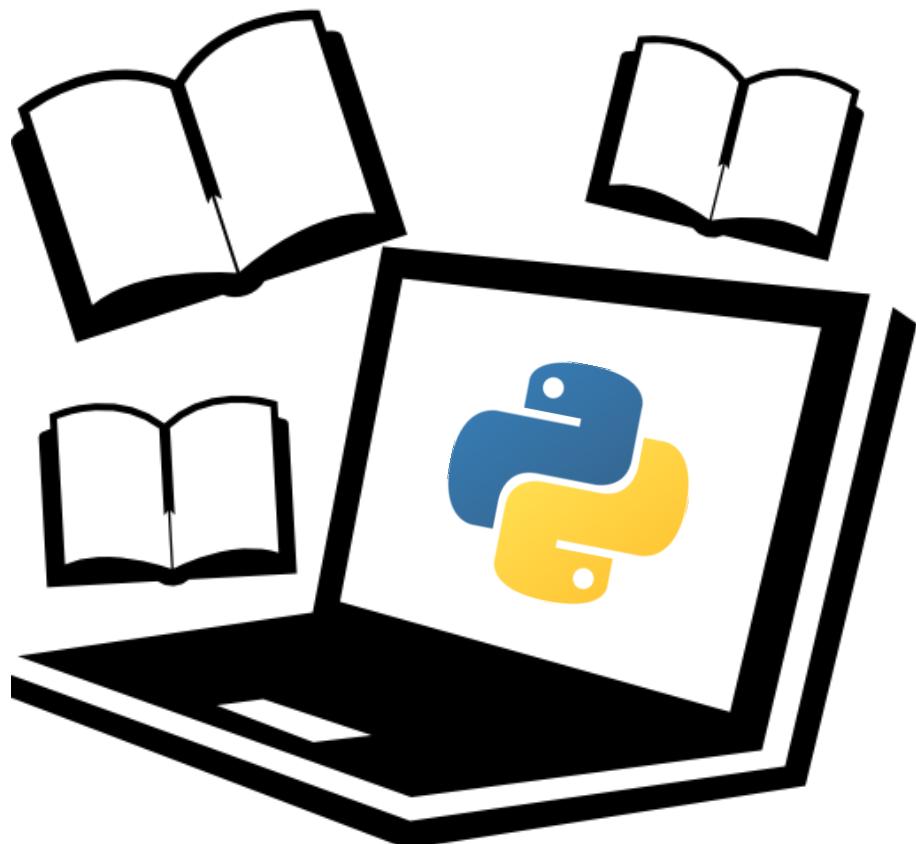


# **APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.**



**Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Multiplataforma.**

**Curso:** 2019/2020

**Tutora:** Elena Jiménez Muñoz

**Fecha de entrega:** 15/05/2020



**Daniel Barrio Aguilera**

## ÍNDICE DE CONTENIDO

<b>1. ESTUDIO DEL PROBLEMA Y ANÁLISIS DEL SISTEMA .....</b>	<b>1</b>
1.1. Introducción.....	1
1.2. Descripción.....	1
1.3. Objetivos .....	1
<b>2. RECURSOS NECESARIOS PARA EL DESARROLLO .....</b>	<b>1</b>
2.1. Recursos humanos.....	1
2.2. Recursos de hardware .....	1
2.3. Recursos de software.....	1
<b>3. PLANIFICACIÓN.....</b>	<b>2</b>
3.1. Metodología para el diseño .....	2
3.2. Secuencia de desarrollo .....	2
-    Modelo entidad – relación.....	2
-    Diagrama de calses .....	3
-    Casos de uso.....	4
-    Flujo de datos.....	5
3.3. Actividades y cálculo de tiempos de realización .....	8
<b>4. DESARROLLO .....</b>	<b>8</b>
4.1. Secuencia real del desarrollo .....	8
4.2. Modelo relacional y normalización de la base de datos .....	9
4.3. Código fuente.....	9
•    Implementación de la base de datos .....	9
•    Generación de datos automática .....	11
•    Gestión general del contenido de las tablas .....	12
4.4. Interfaz .....	14
-    Clase App controladora todas las posibles pantallas.....	16
-    Pantalla de inicio de sesión.....	17
-    Pantalla de opciones de administrador o de mantenimiento .....	18
-    Pantallas de gestión de contenido.....	19
4.5. Funciones y procedimientos .....	25
-    Inicio de sesión.....	25
-    Mostrar todo el contenido.....	26
-    Buscar según ciertos valores.....	26



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

- Añadir contenido .....	26
- Eliminar contenido.....	27
- Modificar contenido .....	27
- Finalizar préstamo .....	28
- Generar informe en formato pdf de un préstamo .....	28
<b>5. FASE DE PRUEBAS.....</b>	<b>30</b>
5.1. Pruebas realizadas .....	31
5.2. Resultados de las pruebas .....	32
<b>6. CONCLUSIONES FINALES.....</b>	<b>35</b>
<b>7. DIFICULTADES .....</b>	<b>35</b>
<b>8. BIBLIOGRAFÍA.....</b>	<b>36</b>
<b>9. ANEXOS.....</b>	<b>36</b>
Anexo I. Manual de instalación.....	36
Anexo II. Manual de uso. ....	36
Anexo III. Presentación de diapositivas. ....	36



Daniel Barrio Aguilera

## ÍNDICE DE TABLAS E IMÁGENES

Img. Modelo entidad-relación de la base de datos.....	2
Img. Diagrama de clases de main.py .....	3
Img. Diagrama de clases de database.py .....	3
Img. Casos de uso de un profesor .....	4
Img. Casos de uso de un encargado .....	4
Img. Casos de uso de un administrador .....	5
Img. Flujo de datos general .....	5
Img. Flujo de datos para la gestión de préstamos .....	6
Img. Flujo de datos para la gestión del material .....	6
Img. Flujo de datos para la gestión de departamentos.....	7
Img. Flujo de datos para la gestión de profesores .....	7
Tbl. Diagrama de Gantt inicial de marzo.....	8
Tbl. Diagrama de Gatt inicial de abril .....	8
Tbl. Diagrama de Gantt inicial de mayo .....	8
Tbl. Diagrama de Gantt final de marzo .....	8
Tbl. Diagrama de Gantt final de abril.....	9
Tbl. Diagrama de Gantt final de mayo.....	9
Img. Implementación de la base de datos .....	10
Img. Generación de datos automatica .....	11
Img. Función para mostrar todo (profesores) .....	12
Img. Función para añadir contenido (profesores).....	12
Img. Función para modificar contenido (profesores).....	13
Img. Función para eliminar contenido (profesores).....	13
Img. Función para buscar contenido (profesores) .....	13
Img. Función para finalizar préstamos .....	14
Img. Pantalla de inicio de sesión conceptual.....	14
Img. Pantalla de opciones de administrador conceptual .....	15
Img. Pantalla de gestión de departamentos conceptual .....	15
Img. Código de la clase App.....	16
Img. Código de la clase StartPage (Pantalla de inicio de sesión) .....	17
Img. Pantalla de inicio final.....	18
Img. Pantalla de opciones de administrador final.....	18



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

Img. Pantalla de opciones de mantenimiento .....	19
Img. Pantalla de gestión de departamentos .....	19
Img. Código de la clase DepartsPage (Pantalla de gestión de departamentos) .....	20
Img. Pantalla de gestión del material.....	21
Img. Pantalla de gestión de usuarios.....	21
Img. Pantalla de gestión de los departamentos de un profesor seleccionado .....	22
Img. Pantalla de selección para añadir departamentos a un profesor seleccionado ....	22
Img. Pantalla de gestión de préstamos para administradores y encargados .....	23
Img. Pantalla de creación de préstamos para administradores y encargados .....	23
Img. Pantalla de gestión de préstamos para profesores.....	24
Img. Pantalla de creación de préstamos para profesores.....	24
Img. Código de la función login (inicio de sesión) .....	25
Img. Código de la función view_profes_command (mostrar todos los profesores).....	26
Img. Código de la función search_profes_command (buscar profesores) .....	26
Img. Código de la función add_profe_command (añadir profesor) .....	26
Img. Código de la función delete_profe_command (eliminar profesor) .....	27
Img. Código de la función update_profe_command (modificar profesor) .....	27
Img. Código de la función finish_prestamo_command (finalizar préstamo).....	28
Img. Código de la función print_prestamo_command (generar informe de un préstamo) .....	29
Img. Muestra de informe en formato pdf de un préstamo .....	30
Tbl. Pruebas realizadas .....	31
Tbl. Resultados de las pruebas .....	32



## 1. ESTUDIO DEL PROBLEMA Y ANÁLISIS DEL SISTEMA

### 1.1. Introducción

El interés sobre este proyecto recae en la posibilidad de manejar datos reales de una base de datos con una interfaz gráfica que se adapte a la pantalla en la que se esté usando, pudiendo reforzar así los conocimientos obtenidos principalmente en los módulos de programación, bases de datos y desarrollo de interfaces.

### 1.2. Descripción

La aplicación que se debe obtener con este proyecto deberá mostrar de manera correcta la información que solicite el usuario y que se maneje correctamente en caso de ser modificada por el mismo.

Los perfiles de usuario que se podrán usar en esta aplicación tendrán diferentes permisos. Los permisos de **administrador** permitirán manejar **todo el contenido** de la base de datos (departamentos, profesores, material y préstamos). Como **encargado** se podrán controlar los aspectos relacionados con el **material** y los **préstamos** de todos los usuarios. Por último, como **profesor** se podrá acceder únicamente a los **préstamos** de dicho usuario.

### 1.3. Objetivos

El resultado deberá ser una aplicación con una interfaz responsive (adaptable al tamaño de la pantalla en la que se encuentre), con un diseño atractivo y sencillo para un fácil uso y que maneje sin problema las gestiones de la base de datos.

## 2. RECURSOS NECESARIOS PARA EL DESARROLLO

### 2.1. Recursos humanos

Se contará con **un único desarrollador** para el diseño de la interfaz, la creación del código para el funcionamiento del programa y la formación de la base de datos.

### 2.2. Recursos de hardware

A la hora de trabajar se dispondrá de **dos equipos**, el personal del desarrollador (Intel i5 5200U con 16 GB de RAM) y el disponible en el puesto de trabajo del mismo (Intel Xeon W-2102 con 32 GB de RAM).

### 2.3. Recursos de software

Los entornos de trabajo que se encontrarán para el desarrollo son los IDE **Visual Studio Code** y **PyCharm**. Ambos se usarán para desarrollar la interfaz de la aplicación, su código de funcionamiento y la base de datos.

El código de la aplicación estará basado en el lenguaje de programación **Python** en su versión 3.8, para diseñar la interfaz se recurrirá a la librería **tkinter** propia del mismo lenguaje y la base de datos se creará mediante la librería **sqlite3** perteneciente también a Python. Otras librerías usadas han sido **PIL** para manejar las imágenes de la interfaz, **datetime** para obtener la fecha de realización y finalización de los préstamos y, para generar y guardar informes de los préstamos en formato pdf se optó por usar **fpdf** y **easygui**.



### 3. PLANIFICACIÓN

#### 3.1. Metodología para el diseño

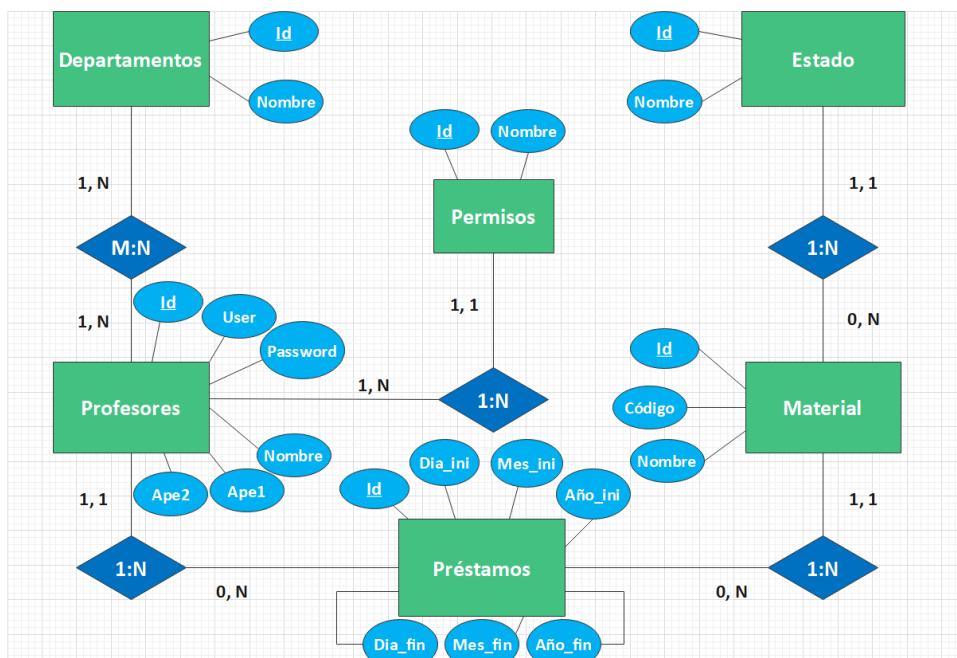
Tras estudiar el problema, se comenzará diseñando la base de datos en la que se gestionará toda la información del centro creando su diagrama entidad-relación, relacional y normalizado. Se continuará haciendo un diseño conceptual a mano de la interfaz con la que se trabajará en la aplicación en base a las tablas obtenidas en la base de datos. Para seguir, se creará el código Python correspondiente a las funciones de cada elemento de la interfaz y. Estas funciones serán probadas en consola y, tras verificar su correcto funcionamiento, se implementarán en la interfaz final y se realizará un banco de pruebas para comprobar que no haya ninguna falla entre las funciones y la interfaz.

#### 3.2. Secuencia de desarrollo

Para comenzar con la implementación del código deberemos diseñar el **modelo entidad – relación** de la base de datos, para poder controlar la información que se manejará en la aplicación; el **diagrama de clases** para tener claras las funciones y atributos de las mismas, sus **casos de uso** posibles y el **flujo de datos**.

- **Modelo entidad – relación**

La información que manejaremos en la base de datos serán los **préstamos**, la cual almacenaría el material prestado, el solicitante, su fecha de realización y finalización; el **material**, guardando su nombre, un código identificador y su estado, los **profesores**, los cuales tendrán su información personal (nombre y apellidos), un usuario, contraseña y un nivel de **permisos** determinado y podrán pertenecer a uno o más **departamentos**.



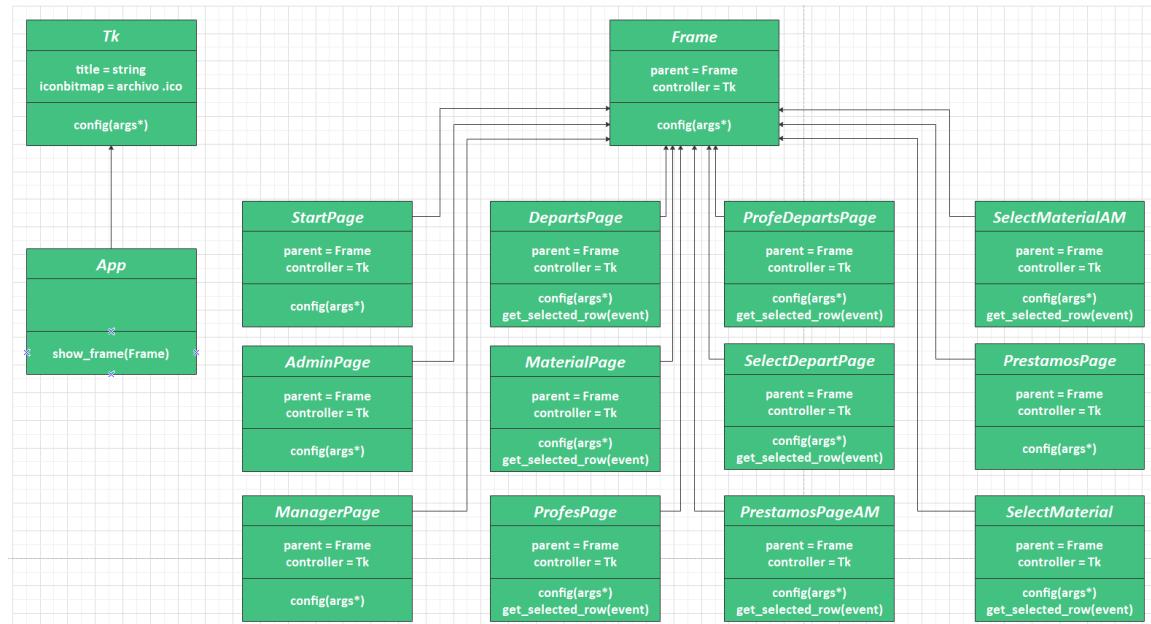
Img. Modelo entidad-relación de la base de datos



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

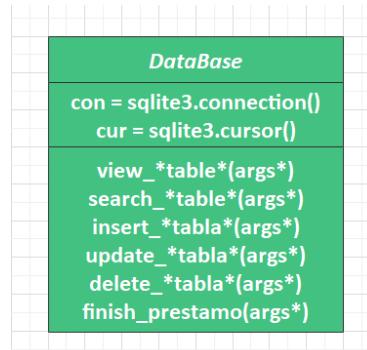
### - Diagrama de clases

Todas las clases del archivo **main.py** heredarán de **Frame**, salvo su manejador, el cual heredará de **Tk**, clases ya existentes y pertenecientes a la librería tkinter.



Img. Diagrama de clases de main.py

La clase de la base de datos se creará desde cero, nombrada como DataBase y perteneciente al fichero database.py.



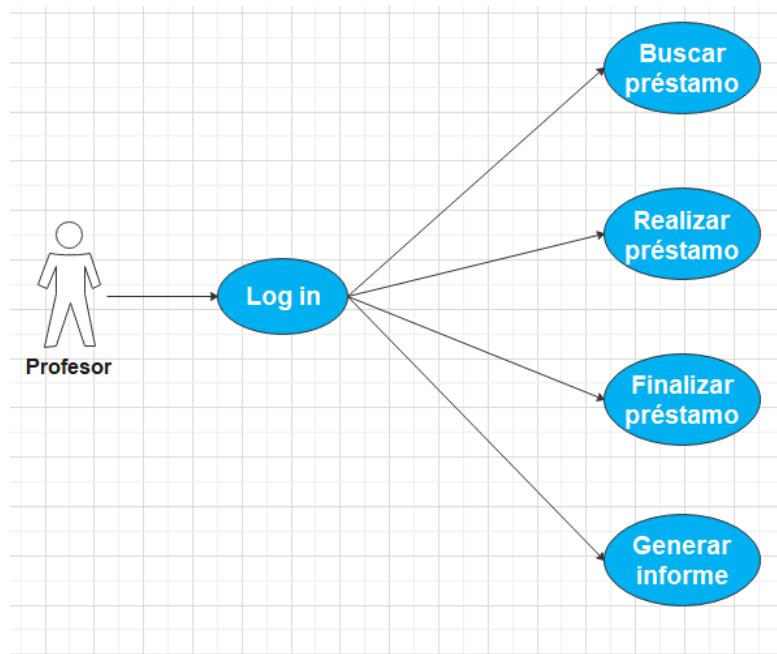
Img. Diagrama de clases de database.py



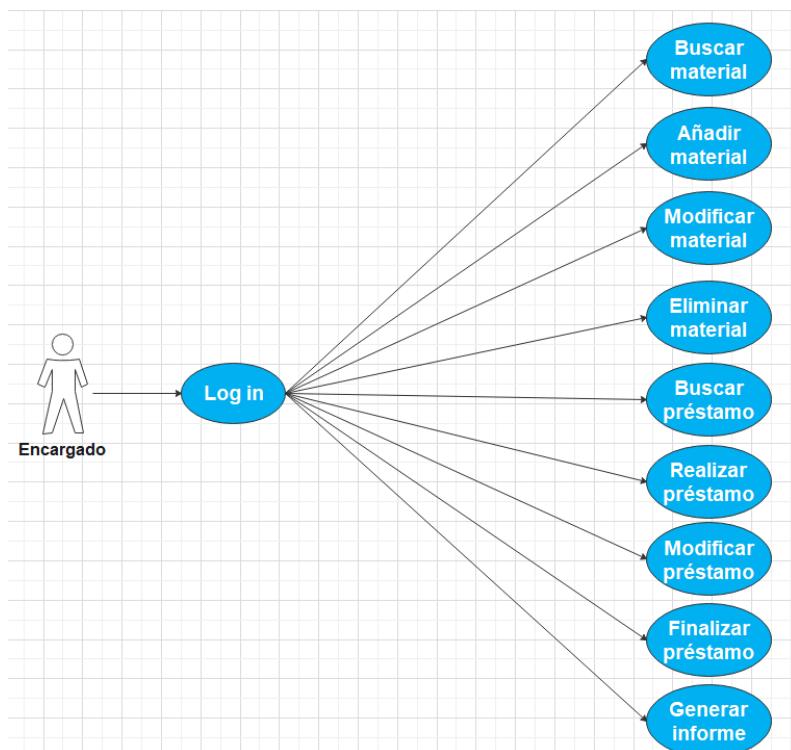
## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### - Casos de uso

Dependiendo de los permisos con los que acceda a la aplicación el usuario, se podrán dar diferentes casos de uso para cada uno de ellos.



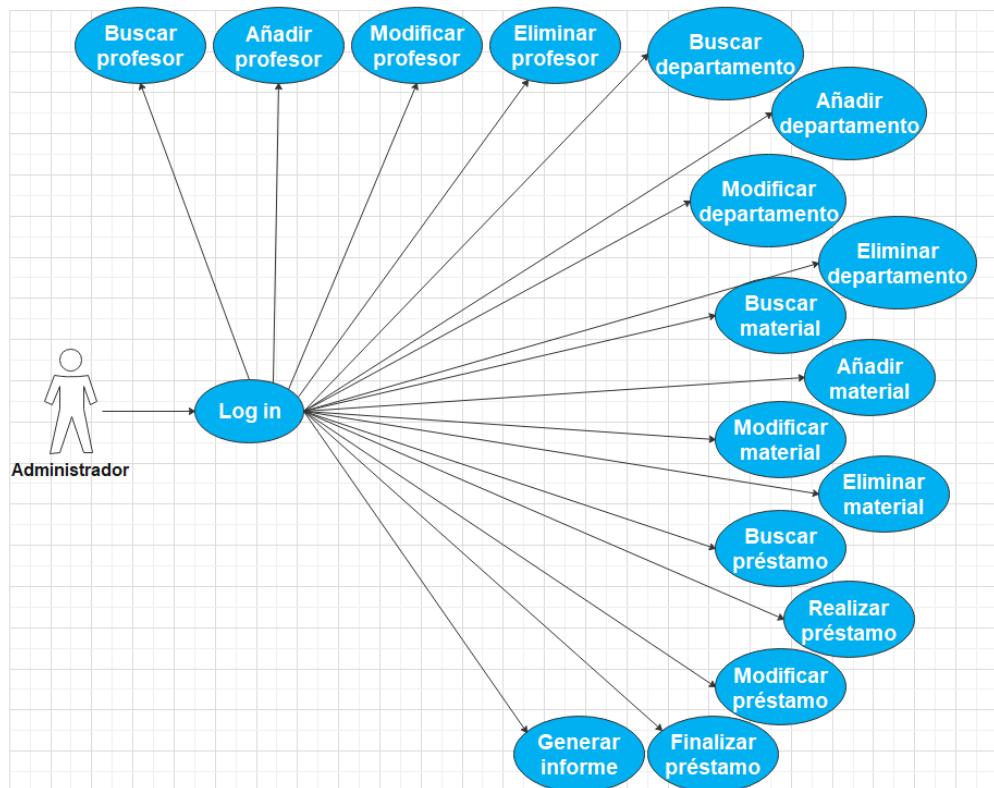
Img. Casos de uso de un profesor



Img. Casos de uso de un encargado



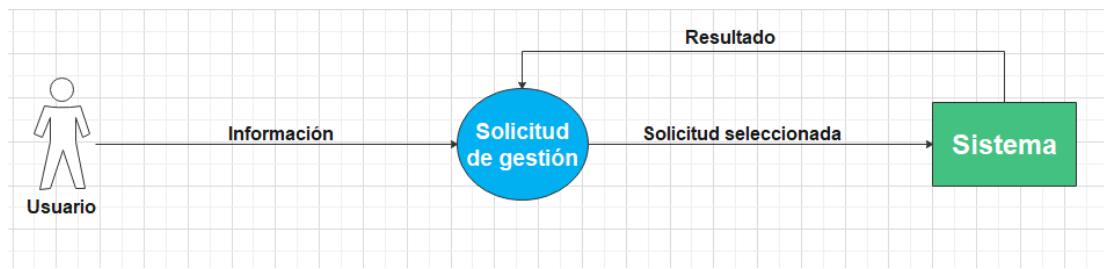
## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.



Img. Casos de uso de un administrador

### - Flujo de datos

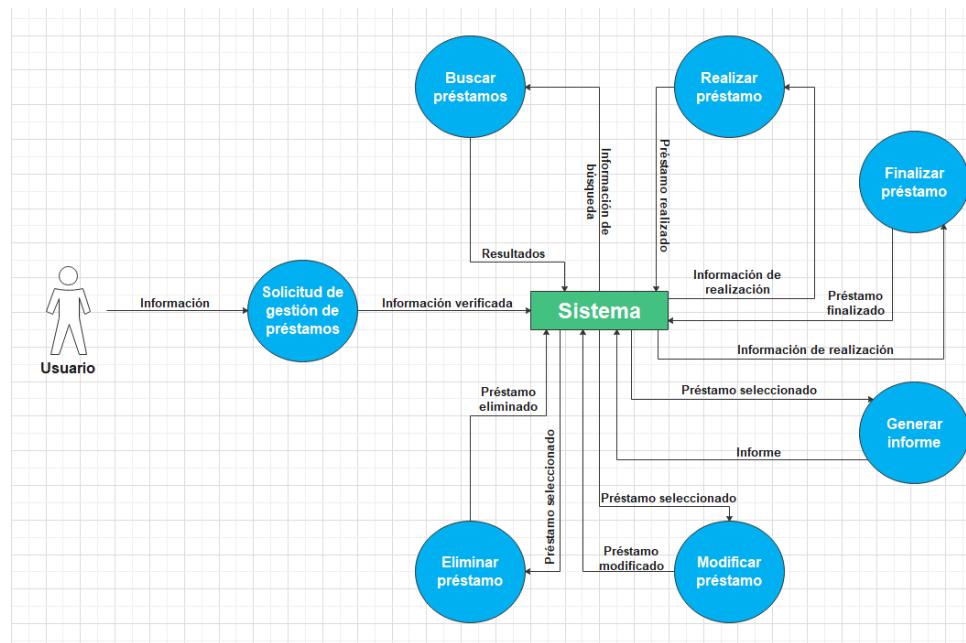
El flujo de datos general independientemente de cada caso es el siguiente:



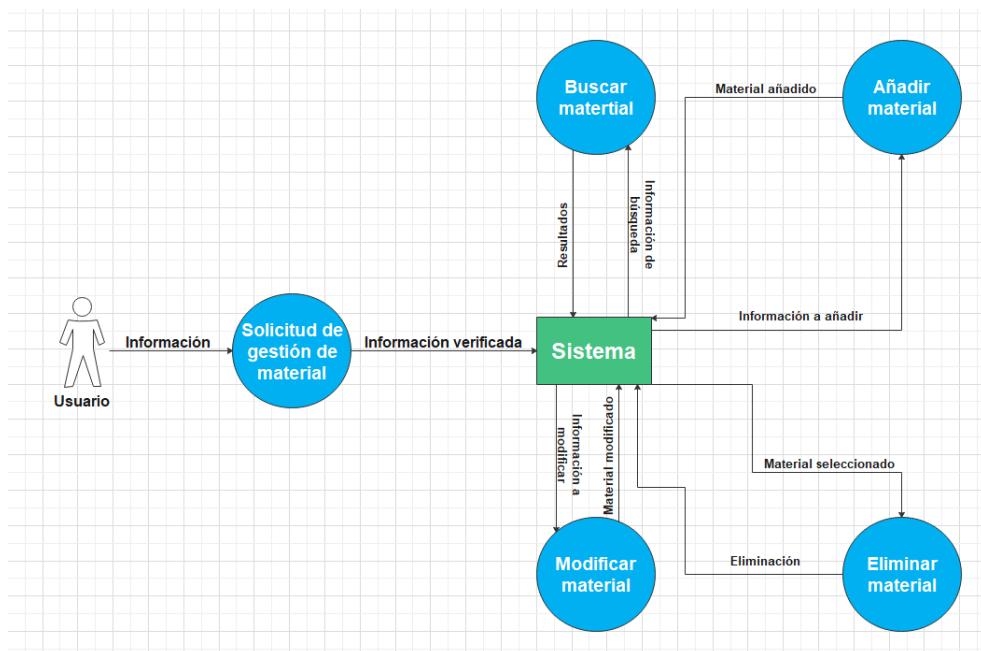
Img. Flujo de datos general

Dependiendo de la solicitud de gestión enviada al sistema, este reaccionará en función de la solicitud y del contenido a gestionar.

## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

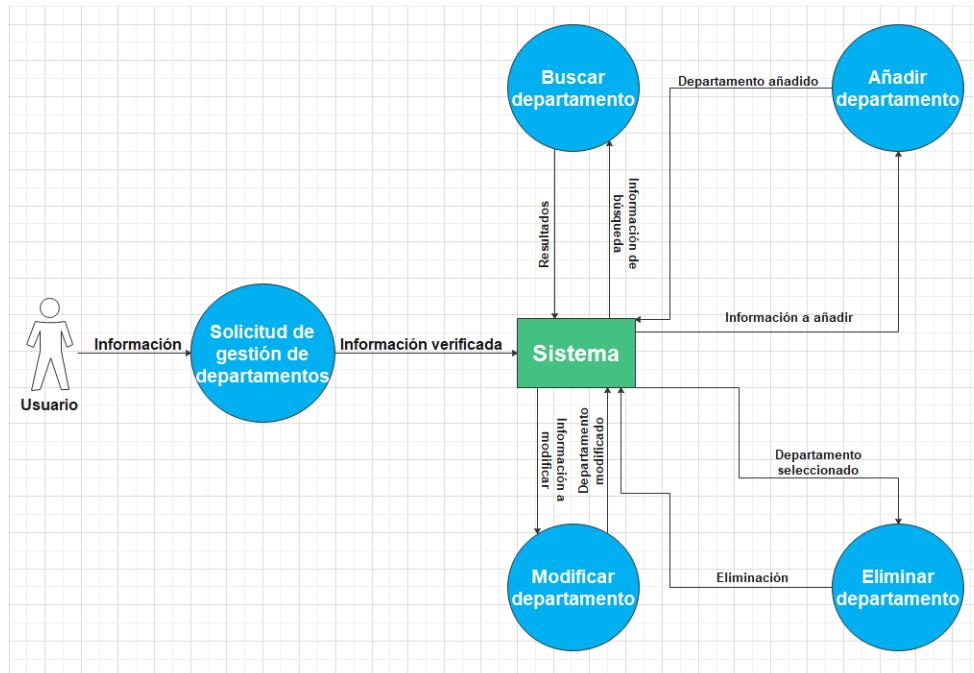


Img. Flujo de datos para la gestión de préstamos

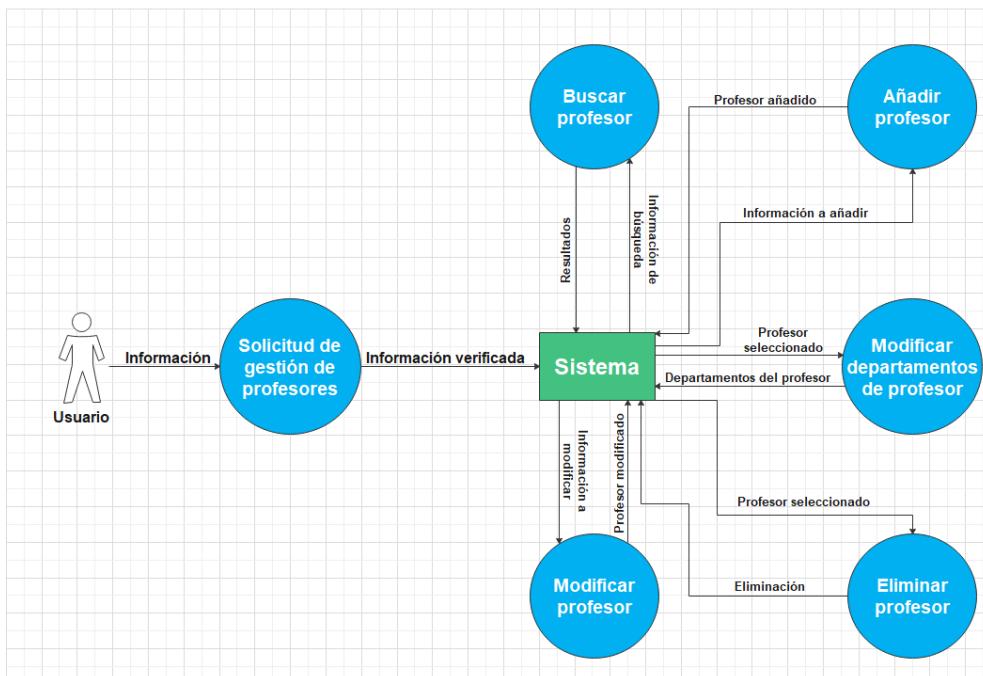


Img. Flujo de datos para la gestión del material

## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.



Img. Flujo de datos para la gestión de departamentos



Img. Flujo de datos para la gestión de profesores

## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### 3.3. Actividades y cálculo de tiempos de realización

Dado el plazo de tiempo desde el 2 de marzo de 2020 hasta el 15 de mayo de 2020, la planificación para las tareas fue la mostrada a continuación:

TAREAS				Marzo																															
Nº	Nombre	DURACIÓN	INICIO	FIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	Estudio del problema	1 día	02-mar	02-mar																															
2	Desarrollo	33 días	03-mar	14-abr																															
3	Realización de pruebas	26 días	15-abr	10-may																															
4	Implementación final	5 días	11-may	15-may																															
5	Documentación	65 días	02-mar	15-may																															

Tbl. Diagrama de Gantt inicial de marzo

TAREAS				Abril																															
Nº	Nombre	DURACIÓN	INICIO	FIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	Estudio del problema	1 día	02-mar	02-mar																															
2	Desarrollo	33 días	03-mar	14-abr																															
3	Realización de pruebas	26 días	15-abr	10-may																															
4	Implementación final	5 días	11-may	15-may																															
5	Documentación	65 días	02-mar	15-may																															

Tbl. Diagrama de Gantt inicial de abril

TAREAS				Mayo																														
Nº	Nombre	DURACIÓN	INICIO	FIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	Estudio del problema	1 día	02-mar	02-mar																														
2	Desarrollo	33 días	03-mar	14-abr																														
3	Realización de pruebas	26 días	15-abr	10-may																														
4	Implementación final	5 días	11-may	15-may																														
5	Documentación	65 días	02-mar	15-may																														

Tbl. Diagrama de Gantt inicial de mayo

Se decidió dar prioridad al desarrollo, al cual se le ha asignado la mayor cantidad de tiempo (26 días), junto con la documentación, que se cumplimentará a medida que se avanza con todas las fases del proyecto. El estudio del problema sería abordado en el primer día para poder comenzar lo antes posible con la siguiente tarea. Tras el desarrollo, la implementación final ocuparía 5 días, en los cuales se realizarán las pruebas de control de uso y se implementarían los primeros ejecutables. Estos plazos pueden variar en función de las dificultades que surjan y la velocidad que se lleve en su realización.

## 4. DESARROLLO

### 4.1. Secuencia real del desarrollo

Tras concluir las tareas establecidas, su transcurso real resultó variar respecto al inicial. El desarrollo fue más duradero de lo previsto (10 días más) debido a las complicaciones surgidas, pero se completó a tiempo y se pudo llevar a cabo una buena implementación final a pesar de solo disponer de 16 días en vez de 26.

TAREAS				Marzo																														
Nº	Nombre	DURACIÓN	INICIO	FIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	Estudio del problema	1 día	02-mar	02-mar																														
2	Desarrollo	43 días	03-mar	14-abr																														
3	Realización de pruebas	16 días	15-abr	10-may																														
4	Implementación final	5 días	11-may	15-may																														
5	Documentación	65 días	02-mar	15-may																														

Tbl. Diagrama de Gantt final de marzo

TAREAS				Abril																														
Nº	Nombre	DURACIÓN	INICIO	FIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	Estudio del problema	1 día	02-mar	02-mar																														
2	Desarrollo	43 días	03-mar	14-abr																														
3	Realización de pruebas	16 días	15-abr	10-may																														
4	Implementación final	5 días	11-may	15-may																														
5	Documentación	65 días	02-mar	15-may																														



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

Tbl. Diagrama de Gantt final de abril

Nº	Nombre	TAREAS				Mayo												
		DURACIÓN	INICIO	FIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Estudio del problema	1 día	02-mar	02-mar														
2	Desarrollo	43 días	03-mar	14-abr														
3	Realización de pruebas	16 días	15-abr	10-may														
4	Implementación final	5 días	11-may	15-may														
5	Documentación	65 días	02-mar	15-may														

Tbl. Diagrama de Gantt final de mayo

A lo largo del desarrollo de la aplicación se implementó la base de datos descrita en el modelo entidad – relación anterior y en base a la clase diseñada para la misma, las interfaces tomando de modelo el diagrama de clases planificado y las funciones y procedimientos para que la interfaz gestione correctamente la base de datos.

### 4.2. Modelo relacional y normalización de la base de datos

- **Departamentos** (Id, Nombre)
- **Permisos** (Id, Nombre)
- **Profesores** (Id, User, Password, Nombre, Ape1, Ape2, Id\_Depart(Fk), Id\_Permisos(Fk))
- **Estado** (Id, Nombre)
- **Material** (Id, Código, Nombre, Id\_Estado(Fk))
- **Préstamo** (Id, Día\_Ini, Mes\_ini, Año\_ini, Día\_fin, Mes\_fin, Año\_fin, Id\_Material(Fk), Id\_Profe(Fk))

Debido a la **relación con cardinalidad M:N** entre las tablas **Departamentos** y **Profesores**, crearemos una tabla para establecer la relación de su contenido.

- **Profesores\_departamentos** (Id\_Profe(Fk), Id\_Depart(Fk))

La **normalización** de la Primera Forma Normal (1FN), Segunda Forma Normal (2FN), Tercera Forma Normal (3FN) y Forma normal de Boyce-Codd (FNBC) se cumplen directamente en el modelo relacional.

### 4.3. Código fuente

- **Implementación de la base de datos**

La traducción de estos modelos de la base de datos a código en Python resultó en la creación de una clase para la base de datos llamada **DataBase** que heredará de la ya existente en la librería de Python **sqlite3** y se guardará en el archivo **database.py**, donde se crearán las tablas anteriores y unos datos por defecto junto con los métodos para tratar la gestión de dichas tablas. Se optó por utilizar el método **.upper()** al introducir, buscar y modificar los datos (salvo **user** y **password** de la tabla **profesores**) para evitar problemas de gestión con las letras mayúsculas y minúsculas, tratando así solo con letras mayúsculas.



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

```
4   # Clase de la base de datos
5   class DataBase:
6       def __init__(self):
7           # Creación y conexión a la base de datos
8           self.con = sqlite3.connect('prestamos.db')
9
10          # Cursor para recorrer la base de datos
11          self.cur = self.con.cursor()
12
13          # Creación de las tablas
14          self.cur.execute("""CREATE TABLE if NOT EXISTS departamentos(
15              id INTEGER PRIMARY KEY,
16              nombre text NOT NULL);""")
17
18          self.cur.execute("""CREATE TABLE if NOT EXISTS permisos(
19              id INTEGER PRIMARY KEY,
20              nombre text NOT NULL);""")
21
22          self.cur.execute("""CREATE TABLE if NOT EXISTS profesores(
23              id INTEGER PRIMARY KEY,
24              user text NOT NULL,
25              password text NOT NULL,
26              nombre text NOT NULL,
27              ape1 text NOT NULL,
28              ape2 text,
29              id_permisos INTEGER NOT NULL,
30              FOREIGN KEY(id_permisos) REFERENCES permisos(id));""")
31
32          self.cur.execute("""CREATE TABLE if NOT EXISTS profesores_departamentos(
33              id_profesor INTEGER NOT NULL,
34              id_depart INTEGER NOT NULL,
35              FOREIGN KEY(id_profesor) REFERENCES profesores(id),
36              FOREIGN KEY(id_depart) REFERENCES departamentos(id));""")
37
38          self.cur.execute("""CREATE TABLE if NOT EXISTS estado(
39              id INTEGER PRIMARY KEY,
40              nombre text NOT NULL);""")
41
42          self.cur.execute("""CREATE TABLE if NOT EXISTS material(
43              id INTEGER PRIMARY KEY,
44              nombre text NOT NULL,
45              codigo text NOT NULL,
46              id_estado INTEGER NOT NULL,
47              FOREIGN KEY(id_estado) REFERENCES estado(id));""")
48
49          self.cur.execute("""CREATE TABLE if NOT EXISTS prestamos(
50              id INTEGER PRIMARY KEY,
51              id_material INTEGER NOT NULL,
52              id_profesor INTEGER NOT NULL,
53              dia_ini text NOT NULL,
54              mes_ini text NOT NULL,
55              ano_ini text NOT NULL,
56              dia_fin text,
57              mes_fin text,
58              ano_fin text,
59              FOREIGN KEY(id_material) REFERENCES material(id),
60              FOREIGN KEY(id_profesor) REFERENCES profesor(id));""")
```

Img. Implementación de la base de datos



# APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

- **Generación de datos automática**

```
62      # Generación automática de los permisos de ADMINISTRADOR, ENCARGADO y PROFESOR para los usuarios
63      is_admin = False
64      is_manager = False
65      is_profe = False
66      self.cur.execute("SELECT nombre FROM permisos;")
67      estados = self.cur.fetchall()
68      for est in estados:
69          if str(est) == "('ADMINISTRADOR',)":
70              is_admin = True
71          elif str(est) == "('ENCARGADO',)":
72              is_manager = True
73          elif str(est) == "('PROFESOR',)":
74              is_profe = True
75      if not is_admin:
76          self.cur.execute("""INSERT INTO permisos (nombre) VALUES
77                          ('ADMINISTRADOR');""")
78      if not is_manager:
79          self.cur.execute("""INSERT INTO permisos (nombre) VALUES
80                          ('MANAGER');""")
81      if not is_profe:
82          self.cur.execute("""INSERT INTO permisos (nombre) VALUES
83                          ('PROFESOR');""")
84
85      # Generación automática de los usuarios ADMIN y MANAGER para la gestión de la base de datos desde su creación
86      is_admin = False
87      is_manager = False
88      self.cur.execute("SELECT user FROM profesores;")
89      profes_iniciales = self.cur.fetchall()
90      for prof_ini in profes_iniciales:
91          if str(prof_ini) == "('admin',)":
92              is_admin = True
93          elif str(prof_ini) == "('manager',)":
94              is_manager = True
95      if not is_admin:
96          self.cur.execute("""INSERT INTO profesores (user, password, id_permisos, nombre, ape1, ape2) VALUES
97                          ('admin', 'admin', 1, 'ADMIN', 'ADMIN', '');""")
98      if not is_manager:
99          self.cur.execute("""INSERT INTO profesores (user, password, id_permisos, nombre, ape1, ape2) VALUES
100                         ('manager', 'manager', 2, 'MANAGER', 'MANAGER', '');""")
101
102     # Generación automática de los ESTADOS DISPONIBLE y NO DISPONIBLE de los materiales a dar en los préstamos
103     is_libre = False
104     is_prestado = False
105     self.cur.execute("SELECT nombre FROM estado;")
106     estados = self.cur.fetchall()
107     for est in estados:
108         if str(est) == "('DISPONIBLE',)":
109             is_libre = True
110         elif str(est) == "('NO DISPONIBLE',)":
111             is_prestado = True
112     if not is_libre:
113         self.cur.execute("""INSERT INTO estado (nombre) VALUES
114                         ('DISPONIBLE');""")
115     if not is_prestado:
116         self.cur.execute("""INSERT INTO estado (nombre) VALUES
117                         ('NO DISPONIBLE');""")
```

Img. Generación de datos automática



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

- **Gestión general del contenido de las tablas**

Se toma de base la tabla *Departamentos* al ser la más sencilla, de la que se tomarán las mismas funciones para el resto, salvo para los préstamos.

Se dispondrá de procedimientos y funciones para **ver todo el contenido** de las tablas, **buscar** en base a los campos que se dispongan en la interfaz, **añadir** o **modificar** el contenido y **eliminar** entradas. En el caso de los **préstamos** también se podrán **finalizar**.

En caso de haber alguna relación con otra tabla al eliminar o modificar una entrada de la base de datos, el campo relacionado también intervendrá en las modificaciones.

```
254     # Mostrar todos los profesores y usuarios
255     def view_profes(self):
256         # Seleccionamos y devolvemos el todos los profesores guardados
257         self.cur.execute("""SELECT profesores.id, permisos.nombre||'-'|| profesores.nombre, profesores.apel1, profesores.apel2,
258                         profesores.user, profesores.password
259                         FROM profesores, permisos
260                         WHERE permisos.id = profesores.id_permisos;""")
261         result = self.cur.fetchall()
262         return result
```

Img. Función para mostrar todo (profesores)

```
264     # Añadir profesor
265     def insert_profe(self, permisos, nombre, apel1, apel2, user, password):
266         id_permisos = 0
267         # Valor para ID_PERMISOS en caso de que los permisos sean de "Administrador".
268         if permisos.upper() == "ADMINISTRADOR":
269             id_permisos = 1
270         # Valor para ID_PERMISOS en caso de que los permisos sean de "Encargado".
271         elif permisos.upper() == "ENCARGADO":
272             id_permisos = 2
273         # Valor para ID_PERMISOS en caso de que los permisos sean de "Profesor".
274         elif permisos.upper() == "PROFESOR" or permisos == "":
275             id_permisos = 3
276
277         # Se comprueba que se han introducido unos permisos correctos.
278         if id_permisos != 0:
279             if permisos == "":
280                 # Si no se ha introducido ningún tipo depermiso, se alerta de que se establecerán los permisos por defecto ("Profesor").
281                 messagebox.showinfo("Atención!", "Se asignarán al nuevo usuario los permisos por defecto ('Profesor').")
282             # Añadimos el profesor a la tabla
283             self.cur.execute("INSERT INTO profesores (user, password, id_permisos, nombre, apel1, apel2) VALUES (?, ?, ?, ?, ?, ?);",
284                             (user, password, id_permisos, nombre.upper(), apel1.upper(), apel2.upper()))
285             self.con.commit()
286         else:
287             # Alerta en caso de que los permisos sean incorrectos
288             messagebox.showwarning("Error", "Los permisos que desea asignar al nuevo usuario son incorrectos. Deben ser de 'Administrador', 'Encargado' o 'Profesor'.")
```

Img. Función para añadir contenido (profesores)



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

```
290 # Modificar profesor
291 def update_profe(self, id_update, permisos, nombre, ape1, ape2, user, password):
292     # Si se ha seleccionado una entrada, se avisa al usuario por si quiere no modificarla como ha indicado
293     if messagebox.askokcancel("Atención", "¿Está seguro de modificar la entrada seleccionada con los datos que va a proporcionar?"):
294         id_permisos = 0
295         if permisos.upper() == "ADMINISTRADOR":
296             id_permisos = 1
297         elif permisos.upper() == "ENCARGADO":
298             id_permisos = 2
299         elif permisos.upper() == "PROFESOR":
300             id_permisos = 3
301
302         if id_permisos != 0:
303             # Se actualiza la entrada con el id dado (id_update) y se le dan los parámetros introducidos
304             self.cur.execute("UPDATE profesores SET user = ?, password = ?, id_permisos = ?, nombre = ?, ape1 = ?, ape2 = ? WHERE id = ?",
305                             (user, password, id_permisos, nombre.upper(), ape1.upper(), ape2.upper(), id_update))
306             self.con.commit()
307
308     else:
309         # Alerta en caso de que los permisos sean incorrectos
310         messagebox.showwarning("Error", "Los permisos que desea asignar al usuario seleccionado son incorrectos. Deben ser de 'Administrador', 'Encargado' o 'Profesor'.")
```

Img. Función para modificar contenido (profesores)

```
312 # Eliminar profesor
313 def delete_profe(self, id_delete):
314     # Si se ha seleccionado una entrada, se avisa al usuario por si quiere no eliminarla en realidad
315     if messagebox.askokcancel("Atención", "¿Está seguro de borrar la entrada seleccionada?"):
316         # Se eliminan las relaciones que tuviera en la tabla de préstamos
317         self.cur.execute("DELETE FROM prestamos WHERE id_profesor = "+id_delete+";")
318         self.con.commit()
319
320         # Se eliminan las relaciones que tuviera en la tabla profesores_departamentos
321         self.cur.execute("DELETE FROM profesores_departamentos WHERE id_profesor = "+id_delete+";")
322         self.con.commit()
323
324         # Se elimina el profesor con el id dado (id_delete)
325         self.cur.execute("DELETE FROM profesores WHERE id = "+id_delete+";")
326         self.con.commit()
```

Img. Función para eliminar contenido (profesores)

```
328 # Buscar profesor
329 def search_profe(self, permisos, nombre, ape1, ape2, user, password):
330     query = """SELECT profesores.id, permisos.nombre||'-', profesores.nombre, profesores.apel1, profesores.apel2,
331             profesores.user, profesores.password
332             FROM profesores, permisos
333             WHERE permisos.id = profesores.id_permisos"""
334
335     # Comprobamos sobre que datos se desea buscar
336     if permisos != "":
337         query += " AND permisos.nombre = '"+permisos.upper()+"'"
338     if nombre != "":
339         query += " AND profesores.nombre = '"+nombre.upper()+"'"
340     if ape1 != "":
341         query += " AND profesores.apel1 = '"+ape1.upper()+"'"
342     if ape2 != "":
343         query += " AND profesores.apel2 = '"+ape2.upper()+"'"
344     if user != "":
345         query += " AND profesores.user = '"+user+"'"
346     if password != "":
347         query += " AND profesores.password = '"+password+"'"
348
349     # Seleccionamos y devolvemos la información del profesor que coincida con los parámetros dados
350     self.cur.execute(query)
351     result = self.cur.fetchall()
352
353     return result
```

Img. Función para buscar contenido (profesores)



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

```
571  
572     # Finalizar préstamo para cualquier usuario  
573     def finish_prestamo(self, id_prestamo, id_material, dia_fin, mes_fin, ano_fin):  
574         # Se alerta de la finalización del préstamo al usuario  
575         if messagebox.askokcancel("Atención", "Está seguro de terminar el préstamo?"):  
576             # Se le añade la fecha de finalización al préstamo indicado  
577             self.cur.execute("UPDATE prestamos SET dia_fin = ?, mes_fin = ?, ano_fin = ? WHERE id = ?",  
578                         (dia_fin, mes_fin, ano_fin, id_prestamo))  
579             self.con.commit()  
580  
581         # Cambiamos el estado del material prestado a DISPONIBLE (id_estado = 1)  
582         self.cur.execute("UPDATE material SET id_estado = 1 WHERE id = "+id_material)  
583         self.con.commit()
```

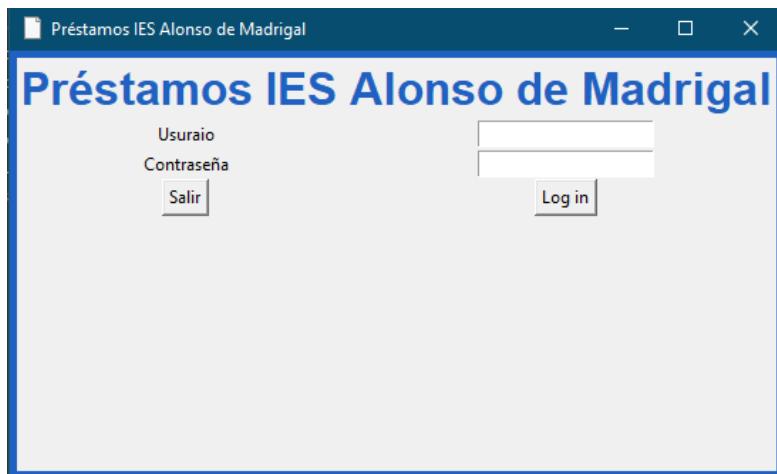
Img. Función para finalizar préstamos

### 4.4. Interfaz

Para la interfaz, como se mencionó al comienzo, se utilizará la librería **tkinter**, mediante la cual se crearán las **clases** que heredarán de **Frase** en el archivo **main.py** correspondientes a cada pantalla, todas controladas por una misma clase que heredará de Tk, clase perteneciente a tkinter. En el apartado visual se decidió tomar una paleta de colores **azules y grises**, basados en el logo del IES Alonso de Madrigal.

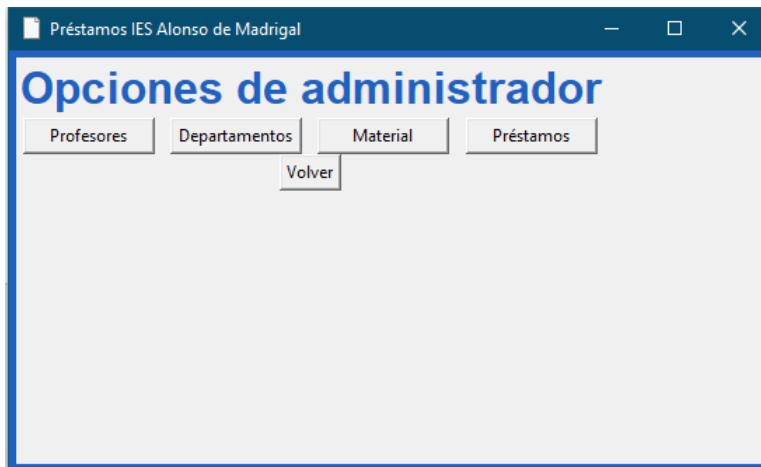
La idea del diseño era comenzar en una pantalla que pidiera un usuario y contraseña, a partir de los cuales se accedería a la pantalla con las opciones de gestión permitidas en función de los permisos que tuviera el usuario.

El **diseño inicial** de algunas de estas pantallas fue el siguiente:

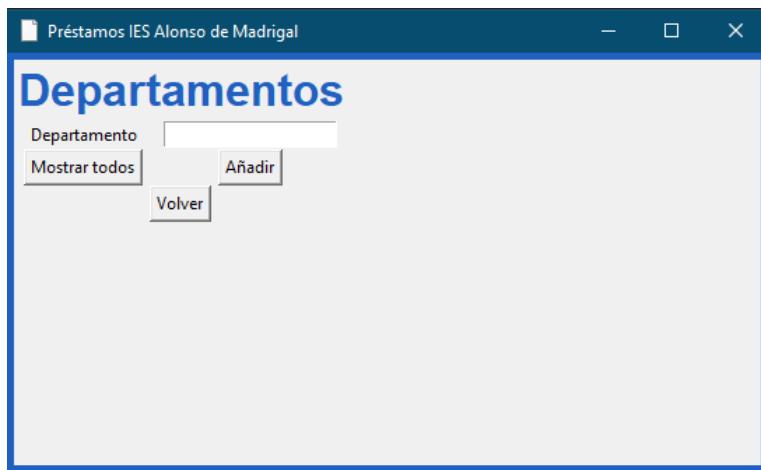


Img. Pantalla de inicio de sesión conceptual





Img. Pantalla de opciones de administrador conceptual



Img. Pantalla de gestión de departamentos conceptual

Tras algunos cambios y tomando como base definitiva el diseño de la página web de Python ([www.python.org](http://www.python.org)) se obtuvo el diseño mostrado a continuación basado en torno a una lista con entradas de texto sobre la misma para mostrar los diferentes campos de la información que se muestre y, a la derecha de la lista, los botones con las diferentes opciones de gestión de dicha información, la cual se almacena al ser seleccionada en la lista en una tupla temporal.

## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### - Clase App controladora todas las posibles pantallas

Será la encargada de generar el cuerpo de la interfaz y sobre la que se dibujarán todas las pantallas creadas para gestionar la base de datos.

```
35 # Clase principal
36 class App(tkinter.Tk):
37
38     def __init__(self, *args, **kwargs):
39         tkinter.Tk.__init__(self, *args, **kwargs)
40
41         # Tamaño mínimo de la ventana
42         self.minsize(1000, 500)
43
44         # Fuentes para los elementos de la interfaz
45         self.title_font = font.Font(family="Calibri", size=32, weight="bold", slant="italic")
46         self.second_title_font = font.Font(family="calibri", size=16, weight="bold", slant="italic")
47         self.label_font = font.Font(family="calibri", size=12, weight="bold", slant="roman")
48         self.entry_font = font.Font(family="Calibri", size=12, weight="normal", slant="roman")
49         self.button_font = font.Font(family="Calibri", size=12, weight="bold", slant="roman")
50         self.list_font = font.Font(family="Calibri", size=16, weight="bold", slant="roman")
51
52         # Icono y nombre de la aplicación
53         self.iconbitmap("logo_alomadrigal.ico")
54         self.title("Préstamos IES Alonso de Madrigal")
55
56         self.config(bd=5, bg="#003468")
57
58         # frame contenedor sobre el que pondremos el frame que deseemos visualizar
59         self.container = tkinter.Frame(self)
60         self.container.pack(anchor="center", fill="both", expand=True)
61         self.container.grid_rowconfigure(0, weight=1)
62         self.container.grid_columnconfigure(0, weight=1)
63
64         # Ponemos todos los frames uno encima del otro mediante su nombre. El que quede encima sera el visible
65         self.frames = {}
66
67         for f in (SelectMaterialPageProf, PrestamosPageProf, SelectMaterialPageAM, PrestamosPageAM, MaterialPage,
68                   ProfDepartPage, SelectDepartPage, ProfesPage, DepartPage, ManagerPage, AdminPage, StartPage):
69             page_name = f.__name__
70             frame = f(parent=self.container, controller=self)
71             self.frames[page_name] = frame
72             frame.grid(row=0, column=0, sticky="nsew")
73
74         # Forzamos que sea visible el frame de la página de inicio ("StartPage")
75         self.show_frame("StartPage")
76
77         # Función para mostrar el frame según su nombre (page_name):
78         def show_frame(self, page_name):
79             self.frame = self.frames[page_name]
80             self.frame.tkraise()
```

Img. Código de la clase App



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### - Pantalla de inicio de sesión

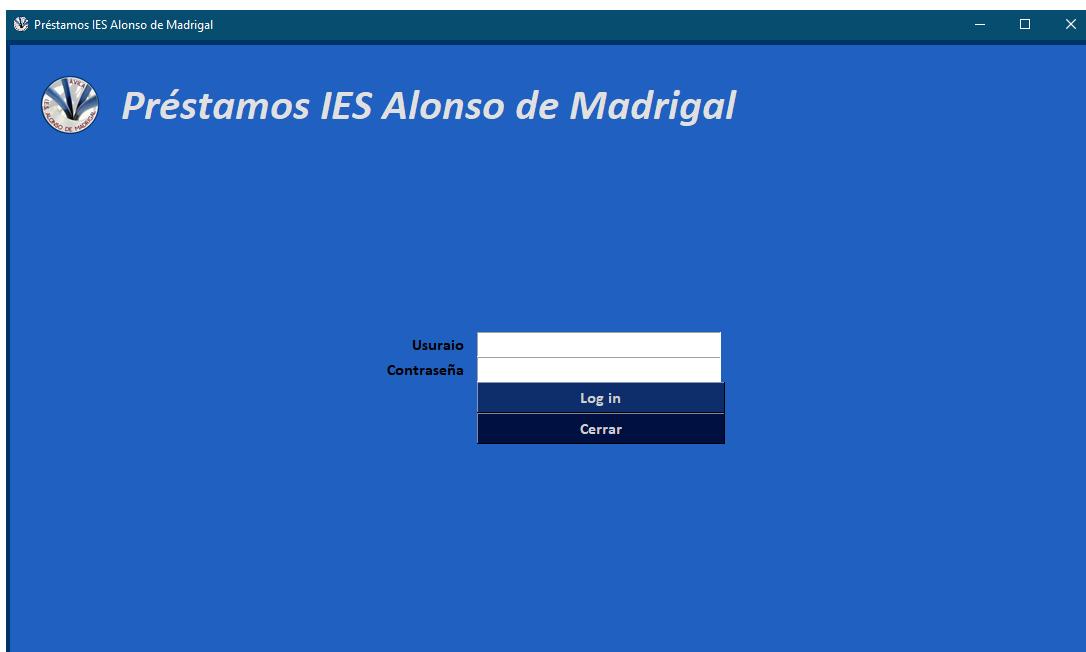
Tendremos a nuestra disposición dos entradas de texto, una para el usuario y otra para la contraseña, y dos botones, uno para acceder con el usuario y la contraseña que introduzcamos y el otro para salir de la aplicación.

```
132 # Página de inicio de sesión
133 class StartPage(tkinter.Frame):
134
135     def __init__(self, parent, controller):
136         tkinter.Frame.__init__(self, parent)
137         self.config(bg="#2060C0", bd=20)
138         self.controller = controller
139
140         # Frames para contener la cabecera y el cuerpo de la página
141         self.header = tkinter.Frame(self, bg="#2060C0")
142         self.header.pack(anchor="center", fill="both", expand=True)
143
144         self.body = tkinter.Frame(self, bg="#2060C0")
145         self.body.pack(anchor="center", fill="both", expand=True)
146
147         self.body.grid_columnconfigure(0, weight=1)
148         self.body.grid_columnconfigure(1, weight=1)
149
150         # Contenido de la cabecera con el logo del IES Alonso de Madrigal y el nombre de la app
151         global image
152         self.resized_image = image.resize((60, 60), Image.ANTIALIAS)
153         self.img_logo = ImageTk.PhotoImage(self.resized_image)
154
155         self.title_img = tkinter.Label(self.header, image=self.img_logo, bg="#2060C0", bd=10)
156         self.title_img.grid(row=0, column=0, sticky="nsew")
157
158         self.lb_title = tkinter.Label(self.header, text="Préstamos IES Alonso de Madrigal", font=controller.title_font, bg="#2060C0", fg="#E0E0E0", bd=10)
159         self.lb_title.grid(row=0, column=1, columnspan=2, sticky="nsew")
160
161         # Campo para introducir el usuario
162         self.lb_user = tkinter.Label(self.body, text="Usuario", font=controller.label_font, bg="#2060C0")
163         self.lb_user.grid(row=0, column=0, padx=10, sticky="nse")
164
165         self.user_text = tkinter.StringVar()
166         self.ent_user = tkinter.Entry(self.body, textvariable=self.user_text, width=30, font=controller.entry_font)
167         self.ent_user.grid(row=0, column=1, sticky="nsw")
168         # El entry para el nombre de usuario estará seleccionado de manera predeterminada
169         self.ent_user.focus()
170
171         # Campo para introducir la contraseña
172         self.lb_pass = tkinter.Label(self.body, text="Contraseña", font=controller.label_font, bg="#2060C0")
173         self.lb_pass.grid(row=1, column=0, padx=10, sticky="nse")
174         self.pass_text = tkinter.StringVar()
175         self.ent_pass = tkinter.Entry(self.body, textvariable=self.pass_text, width=30, font=controller.entry_font)
176         self.ent_pass.grid(row=1, column=1, sticky="nsw")
177         # Ciframos la contraseña para que al escribir solo se vean asteriscos (*)
178         self.ent_pass.config(show="*")
179
180         self.btn_login = tkinter.Button(self.body, text="Log in", font=controller.button_font, bg="#0E2E6B", fg="#CFCFCF", bd=1, width=30, command=lambda: login(self.controller))
181         self.btn_login.grid(row=2, column=1, sticky="nsw")
182
183         self.btn_exit = tkinter.Button(self.body, text="Cerrar", font=controller.button_font, bg="#001040", fg="#CFCFCF", bd=1, width=30, command=close)
184         self.btn_exit.grid(row=3, column=1, sticky="nsw")
```

Img. Código de la clase StartPage (Pantalla de inicio de sesión)



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.



Img. Pantalla de inicio final

- Pantalla de opciones de administrador o de mantenimiento

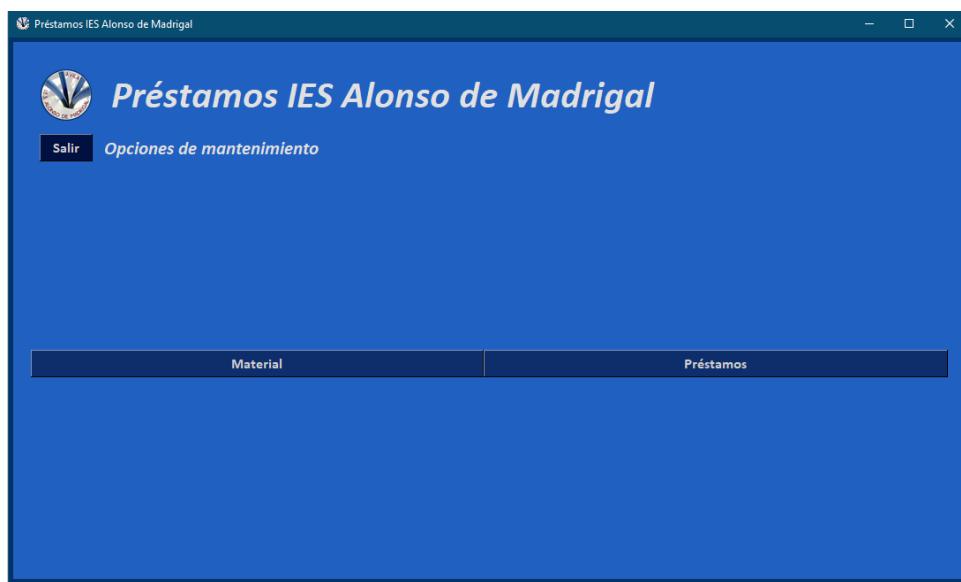


Img. Pantalla de opciones de administrador final



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

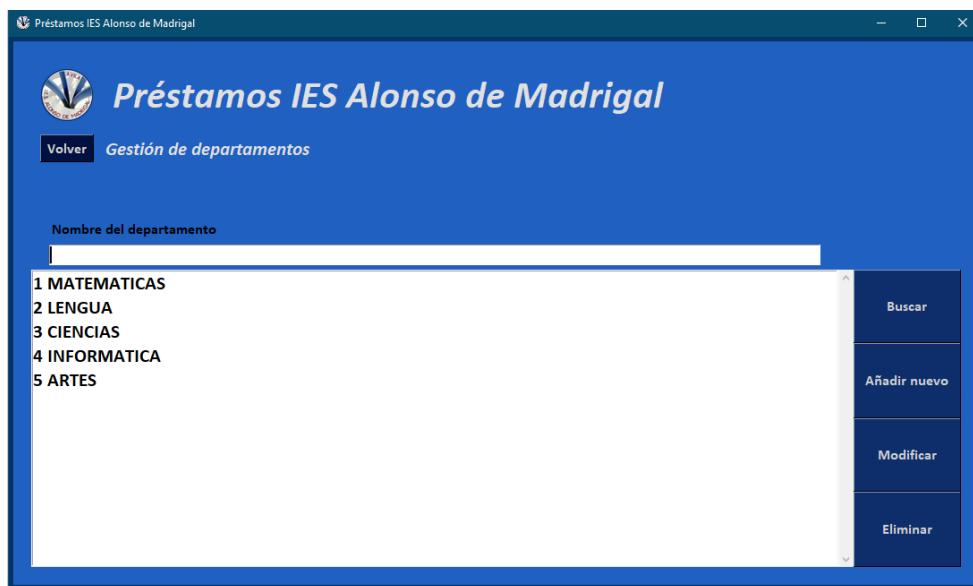
La pantalla de opciones de mantenimiento es la misma que la de opciones de administrador, pero solo con las opciones de gestión de material y de préstamos, las correspondientes a su nivel de permisos.



Img. Pantalla de opciones de mantenimiento

### - Pantallas de gestión de contenido

La más básica de todas es la de *Departamentos*, pero como se verá en las imágenes, todas están basadas en torno al mismo diseño, una lista con entradas de texto encima para poder gestionar la información y a la derecha los botones para llamar a las diferentes funciones.



Img. Pantalla de gestión de departamentos



# APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

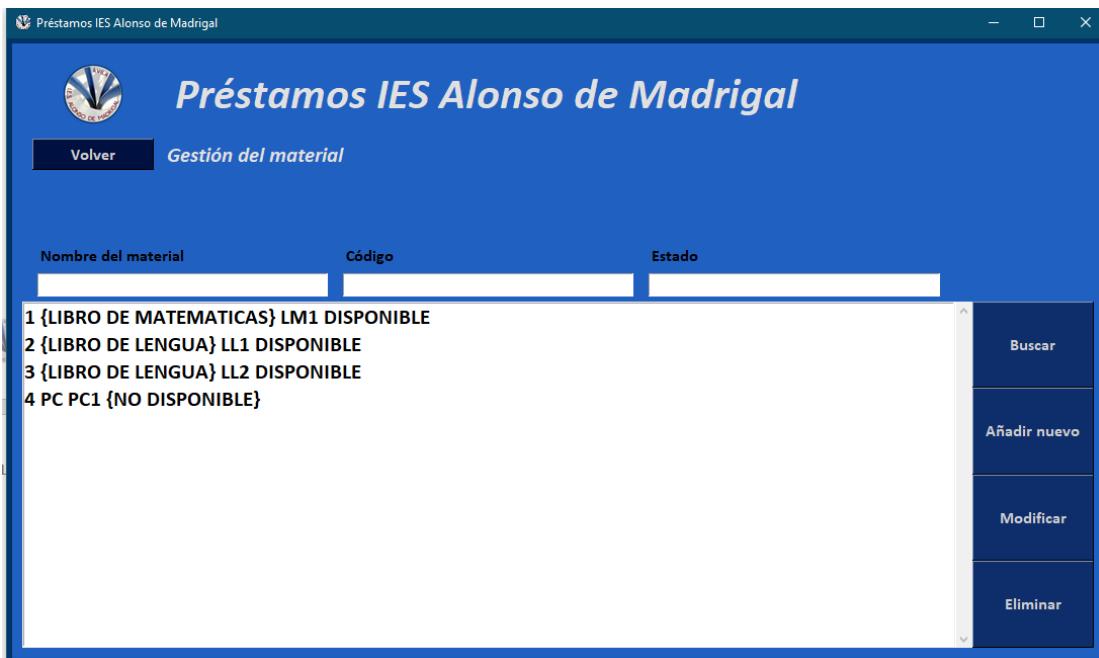
```
376 # Página de consultas en la tabla DEPARTAMENTOS
377 class DepartsPage(tkinter.Frame):
378
379     def __init__(self, parent, controller):
380         tkinter.Frame.__init__(self, parent)
381         self.config(bg="#2060C0", bd=20)
382         self.controller = controller
383
384         # Frames para contener la cabecera y el cuerpo de la página
385         self.header = tkinter.Frame(self, bg="#2060C0")
386         self.header.pack(anchor="center", fill="both", expand=True)
387
388         self.body = tkinter.Frame(self, bg="#2060C0")
389         self.body.pack(anchor="center", fill="both", expand=True)
390
391         self.body.grid_rowconfigure(2, weight=1)
392         self.body.grid_rowconfigure(3, weight=1)
393         self.body.grid_rowconfigure(4, weight=1)
394         self.body.grid_rowconfigure(5, weight=1)
395         self.body.grid_columnconfigure(0, weight=1)
396
397         # Variables globales usadas
398         global image
399
400         # Cabecera con el logo del IES Alonso de Madrigal y el nombre de la app
401         self.resized_image = image.resize((60, 60), Image.ANTIALIAS)
402         self.img_logo = ImageTk.PhotoImage(self.resized_image)
403
404         self.title_img = tkinter.Label(self.header, image=self.img_logo, bg="#2060C0", bd=10)
405         self.title_img.grid(row=0, column=0, sticky="nsew")
406
407         self.lb_title = tkinter.Label(self.header, text="Préstamos IES Alonso de Madrigal", font=controller.title_font, bg="#2060C0", fg="#E0E0E0", bd=10)
408         self.lb_title.grid(row=0, column=1, sticky="nsw")
409
410         self.lb_second_title = tkinter.Label(self.header, text="Gestión de departamentos", font=controller.second_title_font, bg="#2060C0", fg="#E0E0E0", pady=5)
411         self.lb_second_title.grid(row=1, column=1, sticky="nsw")
412
413         self.btn_back = tkinter.Button(self.header, text="Volver", font=controller.button_font, bg="#001040", fg="#CFCFCF", bd=1, command=lambda: self.controller.show_frame("Adm"))
414         self.btn_back.grid(row=1, column=0, padx=10, sticky="ew")
415
416         # Cuerpo con las opciones de visualización y gestión de los departamentos
417         self.lb_nombre_dep = tkinter.Label(self.body, text="Nombre del departamento", font=controller.label_font, bg="#2060C0")
418         self.lb_nombre_dep.grid(row=0, column=0, padx=20, sticky="nsw")
419
420         self.nombre_dep_text = tkinter.StringVar()
421         self.ent_nombre_dep = tkinter.Entry(self.body, textvariable=self.nombre_dep_text, font=controller.entry_font)
422         self.ent_nombre_dep.grid(row=1, column=0, padx=20, pady=5, sticky="nsew")
423
424         self.list_dep = tkinter.Listbox(self.body, font=controller.list_font)
425         self.list_dep.grid(row=2, column=0, rowspan=4, sticky="nsew")
426
427         self.sb = tkinter.Scrollbar(self.body, orient="vertical", command=self.list_dep.yview)
428         self.sb.grid(row=2, column=1, rowspan=4, sticky="nsw")
429
430         self.list_dep.configure(yscrollcommand=self.sb.set)
431         self.list_dep.bind('<ListboxSelect>', self.get_selected_row)
432
433         self.btn_search = tkinter.Button(self.body, text="Buscar", width=14, font=controller.button_font, bg="#0E2E6B", fg="#CFCFCF", bd=1, command=lambda: search_depart_command())
434         self.btn_search.grid(row=2, column=2, sticky="nsw")
435
436         self.btn_add = tkinter.Button(self.body, text="Añadir nuevo", width=14, font=controller.button_font, bg="#0E2E6B", fg="#CFCFCF", bd=1, command=lambda: add_depart_command())
437         self.btn_add.grid(row=3, column=2, sticky="nsw")
438
439         self.btn_update = tkinter.Button(self.body, text="Modificar", width=14, font=controller.button_font, bg="#0E2E6B", fg="#CFCFCF", bd=1, command=lambda: update_depart_command())
440         self.btn_update.grid(row=4, column=2, sticky="nsw")
441
442         self.btn_delete = tkinter.Button(self.body, text="Eliminar", width=14, font=controller.button_font, bg="#0E2E6B", fg="#CFCFCF", bd=1, command=lambda: delete_depart_command())
443         self.btn_delete.grid(row=5, column=2, sticky="nsw")
444
445         view_departs_command(self)
446
447     def get_selected_row(self, event):
448         global selected_tuple_dep
449         try:
450             index = self.list_dep.curselection()[0]
451             selected_tuple_dep = self.list_dep.get(index)
452             self.ent_nombre_dep.delete(0, "end")
453             self.ent_nombre_dep.insert("end", selected_tuple_dep[1])
454         except:
455             del selected_tuple_dep
```

Img. Código de la clase DepartsPage (Pantalla de gestión de departamentos)



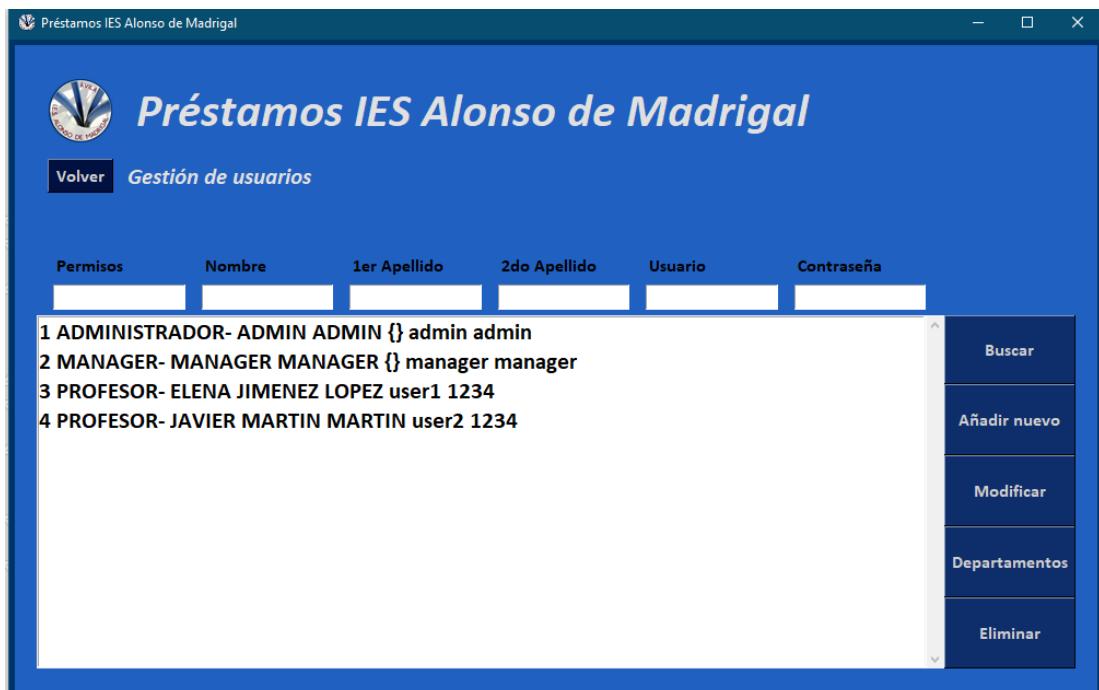
## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

Para la gestión del material trataremos también su código y el estado en el que esté.



Img. Pantalla de gestión del material

Para los usuarios se podrá manejar su información personal, los permisos, su usuario y contraseña. Su creación y modificación deberá hacerse desde un perfil con permisos de administrador.

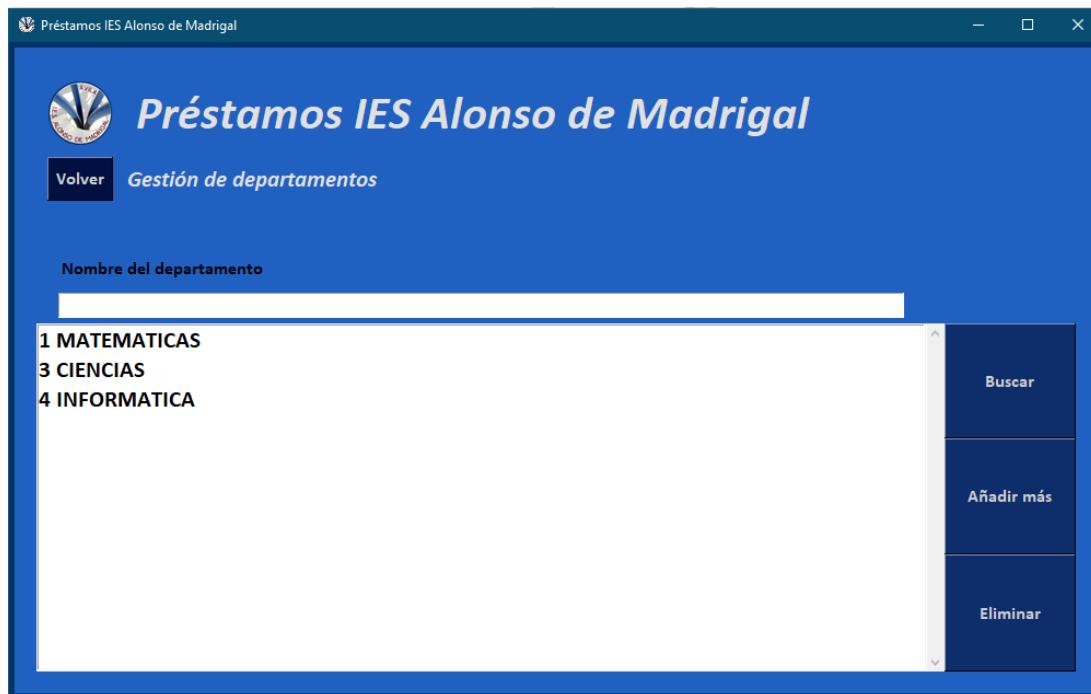


Img. Pantalla de gestión de usuarios

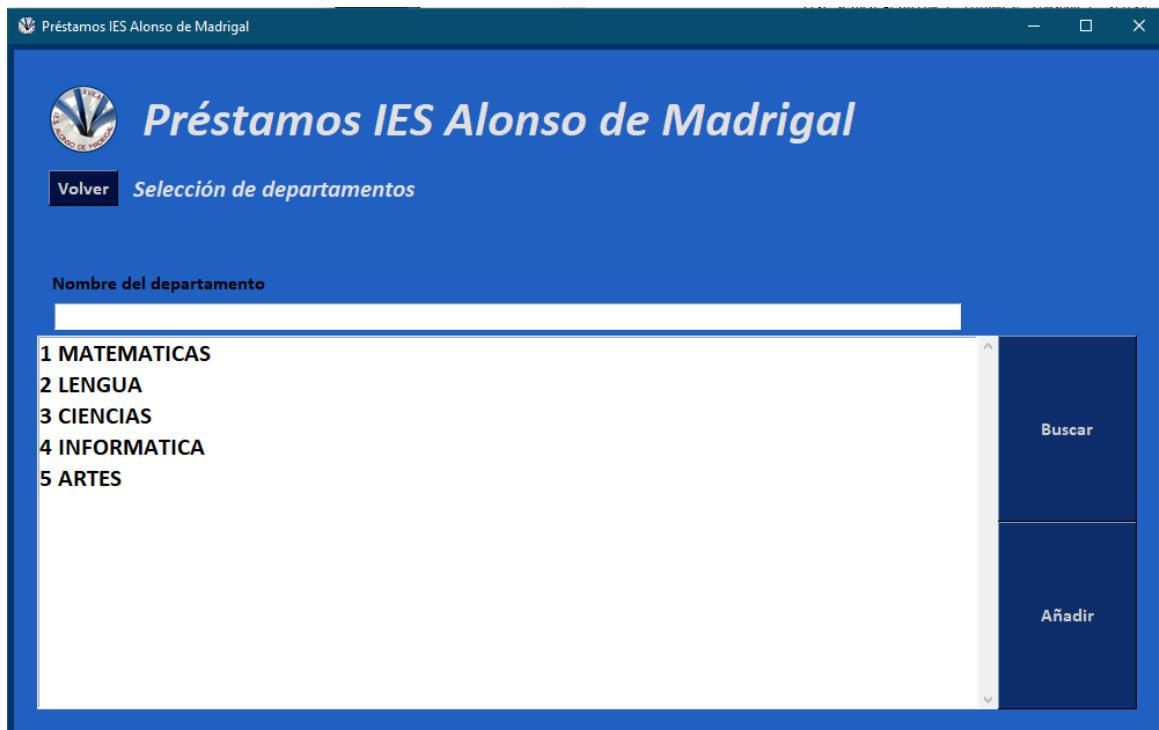


## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

La gestión de los departamentos a los que pertenece cada usuario se realiza a través del botón departamentos, mediante el cual, si se ha seleccionado un profesor, veremos los departamentos a los que pertenece y la posibilidad de añadirlo a otros en los que no esté.



Img. Pantalla de gestión de los departamentos de un profesor seleccionado

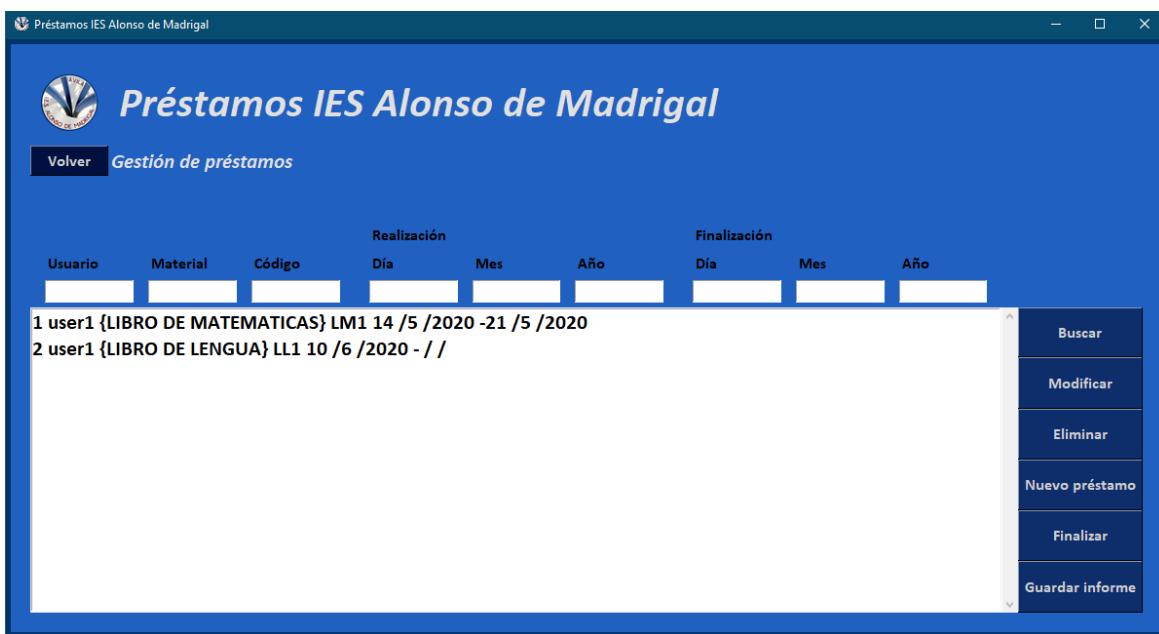


Img. Pantalla de selección para añadir departamentos a un profesor seleccionado

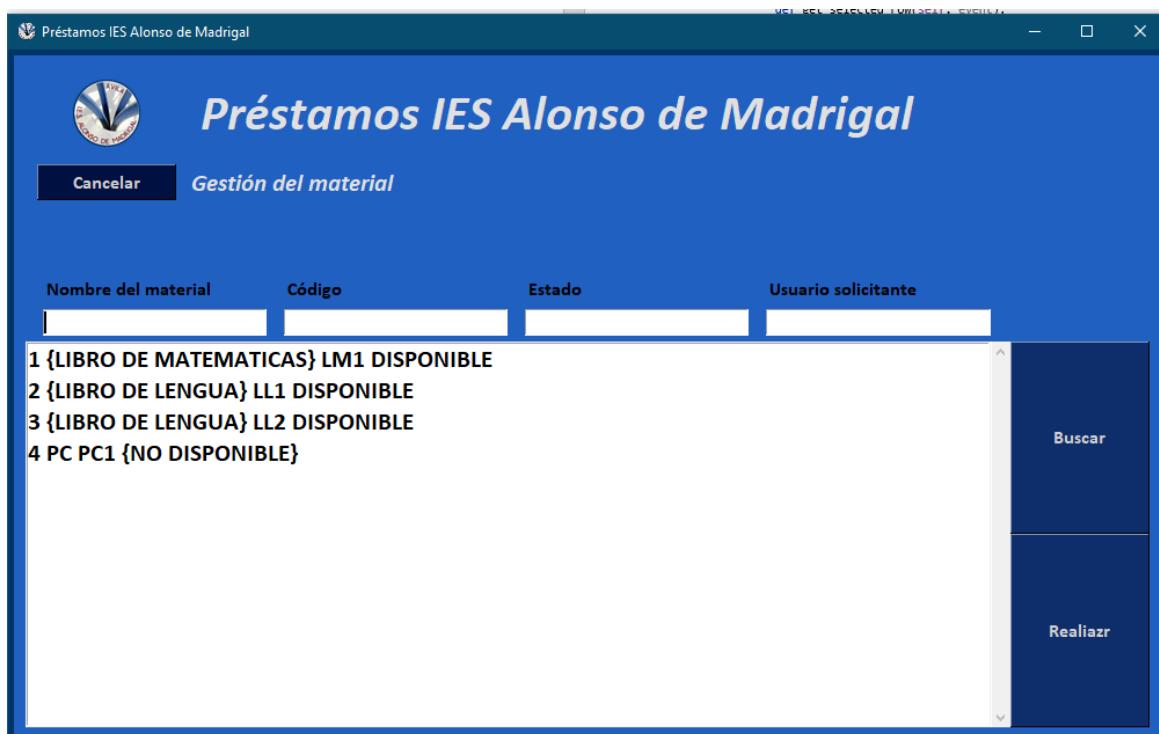


## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

La gestión de los préstamos como administrador o encargado facilita el control de todos los realizados y su solicitud por cualquier usuario. Al realizar uno se conducirá a una pantalla en la que se verá todo el material disponible y su estado.



Img. Pantalla de gestión de préstamos para administradores y encargados

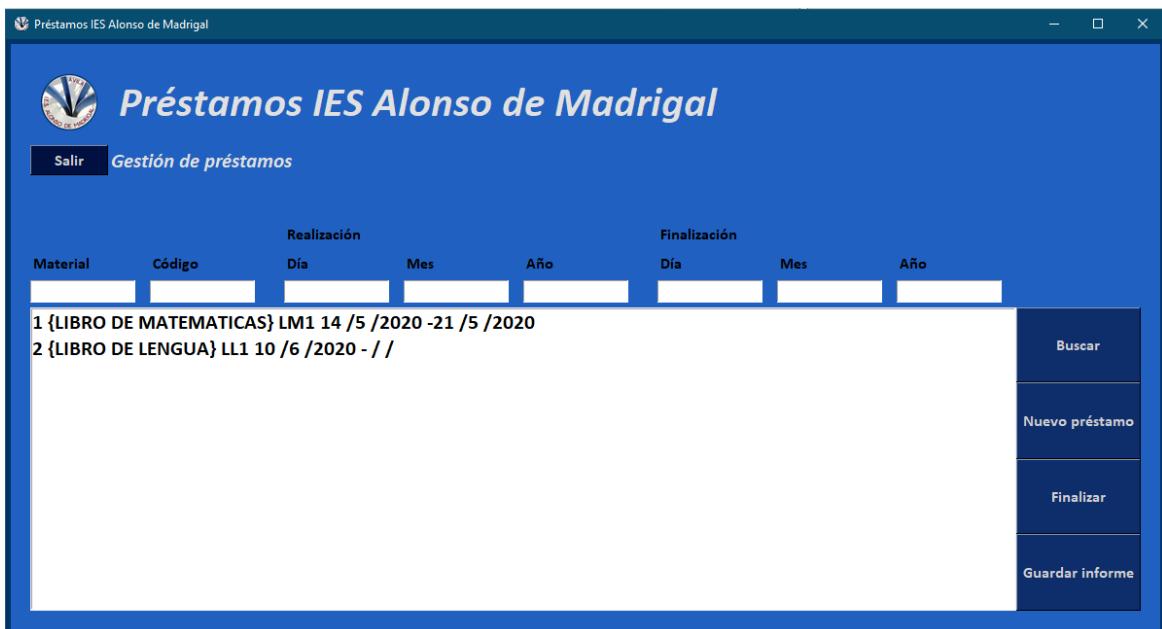


Img. Pantalla de creación de préstamos para administradores y encargados

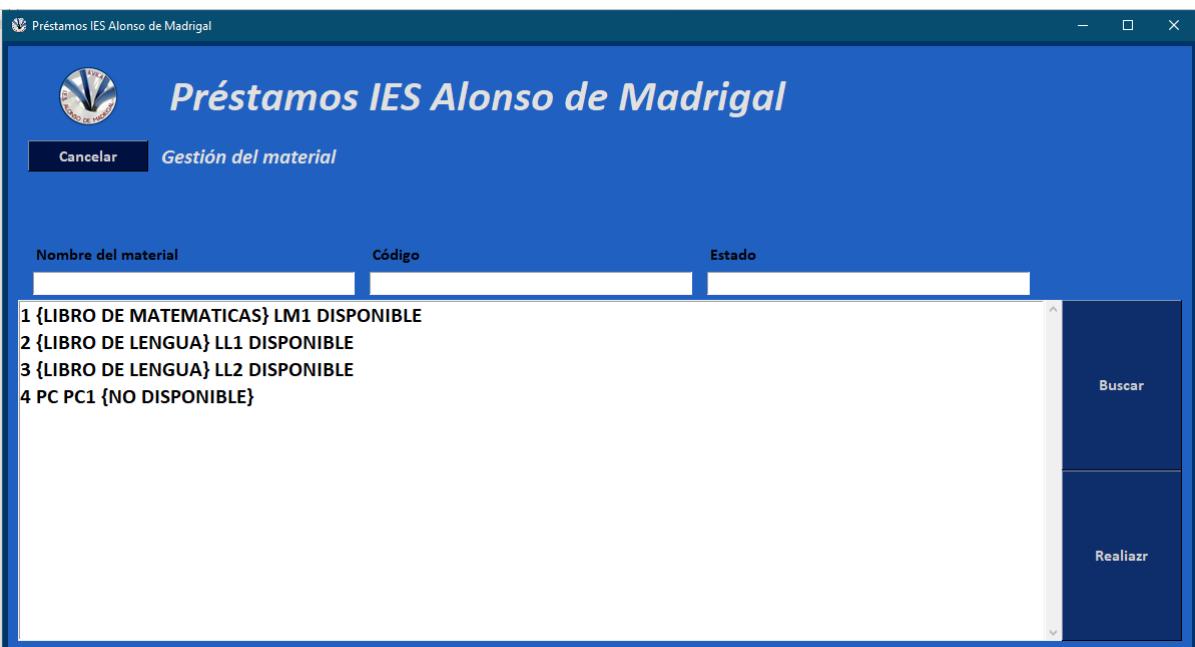


## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

En el caso de acceder a los préstamos como un profesor, solo se podrán gestionar y realizar para dicho profesor.



Img. Pantalla de gestión de préstamos para profesores



Img. Pantalla de creación de préstamos para profesores



#### 4.5. Funciones y procedimientos

Una vez lista la interfaz y la base de datos hay que implementar las funciones y procedimientos pertinentes en el archivo **main.py** para que la interfaz interactúe correctamente con la base de datos y que la información se trate correctamente.

##### - Inicio de sesión

Se comprueba que el usuario y la contraseña que se introduzcan estén registrados y se correspondan, en cuyo caso afirmativo, se accederá a la pantalla de gestión correspondiente con los permisos del usuario.

```

82 #Función para iniciar sesión
83 def login(parent, user, password, controller):
84     global cur, select_page, user_login
85     is_user = False
86     is_password = False
87
88     # Comprobamos que se inicia sesión con un usuario existente
89     cur.execute("SELECT user FROM profesores;")
90     users = cur.fetchall()
91
92     if user == "" or password == "":
93         messagebox.showwarning("Log in incorrecto", "El nombre de usuario y/o la contraseña no se han introducido.")
94     else:
95         for u in users:
96             if str(u) == "("+user+",)":
97                 is_user = True
98
99             if not is_user:
100                 messagebox.showwarning("Log in incorrecto", "El nombre de usuario introducido no está registrado.")
101             else:
102                 cur.execute("SELECT password FROM profesores WHERE user = '"+user+"';")
103                 passwords = cur.fetchall()
104
105                 for p in passwords:
106                     if str(p) == "("+password+",)":
107                         is_password = True
108
109             # Comprobamos que la contraseña introducida es la correspondiente al usuario
110             if not is_password:
111                 messagebox.showwarning("Log in incorrecto", "La contraseña introducida es incorrecta.")
112             else:
113                 if user == "admin":
114                     user_login = user
115                     parent.show_frame("AdminPage")
116                     select_page = "AdminPage"
117                 elif user == "manager":
118                     user_login = user
119                     parent.show_frame("ManagerPage")
120                     select_page = "ManagerPage"
121                 else:
122                     user_login = user
123                     parent.show_frame("PrestamosPageProf")
124                     view_prestamos_command()
125                     select_page = "StartPage"
126
127             # Vaciamos los campos de usuario y contraseña para que no se mantengan al salir y ponemos focus en el campo del usuario
128             controller.ent_user.delete(0, "end")
129             controller.ent_pass.delete(0, "end")
130             controller.ent_user.focus()

```

Img. Código de la función login (inicio de sesión)



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### - Mostrar todo el contenido

Se mostrará en la lista de la pantalla en la que nos encontremos todas las entradas de la tabla de la base de datos correspondiente.

```
672     def view_profes_command(parent):
673         global db
674         parent.list_prof.delete(0, "end")
675         for row in db.view_profes():
676             parent.list_prof.insert("end", row)
```

*Img. Código de la función view\_profes\_command (mostrar todos los profesores)*

### - Buscar según ciertos valores

Aparecerá en la lista de la pantalla en la que nos encontremos todas las entradas correspondientes a los valores de búsqueda introducidos. En caso de no introducir ningún valor, se mostrarán todas las entradas para esa pantalla.

```
677     def search_profe_command(parent):
678         global db
679         if parent.permisos_text.get() == "" and parent.nombre_prof_text.get() == "" and parent.ape1_text.get() == "" and parent.ape2_text.get() == "":
680             view_profes_command(parent)
681         else:
682             parent.list_prof.delete(0, "end")
683             for row in db.search_profe(parent.permisos_text.get(), parent.nombre_prof_text.get(), parent.ape1_text.get(), parent.ape2_text.get(),
684                                     parent.user_text.get(), parent.pass_text.get()):
685                 parent.list_prof.insert("end", row)
```

*Img. Código de la función search\_profes\_command (buscar profesores)*

### - Añadir contenido

Añadirá a la tabla la información correspondiente tras verificar que se ha introducido correctamente.

```
688     def add_profe_command(parent):
689         global db, cur, id_user_departs
690         is_user = False
691         if parent.nombre_prof_text.get() == "" or parent.ape1_text.get() == "" or parent.user_text.get() == "" or parent.pass_text.get() == "":
692             messagebox.showwarning("Error", "Hay campos obligatorios del profesor sin información.")
693         else:
694             # Comprobamos que el nombre de usuario para la aplicación (USER) a añadir no esté ya registrado
695             cur.execute("SELECT user FROM profesores;")
696             users = cur.fetchall()
697
698             for u in users:
699                 if str(u) == ("'"+parent.user_text.get()+"',"):
700                     is_user = True
701                     break
702                 else:
703                     is_user = False
704             if is_user:
705                 messagebox.showwarning("Error", "El nombre de usuario ya existe. Elija uno diferente.")
706             else:
707                 db.insert_profe(parent.permisos_text.get(), parent.nombre_prof_text.get(), parent.ape1_text.get(), parent.ape2_text.get(),
708                                 parent.user_text.get(), parent.pass_text.get())
709                 view_profes_command(parent)
710                 parent.ent_permisos.delete(0, "end")
711                 parent.ent_nombre_prof.delete(0, "end")
712                 parent.ent_ape1.delete(0, "end")
713                 parent.ent_ape2.delete(0, "end")
714                 parent.ent_user.delete(0, "end")
715                 parent.ent_pass.delete(0, "end")
```

*Img. Código de la función add\_profe\_command (añadir profesor)*



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### - Eliminar contenido

Borra de la base de datos una entrada seleccionada en la lista de la pantalla en la que nos encontramos.

```
717 def delete_profe_command(parent):
718     global db
719     try:
720         db.delete_profe(str(selected_tuple_prof[0]))
721         view_profes_command(parent)
722         parent.ent_permisos.delete(0, "end")
723         parent.ent_nombre_prof.delete(0, "end")
724         parent.ent_apel1.delete(0, "end")
725         parent.ent_apel2.delete(0, "end")
726         parent.ent_user.delete(0, "end")
727         parent.ent_pass.delete(0, "end")
728     except:
729         # Alerta en caso de que no haya ninguna entrada seleccionada
730         messagebox.showwarning("Error", "No se ha seleccionado ningún profesor para eliminar.")
```

Img. Código de la función `delete_profe_command` (eliminar profesor)

### - Modificar contenido

Se cambiarán los datos de una entrada seleccionada de la base de datos por los introducidos por el usuario tras comprobar que han sido proporcionados correctamente.

```
732 def update_profe_command(parent):
733     global db, cur
734     is_original = True
735     is_user = False
736     try:
737         if parent.nombre_prof_text.get() == "" or parent.apel1_text.get() == "" or parent.user_text.get() == "" or parent.pass_text.get() == "":
738             messagebox.showwarning("Error", "Hay campos obligatorios del profesor sin información.")
739         else:
740             # Permitimos que el nombre de usuario (USER) pueda ser el que se introdujo originalmente y no cambie
741             cur.execute("SELECT user FROM profesores WHERE id = " + str(selected_tuple_prof[0]))
742             current_user = cur.fetchall()
743             for u in current_user:
744                 if str(u) == "("+parent.user_text.get()+",)":
745                     is_original = True
746                     break
747                 else:
748                     is_original = False
749                     # Comprobamos que el nombre de usuario para la aplicación (USER) a añadir no esté ya registrado
750                     cur.execute("SELECT user FROM profesores;")
751                     users = cur.fetchall()
752                     for u in users:
753                         if str(u) == "("+parent.user_text.get()+",)":
754                             is_user = True
755                             break
756                         else:
757                             is_user = False
758             if not is_original and is_user:
759                 messagebox.showwarning("Error", "El nombre de usuario ya existe. Elija uno diferente.")
760             else:
761                 db.update_profe(str(selected_tuple_prof[0]), parent.permisos_text.get(), parent.nombre_prof_text.get(),
762                                 parent.apel1_text.get(), parent.apel2_text.get(),
763                                 parent.user_text.get(), parent.pass_text.get())
764                 view_profes_command(parent)
765                 parent.ent_permisos.delete(0, "end")
766                 parent.ent_nombre_prof.delete(0, "end")
767                 parent.ent_apel1.delete(0, "end")
768                 parent.ent_apel2.delete(0, "end")
769                 parent.ent_user.delete(0, "end")
770                 parent.ent_pass.delete(0, "end")
771     except:
772         # Alerta en caso de que no haya ninguna entrada seleccionada
773         messagebox.showwarning("Error", "No se ha seleccionado ningún profesor para modificar.")
```

Img. Código de la función `update_profe_command` (modificar profesor)



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### - Finalizar préstamo

Esta función es única de los préstamos, la cual marca como fecha de finalización el día en el que se realice la acción.

```
1758     def finish_prestamo_command(parent):
1759         global db, cur, now
1760         is_finished = False
1761         m = ""
1762         try:
1763             # Comprobamos que el préstamo no haya sido ya finalizado
1764             cur.execute("SELECT dia_fin FROM prestamos WHERE id = "+str(selected_tuple_prest_user[0]))
1765             fin = cur.fetchall()
1766             for f in fin:
1767                 if str(f) == "('')":
1768                     is_finished = False
1769                 else:
1770                     is_finished = True
1771
1772             if is_finished:
1773                 messagebox.showwarning("Error", "El préstamo seleccionado ya ha sido finalizado.")
1774             else:
1775                 # Obtenemos el id del material del que se finaliza el préstamo
1776                 cur.execute("SELECT id FROM material WHERE nombre = ? AND codigo = ?", (parent.mat_prest_text.get(), parent.cod_prest_text.get()))
1777                 id_mat = cur.fetchall()
1778                 for id_m in id_mat:
1779                     for im in id_m:
1780                         m = str(im)
1781                         print(m)
1782                         db.finish_prestamo(str(selected_tuple_prest_user[0]), m, now.day, now.month, now.year)
1783                         view_prestamos_command()
1784                         parent.ent_mat_prest.delete(0, "end")
1785                         parent.ent_cod_prest.delete(0, "end")
1786                         parent.ent_dia_ini.delete(0, "end")
1787                         parent.ent_mes_ini.delete(0, "end")
1788                         parent.ent_ano_ini.delete(0, "end")
1789                         parent.ent_dia_fin.delete(0, "end")
1790                         parent.ent_mes_fin.delete(0, "end")
1791                         parent.ent_ano_fin.delete(0, "end")
1792
1793         except:
1794             # Alerta en caso de que no haya ninguna entrada seleccionada
1795             messagebox.showwarning("Error", "No se ha seleccionado ningún préstamo para finalizar.")
```

Img. Código de la función `finish_prestamo_command` (finalizar préstamo)

### - Generar informe en formato pdf de un préstamo

Otra función propia de los préstamos nos permite generar informes en formato pdf de la información registrada de los mismos en la base de datos.



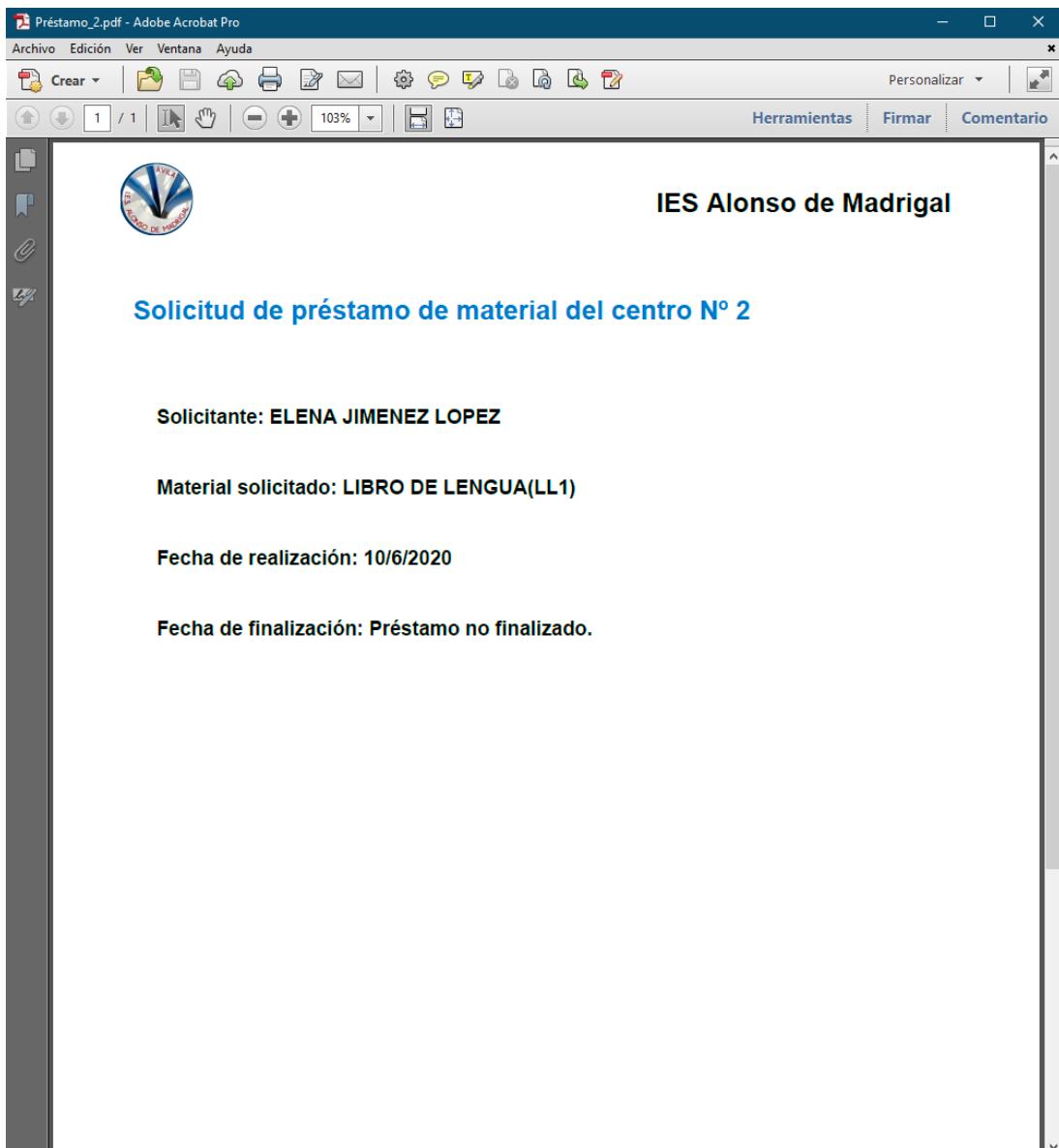
## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

```
1312     def print_prestamo_command(parent):
1313         global db, cur, user_login
1314         try:
1315             # Comprobamos los permisos del usuario para guardar el nombre correcto del realizador del préstamo
1316             fn = ""
1317             cur.execute("""SELECT id_permisos FROM profesores
1318                         WHERE user = """+user_login+"""")
1319
1320             permisos = cur.fetchall()
1321
1322             if str(permisos) == "[[1,]]" or str(permisos) == "[[2,]]":
1323                 user = parent.user_prest_text.get()
1324                 id_prest = str(selected_tuple_prest_admin[0])
1325             else:
1326                 user = user_login
1327                 id_prest = str(selected_tuple_prest_user[0])
1328
1329             # Obtenemos el nombre completo registrado para el usuario del préstamo
1330             cur.execute("""SELECT nombre||' '|ape1||' '|ape2 FROM profesores
1331                         WHERE user = """+user+"""")
1332             full_name = cur.fetchall()
1333             for full_n in full_name:
1334                 for f_n in full_n:
1335                     fn = f_n
1336
1337             # Creamos el pdf
1338             pdf = FPDF()
1339
1340             # Creamos la página
1341             pdf.add_page()
1342             pdf.set_xy(0, 0)
1343             pdf.set_left_margin(16)
1344             pdf.set_right_margin(16)
1345             pdf.set_top_margin(10)
1346
1347             # cabecera de pagina
1348             pdf.image('logo_alomadrigal.png', x=14, y=4, w=16, h=16)
1349             pdf.set_font('Arial', 'B', 16)
1350             pdf.cell(0, 5, "", 0, 1)
1351             pdf.cell(176, 16, "IES Alonso de Madrigal", 0, 1, 'R')
1352
1353             # contenido de la pagina
1354             pdf.set_text_color(30, 120, 200)
1355             pdf.set_font('Arial', 'B', 16)
1356             pdf.cell(0, 10, "", 0, 2)
1357             pdf.cell(50, 10, "Solicitud de préstamo de material del centro N° "+id_prest, 0, 2, 'L')
1358
1359             pdf.cell(0, 10, "", 0, 1)
1360
1361             pdf.set_text_color(0, 0, 0)
1362             pdf.set_font('Arial', 'B', 12)
1363             pdf.cell(5)
1364             pdf.cell(80, 15, "Solicitante: "+fn, 0, 1, 'L')
1365
1366             pdf.set_font('Arial', 'B', 12)
1367             pdf.cell(5)
1368             pdf.cell(80, 15, "Material solicitado: "+parent.mat_prest_text.get()+"+"+parent.cod_prest_text.get(), 0, 1, 'L')
1369
1370             pdf.set_font('Arial', 'B', 12)
1371             pdf.cell(5)
1372             pdf.cell(80, 15, "Fecha de realización: "+parent.dia_ini_text.get()"/"+parent.mes_ini_text.get()"/"+parent.ano_ini_text.get(), 0, 1, 'L')
1373
1374             if parent.dia_fin_text.get() != "" and parent.mes_fin_text.get() != "" and parent.ano_fin_text.get() != "":
1375                 fecha_fin = parent.dia_fin_text.get()"/"+parent.mes_fin_text.get()"/"+parent.ano_fin_text.get()
1376             else:
1377                 fecha_fin = "Préstamo no finalizado."
1378
1379             pdf.set_font('Arial', 'B', 12)
1380             pdf.cell(5)
1381             pdf.cell(80, 15, "Fecha de finalización: "+fecha_fin, 0, 1, 'L')
1382             # Generación del PDF
1383             try:
1384                 file = easygui.filesavebox('SAVE', 'Save File', 'Préstamo_'+id_prest+'.pdf', filetypes=[ "*.pdf" ])
1385                 pdf.output(file, 'F')
1386                 messagebox.showinfo("Informe generado", "El informe ha sido generado con éxito.")
1387             except:
1388                 pass
1389             except:
1390                 # Alerta en caso de que no haya ninguna entrada seleccionada
1391                 messagebox.showwarning("Error", "No se ha seleccionado ningún préstamo del que guardar un informe.")
```

Img. Código de la función print\_prestamo\_command (generar informe de un préstamo)



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.



Img. Muestra de informe en formato pdf de un préstamo

## 5. FASE DE PRUEBAS

Para comprobar el correcto funcionamiento de la aplicación se llevó a cabo una serie de pruebas buscando forzar errores y también introduciendo datos correctos para comprobar que en cualquier caso que se de en un uso real, la aplicación responda correctamente.



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

### 5.1. Pruebas realizadas

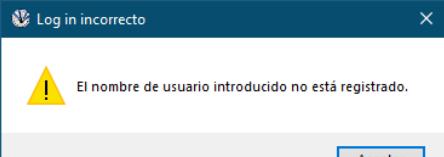
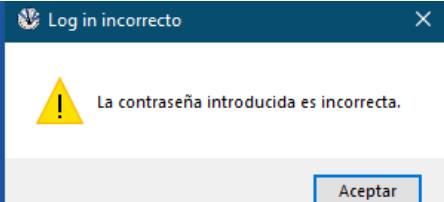
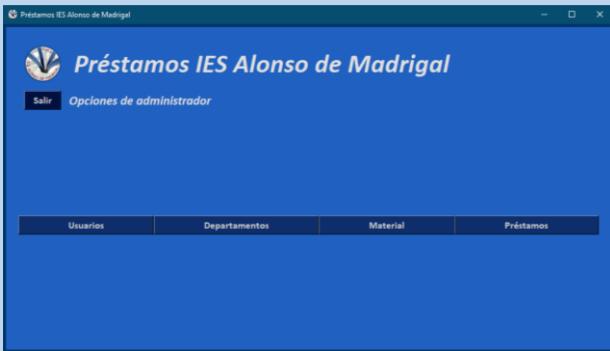
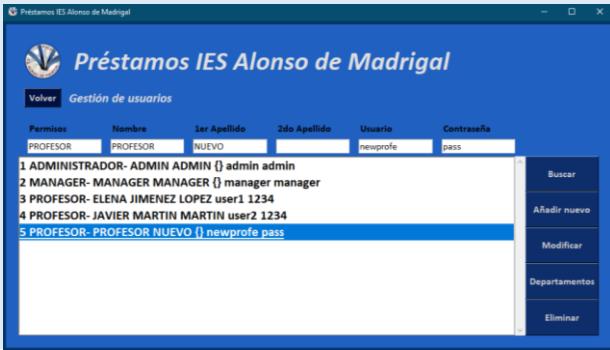
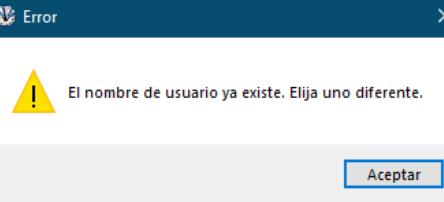
Tbl. Pruebas realizadas

Nº de prueba	Acciones realizadas	Funcionamiento esperado
1	Iniciar sesión sin estar registrado.	Alerta. No está registrado. No se accede a la aplicación.
2	Iniciar sesión con contraseña incorrecta.	Alerta. La contraseña es incorrecta. No se accede a la aplicación.
3	Iniciar sesión estando registrado.	Acceso a la pantalla con los permisos correspondientes.
4	Añadir un profesor totalmente nuevo	Profesor registrado correctamente.
5	Añadir un profesor con nombre de usuario registrado.	Alerta. El nombre de usuario ya está registrado. Elija uno diferente.
6	Eliminar un profesor que pertenezca a varios departamentos.	Profesor y relación con departamentos eliminados, pero no los departamentos.
7	Añadir material totalmente nuevo.	Material registrado correctamente.
8	Añadir material con código ya registrado.	Alerta. El código del material ya está registrado. Elija uno diferente.
9	Establecer material en préstamo como No Disponible.	Alerta. Consulte la disponibilidad con el solicitante del material.
10	Añadir departamento totalmente nuevo.	Departamento registrado correctamente.
11	Añadir departamento ya registrado.	Alerta. El departamento ya está registrado. Elija un nombre diferente.
12	Eliminar departamento al que pertenecen profesores.	Departamento y relación con profesores eliminados, pero no los profesores.
13	Realizar préstamo de material Disponible.	Préstamo realizado correctamente.
14	Realizar préstamo de material No Disponible.	Alerta. El material no está disponible. Préstamo denegado.
15	Finalizar préstamo abierto.	Préstamo finalizado correctamente.
16	Finalizar préstamo cerrado.	Alerta, el préstamo ya ha sido finalizado. Acción denegada.

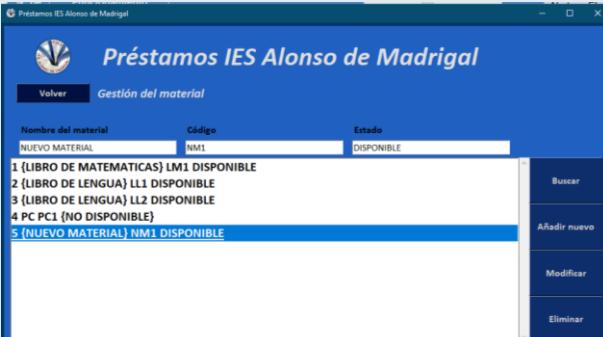
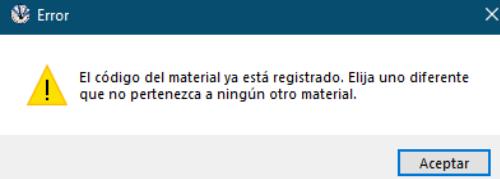
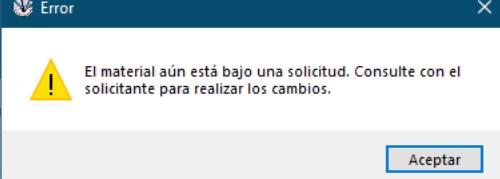
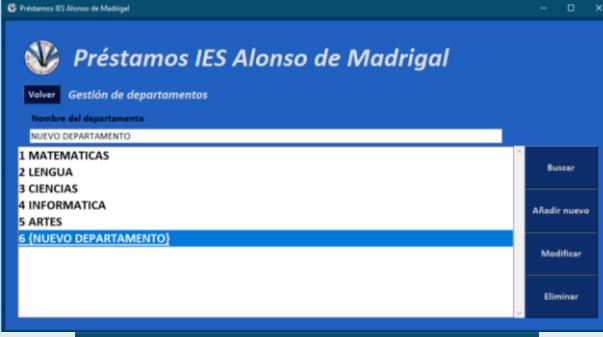
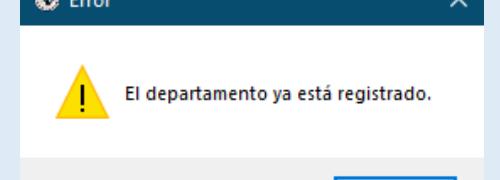


## 5.2. Resultados de las pruebas

Tbl. Resultados de las pruebas

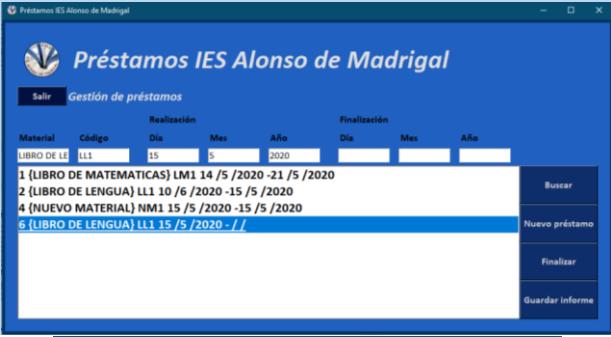
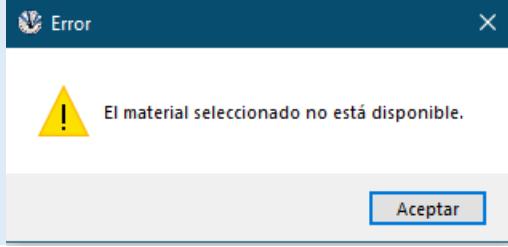
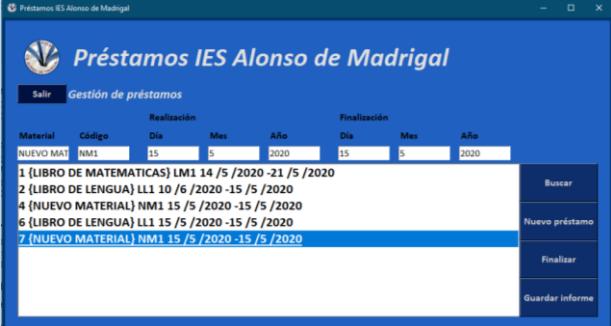
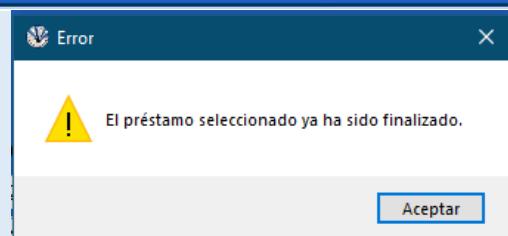
Nº de prueba	Funcionamiento esperado	Resultado
1	Alerta. No está registrado. No se accede a la aplicación.	
2	Alerta. La contraseña es incorrecta. No se accede a la aplicación.	
3	Acceso a la pantalla con los permisos correspondientes.	- Se accedió como administrador. 
4	Profesor registrado correctamente.	- Se introdujeron los datos seleccionados. 
5	Alerta. El nombre de usuario ya está registrado. Elija uno diferente.	
...	...	...

## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

6	Profesor y relación con departamentos eliminados, pero no los departamentos.	<ul style="list-style-type: none"> <li>- Se eliminó el profesor con usuario “user1”, perteneciendo a los departamentos 1, 4 y 5.</li> <li>- Se eliminó su relación, pero no los departamentos.</li> </ul>
7	Material registrado correctamente.	
8	Alerta. El código del material ya está registrado. Elija uno diferente.	
9	Alerta. Consulte la disponibilidad con el solicitante del material.	
10	Departamento registrado correctamente.	<ul style="list-style-type: none"> <li>- Se introdujeron los datos seleccionados.</li> </ul> 
11	Alerta. El departamento ya está registrado. Elija un nombre diferente.	
...	...	...



## APLICACIÓN DE CONTROL DE RECURSOS PARA UN CENTRO EDUCATIVO.

12	Departamento y relación con profesores eliminados, pero no los profesores.	<ul style="list-style-type: none"> <li>- Se eliminó el departamento “CIENCIAS” al cual pertenecía el profesor con usuario “newuser”.</li> <li>- Se eliminó su relación, pero no los profesores.</li> </ul>
13	Préstamo realizado correctamente.	
14	Alerta. El material no está disponible. Préstamo denegado.	
15	Préstamo finalizado correctamente.	<ul style="list-style-type: none"> <li>- Se usó el préstamo seleccionado para finalizar.</li> </ul> 
16	Alerta, el préstamo ya ha sido finalizado. Acción denegada.	



## 6. CONCLUSIONES FINALES

El resultado obtenido tras las pruebas realizadas y la compilación final de un ejecutable de la aplicación se puede considerar satisfactorio.

Los objetivos marcados por el proyecto se cumplen, los cuales eran obtener una aplicación con un diseño responsive para que el personal de un centro docente, en este caso el IES Alonso de Madrigal, gestione los préstamos de material a los profesores.

A pesar de haber cumplido los objetivos, cabe la posibilidad de llevar a cabo mejoras en el diseño responsive de la interfaz para que se adapte mejor a pantallas más grandes, simplificar el manejo de la información y que no sea necesario introducir todos los datos siempre escribiendo o que el informe pueda llevar algo más de contenido y se muestre con diseño diferente.

## 7. DIFICULTADES

- El ícono de la barra de navegación de la aplicación no sale correctamente a pesar de ser un archivo con formato .ico. Se consiguió solucionar gracias a generar un archivo.ico desde el portal web [www.imagen.online-convert.com/es/convertir-a-ico](http://www.imagen.online-convert.com/es/convertir-a-ico).
- Inicialmente la información de la base de datos se mostraría sobre labels, pero solo se mostraba el último valor de la última entrada devuelta por la misma. Se solventó implementando una lista en la que se muestra la información en vez de las labels.
- Al compilar el ejecutable surgía un error al compilar y una vez compilado, al ejecutar. La solución fue reinstalar Python para que no diera problemas al realizar la compilación y, una vez compilado, poner las imágenes y la base de datos (si ya está creada tras haber hecho pruebas) en la misma carpeta que el ejecutable (carpeta dist).



## 8. BIBLIOGRAFÍA

- Hektor Profe: <https://docs.hektorprofe.net/python/>
- Stackoverflow: <https://stackoverflow.com/>
- Manjurul Hoque (canal de YouTube):  
[https://www.youtube.com/channel/UCN36jH1gauTEzZOb\\_joeusA](https://www.youtube.com/channel/UCN36jH1gauTEzZOb_joeusA)
- Codemy.com (canal de YouTube):  
<https://www.youtube.com/channel/UCFB0dxMudkws1q8w5NJEAmw>
- Python GUI Programming with Tkinter (libro de Alan D. Moore).

## 9. ANEXOS

Anexo I. Manual de instalación.

Anexo II. Manual de uso.

Anexo III. Presentación de diapositivas.

