P.Web

Manual técnico

Gestión de matriculación y calificación

Manual técnico

Rafael Carlos Soriano Mármol
I02somar@uco.es
Programación Web

Índice

1. Introducción y datos de acceso

Una herramienta de tal envergadura como puede ser Sigma o cualquier sistema de gestión de matriculaciones y calificaciones requiere un tiempo de desarrollo considerable de un buen equipo de trabajo, pues se trata de una aplicación muy compleja que será la base del funcionamiento de una universidad.

Durante el desarrollo de esta aplicación web, he tratado de incorporar las máximas funcionalidades posibles, de manera que he desarrollado una herramienta funcional bastante completa que, sin embargo, se puede mejorar y optimizar con la inclusión de numerosas funcionalidades y gestiones.

En mi aplicación he incluido:

- Gestión básica de usuarios
- Matriculación y gestión de alumnos
- Consulta y gestión de calificaciones
- Gestión de titulaciones
- Gestión de asignaturas
- Gestión de profesores
- Formulario de contacto
- Mejoras importantes en el panel de administración de Django

He de destacar que, como he indicado en el último punto, he realizado diversas mejoras en el panel de administración, pues en una herramienta de estas características la labor del equipo de administración es de gran importancia.

En la aplicación se han incluido diversos ejemplos de los distintos apartados. Los datos de acceso para diversos usuarios son:

Rol	*Nombre Usuario*	*Contraseña*
Administrador	i02somar	i02somar
Usuario sin matricular	pruebaSinMatricula	prueba
Usuario matriculado	pruebaMatriculado	prueba

2. Modelo

2.1. Introducción al sistema y visión general de la base de datos

Se ha separado la gestión de usuarios de la aplicación web de la gestión de las matriculaciones, a fin de facilitar la gestión por separado de los distintos tipos de usuarios (como pueden ser alumnos, profesores u otro personal de la universidad) del sistema de matriculaciones.

En la aplicación llamada "users" se encuentra la gestión de usuarios de la página. Se trata de una adaptación de la aplicación ya realizada en prácticas utilizando la clase "User" proporcionada por Django.

En la aplicación llamada "principal" se encuentran definidos los modelos del sistema de gestión de matriculación y calificación. A continuación se expondrán las tablas de la base de datos mostrando la salida del comando "*python manage.py sql principal*", y posteriormente se detallará un diagrama UML que mostrará tanto las tablas como aclaraciones sobre sus relaciones.

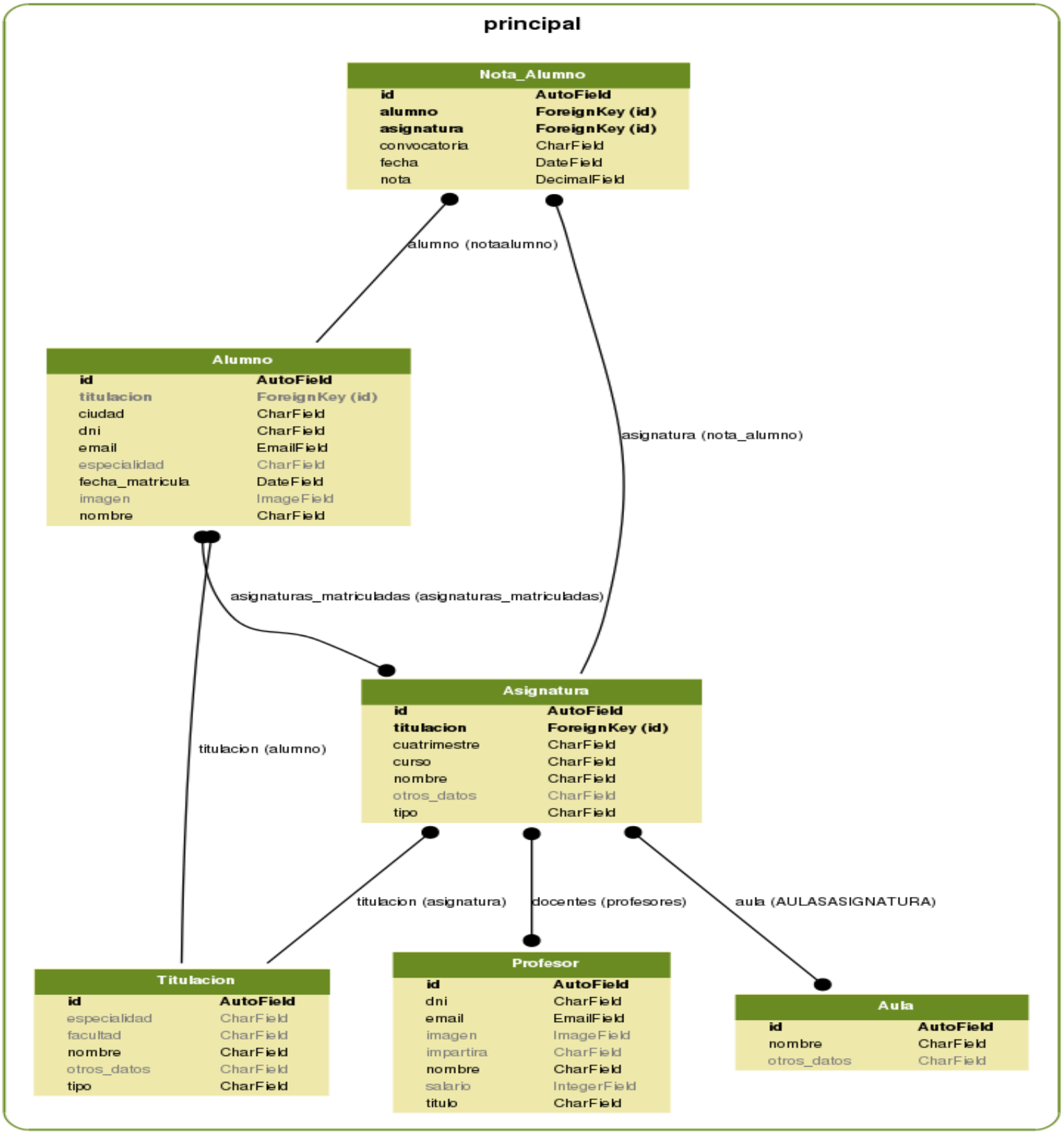
Nota: las clases del modelo han sido validadas con el comando:
"*python manage.py validate*"

```
CREATE TABLE "principal_profesor" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(50) NOT NULL,
    "dni" varchar(10) NOT NULL UNIQUE,
    "email" varchar(50) NOT NULL UNIQUE,
    "titulo" varchar(30) NOT NULL,
    "salario" integer,
    "impartira" varchar(13),
    "imagen" varchar(100)
);
CREATE TABLE "principal_aula" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(30) NOT NULL UNIQUE,
    "otros_datos" varchar(300)
);
CREATE TABLE "principal_titulacion" (
    "id" integer NOT NULL PRIMARY KEY,
    "nombre" varchar(30) NOT NULL UNIQUE,
    "tipo" varchar(13) NOT NULL,
    "facultad" varchar(30),
    "especialidad" varchar(300),
    "otros_datos" varchar(300)
);
```

```
CREATE TABLE "principal_asignatura_docentes" (
   "id" integer NOT NULL PRIMARY KEY,
   "asignatura_id" integer NOT NULL,
   "profesor_id" integer NOT NULL REFERENCES "principal_profesor" ("id"),
   UNIQUE ("asignatura_id", "profesor_id")
);
CREATE TABLE "principal_asignatura_aula" (
   "id" integer NOT NULL PRIMARY KEY,
   "asignatura_id" integer NOT NULL,
   "aula_id" integer NOT NULL REFERENCES "principal_aula" ("id"),
   UNIQUE ("asignatura_id", "aula_id")
);
CREATE TABLE "principal_asignatura" (
   "id" integer NOT NULL PRIMARY KEY,
   "nombre" varchar(30) NOT NULL,
   "otros_datos" varchar(300),
   "titulacion_id" integer NOT NULL REFERENCES "principal_titulacion" ("id"),
   "curso" varchar(13) NOT NULL,
   "cuatrimestre" varchar(13) NOT NULL,
   "tipo" varchar(35) NOT NULL
);
CREATE TABLE "principal_alumno_asignaturas_matriculadas" (
   "id" integer NOT NULL PRIMARY KEY,
   "alumno_id" integer NOT NULL,
   "asignatura_id" integer NOT NULL REFERENCES "principal_asignatura" ("id"),
   UNIQUE ("alumno_id", "asignatura_id")
);
CREATE TABLE "principal_alumno" (
   "id" integer NOT NULL PRIMARY KEY,
   "nombre" varchar(30) NOT NULL,
   "dni" varchar(10) NOT NULL UNIQUE,
   "ciudad" varchar(20) NOT NULL,
   "email" varchar(50) NOT NULL UNIQUE,
   "imagen" varchar(100),
   "titulacion_id" integer REFERENCES "principal_titulacion" ("id"),
   "especialidad" varchar(30),
   "fecha_matricula" date NOT NULL
);
CREATE TABLE "principal_nota_alumno" (
   "id" integer NOT NULL PRIMARY KEY,
   "alumno_id" integer NOT NULL REFERENCES "principal_alumno" ("id"),
   "asignatura_id" integer NOT NULL REFERENCES "principal_asignatura" ("id"),
   "fecha" date NOT NULL,
   "convocatoria" varchar(13) NOT NULL,
   "nota" decimal NOT NULL
);
```

2.2. Diagrama UML

Este diagrama se ha realizado utilizando la herramienta "graph_models" proporcionada por "django_extensions":



- En el diagrama, se resaltan los distintos campos de las clases/tablas de la siguiente manera:
  - **Negrita:** Clave primaria (ID) y claves foráneas.
  - **Gris:** Campos opcionales
- Un alumno puede tener varias nota_alumno, y una nota_alumno debe pertenecer a un solo alumno: *alumno es clave foránea de la clase Nota_Alumno, recibiendo sus datos de la clase Alumno*
  - *Relación 1:N*
- Una nota_alumno debe pertenecer a una única asignatura, y una asignatura puede tener varias nota_alumno: *asignatura es clave foránea de la clase Nota_Alumno, recibiendo sus datos de la clase Asignatura*
  - *Relación 1:N*
- Un alumno debe pertenecer a una única titulación, y una titulación puede tener varios alumnos: *titulación(clase Alumno) es clave foránea, recibiendo sus datos de la clase/tabla Titulación*
  - *Relación 1:N*
- Un alumno puede estar matriculado en varias asignaturas, y una asignatura puede tener varios alumnos matriculados
  - *Relación N:N*
- Una asignatura debe pertenecer a una titulación, y una titulación debe incluir varias asignaturas. *titulación (de la clase Asignatura) es clave foránea, recibiendo sus datos de la clase/tabla Titulación*
  - *Relación 1:N*
- Un profesor puede impartir clases de varias asignaturas, y una asignatura puede ser impartida por varios profesores
  - *Relación N:N*
- Una asignatura puede tener asignadas varias aulas, y un aula puede ser asignada para varias asignaturas
  - *Relación N:N*

3. Página de inicio y plantilla base. Login en el sistema

Nota: todas las plantillas se encuentran en el directorio pw->plantillas

La página de inicio (inicio.html) de la que derivan todas las demás (plantilla base) se ha realizado utilizando bootstrap, de manera que dispone de una visualización moderna que se adapta a dispositivos móviles (ver última sección del manual de usuario para ver ejemplos de visualización en un dispositivo móvil).

La plantilla comprobará si el usuario está logueado en el sistema utilizando *{% if user.is_authenticated %}* tanto para mostrar el mensaje de bienvenida y link de "logout" como para ofrecer la funcionalidad/link de matriculación.

Para mostrar el menú desplegable de Administración, la plantilla comprobará si el usuario logueado es un administrador utilizando: *{% if user.is_active and user.is_staff %}*. En caso contrario, este menú no se mostrará.

En cuanto a la muestra del link para comprobar las calificaciones de un alumno, la plantilla comprobará si el usuario logueado está matriculado, utilizando *{% if alumno.id != None %}*. Esto se ha realizado en la vista de inicio (views.py de la aplicación "principal") capturando en una variable el email del usuario logueado en el sistema y posteriormente comparándolo con los emails almacenados en la base de datos de alumnos matriculados (al realizar la matrícula se indica que el email ha de ser el mismo que el utilizado para el usuario del sistema) de la siguiente manera:

```
EMAIL= request.session.get('email')
try:
        alumno=          Alumno.objects.get(email=EMAIL)
except Alumno.DoesNotExist:
        alumno = None
```

Previamente, para poder utilizar las variables registradas en la sesión de manera correcta, se ha de añadir al fichero de configuración:

```
from django.conf import global_settings
TEMPLATE_CONTEXT_PROCESSORS=
global_settings.TEMPLATE_CONTEXT_PROCESSORS + (
    'django.core.context_processors.request',
)
```

En la vista de la página principal también se ha incluido un formulario para realizar el login. El sistema de login se ha realizado con todas las comprobaciones "try/except" para garantizar un logueo correcto, o, en su defecto elegir la notificación adecuada. Todo ello además de registrar la citada variable de sesión email en el momento del acceso: *request.session['email'] = USER.email*

4. Presentación de datos. Seguridad

Para la presentación de los datos se ha realizado de la manera común en Django y realizada tanto en las prácticas de la asignatura como en la teoría. En los casos en los que se requiere que el usuario sea administrador (como para ver los datos extra de los profesores) en las plantillas se ha realizado la comprobación citada previamente: *{% if user.is_active and user.is_staff %}.*

Los menús que solo deben ser visualizados y accedidos por el usuario administrador (como por ejemplo, modificar/añadir/eliminar profesores o consulta del listado de calificaciones generales), se ha utilizado en las vistas correspondientes el decorador "@staff_member_required", y los apartados en los que el usuario ha de estar logueado se ha incluido el decorador "@login_required" (como por ejemplo, consulta de calificaciones privadas de un alumno).

5. Matriculaciones

Para el sistema de matriculaciones, he utilizado el sistema descrito previamente en la sección de la página principal. En primer lugar, en la vista se comprobará el email del usuario logueado con la base de datos de alumnos, puesto que se especifica que el email debe ser el mismo que el utilizado para el registro en el sistema.

```
EMAIL= request.session.get('email')
        try:
                alumno= Alumno.objects.get(email=EMAIL)
        except Alumno.DoesNotExist:
                alumno = None
```

Una vez en la vista, en la plantilla se comprobará si ha habido alguna coincidencia en la base de datos: *{% if request.session.email == alumno.email %}.* En ese caso, se mostrará una notificación al usuario de que ya se encuentra matriculado, y que para realizar cualquier cambio se dirija a la secretaría de su centro o utilice el formulario de contacto.

6. Calificaciones privadas de un alumno

Como se ha explicado previamente, para acceder a este link el usuario debe estar logueado y matriculado. En la vista correspondiente (*def calificaciones_privadas(request,id_usuario):*) se tomaran los datos del alumno y todos los datos de las calificaciones. En la plantilla, al margen del bucle para recorrer todas las notas almacenadas, se realizara una comprobación para mostrar únicamente las notas correspondientes a dicho alumno:

```
{% for calificacion in dato %}
        {% if alumno.nombre == calificacion.alumno.nombre %}
```

De esta manera, se comprobará el nombre del alumno consultante (alumno.nombre) con el registrado en la base de datos (calificacion.alumno.nombre). En caso de que el alumno tratara de acceder a calificaciones de otro alumno, se le notificará un error de permisos.

7. Formulario de contacto

La realización de un formulario de contacto funcional es muy sencilla en django. Solo he tenido que realizar un formulario simple e incluir en el fichero configuración los datos de una cuenta google que he creado para el proyecto:

```
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER =
'fakepwcontacto@gmail.com'EMAIL_HOST_PASSWORD
= 'programacionweb'
EMAIL_PORT = 587
```

8. Modificaciones del panel de administración

Como he comentado previamente, el panel de administración es muy importante para una herramienta de estas características.

En primer lugar, para detectar el idioma del navegador y poder mostrar el panel de administración en dicho idioma, se ha añadido al fichero de configuración settings.py las siguientes líneas:

```
MIDDLEWARE_CLASSES = (
    'django.middleware.locale.LocaleMiddleware',
)
```

8.1. Modificaciones del archivo "admin.py" (aplicación "principal")

Para poder añadir varias notas a un alumno de manera fácil y sencilla, en primer lugar se diseña una clase con datos extra del modelo Nota_Alumno, que será incluida tanto en los registros de la clase Alumno(llamada AlumnoAdmin) como en la de asignaturas.

La clase sería esta:

```
class InlineNota(admin.TabularInline):
        model = Nota_Alumno
        extra = 5 #numero de campos extras
```

A modo de explicación de todas las mejoras incluidas, voy a exponer la clase AlumnoAdmin del archivo "admin.py" de la aplicación principal detallando cada campo.

class AlumnoAdmin(admin.ModelAdmin):

inlines = [InlineNota] *#Citado previamente. Permite añadir varias notas desde la ficha de edición de un alumno*

list_display = ('nombre', 'dni', 'email', 'titulacion', 'fecha_matricula', 'matricula_Reciente') *#Permite mostrar más datos de un alumno en el panel de administración sin necesidad de entrar en su ficha*

list_filter = ['titulacion'] *#barra lateral de filtrado por titulación*

list_per_page = 10 *#muestra 10 alumnos por página. Mejora la visualización*

search_fields = ['nombre', 'dni', 'email', 'titulacion'] *#búsqueda por datos del alumno*

date_hierarchy = 'fecha_matricula' *#filtrado superior según fechas de las matriculaciones*

filter_horizontal = ('asignaturas_matriculadas',) *#mejor visualización de manytomany*

En otros casos, se ha incluido la opción de poder editar algunos campos sin necesidad de entrar a la ficha de edición correspondiente. Esto se realiza con la opción:

"list_editable = ['campo1', '…',]"

Por ejemplo, un administrador podría cambiar el título de un profesor o su dirección de correo electrónico sin entrar en su ficha, directamente desde el listado de profesores, ya que se ha establecido la opción de la siguiente manera:

" list_editable = ['titulo', 'email'] "