Trabajo final Ingeniería del Software

UNIVERSIDAD D CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

Grupo 28

Cuadrado Bastón, Francisco
Rodríguez Aguayo, Iván
Ruiz Ventura, Andrés

Indice

1.Definición del problema

El profesorado de la asignatura de Ingeniería del Software quiere informatizar los datos de contacto de los alumnos en un programa informático que guarde esta información. Para ello, sólo habrá un profesor coordinador de la asignatura, y tantos profesores ayudantes como se quiera.

2.Extracción de requisitos

Partes interesadas:

- Profesor encargado de la asignatura.
- Profesores ayudantes.

Funcionales:

- Mostrar información de alumno: los campos de información que debe mostrar cada alumno son: dni, nombre, apellidos, teléfono, email corporativo, dirección(calle), curso más alto, fecha de nacimiento, equipo al que pertenece, es líder o no.
- Añadir alumno: Se permitirá añadir nuevos alumnos al sistema siempre y cuando no se sobrepase el límite de usuarios. No deben existir alumnos con los mismos campos identificadores.
- Buscar alumno: Se permitirá la búsqueda de alumnos tanto por su DNi, como por su apellido o grupo de trabajo, siendo el caso que los apellidos sean iguales, se muestre la información de ambos alumnos.
- Buscar grupo: se debe mostrar todos sus integrantes.
- Editar alumno: Se podrán modificar los datos de cada alumno, para ello buscando por su DNi, o bien por su apellido, en el caso de conflicto por varios alumnos con el mismo apellido, se permitirá elegir cuál es el que hay que modificar.

- Eliminar alumno: Se podrá eliminar a un alumno de la base de datos buscándolo por DNi o sus apellidos, y como en el caso anterior, si hay varios alumnos, se pedirá confirmación pediendo el DNI de cual hay que eliminar.
- Copia de seguridad: Se podrá hacer una copia de seguridad de los datos de los alumnos en cualquier momento, y cargar copias anteriores de ellos.
- Mostrar lista de alumnos: Se permitirá mostrar la lista de alumnos ordenados por nombre, apellidos, DNI o curso más alto de forma ascendente / descendente.
- Diversos usuarios del sistema: Se permite que existan diversos profesores en el sistema, solo habiendo un profesor coordinador, los demas serán ayudantes.

No funcionales:

- El sistema operativo objetivo del programa es GNU/Linux, siendo opcional su ejecución en otros sistemas.
- Toda información es obligatoria, excepto el equipo al que pertenece y si es líder o no.
- Dentro de la propia base de datos no se debe almacenar información relevante al profesor.
- La ejecución del programa se llevará a cabo mediante la consola de comandos.
- Software lo más sencillo posible de utilizar.
- Las copias de seguridad se realizarán en ficheros binarios.
- Como máximo podemos almacenar datos de 150 alumnos.
- Sin límite de alumnos por grupo.
- Como máximo un líder por cada grupo.
- Idioma obligatorio: castellano.
- No puede existir dos alumnos con el mismo DNI ni email.
- Al ejecutar el programa se imprime la lista de alumnos se debe generar un fichero html o markdown que se mostrará al final del programa.
- El programa se realizará en C++.

Prioridad de requisitos:

1. Compatible con la plataforma Linux / C++.
2. Ejecución del programa mediante interfaz de comandos.
3. Buscar alumno.
4. Añadir alumno.
5. Imprimir datos.
6. Editar alumno.

7. Eliminar alumno.
8. Imprimir grupo.
9. Visualizar lista.
10. Guardar copia de seguridad.
11. Cargar copia de seguridad.
12. Tamaño máximo de la lista: 150 alumnos.
13. Ficheros binarios.
14. Generar fichero html o markdown al finalizar el programa.

3.Historias de usuario

(ANVERSO)
ID: 001 **Añadir alumno**

Como profesor, me gustaría poder almacenar información fundamental de mis alumnos: DNI, nombre, apellidos, email corporativo, dirección, curso más alto matriculado, fecha de nacimiento, equipo al que pertenece y si es líder o no.

Prioridad: 2 **Depende de:** Buscar alumno

(REVERSO)

- Se debe comprobar que no haya más de 150 alumnos.
- No pueden existir varios alumnos con el mismo DNI.
- Todos los campos son obligatorios.

(ANVERSO)
ID: 002 **Mostrar alumno**

Como usuario, debo poder buscar a un alumno en específico de mi base de datos, bien sea por DNi o por apellido, y obtener sus datos.

Prioridad: 3 **Depende de:** Buscar alumno

(REVERSO)

- El alumno debe existir en la base de datos.
- Si se busca por apellido y hay más de una persona con el mismo apellido, se deberá recurrir a una búsqueda por DNi.

(ANVERSO)
ID: 003 **Editar alumno**

Permite editar la información del alumno.

Prioridad: 4 **Depende de:** Buscar alumno

(REVERSO)

- No puedes asignarle el DNi de otro alumno.
- No puede volverse líder de su grupo si ya tiene un líder el grupo.

(ANVERSO)
ID: 004 **Eliminar alumno**

Permite eliminar de la base de datos a un alumno.

Prioridad: 5 **Depende de:** Buscar alumno

(REVERSO)

- Busca al alumno a borrar, y lo elimina del vector.

(ANVERSO)
ID: 005 **Imprimir grupo**

Muestra los integrantes del grupo proporcionado.

Prioridad: 6 **Depende de:** Buscar alumno

(REVERSO)

- Recorre el vector buscando alumnos que pertenezcan al grupo elegido.

(ANVERSO)
ID: 006 **Mostrar lista de alumnos**

Muestra por consola y por una página web la información de los alumnos dentro de la base de datos.

Prioridad: 7 **Depende de:** Ninguna

(REVERSO)

- Imprime por pantalla y guarda en un fichero HTML los datos de los alumnos, luego mediante un comando lo carga en el navegador predefinido.

(ANVERSO)
ID: 007 **Guardar fichero**

Guarda toda la información en una copia de seguridad dentro del sistema de archivos del ordenador donde se ejecute.

Prioridad: 8 **Depende de:** Ninguna

(REVERSO)

- Si no es capaz de crear el archivo, o si no puede escribir en el, da mensaje de error y se cierra.

(ANVERSO)
ID: 008 **Cargar fichero**

Carga la información de una copia de seguridad alojada en el sistema de archivos del ordenador al programa.

Prioridad: 9 **Depende de:** Ninguna

(REVERSO)

- El fichero debe estar en la misma carpeta que el **.exe** del programa.

(ANVERSO)
ID: 009 **Buscar alumno.**

En la búsqueda de los alumnos, quiero tener la posibilidad de filtrar los atributos por DNI, Apellido o Grupo de trabajo correspondiente.

Prioridad: 1 **Depende de:** Ninguna

(REVERSO)

- Se permitirá la búsqueda de alumnos tanto por su DNI, como por su Apellido o grupo de trabajo.
- Si se busca un grupo, se mostrarán todos los integrantes del mismo.

4.Casos de uso

Añadir alumno

ID: 001
Breve descripción: Añadir a nuestra base de datos la información de un alumno
Actores principales: Usuario
Actores secundarios: Alumnos
Precondiciones:
Ninguna
Flujo principal:
1. El programa le pide al usuario los datos del usuario
2. El usuario introduce los datos
Postcondiciones:
Los datos se almacenan en la base de datos
Flujos alternativos:
3.a. Todos los campos son obligatorios, si alguno no se introduce se repetirá la pregunta del campo hasta que se introduzca
3.b. El DNi ya existe, no se añade alumno y permite cambiar el DNi introducido
3.c. Se ha llegado al número máximo de alumnos, por lo que el alumno no se almacenará, dará un aviso y volverá al menú
3.d. Si el grupo tiene ya líder, se le marca como miembro; si no lo tiene se pregunta si él es el líder

Mostrar alumno

ID: 002
Breve descripción: Se realiza la búsqueda de un alumno en la base de datos y te muestra sus datos
Actores principales: Usuario
Actores secundarios: Alumnos
Precondiciones:
Debe existir al menos un alumno
Flujo principal:
1. El usuario introduce el DNi o apellido del alumno a buscar
2. El sistema busca a ese alumno
2.1. Si se busca por DNi, te muestra los datos del alumno
2.2. Si se busca por apellido y hay varias personas con el mismo apellido, te pide el DNi para identificar cual de todos es
Postcondiciones:
Te muestra los resultados por pantalla
Flujos alternativos:
2.a. Si el alumno no existe, te envía un mensaje de error

Editar alumno

ID: 003

Breve descripción: Busca un alumno en específico y modifica los datos guardados de él

Actores principales: Usuario

Actores secundarios: Alumnos

Precondiciones:

Debe existir el alumno

Flujo principal:

1. El usuario introduce los datos del alumno al cual modificar

2. El sistema busca al alumno

3. Se le pide al usuario los nuevos datos del alumno

Postcondiciones:

El sistema almacena los datos del alumno modificados

Flujos alternativos:

2.a. Si no existe el alumno, imprime un mensaje de error

3.a. Si el DNi coincide con el de otro alumno, no se actualiza el alumno y manda de vuelta al menú de edición de alumno

3.b. Se comprueba si es líder de grupo o no, si lo es se le pregunta si quiere dejar de serlo, si no lo es y en su grupo no hay líder se pregunta si quiere serlo, si lo hay directamente no le deja elegir nada

Eliminar alumno

ID: 004

Breve descripción: Como usuario quiero poder eliminar un alumno

Actores principales: Usuario

Actores secundarios: Alumnos

Precondiciones:

El alumno debe existir

Flujo principal:

1. El usuario introduce la información del alumno al que desea eliminar

2. El sistema busca al alumno en el vector

3. El sistema identifica al alumno

Postcondiciones:

Se eliminan del sistema todos los datos del alumno

Flujos alternativos:

3.a. Si el alumno no existe, no elimina nada y manda un mensaje de error

Imprime grupo

ID: 005

Breve descripción: Imprime por pantalla los integrantes del grupo seleccionado e información de ellos

Actores principales: Usuario

Actores secundarios: Alumnos

Precondiciones:

Ninguna

Flujo principal:

1. El usuario introduce el grupo que desea visualizar

2. El sistema busca el grupo en la base de datos y recopila la información

Postcondiciones:

Imprime por pantalla los integrantes del grupo

Flujos alternativos:

2.a. Si el grupo está vacío, imprime un mensaje de que se encuentra sin miembros

Mostrar lista de alumnos

ID: 006
Breve descripción: Quiero poder imprimir los datos de todos los alumnos y volcarlo a un fichero HTML
Actores principales: Usuario
Actores secundarios: Alumnos
Precondiciones:
1. Deben existir alumnos en el programa
Flujo principal:
1. El usuario pide la información de los alumnos
2. El sistema imprime por consola la información de los alumnos
3. El sistema escribe en un fichero HTML la información de los alumnos
4. El sistema abre el fichero HTML en el navegador
Postcondiciones:
1. El sistema muestra por consola la información de los alumnos y también abre el archivo HTML en el navegador para visualizarla
Flujos alternativos:
2.a. No hay alumnos en el sistema y manda mensaje de error.
3.a. Error al crear el fichero HTML

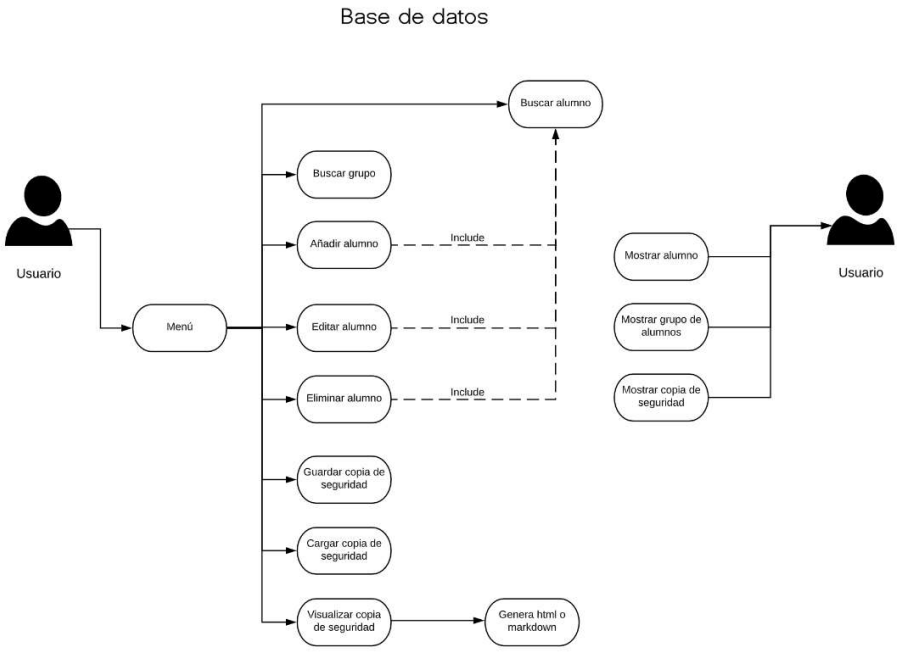
Guardar fichero

ID: 007
Breve descripción: Quiero poder guardar en un fichero mi lista de alumnos
Actores principales: Usuario
Actores secundarios: Alumnos
Precondiciones:
1. Debe existir algún alumno en el sistema
Flujo principal:
1. El usuario pide guardar la información en el fichero
Postcondiciones:
1. El sistema almacena los alumnos en un fichero binario en el sistema
Flujos alternativos:
1.a. No hay alumnos en el sistema, no se puede almacenar y manda mensaje de error
1.b. No se puede guardar el fichero y manda mensaje de fallo

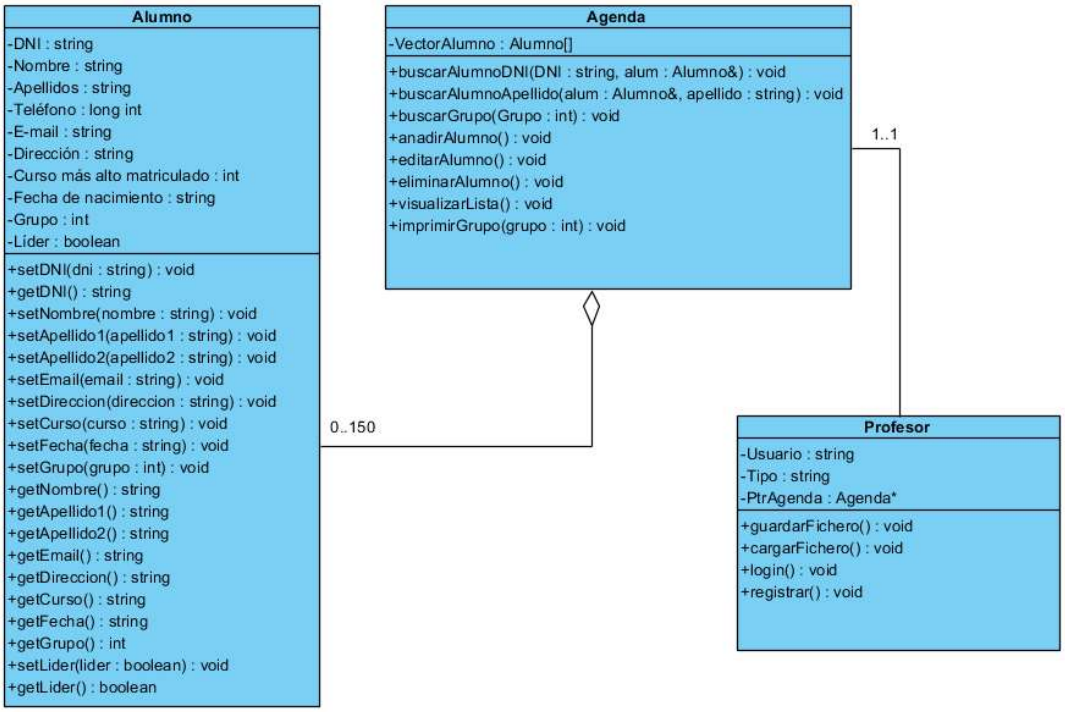
Cargar fichero

ID: 008
Breve descripción: Quiero poder cargar el fichero de mi lista de alumnos
Actores principales: Usuario
Actores secundarios: Alumnos
Precondiciones:
1. Debe existir el fichero en el sistema
Flujo principal:
1. El usuario pide cargar el fichero de alumnos
2. El sistema abre el archivo binario y recoge la información de él, guardandola en el programa
Postcondiciones:
1. El sistema almacena los datos del fichero en el programa
Flujos alternativos:
1.a. No existe el fichero en el sistema y manda mensaje de error
2.a. No se puede cargar correctamente el fichero y manda mensaje de fallo

Diagrama de caso de uso

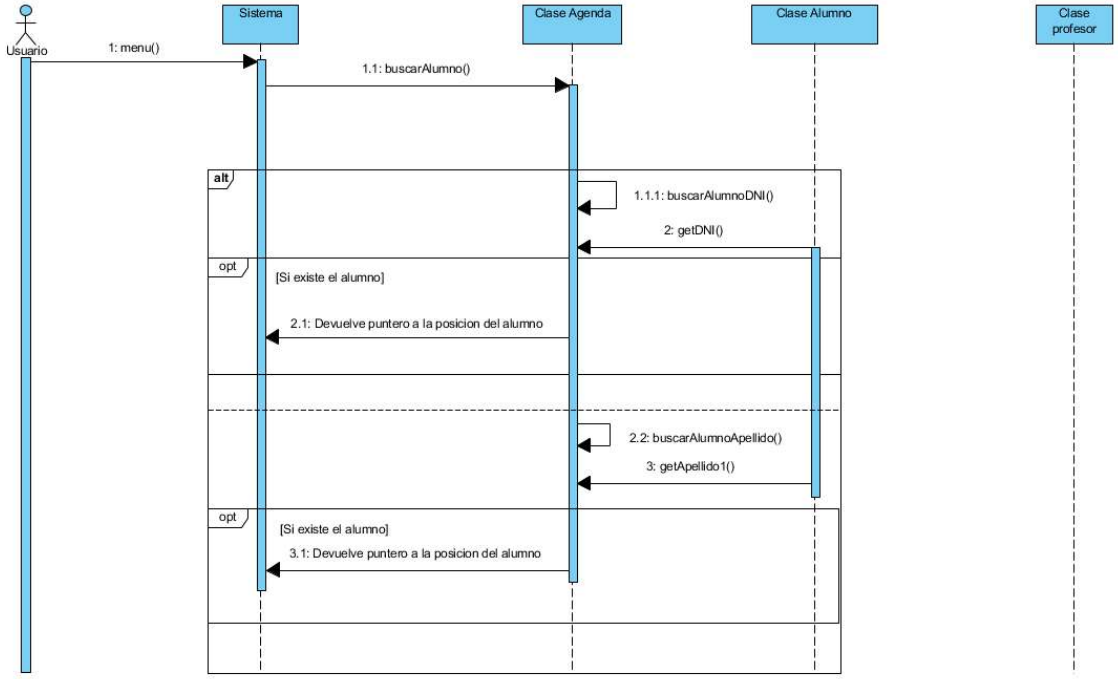


5.Diagrama de clases

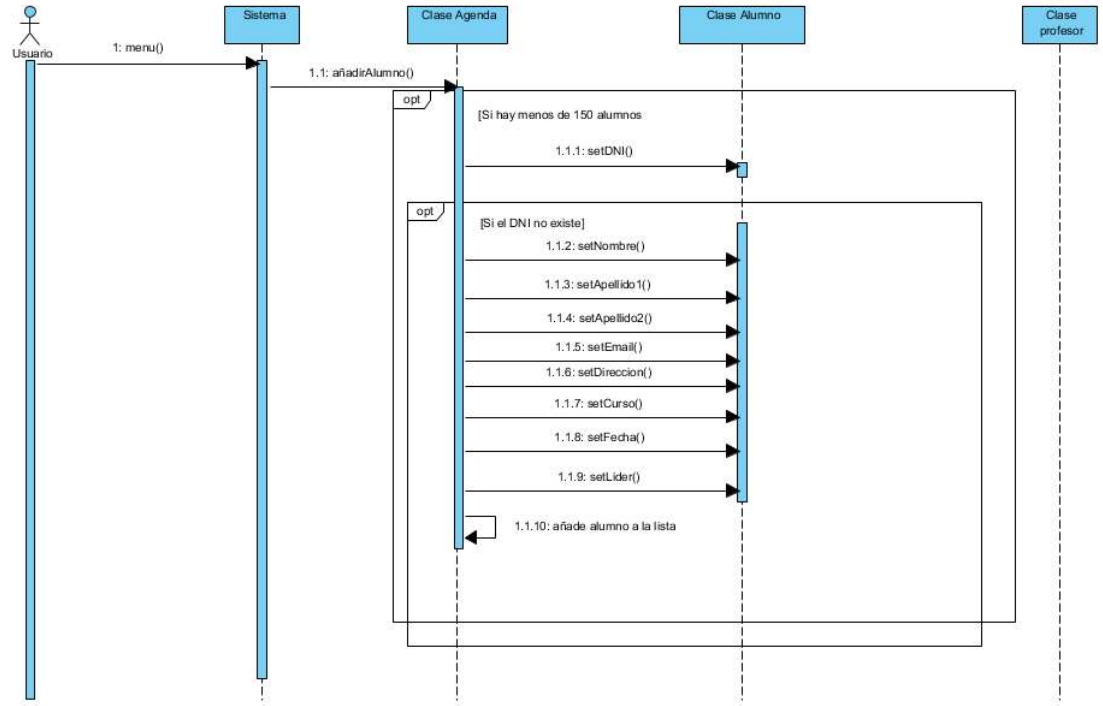


6.Diagramas de secuencia

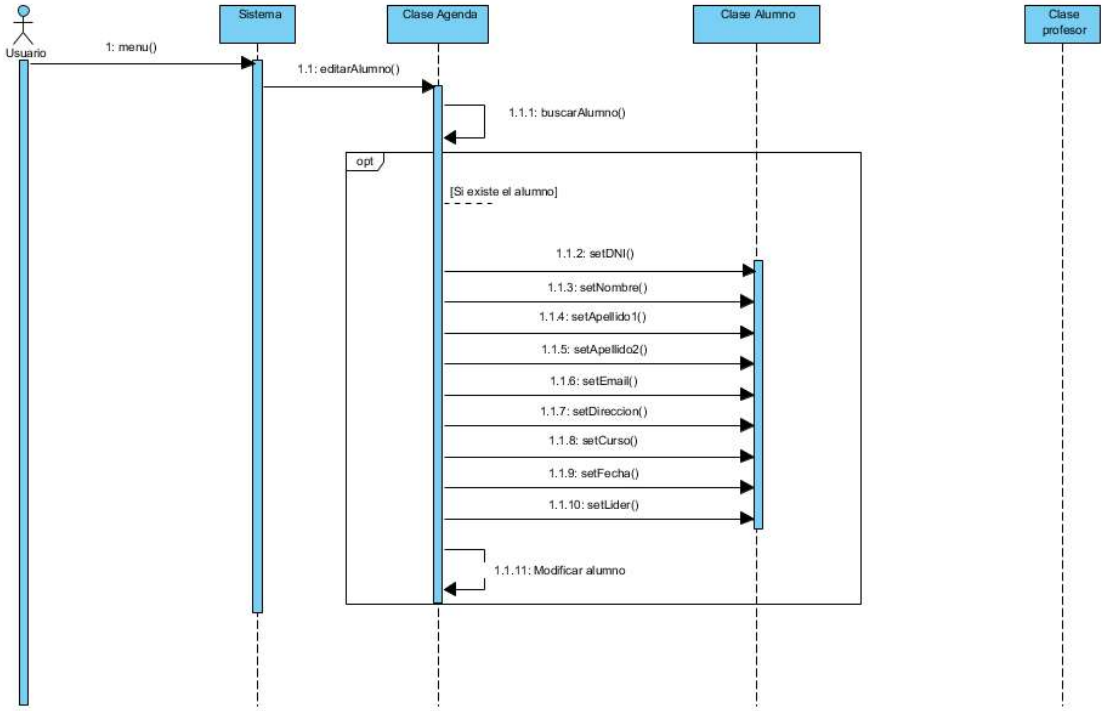
Buscar alumno



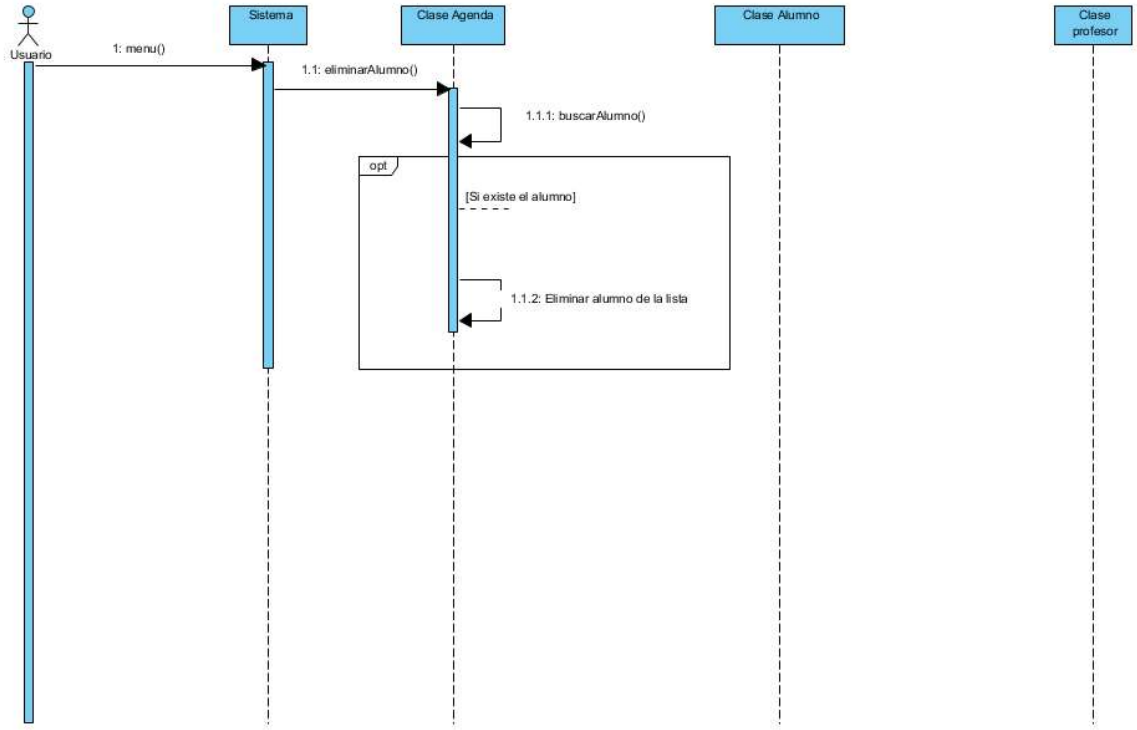
Añadir alumno



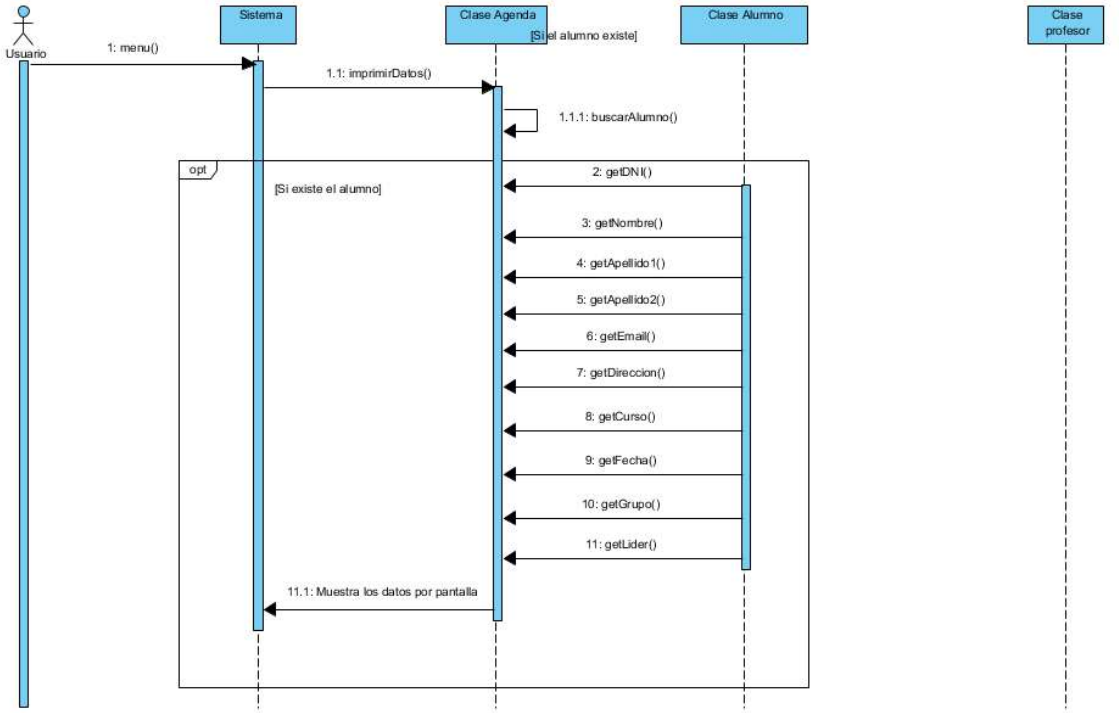
Editar alumno



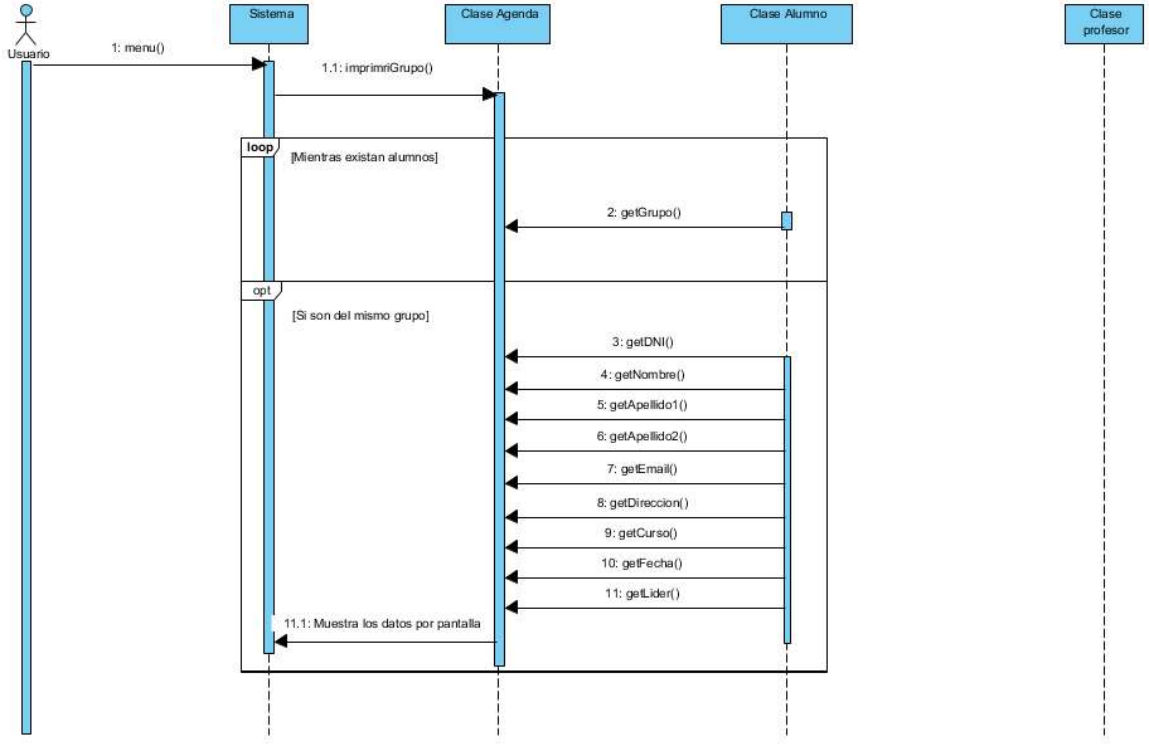
Eliminar alumno



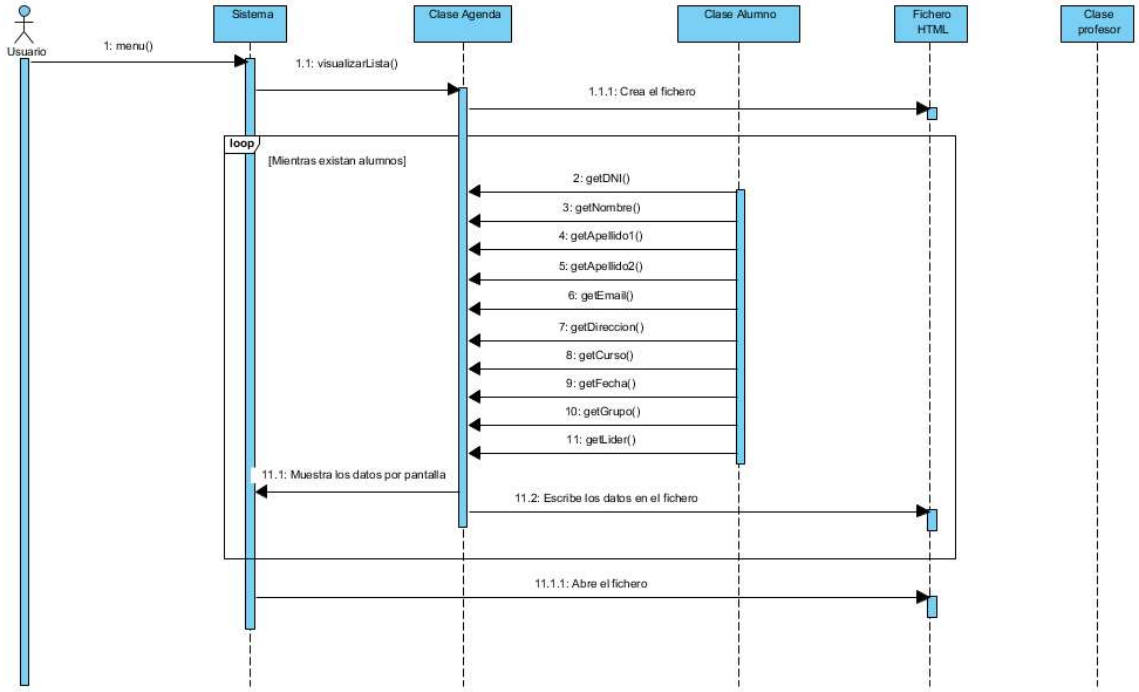
Imprimir datos



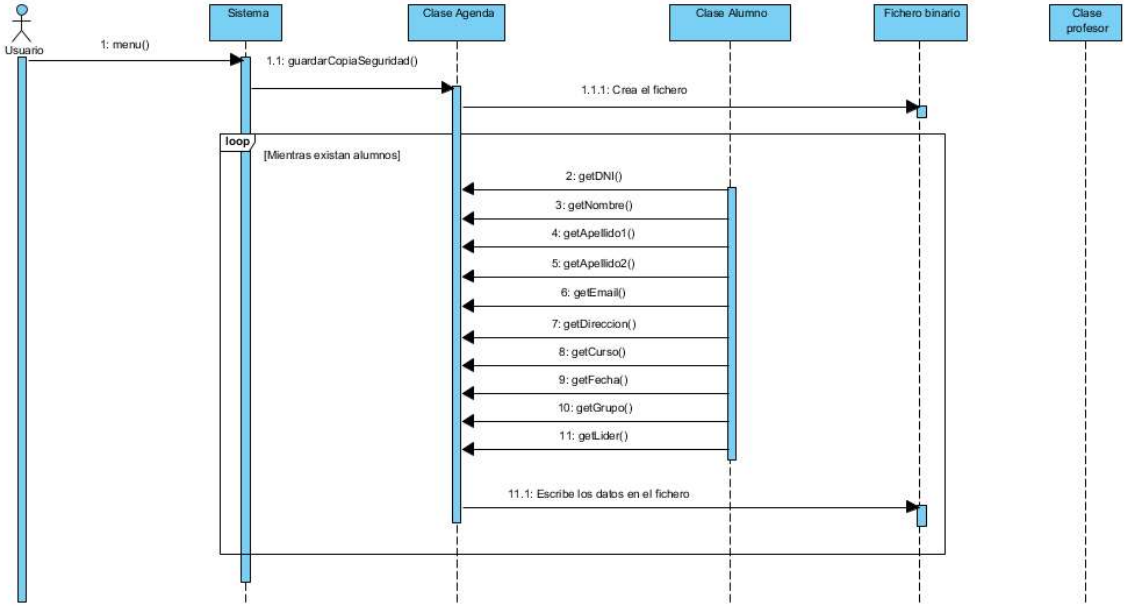
Imprimir grupo



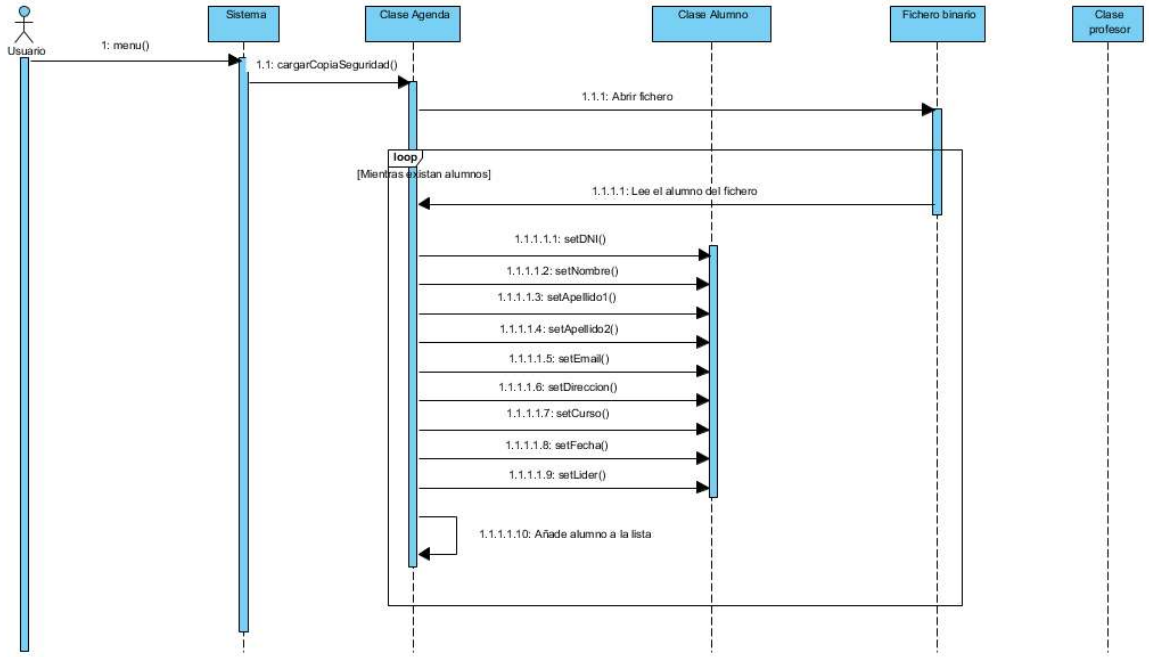
Visualizar lista



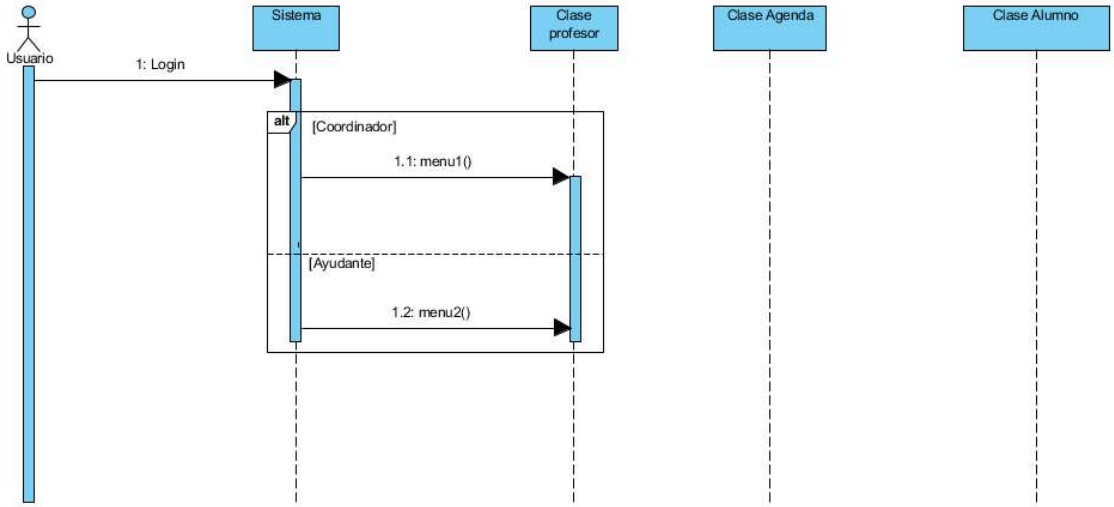
Guardar copia de seguridad



Cargar copia de seguridad



Login



7.Metodología scrum

7.1.Product backlog

Prioridad de tareas:

1. Funciones auxiliares: alumno.h (1h), agenda.h (1h), main.cc (1h).
2. Buscar alumno (2h)
3. Añadir alumno (2h)
4. Imprimir datos (2h)
5. Editar alumno (2h)
6. Eliminar alumno (2h)
7. Imprimir grupo (2h)
8. Visualizar lista (2h)
9. Guardar fichero (2h)
10. Cargar fichero (2h)
11. Corrección de errores (9h)

7.2.Sprint backlog

Asignación de tareas:

Sprint 1:

- Francisco Cuadrado Bastón - XescoC: funciones auxiliares(main.cc, agenda.h), buscar alumno.
- Andrés Ruiz Ventura - i72ruvea: añadir alumno, imprimir datos.
- Iván Rodriguéz Aguayo - Skr0tex: funciones auxiliares(alumno.h), editar alumno, eliminar alumno.

Sprint 1:

- Francisco Cuadrado Bastón - XescoC: imprimir grupo, visualizar lista, corrección de errores.
- Andrés Ruiz Ventura - i72ruvea: guardar fichero, cargar fichero.
- Iván Rodriguéz Aguayo - Skr0tex: corrección de errores.

7.3.Gráfico burndown

8.Bibliografía

-Moodle ingeniería del software: https://moodle.uco.es/m1819/course/view.php?id=2230