



**UNIVERSIDAD DE CÓRDOBA**



**Ingeniería del Software, Conocimiento y Bases de Datos**

**GRADO DE INGENIERÍA INFORMÁTICA**

**INGENIERÍA DEL SOFTWARE**

**ENTREGA FINAL DE PRÁCTICAS**

**Autor/es :**

**Cabrera Delgado, Manuel  
Fernández Urbano, Tomás  
Lucena Téllez, Sergio**

***Fecha 19/12/2018***

# ÍNDICE DE CONTENIDOS

---

1	<a href="#">DEFINICIÓN DEL PROBLEMA</a>	5
2	<a href="#">Extracción de requisitos</a>	6
3	<a href="#">Historias de usuario</a>	8
4	<a href="#">Casos de uso</a>	12
5	<a href="#">Diagramas de clase</a>	19
6	<a href="#">Diagramas de secuencia</a>	21
7	<a href="#">Metodología SCRUM</a>	29
7.1	<a href="#">Product backlog</a>	30
7.2	<a href="#">Sprint backlog</a>	34
7.3	<a href="#">Burndown chart</a>	36
8	<a href="#">Matrices de validación</a>	37
9	<a href="#">Bibliografía</a>	39

# ÍNDICE DE FIGURAS

---

<b>Figura 3.1.</b> Historia usuario logear	8
<b>Figura 3.2.</b> Historia usuario registrar	9
<b>Figura 3.3.</b> Historia usuario insertar	9
<b>Figura 3.4.</b> Historia usuario buscar	9
<b>Figura 3.5.</b> Historia usuario mostrar un alumno	10
<b>Figura 3.6.</b> Historia usuario mostrar listado	10
<b>Figura 3.7.</b> Historia usuario modificar	10
<b>Figura 3.8.</b> Historia usuario borrar	11
<b>Figura 3.9.</b> Historia usuario guardar	11
<b>Figura 3.10.</b> Historia usuario cargar	11
<b>Figura 4.1.</b> Caso de uso borrar	12
<b>Figura 4.2.</b> Caso de uso buscar	13
<b>Figura 4.3.</b> Caso de uso cargar	13
<b>Figura 4.4.</b> Caso de uso insertar	14
<b>Figura 4.5.</b> Caso de uso logear	14
<b>Figura 4.6.</b> Caso de uso modificar	15
<b>Figura 4.7.</b> Caso de uso mostrar listado	15
<b>Figura 4.8.</b> Caso de uso mostrar alumno	16
<b>Figura 4.9.</b> Caso de uso registrar	16

<b>Figura 4.10.</b> Caso de uso guardar	17
<b>Figura 5.1.</b> Diagrama de clases	20
<b>Figura 6.1.</b> Diagrama de secuencias de insertar	22
<b>Figura 6.2.</b> Diagrama de secuencias de borrar	23
<b>Figura 6.3.</b> Diagrama de secuencias de loguearse	23
<b>Figura 6.4.</b> Diagrama de secuencias de buscar	24
<b>Figura 6.5.</b> Diagrama de secuencias de cargar	24
<b>Figura 6.6.</b> Diagrama de secuencias de modificar	25
<b>Figura 6.7.</b> Diagrama de secuencias de guardar	26
<b>Figura 6.8.</b> Diagrama de secuencias de mostrar	26
<b>Figura 6.9.</b> Diagrama de secuencias de mostrar todos	27
<b>Figura 6.10.</b> Diagrama de secuencias de registrarse	28
<b>FIGURA 7.1.1</b> Product backlog prioridad 0	30
<b>FIGURA 7.1.2</b> Product backlog prioridad 1	30
<b>FIGURA 7.1.3</b> Product backlog prioridad 2	31
<b>FIGURA 7.1.4</b> Product backlog prioridad 3	31
<b>FIGURA 7.1.5</b> Product backlog prioridad 4	31
<b>FIGURA 7.1.6</b> Product backlog prioridad 5	32
<b>FIGURA 7.1.7</b> Product backlog prioridad 6	32
<b>FIGURA 7.1.8</b> Product backlog prioridad 7	32
<b>FIGURA 7.1.9</b> Product backlog prioridad 8	33
<b>FIGURA 7.2.1</b> Sprint backlog	34
<b>FIGURA 7.3.1</b> Product backlog	36

# ÍNDICE DE TABLAS

---

<b>Tabla 8.1</b> Matriz RF/CU	37
<b>Tabla 8.2</b> Matriz CU/Clases	38

# 1

## DEFINICIÓN DEL PROBLEMA

---

UcoAgend nace con la idea de hacer más llevable la gestión de una clase al profesorado ofreciendo diversas funcionalidades que facilitan el trabajar con la información de los alumnos de los cuales el profesor imparte clase. La jerarquía implementada dentro del propio sistema permite especificar el rol de todo profesor que se encuentra vinculado al sistema mostrando los servicios a los que pueda acceder según este. En caso de ser coordinador puede realizar copias de seguridad de la información del alumnado cuando él decida.

UcoAgend permite, además, que múltiples profesores puedan registrarse o iniciar sesión en el sistema y poder realizar cualquier operación con los datos de sus alumnos.

# 2

## EXTRACCIÓN DE REQUISITOS

---

Este apartado corresponde al proceso por el cual los clientes descubren, revelan y comprenden los requisitos que desean del sistema. Para este proceso se realizó una técnica de captura y análisis de requisitos, más concretamente, las entrevistas con el cliente. A continuación, exponemos los requisitos extraídos en la entrevista con el cliente.

**Requisitos funcionales:** requerimientos explícitos sobre las funcionalidades implementadas en el programa.

1. Insertar datos del usuario
2. Buscar un usuario mediante DNI, apellido o equipo al que pertenece
3. Modificar un usuario (previamente buscar el usuario)
4. Borrar usuario (previamente buscar el usuario)
5. Mostrar un usuario o varios usuarios, por orden (previamente buscar el usuario)
6. Guardado automático y manual
7. Cargar la copia de seguridad
8. Mostrar el listado completo de alumnos
9. Registrar un profesor en el sistema
10. Iniciar sesión como profesor en el sistema

**Requisitos no funcionalidades:** requerimientos sobre cómo debe realizarse el desarrollo del proyecto. Es decir, son restricciones de los servicios o funciones ofrecidas por el sistema.

1. Lenguaje de programación C++
2. Compatible con GNU/linux
3. Fácil de entender y visualmente agradable
4. Copias de seguridad en ficheros binarios
5. Formato de fichero markdown
6. Clase máxima de 150 alumnos
7. Solo un líder por grupo y grupos formados por tres alumnos
8. Orden alfabéticamente por apellido o nombre, de manera numérica del DNI sin importar letra o curso más alto en el que está matriculado, y todo lo anterior de manera ascendente o descendente
9. El profesor es el encargado de ejecutar las operaciones sobre los datos
10. El coordinador, además de hacer las operaciones de los profesores, se encarga de guardar/cargar los ficheros
11. Se genera un fichero binario tanto para las credenciales de cada profesor como para todos los datos de los alumnos



# 3

## HISTORIAS DE USUARIO

---

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos.

### (ANVERSO)

ID: 009 Logear un profesor

Como profesor quiero autenticarme en el sistema.  
Prioridad : 0

### (REVERSO)

- Controlar los accesos al sistema mediante la ID y contraseñas.
- Dichas credenciales se almacenarán en un fichero binario.

**Figura 3.1.** Historia usuario logear

**Anverso:** parte delantera de la historia de usuario donde se muestra.

1. Identificador de la funcionalidad descrita tanto numérico como literal.
2. Descripción en lenguaje común de la funcionalidad proporcionada por el cliente.
3. Prioridad de la función respecto a las demás implementaciones del sistema, en función de la prioridad se le destinarán más o menos recursos y tiempo respecto a las demás funcionalidades.

**Reverso:** traducción de las peticiones del usuario en un lenguaje más manejable para el desarrollador.

### (ANVERSO)

ID: 010 Registrar un profesor

Como profesor quiero registrarme en el sistema.  
Prioridad : 0

### (REVERSO)

- Añadir su ID y su contraseña al fichero binario de las credenciales.

**Figura 3.2.** Historia usuario registrar

### (ANVERSO)

ID: 001 Insertar datos de un alumn@

Como profesor quiero poder añadir un nuevo alumno con todos sus datos.  
Prioridad : 1  
Tiempo estimado : 2 horas

### (REVERSO)

- Quiero insertar todos los datos obligatorios sobre el alumno.
- Se debe mostrar si falta algún campo obligatorio por rellenar.

**Figura 3.3.** Historia usuario insertar

### (ANVERSO)

ID: 002 Buscar un alumn@

Como profesor quiero buscar un alumno mediante DNI, apellido o al equipo al que pertenece para ver su informa  
Prioridad : 2  
Tiempo estimado : 2 horas

< >

### (REVERSO)

- Quiero buscar cualquier alumno según su DNI, apellido o equipo.
- Se debe mostrar si es líder de grupo o no.

**Figura 3.4.** Historia usuario buscar

### (ANVERSO)

ID: 005 Mostrar un/a alumn@

Como usuario quiero ver los datos de un alumn@.  
Prioridad : 3  
Tiempo estimado : 1 hora

### (REVERSO)

- Quiero ver los datos de cualquier alumno.
- En caso de buscar por grupo, poder mostrar el listado ordenado de manera ascendente o descendente.

**Figura 3.5.** Historia usuario mostrar un alumno

### (ANVERSO)

ID: 008 Mostrar el listado de alumn@s

Como profesor quiero ver todos los datos de los alumn@s.  
Prioridad : 4  
Tiempo estimado : 1 hora

### (REVERSO)

- Quiero mostrar los datos de todos los alumnos existentes en el fichero.
- Poder mostrar el listado ordenado de manera alfabética por apellido o nombre, de manera numérica del DNI si

< >

**Figura 3.6.** Historia usuario mostrar listado

### 🔗 (ANVERSO)

ID: 003 Modificar datos de un alumn@

Como profesor quiero poder modificar algunos datos de un alumno ya existente.  
Prioridad : 5  
Tiempo estimado : 2 horas

### (REVERSO)

- Quiero poder modificar alguno de los datos sobre el alumno.
- No se permitirá la modificación si no se rellena algún dato obligatorio.

**Figura 3.7.** Historia usuario modificar

### (ANVERSO)

ID: 004 Borrar un alumn@

Como profesor quiero borrar un alumno.  
Prioridad : 6  
Tiempo estimado : 1 hora

### (REVERSO)

- Quiero buscar cualquier alumno según su DNI o apellido.
- Se debe advertir si es líder o no, para saber si algún equipo se queda sin líder.

**Figura 3.8.** Historia usuario borrar

### (ANVERSO)

ID: 006 Guardar copia de seguridad

Como profesor quiero tener la opción de guardar el fichero cuando yo quiera, pero también de manera automática.  
Prioridad : 7  
Tiempo estimado : 1 horas

< >

### (REVERSO)

- Quiero tener la opción de guardar manualmente.
- Se debe guardar automáticamente cada cierto tiempo por si se cierra inesperadamente el fichero.

**Figura 3.9.** Historia usuario guardar

### (ANVERSO)

ID: 007 Cargar copia de seguridad.

Como profesor quiero tener la opción de cargar el fichero que se guardó en caso de error o fallo en el sistema.  
Prioridad : 8  
Tiempo estimado : 1 hora

< >

### (REVERSO)

- Debo de tener un guardado previo de un fichero.

**Figura 3.10.** Historia usuario cargar

# 4

## CASOS DE USO

---

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas

### Borrar un alumn@

---

ID : 004

Breve descripción: Borrar del sistema la información de un alumn@.

Actores principales: Profesor.

Actores secundarios: Alumno.

Precondiciones:

1. Que exista el/la alumn@ que deseamos borrar.

Flujo principal:

1. El caso de uso comienza cuando el profesor quiere eliminar los datos un alumno.
2. Buscamos que exista el alumn@ a eliminar.
3. Eliminamos todos los datos pertenecientes al alumn@ del fichero.

Postcondiciones:

1. El fichero se modifica.
2. El alumn@ ya no existe en el fichero.
3. Si se elimina a un líder de equipo, se da la opción de elegir el otro.

Flujos alternativos:

2. a Si no existe el alumn@, se advertirá de ello.
3. a Si se elimina de forma incorrecta, el archivo no se modifica.

**Figura 4.1.** Caso de uso borrar

## Buscar un alumn@

---

ID : 002

**Breve descripción:** Buscar un alumno mediante DNI, apellido o equipo al que pertenece

**Actores principales:** Profesor.

**Actores secundarios:** Alumno.

**Precondiciones:**

1. Haber introducido previamente al alumn@

**Flujo principal:**

1. El caso de uso comienza cuando el profesor busca un alumn@
2. El profesor introduce el campo por el cual quiere buscarlo

**Postcondiciones:**

1. El sistema podrá realizar diversas operaciones sobre el alumn@

**Flujos alternativos:**

- 2.a Si no existe el alumno, se advertirá de ello

---

**Figura 4.2.** Caso de uso buscar

## Cargar un fichero

---

ID : 007

**Breve descripción:** Carga el fichero con la información de los alumnos.

**Actores principales:** Profesor.

**Actores secundarios:** Alumno.

**Precondiciones:**

1. Que exista un fichero.

**Flujo principal:**

1. Pedimos nombre del fichero a cargar.
2. Al abrir el sistema se cargan todos los datos que existían antes en el fichero.

**Postcondiciones:**

1. Ya está disponible toda la información sobre los alumn@s para el profesor.

**Flujos alternativos:**

- 2.a Si el archivo está corrupto no se cargará.

---

**Figura 4.3.** Caso de uso cargar

## Insertar datos alumno

ID : 001

**Breve descripción:** Introduce en el sistema la información de un alumno.

**Actores principales:** Profesor.

**Actores secundarios:** Alumno.

**Precondiciones:**

1. Buscar el alumno, por si previamente ya se ha añadido.

**Flujo principal:**

1. El caso de uso comienza cuando el profesor quiere introducir un alumno.
2. El sistema recoge los datos introducidos por teclado.

**Postcondiciones:**

1. Se escriben los datos del alumno en el fichero.

**Flujos alternativos:**

Si el alumno ya existe en la base, se advertirá y no se permitirá la insercción. 2.a Si no se rellenan campo obligatorios, se advierte al profesor.

**Figura 4.4.** Caso de uso insertar

## Logear un profesor

ID : 009

**Breve descripción:** El profesor introduce su ID y su contraseña para acceder al sistema.

**Actores principales:** Persona no identificada.

**Actores secundarios:** Ninguno.

**Precondiciones:**

1. El ID y la contraseña deben existir en las credenciales del sistema.

**Flujo principal:**

1. El caso de uso comienza cuando el profesor quiere acceder al sistema.
2. Introduce su ID y su contraseña.
3. Se comprueba su existencia.

**Postcondiciones:**

1. Se accede al sistema.

**Flujos alternativos:**

3. a Si no existe su ID y su contraseña en la credencial, se da la opción a registrarse.

**Figura 4.5.** Caso de uso logear

## Modificar alumn@

---

ID : 003

**Breve descripción:** Se modifican los datos de un alumn@.

**Actores principales:** Profesor.

**Actores secundarios:** Alumno.

**Precondiciones:**

1. Haber introducido previamente al alumn@.

🔗 **Flujo principal:**

1. Buscamos el alumno por DNI o apellidos para comprobar que existe.
2. Se modifican los datos deseados.
3. El sistema recoge los datos introducidos por teclado.

**Postcondiciones:**

1. Se sobrescriben los datos en el fichero.

**Flujos alternativos:**

1. a Si no existe el alumno, se advierte al profesor.
2. a Si el formato de los datos no es el correcto, los datos no se modificaran.
3. b Si quitamos el líder de un grupo, pedir un nuevo líder.

**Figura 4.6.** Caso de uso modificar

## 🔗 Mostrar listado Alumn@s

---

ID : 008

**Breve descripción:** Se muestran todos los alumn@s añadidos.

**Actores principales:** Profesor.

**Actores secundarios:** Alumno.

**Precondiciones:**

1. La existencia del fichero que contiene los datos.

**Flujo principal:**

1. Se carga el fichero.
2. Se creará un fichero markdown con los datos de los alumn@s.

**Postcondiciones:**

1. Se muestran los datos de los alumn@s por pantalla.

**Flujos alternativos:**

1. a Si no se carga el fichero se mostrará un error

**Figura 4.7.** Caso de uso mostrar listado



## Mostrar un/@s alumn@/s

---

ID : 005

**Breve descripción:** Mostrar los datos de un/@s alumn@/s.

**Actores principales:** Profesor.

**Actores secundarios:** Alumno.

**Precondiciones:**

1. Que existan l@s alumn@/s que deseamos mostrar.

**Flujo principal:**

1. El caso de uso comienza cuando el profesor quiere visualizar los datos de un alumn@/s.
2. Buscamos que exista el alumn@ a mostrar.
3. Se pregunta el orden de los datos que se van a mostrar.
4. Se creará un fichero markdown con los datos de los alumn@s.

**Postcondiciones:**

1. Se muestra por pantalla.

**Flujos alternativos:**

2. a Si no existe el alumn@, no se mostrará los datos.
3. a Si introducen un orden incorrecto, se volverá a pedir.

**Figura 4.8.** Caso de uso mostrar alumno

## Registrar un profesor

---

ID : 010

**Breve descripción:** El profesor añade su ID y su contraseña a la credencial para acceder al sistema.

**Actores principales:** Persona no identificada.

**Actores secundarios:** Ninguno.

**Precondiciones:**

1. El ID y la contraseña no existan en las credenciales del sistema.

**Flujo principal:**

1. El caso de uso comienza cuando el profesor quiere inscribirse al sistema.
2. Introduce su ID y su contraseña.
3. Se comprueba que la ID no exista.

**Postcondiciones:**

1. Se añade el profesor a la credencial.
2. Se comunica el registro con éxito.

**Flujos alternativos:**

3. Si existe la ID, se manda un mensaje de error.

**Figura 4.9.** Caso de uso registrar

## 🔗 Guardar un fichero

---

ID : 006

Breve descripción: Guardar la información de los alumnos en un fichero.

Actores principales: Profesor.

Actores secundarios: Alumno.

Precondiciones:

1. Haber introducido o modificado previamente a un alumn@.

Flujo principal:

1. El caso de uso comienza cuando el profesor quiere guardar la nueva información.
2. Se introduce el nombre del fichero a guardar.
3. Abrir fichero.
4. Añadir o modificar los datos de algún alumn@.
5. Cerrar fichero.

Postcondiciones:

1. Al abrir de nuevo el fichero se muestren bien las últimas modificaciones.

Flujos alternativos:

2.a Si el fichero existe se dará la opción de sobrescribir. 2.b Si no se desea sobrescribir el fichero, se cancela la operación de guardar.

---

**Figura 4.10.** Caso de uso guardar

Los componentes que conforman un caso de uso son:

1. **ID:** identificador numérico de la función, coincide con el identificador de los casos de uso.
2. **Actores:** se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, además de entidades abstractas, como el tiempo. (Jacobson, I., P. Jonsson, M. Christerson and G. Overgaard, Ingeniería de Software Orientada a Objetos - Un acercamiento a través de los casos de uso. Addison Wesley Longman, Upper Saddle River, N.J., 1992.)
3. **Precondiciones:** acciones que deben realizarse antes poder entrar en el flujo principal.

4. **Flujo principal:** acciones que se realizan en el sistema en caso de que el funcionamiento sea el correcto.
5. **Postcondiciones:** modificaciones en el sistema o el programa que ocurrirán tras la finalización correcto de la función.
6. **Flujos alternativos:** acciones que ocurrirán en el sistema o en el programa en caso de que el flujo principal no funcione correctamente.

# 5

## DIAGRAMA DE CLASES

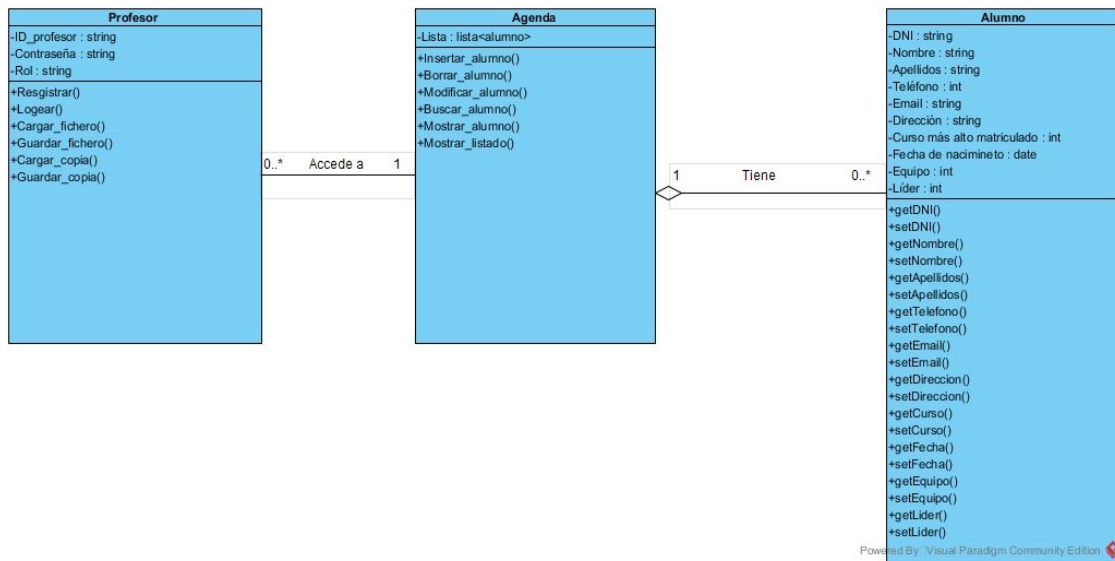
---

El diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, sus métodos y las relaciones entre objetos.

En el diagrama de clases podemos observar información adicional sobre los atributos o métodos, viendo su visibilidad. La visibilidad puede ser privada (-) o pública (+), si está privada significa que solo se puede acceder al miembro de la clase desde la propia clase, sin embargo, si está pública se puede acceder desde cualquier lugar.

Otra información que se puede observar a simple vista son las relaciones entre las clases del sistema. En nuestro diagrama de clase tenemos dos relaciones diferentes:

- **Asociación:** es una relación estructural que describe un conjunto de enlaces, las cuales representan conexiones a través de objeto.
- **Agregación:** es una clase especial de asociación que representa una relación de estructura entre un conjunto y sus partes. Esta relación significa "forma parte de", por ejemplo, la agenda está formada por uno o muchos alumnos.



**Figura 5.1** Diagrama de clases

En nuestro diagrama tenemos tres clases: profesor, agenda y alumno.

La clase profesor tiene tres atributos, los cuales son ID y contraseña, que se usarán para registrarse y loguearse, y rol que se usará para saber si es coordinador o profesor. Dependiendo del rol que tenga podrá hacer registrarse, loguearse, cargar y guardar copia o, además de todas las anteriores, cargar y guardar fichero.

La clase agenda tiene un solo atributo que es el nombre de la lista. Tiene 6 métodos a través de ellos se manipulará la información de los alumnos a nuestras necesidades.

La clase alumno tiene tantos datos como el cliente quiera almacenar, en nuestro caso son 10 campos los que el cliente quiere almacenar que son: DNI, nombre, apellidos, teléfono, email, dirección, fecha de nacimiento, curso más alto matriculado, equipo y si es líder o no de su equipo. En sus métodos están los observadores y los modificadores de cada campo del alumno.

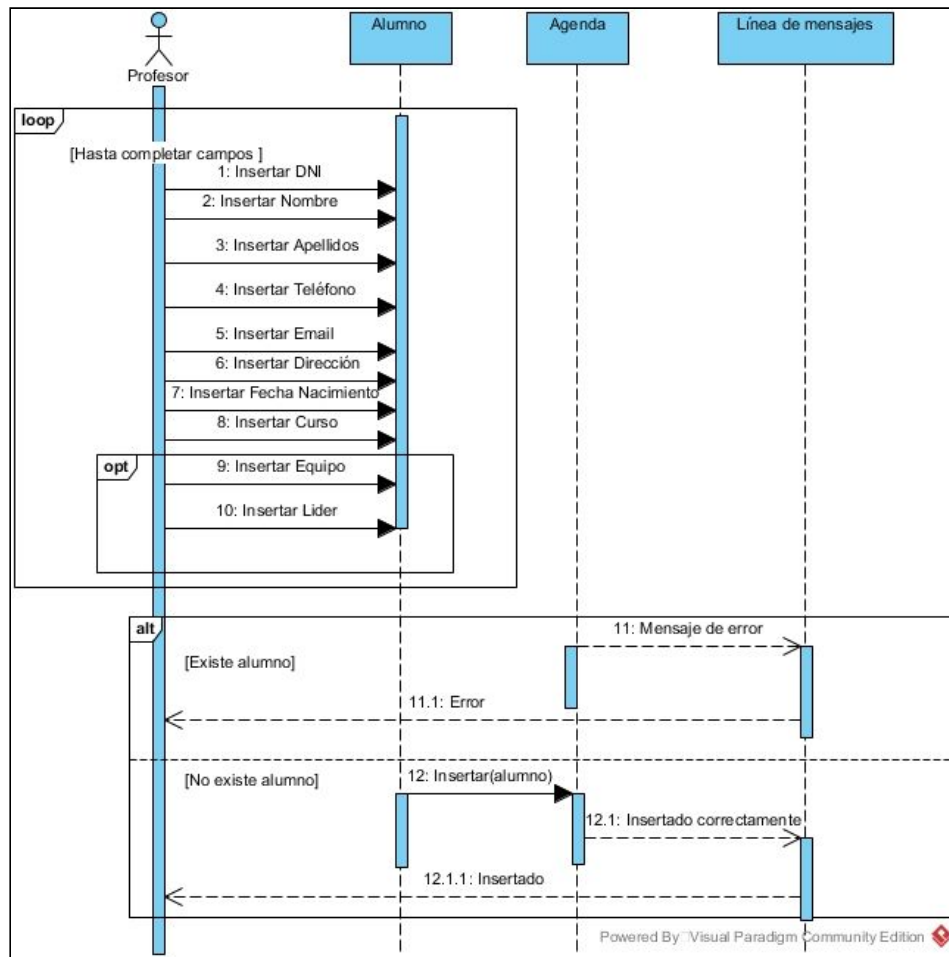
# 6

## DIAGRAMAS DE SECUENCIA

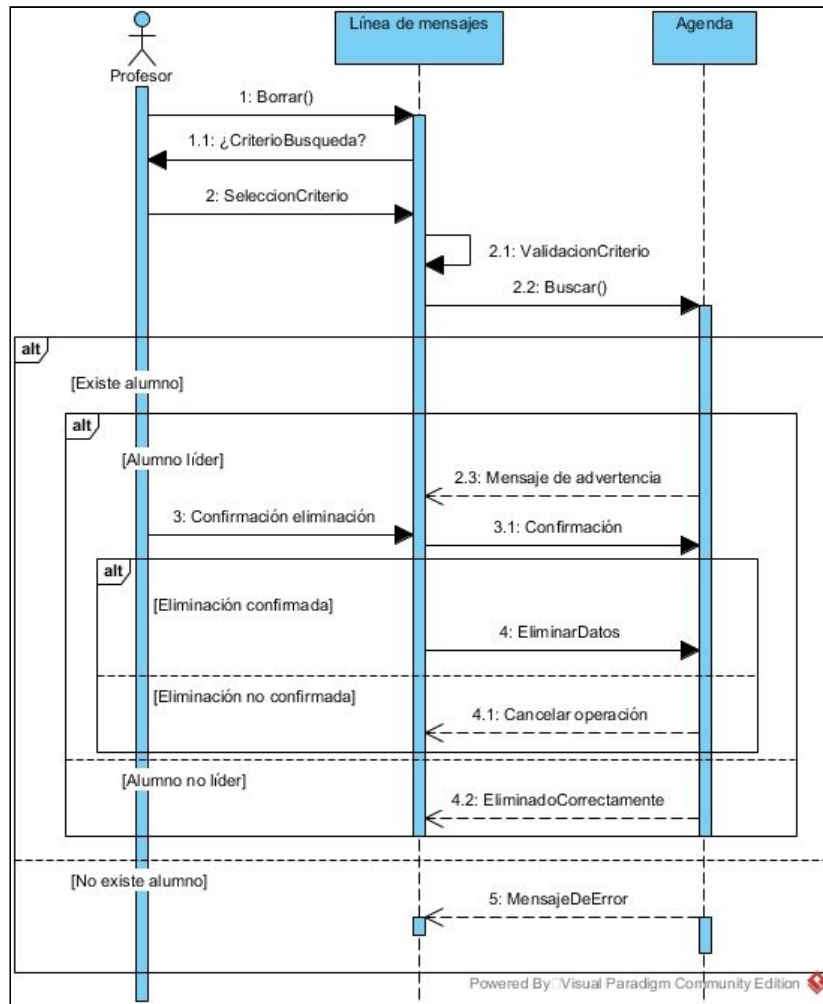
---

El diagrama de secuencia es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información haciendo énfasis en la secuencia de los mensajes intercambiados por los objetos. Se realiza un diagrama por cada caso de uso, ya que el diagrama de secuencia explica más detalladamente cómo se realiza la acción.

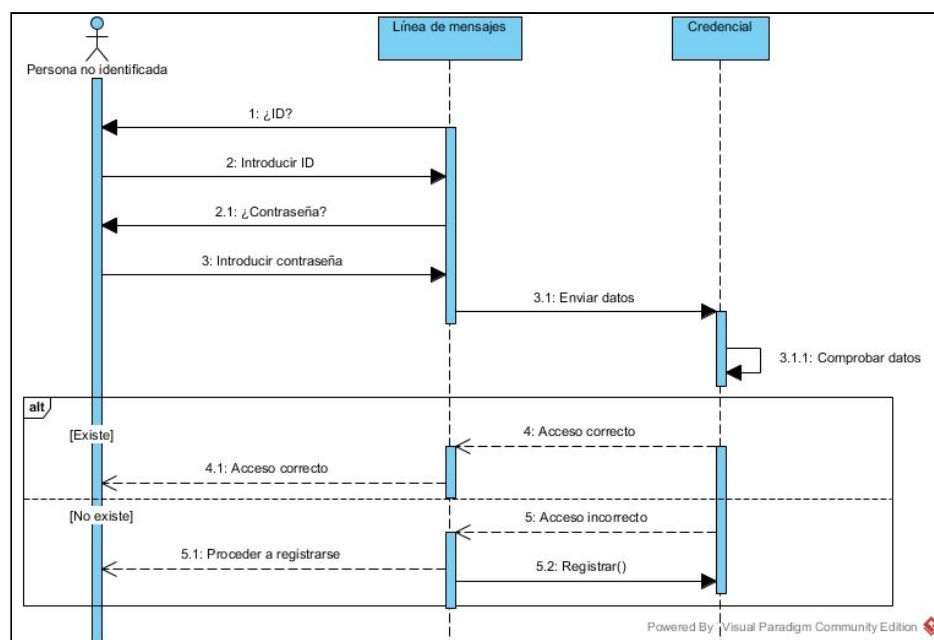
Un diagrama de secuencia tiene dos dimensiones, el eje vertical representa el tiempo y el eje horizontal los diferentes objetos. El tiempo avanza desde la parte superior del diagrama hacia la inferior. Cada objeto tiene asociados una línea de vida y focos de control. La línea de vida indica el intervalo de tiempo durante el que existe ese objeto.



**Figura 6.1.** Diagrama de secuencia de insertar

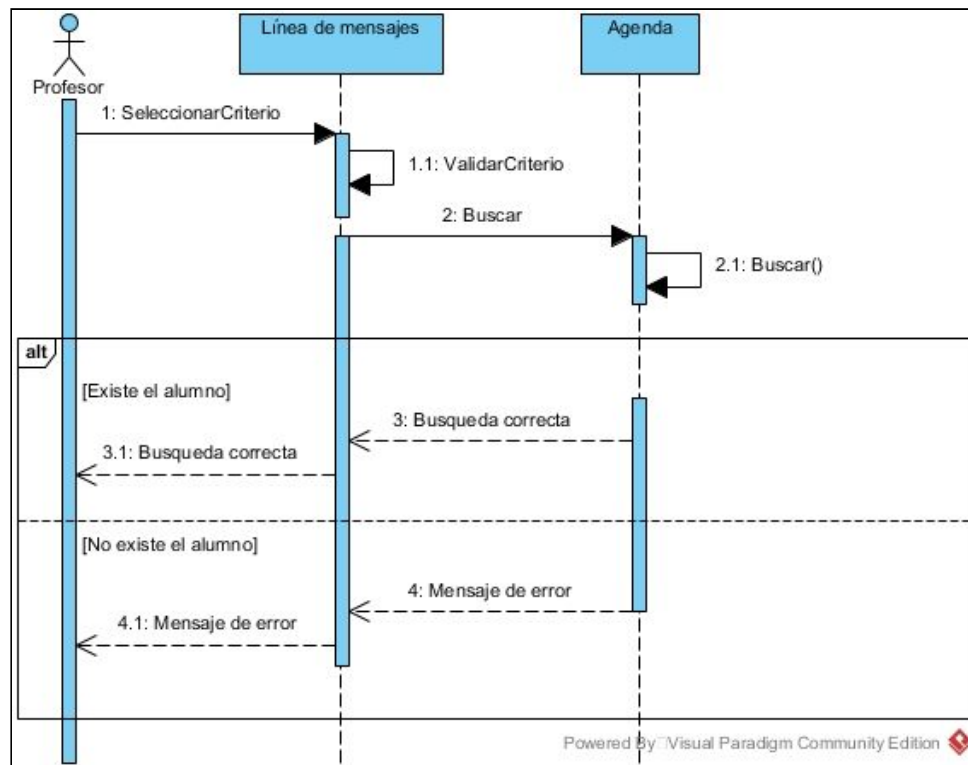


**Figura 6.2.** Diagrama de secuencia de borrar

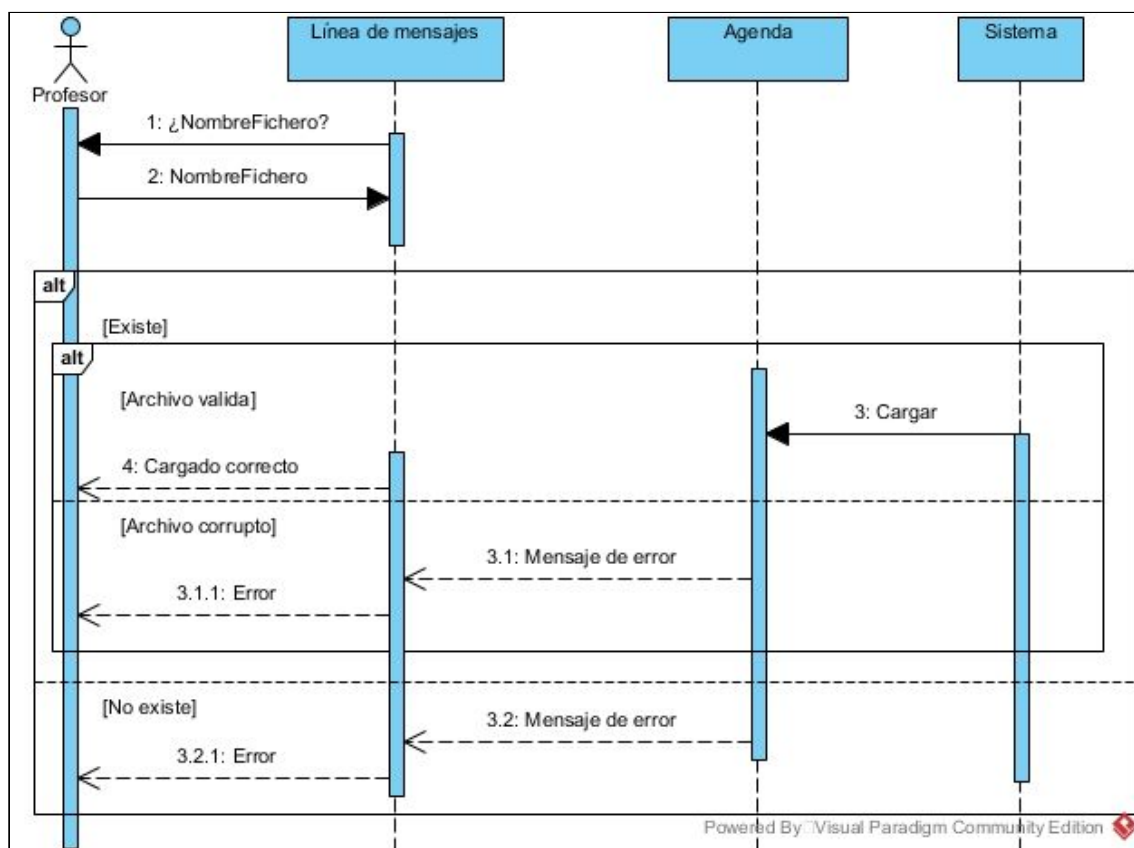


**Figura 6.3.** Diagrama de secuencia de loguearse

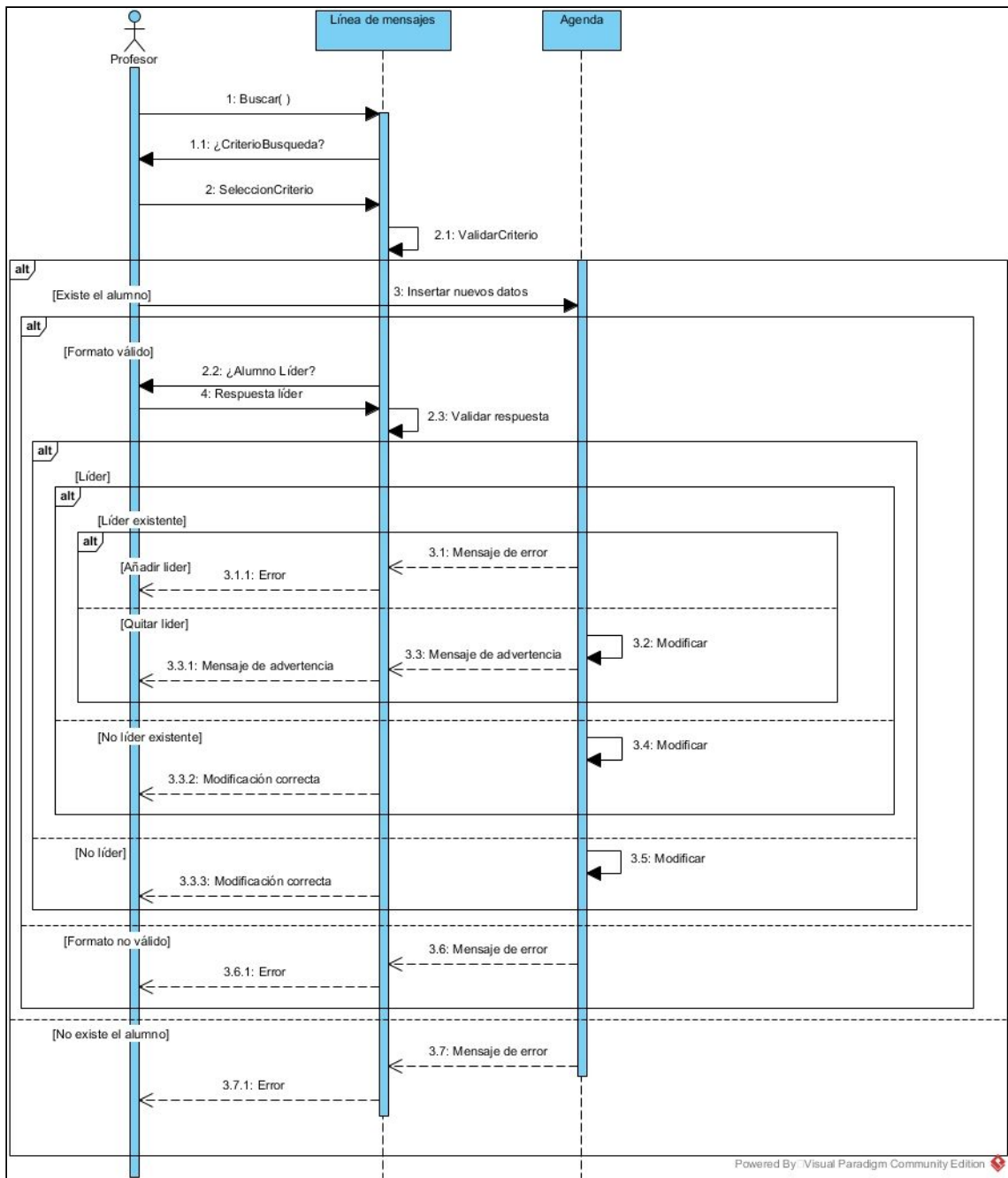




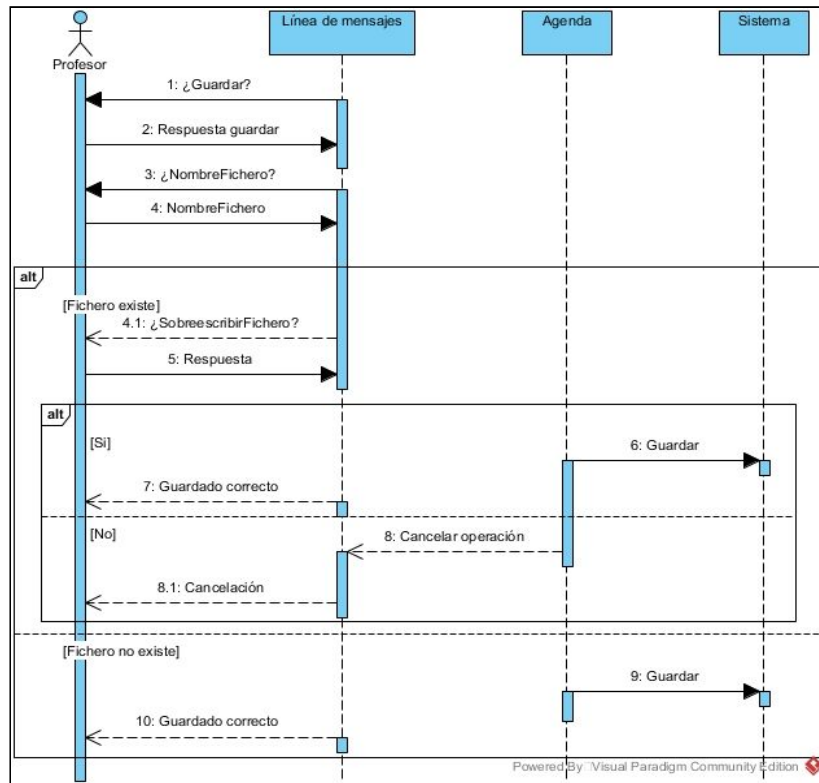
**Figura 6.4.** Diagrama de secuencia de buscar



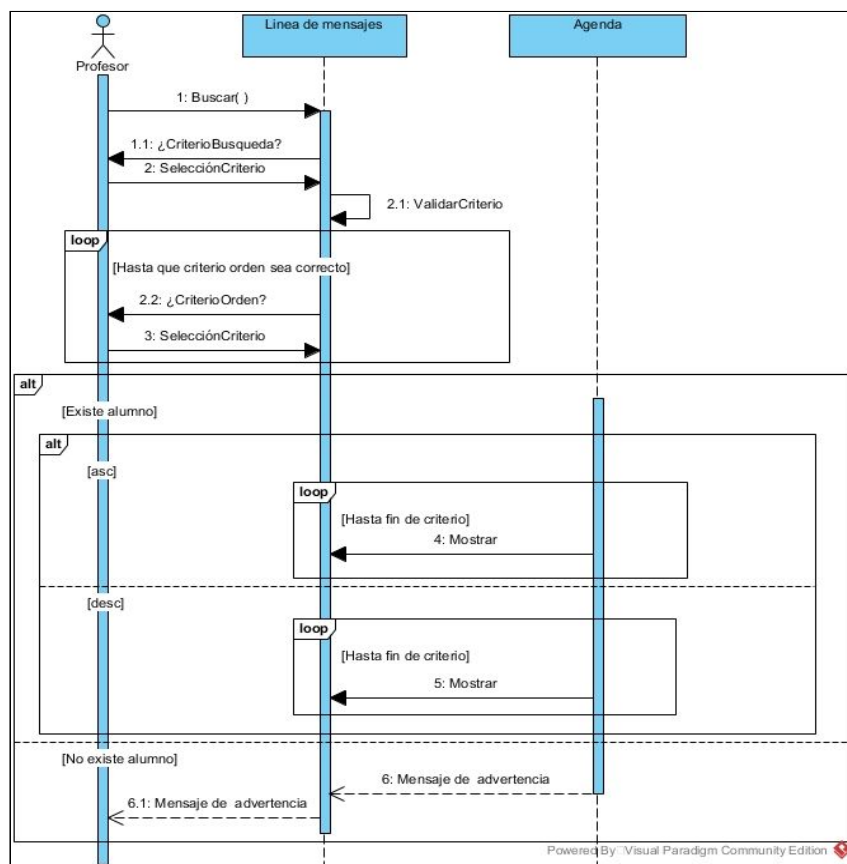
**Figura 6.5.** Diagrama de secuencia de cargar



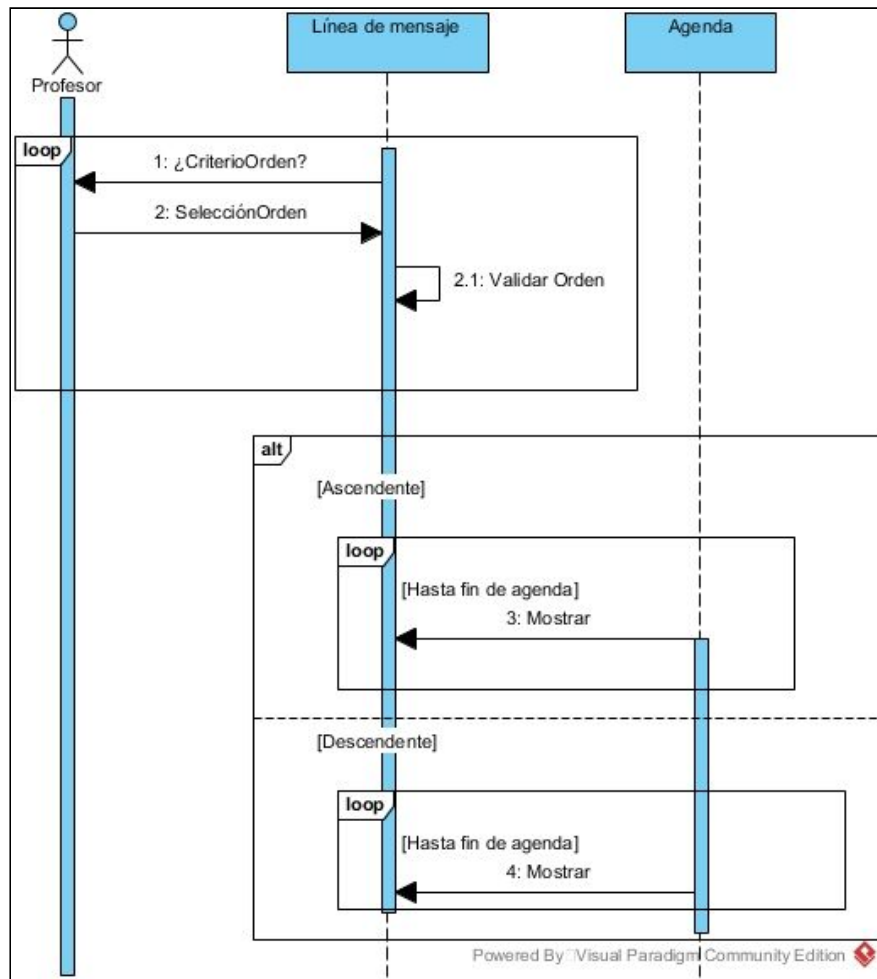
**Figura 6.6.** Diagrama de secuencia de modificar



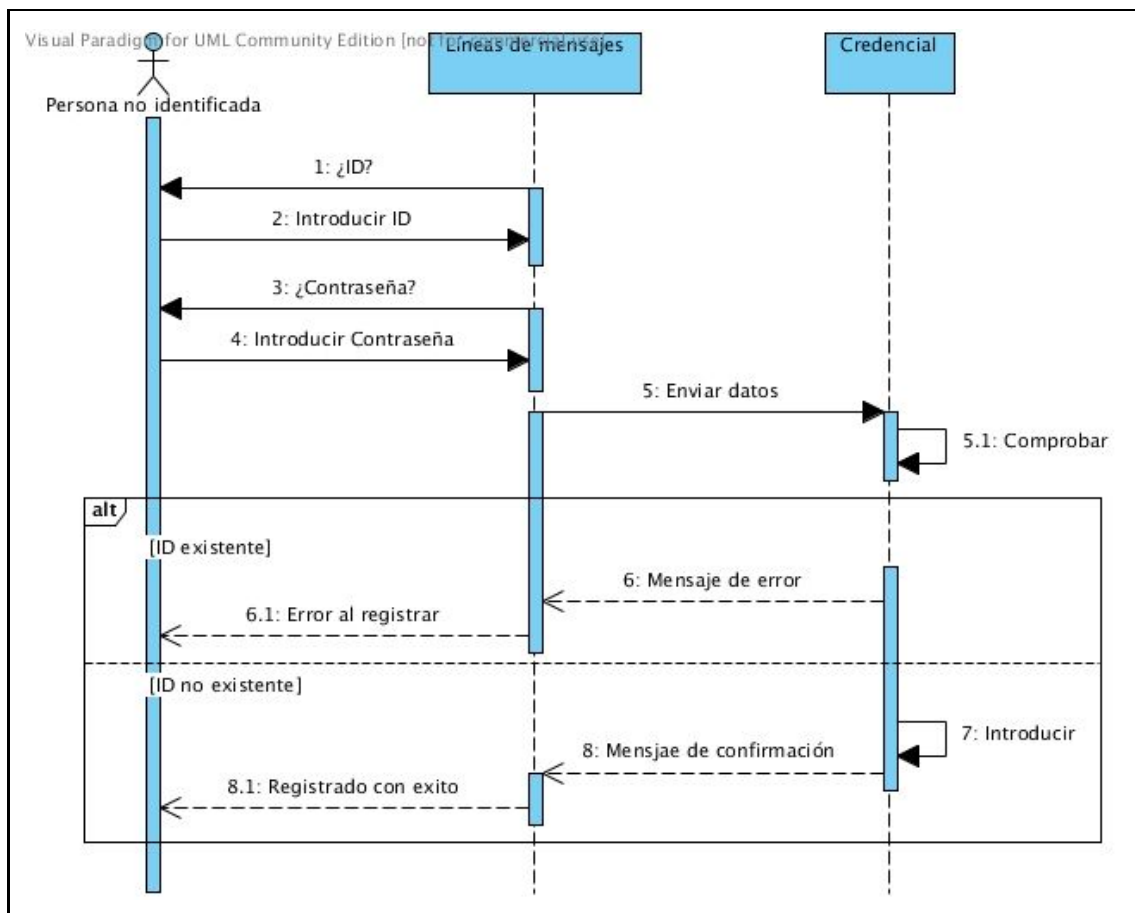
**Figura 6.7.** Diagrama de secuencia de guardar



**Figura 6.8.** Diagrama de secuencia de mostrar



**Figura 6.9.** Diagrama de secuencia de mostrar todos



**Figura 6.10.** Diagrama de secuencia de registrarse

# 7

## METODOLOGÍA SCRUM

---

La metodología SCRUM es un proceso ágil que permite centrarse en ofrecer el más alto valor de negocio en el menor tiempo posible. El negocio se encarga de fijar unas prioridades y los equipos se organizan con el fin de buscar la manera de entregar las funcionalidades de más alta prioridad en el menor tiempo posible.

En la metodología SCRUM el producto avanza en una serie de *sprints* de dos semanas, más o menos, de duración. Tras cada *sprint* cualquiera puede ver el software funcionar y decidir si liberarlo o seguir mejorándolo en otro *sprint*.

## 7.1 Product backlog

Un *product backlog* consiste en un documento que recoge todos los requisitos como elementos de una lista ordenadas por prioridades. En general, es la agrupación de todas las historias de usuario del producto.

### Product backlog

#### (ANVERSO)

ID: 009 Logear un profesor

Como profesor quiero autenticarme en el sistema.  
Prioridad : 0  
Tiempo estimado : 2

#### (REVERSO)

- Controlar los accesos al sistema mediante la ID y contraseñas.
- Dichas credenciales se almacenarán en un fichero binario.

#### (ANVERSO)

ID: 010 Registrar un profesor

Como profesor quiero registrarme en el sistema.  
Prioridad : 0  
Tiempo estimado : 2

#### (REVERSO)

- Añadir su ID y su contraseña al fichero binario de las credenciales.

**Figura 7.1.1.** Product backlog prioridad 0

#### (ANVERSO)

ID: 001 Insertar datos de un alumn@

Como profesor quiero poder añadir un nuevo alumno con todos sus datos.  
Prioridad : 1  
Tiempo estimado : 2 horas

#### (REVERSO)

- Quiero insertar todos los datos obligatorios sobre el alumno.
- Se debe mostrar si falta algún campo obligatorio por rellenar.

**Figura 7.1.2.** Product backlog prioridad 1

### (ANVERSO)

ID: 002 Buscar un alumn@

Como profesor quiero buscar un alumno mediante DNI, apellido o al equipo al que pertenece para ver su informa  
Prioridad : 2  
Tiempo estimado : 2 horas

< >

### (REVERSO)

- Quiero buscar cualquier alumno según su DNI, apellido o equipo.
- Se debe mostrar si es líder de grupo o no.

**Figura 7.1.3.** Product backlog prioridad 2

### (ANVERSO)

ID: 005 Mostrar un/a alumn@

Como usuario quiero ver los datos de un alumn@.  
Prioridad : 3  
Tiempo estimado : 1 hora

### (REVERSO)

- Quiero ver los datos de cualquier alumno.
- En caso de buscar por grupo, poder mostrar el listado ordenado de manera ascendente o descendente.

**Figura 7.1.4.** Product backlog prioridad 3

### (ANVERSO)

ID: 008 Mostrar el listado de alumn@s

Como profesor quiero ver todos los datos de los alumn@s.  
Prioridad : 4  
Tiempo estimado : 1 hora

### (REVERSO)

- Quiero mostrar los datos de todos los alumnos existentes en el fichero.
- Poder mostrar el listado ordenado de manera alfabética por apellido o nombre, de manera numérica del DNI si

< >

**Figura 7.1.5.** Product backlog prioridad 4



### (ANVERSO)

ID: 003 Modificar datos de un alumn@

Como profesor quiero poder modificar algunos datos de un alumno ya existente.  
Prioridad : 5  
Tiempo estimado : 2 horas

### (REVERSO)

- Quiero poder modificar alguno de los datos sobre el alumno.
- No se permitirá la modificación si no se rellena algún dato obligatorio.

**Figura 7.1.6.** Product backlog prioridad 5

### (ANVERSO)

ID: 004 Borrar un alumn@

Como profesor quiero borrar un alumno.  
Prioridad : 6  
Tiempo estimado : 1 hora

### (REVERSO)

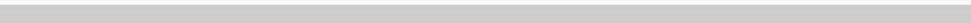
- Quiero buscar cualquier alumno según su DNI o apellido.
- Se debe advertir si es líder o no, para saber si algún equipo se queda sin líder.

**Figura 7.1.7.** Product backlog prioridad 6

### (ANVERSO)

ID: 006 Guardar copia de seguridad

Como profesor quiero tener la opción de guardar el fichero cuando yo quiera, pero también de manera automática.  
Prioridad : 7  
Tiempo estimado : 1 horas

<  >

### (REVERSO)

- Quiero tener la opción de guardar manualmente.
- Se debe guardar automáticamente cada cierto tiempo por si se cierra inesperadamente el fichero.

**Figura 7.1.8.** Product backlog prioridad 7

## (ANVERSO)

ID: 007 Cargar copia de seguridad.

Como profesor quiero tener la opción de cargar el fichero que se guardó en caso de error o fallo en el sistema  
Prioridad : 8  
Tiempo estimado : 1 hora



## (REVERSO)

• Debo de tener un guardado previo de un fichero.

**Figura 7.1.9.** Product backlog prioridad 8

## 7.2 Sprint backlog

Un *sprint backlog* consiste en un documento que recoge las funcionalidades a desarrollar en un *sprint* determinado. Además, quedará reflejado la división de trabajo entre los diferentes miembros del equipo. En nuestro caso, tendremos un documento con cada *sprint* desarrollado con la duración de **una semana** hasta la fecha que se nos indicó de entrega del producto.

### Sprint Backlog

Semana 26/11/2018-02/12/2018

#### División del trabajo

Tomás Fernández se encargará de realizar tanto el Sprint Backlog como el Product Backlog. Procederá a repartir

Manuel Cabrera se encargará de la funcionalidad de registrar un profesor.

Sergio Lucena se encargará de la funcionalidad de logear un profesor.

Tomás Fernández ayudará en el desarrollo de las funcionalidades.

< >

Semana 03/12/2018-09/12/2018

#### División del trabajo

Tomás Fernández se encargará de modificar el Sprint Backlog y repartir las funcionalidades restantes:

Manuel Cabrera se encargará de la funcionalidad de modificar, buscar, mostrar.

Sergio Lucena se encargará de la funcionalidad de borrar, insertar, mostrar listado.

Tomás Fernández ayudará a Manuel y Sergio a realizar las funcionalidades.

Semana 10/12/2018-16/12/2018

#### División del trabajo

Tomás Fernández se encargará de modificar el Sprint Backlog y repartir las funcionalidades restantes:

Manuel Cabrera se encargará de la funcionalidad de cargar y guardar ficheros.

Sergio Lucena se encargará de la funcionalidad de cargar y guardar copias.

Conjuntamente se realizara la implementación de la funcion principal main y de sus respectivas pruebas.

Figura 7.2.1 Sprint backlog

Como se muestra en el *sprint backlog* en el primer *sprint* se repartieron las funcionalidades de mayor prioridad del sistema entre ambos componentes del equipo. Mientras el *Scrum Master* se encargó de realizar la distribución del trabajo y la realización tanto del *sprint backlog* como del *product backlog*.

En el segundo *sprint*, el equipo al completo se encargó de realizar todas las funcionalidades de mayor prioridad consecutivas a las ya realizadas. De nuevo, el *Scrum Master* repartió el trabajo entre ambos componentes y ayudó en su realización y comprobación.

En el último *sprint*, el *Scrum Master* terminó de realizar tanto *product backlog* como *sprint backlog*. Finalmente, todo el equipo se encargó de terminar las funcionalidades de menor prioridad del sistema y las posteriores pruebas y correcciones del sistema.

### 7.3 Burndown charts

Un burndown chart consiste en un documento que se realiza o se modifica tras cada reunión que tenga el equipo conjuntamente. Es una representación gráfica del trabajo por hacer y realizado en un proyecto en el tiempo. En dicha gráfica, se representa en el eje vertical el trabajo restante por hacer y en el eje horizontal el tiempo.

En general, el diagrama representa una serie temporal del trabajo pendiente. Este diagrama es útil para predecir cuándo se completará todo el trabajo.

Reuniones	Horas de trabajo restantes
0	20
1	16
2	12
3	9
4	5
5	0



**Figura 7.3.1** Burndown chart

Observando la gráfica resultante podemos ver que el equipo durante el primer sprint, como ya hemos comentado antes, se encargó de las funcionalidades más prioritarias realizando las horas estimadas de trabajo de dichas funcionalidades. Durante las reuniones 2, 3 y 4 podemos ver que el equipo avanzó considerablemente sus horas de trabajo realizadas ya que se encargaron de completar la mayoría de funcionalidades que completan en sistema. Quedando la última reunión para la puesta en común de las pruebas y revisión del total de las funcionalidades implementadas.

# 8

## MATRICES DE VALIDACIÓN

### 8.1 MATRIZ RF - CU

Matriz en la que se muestran los requisitos funcionales frente a los casos de uso. Cada requisito funcional debe estar cubierto por al menos un caso de uso desarrollado. Con esto aseguramos que todas las funcionalidades requeridas en nuestro sistema son tenidas en cuenta.

	CU. 1	CU. 2	CU. 3	CU. 4	CU. 5	CU. 6	CU. 7	CU. 8	CU. 9	CU. 10
RF 1					✓					
RF 2		✓								
RF 3							✓			
RF 4	✓									
RF 5									✓	
RF 6				✓						
RF 7			✓							
RF 8								✓		
RF 9										✓
RF 10						✓				

**Tabla 8.1** Matriz RF/CU

## 8.2 MATRIZ CU - CLASES

Matriz en la que se muestra los casos de uso frente a las clases. Cada caso de uso debe tener asignado una clase al menos, en caso contrario, faltaría mejorar la definición de la clase, o la creación de otra.

	CU. 1	CU. 2	CU. 3	CU. 4	CU. 5	CU. 6	CU. 7	CU. 8	CU. 9	CU. 10
Profesor			✓	✓		✓				
Agenda	✓	✓			✓		✓	✓	✓	
Alumno					✓		✓			✓

**Tabla 8.2** Matriz CU/Clases



## BIBLIOGRAFÍA Y REFERENCIAS WEB

---

1. Wikipedia : [https://es.wikipedia.org/wiki/Scrum\\_\(desarrollo\\_de\\_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software))  
[https://es.wikipedia.org/wiki/Diagrama\\_de\\_clases](https://es.wikipedia.org/wiki/Diagrama_de_clases)
2. Jacobson, I., P. Jonsson, M. Christerson and G. Overgaard, Ingeniería de Software Orientada a Objetos - Un acercamiento a través de los casos de uso. Addison Wesley Longman, Upper Saddle River, N.J., 1992.
3. Daniel H. Steinberg and Daniel W. Palmer: *Extreme Software Engineering*, Pearson Education, Inc., ISBN 0-13-047381-2