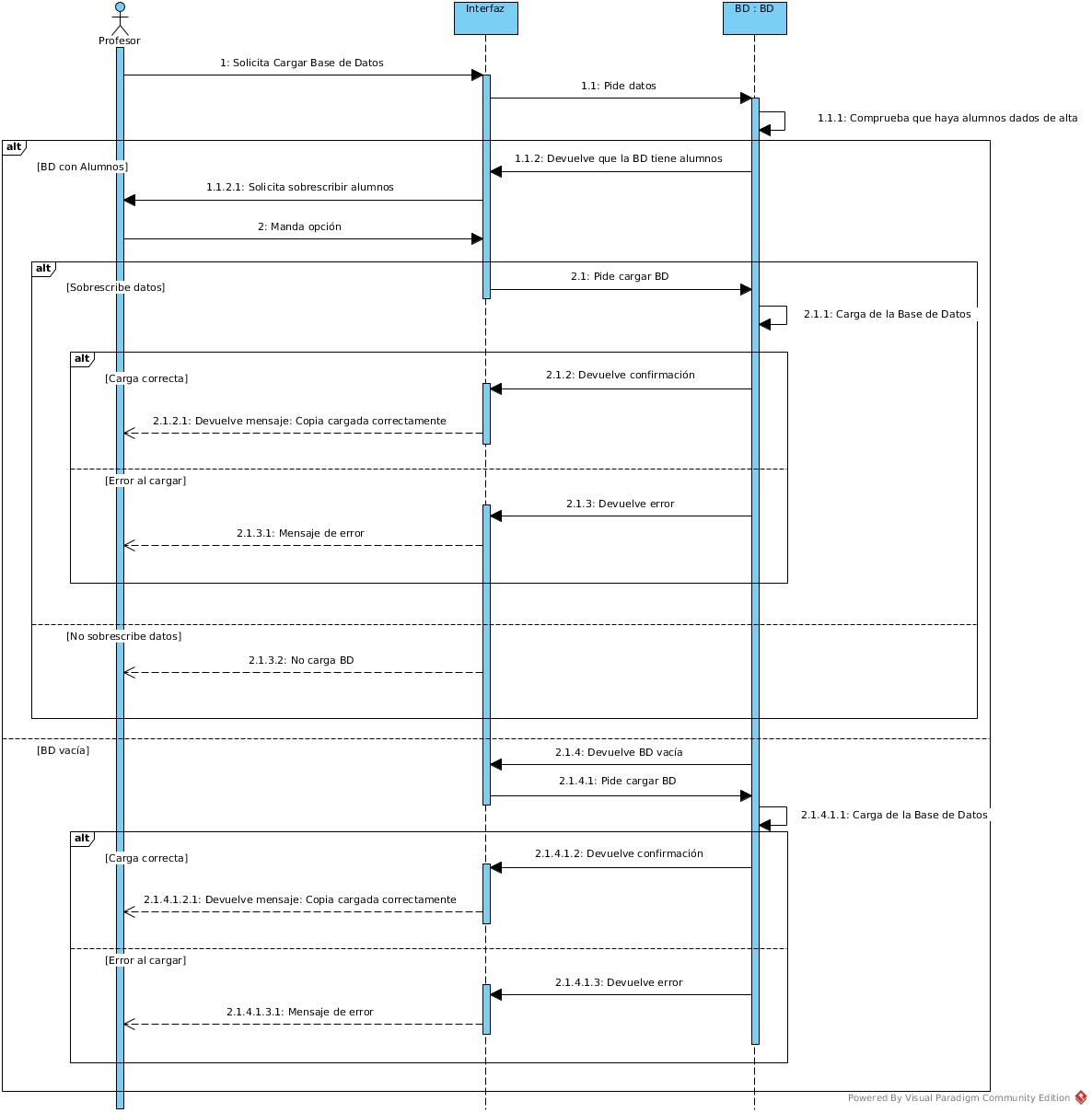
* **SD 07: Guardar Base de Datos**



Permite tanto a profesores como coordinadores guardar los cambios realizados en el ámbito local, es decir, en un fichero binario.

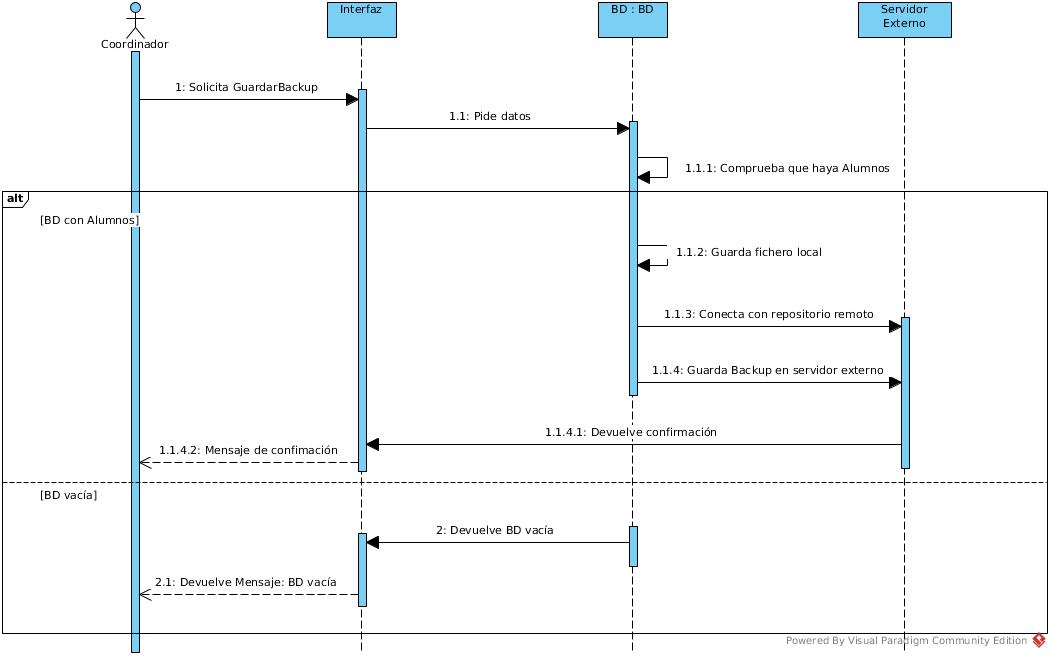
Para ello, antes de guardar nada el sistema se encarga de verificar que la base de datos tiene alumnos almacenado, y, en caso de estar vacía, se pregunta si se desea cargar una copia de la base de datos anterior.

* **SD 08: Cargar Base de Datos**



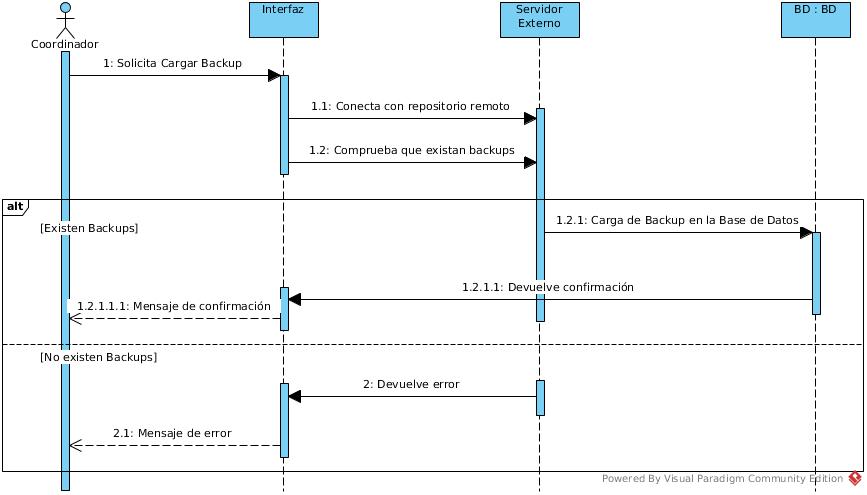
Este diagrama muestra cómo se realiza una carga de la base de datos. Destacar que si la base de datos actual tiene alumnos introducidos, estos se sobrescribirán al cargar la nueva copia.

* **SD 09: Guardar Copia de Seguridad Externa**



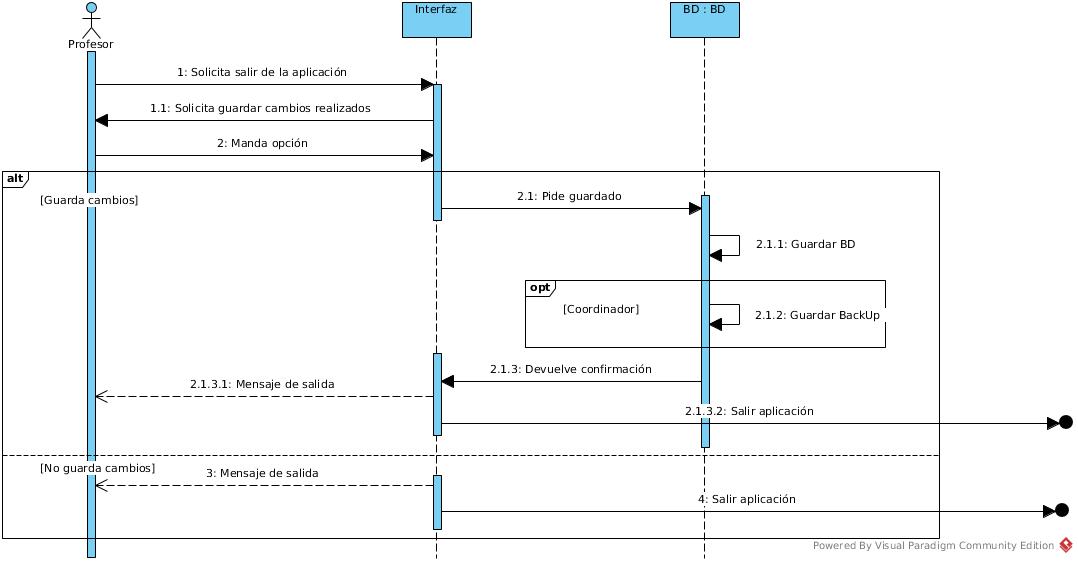
Solo pueden llevarlo a cabo los coordinadores. Realiza una copia de los datos locales en un repositorio del servidor externo GitHub, llevando a cabo previamente una copia local también.

* **SD 10: Cargar Copia de Seguridad Externa**



Solo pueden llevarlo a cabo los coordinadores. Trata de conectar con el repositorio remoto en GitHub y descargar el backup existente en caso de que hubiera uno.

* **SD 11: Salir de la Aplicación**



El usuario desea salir del software de gestión de alumnos. El sistema pregunta al usuario si desea guardar los cambios realizados, y, dependiendo de si el usuario ejerce el rol de ayudante o coordinador; realizará un guardado local o en el servidor remoto, respectivamente.

|  |  |
| --- | --- |
|  |  |
|  | Desarrollo de la aplicación – metodología scrum |
|  |  |

# Introducción

Todo el proceso anterior ha servido de preparación para hacer posible ahora la aplicación de la metodología Scrum, mediante la cual se va a desarrollar la aplicación. Consiste en llevar a cabo una mecánica de trabajo muy estructurada para ir llevando a cabo el desarrollo de manera escalonada y repartida entre los distintos miembros del equipo de desarrollo, dividiendo el trabajo en partes, llamadas sprints, para alcanzar el objetivo final de la forma más eficiente posible y teniendo todo bajo control.

# Product BackLog

Para indicar las funcionalidades que se han de realizar y el número de horas que van a ocupar, se crea el Product BackLog, que sirve como punto de partida de la metodología Scrum, pues sobre este se organiza el resto del trabajo y nos permite ver si la estimación de horas realizada se cumple.

|  |  |  |
| --- | --- | --- |
| Funcionalidad | Prioridad | Horas estimadas |
| Definir Clases | 1 | 1 |
| Definir Menú Aplicación | 1 | 1 |
| HU\_01\_InsertarAlumno | 1 | 1 |
| Interfaz InsertarAlumno | 1 | 1 |
| HU\_02\_BuscarAlumno | 1 | 2 |
| Interfaz BuscarAlumno | 1 | 1 |
| HU\_07\_GuardarBaseDatos | 2 | 2 |
| Interfaz GuardarBaseDatos | 2 | 0,5 |
| HU\_08\_CargarBaseDatos | 2 | 1 |
| Interfaz CargarBaseDatos | 2 | 0,5 |
| HU\_04\_GestionDeLideres | 2 | 1 |
| Interfaz GestionDeLideres | 1 | 0,5 |
| HU\_03\_ModificarAlumno | 3 | 3 |
| Interfaz ModificarAlumno | 3 | 1 |
| HU\_05\_MostrarAlumnos | 3 | 4 |
| Interfaz MostrarAlumnos | 3 | 1 |
| HU\_06\_EliminarAlumno | 3 | 2 |
| Interfaz EliminarAlumno | 3 | 1 |
| HU\_00\_AccesoAplicación | 4 | 3 |
| Interfaz AccesoAplicación | 4 | 1 |
| HU\_09\_GuardarCopiaSeguridadExterna | 4 | 0,25 |
| Interfaz GuardarCopiaSeguridadExterna | 4 | 0,25 |
| HU\_10\_CargarCopiaSeguridadExterna | 4 | 1 |
| Interfaz CargarCopiaSeguridadExterna | 4 | 1 |
| HU\_11\_SalirAplicación | 4 | 1 |
| Interfaz SalirAplicación | 4 | 1 |
| VerificacionRequisitos | 5 | 2 |
| PruebasCasosUso | 5 | 2 |
| DocFinal\_Practica2 | 5 | 2 |
| DocFinal\_Practica3\_DiagramaClases | 5 | 2 |
| DocFinal\_Practica2\_DiagramasSecuencia | 5 | 2 |
| DocumentaciónFinal\_Scrum | 5 | 1 |
| DocumentaciónFinal\_MatricesValidación | 5 | 1 |
| DocumentacionFinal\_Documento | 5 | 2 |

# Sprints Review meeting

El trabajo se ha dividido en tres fases o sprints en los que se ha ido progresando en la realización de la aplicación. En cada sprint, se han fijado unas metas a cumplir, especificadas en sus correspondientes Sprint BackLog.

## Sprints BackLogs

En cada Sprint BackLog se indica la tarea que se le asigna a cada miembro, junto con su prioridad y horas de trabajo estimadas. Este trabajo será posteriormente comentado en las pequeñas reuniones que se realizan durante cada sprint. A continuación, aparecen los tres Sprints BackLogs mencionados anteriormente.

### Sprint BackLog 1

El miembro del equipo **Rafael Cantero Alén**, como **Scrum Master**, se encargará de realizar las siguientes tareas:

* Definir Clases|1|1
* HU\_02\_BuscarAlumno|1|2
* Interfaz BuscarAlumno|1|1

El miembro del equipo **Carlos Freire Caballero**, como integrante del *equipo de desarrollo*, se ocupará de:

* Definir Menú Aplicación|1|1
* HU\_01\_InsertarAlumno|1|1
* Interfaz InsertarAlumno|1|1
* HU\_04\_GestionDeLideres|2|1
* Interfaz GestionDeLideres|1|0,5

El integrante del equipo **Fernando Herrera Poch**, como parte del *equipo de desarrollo*, se hará cargo de:

* HU\_07\_GuardarBaseDatos|2|2
* Interfaz GuardarBaseDatos|2|0,5
* HU\_08\_CargarBaseDatos|2|1
* Interfaz CargarBaseDatos|2|0,5

### Sprint BackLog 2

Notas:

Quedan por testear la funcionalidad de Insertar Alumno y GestiondeLideres Falta por implementar la funcionalidad BuscarAlumno

El miembro del equipo **Rafael Cantero Alén**, como **Scrum Master**, se encargará de realizar las siguientes tareas:

* HU\_02\_BuscarAlumno|1|1
* Interfaz BuscarAlumno|1|0,5
* HU\_06\_EliminarAlumno|3|2
* Interfaz EliminarAlumno|3|1
* HU\_11\_SalirAplicación|4|0,25
* Interfaz SalirAplicación|4|0,25

El miembro del equipo **Carlos Freire Caballero**, como integrante del *equipo de desarrollo*, se ocupará de:

* Interfaz InsertarAlumno|1|1
* HU\_03\_ModificarAlumno|3|3
* Interfaz ModificarAlumno|3|1
* HU\_00\_AccesoAplicación|4|3
* Interfaz AccesoAplicación|4|1

El integrante del equipo **Fernando Herrera Poch**, como parte del *equipo de desarrollo*, se hará cargo de:

* HU\_05\_MostrarAlumnos|3|4
* Interfaz MostrarAlumnos|3|1
* HU\_09\_GuardarCopiaSeguridadExterna|4|0,25
* Interfaz GuardarCopiaSeguridadExterna|4|0,25
* HU\_10\_CargarCopiaSeguridadExterna|4|1
* Interfaz CargarCopiaSeguridadExterna|4|1

### Sprint BackLog 3

En este último sprint se realizará la verificación de los requisitos incluyendo pruebas específicas de los casos de uso, así como la preparación de la documentación a entregar junto con la versión final del código.

El miembro del equipo **Rafael Cantero Alén**, como **Scrum Master**, se encargará de realizar las siguientes tareas:

* DocumentaciónFinal\_Practica3\_DiagramaClases|5|2
* DocumentaciónFinal\_Scrum|5|1

El miembro del equipo **Carlos Freire Caballero**, como integrante del *equipo de desarrollo*, se ocupará de:

* VerificaciónRequisitos|5|2
* PruebasCasosUso|5|2
* DocumentaciónFinal\_Practica3\_DiagramasSecuencia|5|2
* DocumentaciónFinal\_MatricesValidación|5|1

El integrante del equipo **Fernando Herrera Poch**, como parte del *equipo de desarrollo*, se hará cargo de:

* DocumentaciónFinal\_Practica2|5|2
* DocumentacionFinal\_Documento|5|2

## Sprints Meetings

Durante cada uno de los sprints, se llevan a cabo una serie de reuniones entre los miembros del equipo de desarrollo para comentar como progresa cada uno en su trabajo, poner en común impresiones sobre la dinámica de trabajo y expresar algunos de los problemas a los que se han tenido que enfrentar durante la realización de las funcionalidades asignadas para ese sprint, y eso se refleja posteriormente en un documento que recoge aquellos aspectos que se han tratado en la reunión.

### Sprint 1 Meeting 1

**Fecha reunión:04/12/2018 23:00**

Descripción del trabajo

**Rafael Cantero Alén**

Estimadas: 4 h Ejecutadas: 1 h

Notas trabajo:

Está acabando el diseño de clases.

**Carlos Freire Caballero**

Estimadas: 4,5 h Ejecutadas: 2 h

Notas trabajo:

Tenemos dudas a la hora de implementar la función de gestión de líderes.

Quedamos en que vamos a pensar como realizarla entre todos y también en consultarle al profesor cómo se podría llevar a cabo.

**Fernando Herrera Poch**

Estimadas: 5 h Ejecutadas: 3 h

Notas trabajo:

Se ha realizado la funcionalidad GuardarBD y CargarBD sin haberla podido finalizar por completo, ya que parte de la misma requería del desarrollo de la interfaz con el usuario.

Se ha verificado tanto la escritura como la escritura del fichero binario.

Para el desarrollo de la funcionalidad, se han creado las clases necesarias con su estructura mínima.

Problemática encontrada

Es necesario desarrollar el menú de la aplicación para poder llevar a cabo las funcionalidades completas según lo descrito en los diagramas de secuencia. Se consultará con el profesor de la asignatura.

setLider debe devolver un tipo booleano. En el Diagrama de Clases está como void.

No se ha subido a git porque tenemos dudas en cómo trabajar por ramos.

Antes de empezar con el trabajo quizás se hubiera tenido que esperar a tener las clases definidas en sus correspondientes ficheros de cabecera.

### Sprint 1 Meeting 2

**Fecha reunión:07/12/2018 23:00**

Descripción del trabajo

**Rafael Cantero Alén**

Estimadas: 4 h Ejecutadas: 2 h

Notas trabajo:

Ha finalizado la definición de las clases y está desarrollando la funcionalidad de buscarAlumno()

**Carlos Freire Caballero**

Estimadas: 4 h Ejecutadas: 4 h

Notas trabajo:

Ha ejecutado el menú base de la aplicación para que todos vayamos implementando nuestras funcionalidades.

**Fernando Herrera Poch**

Estimadas: 5 h Ejecutadas: 5 h

Notas trabajo:

Ha desarrollado la implementación en el menú de guardarBD. En la comprobación de las funciones guardarBD y cargarBD ha encontrado fallos que hay que corregir.

Problemática encontrada

Nos está costando coordinar el trabajo ya que a veces, es necesario realizar una modificación en una clase y no sabemos si la debe hacer el responsable de las clases o el que detecta la necesidad de modificarlas. Decidimos que estas modificaciones se hagan en la rama local y que se comuniquen para hacerlas en la rama master.

Ha detectado fallos en las funciones cargarBD y guardarBD ya que no lee el fichero binario.

Para la funcionalidad buscarAlumno() se plantean dudas para su implantación y se dan ideas en grupo.

### Sprint 1 Meeting 3

**Fecha reunión:09/12/2018 15:00**

*Descripción del trabajo*

**Rafael Cantero Alén**

Estimadas: 4 h Ejecutadas: 5 h

Notas trabajo:

Ha tenido problemas para desarrollar la funcionalidad buscarAlumno() y no se ha podido desarrollar por completo

**Carlos Freire Caballero**

Estimadas: 4 h Ejecutadas: 5 h

Notas trabajo:

Ha implementado las funcionalidades que tenía que desarrollar, pero ha pensado que se podría mejorar la inserción del alumno usando sólo métodos set y no creando una estructura auxiliar tal y como está ahora.

**Fernando Herrera Poch**

Estimadas: 5 h Ejecutadas: 7 h

Notas trabajo:

No ha podido finalizar la implementación de la funcionalidad cargarBD().

*Problemática encontrada*

En general hemos encontrado los siguientes problemas:

* Coordinación trabajo en grupo.
* Mejorar en el flujo de trabajo en ramas.
* Implementación de funcionalidades, buscarAlumno() y cargarBD().

Propuestas para el siguiente Sprint:

* Finalizar las funcionalidades que faltan. En cargarBD() se consultará con el profesor de la asignatura.
* Mejorar la funcionalidad de insertar alumno con métodos set.
* Hacer una primera implementación sencilla tan pronto sea posible para evaluar la dificultad de cada funcionalidad y así poder tomar las medidas necesarias para desarrollar las funcionalidades en el tiempo establecido.
* Actualizar las horas previstas de desarrollo para aproximarlas a la realidad.

### Sprint 2 Meeting 1

**Fecha reunión:12/12/2018 23:00**

*Descripción del trabajo*

**Rafael Cantero Alén**

Estimadas: 5 h Ejecutadas: 1 h

Notas trabajo:

Se han realizado parcialmente:

* HU\_02\_BuscarAlumno|1|1
* Interfaz BuscarAlumno|1|0,5

Se han realizado con éxito las funciones buscar por dni y buscar por apellidos.

*Problemática encontrada*

Las dificultades en el planteamiento del desarrollo del código han provocado el retraso de esta funcionalidad.

**Carlos Freire Caballero**

Estimadas: 9 h Ejecutadas: 3 h

Notas trabajo:

Se ha realizado:

* Interfaz InsertarAlumno|1|1
* HU\_00\_AccesoAplicacion|4|1
* Interfaz AccesoAplicacion|4|1

Se ha realizado la interfaz de introducir alumno, para así poder disponer de su uso para el resto de funcionalidades.

También se ha llevado a cabo la funcionalidad de acceso a la aplicación, a partir de los credenciales proporcionados por la Universidad, tanto para los profesores ayudantes como para el coordinador, así como su implicación en la interfaz en menuprincipal.cc

*Problematica Encontrada*

Para comprobar la existencia de un alumno necesitaba de la funcion buscar.

**Fernando Herrera Poch**

Estimadas: 7,5 h Ejecutadas: 3 h

Notas trabajo:

Se han ejecutado parcialmente

* HU\_05\_MostrarAlumnos|3|4
* Interfaz MostrarAlumnos|3|1

Se han estado probando unos scripts para realizar las copias de seguridad externas y cargarlas. Tiene buena pinta.

*Problemática encontrada*

Del método sort() de list ha costado definir el operador <, después de esto ha sido bastante fácil

### Sprint 2 Meeting 2

**Fecha reunión:14/12/2018 15:00**

*Descripción del trabajo*

**Rafael Cantero Alén**

Estimadas: 5 h Ejecutadas: 3 h

Notas trabajo:

Se han finalizado las funcionalidades siguientes:

* HU\_02\_BuscarAlumno|1|1
* Interfaz BuscarAlumno|1|0,5
* HU\_11\_SalirAplicación|4|0,25
* Interfaz SalirAplicación|4|0,25

Cuando se acostumbra al trabajo en ramas personales se ve favorecido el trabajo en equipo.

*Problemática encontrada*

**Carlos Freire Caballero**

Estimadas: 9 h Ejecutadas: 8 h

Notas trabajo:

* HU\_03\_Modificar Alumno|3|3
* Interfaz InsertarAlumno|1|1

*Problematica encontrada*

Demasiada demora en la interfaz de insertar alumno, se han empleado un mayor numero de horas del estimado.

**Fernando Herrera Poch**

Estimadas: 7,5 h Ejecutadas: 12 h

Notas trabajo:

Están finalizadas las funcionalidades encargadas

* HU\_05\_MostrarAlumnos|3|4
* Interfaz MostrarAlumnos|3|1
* HU\_09\_GuardarCopiaSeguridadExterna|4|0,25
* Interfaz GuardarCopiaSeguridadExterna|4|0,25
* HU\_10\_CargarCopiaSeguridadExterna|4|1
* Interfaz CargarCopiaSeguridadExterna|4|1

En relación al flujo de trabajo ha sido más que óptimo. El trabajo en las ramas personales facilita mucho y crea menos confusión. La funcionalidad de las copias externas funcionan bien, aunque habría que testearlas al máximo ya que el repositorio está en la carpeta general y no esta en la que posiblemente se descargue para ejecutar el código. Habrá que testearlo en modo profesor

*Problemática encontrada*

### Sprint 2 Meeting 3

**Fecha reunión:16/12/2018 15:00**

*Descripción del trabajo*

**Rafael Cantero Alén**

Estimadas: 5 h Ejecutadas: 4 h

Notas trabajo:

Se han finalizado las funcionalidades siguientes:

* HU\_06\_EliminarAlumno|3|2
* Interfaz EliminarAlumno|3|1

Con ello quedan finalizadas todas las funcionalidades asignadas.

*Problemática encontrada*

**Carlos Freire Caballero**

Estimadas: 9 h Ejecutadas: 16 h

Notas trabajo:

* Interfaz ModificarAlumno|5|3

Además de la implementación de modificar alumno, se han realizado una revisión general al código para verificar que cumple con los requisitos establecidos por el cliente.

*Problemática encontrada*

**Fernando Herrera Poch**

Estimadas: 7,5 h Ejecutadas: 16 h

Notas trabajo:

Se han mejorado tanto la interfaz como los controles de errores de las funciones

* HU\_05\_MostrarAlumnos|3|4
* Interfaz MostrarAlumnos|3|1
* HU\_09\_GuardarCopiaSeguridadExterna|4|0,25
* Interfaz GuardarCopiaSeguridadExterna|4|0,25
* HU\_10\_CargarCopiaSeguridadExterna|4|1
* Interfaz CargarCopiaSeguridadExterna|4|1

*Problemática encontrada*

No hemos previsto en nuestra planificación la cantidad de trabajo que supone realizar la interfaz de las funciones y las pruebas de las funciones necesarias para su correcta implantación. Esto ha provocado que hayamos superado con creces las horas previstas de ejecución del proyecto.

En cuanto a las funcionalidades, no todas están implementadas con el mismo nivel de detalle y hay algunos requisitos que se han quedado pendientes de ejecutar, como eliminar por Apellido o cuando se sale de la aplicación el hecho de preguntar si se desea guardar la copia de seguridad si se es profesor o de guardar la copia externa. Se realiza directamente.

### Sprint 3 Meeting 1

**Fecha reunión:20/12/2018 23:00**

Descripción del trabajo

**Rafael Cantero Alén**

Estimadas: 4 h

Ejecutadas: 1 h

Notas trabajo:

Se ha realizado la actualización del diagrama de clases.

Problemática encontrada

**Carlos Freire Caballero**

Estimadas: 7 h Ejecutadas: 1 h

Notas trabajo:

Se han realizado las matrices de trazabilidad de Requisitos Funcionales frente a Casos de Uso y de Casos de Uso frente a Clases.

Problemática encontrada

Duda acerca del Requisito Funcional de Introducir Alumno, Modificar Alumno y Eliminar Alumno sobre

si son cubiertos únicamente por su respectivo Caso de Uso o también por el Caso de Uso de Buscar Alumno.

De igual forma ocurriría con la Gestión de Líderes para Introducir y Modificar

**Fernando Herrera Poch**

Estimadas: 4 h

Ejecutadas: 1 h

Notas trabajo:

Se ha realizado el documento base, incluyendo el Índice del documento y la preparación de las referencias a tablas, ilustraciones y bibliografía.

Problemática encontrada

### Sprint 3 Meeting 2

**Fecha reunión:22/12/2018 22:00**

*Descripción del trabajo*

**Rafael Cantero Alén**

Estimadas: 4 h

Ejecutadas: 1,5 h

Notas trabajo:

Se ha modificado el diagrama de clases y se ha realizado la documentación referente al mismo.

*Problemática encontrada*

**Carlos Freire Caballero**

Estimadas: 7 h

Ejecutadas: 5 h

Notas trabajo:

Se ha realizado la revisión de los SD, también se han modificado las matrices de validación y finalmente se han dado algunos retoques al código.

*Problemática encontrada*

**Fernando Herrera Poch**

Estimadas: 4 h

Ejecutadas: 3 h

Notas trabajo:

Se ha incluido en la documentación los requisitos de la aplicación, las historias de usuario y los casos de uso. Falta por mejorar las referencias bibliográficas y las relaciones de las tablas e ilustraciones.

*Problemática encontrada*

### Sprint 3 Meeting 3

**Fecha reunión:23/12/2018 23:00**

Descripción del trabajo

**Rafael Cantero Alén**

Estimadas: 4 h

Ejecutadas: 3 h

Notas trabajo:

Se ha realizado la documentación correspondiente a la metodología Scrum.

Problemática encontrada

**Carlos Freire Caballero**

Estimadas: 7 h

Ejecutadas: 7 h

Notas trabajo:

Inclusión de la documentación pertinente (matrices de validación y diagramas de secuencia) en el documento final.

Problemática encontrada

**Fernando Herrera Poch**

Estimadas: 4 h

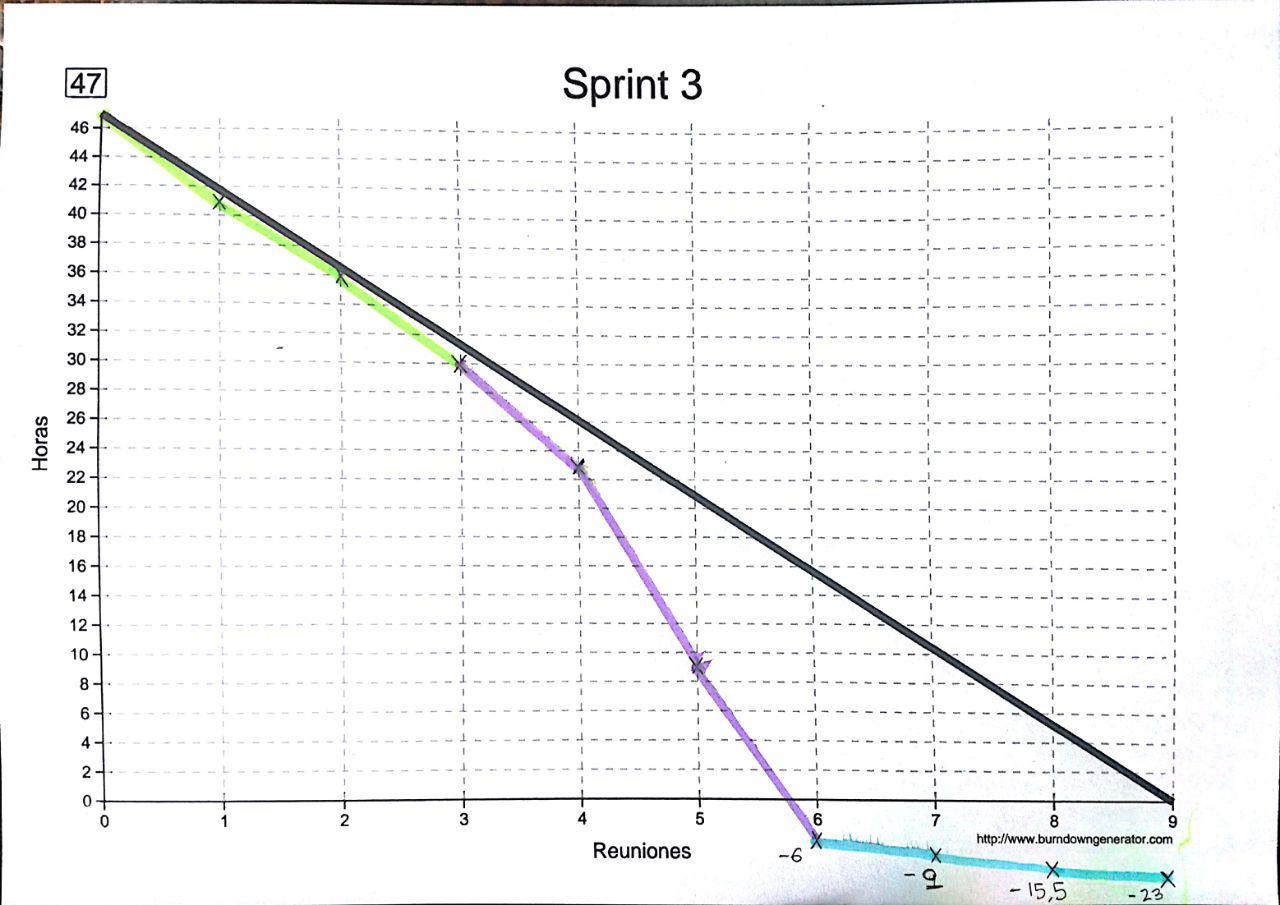
Ejecutadas: 4 h

Notas trabajo:

Se finaliza el trabajo con la inclusión de las referencias bibliográficas y la revisión del documento final.

Problemática encontrada

## Burndown Chart

Para la comparación de las horas destinadas a la realización de la aplicación con las horas que se habían estimado previamente en el Product BackLog, disponemos del Burndown Chart, una gráfica en la que se muestra el avance de la aplicación en horas respecto a cada reunión que se ha producido.

## Sprint Retrospective Meeting

El objeto de la reunión de retrospectiva es mejorar la aplicación de la metodología Scrum por parte del equipo de trabajo. Cada sprint debe terminar con el Sprint Retrospective. Dada la naturaleza del proyecto y su temporalidad se realizará una única reunión para evaluar el resultado final del proceso de Scrum destacando los aspectos en los que se pudiera mejorar para las siguientes ocasiones en las que nos enfrentemos al desarrollo de un sistema software basado en la metodología ágil Scrum.

### Reunión: Retrospective meeting

**Fecha reunión:23/12/2018 23:30**

ASPECTOS A DESTACAR DEL TRABAJO DESARROLLADO

* Uno de los aspectos que más han incidido en el desarrollo del trabajo ha sido pensar que el Código se implementaría en tres sesiones de Sprint. Con esa base desarrollamos una planificación de la implantación del código que tuvimos que comprimir en dos semanas. Como fuimos informados en la segunda semana tuvimos que desarrollar las funciones previstas en dos semanas en una sóla. Tuvimos poco tiempo para ir analizando los problemas y los tuvimos que ir resolviendo sobre la marcha y no como estaba previsto en las reuniones del Sprint. Estas reuniones se basaron en poner en común el trabajo que habíamos realizado, analizar los problemas de desarrollo, proponer soluciones y ver qué teníamos pendiente.
* Ha habido ocasiones en que hemos tenido que desarrollar funciones que no nos eran propias para poder evolucionar en nuestras funcionalidadades. Después las poníamos en común y las acababa de desarrollar quién tuviera esa funcioanalidad. El hecho de no estar trabajando en el mismo horario ni estar disponibles cuando el compañero lo necesitase ha hecho que tengamos que utilizar funciones tipo STUB no completamente definidas de acuerdo a las
* La metodología, bien llevada a cabo, es muy efectiva, pues nos permite a los integrantes del equipo ver como avanzan o no los progresos de nuestros compañeros, así como una comunicación fluida y rápida acerca de lo que se está trabajando.

ASPECTOS A MEJORAR

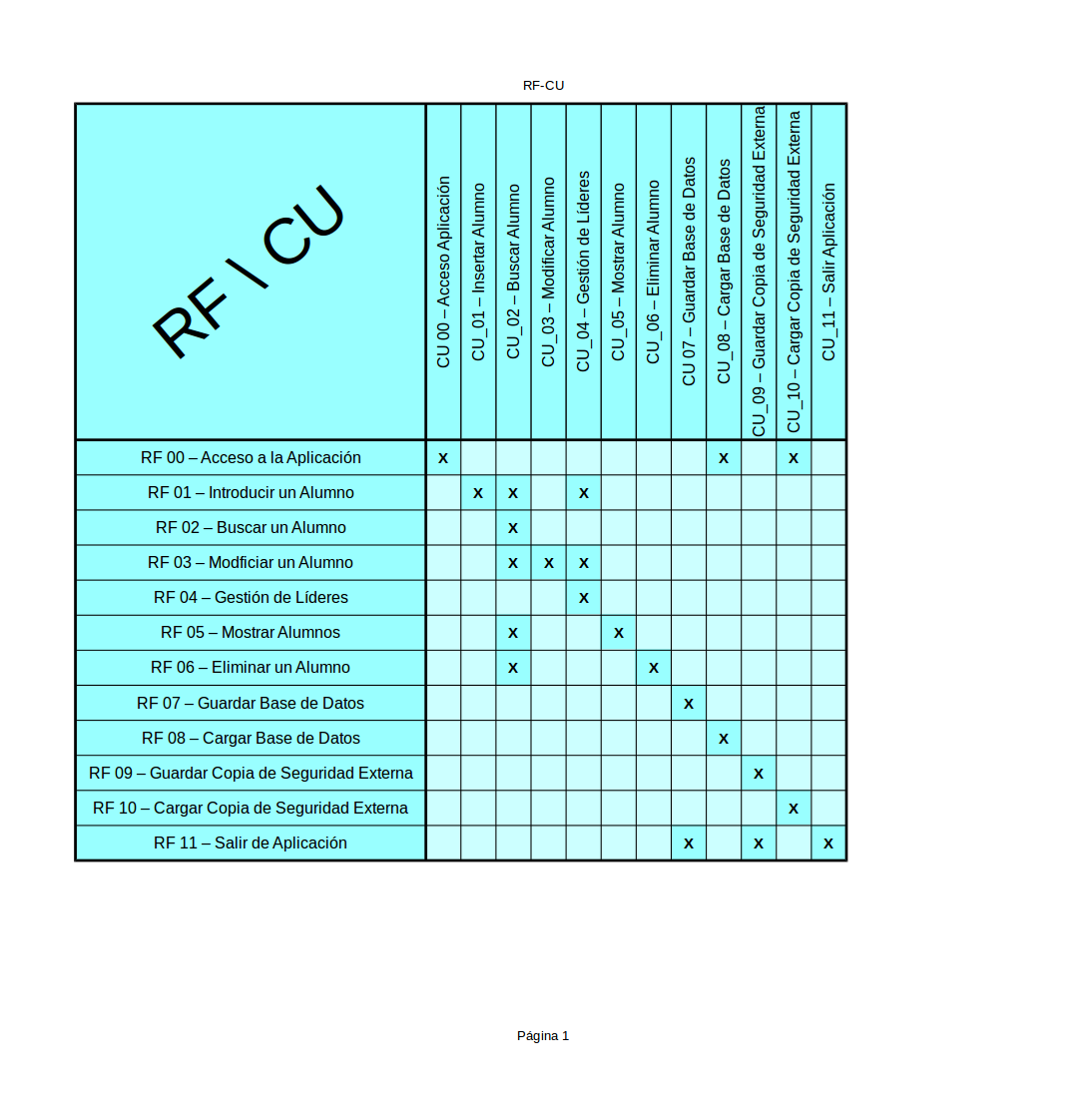
* El primer Sprint no fue muy productivo ya que quedaron funciones sin desarrollar plenamente, ya que no tuvieron ningún proceso de validación, simplemente se codificaron sin probarlas de ninguna manera. Después de este primer Sprint, se empezó a crear una interfaz con la que poder probar las funciones que se iban codificando. Esta interfaz finalmente se convirtió en el menú de la aplicación.
* Respecto al código se han cometido algunos errores debidos fundamentalmente al tiempo de desarrollo y a la inexperiencia programando c++. Entre ellas se encuentran:

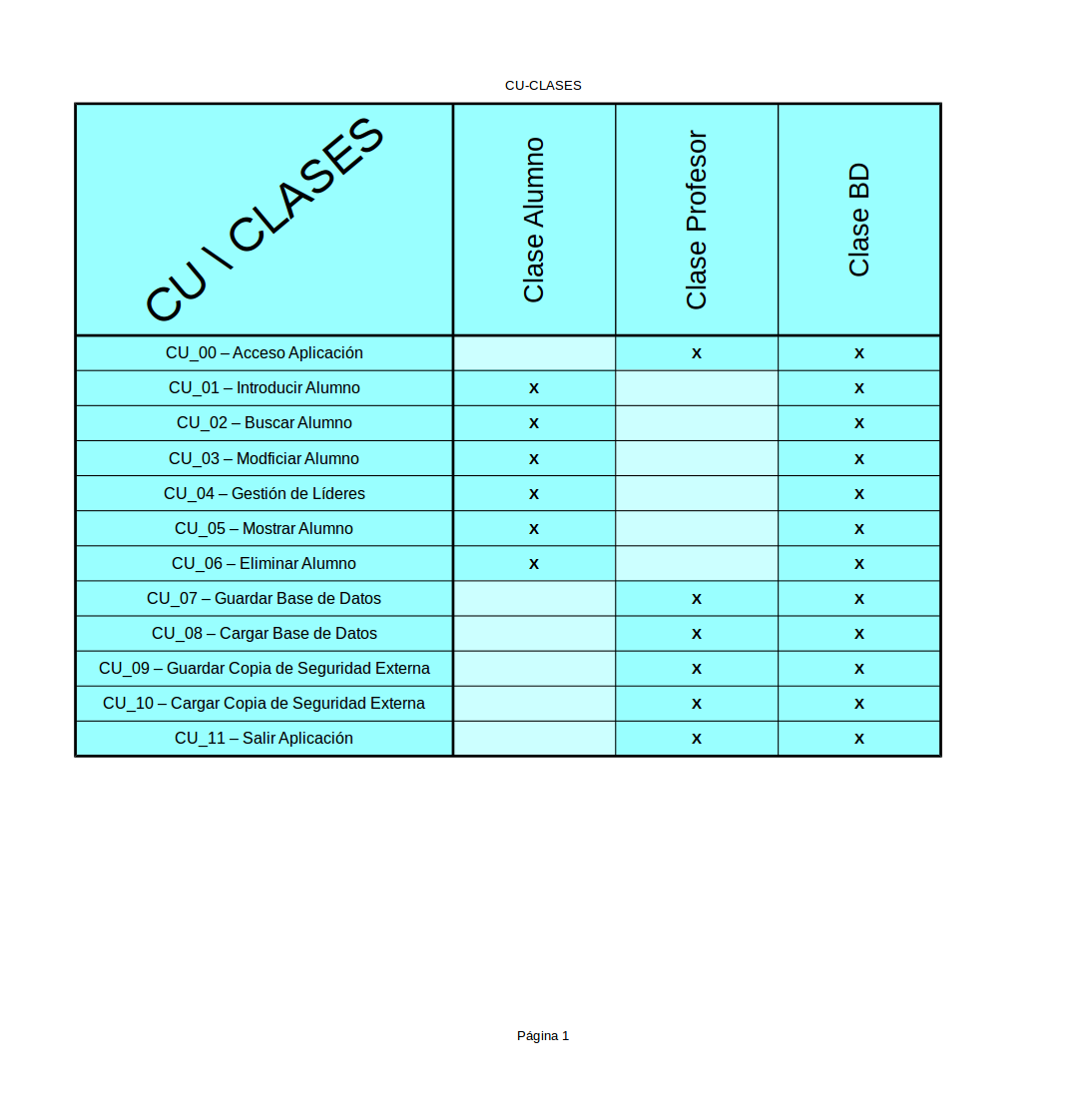
1 Interfaz demasiado compleja. Se podría haber trabajado en un patrón de diseño tipo Composite para la definición del menú. 2 Hay partes de la interfaz repetidas que se podrían haber integrado en funciones propias de la interfaz, como las de guardar/cargar copia de seguridad.

* Confirmación de los tiempos de entrega antes de comenzar el trabajo.
* En la planificación del trabajo fuimos muy optimismas con el tiempo de desarrollo pensando que lo desarrollaríamos en menos tiempo del que finalmente hemos necesitado. Este hecho fundamentalmente se ha debido a nuestra poca experiencia programando, a no haber previsto que el código hay que ir probándolo conforme lo vamos desarrollando. Es decir, no planteamos, que las funciones había que desarrollarlas, sino también probarlas mínimamente.
* Es preciso mejorar en el diseño de las clases, ya que de lo inicialmente previsto, han habido datos privados no usados (puntero a la base de datos en la clase profesor) o funciones que hemos tenido que crear que no estaban inicialmente previstas, como la de insertar alumno, que podía recibir un objeto alumno o bien una estructura de alumnos.
* Durante el desarrollo del código nos ha ocurrido, sobre todo al final, que la modificación de una función, ha afectado a otras partes del código. Ha sido el caso de la función de insertarAlumno(Alumno nuevoAlumno);. Si hay una función definida de una forma, es mejor preguntar si alguien la está usando en otra parte del código, antes de modificarla. En caso de duda, es mejor crear otra nueva para la funcionalidad que se esté desarrollando ya que no sabemos si la modificación puede afectar a otra parte del código.
* Debido a que es la primera vez que trabajamos con ramas en git, nos ha costado, sobre todo al inicio del desarrollo del código, adaptarnos a esta nueva forma de trabajo, provocando que trabajasemos un poco más lentos e inseguros, hasta que nos hemos acostumbrado plenamente a esta dinámica.
* La comunicación no ha sido tan buena como cabría esperar, dando lugar a errores del tipo: modificar parte de una funcionalidad que está realizando un compañero, o tener trabajo adelantado en la rama individual que debería estar en la rama master. Entendemos que se necesita más práctica y más trabajo. Pese a todo, el trabajo en general ha dado buenos frutos, pues en un tiempo relativamente corto se ha llevado a cabo la realización de un "software" capaz de gestionar los alumnos de una asignatura.

|  |  |
| --- | --- |
|  |  |
|  | VERIFICACIÓN DE REQUISITOS |

## Matrices de Verificación.





|  |  |
| --- | --- |
|  |  |
|  | Bibliografía y referencias Web |
|  |  |

ALFARO-HERRERA, J. C. (2017). Scrum como metodología para proyectos de redes. *Revista de Tecnologías Computacionales*, 38-42.

Arias Chaves, M. (2007). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *Revista InterSedes*.