Documentación Prácticas Ingeniería del Software

0. Índice

1. Práctica 2:

1.1. Definición del problema.

-Se trata de crear una agenda o base de datos que almacene los datos e información de una serie de alumnos, de como máximo 150 alumnos.

-La información de los alumnos a almacenar es:

DNI, nombre y apellidos, teléfono, email de la UCO, domicilio, fecha de nacimiento, curso más alto que estén cursando, equipo de trabajo, líder del grupo o equipo de trabajo y la nota.

-En la creación de la agenda, se debe programar en c++ y las funciones que contengan se deben incluir en clases. Las funciones principales a realizar son:

Introducir alumnos, mostrar alumnos, buscar alumnos, modificar alumnos, borrar alumnos, realizar copia de seguridad, cargar copia de seguridad.

Con respecto a los profesores, se debe diferenciar si son coordinadores o ayudantes de la asignatura que impartan, dentro de un determinado departamento, así como guardar datos sobre ellos para que no estén metiendo siempre sus credenciales.

1.2 Extracción de requisitos

-Para la extracción de requisitos hemos reunido aquellas restricciones y funciones que debe de realizar nuestro programa, así como, los diferentes actores que intervienen en el propio programa.

-Existen dos actores: principales: que es el profesor; y secundarios: son los alumnos.

-Los datos que debe almacenar el programa son:

> DNI, nombre y apellidos, teléfono, email de la UCO, domicilio, fecha de nacimiento, curso más alto que estén cursando, equipo de trabajo, líder del grupo o equipo de trabajo y la nota.

-Los requisitos funcionales representa lo que debe hacer el sistema, según unas funciones determinadas se llega al comportamiento deseado del programa. Los requisitos funcionales son:

> ·Prioridad 1: Introducir alumnos.
>
> ·Prioridad 1: Mostrar alumnos.
>
> ·Prioridad 1: Buscar alumnos.

·Prioridad 2: Modificar alumnos.

·Prioridad 2: Borrar alumnos.

·Prioridad 3: Realizar copia de seguridad.

·Prioridad 3: Cargar copia de seguridad.

Cuanto menor sea la prioridad más importante será esa función en el programa.

-Requisitos no funcionales: son aquellos que establecen las restricciones del sistema.

·Programación en C++, siendo compatible en LINUX.

·Lenguaje de documentación: MarkDown.

·Ficheros de guardado de los alumnos en binario.

·Número total de alumnos 150.

·Interfaz: líneas de comandos.

·Visualización de los alumnos en línea de comandos y lenguaje HTML.

·Número mínimo de personas para la creación de un grupo: 1.

·Un grupo puede no tener líder.

1.3 Historias de usuario

Para las historias de usuario hemos establecido las diferentes funciones de cada operación que puede realizar el sistema dependiendo de la elección del profesor.

Indica lo que realiza o el comportamiento que tiene el programa al ejecutar la función definida anteriormente.

-Funciones con prioridad 1: son las funciones principales del programa, sin ellas sería imposible crear la agenda de alumnos:

Introducir alumnos con ID: 01, permite introducir alumnos en la agenda siempre que no se superen los 150 alumnos, además es obligatorio introducir todos los parámetros menos equipo y líder. Si por cualquier caso, se introduce un DNI o email, ya existente en la agenda, se producirá un error y se tendrán que volver a introducir los datos.

Historia de usuario 1.

(Anverso)

ID: 01 **Introducir alumnos.**

`Como usuario quiero poder introducir los datos de un alumno.`

Prioridad: 1.

(Reverso)

- Al introducir un usuario, debes introducir obligatoriamente todos los parámetros menos "equipo" y "lider".
- Si introduces un DNI o email que ya existe, mostrara un mensaje de error, y no se podrá introducir al alumno.

Mostrar alumnos con ID: 02, permite poder visualizar los datos de alumnos introducidos con anterioridad, para ello se puede elegir el número de alumnos a poder visualizar, así como, la forma en que se ordene la visualización ya sea: orden ascendente/descendente el equipo, nombre y apellidos, DNI o curso. Si al mostrar los alumnos alguno de ellos es líder, se resaltará su nombre con respecto a los que no son.

Historia de usuario 2.

(Anverso)

ID: 02 **Mostrar alumnos.**

`Como usuario quiero poder visualizar los datos de los alumnos.`

Prioridad: 1.

(Reverso)

- Los datos se puedn mostrar en orden alfabetico ascendente y descendente, por nombre y apellidos.
- Los datos se pueden mostrar en orden numérico ascendente y descendente respecto al DNI.
- Los datos se podran mostrar ascendente y descendente respecto al curso más alto matriculado.
- Se mostrarán el número de alumnos que el usuario desee.
- Se pueden mostrar los usuarios según su grupo.
- Si el usuario es lider de grupo, el nombre aparecerá en rojo en los ficheros HTML y entre * en línea de comandos.

Buscar alumnos con ID: 03, el usuario con esta función pretende buscar a los alumnos que desee según su DNI, apellido o email, también es posible buscarlos por grupo. Si al buscar por apellido, hay varios, entonces se mostrarán todos ellos.

Historia de usuario 3.

(Anverso)

ID: 03 **Buscar alumnos.**

`Como usuario quiero poder buscar los datos de los alumnos.`

Prioridad: 1.

(Reverso)

- La busqueda se hará mediante el DNI, apellidos o email.
- Si varios alumnos con el mismo apellido se mostraran todos.
- Podemos buscar a los alumnos por su grupo.

-Funciones con prioridad 2: son aquellas que dependen de las funciones con prioridad 1, ya que sin esas funciones sería imposible implementar estas:

Modificar alumnos con ID: 04, esta función permite modificar la información almacenada de los alumnos introducidos anteriormente. Si como en el caso de introducir, el DNI o email coinciden con otro saltará un mensaje de error y no guardará ningún cambio realizado en ese alumno.

Historia de usuario 4.

(Anverso)

ID: 04 **Modificar alumnos.**

`Como usuario quiero poder modificar los datos de los alumnos.`

Prioridad: 2.

(Reverso)

- Si al modificar el DNI o email de un alumno ya existente, se mostrará un mesaje de error y no guardará los cambios.

Borrar alumnos con ID: 05, con esta función se pretende borrar los alumnos que el profesor quiera, haciendo una previa búsqueda de ellos. Al hacer la operación borrar, el sistema mandará un mensaje de aviso para corroborar que se quiere borrar a uno o más alumnos.

Historia de usuario 5.

Anverso)

ID: 05 **Borrar alumnos.**

`Como usuario quiero poder borrar los datos de los alumnos.`

Prioridad: 2.

(Reverso)

- El programa preguntara si el usuario quiere borrar los datos antes de realizar la operación.

-Funciones con prioridad 3: son aquellas que dependen de las de prioridad 1 y 2, es decir, para que se puedan hacer primero tienen que introducirse alumnos, y después para poder hacer una copia de seguridad se debe modificar los datos de los alumnos:

Realizar copia de seguridad con ID: 06, el programa hará una copia de seguridad de los datos de los alumnos, siempre que haya introducido al menos 1 alumno, en caso contrario, mandará un mensaje de error el sistema cerrando el programa. Al realizar la copia, el programa enviará un mensaje de aviso para confirmar el salvado de los datos. La copia de seguridad se hará de forma manual, es decir, siempre que el usuario, en nuestro caso el profesor, lo desee, y no de forma automática, que es cuando termina una operación.

Historia de usuario 6.

(Anverso)

ID: 06 **Realizar copia de seguridad.**

```
Como usuario quiero poder realizar una copia de seguridad los datos de los alumnos.
```

Prioridad: 3.

(Reverso)

- El usuario hará de forma manual una copia de seguridad.
- El programa preguntará si quiere guardar los datos antes de cerrarlo.

Cargar copia de seguridad con ID: 07, esta función carga una copia de seguridad, para que esto sea posible, debe de haberse realizado anteriormente al menos una copia de seguridad. La forma de cargar la copia de seguridad es manualmente, es decir, el usuario decide cuando cargar la copia de seguridad.

Historia de usuario 7.

(Anverso)

ID: 07 **Cargar copia de seguridad.**

```
Como usuario quiero poder realizar una copia de seguridad los datos de los alumnos.
```

Prioridad: 3.

(Reverso)

- El usuario cargará de forma manual la copia de seguridad.

1.4 Casos de uso

En los casos de uso hemos reunido las diferentes interacciones que tienen las operaciones o funciones principales del programa, es decir, los requisitos funcionales provenientes del documento de extracción de

requisitos. Representan los diferentes caminos que realiza el sistema, dependiendo una serie de condiciones.

Es una lista de acciones del sistema que definen una serie de interacciones entre los actores y el sistema. Los casos de uso se dividen en varias partes: descripción, actores (indica los roles que intervienen en el sistema), precondiciones (restricciones del sistema antes de poder ejecutar la operación descrita), flujo principal (función de la operación), postcondiciones (la operación que realiza el sistema al terminar la función) y flujo alternativo (indica el camino de la operación en unos casos específicos).

-Introducir alumnos: el profesor introduce los datos los alumnos. Para que se pueda realizar esta operación es necesario que no se encuentre la memoria llena, es decir, que la agenda no supere los 150 alumnos almacenados.

Como flujo principal, esta función permite introducir los datos de los alumnos de forma secuencial (no se pueden introducir todos los alumnos a la vez).

Como postcondición el sistema guardará los datos de los alumnos introducidos.

Como flujo alternativo, si la memoria está llena el sistema mostrará un mensaje de error por pantalla avisando al usuario que no se pueden introducir más alumnos. Otro flujo alternativo es que los datos de un alumno coincidan con otro y el sistema enviará un error para volver a introducir los datos del alumno.

Introducir alumnos.

ID: 01

Breve descripción: El usuario introduce los datos de un alumno.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. Debe de haber memoria disponible (menos de 150 alumnos).
```

Flujo principal:

```
1. El caso empieza cuando el usuario desea introducir un alumno nuevo.
2. El usuario introduce los daos del alumno.
```

Postcondición:

- El sistema guarda los datos introducidos.

Flujo alternativo:

2.a. Si ya existe el alumno, se muestra un mensaje de error.

2.b. Si no hay memoria, se mostrará un mensaje de error.

-Mostrar alumnos: el sistema muestra los datos de los alumnos que el profesor quiera, para ello existe la precondición de que por lo menos exista un alumno que haya sido introducido anteriormente en la base de datos.

El flujo principal que se define en este caso de uso es: el sistema muestra uno o varios alumnos, habiendo recogido la información de cada uno de ellos con anterioridad.

La postcondición es que el sistema muestra los datos de los alumnos requeridos por pantalla.

Y el flujo alternativo es que al menos deben de estar almacenados los datos de alumnos que se quieran mostrar, si no, enviará un mensaje de error.

Mostrar alumnos.

ID: 02

Breve descripción: El sistema muestra los datos de los alumnos.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. El alumno debe existir.
```

Flujo principal:

```
1. El caso empieza cuando el sistema quiere mostrar uno o varios alumnos.
2. El sistema recoge los datos de los alumnos.
```

Postcondición:

- El sistema muestra los datos por pantalla.

Flujo alternativo:

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

-Buscar alumnos: el profesor puede buscar uno o varios alumnos.

Para ello es necesario que los datos de los alumnos coincidan con el alumno a buscar, es decir, que los alumnos se encuentren almacenados en la agenda.

El flujo principal es que el profesor introduce el DNI, email o apellidos del alumno o los alumnos a buscar.

La postcondición es que el sistema buscará según los datos introducidos.

Y el flujo alternativo viene determinado por el mensaje de error que el sistema muestra por pantalla dada la no existencia de alumnos que coincidan con los datos introducidos para la búsqueda.

Buscar alumnos.

ID: 03

Breve descripción: El usuario puede buscar uno o varios alumnos.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. El o los alumnos deben existir.
```

Flujo principal:

```
1. El caso empieza cuando el usuario desea buscar uno o varios alumnos.
2. El usuario introduce el DNI, apellidos o email.
```

Postcondición:

- El sistema realizará la operación especificada antes de empezar la busqueda.

Flujo alternativo:

2.a. Si no existe el alumno se mostrará un mensaje de error.

-Modificar alumnos: el profesor quiere modificar los datos de un alumno en concreto, por lo que como restricción antes de poder realizar esta operación es que exista el alumno que se desee modificar.

El flujo principal: el caso de uso comienza cuando el profesor decide modificar los datos de un alumno, que pueden ser aquellos que el profesor quiera, por ejemplo, todos o solo el teléfono y la nota.

La postcondición al igual que en los casos de uso anteriores se trata de realizar la operación descrita, que en este caso es: modificar los datos de los alumnos y almacenarlos en la agenda.

Y el flujo alternativo se puede definir como los mensajes de error que el sistema envía o muestra por pantalla al profesor al comprobar que, al modificar los alumnos, coinciden los datos únicos, como pueden ser el DNI y el email.

Modificar alumnos.

ID: 04

Breve descripción: El usuario puede modificar los datos de un alumno.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. El o los alumnos deben existir.
```

Flujo principal:

```
1. El caso empieza cuando el usuario desea modificar los datos de un alumno.
2. El programa preguntará que datos deseas modificar.
3. El usuario introducirá los nuevos datos.
```

Postcondición:

- El sistema sobreescribirá los datos.

Flujo alternativo:

2.a. Si no existe el alumno se mostrará un mensaje de error.

2.b. Si el DNI o email coincide con otro se mostrará un mensaje de error.

-Borrar alumnos: el profesor borra aquellos alumnos que desee.

La precondición que se define en este caso de uso es que debe existir el alumno a borrar.

El flujo principal sería introducir los datos de los alumnos a eliminar, en este caso, el DNI y el email, y buscar al alumno que coincida con esos datos para

eliminarlo. Por último, saldrá un mensaje de confirmación para borrar al alumno. También será posible borrar todos los alumnos introducidos.

La postcondición será borrar los alumnos deseados.

El flujo alternativo muestra un mensaje de error al no existir los datos del alumno que se quiere eliminar.

Borrar alumnos.

ID: 05

Breve descripción: El usuario puede borrar los datos de un alumno.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. El o los alumnos deben existir.
```

Flujo principal:

```
1. El caso empieza cuando el usuario desea borrar los datos de un alumno.
2. El usuario introducira el DNI o email del alumno que quiere eliminar.
3. El programa volverá a preguntar si quiere borrar los datos.
```

Postcondición:

- El sistema borrará los datos.

Flujo alternativo:

2.a. Si no existe el alumno se mostrará un mensaje de error.

-<u>Realizar copia de seguridad:</u> el profesor manualmente desea realizar una copia de seguridad de los datos almacenados, para no perderlos posteriormente ante cualquier fallo externo al programa. Como mínimo debe existir un alumno en la agenda (precondición).

El flujo principal determina que el programa realiza la copia de seguridad de los datos almacenados.

La postcondición será crear la copia de seguridad por parte del sistema.

Y por último el flujo alternativo será que solo se puede hacer una copia de seguridad si hay por lo menos un alumno introducido, por lo que, en este caso se mostrará un mensaje de error por pantalla.

Realizar copia de seguridad.

ID: 06

Breve descripción: El usuario realizará manualmente una copia de seguridad.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. Deben de existir como mínimo los datos de un alumno.
```

Flujo principal:

```
1. El caso empieza cuando el usuario desea crear una copia de seguridad.
```

Postcondición:

- El sistema creará la copia de seguridad.

Flujo alternativo:

2.a. Si no hay ningún dato guardado, mostrará un mesaje de error.

-Cargar copia de seguridad: el profesor decide cargar una copia de seguridad que haya sido realizada anteriormente (precondición), si se cumple, el programa en el flujo principal se encargará de cargar una copia de seguridad, lo más normal es que se cargue la más reciente, también puede servir si en alguno ejecución anterior del programa se han guardado datos, y ahora en esta ejecución se quieren utilizar mediante las funciones principales del propio programa (flujo principal).

La postcondición se trata de cargar la copia de seguridad en el sistema.

Y el flujo alternativo, se encargará como en la mayoría de los casos de mandar un mensaje de error, aunque en este caso en específico será por la no existencia de una copia de seguridad guardada en el sistema.

Cargar copia de seguridad.

ID: 07

Breve descripción: El usuario cargará manualmente una copia de seguridad.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

```
1. Deben de existir una copia de seguridad.
```

Flujo principal:

```
1. El caso empieza cuando el usuario desea cargar una copia de seguridad.
```

Postcondición:

- El sistema cargará la copia de seguridad.

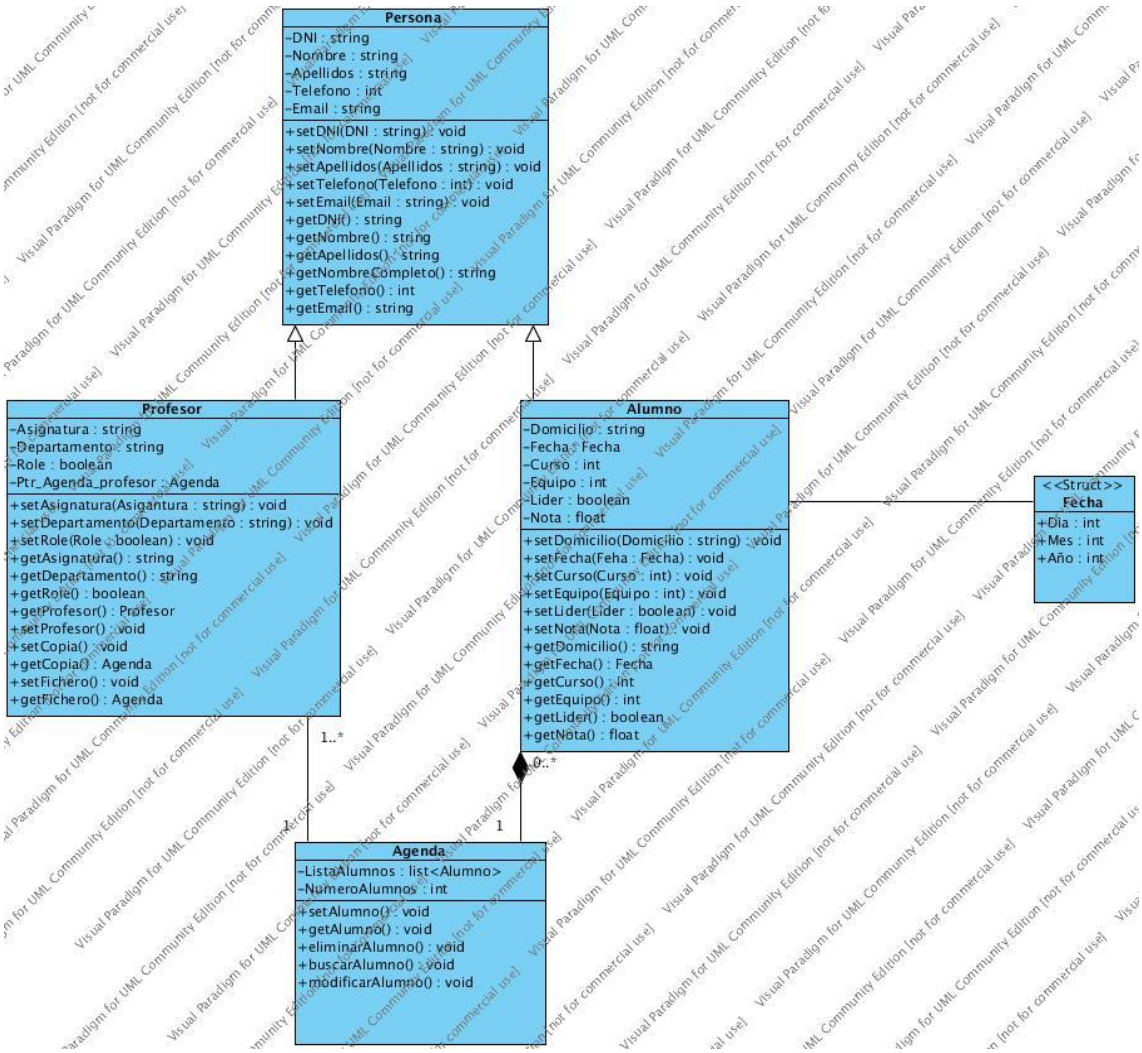
Flujo alternativo:

2.a. Si no hay ninguna copia de seguridad, mostrará un mesaje de error.

---

2. Practica 3

En esta práctica realizamos el diagrama de clases mediante Visual Paradigm de nuestro programa. Podemos encontrar 4 clases diferentes, las cuales son Persona, de las cuales heredan Profesor y Alumno (clases específicas, generalización) y por último la clase Agenda, relacionada con las anteriores pero sólo depende de la clase Profesor.



Por otra parte, hubo modificaciones en algunos casos de uso e historias de usuario. Y asimismo se añadieron los diagramas de secuencia de cada caso de uso, quedando de la siguiente forma:

1) Introducir alumno.

-Caso de uso:

ID: 01

Breve descripción: El usuario introduce los datos de un alumno.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. Debe de haber memoria disponible (menos de 150 alumnos).

Flujo principal:

1. El caso empieza cuando el usuario desea introducir un alumno nuevo.

2. El usuario introduce los daos del alumno.

Postcondición:

- El sistema guarda los datos introducidos.

Flujo alternativo:

2.a. Si ya existe el alumno, se muestra un mensaje de error.

2.b. Si no hay memoria, se mostrará un mensaje de error.

-Historia de usuario:

Historia de usuario 1.

(Anverso)

ID: 01 **Introducir alumnos.**

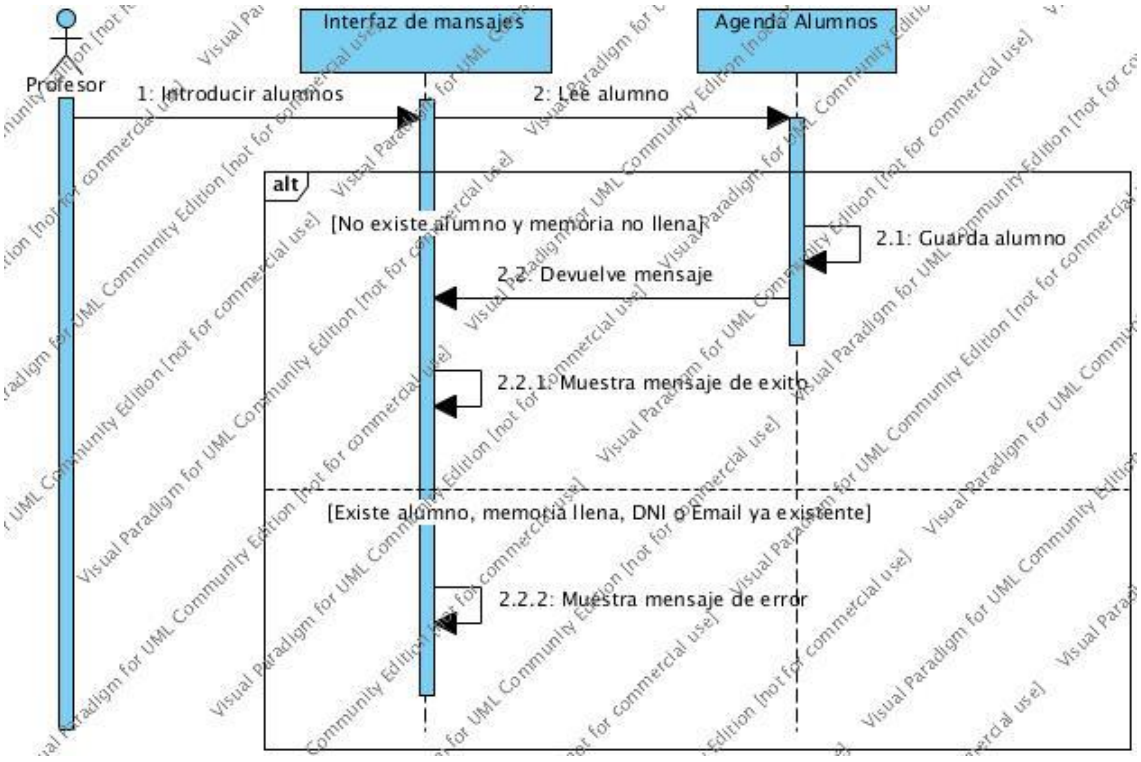
Como usuario quiero poder introducir los datos de un alumno.

Prioridad: 1.

(Reverso)

- Al introducir un usuario, debes introducir obligatoriamente todos los parámetros menos "equipo" y "lider".
- Si introduces un DNI o email que ya existe, mostrara un mensaje de error, y no se podrá introducir al alumno.

- -Diagrama de secuencia:



2) Mostrar alumnos.

-Caso de uso:

ID: 02

Breve descripción: El sistema muestra los datos de los alumnos.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. El alumno debe existir.

Flujo principal:

1. El caso empieza cuando el sistema quiere mostrar uno o varios alumnos.

2. El sistema recoge los datos de los alumnos.

Postcondición:

- El sistema muestra los datos por pantalla.

Flujo alternativo:

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

-Historia de usuario:

(Anverso)

ID: 02 **Mostrar alumnos.**

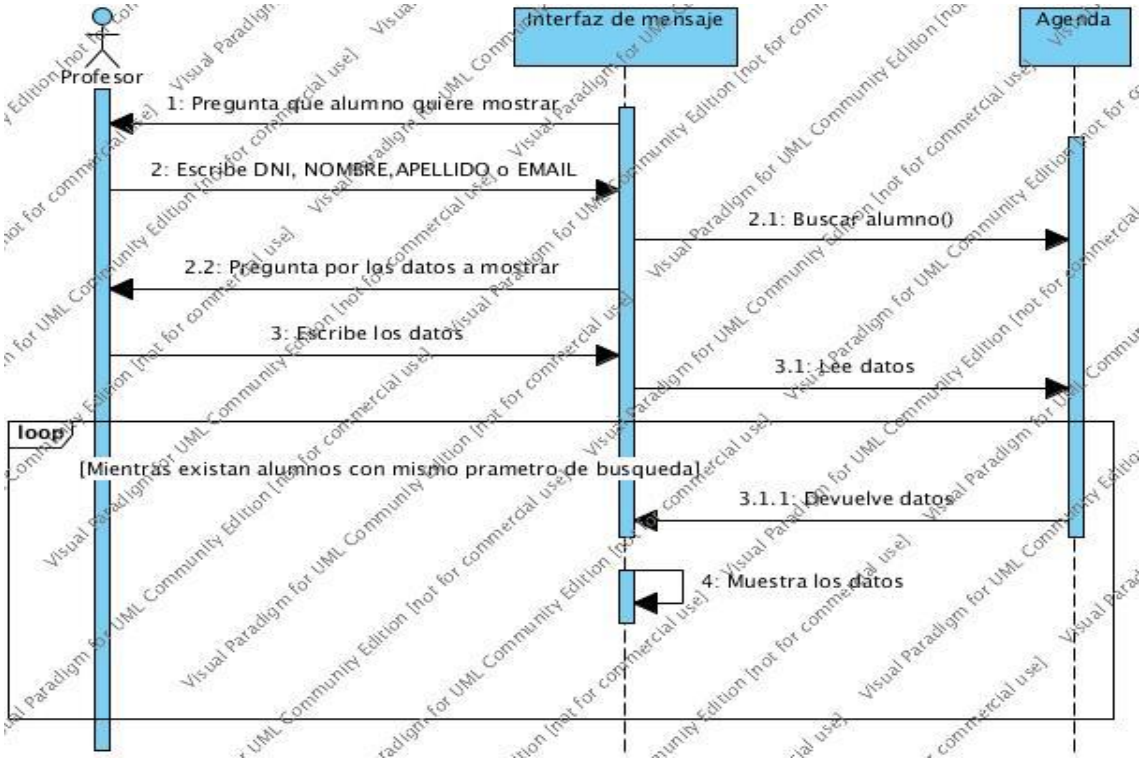
Como usuario quiero poder visualizar los datos de los alumnos.

Prioridad: 1.

(Reverso)

- Los datos se pueden mostrar en orden alfabético ascendente y descendente, por nombre y apellidos.
- Los datos se pueden mostrar en orden numérico ascendente y descendente respecto al DNI.
- Los datos se podrán mostrar ascendente y descendente respecto al curso más alto matriculado.
- Se mostrarán el número de alumnos que el usuario desee.
- Se pueden mostrar los usuarios según su grupo.
- Si el usuario es líder de grupo, el nombre aparecerá en rojo en los ficheros HTML y entre * en línea de comandos

- -Diagrama de secuencia:



3) Buscar alumnos.

-Caso de uso:

ID: 03

Breve descripción: El usuario puede buscar uno o varios alumnos.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. El o los alumnos deben existir.

Flujo principal:

1. El caso empieza cuando el usuario desea buscar uno o varios alumnos.

2. El usuario introduce el DNI, apellidos o email.

Postcondición:

- El sistema realizará la operación especificada antes de empezar la busqueda.

Flujo alternativo:

2.a. Si no existe el alumno se mostrará un mensaje de error.

-Historia de usuario:

(Anverso)

ID: 03 **Buscar alumnos.**

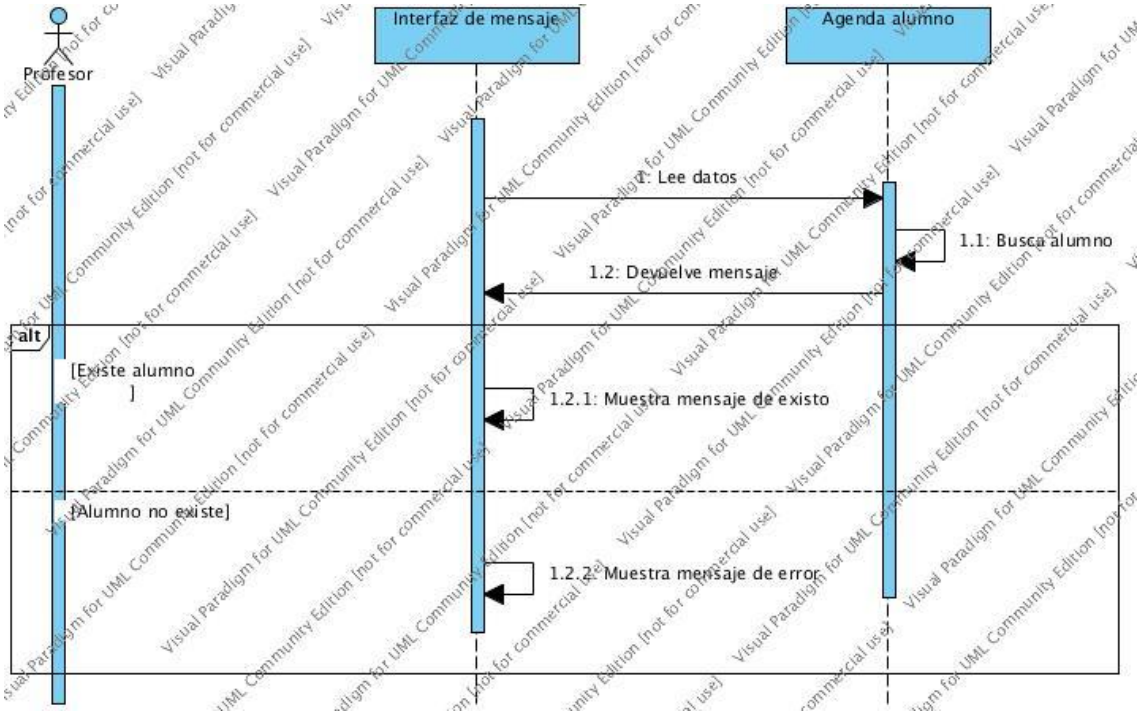
Como usuario quiero poder buscar los datos de los alumnos.

Prioridad: 1.

(Reverso)

- La busqueda se hará mediante el DNI, apellidos o email.
- Si varios alumnos con el mismo apellido se mostraran todos.
- Podemos buscar a los alumnos por su grupo.

-Diagrama de secuencia:



4) Modificar alumnos.

-Caso de uso:

ID: 04

Breve descripción: El usuario puede modificar los datos de un alumno.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. El o los alumnos deben existir.

Flujo principal:

1. El caso empieza cuando el usuario desea modificar los datos de un alumno.

2. El programa preguntará que datos deseas modificar.

3. El usuario introducirá los nuevos datos.

Postcondición:

- El sistema sobreescribirá los datos.

Flujo alternativo:

2.a. Si no existe el alumno se mostrará un mensaje de error.

2.b. Si el DNI o email coincide con otro se mostrará un mensaje de error.

-Historia de usuario:

(Anverso)

ID: 04 **Modificar alumnos.**

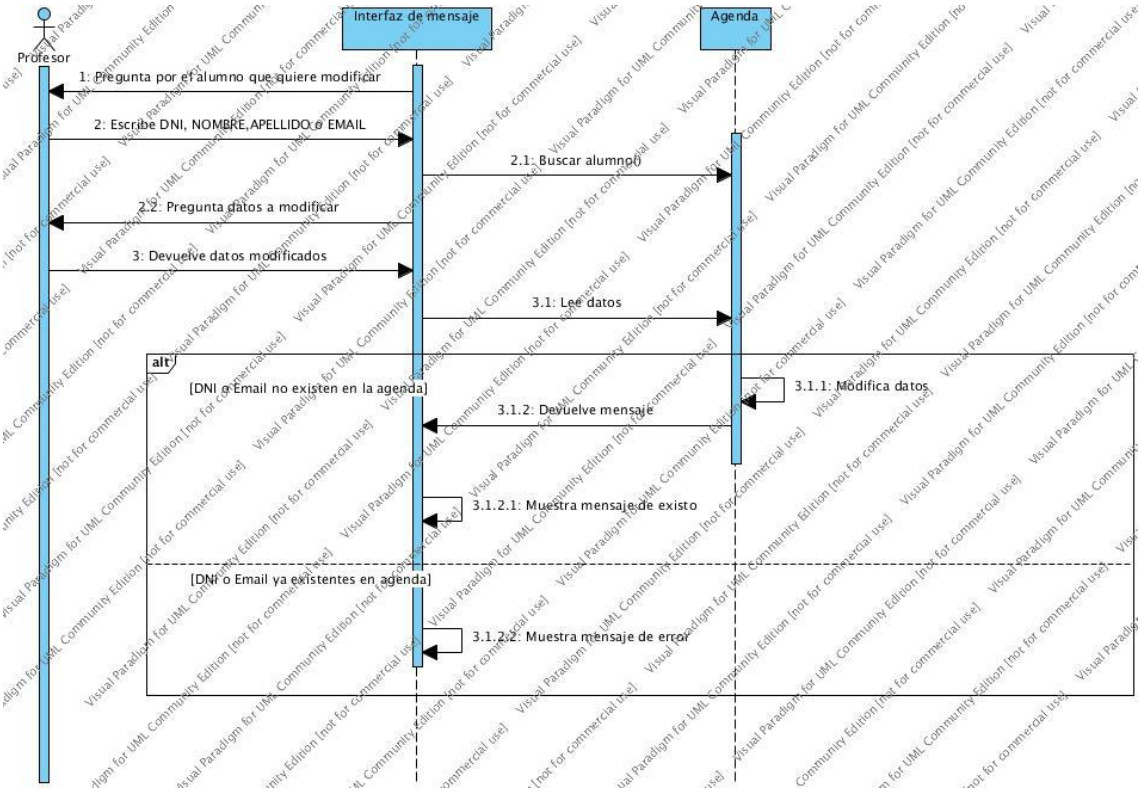
Como usuario quiero poder modificar los datos de los alumnos.

Prioridad: 2.

(Reverso)

- Si al modificar el DNI o email de un alumno ya existente, se mostrará un mensaje de error y no guardará los cambios.

-Diagrama de secuencia:



5) Borrar alumnos.

-Caso de uso:

ID: 05

Breve descripción: El usuario puede borrar los datos de un alumno.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. El o los alumnos deben existir.

Flujo principal:

1. El caso empieza cuando el usuario desea borrar los datos de un alumno.

2. El usuario introducira el DNI o email del alumno que quiere eliminar.

3. El programa volverá a preguntar si quiere borrar los datos.

Postcondición:

- El sistema borrará los datos.

Flujo alternativo:

2.a. Si no existe el alumno se mostrará un mensaje de error.

-Historia de usuario:

(Anverso)

ID: 05 **Borrar alumnos.**

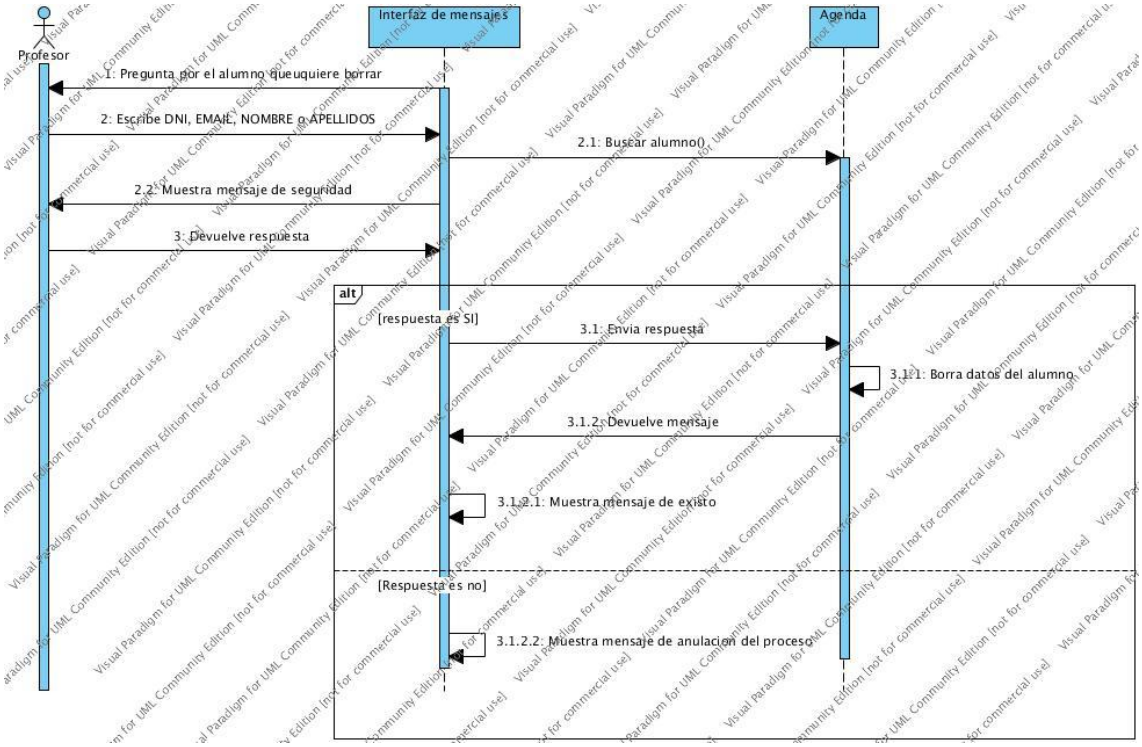
Como usuario quiero poder borrar los datos de los alumnos.

Prioridad: 2.

(Reverso)

- El programa preguntara si el usuario quiere borrar los datos antes de realizar la operación.

-Diagrama de secuencia:



6) Realizar copia de seguridad.

-Caso de uso:

ID: 06

Breve descripción: El usuario realizará manualmente una copia de seguridad.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. Deben de existir como mínimo los datos de un alumno.

Flujo principal:

1. El caso empieza cuando el usuario desea crear una copia de seguridad.

Postcondición:

- El sistema creará la copia de seguridad.

Flujo alternativo:

2.a. Si no hay ningún dato guardado, mostrará un mesaje de error.

-Historia de usuario:

(Anverso)

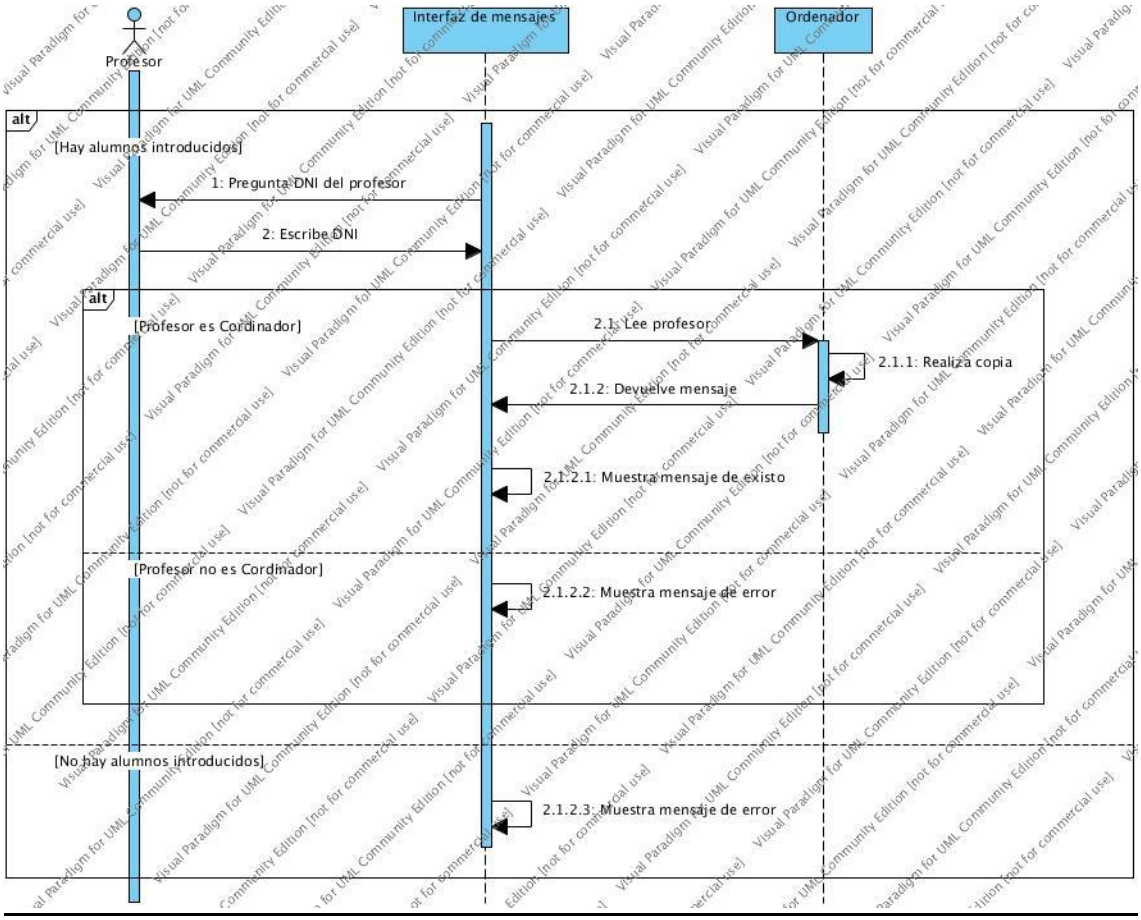
ID: 06 **Realizar copia de seguridad.**

Como usuario quiero poder realizar una copia de seguridad los datos de los alumnos.

Prioridad: 3.

(Reverso)

- El usuario hará de forma manual una copia de seguridad.
- El programa preguntará si quiere guardar los datos antes de cerrarlo.
- Diagrama de secuencia:



7) Cargar copia de seguridad.

-Caso de uso:

ID: 07

Breve descripción: El usuario cargará manualmente una copia de seguridad.

Actores principales: Profesor.

Actores secundarios: Alumnos.

Precondiciones:

1. Deben de existir una copia de seguridad.

Flujo principal:

1. El caso empieza cuando el usuario desea cargar una copia de seguridad.

Postcondición:

- El sistema cargará la copia de seguridad.

Flujo alternativo:

2.a. Si no hay ninguna copia de seguridad, mostrará un mensaje de error.

-Historia de usuario:

(Anverso)

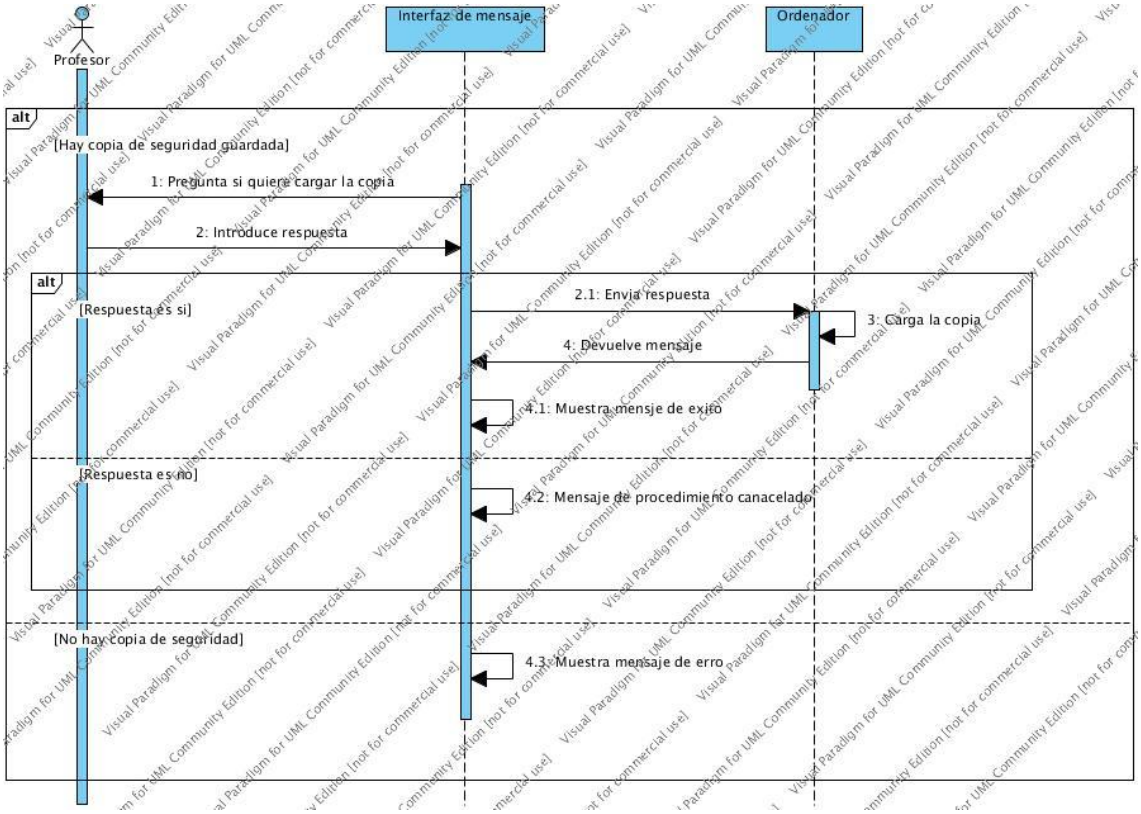
ID: 07 **Cargar copia de seguridad.**

Como usuario quiero poder realizar una copia de seguridad los datos de los alumnos.

Prioridad: 3.

(Reverso)

- El usuario cargará de forma manual la copia de seguridad.
- Diagrama de secuencia:



3. Practica 4:

A continuación, vamos a explicar lo realizado en la última práctica. En esta práctica, hemos repartido el trabajo de programación del programa de la forma más cualitativa posible, utilizando la metodología SCRUM para ir documentando este reparto de tareas.

Para ello, lo primero que hemos realizado es la definición del Product Backlog. En este documento se encuentran todas las historias de usuario definidas anteriormente, ordenadas por su prioridad funcional. O sea, la prioridad indicada en este documento es la prioridad desde el punto de vista del programador.

Hay que hacer un inciso importante, en el Product Backlog también se ha incluido un nuevo apartado, que es la creación de las clases que los programadores usaran para crear el programa. Esto se hace porque, aunque no sea una historia de usuario, la creación de estas clases es esencial para la creación del programa, por eso su prioridad es la más alta.

También es importante introducir el tiempo estimado de implementación de las diferentes historias de uso, para así poder calcular el tiempo que se va ha invertir en la creación de nuestro programa.

A continuación, podrá ver el Product Backlog:

Product Backlog

Crear las clases necesarias

Prioridad -1.

Buscar alumnos.

Prioridad 0.

Tiempo estimado de implementación: 2 horas.

Introducir alumnos.

Prioridad 1.

Tiempo estimado de implementación: 2 horas.

Realizar copia de seguridad

Prioridad 2.

Tiempo estimado de implementación: 3 horas.

Cargar copia de seguridad

Prioridad 3.

Tiempo estimado de implementación: 3 horas.

Mostar alumnos.

Prioridad 4.

Tiempo estimado de implementación: 1 horas.

Modificar alumnos.

Prioridad 5.

Tiempo estimado de implementación: 2 horas.

Borrar alumnos.

Prioridad 6.

Tiempo estimado de implementación: 1 horas.

Después de realizar el Product Backlog, se pasó a realizar el Sprint Backlog, un documento donde se indica que parte del programa se encarga de programar cada alumno.

Este documento, se creó en varias reuniones, ya que es difícil repartir de manera equitativa la cantidad de trabajo en la primera reunión que ha tenido el grupo. Por lo tanto, viendo como avanzaba cada uno con sus tareas y viendo cuanto tiempo quedaba para la entrega del programa, en cada reunión se hacían cambios en el Sprint Backlog.

Al final, el documento se quedó de esta manera:

Sprint Backlog

Unai Friscia Pérez.

- Product Backlog
- Sprint Backlog
- Crea struct fecha
- Crea clase alumno
- Crear clase agenda

Jose Manuel Gil Rodríguez.

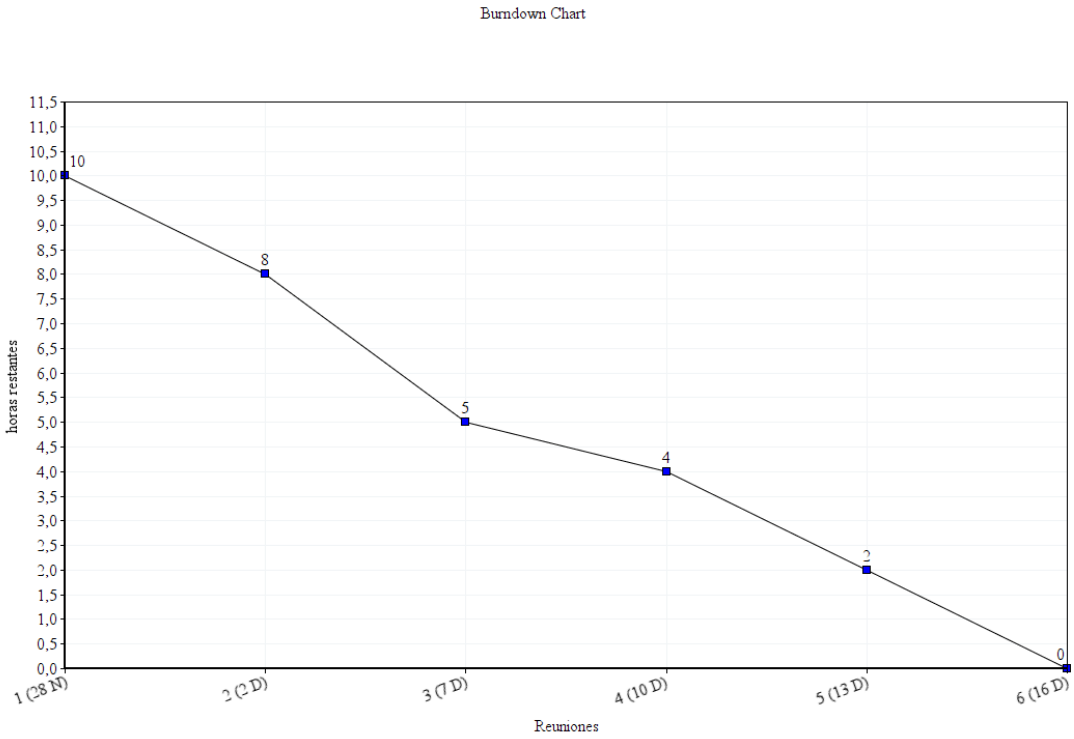
- Crea clase Persona
- Crea clase Alumno
- Crear programa principal
- Resolución de errores

Javier Gil Moya.

- Crea clase Profesor

- Crea clase alumno
- Resolución de errores

Por último, también se realizó el Burndown Char. Este documento es una gráfica, donde se indica el tiempo que resta de trabajo, según el número de reuniones que ha habido entre los integrantes del grupo.



Como se puede apreciar, el grupo ha tenido un total de seis reuniones, en las cuales, el tiempo estimado de trabajo que quedaba iba disminuyendo. Así hasta llegar al día 16, que fue nuestra última reunión, y también el día de la entrega del trabajo.

A continuación, vamos a realizar las matrices de validación:

Matriz de Requisitos Funcionales – Casos de Uso.

Requisitos Funcionales	Casos de Uso
Buscar alumno	Nº 3 => Buscar Alumno
Introducir alumno	Nº 1 => Introducir Alumno Nº 3 => Buscar Alumno
Realizar copia de seguridad	Nº 6 => Realizar copia de seguridad
Cargar copia de seguridad	Nº 7 => Cargar copia de seguridad

Mostrar alumno	Nº 3 => Buscar Alumno Nº 2 => Mostar Alumno
Modificar alumno	N º 3 => Buscar Alumno Nº 4 => Modificar Alumno
Borrar Alumno	Nº 3 => Buscar Alumno Nº 5 => Borrar Alumno

Matriz Caso de Uso – Clase

Casos de Uso	Clase
Nº 1 => Introducir Alumno	Agenda Alumno Persona
Nº 2 => Mostrar alumno	Agenda Alumno Persona
Nº 3 => Buscar Alumno	Agenda Alumno Persona
Nº 4 => Modificar Alumno	Agenda Alumno Persona
Nº 5 => Borrar Alumno	Agenda Alumno Persona
Nº 6 => Realizar copia de seguridad	Agenda Profesor Persona
Nº 7 => Cargar copia de seguridad	Agenda Profesor Persona