



# PROGRAMACIÓN WEB 2



**Profesor(a):**

Carlo Jose Luis Corrales Delgado

**Estudiantes:**

Mamani Anahua, Victor Narciso  
Cuno Cahuari, Armando Steven

**Repositorio GitHub:**

<https://github.com/VictorMA18/Lab06-Django>

**Video:**

28 de mayo, 2024

## Clases en Models.py

Este código define tres modelos en Django: Alumno, Curso, y Nota. El modelo Alumno tiene un campo nombre de tipo CharField con un máximo de 255 caracteres. El modelo Curso contiene un campo curso que es un CharField de hasta 100 caracteres y debe ser único. El modelo Nota establece relaciones con los modelos Alumno y Curso mediante claves foráneas (ForeignKey), y tiene tres campos nota, nota2, y nota3, todos ellos de tipo DecimalField con un máximo de cuatro dígitos y dos decimales, inicializados en 0. Además, cada modelo tiene un método str que retorna una representación legible del objeto.

```
1 from django.db import models
2
3 class Alumno(models.Model):
4     nombre = models.CharField(max_length=255)
5
6     def __str__(self):
7         return self.nombre
8
9 class Curso(models.Model):
10     curso = models.CharField(max_length=100, unique=True)
11     def __str__(self):
12         return self.curso
13
14 class Nota(models.Model):
15     alumno = models.ForeignKey(Alumno, on_delete=models.CASCADE)
16     curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
17     nota = models.DecimalField(max_digits=4, decimal_places=2, default=0)
18     nota2 = models.DecimalField(max_digits=4, decimal_places=2, default=0)
19     nota3 = models.DecimalField(max_digits=4, decimal_places=2, default=0)
```

Figura 1: Código de Models

## Funciones en Views.py

El código proporciona seis funciones de vista en Django para gestionar un sistema escolar. La función crear-alumno(request) maneja la creación de nuevos registros de alumnos, mientras que lista-alumnos(request) recupera y muestra todos los registros de alumnos. Por otro lado, crear-curso(request) gestiona la creación de nuevos registros de cursos, y lista-cursos(request) muestra todos los cursos existentes. Además, crear-nota(request) permite la creación de nuevas notas, y lista-notas(request) muestra todas las notas registradas. Cada función valida y procesa los formularios correspondientes, redirigiendo a las páginas de lista respectivas después de guardar los datos en la base de datos. En conjunto, estas funciones proporcionan una interfaz completa para gestionar los datos de alumnos, cursos y notas en el sistema de notas.

```
1 from django.shortcuts import render, redirect
2 from .forms import AlumnoForm, CursoForm, NotaForm
3 from .models import Alumno, Curso, Nota
4 # Create your views here.
5 def crear_alumno(request):
6     if request.method == 'POST':
7         form = AlumnoForm(request.POST)
8         if form.is_valid():
9             form.save()
10            return redirect('lista_alumnos')
11        else:
12            form = AlumnoForm()
13            return render(request, 'colegio/crear_alumno.html', {'form': form})
14
15 def lista_alumnos(request):
16     alumnos = Alumno.objects.all()
17     return render(request, 'colegio/lista_alumnos.html', {'alumnos': alumnos})
18
19 def crear_curso(request):
20     if request.method == 'POST':
21         form = CursoForm(request.POST)
22         if form.is_valid():
23             form.save()
24             return redirect('lista_cursos')
25        else:
26            form = CursoForm()
27            return render(request, 'colegio/crear_curso.html', {'form': form})
```

Figura 2: Código de Views

```
29 def lista_cursos(request):
30     cursos = Curso.objects.all()
31     return render(request, 'colegio/lista_cursos.html', {'cursos': cursos})
32
33 def crear_nota(request):
34     if request.method == 'POST':
35         form = NotaForm(request.POST)
36         if form.is_valid():
37             form.save()
38             return redirect('lista_notas')
39        else:
40            form = NotaForm()
41            return render(request, 'colegio/crear_nota.html', {'form': form})
42
43 def lista_notas(request):
44     notas = Nota.objects.all()
45     return render(request, 'colegio/lista_notas.html', {'notas': notas})
```

Figura 3: Código de Views

## Funciones en Forms.py

El código proporciona tres formularios en Django para los modelos de datos relacionados con un sistema escolar. El formulario `AlumnoForm` está asociado al modelo `Alumno`, permitiendo la creación de nuevos registros de alumnos con un campo para el nombre del alumno. `CursoForm` está vinculado al modelo `Curso`, facilitando la creación de nuevos registros de cursos. Por último, `NotaForm` se asocia al modelo `Nota`, ofreciendo campos para ingresar información sobre la nota de un alumno en un curso específico, incluyendo múltiples notas si es necesario. Cada formulario está diseñado para interactuar directamente con su respectivo modelo, simplificando el proceso de captura y almacenamiento de datos relacionados con el sistema escolar.

```
1 from django import forms
2 from .models import Alumno, Curso, Nota
3
4 class AlumnoForm(forms.ModelForm):
5     nombre = forms.CharField(max_length=255, label='Nombre del Alumno')
6
7     class Meta:
8         model = Alumno
9         fields = ['nombre']
10
11     def save(self, commit=True):
12         alumno = super().save(commit=False)
13         if commit:
14             alumno.save()
15         return alumno
16
17 class CursoForm(forms.ModelForm):
18     class Meta:
19         model = Curso
20         fields = ['curso']
21
22 class NotaForm(forms.ModelForm):
23     class Meta:
24         model = Nota
25         fields = ['alumno', 'curso', 'nota', 'nota2', 'nota3']
```

Figura 4: Código de Forms

## Funciones en Urls.py en la aplicación

El bloque de código define un conjunto de rutas en Django que asignan solicitudes HTTP a funciones de vista específicas dentro de la aplicación. Cada ruta, configurada mediante el método `path()`, está asociada a una función de vista correspondiente en el módulo de vistas. Por ejemplo, `/crearAlumno/` dirige las solicitudes a la función `crear-alumno`, encargada de la creación de nuevos registros de alumnos, mientras que `/listaAlumno/` muestra la lista de todos los alumnos registrados utilizando la función `lista-alumnos`. Además, `/crearNota/` dirige las solicitudes a la función `crear-nota` para crear nuevas notas, y `/listaNota/` muestra todas las notas registradas utilizando la función `lista-notas`. Este enfoque permite a los usuarios interactuar con diversas funcionalidades del sistema de notas, como la gestión de alumnos, cursos y notas, a través de URLs específicas y funciones de vista asociadas.

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('crearAlumno/', views.crear_alumno, name='crear_alumno'),
6     path('listaAlumno/', views.lista_alumnos, name='lista_alumnos'),
7     path('crearCurso/', views.crear_curso, name='crear_curso'),
8     path('listaCurso/', views.lista_cursos, name='lista_cursos'),
9     path('crearNota/', views.crear_notas, name='crear_notas'),
10    path('listaNota/', views.lista_notas, name='lista_notas'),
11]
```

Figura 5: Código de Urls.py en la aplicación

## Funciones en Urls.py en el proyecto

El bloque de código configura las rutas principales de la aplicación Django. La ruta `/admin/` dirige a la interfaz de administración predeterminada de Django, mientras que la ruta `/notas/` es manejada por la aplicación `colegio.urls`, que a su vez incluye un conjunto de rutas definidas en el archivo `urls.py`.

de la aplicación colegio. Esto permite una estructura de URL modular, donde las rutas relacionadas con las notas (definidas en colegio.urls) se pueden gestionar de manera independiente de las rutas de administración u otras partes de la aplicación.

```
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('notas/', include('colegio.urls'))
23 ]
```

Figura 6: Código de Urls.py en el proyecto

## Los archivos HTML

### crear\_alumno.html

```
1 {% load static %}
2 <!DOCTYPE html>
3 <html>
4 <head>
5     <title>Crear Alumnos</title>
6     <link rel="stylesheet" type="text/css" href="{% static 'css/style_formulario.css'%}"></link>
7 </head>
8 <body>
9     <div class="content">
10         <h1>Crear Alumno</h1>
11         <form method="post">
12             {% csrf_token %}
13             {{ form.as_p }}
14             <button type="submit">Guardar</button>
15         </form>
16         <a href="{% url 'lista_alumnos' %}">Ver todas los alumnos</a>
17     </div>
18 </body>
19 </html>
```

Figura 7: Código de crearAlumno.html

# Crear Alumno

Nombre del Alumno:

Figura 8: Pagina

lista\_alumnos.html

```
1  {% load static %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>Lista de Alumnos</title>
6      <link rel="stylesheet" type="text/css" href="{% static 'css/style_lista.css'%}"></link>
7  </head>
8  <body>
9      <div class="content">
10         <h1>Lista de Alumnos</h1>
11         <ul>
12             {% for alumno in alumnos %}
13                 <li>{{ alumno.nombre }}</li>
14             {% endfor %}
15         </ul>
16         <a href="{% url 'crear_alumno' %}">Crear nuevo alumno</a>
17     </div>
18 </body>
19 </html>
```

Figura 9:Codigo de listaAlumnos.html



Figura 10: Pagina

## crear\_curso.html

```
1  {% load static %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>Crear Cursos</title>
6      <link rel="stylesheet" type="text/css" href="{% static 'css/style_formulario.css'%}"></link>
7  </head>
8  <body>
9      <div class="content">
10         <h1>Crear Curso</h1>
11         <form method="post">
12             {% csrf_token %}
13             {{ form.as_p }}
14             <button type="submit">Guardar</button>
15         </form>
16         <a href="{% url 'lista_cursos' %}">Ver todas los cursos</a>
17     </div>
18 </body>
19 </html>
```

Figura 11: Codigo de crearCurso.html



# Crear Curso

Curso:

Figura 12: Pagina

lista\_cursos.html

```
1  {% load static %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>Lista de Cursos</title>
6      <link rel="stylesheet" type="text/css" href="{% static 'css/style_lista.css'%}"></link>
7  </head>
8  <body>
9      <div class="content">
10         <h1>Lista de Cursos</h1>
11         <ul>
12             {% for curse in cursos %}
13                 <li>{{ curse.curso }}</li>
14             {% endfor %}
15         </ul>
16         <a href="{% url 'crear_curso' %}">Crear nuevo curso</a>
17     </div>
18 </body>
19 </html>
```

Figura 13: Código de listaCurso.html



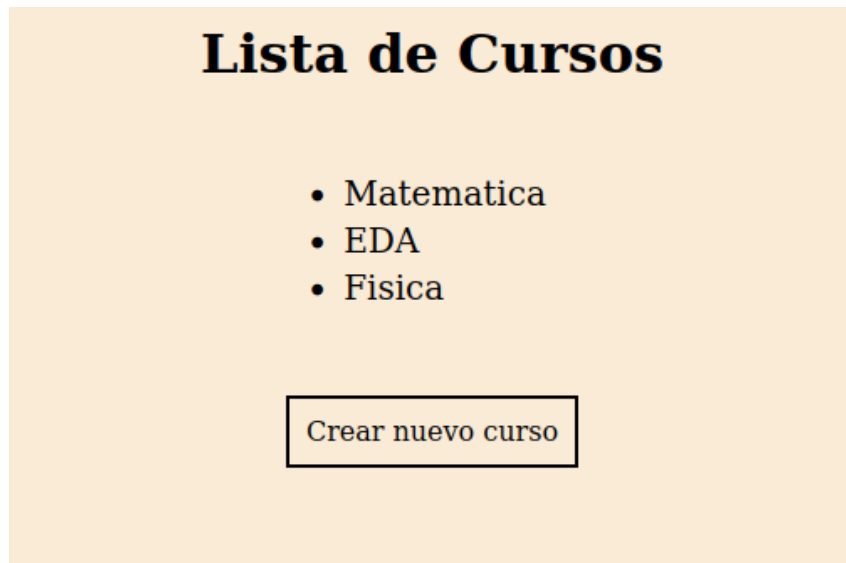


Figura 14: Pagina

## crear\_notas.html

```
1  {% load static %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>Crear Notas</title>
6      <link rel="stylesheet" type="text/css" href="{% static 'css/style_crearNota.css'%}"></link>
7  </head>
8  <body>
9      <div class="content">
10         <h1>Crear Nota</h1>
11         <form method="post">
12             {% csrf_token %}
13             {{ form.as_p }}
14             <button type="submit">Guardar</button>
15         </form>
16         <a href="{% url 'lista_notas' %}">Ver todas las notas</a>
17     </div>
18 </body>
19 </html>
```

Figura 15: Codigo de crearNota.html

## Crear Nota

**Alumno:**

**Curso:**

**Nota:**

**Nota2:**

**Nota3:**

Figura 16: Pagina

## lista\_notas.html

```

1  {% load static %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5    <title>Lista de Notas</title>
6    <link rel="stylesheet" type="text/css" href="{% static 'css/style_table.css'%}"></link>
7  </head>
8  <body>
9    <div class="content">
10     <h1>Lista de Notas</h1>
11     <table border="1">
12       <thead>
13         <tr>
14           <th> Alumno </th>
15           <th> Curso </th>
16           <th> Nota 1 </th>
17           <th> Nota 2 </th>
18           <th> Nota 3 </th>
19         </tr>
20       </thead>
21       <tbody>
22         {% for nota in notas %}
23         <tr>
24           <td>{{ nota.alumno.nombre }}</td>
25           <td>{{ nota.curso.curso }}</td>
26           <td>{{ nota.nota }}</td>
27           <td>{{ nota.nota2 }}</td>
28           <td>{{ nota.nota3 }}</td>
29         </tr>
30         {% endfor %}
31       </tbody>
32     </table>
33     <a href="{% url 'crear_nota' %}">Crear nueva nota</a>
34   </div>
35 </body>
36 </html>

```

Figura 17: Código de listaNota.html

## Lista de Notas

Alumno	Curso	Nota 1	Nota 2	Nota 3
Steven	EDA	20.00	17.00	17.00
Maria	Fisica	18.00	17.00	18.00

Crear nueva nota

Figura 18: Pagina

## Ejecutar el servidor

El comando `python manage.py makemigrations colegio` se utiliza en Django para crear nuevas migraciones basadas en los cambios realizados en los modelos del aplicativo colegio. Las migraciones son archivos que describen cómo modificar la estructura de la base de datos para mantenerla sincronizada con los modelos de Django. Este comando analiza los modelos en la aplicación colegio y genera los archivos de migración necesarios para aplicar esos cambios a la base de datos.

```
python manage.py makemigrations colegio
```

El comando `python manage.py runserver` se utiliza en Django para iniciar el servidor de desarrollo local. Una vez ejecutado, el servidor se activa y permite acceder a la aplicación web en desarrollo a través de un navegador en la dirección `http://localhost:8000/` por defecto. Es una herramienta fundamental durante el proceso de desarrollo de aplicaciones web con Django, ya que proporciona un entorno de prueba para probar y depurar el código antes de desplegar la aplicación en un entorno de producción.

```
python manage.py runserver
```

## Acceder a la aplicación en el navegador

Crear un alumno en `http://127.0.0.1:8000/notas/crearAlumno/`

Crear un curso en `http://127.0.0.1:8000/notas/crearCurso/`

Crear una nota en `http://127.0.0.1:8000/notas/crearNota/`

**URL de video de explicación:**

**URL de repositorio de GitHub:** <https://github.com/VictorMA18/Lab06-Django>