



# Universidad de Valladolid

**E.U. DE INFORMÁTICA (SEGOVIA)**

**Grado en Ingeniería Informática  
de Servicios y Aplicaciones**

Sistema de Información del Grado en Ingeniería  
Informática de Servicios y Aplicaciones

(SIGIISA-UVa)

**Alumno:** Jose Carlos Sancho Sanz

**Tutor:** Fernando Díaz Gómez



**[SIGIISA-UVa]**

## **PRÓLOGO A LA MEMORIA**

*Dedicado con infinito cariño a mi Aurora y mi Ángeles  
Na iubes; foarte multi!*



**Universidad de Valladolid**



## UNAS REFLEXIONES PERSONALES

*Siendo éste un documento mayoritariamente técnico, no tendría sentido dedicarle un prólogo y darle así un tratamiento literario.*

*Pero me temo que nunca más tendré la oportunidad de dejar constancia por escrito de mis reflexiones personales sobre estos dos últimos años.*

*Así que aprovecho la ocasión que me brinda la posibilidad de que este documento permanezca para siempre colgado en Internet a disposición de todo el mundo para dejar unas líneas de carácter personal.*

*Comenzaré hablando sobre el Curso Puente, del cual tengo el recuerdo de haberlo podido realizar por casualidad, gracias a que trabajaba dos días por la tarde y dos por la mañana en lugar de tener turno de tarde semanal, como este año.*

*De este curso, me pareció realmente agradable tener a los profesores tan cercanos, en clases de 10-12 personas. Cuando uno tiene 20 años, mira a los profesores desde más abajo, y hacia arriba, como un niño pequeño. Y en general la impresión de ese curso fue poder tratar con ellos de igual a igual.*

*No quiero dejar una huella clasista en estas frases, sino simplemente decir que me pareció un clima muy amigable, y una forma muy cálida de retomar los estudios, aunque de un modo u otro, un informático nunca se desprende del todo de estas cosas, es necesario si no queremos quedar desactualizados y, de hecho, lo llevamos “integrado en placa” por nuestra forma de ser.*

*También los compañeros propiciaron ese clima, y la verdad es que fue un curso estupendo en lo humano, a pesar de las sombras de los recortes, y más allá de las ganas iniciales por aprobar el curso sin más, simplemente por obtener el Título de Graduado en Ingeniería Informática.*

*En ese sentido, el TFG me parece un colofón a este retorno a los estudios en el cual tenemos que demostrar que nos merecemos ese título, y esforzarnos para ello a lo largo de un verano entero, como es mi caso, o intentando compatibilizarlo con el trabajo, como será el de otros.*

*También en mi caso, el TFG ha coincidido en el tiempo con un cambio en el lugar de trabajo con horarios más difíciles y una triplemente adorable fractura de tibia y peroné, que han trastocado los planes iniciales por completo. Y aunque algunos momentos del Curso Puente fueron duros, y comprendí por primera vez la dificultad de compatibilizar el trabajo y los estudios, este año ha sido peor, porque nada era compatible con estar tumbado casi 24h al día durante días y días.*

## [SIGIISA-UVa] Prólogo a la Memoria

*Así pues, este TFG representa el esfuerzo puntual de unas semanas en las que por fin he podido tener dedicación exclusiva y completa, y también representa el reto personal de marcarme un objetivo y llevarlo a cabo mediante lo que más me gustó siempre de la Informática: la Programación.*

*También quería dejar unas palabras de despedida a la gente que he conocido a lo largo de estos años: alumnos y profesores, cuando he sido alumno; y más alumnos y más profesores, ahora que he cambiado de bando, además de personal de administración y servicios. He tenido la suerte de conocer a personas maravillosas entre todos ellos, incluso entre los que no parece que puedan aportar nada a tu vida.*

*Y ha sido en estos años como profesor de adultos en los que he descubierto que siempre hay alguien del que se puede aprender algo, siempre hay alguien que puede sorprenderte, y siempre hay alguien que puede hacerte reír.*

*Sinceramente, de todas las características que definen a una persona, la que me más he ido apreciando a lo largo de estos años es la de ser buena persona. Nunca me cansaré de conocer buenas personas, y nunca podré olvidarlas aunque pasen los años.*

*Un abrazo sincero para todos ellos y espero que la vida os trate bien.*

*Finalmente, quiero dejar escrito aquí el agradecimiento de todo corazón a las personas que me han acompañado en los malos ratos. Al final, tienen que ser los acontecimientos más graves los que le hacen a uno darse cuenta de que hay cosas más importantes en la vida que aquellas por las que luchamos con más empeño.*

*Así que no puedo despedirme de otra forma que deseando salud a todo el mundo y sobre todo a aquellas personas que son buenas personas.*



## INTRODUCCIÓN

En el siguiente documento se pretende presentar la aplicación desarrollada, comentar el origen y la motivación que llevó a hacer realidad este TFG, los objetivos del mismo, y detallar todas las características técnicas que permitan entender qué hace la aplicación y cómo lo hace.

**SIGIISA-UVa** proviene del acrónimo de “Sistema de Información del Grado en Ingeniería Informática de Servicios y Aplicaciones”, que es el título real del Trabajo Fin de Grado que se presenta aquí. De aquí en adelante se usará el acrónimo para identificar específicamente a la aplicación y el nombre largo para identificar al TFG en general.

Como descripción básica de la herramienta desarrollada, se puede decir que es una aplicación para Microsoft Access que permite gestionar los alumnos y profesores del Grado en Ingeniería Informática de Servicios y Aplicaciones, centrándose sobre todo en las cuestiones relativas a los tutores, las propuestas de TFG, los anteproyectos presentados por los alumnos, las asignaciones de TFG y su posterior seguimiento a lo largo del tiempo.

Finalmente, dado que esta aplicación se utilizará en un ambiente real de trabajo dentro de la Universidad, es especialmente necesario dejar indicaciones técnicas en esta memoria para la persona o personas que retomen el mantenimiento y la posible ampliación del trabajo presentado.

## VISIÓN GENERAL DEL DOCUMENTO

En este apartado se comenta la estructuración de la documentación, mediante una breve descripción de cada bloque de contenido.

La documentación física presentada consta de:

- Un documento impreso en papel y encuadernado.
- Un CD-ROM con la documentación en formato PDF y la aplicación SIGIISA-UVa para su uso y/o prueba por parte de los miembros del tribunal o alumnos interesados en este TFG.

**IMPORTANTE:** Dado que la base de datos de la aplicación contiene ciertos datos personales que podrían estar protegidos por la Ley Orgánica 15/1999 de 13 de diciembre, de Protección de Datos de Carácter Personal, la versión de la aplicación que se ha incluido en el CD-ROM no contiene dato alguno.

Por esto, para poder realizar pruebas con esta versión de la aplicación, deberán introducirse los datos manualmente a través de los formularios provistos por la propia aplicación.

Por otra parte, la documentación en sí se divide en tres grandes secciones:

- **Sección 1: Memoria del TFG**

En esta sección se detallan cuestiones generales sobre la aplicación, tales como la motivación que llevó a elegir este TFG, los objetivos que persigue, los criterios de desarrollo, las metodologías y tecnologías utilizadas, las conclusiones obtenidas,...

- **Sección 2: Manual técnico**

En esta sección se tratan de manera detallada aspectos sobre las etapas de análisis y diseño, los requisitos específicos, la funcionalidad implementada en forma de casos de uso, el diseño de la base de datos y las pruebas que se han realizado sobre la aplicación hasta llegar a su etapa final.

- **Sección 3: Manual de usuario**

En esta sección se explican los requisitos necesarios para poder ejecutar la aplicación, y su utilización por parte de un usuario sin formación específica a través de la interfaz de usuario que se ha creado a tal efecto.

## ÍNDICE DE CONTENIDOS

<b>PRÓLOGO A LA MEMORIA:</b> .....	<b>3</b>
Unas reflexiones personales .....	5
Introducción .....	7
Visión general del documento .....	7
<b>ÍNDICE DE CONTENIDOS:</b> .....	<b>9</b>
<b>SECCIÓN 1: MEMORIA DEL TFG</b> .....	<b>11</b>
1. Justificación .....	13
2. Ámbito de la aplicación .....	15
3. Objetivos .....	15
4. Cuestiones metodológicas .....	18
5. Herramientas empleadas .....	19
6. Planificación .....	20
6.1. Fases de trabajo .....	20
6.2. Estimación temporal .....	21
6.3. Estimación de costes .....	23
7. Implementación .....	24
7.1. Microsoft Access .....	24
7.2. Modo de trabajo con Access .....	26
7.3. Dificultades encontradas .....	28
8. Conclusiones .....	31
9. Posibles ampliaciones .....	33
10. Glosario de acrónimos .....	35
11. Referencias web y bibliografía .....	37
<b>SECCIÓN 2: MANUAL TÉCNICO</b> .....	<b>39</b>
1. Introducción .....	41
2. Especificación de Requisitos del Sistema .....	41
2.1. Requisitos funcionales .....	41
2.2. Requisitos no funcionales .....	44
3. Especificación de los casos de uso .....	45
3.1. Casos de uso de tipo 1: Alumnos .....	46
3.2. Casos de uso de tipo 2: Profesores .....	49
3.3. Casos de uso de tipo 3: TFGs .....	52
3.4. Casos de uso de tipo 4: Informes .....	56
4. Análisis y diseño de la base de datos .....	58
4.1. Esquema Entidad-Relación .....	59
4.2. Esquema Relacional .....	60
4.3. Diccionario de datos .....	62
5. Pruebas .....	68
5.1. Pruebas parciales .....	69
5.2. Pruebas de integración .....	71

<b>SECCIÓN 3: MANUAL DE USUARIO .....</b>	<b>73</b>
1. Requerimientos del sistema .....	75
2. Manual de instalación .....	76
3. Manual de uso .....	77
3.1. Preguntas de seguridad .....	77
3.2. Comportamientos comunes .....	78
3.3. Listado de formularios y sus funciones .....	79
3.3.1. Menús .....	79
3.3.2. Gestión de Alumnos .....	82
3.3.3. Gestión de Profesores .....	87
3.3.4. Gestión de TFGs .....	93
3.3.5. Informes .....	99
3.4. Índice de operaciones con formularios .....	100

**[SIGIIISA-UVa]**

**SECCIÓN 1:**

**MEMORIA DEL TFG**



**Universidad de Valladolid**



## 1. JUSTIFICACIÓN

En este apartado de la memoria se intentará explicar el porqué del TFG, es decir, las motivaciones que inician el proceso de desarrollo que lleva desde las etapas de planificación hasta la implementación definitiva de la aplicación.

Así, el primer aspecto a comentar sería el elemento inicial que desencadena el resto de pasos.

Inicialmente, mi gusto personal por Delphi y las aplicaciones de escritorio, me llevaba a la continuación de mi Proyecto Fin de Carrera (**WinCAC: Formato de Compresión Multicapa**), aunque no sabía en qué dirección exacta, dado que ese proyecto se orientó de forma educativa, más que como una herramienta en condiciones de competir con las de su categoría.

Otro campo que suscitaba mi interés personal era la codificación de formatos multimedia, sobre todo la de audio, con gran interés y cierto grado de conocimiento del codificador de audio de mayor calidad en la generación de archivos mp3 y más utilizado de las dos últimas décadas, a pesar del paso del tiempo: LAME.

Sin embargo, existen cientos de codecs multimedia, y diseñar uno nuevo requeriría de un gran esfuerzo, como en su día requirió poder realizar una compresión básica, apoyada en algoritmos de inventiva propia.

En ambos casos, la dedicación horaria excedería en mucho la que se pretende con un Trabajo Fin de Grado, por lo que busqué una propuesta de algún profesor que se acomodara mejor a esa carga de trabajo.

El tutor, Fernando Díaz, me explicó sus dos propuestas, en base a las necesidades personales de administración de los asuntos de la Universidad.

En su calidad de Coordinador de Titulación, actualmente necesita un sistema de información para gestionar la entrada de nuevos alumnos cada año, la plantilla de profesores, la asignación de tutores, el seguimiento de Trabajos Fin de Grado..., elementos todos ellos asociados con el nuevo Grado en Ingeniería Informática de Servicios y Aplicaciones.

El sistema de gestión actual consiste en hojas de cálculo de Microsoft Excel, que se referencian entre sí para mostrar la información deseada mediante funciones de texto y búsqueda de celdas mayoritariamente.

Este sistema le permite un seguimiento a un nivel aceptable, pero no es escalable para un mayor volumen de alumnos y profesores, ni a lo largo del tiempo, ni es cómodo a la hora de cambiar las funciones que transforman los datos, ni permite realizar cierto tipo de operaciones.

## [SIGIISA-UVa] Sección 1: Memoria del TFG

Además, es necesario tener una gran cantidad de redundancias en los datos, para identificar los objetos de los cuales se quiere obtener información.

Por esto, es necesaria una herramienta a medida que facilite el proceso de gestión de esa información, automatizando tareas, y a la vez, permitiendo a usuarios menos avanzados utilizar la herramienta.

En este caso, un auxiliar administrativo podría realizar labores básicas de gestión, como añadir alumnos y profesores, asignar tutores,...

La otra propuesta tenía relación con la gestión documental mediante DropBox, y el API que publicaron no mucho tiempo atrás.

Entre las dos, elegí la que se presenta en este TFG, dado que había trabajado anteriormente con Excel y Access en la Diputación Provincial de Segovia como soporte y desarrollador, y esto me podría dar un poco de ventaja inicial a la hora de llevarlo a cabo.

El trabajo, así, consistiría en cambiar el anterior sistema basado en hojas de cálculo de Excel por uno nuevo basado en una base de datos en Access, ajustándose lo mejor posible a la situación actual, pero al mismo tiempo facilitando el acceso a usuarios de menor formación.

Por tanto, este TFG no basa su desarrollo en algo ya creado, o previamente estudiado, sino que parte de una necesidad real, gestionada a través de un conjunto de datos más o menos descentralizado y sin interfaz de usuario, que hace casi imposible que sea utilizado por otras personas.

Y la solución a tal problema es la realización de una herramienta *ad-hoc*, en base a los requerimientos del tutor, que es el que tiene que reunir y tratar esos datos, y obtener la información para llevar a cabo su propio trabajo.

Finalmente, respecto a la vinculación de este Trabajo Fin de Grado con las competencias del Título de Graduado en Ingeniería Informática de Servicios y Aplicaciones, habría que decir que exceptuando las competencias centradas en Internet, redes, dispositivos móviles,... existe un claro enlace entre las competencias generales, específicas de la rama Informática, y de tecnología específica y las competencias necesarias para desarrollar este Trabajo Fin de Grado.

Sobre todo, habría que destacar las competencias:

- E.11, sobre los procedimientos algorítmicos
- E.12, sobre los tipos y estructuras de datos
- E.18, sobre los Sistemas de Información
- E.21, sobre los interfaces de usuario
- E.34, sobre la participación en todas las etapas de desarrollo

## 2. ÁMBITO DE LA APLICACIÓN

Dado que la aplicación desarrollada es una herramienta ad-hoc, su ámbito de aplicación se estrecha hasta tal punto de no tener utilidad más allá de aquella por la que ha sido diseñada.

Así pues, el ámbito de aplicación se aleja de lo que se consideraría uso personal, en el hogar, como podría ser cualquier aplicación de escritorio, y también del uso comercial, con objetivos de beneficio económico. Así, el espacio que puede ocupar SIGIISA-UVa se reduce únicamente al de un uso por parte de una organización, en este caso la Universidad de Valladolid, en el campus de Segovia, para gestionar una tarea muy concreta de forma específica para la propia organización.

Respecto al ámbito de despliegue, se puede decir que también será limitado, dado que se ejecutará en muy pocos equipos, o en uno solo, según el caso. Además, también se puede decir que estos equipos ejecutarán el sistema operativo Microsoft Windows, y se requerirá una versión Windows capaz de interpretar instrucciones de 32 bit. Además, la aplicación se ejecuta bajo Microsoft Office Access 2003, por lo que se requiere una versión de Office Professional igual o superior a la del año 2003.

La ejecución se hará de forma local, por lo que no se tendrá en cuenta la posibilidad de ejecución en red local, ya sea a través de un servidor, o directamente, como recurso compartido de red. Esta podría ser una necesidad en el futuro, y Access permite cierto nivel de concurrencia que facilitaría la ampliación de la aplicación sin demasiado trabajo adicional.

Finalmente... el tipo de los usuarios que accederán a la aplicación se limita a dos: el del Coordinador de Titulación y el del auxiliar administrativo, por lo que deben existir dos formas de trabajar con la aplicación que respondan a las necesidades y capacidades de cada perfil.

## 3. OBJETIVOS

Inicialmente se definieron una serie de objetivos, que se presentaron en el anteproyecto, y se comentan ahora en detalle. Pero antes de hacerlo, hay que partir del hecho de que el sistema actual está basado en hojas de cálculo, que contienen:

- Datos tabulados
- Fórmulas que actúan sobre estos datos mediante:
  - Funciones de concatenación
  - Funciones de decisión
  - Funciones de búsqueda por contenido de celda
- Referencias a celdas de otras hojas u otros libros

Así, los objetivos propuestos para la aplicación se basan en facilitar la inclusión de nuevos datos en el sistema o la obtención de información a partir de ellos, sin necesidad de tener elevados conocimientos de hojas de cálculo, ni tener especiales precauciones con su tratamiento.

Los objetivos propuestos inicialmente son:

- **Desarrollar una interfaz de gestión de fácil manejo a medida para este sistema de información concreto**

A través de este objetivo, un usuario puede interactuar con el sistema de información sin una cualificación especial ni muchos conocimientos previos sobre el sistema actual.

Es decir, este objetivo pretende elevar la accesibilidad del sistema de información a casi cualquier tipo de usuario, y disminuir el nivel de formación necesario para poder trabajar con el.

- **Gestionar las altas, las bajas,... de alumnos**

Este objetivo está centrado en la gestión de alumnos, de tal forma que el sistema debe permitir añadir, consultar y modificar los datos relevantes de los alumnos del Grado en Informática, que como se comprobará más adelante, es un conjunto de datos mínimo.

Habría que reseñar que las bajas no se contemplan como la eliminación del alumno del sistema, sino que simplemente se considera que el alumno no está activo.

- **Gestionar las altas, las bajas, asignación de departamentos,... de profesores**

A través de este objetivo, se debe permitir la gestión de departamentos, al igual que la de alumnos comentada anteriormente.

Actualmente, este apartado es bastante estático, por lo que no merece mención específica.

- **Gestionar la asignación de tutores a los nuevos alumnos del Grado**

A los nuevos alumnos del Grado se les asigna un tutor personal, por lo que al inicio de cada curso se deben definir nuevos grupos de tutoría y asignárselos a los alumnos.

Este objetivo se terminó incluyendo dentro de la gestión de alumnos.

- **Gestionar los aspectos de los TFG (propuestas de los tutores, asignación de tutor,...)**

Alrededor de los TFG hay una serie de cuestiones que se deben tener en cuenta y se debe ofrecer la posibilidad de gestionarlas.

Un alumno tiene la posibilidad de realizar un TFG mediante un anteproyecto personal que traslada a un profesor para que se convierta en su tutor, o aceptando una oferta propuesta por un profesor, que obviamente, será su tutor.

Este objetivo es el que define las posibilidades de gestión sobre estos supuestos, además del hecho concreto de la asignación de un TFG al alumno, que se produce al presentar un anteproyecto propio o aceptar y presentar una propuesta.

- **Obtener informes con la información relevante en cada momento**

La posibilidad de Access de generar informes con consultar personalizadas es la que llevó a incluir este objetivo.

Los formularios de acceso a los datos ofrecen usualmente información sobre un único objeto a la vez, y hay que cambiar de objeto para obtener más información de la misma categoría.

A través de los informes se obtienen listados con la información obtenida sobre varios objetos a la vez, y filtrada de acuerdo a los criterios que se necesiten.

- **Exportar datos en bruto con el fin de elaborar nueva información a partir de ellos**

Mediante este objetivo se pretendía que el sistema pudiera obtener información sin tratamiento alguno en la presentación, pero que fuera fácilmente importable por otras aplicaciones.

Por ejemplo, que existiera la posibilidad de exportar en bruto una lista de alumnos con TFG defendido, sin ningún tipo de formato. Luego, en Excel se podría importar esa lista para tratar la información de la forma deseada, como podría ser agrupar los alumnos por curso, utilizar un formato distinto para cada curso, y generar un gráfico de sectores que indique el porcentaje de cada nota (suspensos, aprobados, notables, sobresalientes y matrículas de honor).

Este objetivo supone la capacidad de externalizar los datos del sistema de información desarrollado, para realizar otro tipo de tratamientos en programas con otras funcionalidades.

Por otra parte, también existen objetivos implícitos, como el hecho de obtener un sistema que pueda ser usado por los dos perfiles comentados en el ámbito de aplicación, cada uno con una visibilidad muy distinta del sistema de información:

- El Coordinador de Titulación, que podrá acceder directamente a los datos y crear consultas e informes por sí mismo
- El auxiliar administrativo, que sólo tendrá acceso a la interfaz gráfica

Finalmente, existe el objetivo de la amigabilidad en el entorno de trabajo, que está orientado sobre todo al segundo perfil de los anteriormente comentados. Ese objetivo persigue facilitar el trabajo con el sistema, disminuir la complejidad de las operaciones a realizar con el, y reducir el riesgo de fallos humanos.

## 4. CUESTIONES METODOLÓGICAS

En este apartado, debería hablarse del tipo de modelo de desarrollo que se ha seguido y su ciclo de vida.

Sin embargo, no se ha seguido ningún modelo estándar, dado que toda la funcionalidad requerida por la aplicación estaba previamente determinada y ofrecida por las hojas de cálculo en Excel de las que partía el problema.

Por tanto, la etapa de análisis se limitaba esencialmente a entrevistas con preguntas puntuales para saber la forma de actuar en algunos casos, o conocer todas las posibilidades a contemplar en ciertas circunstancias.

Los requisitos funcionales derivan precisamente de la funcionalidad del sistema anterior, y por tanto, vienen dados de antemano.

Por otra parte, el sistema de información se iba a crear como una aplicación de Microsoft Access, por lo que no puede contemplarse, como en otros casos ni una arquitectura cliente-servidor ni un modelo de implementación orientado a objetos, basado en una jerarquía de clases. La atención en Access queda focalizada sobre los datos, almacenados en tablas, y sobre esas tablas se construyen los formularios, consultas e informes.

En cuanto a las tecnologías utilizadas con respecto a la metodología, habría que hacer especial mención a las herramientas CASE para los modelos Entidad/Relación y Relacional, pero a un nivel de simple dibujo, ya que en Access la creación de las tablas tiene que ser manual, no se puede automatizar a través de un diseño previo y la ejecución de un script.

Y respecto a los lenguajes de programación, en Microsoft Access se usa VBA (Visual Basic for Applications), que es específico para algunas aplicaciones de Microsoft Office, y también se usa SQL para las consultas.

## 5. HERRAMIENTAS EMPLEADAS

No se ha requerido la utilización de muchos programas en la realización de este TFG, ya que generalmente, todo está centrado en Access.

Las herramientas utilizadas son:

- **brModelo v2.0**: Se ha utilizado como herramienta CASE para crear los diagramas de E/R y relacional. Es un programa de funcionalidad minimalista, ya que sólo permite dibujar ese tipo de diagramas. Se desarrolló en Delphi y se distribuye de forma gratuita con licencia *freeware* para los estudiantes de informática brasileños.

La razón para elegir este programa ha sido un simple criterio de limpieza, ya que se puede ejecutar sin instalar nada, y se compone de un único archivo ejecutable que deja un rastro mínimo en el sistema.

<http://sis4.com/brModelo>

- **Microsoft Paint**: Es una herramienta muy utilizada a pesar de su sencillez, y viene incluida en cualquier sistema operativo de la familia Windows. Se ha utilizado para la generación de recursos gráficos sencillos, y tratamiento de otros.
- **Microsoft Access 2003**: Es un programa para la gestión básica de bases de datos, incluido sólo en la edición Professional y superiores de Microsoft Office.

Access puede facilitar muchísimo el desarrollo de una aplicación básica para un usuario único, ya que a través de un asistente y en pocos pasos, se pueden crear formularios con formato estándar y controles asociados para acceder a datos que provienen de tablas o consultas. Pero sin programación, sólo tiene un comportamiento genérico, y no realiza validaciones sobre lo que se introduce, más allá de comprobación de tipos, y otras reglas que se añadan; y además guarda todos los datos que se introduzcan, sin esperar a que sea el usuario el que decida cuándo quiere guardarlos.

Access se ha utilizado de tres formas distintas:

- Modo "*diseñador*": Se ha usado este modo en la etapa de construcción de las tablas (modelo lógico) mediante ratón y teclado, para el diseño de la interfaz y de algunas consultas.
- Modo "*gestor de BD*": Algunas consultas se han escrito por completo en SQL, para pruebas o posterior escritura de código, y es el motor interno de Access el que las interpreta y ejecuta.
- Modo "*programador*": Se ha usado este modo en la etapa de implementación para definir el comportamiento de la aplicación en respuesta a los eventos que se produzcan, mediante código en lenguaje VBA.

Hay que recalcar que el lenguaje VBA no es un lenguaje orientado a objetos, que permita crear clases, tener herencia,...

El VBA es un lenguaje orientado a eventos y que usa objetos, que son predefinidos y no se pueden extender. Esto se acomoda a los programas del paquete Office, ya que los objetos predefinidos son precisamente los objetos de Office (líneas, párrafos, celdas, filas y columnas, tablas, formularios, controles,...).

Sin embargo, el VBA está presente en aplicaciones externas a Microsoft Office, como Project o Visio, e incluso en programas que no son de Microsoft, como CorelDRAW o AutoCAD.

Finalmente, Access permite el acceso concurrente, por lo que se podrían usar sus bases de datos para aplicaciones multiusuario y en red local. Sin embargo, utiliza bloqueos bastante estrictos, por lo que a veces no es posible el acceso simultáneo a ciertos datos.

Una solución a esto es separar el código y la interfaz de la base de datos, de forma que se puede crear en Visual Basic la capa de presentación (los formularios), agregarles individualmente la capa de lógica de negocio mediante programación y mantener la capa de acceso a datos (las tablas) en Access.

Pero si el sistema crece, y se requiere un sistema gestor de base de datos profesional, se requeriría una migración a SQL Server, que realizándose desde Access, es un proceso relativamente sencillo.

## 6. PLANIFICACIÓN

En este apartado, se deberían comentar las fases en las que se divide el Trabajo Fin de Grado, aproximar el tiempo de desarrollo, y finalmente calcular un presupuesto.

### **6.1. Fases de trabajo**

Inicialmente, en el anteproyecto, se establecieron las siguientes fases:

0. Documentación
1. Planificación
2. Análisis de requisitos
3. Especificación de Requisitos del Sistema
4. Diseño del modelo de datos
5. Diseño de los subsistemas
6. Implementación y pruebas
7. Pruebas de integración

La fase de documentación se inicia al principio del proceso, y se extiende a lo largo de todo el desarrollo, a diferencia de las demás fases, que en general se realizan de forma excluyente, y no en paralelo.

Además, cada fase ya tenía una estimación temporal preliminar, como se verá a continuación.

## 6.2. Estimación temporal

El TFG se iba a comenzar en Marzo de 2012 y se iba a entregar a principios de Septiembre de 2012, compatibilizándolo con el Curso Puente, por lo que la duración inicial era de 5 meses y medio, y la estimación temporal de las fases se ajustaba a ese periodo.

Además, se distinguieron dos etapas diferenciadas, una en la cual la carga horaria sería baja, durante el curso escolar; y otra en las vacaciones, con una carga alta de trabajo, como la de una jornada laboral.

El diagrama de Gantt que se presentó en el anteproyecto era este:

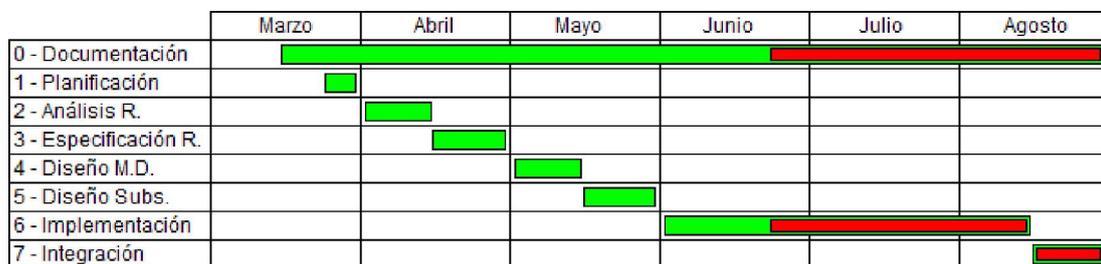


Figura 6A: Estimación temporal inicial

Sin embargo, la estimación temporal debe realizarse por criterios menos subjetivos, y usualmente se utiliza el modelo de estimación COCOMO, basado en datos estadísticos, y que en función del tamaño del software y una serie de factores adicionales, permite obtener una primera estimación mediante el modelo básico, y luego refinarla con el modelo intermedio.

### Estimación mediante el modelo básico de COCOMO

Sirve para evaluar de forma preliminar el esfuerzo necesario para llevar a cabo el desarrollo, a partir de una estimación del número de líneas de código (KLDC), y en base a tres entornos de desarrollo distintos:

Modo de desarrollo	Esfuerzo en personas-mes	Tiempo de desarrollo
Orgánico	PM=3,2 KLDC <sup>1,05</sup>	TD=2,5 PM <sup>0,38</sup>
Empotrado	PM=3,0 KLDC <sup>1,12</sup>	TD=2,5 PM <sup>0,35</sup>
Semi-libre	PM=2,8 KLDC <sup>1,2</sup>	TD=2,5 PM <sup>0,32</sup>

Tabla 6B: Ecuaciones de COCOMO que se aplican en cada entorno de desarrollo

El entorno típico para el desarrollo es el semi-libre, dado que el orgánico se relaciona con entornos sin presiones en el tiempo, y el empotrado en entornos con requisitos muy restrictivos y gran volatilidad.

## [SIGIISA-UVa] Sección 1: Memoria del TFG

Por tanto, con el modelo semi-libre, y partiendo de una aproximación inicial de 2000 líneas de código, puesto que en Access gran parte del trabajo se realiza mediante diseño, obtenemos un esfuerzo nominal de:

$$PM=2,8 \times 2^{1,2} = 6,43 \text{ personas-mes}$$

Y un tiempo de desarrollo asociado de:

$$TD=2,5 \times 6,43^{0,32} = 4,53 \text{ meses}$$

Estos son valores nominales, que se pueden refinar a posteriori.

### Estimación mediante el modelo intermedio de COCOMO

Una vez que se ha identificado los principales componentes del sistema, se puede estimar el tiempo mediante este método, que consiste en aplicar correcciones al esfuerzo nominal obtenido anteriormente con factores de ajuste que vienen dados por 15 atributos:

Atributos	Valor					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos de software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	
Tamaño de Base de datos		0,94	1,00	1,08	1,16	
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria virtual			1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1,00	1,15	1,30	
Tiempo de respuesta		0,87	1,00	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	
Calidad de los programadores	1,42	1,17	1,00	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90		
Experiencia en el lenguaje	1,14	1,07	1,00	0,95		
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,10	1,00	0,91	0,82	
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83	
Restricciones de tiempo de desarrollo	1,22	1,08	1,00	1,04	1,10	

Tabla 6C: Factores de ajuste para el COCOMO intermedio

En este caso, los factores que se han considerado son:

- Tamaño de la base de datos: Bajo (0,94)
- Capacidad de análisis: Alta (0,86)
- Experiencia en la aplicación: Baja (1,13)
- Calidad de los programadores: Alta (0,86)
- Restricciones de tiempo de desarrollo: Alto (1,04)

Ahora se refina el cálculo del esfuerzo y el tiempo de desarrollo:

$$\text{Esfuerzo (PM)} = 6,43 \text{ (esfuerzo nominal)} \times 0,94 \times 0,86 \times 1,13 \times 0,86 \times 1,04 = 5,25 \text{ personas-mes}$$

$$TD = 2,5 \times 5,25^{0,32} = 4,25 \text{ meses}$$

### **6.3. Estimación de costes**

Los costes se van a dividir en costes de personal y de material.

#### **Costes de personal**

A partir del dato del esfuerzo obtenido por el COCOMO intermedio, se puede estimar un valor para el coste de personal. Suponiendo que los programadores son de calidad alta, como se especificó en los factores de ajuste, el sueldo de cada uno se establece en 1500€ brutos. Por tanto:

$$\text{Coste según COCOMO} = 5,25 \times 1500 = 7875€ + \text{IVA} = 9529€$$

Sin embargo, se podría mirar desde el punto de vista de que el desarrollo es de un TFG y lo va a realizar una persona, durante el tiempo especificado por los créditos ECTS y recomendado por la normativa (300-360h), por lo que podríamos establecer un sueldo de 20€/hora y obtener un coste de personal de 6000-7200€ + IVA.

Se tomará como referencia el coste y el tiempo según COCOMO, partiendo del supuesto de que éste fuera un proyecto que se encarga a un estudio.

#### **Costes de material**

A continuación se establecerán los elementos necesarios para el desarrollo de la aplicación, dado que el entorno de trabajo ya cuenta con los elementos necesarios para el despliegue.

Concepto	Importe
Ordenador portátil (gama media-baja)	450€
Windows XP Professional	300€
Microsoft Office 2003 Professional (es necesario Access)	360€
brModelo	-
Microsoft Paint	-

Tabla 6D: Costes de material

Los precios expuestos incluyen IVA, y se refieren al momento en el que el producto estaba en venta (hoy en día no se puede comprar Office 2003).

El presupuesto final en base al modelo de COCOMO será de:

$$\text{Coste final} = 9529 + 450 + 300 + 360 = 10639€$$

## 7. IMPLEMENTACIÓN

En este apartado se va a comentar cómo funciona Access, sus ventajas e inconvenientes, el modo de trabajo que se ha seguido, y las dificultades encontradas.

### **7.1. Microsoft Access**

Microsoft Access es una herramienta de bases de datos integrada en el paquete de Microsoft Office, pero sólo en la edición Profesional y superiores.

No se considera una herramienta profesional, sino más bien a nivel de uso personal en el hogar. Sin embargo, es un sistema gestor de bases de datos relacional, y permite la creación de los conceptos principales que giran alrededor de una base de datos: tablas, vistas e interfaz de usuario.

Además, permite el acceso externo a los datos a través de aplicaciones programadas con Visual Basic, sin necesidad de añadir mucho código ni crear conexiones ODBC.

Microsoft Jet es el motor de base de datos interno de Access, y las librerías de Jet se incluyen en las instalaciones de Windows, e incluso se actualizan a través de Windows Update. De esta forma, las aplicaciones hechas en Visual Basic pueden acceder a las bases de datos de Access sin ni siquiera tener Access instalado.

Sin embargo, el lenguaje embebido que se usa en Access es una variante del Visual Basic llamada VBA, como se comentó en el apartado de metodología, respecto de las tecnologías utilizadas.

VBA es una especie de Visual Basic ampliado y adaptado a los objetos de cada aplicación en la que se integre: Word, Excel, Access, Project, CorelDRAW,... de tal forma que se pueden manipular los objetos propios de cada uno de esos programas a través de módulos de código que se pueden ejecutar, por ejemplo, en respuesta al evento de hacer clic en un botón personalizado de la barra de herramientas.

Finalmente... para el acceso a datos desde el código VB/VBA se usa una tecnología llamada ADO (ActiveX Data Objects), que provee funciones de acceso a los datos almacenados en la base de datos.

### **Objetos de Access**

Los objetos de Access como base de datos son:

- Tablas: Almacenan los datos
- Consultas: Vistas para obtener información relacionando tablas
- Formularios: Pantallas de acceso a los datos con un formato
- Informes: Listados en base a ciertas condiciones

Especial mención reciben los formularios, ya que se les puede dotar de un comportamiento a elegir por el desarrollador mediante código que se introduce como respuesta a eventos sobre los controles de los que disponga el formulario. Por ejemplo, un formulario debería tener un botón para cerrarlo, y para ello se requiere código en lenguaje VBA en el evento *OnClick* de dicho botón.

### **Ventajas de Access**

La ventaja principal es que en Access toda la base de datos y sus objetos asociados se encuentran centralizados en un único lugar.

Las capas de acceso a datos, lógica de negocio y presentación que en las arquitecturas cliente-servidor están distribuidas se juntan en el mismo lugar, y de hecho, van empaquetadas en el mismo archivo, por decirlo así.

Además, el desarrollo de ciertos objetos del sistema puede ser relativamente fácil si no hay muchos requisitos de accesibilidad, seguridad, validación,... y si se quiere instaurar una política que asegure la calidad en esos aspectos, viene provisto de un lenguaje de programación adaptado específicamente para tratar con sus propios objetos.

En ese sentido, Access se considera una herramienta WYSIWYG (“lo que ves es lo que obtienes”), dado que no se requiere nada más que lo que el propio Access ofrece para la creación y manipulación de sus objetos.

Finalmente, el motor de base de datos Jet es el más indicado sobre otros en ciertas circunstancias, como es el caso, ya que:

- No hay mucha complejidad
- El número de usuarios es mínimo
- El volumen de datos que se moverá es bajo
- La integridad de los datos no es una cuestión vital

Es cierto que Microsoft SQL Server y su motor MSDE ofrecen cobertura para todo tipo de situaciones, pero aumentan la complejidad del desarrollo.

### **Inconvenientes de Access**

El mayor inconveniente desde el punto de vista de las posibilidades de Access es que no es un sistema basado en la arquitectura cliente-servidor. Por tanto, ante la necesidad de implantar un sistema con acceso simultáneo desde distintos equipos, Access no puede dar una respuesta eficiente y fiable, ni siquiera garantizar el acceso a los datos, ya que sus bloqueos son demasiado estrictos.

Además, Access sólo puede trabajar con una base de datos al mismo tiempo, así que si fuera necesario trabajar con más de una, habría que considerar SQL Server.

Otra característica negativa de Access es que no tiene control de acceso por usuarios, de tal forma que no hay diferencia de usuarios entre administradores, gestores de contenido y usuario final, todos los usuarios tienen todos los privilegios. Esto se puede mitigar de una forma que se comentará más adelante.

Por otra parte, Access no tiene mucha capacidad de preservar la integridad en los datos, ya que no tiene un sistema transaccional basado en *logs*. Si se produce una interrupción de corriente en medio de una operación de modificación en los datos, y existen logs, la base de datos podría repararse automáticamente y evitar corrupción o pérdidas en los datos. Por tanto, las bases de datos en Access requieren copias de seguridad con más frecuencia, aunque siempre depende del volumen de movimientos.

## **7.2. Modo de trabajo con Access**

En Access hay varios modos de trabajo, en función de la forma en la que se quiera trabajar, o del nivel de conocimientos del usuario.

### **Las tablas**

Para crear tablas, un usuario novel puede usar un asistente, que pregunta qué tipo de base de datos es, y que campos quiere, ofreciendo campos genéricos como clientes, empleados, pedidos,... de tal forma que simplemente se eligen los campos y la tabla se crea automáticamente, pero es Access quien decide todas las cuestiones del modelo lógico y físico (tipos de datos de los campos, relaciones,...).

También se pueden crear directamente sobre una tabla sin tipo, añadiendo datos, y Access detectará los tipos en base a lo que se introduzca, y configurando luego el número y nombre de las columnas de la tabla. Este estilo recuerda al modo de funcionar de Excel.

El modo de trabajo utilizado en el desarrollo de esta aplicación ha sido el de la "*vista diseño*", mencionado como "modo diseñador" cuando se habló de Access en las herramientas utilizadas.

En este modo, hay que definir cada uno de los campos de una tabla, dándole un nombre y un tipo, describiendo su tamaño, su formato, si es un dato requerido, si acepta valores nulos,... y también es necesario definir claves principales y ajenas en las tablas, para establecer luego las relaciones que existen entre ellas y su cardinalidad.

### **Las consultas**

Como en el caso anterior, un usuario poco avanzado, puede construir consultas con un asistente, que le permite elegir los campos que quiere obtener de las tablas deseadas, teniendo en cuenta sus relaciones.

Existe también la “vista diseño” para consultas, y en este caso, el sistema es más complejo, pero también es eminentemente visual. A este método se le llama QBE (*Consultas a través de un ejemplo*), ya que el usuario puede arrastrar los campos que desea obtener de las tablas que tiene a la vista, y aplicar criterios sobre ellos, como relaciones de comparación numérica u operaciones de texto; y finalmente Access traduce esos campos y criterios en una consulta en lenguaje SQL.

El modo de trabajo utilizado ha sido el de la vista diseño, pero la mayor parte de las ocasiones, se ha entrado directamente en la instrucción SQL para escribirla manualmente, ya que es una forma más limpia y comprensible.

## Los formularios

Para crear formularios se puede usar un asistente, que de nuevo preguntará que se desea tener en el formulario: campos de tabla, resultados de consultas,... y generará controles para mostrar estos datos. El resultado es un formulario con un aspecto visual genérico y nada atractivo.

Por defecto, se usa un formato para los formularios que permite ver varios registros a la vez, como si fuera un listado, pero con el asistente se pueden requerir formularios que muestren sólo los campos de un registro y se disponga de controles para cambiar de registro, aunque el resultado también es el de un formulario genérico y aburrido.



Figura 7A: Control de posición (inicial, anterior, posterior, final, nuevo)

La alternativa es crear formularios mediante la “vista diseño”, con la que se obtiene un formulario vacío del que se pueden cambiar todas sus características (ancho, alto, color,...).

A este formulario se le pueden incorporar controles básicos de acceso a datos, como etiquetas, cajas de texto, cuadros combinados,... y también se pueden modificar las propiedades de cada uno.

Sobre todo, hay ciertas propiedades interesantes en un control, como:

- El formato: Permite obligar a que un control sólo pueda contener datos de un tipo determinado, como números, texto, fechas,...
- La máscara de entrada: Permite limitar el tipo y la cantidad de caracteres que se pueden escribir (letras, números, mayúsculas,...).
- El origen de la fila: Permite que un control, como por ejemplo un cuadro combinado, pueda obtener directamente sus datos a partir de una consulta, sin necesidad de usar código.
- La regla de validación: Permite forzar a que un control no quede vacío, o que sea un número positivo, o una fecha entre dos límites.
- Los eventos: Cada control tiene asociados unos eventos propios, como “al hacer clic”, “al presionar una tecla”, “al perder el enfoque”... y se puede establecer mediante instrucciones en lenguaje VBA cuál será el comportamiento que tenga ese control al surgir el evento.

Los propios formularios, de hecho, pueden obtener sus datos de una tabla, o una consulta, que también se creará en vista diseño, a través de una propiedad llamada “origen del registro”.

Si un formulario obtiene los datos de esa forma, se pueden agregar controles para mostrar los datos que el formulario provee a través de una propiedad llamada “origen del control”, que debe establecerse en cada control.

Si un control tiene su origen de datos en algún campo provisto por el formulario, se considera un campo “dependiente”, en contraposición a los campos independientes, que son los demás. Esto es relevante, ya que los campos dependientes no aceptan la introducción de cualquier tipo de datos por el usuario, sino sólo el tipo de datos que se estableció en el diseño de la tabla de la cual obtienen sus datos.

Así que el modo de trabajo sobre los formularios ha sido el de crearlos en vista diseño, estableciendo todos los orígenes de datos posibles para evitar realizar acceso a datos mediante código, y agregando código en los eventos que se ha considerado necesarios, tanto para modificar el comportamiento del formulario, como para acceder a los datos en circunstancias que no se pueden cubrir con las propiedades mencionadas.

### **Los informes**

Como en los casos anteriores, los informes también se pueden crear con asistente, que preguntará los campos que se quieren mostrar en el listado.

Pero lo normal para no obtener listados genéricos es realizarlos en la “vista diseño”, en la que se eligen los campos del informe a través de una tabla o consulta, que también se creará en vista diseño, y se elige la posición, el formato,... de cada campo de los registros a mostrar.

### **7.3. Dificultades encontradas**

La dificultad más importante ha sido el hecho de darse cuenta de que un formulario de Access sin código personalizado por detrás es bastante improductivo y a veces impredecible.

Es necesario crear los formularios manualmente en “vista diseño”, pero además hay que personalizar su comportamiento, a veces hasta tal punto, que da la sensación de que las propiedades para el control de datos no tienen sentido, ya que no cubren todas las necesidades de validación y hay que recurrir al código.

Por ejemplo, para validar una fecha, en teoría basta con establecer en las propiedades del control que el formato es el de una fecha corta. Pero si se hace así, y se introduce texto, el sistema genera un mensaje un poco genérico, pero no corrige el error, ni deshace la edición, ni permite seguir

escribiendo en otros controles hasta que no se corrija, e incluso dificulta el cierre del formulario, mostrando de nuevo el mensaje de error.

Aquí se puede ver un mensaje de ejemplo al introducir texto en un campo cuyo formato es el de un número entero:

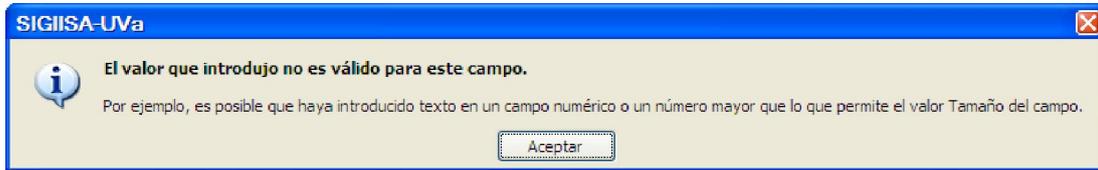


Figura 7B: Mensaje de error al escribir datos erróneos

Dado que todos los datos que se introducen en la aplicación SIGIISA-UVa son validados para evitar errores al introducirlos en la base de datos, es necesario evitar que mensajes de error como este molesten a un usuario menos avanzado.

Aún así para los controles dependientes, esa es la forma usual de la validación de datos, ya que Access prioriza el tipo de datos del campo sobre la validación personalizada que se ejerce mediante el código.

También pasó algo parecido con las máscaras de entrada, cuyo comportamiento producía ciertos inconvenientes desagradables, como por ejemplo, que al hacer clic en medio de un campo de 10 caracteres, el cursor se sitúe de tal forma que sólo se pueden escribir los 5 últimos (porque el cursor no se puso al principio del campo) y hay que borrar el campo y empezar de nuevo. Un usuario que avance de campo en campo con el tabulador no lo notará, pero uno que lo haga con ratón lo verá extremadamente molesto.

Pero mucho más grave es el comportamiento por defecto de los formularios de Access en ausencia de una lógica que defina en qué momento se produce la actualización de datos sobre las tablas.

Un formulario graba por defecto cualquier dato que cumpla la validación de tipos, simplemente al cambiar de registro, o cerrar el formulario.

La solución aportada pasa por obligar al formulario a deshacer los cambios hechos en él si no se cumple algún requisito, como por ejemplo, sólo se guarda el registro si se hace clic en un botón creado expresamente para guardarlo. Obviamente, esto se consigue sólo mediante código.

Y otra dificultad ha sido el sistema “dual” de propiedades de Access, que en la “vista diseño” está en español y en el código se debe escribir en inglés. Por ejemplo, para cambiar el color de una caja de texto, en la vista diseño se puede encontrar bajo la propiedad “color de fondo”, pero en el código se debe usar `.BackColor` para modificarlo.

## [SIGIISA-UVa] Sección 1: Memoria del TFG

La ayuda de Access contempla ambas situaciones a la vez, por separado y no suele ayudar mucho para solventar situaciones en las que se buscan determinadas propiedades para un control.

Otra cosa que es desagradable es que la ayuda no tenga listados exhaustivos de funciones de determinado tipo, como por ejemplo un listado de funciones de trabajo con cadenas de texto.

Existe un listado por categorías en cierto componente llamado “*Generador de expresiones*”, pero ahí las funciones también están traducidas al español, con lo cual no se puede establecer un paralelismo a la hora de buscar determinada función para usarla en código.

Por ejemplo, fue muy difícil encontrar la función que extrae una subcadena de una cadena dada, ya que en el generador de expresiones aparece como “*Medio*”, sin especificar lo que hace y no se encontraba nada al buscar en la ayuda de VBA. Al final, la función de VBA era “*Mid*”, y aunque se puede decir que la relación es obvia, hay que depender de Internet para no desperdiciar mucho tiempo buscando en la ayuda provista por Access.

También hubo dificultades menores alrededor de Access como su baja capacidad a la hora de incluir recursos gráficos y gestionar las transparencias. La única forma de poder utilizar un botón con un icono circular es incrustando la imagen con formato de icono de 256 colores y transparencia prehabilitada.

Access no permite cargar recursos gráficos de 32 bit con transparencias por canal alpha, ni tiene un sistema de carga basado en que el primer píxel representa el color transparente, como usualmente ocurre en otros entornos de desarrollo.

Esto limita mucho las posibilidades de cara a hacer la interfaz más amigable o crear efectos más avanzados, además de generar un trabajo extra con programas de edición gráfica para adaptar los recursos a los formatos aceptados por Access (básicamente, mapas de bits e iconos).

Finalmente, hay una dificultad más a comentar que merece una mención especial. Al abrir una base de datos de Access, lo que se observa es una ventana MDI de control que contiene todos los objetos creados en la base de datos (tablas, consultas, formularios,...). Esto es realmente inadecuado si el que abre la base de datos no es un usuario intermedio o avanzado, porque no sabrá qué tiene que hacer a partir de ahí.

La solución propuesta ha sido establecer un sistema por el cual si se abre la base de datos con doble clic, sólo se pueden ver los formularios, y maximizados para que no se vea lo que hay debajo. Es imposible impedir que se restauren a su tamaño original, pero al menos se ha impedido que sean minimizados o cerrados manualmente en lugar del sistema de navegación con botones que se ha añadido.

También se han ocultado los elementos de la barra de herramientas que permitirían cambiar un formulario de la “vista formulario” a la “vista diseño”, de forma que es virtualmente imposible modificar los objetos de la base de datos o el código que subyace tras los formularios.

Aún así, es imposible ocultar más cosas, y el menú de Access en versión reducida sigue a la vista. No es la versión completa, por lo que tampoco se puede cambiar de vista a través del menú.

Con este sistema, se dificulta el acceso para el desarrollador/administrador de la base de datos, y es necesario abrir la base de datos con la combinación Shift+Enter para poder ver la ventana MDI de control que contiene todos los objetos creados en la base de datos.

## 8. CONCLUSIONES

La primera conclusión que se ha sacado de todo el proceso es que desarrollar con Access ofrece una mezcla de satisfacción y frustración.

Los entornos de desarrollo con funciones muy básicas requieren la combinación de muchas de ellas para obtener resultados aprovechables, pero son extremadamente flexibles. Los entornos con funciones muy elaboradas que realizan mucho trabajo al mismo tiempo ahorran mucho tiempo en codificación, pero son más rígidos y menos adaptables.

Un ejemplo de esto sería:

- Un lenguaje que provee la función ValidarCampo requiere recorrer el formulario entero para validarlo por completo, pero permite evitar algún elemento si es necesario, o tener un tratamiento distinto para los elementos deseados.
- Un lenguaje que provee la función ValidarFormulario hace todo el trabajo por sí mismo, pero no permite tener distintas posibilidades para cada elemento si es necesario.

Así que la valoración personal sobre Access es que permite hacer muchas cosas, pero el grado de personalización sobre el comportamiento por defecto de los elementos es mucho menor.

Sin embargo, allá donde ha sido posible, se ha modificado el comportamiento por defecto para asegurar una mayor calidad en el producto entregado. Por ejemplo, en la mayoría de los campos, la validación por Access era tan poco orientativa que se decidió incluir un pequeño recordatorio en la parte superior con las 2 o 3 cuestiones más importantes, y un mensaje para obtener más información sobre los requisitos del formulario haciendo clic sobre dicho recordatorio.

## [SIGHISA-UVa] Sección 1: Memoria del TFG

Otra de las conclusiones podría ser que el sistema de información se ha dejado ligeramente abierto para contemplar distintas posibilidades que pueden ocurrir en un futuro, como por ejemplo, que un alumno presente un anteproyecto, y algún tiempo más tarde decida presentar otro distinto sin llegar a presentar un TFG sobre el primer anteproyecto.

También se ha dejado la libertad de modificar campos que en teoría nunca se van a modificar, como la fecha de una convocatoria. Así que el sistema esta abierto a realizar ciertas operaciones, pero si el futuro desarrollador desea ser más restrictivo, los cambios necesarios son mínimo.

Por el contrario, se ha dejado restringida la posibilidad de modificar un campo de carácter vital propagando la modificación a través de la base de datos. En este sentido, debería ser el administrador el que decida en base a sus gustos personales si a un alumno se le permite cambiar su NIA, o si se permite cambiar el identificador de un TFG una vez que se le ha asignado a un alumno dicho TFG.

El resultado final es un sistema que seguramente recibirá refinamientos debido a la adaptación de este tipo de necesidades al entorno real de trabajo, pero que al mismo tiempo asegura un cierto nivel de automatismo y calidad en la forma de trabajar.

También hay detalles visuales a tener en cuenta respecto de la calidad, ya que se ha elegido un código de colores para los distintos tipos de gestiones:

- Azul: Alumnos, modalidades de entrada y anteproyectos
- Verde: Profesores, áreas, grupos de tutoría y propuestas de TFG
- Rojo: Asignaciones de TFG, convocatorias, y defensas
- Amarillo: Informes

Y los campos de los formularios se colorean de blanco, amarillo o rojo, según el valor introducido, para indicar estados de aviso o error.

Además, se ha querido mantener una interfaz mínima que evite confusión con una barra de herramientas llena de botones con distintas opciones. En cada formulario hay uno o dos botones, como máximo que permiten guardar, o volver atrás sin guardar.

Otra cuestión visual ha sido el empeño por obtener una interfaz de un tamaño lo suficientemente pequeño, aunque limpia y clara, para ser mostrada en ordenadores con pantallas pequeñas, como los *netbook*.

Finalmente, otro criterio de calidad añadido es el ofrecimiento de soporte y consultoría de un año sobre el producto entregado. Dado que éste va a ser un sistema con uso en un entorno real, lo lógico en este caso sería:

- Ayudar en la corrección de algún posible error
- Orientar en el futuro desarrollo de objetos de la base de datos, sobre todo formularios e informes
- Facilitar manuales o tutoriales al futuro desarrollador
- Tener disponibilidad por correo electrónico
- Asistir en persona si es necesario, una vez por semana

Aunque se ha mencionado antes las conclusiones positivas por su mayor relevancia en la valoración final, también es necesario comentar las conclusiones negativas para completar este apartado.

Una de ellas es la lejanía de la realidad respecto de la planificación temporal realizada. Para empezar, por la dificultad de compatibilizar el trabajo con los estudios del Curso Puente, y además intentar llevar a cabo el TFG aunque fuera a ritmo bajo.

Pero lo peor son los contratiempos, contra los que a veces es imposible luchar, y distintos problemas laborales en el verano, y otras cuestiones personales han inutilizado totalmente las ventajas de la planificación.

También se estimó en el COCOMO una aproximación de unas 2000 líneas de código, pero tras comprobar que Access no se comporta genéricamente de acuerdo a los criterios de calidad requeridos, pues fue necesario añadir unas 1000-1200 líneas de código más. Esto también habría afectado al presupuesto y la planificación temporal, aunque no se hubieran producido los contratiempos comentados anteriormente.

Es por esto que al final no se han desarrollado todas las funciones requeridas por la aplicación, y se han quedado sin incluir algunos de los requisitos de la especificación funcional del anteproyecto, como los informes de estadísticas y la exportación de datos en texto plano.

## **9. POSIBLES AMPLIACIONES**

Como se ha comentado anteriormente, hay algunos requisitos que no se han terminado incluyendo en la versión que se entrega finalmente.

De esta forma, esas serán precisamente las ampliaciones que seguramente se realizarán primero.

Además, las hojas de cálculo ofrecían una funcionalidad de generación de código HTML a partir de ciertas plantillas, concatenadas a ciertos campos. De esa forma, se generan bloques de código que contienen resúmenes de los anteproyectos, las propuestas de TFG, las asignaciones de TFG, los tribunales y las defensas de TFG. Estos bloques de código son los que se copian a la página web de la Escuela Universitaria de Informática de Segovia con el objetivo de hacer pública esa información.

Esta ampliación también requiere poca intervención, así que sería una buena toma de contacto para el futuro desarrollador, y quizá un buen motivo para concertar una reunión de trabajo.

Otra funcionalidad que tenían las hojas de cálculo es la obtención de tasas de rendimiento, de éxito,... y que quizá sea implementada en un futuro.

También es relevante aclarar que no se prevé la posibilidad de tener que desplegar la base de datos en una infraestructura de red, ni con múltiples accesos simultáneos.

Pero si algún día la aplicación se diversifica en cuanto a funciones ofrecidas, y se desea que los empleados de la Universidad tengan acceso a ella, se ha buscado información al respecto.

En ese hipotético caso, existe la posibilidad de convertir esta aplicación en un Microsoft Access Project, lo cual permitiría que las tablas, las relaciones y las consultas pasen a estar dentro de Microsoft SQL Server, y los formularios de acceso se mantengan dentro de una aplicación de Access, de tal forma que a través de la tecnología OLE DB, la aplicación se podría desplegar en ordenadores clientes con Access, manteniendo la base de datos en un servidor con SQL Server.

Esta operación puede verse con más detalle en esta página web:

<http://office.microsoft.com/en-us/access-help/about-an-access-project-adp-HP005273103.aspx>

Finalmente, habría que hablar de la versión de Microsoft Office, ya que a lo mejor es necesario realizar una migración en el futuro.

Por experiencia propia en la Diputación Provincial de Segovia, las bases de datos que se creaban en Access 97 se mantenían en Access 97 hasta que por renovación de equipos, se instalaba una versión superior, y entonces se hacía la migración.

La razón no es que sean incompatibles, pero a diferencia de los formatos de Word y Excel, las bases de datos de Access presentan pequeñas incompatibilidades hacia versiones inferiores. Por esto, si se accedía a una base de datos desde distintos ordenadores, con diferentes versiones de Access, lo que ocurría al final es que desde la versión más antigua dejaban de funcionar determinadas operaciones.

Por tanto, las migraciones se hacían hacia arriba, y se renovaban todos los ordenadores que accedían a la base de datos afectada, de tal forma que existían bases de datos en Access 97, otras en Access 2000, en Access XP y Access 2003. Y existía un gran temor a la migración a Access 2007.

En este caso, la elección de Access 2003 se debe a que es la que está instalada en el ordenador desde el que se hacen estas gestiones actualmente con las hojas de cálculo.

Si fuera necesaria una migración a Access 2007/2010/2013, habría que tener claro que debe considerarse un paso sin posibilidad de vuelta atrás, y que es posible que haya que hacer pequeños ajustes en el diseño o en la programación, aunque se ha buscado información al respecto y el proceso no es tan traumático como se temía en la Diputación Provincial de Segovia.

## 10. GLOSARIO DE ACRÓNIMOS

Se han utilizado algunos acrónimos en la redacción de esta memoria y se considera necesario aclarar el significado de cada uno, ordenados alfabéticamente:

- **ADO** (*ActiveX Data Objects*)  
Es una tecnología de Access que provee funciones de acceso a los datos almacenados en la base de datos y modos de realizar esos accesos. Es una evolución de la tecnología anterior de Access, llamada **DAO** (Data Access Objects).
- **CASE** (*Computer Assisted Software Engineering*)  
Son herramientas de Ingeniería del Software asistidas por ordenador, y permiten realizar la parte de planificación, análisis y diseño de la aplicación mediante la creación de diagramas de diversos tipos.
- **CD-ROM** (*Compact Disc – Read-Only Memory*)  
Se refiere a un disco compacto de grabación única y lectura múltiple a través de tecnología láser, por lo que se los considera discos ópticos en lugar de magnéticos, como históricamente eran los discos para la grabación de datos digitales.
- **COCOMO** (*COnstructive COst MOdel*)  
Es un modelo de costes utilizado para estimar el esfuerzo necesario para desarrollar un proyecto de software. Sus ecuaciones las derivó Boehm a partir de datos estadísticos y el modelo fue refinado posteriormente en varias ocasiones.
- **LAME** (*LAME Ain't an MP3 Encoder*, recursivamente)  
Es el codificador de audio en el formato MP3 más ampliamente extendido y más activamente desarrollado a pesar del paso del tiempo, ya que se creó como un proyecto de código abierto.
- **MDI** (*Multiple Document Interface*)  
Es un sistema para interfaces gráficas e usuario que permite que una ventana “padre” pueda alojar múltiples ventanas “hijo”. Las ventanas hijo se pueden maximizar, minimizar y restaurar, pero siempre quedarán circunscritas al espacio determinado por las dimensiones de la ventana “padre”.
- **MSDE** (*MicroSoft Database Engine*)  
Es el motor de base de datos de Microsoft SQL Server, y permite muchas más posibilidades y capacidades que el motor de Access, llamado *Jet*, a costa de mayor complejidad. Entre las mejoras que ofrece están la capacidad para instaurar políticas de seguridad y la de conservar la integridad sobre los datos ante determinados errores.

- **ODBC** (*Open DataBase Connectivity*)  
Es un método estándar para compartir datos entre programas y bases de datos mediante el lenguaje SQL. Los entornos de desarrollo o el sistema operativo suelen ofrecer la posibilidad de establecer conexiones ODBC sin necesidad de instalar software adicional.
- **QBE** (*Query by Example*)  
Este acrónimo describe una forma visual de crear consultas en Access. Simplemente, se arrastran campos de una o varias tablas a una zona en la que se eligen los campos que se quieren mostrar y las condiciones que se aplican sobre ellos; y luego Access traduce esta forma de trabajo en una instrucción en lenguaje SQL.
- **SIGIISA-UVa** (Sistema de Información del Grado en Ingeniería Informática de Servicios y Aplicaciones)  
Es el nombre que se ha elegido para la aplicación de este Trabajo Fin de Grado, para evitar usar un nombre tan largo en distintos lugares de la aplicación y de la memoria.
- **SQL** (*Structured Query Language*)  
Es un lenguaje de Cuarta Generación y no procedural para acceso a datos, en el cual no se indica cómo obtener la información deseada, sino cuál es la información que se desea, y es el motor de la base de datos el que tiene que implementar la funcionalidad necesaria para obtener esa información.
- **TFG** (Trabajo Fin de Grado)  
Es una asignatura de los nuevos Grados según el denominado “plan Bolonia”, regulados en España a través del Real Decreto 1393/2007. Citando a la propia ley: “*es un trabajo de reflexión final en el cual el estudiante deberá mostrar, mediante una presentación oral pública ante un tribunal, que ha adquirido el conjunto de competencias asociadas al Título*”. Se ha utilizado el acrónimo tanto en la aplicación como en la memoria, dado que se repite múltiples veces en la semántica del sistema de información desarrollado.
- **VBA** (*Visual Basic for Applications*)  
Es una variante del lenguaje de programación llamado Visual Basic, de Microsoft, adaptada a los programas de Microsoft Office para poder manipular directamente los objetos propios de esos programas. Es un lenguaje interpretado, no compilado, y hace uso de objetos predefinidos por el programa en el que se ejecute (p.ej., botones en Access, celdas en Excel, párrafos en Word...), pero ni está orientado a objetos ni permite crear nuevas clases ni implementa la herencia.
- **WinCAC** (Compresor de Archivos por Capas para Windows)  
Es la aplicación desarrollada en mi Proyecto Fin de Carrera, que permitía comprimir archivos y editarlos sin recomprimir manualmente, mediante algoritmos inventados y con propósito educativo.

## 11. REFERENCIAS WEB Y BIBLIOGRAFÍA

Direcciones URL recomendadas con información genérica sobre Access, VBA y SQL Server, por si es necesaria una migración:

- **Access 2003:**  
<http://msdn.microsoft.com/en-us/library/aa167788.aspx>
- **Microsoft Access 2003 Language Reference:**  
[http://msdn.microsoft.com/en-us/library/aa663079\(office.11\).aspx](http://msdn.microsoft.com/en-us/library/aa663079(office.11).aspx)
- **SQL Server Developer Center:**  
<http://msdn.microsoft.com/en-us/sqlserver/aa336270>

Documentación utilizada en el desarrollo:

- **Libros** de iniciación. Permiten tener una idea de cómo es Access y las capacidades que tiene respecto del diseño visual, sin mencionar las posibilidades que permite el código VBA:
  - **Guía Visual de Access 97**, Miguel Pardo Niebla, Anaya Multimedia, 1998.
  - **Curso de Aula Mentor de Office 2003**, Francisco Hernández Rodríguez y César Casero Fernández, Ministerio de Educación y Ciencia, 2008.
- **Ayuda “online”** (integrada en el programa):
  - La del propio Access.
  - La del Visual Basic de Access.
- **Libros** con información exhaustiva. Ambos cubren de forma extensa cualquier duda que pudiera surgir y hablan sobre otras tecnologías de base de datos, migraciones, ...
  - **Access 2003 VBA Programmer's Reference**, Patricia Cardoza y otros, Wiley Publishing, 2004.
  - **Microsoft Office Access 2003 Bible**, Cary N. Prague y otros, Wiley Publishing, 2004.
- **Manuales** en formato digital, con ejercicios propuestos:
  - **Visual Basic para Aplicaciones del Access 2000**, manual escrito por Fermí Vilà.
  - **Curso de programación con VBA-Access**, manual escrito por Eduardo Olaz.
- **Apuntes** de la asignatura de “Gestión de Proyectos basados en las Tecnologías de la Información”. Se han utilizado estos apuntes sobre todo para la parte de planificación.



**[SIGIISA-UVa]**

**SECCIÓN 2:**

**MANUAL TÉCNICO**



**Universidad de Valladolid**



## 1. INTRODUCCIÓN

Antes de comenzar el manual técnico, hay que hablar de varios factores relevantes a la hora de elaborarlo:

- Aunque existen dos perfiles de acceso a la aplicación, uno de ellos va a usarla, y el otro va a mantenerla y ampliarla. Es decir, en realidad existe un único perfil de acceso, y no hay control de usuarios ni categorías de usuarios que pueden acceder a la aplicación.
- Dado que la aplicación se desarrolla dentro del sistema de Access, metodológicamente no existe la posibilidad de aplicar el paradigma de la orientación a objetos.
- Derivado de lo anterior, no existe la posibilidad de crear jerarquías de clases relacionadas por la herencia.

Por todo lo expuesto anteriormente, en este manual técnico no se encontrarán ciertos diagramas que sí se pueden encontrar en otros manuales, ni habrá un apartado específico para las clases creadas.

## 2. ESPECIFICACIÓN DE REQUISITOS DEL SISTEMA

Se ha hecho una división de los requisitos del sistema en dos categorías, de tal forma que primero se van a comentar los requisitos funcionales, por su grado de importancia, y a continuación los no funcionales.

### **2.1. Requisitos funcionales**

Los requisitos funcionales son aquellas funciones esenciales para que la aplicación pueda llevar a cabo su cometido, es decir, se refieren a las funciones que debe ofrecer la aplicación. El modo de actuación de cada función de la aplicación se describirá más adelante, como casos de uso.

Se han clasificado los requisitos funcionales en cuatro tipos distintos, en base al objeto sobre el que se giran.

#### **Tipo 1: Gestión de alumnos**

Alrededor de los alumnos se han querido considerar también las modalidades por las que un alumno puede entrar a cursar estudios en la Escuela Universitaria de Informática, y los anteproyectos que pueden presentar los alumnos para que se les asigne un TFG.

Es importante recalcar que se ha considerado que no se deben eliminar los alumnos del sistema de información, ya que es mejor que al darlos de baja, consten en el sistema como alumnos con curso de abandono informado. Así, se puede mantener un histórico de alumnos y obtener información de ellos, y también volver a darlos de alta en el sistema de una forma sencilla.

Por otra parte, al igual que los alumnos, se decidió que la aplicación no ofrecería la posibilidad de eliminar ningún otro objeto del discurso, en este caso modalidades de entrada y anteproyectos, para favorecer la posibilidad de obtener información, al igual que ocurre en el sistema actual basado en hojas de cálculo de Excel.

Por tanto, se han descrito los siguientes requisitos funcionales:

- RF1.1: Añadir un nuevo alumno
- RF1.2: Consultar un alumno existente
- RF1.3: Modificar un alumno existente
- RF1.4: Dar de baja a un alumno
- RF1.5: Ver el expediente completo de un alumno
- RF1.6: Añadir una nueva modalidad de entrada
- RF1.7: Consultar una modalidad de entrada existente
- RF1.8: Modificar una modalidad de entrada existente
- RF1.9: Añadir un nuevo anteproyecto
- RF1.10: Consultar un anteproyecto existente
- RF1.11: Modificar un anteproyecto existente

## **Tipo 2: Gestión de profesores**

Los profesores del centro tienen una consideración similar a la de los alumnos, ya que forman parte del sistema de información, y alrededor de ellos se han querido considerar también el área al que pertenece un profesor, los grupos de tutoría que se crean cada año para los nuevos alumnos de grado, a los que se asigna un tutor para los 4 años, y las propuestas de TFG que ofertan los profesores para los alumnos que prefieren la oferta de un profesor en lugar de presentar anteproyecto a la hora de realizar su TFG.

De nuevo, se ha considerado que no es necesaria ni recomendable la eliminación de objetos de este tipo, y en el caso de los profesores, basta con marcar su estado como activo o no activo para darlo de baja.

En base a ello, se han descrito estos requisitos funcionales:

- RF2.1: Añadir un nuevo profesor
- RF2.2: Consultar un profesor existente
- RF2.3: Modificar un profesor existente
- RF2.4: Dar de baja a un profesor
- RF2.5: Añadir un nuevo área
- RF2.6: Consultar un área existente
- RF2.7: Modificar un área existente
- RF2.8: Añadir un nuevo grupo de tutoría
- RF2.9: Consultar un grupo de tutoría existente
- RF2.10: Modificar un grupo de tutoría existente
- RF2.11: Añadir una nueva propuesta de TFG
- RF2.12: Consultar una propuesta de TFG existente
- RF2.13: Modificar una propuesta de TFG existente

### Tipo 3: Gestión de TFGs

Los Trabajos Fin de Grado tienen a su alrededor distintas consideraciones, ya que no tienen razón de existencia si no están asignados a un alumno. Por tanto, en lugar de tener en cuenta los TFGs, lo que se tiene en cuenta son las asignaciones de TFG. También se incluyen en este tipo de requisitos las convocatorias para defender un TFG y la calificación de esas convocatorias.

Por supuesto, este tipo de objetos no debe ser eliminado, dado que si no se eliminan los alumnos con el paso del tiempo, este tipo de objetos son los que dan información sobre el historial de ese alumno, y de ahí viene el requisito funcional RF1.5 (“Ver el expediente completo de un alumno”).

Así, se han descrito los siguientes requisitos funcionales:

- RF3.1: Añadir una nueva asignación de TFG
- RF3.2: Consultar una asignación de TFG existente
- RF3.3: Modificar una asignación de TFG existente
- RF3.4: Añadir una nueva convocatoria de defensa
- RF3.5: Consultar una convocatoria de defensa existente
- RF3.6: Modificar una convocatoria de defensa existente
- RF3.7: Añadir la calificación de una defensa de TFG
- RF3.8: Consultar la calificación de una defensa de TFG existente
- RF3.9: Modificar la calificación de una defensa de TFG existente

### Tipo 4: Informes

Aparte de los requisitos anteriores, que permiten crear, acceder o modificar un objeto en cada operación, es necesario que existan requisitos que permitan obtener listados completos o filtrados, pero con la capacidad de ver información acerca de más de un objeto en cada operación y con el posible objetivo de imprimir esa información.

No se han descrito muchos requisitos funcionales de este tipo, dado que en las hojas de cálculo de Excel ya se ofrece todo lo necesario y no se ha podido evaluar todavía la necesidad de más informes hasta que no se implante el sistema y se realice un uso continuo.

Por tanto, va a ser el futuro administrador/desarrollador quien evalúe la necesidad de añadir funcionalidad de este tipo en el futuro.

Los requisitos funcionales, en este caso, son:

- RF4.1: Ver el listado de alumnos por tutor
- RF4.2: Ver el listado de alumnos por curso de entrada
- RF4.3: Ver el listado de alumnos que han abandonado
- RF4.4: Ver el listado de propuestas de TFG por tutor
- RF4.5: Ver el listado de propuestas de TFG por departamento

## **2.2. Requisitos no funcionales**

Los requisitos no funcionales, a diferencia de los anteriores, no describen información a guardar, ni funciones a realizar por parte de la aplicación.

No se han descrito requisitos no funcionales de rendimiento, seguridad, o concurrencia, dado que la aplicación se ejecutará en un entorno muy localizado con un único usuario accediendo a la misma.

Tampoco se han descrito requisitos de escalabilidad, ya que no se prevé un crecimiento considerable, ni la necesidad de implantar el sistema en red.

Por tanto, los requisitos no funcionales se agrupan en las siguientes categorías:

- **Accesibilidad**
  - La aplicación se debe poder ejecutar en cualquier tipo de ordenador con Access 2003, aunque su pantalla sea de baja resolución, como en el caso de los netbook de 11” (1024x600).
- **Usabilidad**
  - Hay que reducir al máximo posible las distracciones que puede tener el usuario con la aplicación, utilizando para ello la cantidad mínima de elementos de pantalla y ocultando los que no son útiles.
  - Se debe implantar un sistema de ejecución que presente la base de datos de dos formas distintas:
    - Como un conjunto de formularios “en ejecución” para el usuario normal → con Enter o doble clic.
    - Como un conjunto de objetos en “modo diseño” o “modo programación” para el administrador → con Shift+Enter.
  - El sistema de guardado de información debe ser sencillo, y no debe generar pérdidas de tiempo con preguntas innecesarias, tanto si el objetivo es modificar algún dato, como si una modificación se produce por error y simplemente se quiere abandonar sin guardar.
- **Interfaz**
  - La aplicación debe diseñarse para tener una interfaz limpia, con el mínimo número de controles necesarios para realizar las operaciones definidas por los requisitos funcionales.
  - Se debe ofrecer un código de colores que permita asociar rápidamente qué tipo de operaciones se están realizando:
    - Azul: Operaciones asociadas a alumnos (RF tipo 1)
    - Verde: Operaciones asociadas a profesores (RF tipo 2)
    - Rojo: Operaciones relacionadas con TFGs (RF tipo 3)
    - Amarillo: Informes (RF tipo 4)
  - La interfaz debe ser amigable y visualmente atractiva, y los mensajes que genere deben ser educados con el usuario.

### 3. ESPECIFICACIÓN DE LOS CASOS DE USO

Como se dijo en la introducción de este manual técnico, realmente no existen distintos perfiles de acceso a la aplicación, ya que no hay distinciones para el usuario que la usa, y el administrador simplemente la mantiene y la amplía, corrigiendo de vez en cuando pequeños problemas directamente sobre las tablas si por ejemplo se ha insertado un alumno por error, ya que la aplicación no permite el borrado.

Partiendo de ese hecho, no se han identificado más actores que el usuario que accede a la aplicación para usarla como sistema de información.

Por todo lo expuesto anteriormente, en este manual técnico no se encontrará el diagrama completo de casos de uso, que muestra todos los actores y a qué casos de uso pueden acceder (además de la interacción entre casos de uso), ya que el diagrama completo se puede simplificar así:

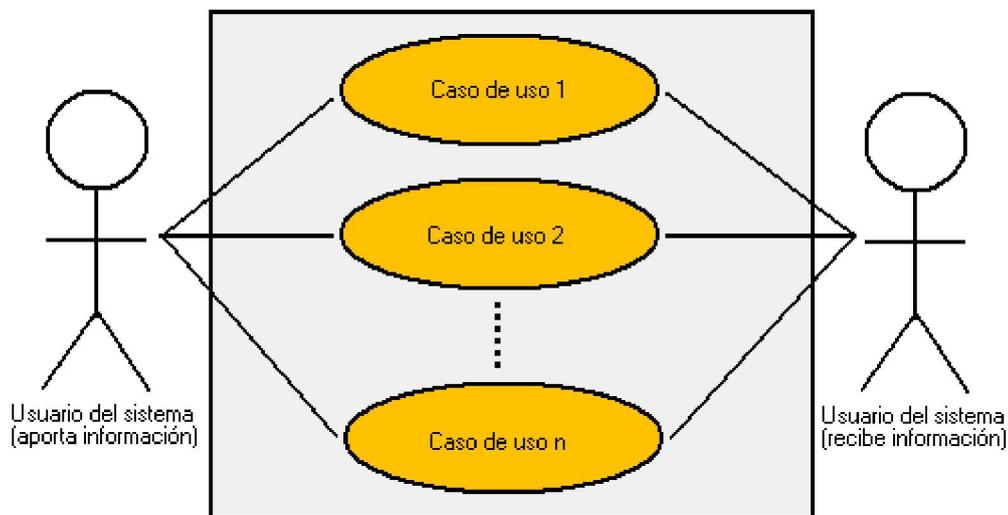


Figura 3A: Representación genérica de los casos de uso

Además, en el proceso de análisis de requisitos, se ha hecho evidente que es innecesario incluir un caso de uso para consultar un objeto y otro caso de uso para modificarlo. Realmente, ambos casos pueden fusionarse, dado que si se ofrece en una pantalla la información de un objeto, es fácil utilizar esa misma pantalla para modificarlo y guardarlo.

Un caso concreto de esto mismo también es el requisito funcional de dar de baja un alumno. Partiendo del hecho de que no se eliminan alumnos de la base de datos y se considera que un alumno se ha dado de baja si tiene curso de abandono informado, basta con aprovechar la pantalla de modificación del alumno para permitir introducir el curso de abandono.

Así pues, no todos los requisitos funcionales se transforman directamente en casos de uso. En todo caso, se ha realizado una clasificación de los casos de uso en base a los mismos tipos que los requisitos funcionales.

### **3.1. Casos de uso de tipo 1: Alumnos**

De 11 requisitos funcionales de este tipo que se especificaron, se pasa a sólo 7 casos de uso, por lo comentado anteriormente (fusión de los requisitos de consultas con los de modificaciones, y la inclusión de la baja del alumno dentro de la consulta/modificación del propio alumno).

#### **CU1.1: Añadir un nuevo alumno**

Objetivo	Añadir un nuevo registro a la base de datos con los datos que se guardan para un alumno: <ul style="list-style-type: none"> <li>• NIA (Número de Identificación del Alumno)</li> <li>• Apellidos</li> <li>• Nombre</li> <li>• Curso de entrada</li> <li>• Modalidad de entrada</li> <li>• Tutor asignado</li> </ul>
Precondición:	Si se quiere asignar un tutor, debe existir algún grupo de tutoría creado previamente para ese curso de entrada.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario introduce los datos secuencialmente.</li> <li>2. El usuario hace clic en “Guardar”.</li> <li>3. Se validan los campos del formulario y se inserta el nuevo alumno en la base de datos.</li> <li>4. Se avisa de la correcta adición del alumno.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si no hay grupos de tutoría creados para el curso de entrada, se avisa al usuario con un mensaje.</li> <li>• Si al guardar, hay algún campo que no pase la validación, no se inserta al alumno en la BD.</li> <li>• Si ya existe un alumno con ese NIA, se avisa al usuario con un mensaje de error.</li> </ul>
Postcondición:	Alumno añadido a la base de datos.

#### **CU1.2: Consultar/Modificar un alumno existente**

Objetivo	Permitir la consulta y/o modificación de los datos de un alumno (excepto su NIA), incluyendo la posibilidad de darlo de baja mediante el campo “Curso de abandono”.
Precondición:	El alumno existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige un alumno de la lista.</li> <li>2. El usuario modifica algún dato del alumno.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se actualiza el alumno en la base de datos.</li> <li>5. Se avisa de la correcta modificación del alumno.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos del alumno en la base de datos.</li> </ul>
Postcondición:	Alumno actualizado en la base de datos.

**CU1.3: Ver el expediente completo de un alumno**

Objetivo	Ver los anteproyectos presentados por un alumno, los TFGs que se le asignaron, sus convocatorias de defensa y las calificaciones de defensa de los TFGs.
Precondición:	El alumno existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige un alumno de la lista.</li> <li>2. Se cargan todos los datos mencionados para ese alumno, si es que existen.</li> </ol>
Flujo alternativo:	-
Postcondición:	- (no se permite modificar estos datos por esa vía)

**CU1.4: Añadir una nueva modalidad de entrada**

Objetivo	<p>Añadir un nuevo registro a la base de datos con los datos que se guardan para una modalidad de entrada:</p> <ul style="list-style-type: none"> <li>• Identificador de la modalidad de entrada</li> <li>• Denominación de la modalidad de entrada</li> <li>• Número de cursos</li> </ul>
Precondición:	-
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario introduce los datos secuencialmente.</li> <li>2. El usuario hace clic en “Guardar”.</li> <li>3. Se validan los campos del formulario y se inserta la nueva modalidad de entrada en la BD.</li> <li>4. Se avisa de la correcta creación de la modalidad.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no pase la validación, no se inserta la modalidad en la BD.</li> <li>• Si ya existe una modalidad con ese identificador, se avisa al usuario con un mensaje de error.</li> </ul>
Postcondición:	Modalidad de entrada añadida a la base de datos.

**CU1.5: Consultar/Modificar una modalidad de entrada existente**

Objetivo	Permitir la consulta y/o modificación de los datos de una modalidad de entrada (excepto su identificador).
Precondición:	La modalidad existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige una modalidad de entrada de la lista.</li> <li>2. El usuario modifica algún dato de la modalidad.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se actualiza la modalidad de entrada en la BD.</li> <li>5. Se avisa al usuario de la correcta modificación de la modalidad de entrada.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos de la modalidad de entrada en la BD.</li> </ul>
Postcondición:	Modalidad de entrada actualizada en la base de datos.

### CU1.6: Añadir un nuevo anteproyecto

Objetivo	Añadir un nuevo registro a la base de datos con los datos que se guardan para un anteproyecto: <ul style="list-style-type: none"> <li>• Fecha en la que se presenta el anteproyecto</li> <li>• Alumno 1 (primer autor)</li> <li>• Alumno 2 (segundo autor): opcional</li> <li>• Tutor 1 (primer tutor)</li> <li>• Tutor 2 (segundo tutor): opcional</li> <li>• Título del anteproyecto</li> </ul>
Precondición:	Los alumnos y los tutores del anteproyecto existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. Se genera automáticamente un identificador para el anteproyecto en base al curso académico que corresponde a la fecha de presentación introducida.</li> <li>2. El usuario introduce los datos secuencialmente.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se inserta el nuevo anteproyecto en la base de datos.</li> <li>5. Se avisa de la correcta adición del anteproyecto.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no pase la validación, no se inserta el anteproyecto en la base de datos.</li> </ul>
Postcondición:	Anteproyecto añadido a la base de datos.

### CU1.7: Consultar/Modificar un anteproyecto existente

Objetivo	Permitir la consulta y/o modificación de los datos de un anteproyecto (excepto su identificador).
Precondición:	El anteproyecto existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige un anteproyecto de la lista.</li> <li>2. El usuario modifica algún dato del anteproyecto.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se actualiza el anteproyecto en la base de datos.</li> <li>5. Se avisa al usuario de la correcta modificación del anteproyecto.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos del anteproyecto en la base de datos.</li> </ul>
Postcondición:	Anteproyecto actualizado en la base de datos.

### **3.2. Casos de uso de tipo 2: Profesores**

De 13 requisitos funcionales se obtienen 8 casos de uso, por la fusión de los requisitos de consultas con los de modificaciones, y la inclusión de la baja del profesor dentro de la consulta/modificación del mismo.

#### **CU2.1: Añadir un nuevo profesor**

Objetivo	Añadir un nuevo registro a la base de datos con los datos que se guardan para un profesor: <ul style="list-style-type: none"> <li>• Identificador del profesor / Alias</li> <li>• Apellidos</li> <li>• Nombre</li> <li>• Tipo: opcional</li> <li>• Área</li> <li>• Web: opcional</li> <li>• e-mail: opcional</li> <li>• Despacho: opcional</li> <li>• Teléfono</li> <li>• Profesor adscrito: SI/NO</li> <li>• Profesor en activo: SI/NO</li> <li>• Observaciones: opcional</li> </ul>
Precondición:	El área asignado existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario introduce los datos secuencialmente.</li> <li>2. El usuario hace clic en "Guardar".</li> <li>3. Se validan los campos del formulario y se inserta el nuevo profesor en la base de datos.</li> <li>4. Se avisa de la correcta adición del profesor.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no pase la validación, no se inserta al profesor en la BD.</li> <li>• Si ya existe un profesor con ese identificador/alias, se avisa al usuario con un mensaje de error.</li> </ul>
Postcondición:	Profesor añadido a la base de datos.

#### **CU2.2: Consultar/Modificar un profesor existente**

Objetivo	Permitir la consulta y/o modificación de los datos de un profesor (excepto su identificador/alias).
Precondición:	El profesor existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige un profesor de la lista.</li> <li>2. El usuario modifica algún dato del profesor.</li> <li>3. El usuario hace clic en "Guardar".</li> <li>4. Se validan los campos del formulario y se actualiza el profesor en la base de datos.</li> <li>5. Se avisa de la correcta modificación del profesor.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos del alumno en la base de datos.</li> </ul>
Postcondición:	Profesor actualizado en la base de datos.

### CU2.3: Añadir un nuevo área

Objetivo	Añadir un nuevo registro a la base de datos con los datos que se guardan para un área: <ul style="list-style-type: none"> <li>• Identificador del área</li> <li>• Departamento al que pertenece el área</li> </ul>
Precondición:	-
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario introduce los datos secuencialmente.</li> <li>2. El usuario hace clic en “Guardar”.</li> <li>3. Se validan los campos del formulario y se inserta el nuevo área de entrada en la base de datos.</li> <li>4. Se avisa de la correcta creación del área.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no pase la validación, no se inserta el área en la BD.</li> <li>• Si ya existe un área con ese identificador, se avisa al usuario con un mensaje de error.</li> </ul>
Postcondición:	Área añadida a la base de datos.

### CU2.4: Consultar/Modificar un área existente

Objetivo	Permitir la consulta y/o modificación de los datos de un área (excepto su identificador).
Precondición:	El área existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige un área de la lista.</li> <li>2. El usuario modifica algún dato del área.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se actualiza el área en la base de datos.</li> <li>5. Se avisa de la correcta modificación del área.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos del área en la base de datos.</li> </ul>
Postcondición:	Área actualizada en la base de datos.

### CU2.5: Añadir un nuevo grupo de tutoría

Objetivo	Añadir un nuevo registro a la base de datos con los datos que se guardan para un grupo de tutoría: <ul style="list-style-type: none"> <li>• Curso académico del grupo de tutoría</li> <li>• Tutor del grupo</li> </ul>
Precondición:	El tutor asignado existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. Se genera automáticamente un identificador para el grupo de tutoría en base al curso académico.</li> <li>2. El usuario introduce los datos secuencialmente.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se inserta el nuevo grupo de tutoría en la base de datos.</li> <li>5. Se avisa de la correcta creación del grupo.</li> </ol>

Flujo alternativo:	<ul style="list-style-type: none"> <li>Si al guardar, hay algún campo que no pase la validación, no se inserta el grupo de tutoría en la base de datos.</li> </ul>
Postcondición:	Grupo de tutoría añadido a la base de datos.

**CU2.6: Consultar/Modificar un grupo de tutoría existente**

Objetivo	Permitir la consulta y/o modificación del tutor asignado a un grupo de tutoría (no se permite cambiar el identificador, basado en el curso académico).
Precondición:	El grupo de tutoría existe previamente en la BD.
Flujo principal:	<ol style="list-style-type: none"> <li>El usuario elige un grupo de tutoría de la lista.</li> <li>El usuario modifica el tutor asignado al grupo de tutoría.</li> <li>El usuario hace clic en “Guardar”.</li> <li>Se validan el único campo del formulario y se actualiza el grupo de tutoría en la base de datos.</li> <li>Se avisa al usuario de la correcta modificación del grupo de tutoría.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>Si al guardar, el único campo que hay no cumple las condiciones de validación, no se actualizan los datos del grupo de tutoría en la base de datos.</li> </ul>
Postcondición:	Grupo de tutoría actualizado en la base de datos.

**CU2.7: Añadir una nueva propuesta de TFG**

Objetivo	<p>Añadir un nuevo registro a la base de datos con los datos que se guardan para una propuesta de TFG:</p> <ul style="list-style-type: none"> <li>Fecha en la que se presenta la propuesta</li> <li>Tutor 1 (primer tutor)</li> <li>Tutor 2 (segundo tutor): opcional</li> <li>Estado de la propuesta</li> <li>Título de la propuesta</li> <li>Resumen de la propuesta</li> </ul>
Precondición:	Los tutores de la propuesta de TFG existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>Se genera automáticamente un identificador para la propuesta de TFG en base al curso académico que corresponde a la fecha de presentación introducida.</li> <li>El usuario introduce los datos secuencialmente.</li> <li>El usuario hace clic en “Guardar”.</li> <li>Se validan los campos del formulario y se inserta la nueva propuesta de TFG en la base de datos.</li> <li>Se avisa de la correcta adición de la propuesta.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>Si al guardar, hay algún campo que no pase la validación, no se inserta la propuesta de TFG en la base de datos.</li> </ul>
Postcondición:	Propuesta de TFG añadida a la base de datos.

### CU2.8: Consultar/Modificar una propuesta de TFG existente

Objetivo	Permitir la consulta y/o modificación de los datos de una propuesta de TFG (excepto su identificador).
Precondición:	La propuesta de TFG existe previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige una propuesta de TFG de la lista.</li> <li>2. El usuario modifica algún dato de la propuesta.</li> <li>3. El usuario hace clic en “Guardar”.</li> <li>4. Se validan los campos del formulario y se actualiza la propuesta en la base de datos.</li> <li>5. Se avisa al usuario de la correcta modificación de la propuesta de TFG.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos de la propuesta en la base de datos.</li> </ul>
Postcondición:	Propuesta de TFG actualizada en la base de datos.

### 3.3. Casos de uso de tipo 3: TFGs

De 9 requisitos funcionales se pasa a 6 casos de uso, por la fusión de los requisitos de consultas con los de modificaciones.

#### CU3.1: Añadir una nueva asignación de TFG

Objetivo	<p>Añadir nuevos registros a la base de datos con los datos que se guardan para una asignación de TFG:</p> <ul style="list-style-type: none"> <li>• Fecha en la que se asigna el TFG</li> <li>• Anteproyecto de origen</li> <li>• Propuesta de origen</li> <li>• Alumno 1 (primer autor)</li> <li>• Alumno 2 (segundo autor): opcional</li> </ul>
Precondición:	Los alumnos de la asignación existen previamente en la base de datos, y el origen del TFG es único.
Flujo principal:	<ol style="list-style-type: none"> <li>1. Se genera automáticamente un identificador para la asignación de TFG en base al curso académico que corresponde a la fecha de asignación introducida.</li> <li>2. El usuario introduce los datos secuencialmente.</li> <li>3. Al elegir un anteproyecto como origen del TFG, se cargan los alumnos que lo presentaron.</li> <li>4. Se genera automáticamente un identificador para la asignación individual del TFG a cada alumno.</li> <li>5. El usuario hace clic en “Guardar”.</li> <li>6. Se validan los campos del formulario y se insertan tanto el TFG como las asignaciones individuales en la base de datos.</li> <li>7. Se avisa de la correcta asignación del TFG.</li> </ol>

Flujo alternativo:	<ul style="list-style-type: none"> <li>Si al guardar, hay algún campo que no pase la validación, no se inserta la asignación de TFG en la base de datos.</li> </ul>
Postcondición:	TFG y asignaciones individuales añadidos a la base de datos.

**CU3.2: Consultar/Modificar una asignación de TFG existente**

Objetivo	Permitir la consulta y/o modificación de los datos de una asignación de TFG (excepto su identificador).
Precondición:	El TFG y las asignaciones individuales de cada alumno a ese TFG existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>El usuario elige una asignación de TFG de la lista.</li> <li>Se cargan los datos del TFG y de las asignaciones individuales de los alumnos.</li> <li>El usuario modifica algún dato de la asignación.</li> <li>Al elegir un anteproyecto como origen del TFG, se pregunta si se desea cargar los datos de los alumnos que lo presentaron, generándose automáticamente nuevos identificadores para la asignación individual del TFG a cada alumno.</li> <li>El usuario hace clic en "Guardar".</li> <li>Se validan los campos del formulario y se actualizan tanto el TFG como las asignaciones individuales en la base de datos.</li> <li>Se avisa al usuario de la correcta modificación de la asignación de TFG.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan los datos de la asignación de TFG en la base de datos.</li> </ul>
Postcondición:	TFG y asignaciones individuales actualizados en la base de datos.

**CU3.3: Añadir una nueva convocatoria de defensa**

Objetivo	<p>Añadir un nuevo registro a la base de datos con los datos que se guardan para una convocatoria de defensa:</p> <ul style="list-style-type: none"> <li>Fecha de la convocatoria de defensa</li> <li>Fecha de publicación de la convocatoria: opcional</li> <li>TFG que se convoca</li> <li>Presidente, secretario y vocal del tribunal</li> <li>Presidente suplente, secretario suplente y vocal suplente del tribunal: opcionales</li> <li>Fecha de la defensa</li> <li>Hora de la defensa: opcional</li> <li>Lugar de la defensa: opcional</li> <li>Se convoca al alumno 1: SI/NO</li> <li>Se convoca al alumno 2 (si existía): SI/NO</li> </ul>
----------	--

Precondición:	El TFG que se convoca, los alumnos que se asignaron a ese TFG y los profesores para el tribunal existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. Se genera automáticamente un identificador para la convocatoria de defensa en base al curso académico que corresponde a la fecha de convocatoria introducida.</li> <li>2. El usuario introduce los datos secuencialmente.</li> <li>3. Al elegir el TFG convocado, se cargan los alumnos a los que se les asignó y su identificador de asignación individual del TFG, y por defecto se marcan como convocados.</li> <li>4. El usuario hace clic en “Guardar”.</li> <li>5. Se validan los campos del formulario y se inserta la convocatoria de defensa en la base de datos.</li> <li>6. Se avisa al usuario de la correcta creación de la convocatoria de defensa.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si se desmarcan todos los alumnos convocados, se muestra un mensaje de error al usuario.</li> <li>• Si al guardar, hay algún campo que no pase la validación, no se inserta la convocatoria de defensa en la base de datos.</li> </ul>
Postcondición:	Convocatoria de defensa añadida a la base de datos.

#### CU3.4: Consultar/Modificar una convocatoria de defensa existente

Objetivo	Permitir la consulta y/o modificación de los datos de una convocatoria de defensa (excepto su identificador).
Precondición:	La convocatoria de defensa, el TFG convocado y las asignaciones individuales de cada alumno a ese TFG existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige una convocatoria de la lista.</li> <li>2. Se cargan los datos de la convocatoria de defensa, el TFG convocado y las asignaciones individuales de los alumnos convocados.</li> <li>3. El usuario modifica algún dato de la convocatoria.</li> <li>4. Al modificar el TFG convocado, se cargan los alumnos a los que se les asignó y su identificador de asignación individual del TFG.</li> <li>5. El usuario hace clic en “Guardar”.</li> <li>6. Se validan los campos del formulario y se actualiza convocatoria de defensa en la BD.</li> <li>7. Se avisa al usuario de la correcta modificación de la convocatoria de defensa.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si se desmarcan todos los alumnos convocados, se muestra un mensaje de error al usuario.</li> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualiza la convocatoria de defensa en la base de datos.</li> </ul>
Postcondición:	Convocatoria de defensa actualizada en la BD.

**CU3.5: Añadir la calificación de una defensa de TFG**

Objetivo	Añadir nuevos registros a la BD con los datos que se guardan para la calificación de una defensa de TFG: <ul style="list-style-type: none"> <li>• Nota numérica y en letra del alumno 1 (sólo si estaba convocado)</li> <li>• Nota numérica y en letra del alumno 2 (sólo si estaba convocado)</li> <li>• El TFG está disponible en UVaDoc: SI/NO</li> </ul>
Precondición:	La convocatoria que se quiere calificar, el TFG convocado y los alumnos que fueron convocados existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige una convocatoria sin calificar de la lista.</li> <li>2. Se carga el TFG convocado y los alumnos que fueron convocados, junto a su identificador de asignación individual del TFG.</li> <li>3. El usuario introduce las calificaciones de los alumnos convocados.</li> <li>4. El usuario hace clic en “Guardar”.</li> <li>5. Se validan los campos del formulario y se insertan las calificaciones de los alumnos en la BD.</li> <li>6. Se avisa al usuario de la correcta adición de la calificación de los alumnos.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no pase la validación, no se insertan las calificaciones en la base de datos.</li> </ul>
Postcondición:	Calificaciones de defensa añadidas a la base de datos.

**CU3.6: Consultar/Modificar la calificación de una defensa de TFG existente**

Objetivo	Permitir la consulta y/o modificación de la calificación de una defensa de TFG.
Precondición:	La convocatoria de defensa, el TFG convocado, las asignaciones individuales de cada alumno a ese TFG y las calificaciones de cada uno existen previamente en la base de datos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elige una convocatoria de la lista.</li> <li>2. Se cargan los datos de la convocatoria de defensa, el TFG convocado, las asignaciones individuales de los alumnos convocados y sus calificaciones.</li> <li>3. El usuario modifica algún dato de la calificación.</li> <li>4. El usuario hace clic en “Guardar”.</li> <li>5. Se validan los campos del formulario y se actualizan las calificaciones en la BD.</li> <li>6. Se avisa al usuario de la correcta modificación de las calificaciones de la defensa del TFG.</li> </ol>

Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si al guardar, hay algún campo que no cumpla las condiciones de validación, no se actualizan las calificaciones de la defensa en la base de datos.</li> </ul>
Postcondición:	Calificaciones de la defensa del TFG actualizadas en la base de datos.

### **3.4. Casos de uso de tipo 4: Informes**

Todos los requisitos funcionales pasan a transformarse en casos de uso.

#### **CU4.1: Ver el listado de alumnos por tutor**

Objetivo	Ver la lista de alumnos a los que se les asigno determinado tutor.
Precondición:	En la base de datos existen alumnos con tutores asignados.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elije el tipo de listado que quiere (“Alumnos por tutor”).</li> <li>2. El usuario elige el tutor del que quiere obtener la información.</li> <li>3. El usuario hace clic en “Generar informe”.</li> <li>4. Aparece una nueva ventana con el listado de alumnos asignados al tutor elegido.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si no se elige ningún tutor, se presenta la lista completa de alumnos agrupados por su tutor asignado.</li> <li>• Si el tutor elegido no tiene ningún alumno asignado, se muestra un mensaje de advertencia.</li> </ul>
Postcondición:	-

#### **CU4.2: Ver el listado de alumnos por curso de entrada**

Objetivo	Ver la lista de alumnos que entraron en un curso determinado.
Precondición:	En la base de datos existen alumnos.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elije el tipo de listado que quiere (“Alumnos por curso de entrada”).</li> <li>2. El usuario introduce el curso de entrada del que quiere obtener la información.</li> <li>3. El usuario hace clic en “Generar informe”.</li> <li>4. Aparece una nueva ventana con el listado de alumnos que entraron en el curso de entrada elegido.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si no se introduce ningún curso, se presenta la lista completa de alumnos agrupados por curso.</li> <li>• Si en el curso elegido no entró ningún alumno, se muestra un mensaje de advertencia.</li> </ul>
Postcondición:	-

**CU4.3: Ver el listado de alumnos que han abandonado**

Objetivo	Ver la lista de alumnos que han abandonado los estudios.
Precondición:	En la base de datos existen alumnos que han abandonado.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elije el tipo de listado que quiere (“Alumnos que han abandonado”).</li> <li>2. El usuario hace clic en “Generar informe”.</li> <li>3. Aparece una nueva ventana con el listado de alumnos que han abandonado.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si no existe ningún alumno con curso de abandono informado, se muestra un mensaje de advertencia.</li> </ul>
Postcondición:	-

**CU4.4: Ver el listado de propuestas de TFG por tutor**

Objetivo	Ver la lista de propuestas de TFG de un tutor determinado.
Precondición:	En la base de datos existen propuestas de TFG y los profesores que las propusieron.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elije el tipo de listado que quiere (“Propuestas de TFG por tutor”).</li> <li>2. El usuario introduce el tutor del que quiere obtener la información.</li> <li>3. El usuario hace clic en “Generar informe”.</li> <li>4. Aparece una nueva ventana con el listado de propuestas del tutor elegido.</li> </ol>
Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si no se introduce ningún tutor, se presenta la lista completa de propuestas agrupadas por tutor.</li> <li>• Si el tutor elegido no ha presentado ninguna propuesta, se muestra un mensaje de advertencia.</li> </ul>
Postcondición:	-

**CU4.5: Ver el listado de propuestas de TFG por departamento**

Objetivo	Ver la lista de propuestas de TFG de un departamento determinado.
Precondición:	En la base de datos existen propuestas de TFG, y los profesores que las propusieron.
Flujo principal:	<ol style="list-style-type: none"> <li>1. El usuario elije el tipo de listado que quiere (“Propuestas de TFG por departamento”).</li> <li>2. El usuario introduce el departamento del que quiere obtener la información.</li> <li>3. El usuario hace clic en “Generar informe”.</li> <li>4. Aparece una nueva ventana con el listado de propuestas del departamento elegido.</li> </ol>

Flujo alternativo:	<ul style="list-style-type: none"> <li>• Si no se introduce ningún departamento, se presenta la lista completa de propuestas agrupadas por departamento.</li> <li>• Si el departamento elegido no ha presentado ninguna propuesta, se muestra un mensaje de advertencia.</li> </ul>
Postcondición:	-

## 4. ANÁLISIS Y DISEÑO DE LA BASE DE DATOS

En este apartado se ha considerado esencial incluir cierta información sobre el problema que resuelve la aplicación. De esta información parte el proceso de modelado, que transforma una situación del mundo real en una base de datos conservando la semántica de dicha situación y contemplando todos los posibles casos que se pueden producir en ella.

Por ello, se incluirán el modelo Entidad-Relación y el modelo Relacional y sus esquemas asociados como parte de ese modelado. Pero primero se incluye aquí una descripción de la situación real a partir de la cual empieza el proceso de modelado:

- En el centro se guarda cierta información sobre los alumnos que comienzan sus estudios de Grado: nombre, apellidos, NIA, curso de entrada, modalidad por la que entró y tutor.
- Un alumno entra por alguna de las modalidades de entrada del Grado, pero a los alumnos de las modalidades de los cursos puente, pasarela y Erasmus (de 1 año de duración) no se les asigna un tutor.
- Los tutores se asignan a los alumnos en base al curso de entrada y los apellidos de los alumnos, de tal forma que se crean grupos de tutoría para rangos de apellidos.
- Los alumnos presentan anteproyectos, individualmente o en pareja, de los que se guarda cierta información: alumnos que lo presentan, tutor o tutores del anteproyecto, título y fecha de presentación.
- En el centro se guarda cierta información sobre los profesores que hay: nombre, apellidos, alias, tipo, área, web, e-mail, despacho, teléfono. Además se desea saber si está adscrito, si está en activo y guardar otras observaciones cualquiera.
- Todos los profesores pertenecen a algún área.
- Los profesores ofertan propuestas de TFG, de las que se guarda: el tutor o tutores de la propuesta, el estado de la propuesta, el título, un resumen y la fecha de la propuesta.
- El centro asigna los TFGs a los alumnos en base a un anteproyecto o a una propuesta y se guarda la fecha de asignación.
- El centro publica convocatorias de defensa de TFG en las que se indica el TFG convocado, los alumnos que se convocan, los miembros titulares y suplentes del tribunal, la fecha de la convocatoria, de la publicación, y la fecha, hora y lugar de la defensa.
- Tras la defensa, se publican las notas de los alumnos convocados.

## 4.1. Esquema Entidad-Relación

Al modelo entidad-relación también se le llama modelo conceptual, ya que plasma los conceptos del mundo real, independiente de la plataforma y paradigma de modelado que se utilice.

La situación descrita anteriormente se puede modelar como describe el siguiente esquema (en el que no se han incluido los atributos de cada entidad para ofrecer mayor limpieza):

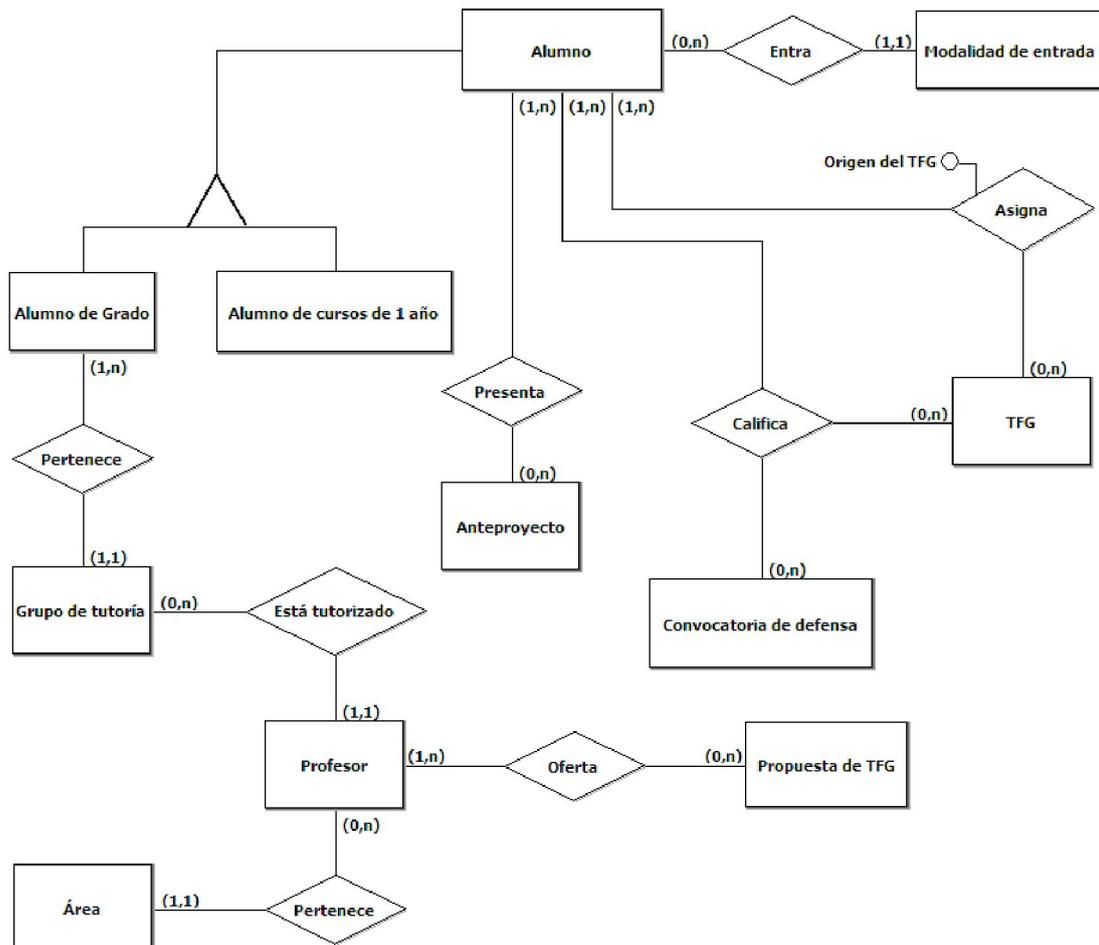


Figura 4A: Esquema entidad-relación (modelo conceptual)

La parte negativa del modelo es que a veces no refleja exactamente la realidad, ya que por ejemplo, la cardinalidad de las relaciones que parten del alumno hacia abajo es (1, 2) en todos los casos:

- Un anteproyecto es presentado por 1 ó 2 alumnos
- Un TFG se asigna a 1 ó 2 alumnos
- En una convocatoria de defensa, se califica a 1 ó 2 alumnos

Igualmente, una propuesta de TFG está ofertada por 1 ó 2 tutores. Sin embargo, esta pérdida de semántica no es tan grave, ya que el proceso de modelado pretende contemplar todas las posibilidades, no a los casos concretos que se puedan dar.

## 4.2. Esquema Relacional

Después de obtener el modelo conceptual, se transforma en modelo lógico, que es un modelo más depurado y basado en el paradigma de modelado que se use en la base de datos. Así, si se tiene una base de datos relacional, el modelo entidad-relación se transforma en modelo relacional.

Algunas herramientas, como brModelo, proveen una función para pasar del modelo conceptual al lógico, pero el resultado no es siempre el adecuado y hay que realizar retoques. Aquí tenemos un esquema relacional preliminar generado con brModelo, parte de la lógica de la solución ya implementada gracias a algunos retoques, y con algunos campos ya visibles:

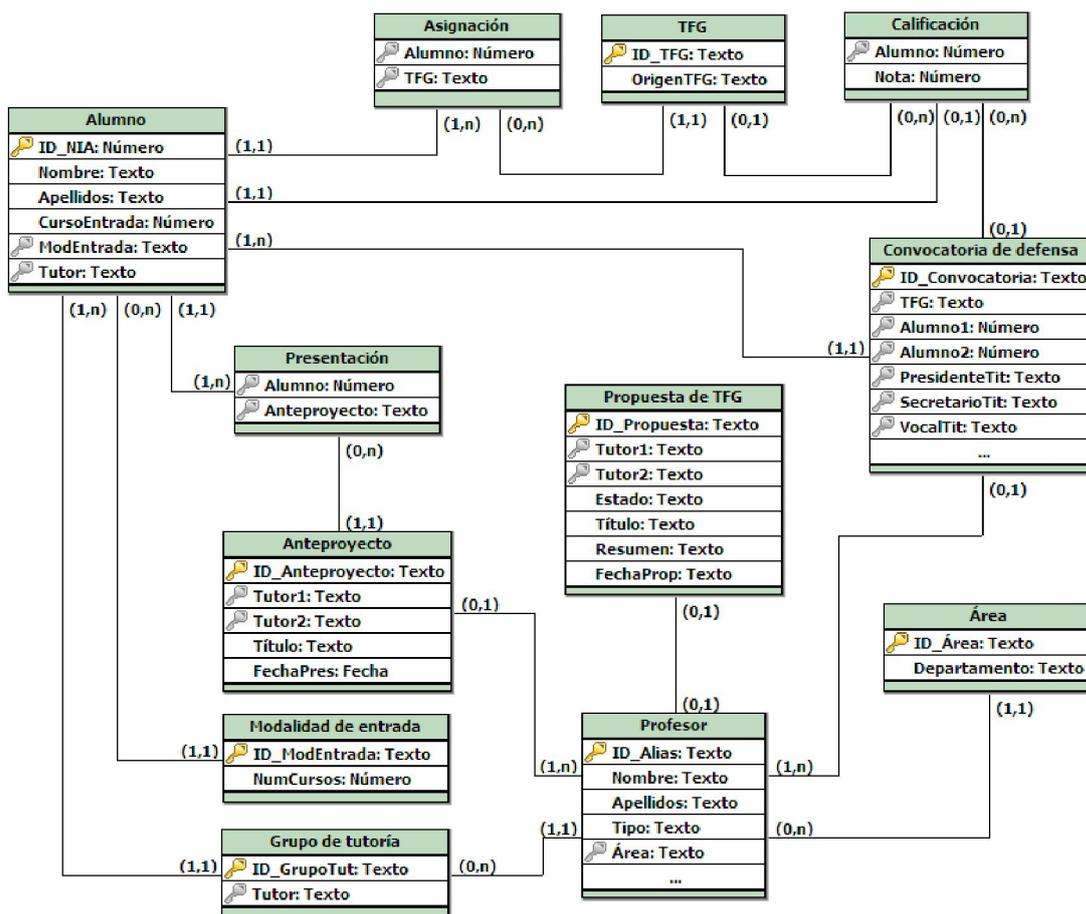


Figura 4B: Borrador del esquema relacional (modelo lógico)

Este esquema sufrió modificaciones para contemplar todas las posibilidades que podrían surgir en la situación real, de tal forma que se hizo más complejo. Además, en el paso al modelo relacional, se aplican ciertos criterios que permiten obtener una base de datos sin redundancias e incoherencias, mediante la normalización y las formas normales.

En este caso, al aplicar algunos de estos criterios, se pierde la semántica de la situación que se pretende modelizar, por lo que se intentó preservar la semántica en la medida de lo posible.

El modelo siguió desarrollandose hasta obtener versiones más completas y refinadas, y aquí tenemos el esquema relacional final, tal y como se ve en Access, para comentar un ejemplo de lo dicho anteriormente.

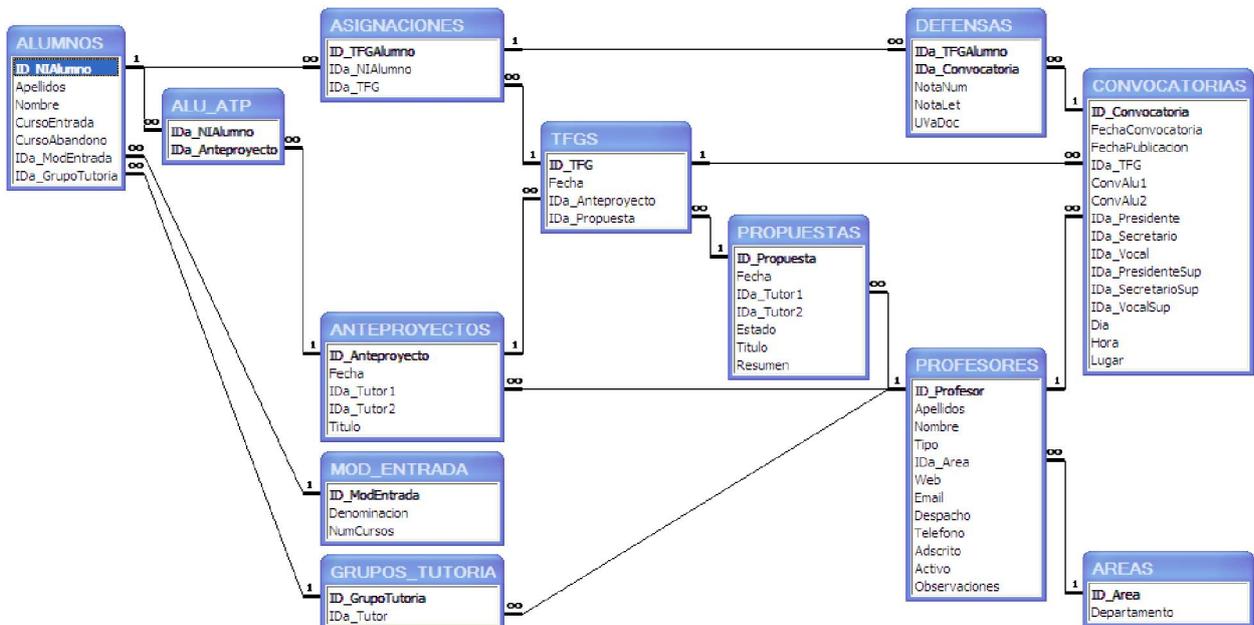


Figura 4C: Esquema relacional (modelo lógico)

La tabla DEFENSAS no tiene clave principal única, y de hecho su clave principal esta compuesta por las claves de otras tablas, por lo que en el proceso de normalización, esa tabla desaparecería, y sus campos viajarían a la tabla de CONVOCATORIAS, por ejemplo como campos duplicados para guardar la nota de cada alumno.

Sin embargo, se ha creído apropiado este modelo, ya que un TFG puede ser defendido por sus autores por separado, con lo cual cada alumno aparecerá en una convocatoria distinta, y la semántica se respeta más diciendo que las defensas son una relación entre las asignaciones individuales del TFG (llamadas TFG-Alumno) y las convocatorias de defensa.

Algo parecido pasa con la tabla ALU\_ATP, cuya existencia en el modelo final se debe a que refleja mejor la semántica de que los alumnos presentan anteproyectos, y esa tabla contiene la relación de alumnos y anteproyectos presentados.

Se podrían pasar los identificadores de los dos alumnos que lo presentan a la tabla ANTEPROYECTOS como campos duplicados, pero en ese caso, no queda muy bien reflejada la semántica, ya que un anteproyecto puede tener 1 ó 2 autores.

Así, el modelo está sujeto también a las consideraciones subjetivas que una u otra persona puedan tener al respecto, pero lo importante es que se adapte bien a la situación que pretende reflejar.

### 4.3. Diccionario de datos

En este apartado se habla del modelo físico tal y como se ha creado finalmente, es decir, del final del proceso de modelado. Presentaremos cada tabla del modelo lógico, especificada a nivel físico.

**Tabla ALUMNOS**

Nombre	Tipo	Requerido	Referencia
<b>ID_NIAlumno</b>	Entero largo	Sí	
	Clave principal: Campo numérico que identifica a un alumno inequívocamente		
Apellidos	Texto (131)	Sí	
	Campo de texto que guarda los 2 apellidos del alumno		
Nombre	Texto (68)	Sí	
	Campo de texto que guarda el nombre del alumno		
CursoEntrada	Entero	Sí	
	Campo numérico que guarda el año de entrada del alumno		
CursoAbandono	Entero	No	
	Campo numérico que guarda el año de abandono del alumno, si se ha producido		
<b>IDa_ModEntrada</b>	Texto (1)	Sí	<b>MOD_ENTRADA</b>
	Clave ajena: Campo de un solo caracter que referencia a la modalidad de entrada del alumno		
<b>IDa_GrupoTutoria</b>	Texto (16)	Sí	<b>GRUPOS_TUTORIA</b>
	Clave ajena: Campo de texto que referencia al grupo de tutoría del alumno, aunque no tenga tutor asignado (para esos casos existe un grupo de tutoría sin tutor asignado)		

**Tabla MOD\_ENTRADA**

Nombre	Tipo	Requerido	Referencia
<b>ID_ModEntrada</b>	Texto (1)	Sí	
	Clave principal: Campo de un solo caracter que identifica una modalidad de entrada para los alumnos		
Denominacion	Texto (68)	Sí	
	Campo de texto que guarda el nombre de la modalidad de entrada		
NumCursos	Byte	Sí	
	Campo numérico que guarda el número de cursos de la modalidad de entrada		

Tabla GRUPOS\_TUTORIA

Nombre	Tipo	Requerido	Referencia
<b>ID_GrupoTutoria</b>	Texto (16)	Sí	
	Clave principal: Campo de texto que identifica un grupo de tutoría, con la forma GRT.●●●●-●●●●●●		
<b>IDa_Tutor</b>	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es tutor del grupo de tutoría		

Tabla ANTEPROYECTOS

Nombre	Tipo	Requerido	Referencia
<b>ID_Anteproyecto</b>	Texto (16)	Sí	
	Clave principal: Campo de texto que identifica un anteproyecto, con la forma ATP.●●●●-●●●●●●		
Fecha	Fecha/Hora	Sí	
	Campo de tipo fecha que guarda la fecha en la que se presenta el anteproyecto		
<b>IDa_Tutor1</b>	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el tutor del anteproyecto		
<b>IDa_Tutor2</b>	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el segundo tutor del anteproyecto, aunque no exista (para esos casos, existe un tutor en la tabla de profesores llamado "No asignado")		
Título	Texto (131)	Sí	
	Campo de texto que guarda el título del anteproyecto		

Tabla ALU\_ATP

Nombre	Tipo	Requerido	Referencia
<b>IDa_NIAlumno</b>	Entero largo	Sí	<b>ALUMNOS</b>
	Clave principal: Campo numérico que identifica a un alumno que ha presentado un anteproyecto Clave ajena: Referencia a dicho alumno		
<b>IDa_Anteproyecto</b>	Texto (16)	Sí	<b>ANTEPROYECTOS</b>
	Clave principal: Campo de texto que identifica el anteproyecto presentado por el alumno Clave ajena: Referencia a dicho anteproyecto		

Tabla PROFESORES

Nombre	Tipo	Requerido	Referencia
<b>ID_Profesor</b>	Texto (32)	Sí	
	Clave principal: Campo de texto que identifica a un profesor (es un alias)		
Apellidos	Texto (131)	Sí	
	Campo de texto que guarda los 2 apellidos del profesor		
Nombre	Texto (68)	Sí	
	Campo de texto que guarda el nombre del profesor		
Tipo	Texto (5)	No	
	Campo de texto que guarda el tipo del profesor (PTUN, PTEU, CDOC, AYUD,...), si se sabe el tipo		
<b>IDa_Area</b>	Texto (5)	Sí	<b>AREAS</b>
	Clave ajena: Campo de texto que referencia al área del profesor (ATC, INF, LSI,...)		
Web	Texto (131)	No	
	Campo de texto que guarda la dirección URL de la página web del profesor, si tiene		
Email	Texto (68)	No	
	Campo de texto que guarda la dirección de correo electrónico del profesor, si tiene		
Despacho	Texto (5)	No	
	Campo de texto que guarda el identificador del despacho del profesor, si tiene		
Teléfono	Texto (9)	No	
	Campo de texto que guarda el teléfono del despacho del profesor, si tiene, con la forma 921 ●●-●●●●		
Adscrito	Sí/No	Sí	
	Campo booleano que guarda si el profesor está adscrito		
Activo	Sí/No	Sí	
	Campo booleano que guarda si el profesor está activo actualmente en el centro universitario		
Observaciones	Texto (131)	No	
	Campo de texto que guarda cualquier consideración que pueda existir sobre el profesor		

Tabla AREAS

Nombre	Tipo	Requerido	Referencia
<b>ID_Area</b>	Texto (5)	Sí	
	Clave principal: Campo de texto que identifica un área		
Departamento	Texto (131)	Sí	
	Campo de texto que guarda el departamento en el que se inscribe el área		

Tabla PROPUESTAS

Nombre	Tipo	Requerido	Referencia
<b>ID_Propuesta</b>	Texto (16)	Sí	
	Clave principal: Campo de texto que identifica una propuesta de TFG, con la forma OFT.●●●●-●●●●●●		
Fecha	Fecha/Hora	Sí	
	Campo de tipo fecha que guarda la fecha en la que se oferta la propuesta de TFG		
<b>IDa_Tutor1</b>	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el primer tutor de la propuesta de TFG		
<b>IDa_Tutor2</b>	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el segundo tutor de la propuesta de TFG, si existe (si no, se elige el tutor llamado "No asignado")		
Estado	Texto (32)	Sí	
	Campo de texto que guarda el estado de la propuesta de TFG (disponible, no disponible, renovada,...)		
Título	Texto (131)	Sí	
	Campo de texto que guarda el título de la propuesta de TFG		
Resumen	Memo	No	
	Campo de texto extenso (65536 caracteres) que guarda un resumen de la propuesta de TFG		

Tabla TFGS

Nombre	Tipo	Requerido	Referencia
<b>ID_TFG</b>	Texto (16)	Sí	
	Clave principal: Campo de texto que identifica un TFG, con la forma TFG.●●●●-●●●●●●		
Fecha	Fecha/Hora	Sí	
	Campo de tipo fecha que guarda la fecha en la que se produce la asignación del TFG a un alumno o dos, y que está guardada en la tabla ASIGNACIONES		
<b>IDa_Anteproyecto</b>	Texto (16)	No	<b>ANTEPROYECTOS</b>
	Clave ajena: Campo de texto que referencia al anteproyecto del cual proviene el TFG, si es que es así		
<b>IDa_Propuesta</b>	Texto (16)	No	<b>PROPUESTAS</b>
	Clave ajena: Campo de texto que referencia a la propuesta de TFG de la cual proviene el TFG, si es que es así		

Tabla ASIGNACIONES

Nombre	Tipo	Requerido	Referencia
<b>ID_TFGAlumno</b>	Texto (18)	Sí	
	Clave principal: Campo de texto que identifica la asignación individual de un alumno a un TFG, con la forma TFG.●●●●-●●●●●●-● si el TFG tiene dos autores, y con la forma TFG.●●●●-●●●●●● si tiene un único autor.		
<b>IDa_NIAlumno</b>	Entero largo	Sí	<b>ALUMNOS</b>
	Clave ajena: Campo numérico que referencia al alumno al que se le ha asignado el TFG		
<b>IDa_TFG</b>	Texto (16)	Sí	<b>TFGS</b>
	Clave ajena: Campo de texto que referencia al TFG que se ha asignado al alumno		

Tabla CONVOCATORIAS

Nombre	Tipo	Requerido	Referencia
<b>ID_Convocatoria</b>	Texto (16)	Sí	
	Clave principal: Campo de texto que identifica una convocatoria de defensa de TFG, con la forma CNV.●●●●-●●●●●●		
FechaConvocatoria	Fecha/Hora	Sí	
	Campo de tipo fecha que guarda la fecha en la que se produce la convocatoria de defensa		

FechaPublicacion	Fecha/Hora	No	
	Campo de tipo fecha que guarda la fecha en la que se publica la convocatoria de defensa, si se conoce ese dato		
IDa_TFG	Texto (16)	Sí	<b>TFGS</b>
	Clave ajena: Campo de texto que referencia al TFG que se convoca para ser defendido		
ConvAlu1	Sí/No	Sí	
	Campo booleano que guarda si el primer alumno del TFG está convocado en la convocatoria de defensa		
ConvAlu2	Sí/No	Sí	
	Campo booleano que guarda si el segundo alumno del TFG está convocado en la convocatoria de defensa		
IDa_Presidente	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el presidente titular del tribunal		
IDa_Secretario	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el secretario titular del tribunal		
IDa_Vocal	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el vocal titular del tribunal		
IDa_PresidenteSup	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el presidente suplente del tribunal		
IDa_SecretarioSup	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el secretario suplente del tribunal		
IDa_VocalSup	Texto (32)	Sí	<b>PROFESORES</b>
	Clave ajena: Campo de texto que referencia al profesor que es el vocal suplente del tribunal		
Dia	Fecha/Hora	Sí	
	Campo de tipo fecha que guarda la fecha (el día) en la que se defenderá el TFG		
Hora	Fecha/Hora	No	
	Campo de tipo fecha que guarda la hora a la que están convocados los alumnos para defender el TFG, si se quiere añadir ese dato		
Lugar	Texto (68)	No	
	Campo de texto que guarda el lugar en el que se producirá la defensa del TFG, si se quiere añadir ese dato		

Tabla DEFENSAS

Nombre	Tipo	Requerido	Referencia
<b>IDa_TFGAlumno</b>	Texto (18)	Sí	<b>ASIGNACIONES</b>
	Clave principal: Campo de texto que identifica la asignación individual de un TFG a un alumno para calificar su defensa del TFG Clave ajena: Referencia a dicha asignación individual		
<b>IDa_Convocatoria</b>	Texto (16)	Sí	<b>CONVOCATORIAS</b>
	Clave principal: Campo de texto que identifica a la convocatoria por la que se ha producido la defensa de TFG que se está calificando Clave ajena: Referencia a dicha convocatoria		
NotaNum	Simple	Sí	
	Campo numérico que guarda la nota del alumno en la defensa de su TFG con un único decimal		
NotaLet	Texto (32)	Sí	
	Campo de texto que guarda la nota del alumno en la defensa de su TFG con letra (Aprobado, Notable,...)		
UVaDoc	Sí/No	Sí	
	Campo booleano que guarda si el TFG se encuentra disponible a través del sistema UVaDoc		

## 5. PRUEBAS

La fase de pruebas tiene por objetivo asegurar la corrección y fiabilidad del sistema, y hacer que se cumpla rigurosamente el comportamiento descrito por los casos de uso.

Las pruebas realizadas a la aplicación siempre van a depender del momento en el que se encuentra el desarrollo. Así, en las etapas previas a la implementación, es importante preguntarse si el modelo contempla todas las posibilidades que pueden surgir en el mundo real, con preguntas del tipo “*what if...?*”), como por ejemplo:

- ¿Qué pasa si un alumno presenta un anteproyecto, pero luego no llega a defender un TFG, y al año siguiente se acoge a una propuesta ofertada por un profesor?
- ¿Qué pasa si un mismo TFG lo quiere defender uno de sus autores en una convocatoria y otro autor en otra convocatoria distinta?
- ¿Qué pasa si un alumno suspende en una primera defensa y se le vuelve a convocar para una segunda defensa?

Las respuestas a estas preguntas son las que llevan a moldear el modelo obtenido para llegar a una solución que evite esos contratiempos.

Es posible que a este tipo de métodos no se le puedan llamar pruebas, pero su función es evitar un comportamiento anómalo de la aplicación final, e identificar errores de diseño, o un análisis menos exhaustivo de lo necesario; en resumen, la finalidad es obtener una aplicación que responda bien ante todas las situaciones posibles que se puedan dar.

Lo que si es evidente es que en la etapa de implementación se deben realizar pruebas a distintos niveles.

### **5.1 Pruebas parciales**

Al desarrollar un formulario, si no depende de otros, es importante probar su funcionalidad, aunque sea reducida, ya que es más fácil detectar un error en un subsistema pequeño que en el sistema global.

Además las pruebas sobre un único subsistema son más extensivas, cubriendo mayor parte del espectro de circunstancias que pueden aparecer.

Las pruebas que se suelen realizar usualmente son las de **caja negra**. Para ello, se seleccionan distintos casos de prueba, tanto mediante técnicas de partición equivalente como de análisis límite.

En las técnicas de **partición equivalente**, se determina el rango de valores que se puede introducir en determinado momento, y se divide ese rango en particiones equivalentes. Luego se toma un valor representativo de cada partición y se prueba.

Por ejemplo: Para la edad de una persona, dividimos el conjunto de valores que puede tomar en dos particiones: números negativos y números positivos. Y luego, tomamos el número -1 como representativo de la primera partición y probamos a introducir ese número como caso de prueba.

Es un ejemplo sencillo, pero la realidad no es muy distinta, ya que el curso de entrada de un alumno puede tener un rango de valores, y podríamos partirlo en 5 particiones distintas:

- Desde  $-\infty$  a 2001 (el campus de la UVa no existía en Segovia)
- Desde 2002 a 2009 (no existían estudios de grado)
- Desde 2010 a 2049 (periodo de validez de la aplicación)
- Desde 2050 a 2099 (periodo de aviso por el uso de una aplicación tan antigua, se debería migrar a otro sistema)
- Desde 2100 a  $+\infty$  (aplicación obsoleta)

Así, podríamos elegir como valores de prueba representativos el 1234, 2005, 2015, 2075 y 2345.

Por otra parte, es conveniente probar la aplicación introduciendo datos de otro tipo distinto al esperado, como por ejemplo una letra o una fecha, y observar como reacciona ante ello.

En las técnicas de **casos límite** lo que se determinan son los valores que pueden dar problemas al estar situados entre un rango y el siguiente de las particiones de equivalencia.

Por ejemplo, en el caso anterior, los casos de prueba serían el 2009, 2010, 2049 y 2050 para ver cómo reacciona la aplicación al introducir esos valores.

También se hacen pruebas de **caja blanca**, en las que se observa el código directamente para detectar esos casos límite, mirando los bucles para ver si tratan todos los casos de forma correcta o dejan algún caso sin comprobar, como por ejemplo  $0\dots n$ ,  $1\dots n$ ,  $0\dots n-1$  ó  $1\dots n-1$ .

Así también se pueden obtener casos de prueba que intenten descubrir nuevos errores de programación.

En todo caso, la realización de pruebas de ejecución del sistema sobre los casos límite suele ser la forma más rápida para encontrar errores, y por tanto, la primera prueba masiva de búsqueda de errores se basa en los casos límite.

Por otra parte, excluyendo los casos límite, el diseño de los casos de prueba suele ser subjetivo. Cada programador tiende a probar partes del código en un sentido determinado y nunca se llegará a tener un sistema totalmente correcto y libre de errores.

Una vez detectado un error se recurre a la fase de **depuración**. En esta fase, y a partir de la información que aporta el caso de prueba, se selecciona la parte del código donde se supone que se localiza el error y se ejecuta el programa paso a paso (o línea a línea). Al ejecutar la aplicación paso a paso, se puede seguir la pista de la ejecución de las instrucciones y observar cómo cambian los datos según se ejecutan dichas instrucciones. Así, se puede saber si una instrucción modifica los datos de forma incorrecta.

Finalmente, en una aplicación de base de datos en la que se lanzan instrucciones SQL compuestas de cadenas literales y variables que contienen datos introducidos por el usuario, es indispensable depurar el código paso a paso antes de que se envíe la instrucción al motor de base de datos. Una instrucción mal construida puede introducir datos no válidos.

Otras cuestiones a tener en cuenta con las instrucciones SQL son:

- La inserción de fechas, que llevan un formato propio y la fecha se inserta en el orden mm/dd/aaaa
- La inserción de datos nulos, que se realiza con NULL, y eso no es lo mismo que una cadena vacía.
- La inserción de valores booleanos, que son Yes/No.
- La inserción de números decimales, que llevan el punto decimal.
- La inserción de cualquier campo que contenga un apóstrofe, que puede deformar por completo la instrucción SQL.

## **5.2 Pruebas de integración**

Una vez que se ha terminado el diseño de casos de prueba para partes pequeñas y localizadas del sistema, la siguiente estrategia de prueba es el diseño de pruebas de integración, que consisten en probar el sistema completo o, partes más extensas del mismo, de tal forma que haya varios subsistemas funcionando conjuntamente.

Cuanto más relacionados entre sí estén los subsistemas, más pruebas de este tipo hay que realizar, dado que unos pueden modificar indebidamente los datos de otros.

En general hay que centrarse más en los primeros subsistemas que se desarrollaron, ya que el posterior desarrollo puede introducir efectos adversos sobre los subsistemas más “antiguos”.

Sin embargo, en el proceso de pruebas, ante la aparición de un error, la causa más probable sea un subsistema de reciente implementación, por lo que las pruebas de integración se entrelazan a veces con las pruebas parciales y no se debe dejar de probar la aplicación de ambas formas.

Finalmente, por muchas pruebas que se realicen, al ser sobre todo basadas en criterios subjetivos, es imposible garantizar el funcionamiento sin errores de ninguna aplicación, y SIGIISA-UVa, en particular, se distribuye sin ninguna garantía. De hecho, casi todas las licencias de distribución de software contemplan dicha ausencia o exclusión de garantía en varios sentidos.

### **Errores encontrados**

En la aplicación se fueron encontrando errores de diverso tipo, posteriormente corregidos, entre los cuales se pueden comentar:

- Que se permitiera la inserción en la base de datos del apóstrofe de un campo.
- Que se permitiera introducir como primer y segundo alumno de un TFG a la misma persona. Igualmente, que se permitiera introducir como primer y segundo tutor de un TFG a la misma persona.
- Que en una convocatoria de defensa, se permitiera no convocar a ninguno de los autores del TFG

### **Errores no solucionados**

En Access, la ruleta del ratón sirve para cambiar de registro, y por defecto, los formularios se establecen sobre varios registros. Sin embargo, en la aplicación siempre se ve sólo un registro a la vez, pero si se usa la ruleta, se vacían los campos para escribir un nuevo registro, lo cual es absurdo en un formulario tal y como están creados. Se ha encontrado una solución a este problema, pero demasiado tarde como para aplicarla y probar que no introduce efectos adversos para el resto de aplicación.



**[SIGIISA-UVa]**

**SECCIÓN 3:**

**MANUAL DE USUARIO**



**Universidad de Valladolid**



## 1. REQUERIMIENTOS DEL SISTEMA

Se han definido requerimientos de hardware y de software para el correcto funcionamiento de la aplicación SIGIISA-UVa.

### Hardware

Vienen dados por los requisitos para instalar y ejecutar Microsoft Office 2003

- Cualquier procesador capaz de ejecutar instrucciones x86 y con frecuencia de 233 MHz o superior, como:
  - Intel Pentium 4
  - Intel Core 2 Duo
  - Intel Core i3/i5/i7 de cualquier generación
  - Intel Pentium Dual Core
  - Intel Atom
  - AMD AthlonXP
  - AMD Athlon64 / Phenom64 (X2 / X3/X4)
  - AMD Athlon II / Phenom II
  - AMD Sempron
  - AMD Fusion
- Al menos 128MB de memoria RAM
- Al menos 400MB de disco duro libre
- Una resolución de pantalla de 1024x600 o superior

### Software

- Cualquier sistema operativo de la familia Microsoft Windows de la familia Win32 o compatible con ella, superior a Windows 2000 SP3:
  - Windows 2000 SP3
  - Windows XP
  - Windows Vista
  - Windows 7
  - Windows 8 (en modo de compatibilidad)
  - Windows 8.1 (en modo de compatibilidad)
- La suite ofimática Microsoft Office edición Professional con el componente Access instalado, versión 2003 o superior:
  - Office 2003 Professional
  - Office 2007 Professional
  - Office 2010 Professional
  - Office 2013 Professional

**IMPORTANTE:** Cualquier modificación hecha en el diseño de la base de datos o los formularios con una versión de Office superior a la 2003 podría convertir la aplicación a un formato no compatible con Access 2003.

Por tanto, se recomienda ejecutar la aplicación sólo en Access 2003.

## 2. MANUAL DE INSTALACIÓN

SIGIISA-UVa no necesita instalación propiamente dicha, ya que basta con descomprimir el archivo que viene incluido en el CD-ROM para poder ejecutar la aplicación.

Sin embargo, es conveniente incluir unos pasos:

1. Insertar el CD-ROM en la unidad lectora
2. Abrir el contenido del CD-ROM y localizar el archivo SIGIISA-UVa.zip



SIGIISA-UVa.zip

3. Abrir el archivo comprimido y descomprimirlo por completo en una ruta del ordenador en la que existan permisos de escritura.
4. Abrir la carpeta de extracción, localizar el archivo SIGIISA-UVa.mdb y enviar un acceso directo del mismo al escritorio.

### Recomendaciones

El archivo comprimido contiene 3 ficheros en su interior:

- **Licencia.txt**: Contiene una breve licencia de descarga de responsabilidades.
- **SIGIISA-UVa.ico**: Es el icono que aparece en la esquina superior izquierda de la aplicación al abrirla, y en la barra de tareas de Windows en cada ventana de SIGIISA-UVa que esté abierta
- **SIGIISA-UVa.mdb**: Es la aplicación en sí y es el único fichero realmente necesario de los tres

Puesto que se van a descomprimir 3 archivos, es recomendable descomprimirlos en una carpeta nueva.

Por otra parte, como ya se ha dicho, es necesario que la carpeta tenga permisos de escritura para el usuario, por lo que no se recomienda usar la estructura de carpetas propias de Windows para descomprimir SIGIISA-UVa. En lugar de ello, debería descomprimirse en nuevas carpetas creadas dentro de las carpetas personales, como:

- Mis Documentos o Documentos
- Escritorio

Finalmente, dado que SIGIISA-UVa compacta su estructura de datos al cerrar la aplicación, se recomienda encarecidamente que la descompresión de los tres archivos se realice en un disco duro, en lugar de una unidad extraíble como un PenDrive. De hacerlo en un PenDrive, no se notaría una velocidad más lenta para la aplicación, sólo que SIGIISA-UVa tardaría un tiempo mayor en cerrarse.

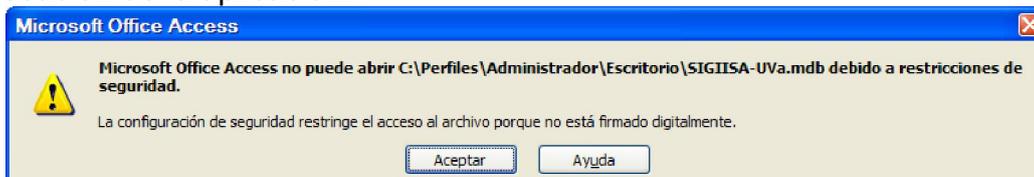
### 3. MANUAL DE USO

En este apartado se muestran las imágenes que puede ver el usuario en cada formulario de la aplicación SIGIISA-UVa, y explicaciones sobre su funcionamiento.

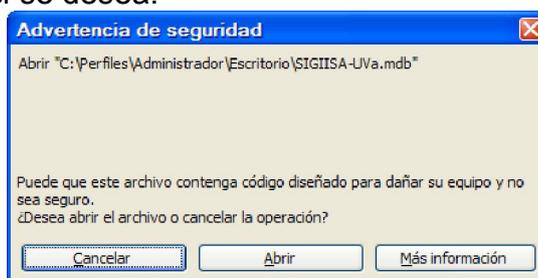
#### 3.1 Preguntas de seguridad

Al abrir la aplicación, pueden aparecer distintas advertencias relacionadas con el nivel de seguridad de Access, que por defecto no permite la ejecución de código o si lo hace, necesita la aceptación por parte del usuario.

Así, se pueden encontrar una ventana como la siguiente, que impediría la ejecución de la aplicación:

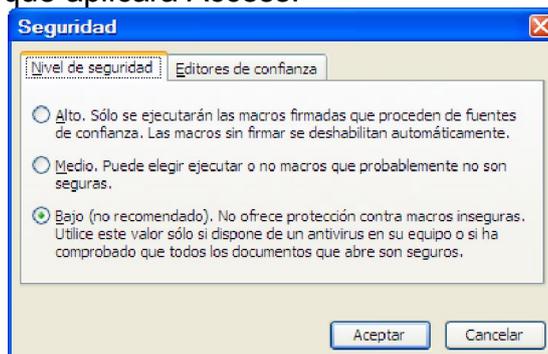


Aunque la ventana más típica es una de advertencia, pero con posibilidad de abrir la aplicación si se desea:



La solución para ambos problemas es entrar en el menú de “Herramientas” de Access, submenú “Macro” y opción “Seguridad...”.

Al hacer clic, aparece un diálogo de configuración que permite cambiar el nivel de seguridad que aplicará Access:



La opción recomendada es nivel medio, por precaución, pero si no se desean advertencias al abrir SIGIISA-UVa, es necesario elegir el nivel bajo.

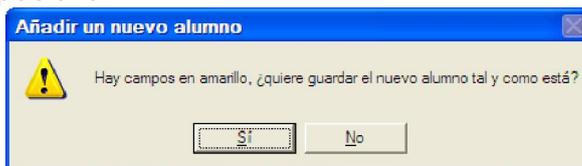
## 3.2 Comportamientos comunes

Todos los formularios de datos de SIGIISA-UVa están configurados para colorear sus campos en función de su contenido, con 3 códigos de color:

- **Blanco:** El contenido del campo es válido y común
- **Amarillo:** El contenido no es común. Por ejemplo:
  - El NIA de un alumno suele tener 6 dígitos, por lo que al escribir 5 ó 7, se podría suponer que es un error del usuario y el campo NIA se colorea de amarillo.
  - Un profesor suele tener despacho y número de teléfono, por lo que no informar de ello en el formulario se considera no común, y el campo adquiere un fondo amarillo.
  - El título de un anteproyecto suele contener varios caracteres, por lo que si se escriben pocos, el campo se colorea de amarillo para advertir al usuario.
- **Rojo:** El contenido no es válido.

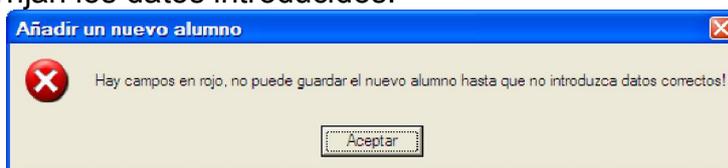
Estos códigos de color sirven para conocer como reaccionará la aplicación:

- Un formulario con todos los campos en blanco permite el guardado de sus datos sin ninguna pérdida de tiempo para el usuario.
- Un formulario con algún campo amarillo y ninguno rojo permite el guardado sólo tras hacer una pregunta al usuario y obtener una respuesta positiva:



Si se responde que no, el formulario vuelve a su estado anterior, esperando que el usuario corrija los datos introducidos y vuelva a intentar guardarlos.

- Un formulario con algún campo rojo no permite el guardado hasta que se corrijan los datos introducidos.



Finalmente, también existe un código de colores para identificar las distintas gestiones que realiza SIGIISA-UVa. Estos colores se han aplicado al fondo de los formularios, para distinguirlos mejor:

- Azul: Alumnos, modalidades de entrada y anteproyectos
- Verde: Profesores, áreas, grupos de tutoría y propuestas de TFG
- Rojo: Asignaciones de TFG, convocatorias, y defensas
- Amarillo: Informes

### 3.3 Listado de formularios y sus funciones

#### 3.3.1. Menús

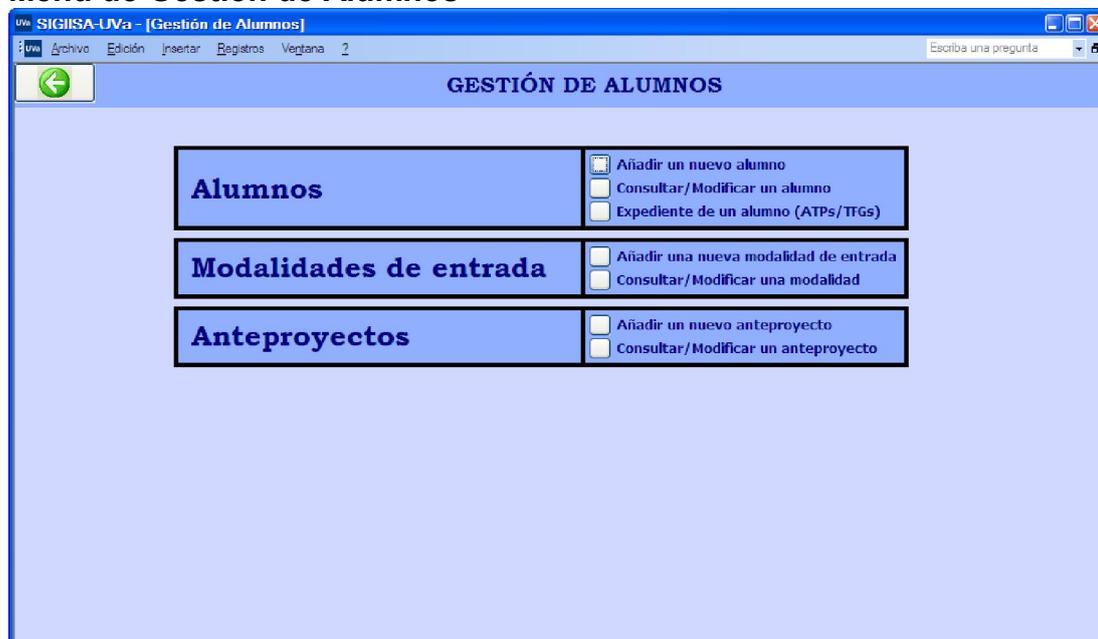
##### Menú principal



Este menú permite el acceso a las gestiones que permite realizar la aplicación, y se puede acceder a ellas mediante el botón cuadrado que se encuentra a la izquierda de cada opción.

También permite cerrar la aplicación con el botón inferior, momento en el que se producirá la compactación de la base de datos.

##### Menú de Gestión de Alumnos

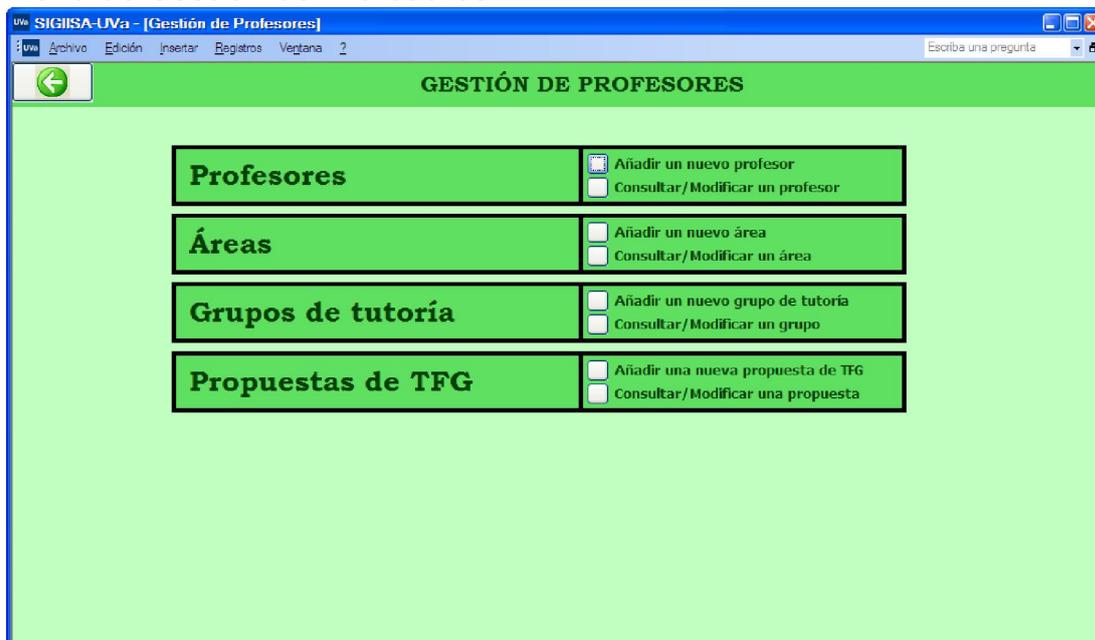


Este menú permite acceder a las funciones específicas para alumnos, haciendo clic en los pequeños botones cuadrados que hay a la izquierda de cada opción de las que se ofrecen:

- Añadir un nuevo alumno
- Consultar/Modificar un alumno existente
- Ver el expediente completo de un alumno
- Añadir una nueva modalidad de entrada
- Consultar/Modificar una modalidad de entrada existente
- Añadir un nuevo anteproyecto
- Consultar/Modificar un anteproyecto existente

También permite volver al menú principal mediante el botón que hay en la parte superior izquierda, o con la tecla ESC (Escape).

### Menú de Gestión de Profesores

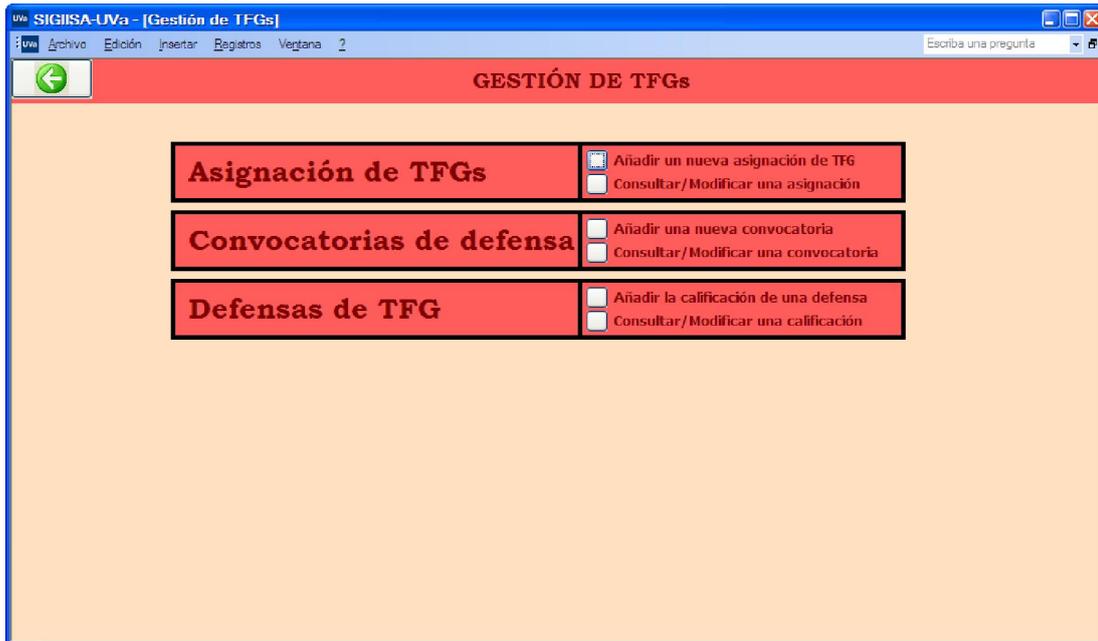


Este menú permite acceder a las funciones específicas para profesores, haciendo clic en los pequeños botones cuadrados que hay a la izquierda de cada opción de las que se ofrecen:

- Añadir un nuevo profesor
- Consultar/Modificar un profesor existente
- Añadir un nuevo área
- Consultar/Modificar un área existente
- Añadir un nuevo grupo de tutoría
- Consultar/Modificar un grupo de tutoría existente
- Añadir una nueva propuesta de TFG
- Consultar/Modificar una propuesta de TFG existente

También permite volver al menú principal mediante el botón que hay en la parte superior izquierda, o con la tecla ESC (Escape).

## Menú de Gestión de TFGs

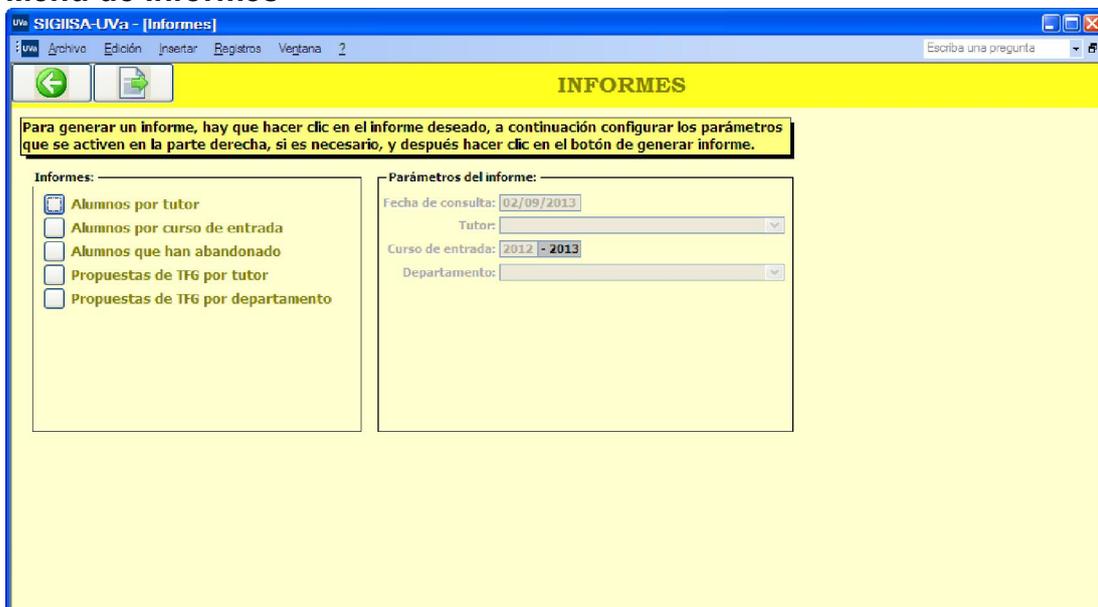


Este menú permite acceder a las funciones específicas para TFGs, haciendo clic en los pequeños botones cuadrados que hay a la izquierda de cada opción de las que se ofrecen:

- Añadir una nueva asignación de TFG
- Consultar/Modificar una asignación de TFG existente
- Añadir una nueva convocatoria de defensa
- Consultar/Modificar una convocatoria de defensa existente
- Añadir la calificación de una defensa de TFG
- Consultar/Modificar la calificación de una defensa de TFG existente

También permite volver al menú principal mediante el botón que hay en la parte superior izquierda, o con la tecla ESC (Escape).

## Menú de Informes



Este menú permite acceder a los informes o listados, según los criterios que se deseen, para poder realizar las siguientes acciones:

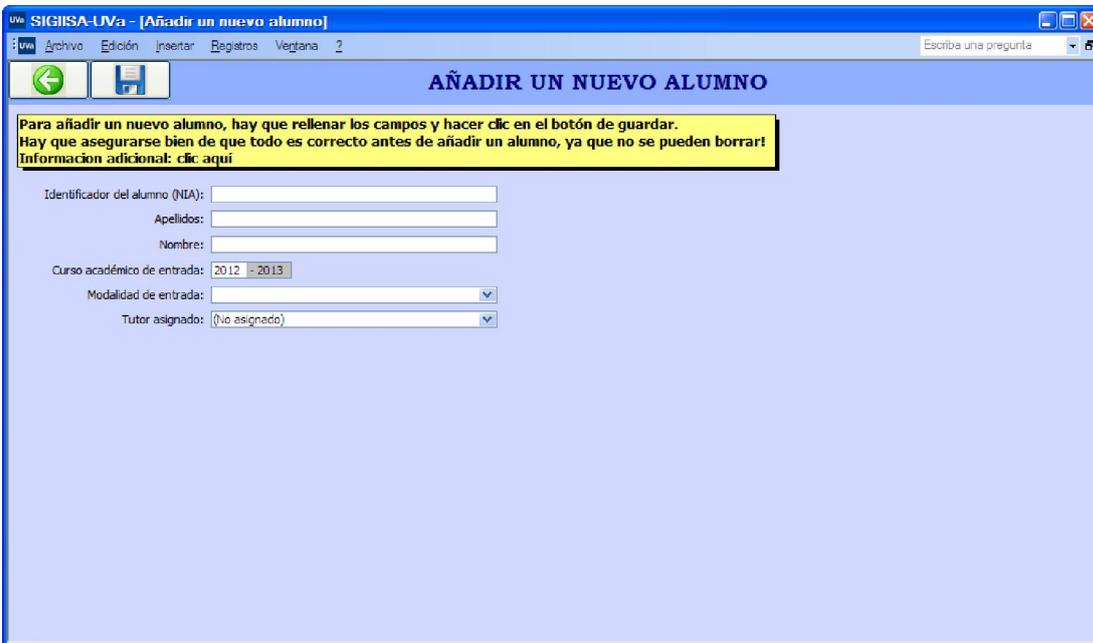
- Ver el listado de alumnos por tutor
- Ver el listado de alumnos por curso de entrada
- Ver el listado de alumnos que han abandonado
- Ver el listado de propuestas de TFG por tutor
- Ver el listado de propuestas de TFG por departamento

También permite volver al menú principal mediante el botón que hay en la parte superior izquierda, o con la tecla ESC (Escape).

Se explicará el funcionamiento del menú de informes en el apartado 3.3.5 de este manual de usuario.

### 3.3.2. Gestión de Alumnos

#### Añadir un nuevo alumno



Este formulario permite añadir un nuevo alumno a la base de datos, introduciendo los siguientes datos:

- **NIA:** Identificador de 6 dígitos.
- **Apellidos:** Los dos apellidos del alumno.
- **Nombre:** El nombre del alumno.
- **Curso académico de entrada:** El curso académico en el que empieza sus estudios el alumno. Al abrir el formulario, se calcula el curso académico actual, pero se puede modificar. Sólo es necesario introducir el primer año, ya que el segundo se autocalcula.
- **Modalidad de entrada:** La modalidad de entrada por la que entra el alumno (1º de Grado, Curso Puente, Erasmus y Pasarela).
- **Tutor asignado:** El tutor del alumno si entra por 1º de Grado. Si no, se deja el tutor como "No asignado".

Una vez introducidos los datos, hay que hacer clic en el botón “**Guardar**” y el nuevo alumno se guardará en la base de datos, excepto si hay un error en los datos, o ya existía un alumno con ese NIA. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, el alumno no se guarda.

### Consultar/Modificar un alumno existente

Este formulario permite consultar los datos de un alumno, buscando dicho alumno mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido a un alumno).

Si se desean modificar los datos del alumno, deben seguirse los mismos criterios que al añadir uno nuevo, aunque en este caso no se permite cambiar el NIA del alumno.

También se permite dejar de considerar al alumno como activo, rellenando el campo del curso de abandono.

Este formulario también permite filtrar los alumnos que aparecen en la lista desplegable para mostrar los alumnos que están activos actualmente (sin baja informada), los que han terminado el TFG, los que han abandonado, o si no se configura nada, mostrar todos.

Finalmente, una vez que se ha elegido un alumno, se permite ver el expediente completo (anteproyectos, asignaciones, convocatorias y defensas de TFG), si procede para el alumno elegido.

Es importante recalcar de nuevo que si se modifica el alumno y no se hace clic en Guardar o se vuelve atrás, el alumno no se guardará.

## Ver el expediente completo de un alumno

**EXPEDIENTE DE UN ALUMNO**

Búsqueda de alumnos:

**Anteproyectos/Propuestas:**  
ATP.2010-2011.15

**Información del anteproyecto/propuesta**

Fecha del anteproyecto o la propuesta:	02/09/2011
Tutor 1:	
Tutor 2:	
Título:	Auditoría Web del Ayuntamiento de Segovia

**TFG asignados:**  
TFG.2010-2011.12

**Información del TFG**

Identificador TFG-Alumno:	TFG.2010-2011.12
Fecha de la asignación del TFG:	02/09/2011
Anteproyecto/Propuesta de origen:	ATP.2010-2011.15
Co-Autor del TFG:	{Autor único}

**Convocatorias de defensa:**  
CIV.2010-2011.12

**Información de la convocatoria**

Fecha de la convocatoria:	07/09/2011
TFG defendido por el alumno:	TFG.2010-2011.12
Presidente:	
Secretario:	
Vocal:	
Fecha y lugar de la defensas:	12/09/2011
Nota obtenida por el alumno:	9,6 (Matriculación de honor)

Este formulario permite tener un vistazo rápido del expediente completo de un alumno:

- Anteproyectos presentados u propuestas aceptadas
- Trabajos Fin de Grado asignados
- Convocatorias de defensa por las que ha sido convocado
- Nota obtenida en la defensa

Este formulario no permite modificar nada, y es el mismo formulario que el que aparece al consultar un alumno y luego intentar ver su expediente.

## Añadir una nueva modalidad de entrada

**AÑADIR UNA NUEVA MODALIDAD DE ENTRADA**

Para añadir una nueva modalidad de entrada, hay que rellenar los tres campos y hacer clic en el botón de guardar. Hay que asegurarse bien de que todo es correcto antes de añadir una modalidad, ya que no se pueden borrar!  
Información adicional: [clic aquí](#)

Identificador de la modalidad:

Denominación de la modalidad:

Número de cursos:

Este formulario permite añadir una nueva modalidad de entrada a la base de datos, introduciendo los siguientes datos:

- **Identificador de la modalidad:** Un único número o letra.
- **Denominación de la modalidad:** El nombre que se le asigna.
- **Número de cursos:** El número de cursos de la modalidad de entrada.

Una vez introducidos los datos, hay que hacer clic en el botón “**Guardar**” y la nueva modalidad se guardará en la base de datos, excepto si hay un error en los datos, o ya existía una modalidad con ese identificador. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, la modalidad no se guarda.

### Consultar/Modificar una modalidad de entrada existente

The screenshot shows a web browser window titled "SIGIISA-UVa - [Consultar/Modificar una modalidad de entrada]". The browser's address bar shows "http://www.uva.es". The page title is "CONSULTAR/MODIFICAR UNA MODALIDAD DE ENTRADA". Below the title, there is a search field labeled "Búsqueda de modalidades:" with the text "Curso Puente" entered. A yellow warning box contains the text: "Para modificar una modalidad de entrada, hay que rellenar los campos y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de modalidad o cerrar. Información adicional: clic aquí". Below the warning box, there are three input fields: "Identificador de la modalidad:" with the value "0", "Denominación de la modalidad:" with the value "Curso Puente", and "Número de cursos:" with a dropdown menu showing the value "1".

Este formulario permite consultar los datos de una modalidad de entrada, buscando dicha modalidad mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido una modalidad).

Si se desean modificar los datos de la modalidad, deben seguirse los mismos criterios que al añadir una nueva, aunque en este caso no se permite cambiar el identificador de la modalidad.

De nuevo, si se modifica la modalidad de entrada y no se hace clic en Guardar o se vuelve atrás, la modalidad no se guardará.

## Añadir un nuevo anteproyecto



The screenshot shows a web browser window titled "SIGIISA-UVa - [Añadir un nuevo anteproyecto]". The browser's address bar shows "uva" and the page title is "AÑADIR UN NUEVO ANTEPROYECTO". A yellow warning box at the top states: "Para añadir un nuevo anteproyecto, hay que rellenar todos los campos y hacer clic en el botón de guardar. Hay que asegurarse bien de que todo es correcto antes de añadir un anteproyecto, ya que no se pueden borrar! Información adicional: clic aquí". Below the warning, the form contains the following fields:

- Identificador autogenerado: ATP.2012-2013.09
- Fecha (dd/mm/aaaa): 12/09/2013
- Alumno 1: [dropdown menu]
- Alumno 2: [dropdown menu]
- Tutor 1: [dropdown menu]
- Tutor 2: (No asignado) [dropdown menu]
- Título: [text input field]

Este formulario permite añadir un nuevo anteproyecto a la base de datos.

El identificador del anteproyecto se autogenera en base a la fecha introducida, así que hay que introducir los demás datos:

- **Fecha:** Fecha en la que se presenta el anteproyecto. Al abrir el formulario, aparece la fecha actual, pero se puede modificar.
- **Alumno 1:** El primer autor del anteproyecto.
- **Alumno 2:** El segundo autor del anteproyecto, si existe. Si no, debe dejarse en blanco.
- **Tutor 1:** El primer tutor del anteproyecto.
- **Tutor 2:** El segundo tutor del anteproyecto, si existe. Si no, debe dejarse como "No asignado".
- **Título:** El título del anteproyecto presentado

Una vez introducidos los datos, hay que hacer clic en el botón "**Guardar**" y el nuevo anteproyecto se guardará en la base de datos, excepto si hay un error en los datos. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, el anteproyecto no se guarda en la base de datos.

## Consultar/Modificar un anteproyecto existente

Este formulario permite consultar los datos de un anteproyecto, buscando dicho anteproyecto mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido un anteproyecto).

Si se desean modificar los datos del anteproyecto, deben seguirse los mismos criterios que al añadir uno nuevo, aunque en este caso no se permite cambiar el identificador del anteproyecto. Es importante recalcar de nuevo que si se modifica el anteproyecto y no se hace clic en Guardar o se vuelve atrás, el anteproyecto no se guardará.

### 3.3.3. Gestión de Profesores

#### Añadir un nuevo profesor

Este formulario permite añadir un nuevo profesor a la base de datos, introduciendo los siguientes datos:

- **Identificador/alias:** Identificador del profesor, usualmente un alias.
- **Apellidos:** Los dos apellidos del profesor.
- **Nombre:** El nombre del profesor.
- **Tipo:** El tipo del profesor (PTUN, PTEU, CDOC, AYUD,...). Si no se sabe, se puede dejar el campo en blanco, aunque el campo se coloreará de amarillo.
- **Área:** El área del profesor (ATC, INF, LSI,...).
- **Web:** La dirección URL de la página web del profesor. Si no tiene, se puede dejar en blanco.
- **e-mail:** La dirección de correo electrónico del profesor. Si no tiene, se puede dejar el campo en blanco, aunque el campo se coloreará de amarillo.
- **Despacho:** El identificador del despacho del profesor. Si no tiene, se puede dejar el campo en blanco, aunque el campo se coloreará de amarillo.
- **Teléfono:** El teléfono del despacho del profesor. Si no tiene, se puede dejar el campo en blanco, aunque el campo se coloreará de amarillo.
- **Información adicional:** Estas dos casillas se marcarán para designar si el profesor está adscrito y/o activo actualmente.
- **Observaciones:** Campo abierto para cualquier consideración que pueda existir sobre el profesor.

Una vez introducidos los datos, hay que hacer clic en el botón “**Guardar**” y el nuevo profesor se guardará en la base de datos, excepto si hay un error en los datos, o ya existía un profesor con ese identificador. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, el profesor no se guarda.

### Consultar/Modificar un profesor existente

The screenshot shows a web browser window titled "SIGIISA-UVa - [Consultar/Modificar un profesor]". The main heading is "CONSULTAR/MODIFICAR UN PROFESOR". Below the heading, there is a search bar with the text "Búsqueda de profesores: DIAZ GOMEZ, FERNANDO". A yellow warning box contains the text: "Para modificar un profesor, hay que rellenar los campos a cambiar y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de profesor o cerrar. Información adicional: clic aquí". The form fields are as follows: "Identificador/alias del profesor:" (text box with "Fernando"), "Apellidos:" (text box with "Diaz Gomez"), "Nombre:" (text box with "Fernando"), "Tipo:" (text box), "Área:" (dropdown menu with "CCIA"), "Web:" (text box), "e-mail:" (text box), "Despacho:" (text box), "Teléfono:" (text box), "Información adicional:" (checkboxes for "El profesor está adscrito" and "Activo actualmente", both checked), and "Observaciones:" (text area).

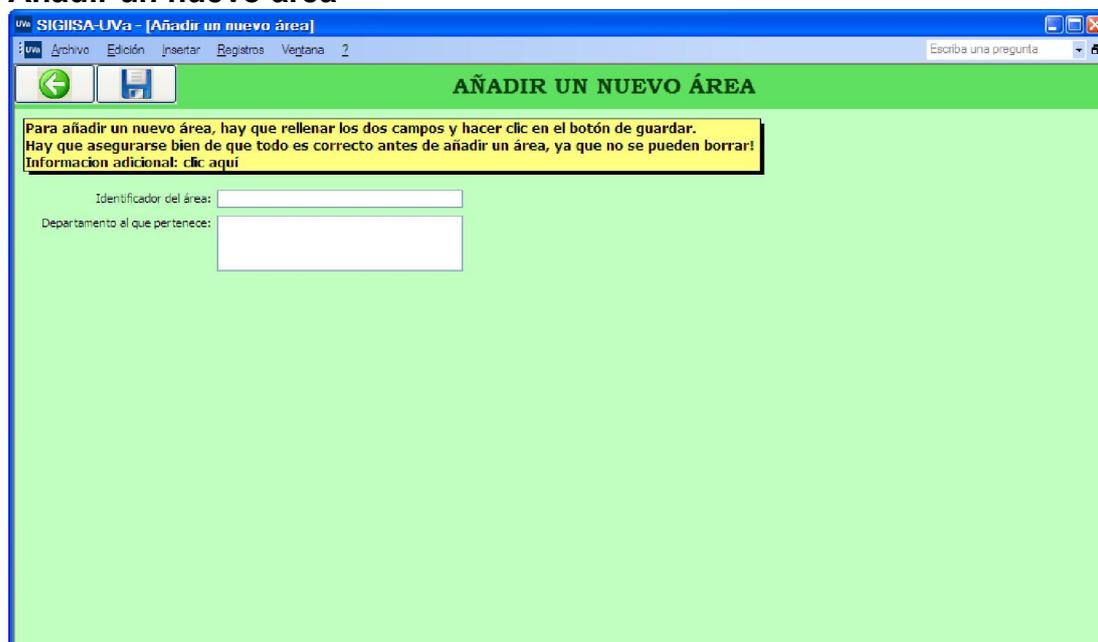
Este formulario permite consultar los datos de un profesor, buscando dicho profesor mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido a un profesor).

Si se desean modificar los datos del profesor, deben seguirse los mismos criterios que al añadir uno nuevo, aunque en este caso no se permite cambiar el identificador/alias del profesor.

También se permite dejar de considerar al profesor como activo, desmarcando la casilla correspondiente. Esto afecta a otros formularios, como por ejemplo el de los anteproyectos, visto anteriormente, en el cual solo aparecen los profesores activos para asignarlos al anteproyecto.

De nuevo, si se modifica el profesor y no se hace clic en Guardar o se vuelve atrás, el profesor no se guardará.

### Añadir un nuevo área



Este formulario permite añadir un nuevo área a la base de datos, introduciendo los siguientes datos:

- **Identificador del área:** Una abreviatura que identifique el área.
- **Departamento:** El departamento en el que se inscribe el área.

Una vez introducidos los datos, hay que hacer clic en el botón “**Guardar**” y el nuevo área se guardará en la base de datos, excepto si hay un error en los datos, o ya existía un área con ese identificador. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, el área no se guarda.

## Consultar/Modificar un área existente

The screenshot shows a web browser window titled 'SIGIISA-UVa - [Consultar/Modificar un área]'. The browser's address bar shows 'uva'. The page has a green header with the title 'CONSULTAR/MODIFICAR UN ÁREA'. Below the header, there is a search bar labeled 'Búsqueda de áreas:' with a dropdown menu showing 'Area de CCIA'. A yellow box with black text contains instructions: 'Para modificar un área, hay que rellenar el departamento y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de área o cerrar. Información adicional: clic aquí'. Below this, there are three input fields: 'Identificador del área:' with the value 'CCIA', 'Departamento al que pertenece:' with the value 'Informática', and 'Profesores de este área:' with the value 'Fernando Diaz Gomez'. There are navigation buttons (back, forward, save) and a search icon in the top left.

Este formulario permite consultar los datos de un área, buscando dicho área mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido un área).

El único dato que se puede modificar en un área es el departamento al que pertenece, ya que no se permite cambiar el identificador del área, y la lista de profesores del área es meramente informativa.

Una vez más, si se modifica el área y no se hace clic en Guardar o se vuelve atrás, el área no se guardará en la base de datos.

## Añadir un nuevo grupo de tutoría

The screenshot shows a web browser window titled 'SIGIISA-UVa - [Añadir un nuevo grupo de tutoría]'. The browser's address bar shows 'uva'. The page has a green header with the title 'AÑADIR UN NUEVO GRUPO DE TUTORÍA'. Below the header, there is a yellow box with black text containing instructions: 'Para añadir un nuevo grupo de tutoría, hay que rellenar los dos campos y hacer clic en el botón de guardar. Hay que asegurarse bien de que todo es correcto antes de añadir un grupo, ya que no se pueden borrar! Información adicional: clic aquí'. Below this, there are three input fields: 'Identificador autogenerado:' with the value 'GRT.2012-2013.07', 'Curso académico del grupo:' with the value '2012 - 2013', and 'Tutor del grupo:' with a dropdown menu. There are navigation buttons (back, forward, save) and a search icon in the top left.

Este formulario permite añadir un nuevo grupo de tutoría a la base de datos.

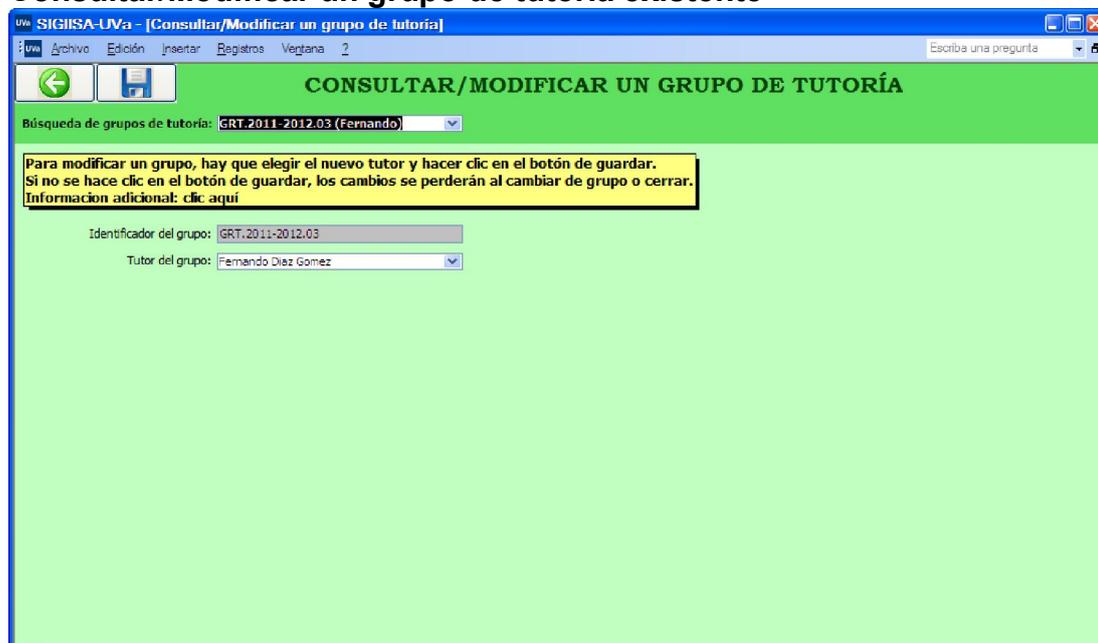
El identificador del grupo de tutoría se autogenera en base al curso académico introducido, así que hay que introducir los demás datos:

- **Curso académico:** El curso para el que se crea el grupo de tutoría. Al abrir el formulario, se calcula el curso académico actual, pero se puede modificar. Sólo es necesario introducir el primer año, ya que el segundo se autocalcula.
- **Tutor del grupo:** El profesor que es tutor del grupo de tutoría.

Una vez introducidos los datos, hay que hacer clic en el botón “Guardar” y el nuevo grupo de tutoría se guardará en la base de datos, excepto si hay un error en los datos. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, el grupo de tutoría no se guarda en la base de datos.

### Consultar/Modificar un grupo de tutoría existente



The screenshot shows a web browser window with the title "SIGIISA-UVa - [Consultar/Modificar un grupo de tutoría]". The browser's address bar shows "uvva" and the page title is "CONSULTAR/MODIFICAR UN GRUPO DE TUTORÍA". Below the title bar, there is a search field labeled "Búsqueda de grupos de tutoría:" with a dropdown menu showing "GRT.2011-2012.03 (Fernando)". A yellow warning box contains the text: "Para modificar un grupo, hay que elegir el nuevo tutor y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de grupo o cerrar. Información adicional: clic aquí". Below the warning box, there are two input fields: "Identificador del grupo:" with the value "GRT.2011-2012.03" and "Tutor del grupo:" with a dropdown menu showing "Fernando Diaz Gomez".

Este formulario permite consultar los datos de un grupo de tutoría, buscando dicho grupo mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido un grupo de tutoría).

El único dato que se puede modificar en un grupo de tutoría es el tutor del grupo, ya que no se permite cambiar el identificador del grupo de tutoría.

De nuevo, si se modifica el grupo de tutoría y no se hace clic en Guardar o se vuelve atrás, el grupo de tutoría no se guardará.

## Añadir una nueva propuesta de TFG

Para añadir una nueva propuesta de TFG, hay que rellenar todos los campos y hacer clic en el botón de guardar. Hay que asegurarse bien de que todo es correcto antes de añadir una propuesta, ya que no se pueden borrar! Información adicional: clic aquí

Identificador autogenerado: OFT\_2012-2013\_18

Fecha (dd/mm/aaaa): 02/09/2018

Tutor 1: [dropdown]

Tutor 2: (No asignado) [dropdown]

Estado de la propuesta: [dropdown]

Título: [text area]

Resumen de la propuesta: [text area]

Este formulario permite añadir una nueva propuesta de TFG a la base de datos.

El identificador de la propuesta de TFG se autogenera en base a la fecha introducida, así que hay que introducir los demás datos:

- **Fecha:** La fecha en la que se oferta la propuesta de TFG. Al abrir el formulario, aparece la fecha actual, pero se puede modificar.
- **Tutor 1:** El primer tutor de la propuesta de TFG.
- **Tutor 2:** El segundo tutor de la propuesta de TFG, si existe. Si no, debe dejarse como “No asignado”.
- **Estado de la propuesta:** El estado de la propuesta de TFG (disponible, no disponible, renovada,...).
- **Título:** El título de la propuesta de TFG.
- **Resumen:** El resumen de la propuesta de TFG.

Una vez introducidos los datos, hay que hacer clic en el botón “Guardar” y la nueva propuesta de TFG se guardará en la base de datos, excepto si hay un error en los datos. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, la propuesta de TFG no se guarda en la base de datos.

## Consultar/Modificar una propuesta de TFG existente

**CONSULTAR/MODIFICAR UNA PROPUESTA DE TFG**

Búsqueda de propuestas de TFG: **OFT.2011-2012.17: Gestor documental ba...**

**Para modificar una propuesta de TFG, hay que rellenar los campos a cambiar y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de propuesta o cerrar. Información adicional: [clic aquí](#)**

Identificador de la propuesta de TFG:

Fecha (dd/mm/aaaa):

Tutor 1:

Tutor 2:

Estado de la propuesta:

Título:

**Resumen de la propuesta:**  
 El objetivo de este TFG es estudiar la API del servicio de almacenamiento Dropbox y mostrar un ejemplo de utilización mediante la creación de una aplicación (CLI o GUI) denominada Gestor Documental. Este gestor deberá ofrecer la posibilidad de crear/abrir un repositorio documental (colección organizada de documentos), gestionar los documentos del mismo (anexar/eliminar documentos), así como publicar y/o compartir documentos de cada repositorio en función de las facilidades soportadas por la API mencionada. La particularidad del gestor es que el almacenamiento no es local, sino que está externalizado (en la nube) gracias al servicio proporcionado por Dropbox. Se recomienda utilizar Java para su desarrollo.

Este formulario permite consultar los datos de una propuesta de TFG, buscando dicha propuesta mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido una propuesta de TFG).

Si se desean modificar los datos de la propuesta de TFG, deben seguirse los mismos criterios que al añadir una nueva, aunque en este caso no se permite cambiar el identificador de la propuesta de TFG. Es importante recalcar de nuevo que si se modifica la propuesta de TFG y no se hace clic en Guardar o se vuelve atrás, la propuesta no se guardará.

### 3.3.4. Gestión de TFGs

#### Añadir una nueva asignación de TFG

**AÑADIR UNA NUEVA ASIGNACIÓN DE TFG**

**Para añadir una nueva asignación de TFG, hay que rellenar los campos pertinentes y hacer clic en el botón de guardar. Hay que asegurarse bien de que todo es correcto antes de añadir una asignación, ya que no se pueden borrar! Información adicional: [clic aquí](#)**

Identificador autogenerado:

Fecha de asignación (dd/mm/aaaa):

Anteproyecto de origen:

Propuesta de origen:

Alumno 1:

Alumno 2:

TFG-Alumno autogenerado:

Este formulario permite añadir una nueva asignación de TFG a la base de datos.

El identificador de la asignación de TFG se autogenera en base a la fecha introducida, así que hay que introducir los demás datos:

- **Fecha:** La fecha en la que se produce la asignación del TFG a los alumnos que se escriban debajo. Al abrir el formulario, aparece la fecha actual, pero se puede modificar.
- **Anteproyecto de origen:** El anteproyecto del cual proviene el TFG, si es que es así. Si se elige un anteproyecto como origen del TFG, se cargarán automáticamente los alumnos del anteproyecto para la asignación del TFG.
- **Propuesta de origen:** La propuesta de TFG de la cual proviene el TFG, si es que es así.
- **Alumno 1:** El primer autor de la propuesta de TFG.
- **Alumno 2:** El segundo autor de la propuesta de TFG, si existe. Si no, debe dejarse en blanco.

Por otra parte, al elegir o ser cargados los alumnos de la asignación, se autogenera un identificador que designa la asignación individual de un alumno a un TFG (el campo TFG-Alumno).

Una vez introducidos los datos, hay que hacer clic en el botón “Guardar” y la nueva asignación de TFG se guardará en la base de datos, excepto si hay un error en los datos. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, la asignación de TFG no se guarda en la base de datos.

### Consultar/Modificar una asignación de TFG existente

The screenshot shows a web browser window with the title "SIGIISA-UVa - [Consultar/Modificar una asignación de TFG]". The browser's address bar shows "uva Archivio Edición Insertar Registros Verdana 2" and a search box with "Escribe una pregunta". The main content area has a red header with the text "CONSULTAR/MODIFICAR UNA ASIGNACIÓN DE TFG". Below the header, there is a search bar with the text "Búsqueda de TFGs asignados: TFG.2011-2012.08: Software para la imp...". A yellow warning box contains the text: "Para modificar una asignación de TFG, hay que rellenar los campos a cambiar y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de asignación o cerrar. Información adicional: clic aquí". Below the warning box, there are several input fields: "Identificador del TFG:" with the value "TFG.2011-2012.08"; "Fecha de asignación (dd/mm/aaaa):" with the value "28/03/2012"; "Anteproyecto de origen:" with a dropdown menu showing "ATP.2011-2012.07: Software para la implanta..."; "Propuesta de origen:" with a dropdown menu; "Alumno 1:" with a dropdown menu showing "TFG-Alumno: TFG.2011-2012.08-1"; and "Alumno 2:" with a dropdown menu showing "TFG.2011-2012.08-2".

Este formulario permite consultar los datos de una asignación de TFG, buscando dicha asignación mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido una asignación de TFG).

Si se desean modificar los datos de la asignación de TFG, deben seguirse los mismos criterios que al añadir una nueva, aunque en este caso no se permite cambiar el identificador de la asignación de TFG.

Una vez más, si se modifica la asignación de TFG y no se hace clic en Guardar o se vuelve atrás, la asignación no se guardará en la base de datos.

### Añadir una nueva convocatoria de defensa

Este formulario permite añadir una nueva convocatoria de defensa de TFG a la base de datos.

El identificador de la convocatoria de defensa se autogenera en base a la fecha de convocatoria introducida, así que hay que introducir los demás datos:

- **Fecha de la convocatoria:** La fecha en la que se produce la convocatoria de defensa.
- **Fecha de publicación:** La fecha en la que se publica la convocatoria de defensa, si se conoce ese dato. Si no, se puede dejar en blanco.
- **TFG convocado:** El TFG que se convoca para ser defendido. Al elegir un TFG, se cargan los nombres de los alumnos que lo tenían asignado y su identificador de asignación individual (TFG-Alumno), y se activan las casillas de verificación de la parte inferior.
- **Presidente del tribunal:** El profesor que es el presidente titular del tribunal.

- **Secretario del tribunal:** El profesor que es el secretario titular del tribunal.
- **Vocal del tribunal:** El profesor que es el vocal titular del tribunal.
- **Presidente suplente:** El profesor que es el presidente suplente del tribunal, si se ha designado uno. Si no, debe dejarse como “No asignado”.
- **Secretario suplente:** El profesor que es el secretario suplente del tribunal, si se ha designado uno. Si no, debe dejarse como “No asignado”.
- **Vocal suplente:** El profesor que es el vocal suplente del tribunal, si se ha designado uno. Si no, debe dejarse como “No asignado”.
- **Fecha de la defensa:** La fecha en la que se defenderá el TFG.
- **Hora de la defensa:** La hora a la que están convocados los alumnos para defender el TFG, si se conoce. Si no, se puede dejar en blanco.
- **Lugar de la defensa:** El lugar en el que se producirá la defensa del TFG, si se conoce. Si no, se puede dejar en blanco.
- **Alumnos convocados:** Los alumnos convocados para la defensa del TFG. Es posible que en alguna convocatoria, sólo se convoque a uno de los dos autores del TFG.

Una vez introducidos los datos, hay que hacer clic en el botón “Guardar” y la nueva convocatoria de defensa de TFG se guardará en la base de datos, excepto si hay un error en los datos. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, la convocatoria de defensa de TFG no se guarda en la base de datos.

### Consultar/Modificar una convocatoria de defensa existente

The screenshot shows a web browser window with the title "SIGIISA-UVa - [Consultar/Modificar una convocatoria de defensa]". The page content is titled "CONSULTAR/MODIFICAR UNA CONVOCATORIA DE DEFENSA".

At the top, there is a search bar with the text "Búsqueda de convocatorias: CIV.2010-2011.04: Intranet de Gestión...".

A yellow warning box contains the text: "Para modificar una convocatoria de defensa, hay que rellenar los campos a cambiar y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de convocatoria o cerrar. Información adicional: clic aquí".

The form fields include:

- Identificador de la convocatoria:
- Fecha convocatoria (dd/mm/aaaa):
- Fecha de publicación:
- TFG convocado:
- Alumno 1:
- Alumno 2:
- Presidente del tribunal:
- Presidente suplente:
- Secretario del tribunal:
- Secretario suplente:
- Vocal del tribunal:
- Vocal suplente:
- Fecha de la defensa:
- Hora de la defensa:
- Lugar de la defensa:
- Marque a los alumnos convocados para la defensa:  Alumno 1:    Alumno 2:

Este formulario permite consultar los datos de una convocatoria de defensa de TFG, buscando dicha convocatoria mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido una convocatoria de defensa).

Si se desean modificar los datos de la convocatoria de defensa, deben seguirse los mismos criterios que al añadir una nueva, aunque en este caso no se permite cambiar el identificador de la convocatoria de defensa.

De nuevo, si se modifica la convocatoria de defensa y no se hace clic en Guardar o se vuelve atrás, la convocatoria no se guardará.

### Añadir la calificación de una defensa de TFG

Para añadir la calificación de una defensa de TFG, hay que rellenar las notas de los alumnos y hacer clic en el botón de guardar. Estas calificaciones se pueden modificar a posteriori, ya que no se genera ningún identificador único asociado. Información adicional: [clic aquí](#)

Búsqueda de convocatorias sin calificar:

Convocatoria:

Fecha de la defensa del TFG:

TFG convocado:

Alumnos convocados: Alumno 1:  TFG-Alumno:   
Alumno 2:

Alumno 1:  Nota numérica:  Nota en letra:

Alumno 2:  Nota numérica:  Nota en letra:

El TFG se encuentra disponible en el sistema de UVaDoc

Este formulario permite añadir la calificación de una defensa de TFG a la base de datos.

Es necesario elegir una convocatoria que no esté calificada de la lista desplegable, para obtener ciertos datos básicos sobre la convocatoria:

- Fecha de la defensa del TFG
- TFG convocado
- Alumnos convocados y sus identificadores de asignación individual (TFG-Alumno)

Una vez elegida la convocatoria, se activarán los campos de los alumnos convocados, que pueden ser uno solo o los dos. Para cada uno de estos alumnos convocados, hay que rellenar:

- **Nota numérica:** La nota del alumno en la defensa de su TFG, con un único decimal.
- **Nota en letra:** La nota del alumno en la defensa de su TFG con letra (Aprobado, Notable,...).

Además, para la convocatoria de defensa, se debe marcar con una casilla de verificación si el TFG presentado se encuentra disponible a través del sistema UVaDoc.

Una vez introducidos los datos, hay que hacer clic en el botón “Guardar” y la calificación de la defensa se guardará en la base de datos, excepto si hay un error en los datos. En todo caso, se mostrará un mensaje al usuario para informarle de lo que ha ocurrido.

Si no se hace clic en Guardar o se vuelve atrás, la calificación de la defensa no se guarda en la base de datos.

### Consultar/Modificar la calificación de una defensa de TFG existente

The screenshot shows a web browser window titled "SIGIISA-UVa - [Consultar/Modificar la calificación de una defensa de TFG]". The browser's address bar shows "uvva" and the page title is "CONSULTAR/MODIFICAR LA CALIFICACIÓN DE UNA DEFENSA DE TFG". Below the browser window, there is a search bar with the text "Búsqueda de convocatorias: CIV.2010-2011.08: Ampliación Proyecto ...". A yellow warning box contains the text: "Para modificar la calificación de una defensa de TFG, hay que rellenar las nuevas notas y hacer clic en el botón de guardar. Si no se hace clic en el botón de guardar, los cambios se perderán al cambiar de convocatoria o cerrar. Información adicional: clic aquí". Below this, there is a section titled "Datos básicos de la convocatoria:" with the following fields: "Fecha de la defensa del TFG: 14/09/2011", "TFG convocado: TFG.2010-2011.05", "TFG-Alumno: TFG.2010-2011.06-1", "Alumnos convocados: Alumno 1: [redacted] TFG.2010-2011.06-2", "Alumno 2: [redacted] TFG.2010-2011.06-2". Below these fields, there are two rows for student grades: "Alumno 1: [redacted] Nota numérica: 9,4 Nota en letra: Sobresaliente" and "Alumno 2: [redacted] Nota numérica: 9,4 Nota en letra: Sobresaliente". At the bottom, there is a checkbox labeled "El TFG se encuentra disponible en el sistema de UVaDoc" which is currently unchecked.

Este formulario permite consultar los datos de la calificación de una defensa, buscando la convocatoria de defensa mediante la lista desplegable que aparece arriba (la zona inferior no aparece hasta que se ha elegido una convocatoria).

Al elegir una convocatoria, se muestran ciertos datos básicos sobre ella, además de las calificaciones de los alumnos convocados:

- Fecha de la defensa del TFG
- TFG convocado
- Alumnos convocados y sus identificadores de asignación individual (TFG-Alumno)

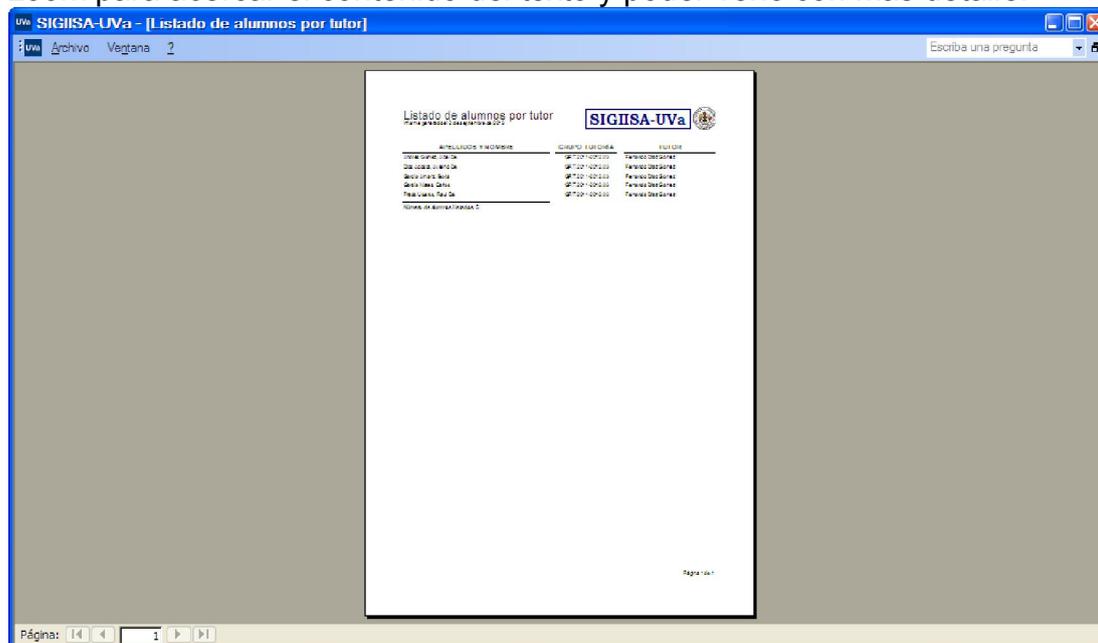
Si se desean modificar las calificaciones, de nuevo hay que insistir en que hay que hacer clic en Guardar. Si no se hace clic en Guardar o se vuelve atrás, las modificaciones en las calificaciones no se guardarán.

### 3.3.5. Informes

El modo de trabajo en este formulario-menú consiste en hacer clic en el botoncito que está a la izquierda de la opción que se desea, y después, en el panel de la derecha configurar las opciones de búsqueda. Finalmente, hay que hacer clic en el botón “**Generar informe**” que está en la parte superior, al lado del botón de volver atrás.

Generalmente, al dejar el campo de búsqueda en blanco, se obtiene un listado completo. Por ejemplo, para buscar los alumnos de un determinado tutor, es necesario elegir el tutor deseado, pero si ese campo se deja en blanco, aparecerá un listado con todos los alumnos agrupados por tutor. En otras ocasiones, no es necesario elegir un criterio de búsqueda, como en el caso del listado de alumnos que han abandonado.

Al presionar el botón “Generar informe”, si hay datos para el criterio elegido, aparecerá una nueva ventana como la de debajo, en la que se puede hacer zoom para acercar el contenido del texto y poder verlo con más detalle:



Es importante tener en cuenta dos cuestiones sobre la ventana que contiene el informe o listado:

- El informe se puede imprimir, a través del menú de Archivo, eligiendo la opción “Imprimir...”.
- Para salir del informe hay que hacerlo a través del menú Archivo, eligiendo la opción “Cerrar” o “Vista preliminar”.

**IMPORTANTE:** Para cerrar un informe **no** debe presionarse el aspa de la ventana, ya que eso cierra la aplicación por completo (este es el comportamiento de todas las ventanas de la aplicación).

Finalmente, el futuro administrador y desarrollador de la aplicación podrá añadir informes fácilmente, por lo que es posible que la lista de informes crezca en un futuro.

### **3.4 Índice de operaciones con formularios**

Si se necesita una ayuda rápida sobre determinado formulario, también se ha querido añadir un índice al final para localizar el modo de funcionamiento de las operaciones que se pueden realizar con SIGIISA-UVa.

#### **Con alumnos:**

Añadir un nuevo alumno .....	82
Consultar/Modificar un alumno existente .....	83
Ver el expediente completo de un alumno .....	84
Añadir una nueva modalidad de entrada .....	84
Consultar/Modificar una modalidad de entrada existente .....	85
Añadir un nuevo anteproyecto .....	86
Consultar/Modificar un anteproyecto existente .....	87

#### **Con profesores:**

Añadir un nuevo profesor .....	87
Consultar/Modificar un profesor existente .....	88
Añadir un nuevo área .....	89
Consultar/Modificar un área existente .....	90
Añadir un nuevo grupo de tutoría .....	90
Consultar/Modificar un grupo de tutoría existente .....	91
Añadir una nueva propuesta de TFG .....	92
Consultar/Modificar una propuesta de TFG existente .....	93

#### **Con TFGs:**

Añadir una nueva asignación de TFG .....	93
Consultar/Modificar una asignación de TFG existente .....	94
Añadir una nueva convocatoria de defensa .....	95
Consultar/Modificar una convocatoria de defensa existente .....	96
Añadir la calificación de una defensa de TFG .....	97
Consultar/Modificar la calificación de una defensa de TFG existente .....	98

#### **Con informes:**

Generar un informe con los criterios deseados .....	99
---	----