**Lab03: Modelos en Django.**

**Capacidades**

* Configurar tablas en Django.

**Seguridad**

* Generar un ambiente seguro.
* Evitar el consumo de alimentos.
* Dejar el ambiente ordenado y limpio.

**Preparación**

* El alumno debe revisar previamente el material cargado.

**Recursos**

* Computadora.

**Instrucciones**

* Usar el programa **python** y **pip** en windows y activar env con **./env/Scripts/activate**.
* Se debe distribuir el desarrollo del laboratorio entre los integrantes.
* Subir el proyecto creado en un solo repositorio de GitHub.
* Poner evidencia de su proyecto finalizando en la sección de desarrollo con el siguiente formato.
  + Nombre del alumno
  + Módulo desarrollado
  + Captura del Frontend
  + Captura de la CLI de ejecución del proyecto
* Todo debe ser desarrollado en inglés.
* Captura de la estructura de tu proyecto en VSC.

# Guía: Aplicación Simple de Cuestionarios con Django

Esta guía te mostrará cómo crear una aplicación básica de cuestionarios con Django, con funcionalidades para crear exámenes, agregar preguntas y especificar respuestas correctas.

# Estructura de la Aplicación

Crearemos 3 rutas principales:

* Lista de exámenes
* Detalle de un examen con sus preguntas
* Formulario para crear exámenes y preguntas

# Paso 1: Configur ación del Entorno

*# Crear directorio del proyecto*

mkdir quiz\_app

cd quiz\_app

*# Crear entorno virtual*

python3 - m venv venv

*# Activar entorno virtual*

source venv/bin/activate

*# Crear directorio src (estructura similar a proyectos frontend)*

mkdir src

cd src

*# Instalar Django*

pip3 install django

*# Crear un archivo requirements.txt*

pip3 freeze > requirements.txt

**Nota**: Si estás usando Git, crea un archivo .gitignore en la raíz del proyecto:

*# Volver a la raíz del proyecto*

cd ..

*# Crear archivo .gitignore*

echo "\_\_pycache\_\_/

\*.py[cod]

\*$py.class

\*.so

.Python

db.sqlite3

.env

venv/

.vscode/" > .gitignore

Esto evitará versionar archivos innecesarios como caché de Python, base de datos SQLite y el entorno virtual.

## Paso 2: Crear el Proyecto Django

*# Asegúrate de estar en el directorio src*

cd src

*# Crear proyecto*

django-admin startproject config .

*# Crear aplicación*

python3 manage.py startapp quiz

## Paso 3: Configuración Básica

Editar config/settings.py para incluir la app:

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'quiz', *# Our application*

]

## Paso 4: Definir los Modelos

Editar quiz/models.py:

from django.db import models

class Exam(models.Model):

"""Model for exams"""

title = models.CharField(max\_length=200, verbose\_name="Título")

description = models.TextField(blank=True, verbose\_name="Descripción")

created\_date = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.title

def get\_question\_count(self):

"""Returns the number of questions in this exam"""

return self.questions.count()

class Question(models.Model):

"""Model for questions"""

exam = models.ForeignKey(Exam, on\_delete=models.CASCADE, related\_name='questions')

text = models.TextField(verbose\_name="Texto de la pregunta")

def \_\_str\_\_(self):

return self.text[:50]

class Choice(models.Model):

"""Model for answer choices"""

question = models.ForeignKey(Question, on\_delete=models.CASCADE, related\_name='choices')

text = models.CharField(max\_length=200, verbose\_name="Texto")

is\_correct = models.BooleanField(default=False, verbose\_name="Es correcta")

def \_\_str\_\_(self):

return self.text

## Paso 5: Migrar la Base de Datos

python3 manage.py makemigrations

python3 manage.py migrate

**Nota**: Recuerda que los archivos de migración (en migrations/) deben versionarse, pero el archivo db.sqlite3 debe excluirse del control de versiones (ya está en el .gitignore que creamos).

## Paso 6: Crear Formularios

Crear archivo quiz/forms.py:

from django import forms

from .models import Exam, Question, Choice

class ExamForm(forms.ModelForm):

class Meta:

model = Exam

fields = ['title', 'description']

widgets = {

'title': forms.TextInput(attrs={'class': 'form-control'}),

'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),

}

class QuestionForm(forms.ModelForm):

class Meta:

model = Question

fields = ['text']

widgets = {

'text': forms.Textarea(attrs={'class': 'form-control', 'rows': 2}),

}

class ChoiceForm(forms.ModelForm):

class Meta:

model = Choice

fields = ['text', 'is\_correct']

widgets = {

'text': forms.TextInput(attrs={'class': 'form-control'}),

'is\_correct': forms.CheckboxInput(attrs={'class': 'form-check-input'}),

}

*# Crear formset para opciones*

ChoiceFormSet = forms.inlineformset\_factory(

Question, Choice, form=ChoiceForm, extra=4, can\_delete=False

)

## Paso 7: Crear Vistas

Editar quiz/views.py:

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.contrib import messages

from django.db import transaction

from .models import Exam, Question, Choice

from .forms import ExamForm, QuestionForm, ChoiceFormSet

def exam\_list(request):

"""View to display a list of all exams"""

exams = Exam.objects.all().order\_by('-created\_date')

return render(request, 'quiz/exam\_list.html', {'exams': exams})

def exam\_detail(request, exam\_id):

"""View to display the details of an exam with its questions"""

exam = get\_object\_or\_404(Exam, id=exam\_id)

questions = exam.questions.all().prefetch\_related('choices')

return render(request, 'quiz/exam\_detail.html', {'exam': exam, 'questions': questions})

def exam\_create(request):

"""View to create a new exam"""

if request.method == 'POST':

form = ExamForm(request.POST)

if form.is\_valid():

exam = form.save()

messages.success(request, 'Examen creado correctamente.')

return redirect('question\_create', exam\_id=exam.id)

else:

form = ExamForm()

return render(request, 'quiz/exam\_form.html', {'form': form})

def question\_create(request, exam\_id):

"""View to add questions to an exam"""

exam = get\_object\_or\_404(Exam, id=exam\_id)

if request.method == 'POST':

question\_form = QuestionForm(request.POST)

if question\_form.is\_valid():

with transaction.atomic():

*# Save the question*

question = question\_form.save(commit=False)

question.exam = exam

question.save()

*# Process the formset for choices*

formset = ChoiceFormSet(request.POST, instance=question)

if formset.is\_valid():

formset.save()

*# Verify that only one option is marked as correct*

correct\_count = question.choices.filter(is\_correct=True).count()

if correct\_count != 1:

messages.warning(request, 'Debe haber exactamente una respuesta correcta.')

else:

messages.success(request, 'Pregunta añadida correctamente.')

*# Decide where to redirect*

if 'add\_another' in request.POST:

return redirect('question\_create', exam\_id=exam.id)

else:

return redirect('exam\_detail', exam\_id=exam.id)

else:

question\_form = QuestionForm()

formset = ChoiceFormSet()

return render(request, 'quiz/question\_form.html', {

'exam': exam,

'question\_form': question\_form,

'formset': formset,

})

## Paso 8: Configurar URLs

Crear quiz/urls.py:

from django.urls import path

from . import views

urlpatterns = [

path('', views.exam\_list, name='exam\_list'),

path('exam/<int:exam\_id>/', views.exam\_detail, name='exam\_detail'),

path('exam/create/', views.exam\_create, name='exam\_create'),

path('exam/<int:exam\_id>/question/add/', views.question\_create, name='question\_create'),

]

Editar config/urls.py:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('quiz.urls')),

]

## Paso 9: Crear Plantillas

Crear estructura de directorios:

mkdir -p quiz/templates/quiz

### Plantilla Base (quiz/templates/base.html)

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>{% block title %}Aplicación de Cuestionarios{% endblock %}</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light mb-4">

<div class="container">

<a class="navbar-brand" href="{% url 'exam\_list' %}">Cuestionarios</a>

<div class="collapse navbar-collapse">

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="{% url 'exam\_list' %}">Exámenes</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{% url 'exam\_create' %}">Crear Examen</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container">

{% if messages %}

{% for message in messages %}

<div class="alert alert-{{ message.tags }} alert-dismissible fade show">

{{ message }}

<button type="button" class="btn-close" data-bs-dismiss="alert"></button>

</div>

{% endfor %}

{% endif %}

{% block content %}{% endblock %}

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

{% block extra\_js %}{% endblock %}

</body>

</html>

### Lista de Exámenes (quiz/templates/quiz/exam\_list.html)

{% extends "base.html" %}

{% block title %}Exámenes{% endblock %}

{% block content %}

<div class="row mb-4">

<div class="col-md-6">

<h1>Exámenes Disponibles</h1>

</div>

<div class="col-md-6 text-end">

<a href="{% url 'exam\_create' %}" class="btn btn-primary">Crear Examen</a>

</div>

</div>

<div class="row">

{% if exams %}

{% for exam in exams %}

<div class="col-md-6 mb-4">

<div class="card">

<div class="card-body">

<h5 class="card-title">{{ exam.title }}</h5>

<p class="card-text">{{ exam.description|truncatechars:100 }}</p>

<p class="text-muted">Preguntas: {{ exam.get\_question\_count }}</p>

<a href="{% url 'exam\_detail' exam.id %}" class="btn btn-primary">Ver Detalles</a>

</div>

</div>

</div>

{% endfor %}

{% else %}

<div class="col-12">

<div class="alert alert-info">

No hay exámenes disponibles.

<a href="{% url 'exam\_create' %}">Crea el primero</a>

</div>

</div>

{% endif %}

</div>

{% endblock %}

### Detalle de Examen (quiz/templates/quiz/exam\_detail.html)

{% extends "base.html" %}

{% block title %}{{ exam.title }}{% endblock %}

{% block content %}

<div class="row mb-4">

<div class="col-md-8">

<h1>{{ exam.title }}</h1>

<p>{{ exam.description }}</p>

</div>

<div class="col-md-4 text-end">

<a href="{% url 'question\_create' exam.id %}" class="btn btn-primary">Añadir Pregunta</a>

</div>

</div>

<div class="mb-4">

<h2>Preguntas</h2>

{% if questions %}

<div class="accordion" id="accordionQuestions">

{% for question in questions %}

<div class="accordion-item">

<h2 class="accordion-header">

<button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#collapse{{ question.id }}">

{{ forloop.counter }}. {{ question.text }}

</button>

</h2>

<div id="collapse{{ question.id }}" class="accordion-collapse collapse" data-bs-parent="#accordionQuestions">

<div class="accordion-body">

<ul class="list-group">

{% for choice in question.choices.all %}

<li class="list-group-item {% if choice.is\_correct %}list-group-item-success{% endif %}">

{{ choice.text }} {% if choice.is\_correct %}<span class="badge bg-success">Correcta</span>{% endif %}

</li>

{% empty %}

<li class="list-group-item">No hay opciones para esta pregunta.</li>

{% endfor %}

</ul>

</div>

</div>

</div>

{% endfor %}

</div>

{% else %}

<div class="alert alert-info">

Este examen no tiene preguntas todavía.

<a href="{% url 'question\_create' exam.id %}">Añade la primera pregunta</a>

</div>

{% endif %}

</div>

<div>

<a href="{% url 'exam\_list' %}" class="btn btn-secondary">Volver a la lista</a>

</div>

{% endblock %}

### Formulario de Examen (quiz/templates/quiz/exam\_form.html)

{% extends "base.html" %}

{% block title %}Crear Examen{% endblock %}

{% block content %}

<div class="row">

<div class="col-md-8 offset-md-2">

<div class="card">

<div class="card-header">

<h2>Crear Nuevo Examen</h2>

</div>

<div class="card-body">

<form method="post">

{% csrf\_token %}

<div class="mb-3">

<label for="{{ form.title.id\_for\_label }}" class="form-label">Título</label>

{{ form.title }}

{% if form.title.errors %}

<div class="text-danger">{{ form.title.errors }}</div>

{% endif %}

</div>

<div class="mb-3">

<label for="{{ form.description.id\_for\_label }}" class="form-label">Descripción</label>

{{ form.description }}

{% if form.description.errors %}

<div class="text-danger">{{ form.description.errors }}</div>

{% endif %}

</div>

<div class="d-flex justify-content-between">

<a href="{% url 'exam\_list' %}" class="btn btn-secondary">Cancelar</a>

<button type="submit" class="btn btn-primary">Continuar</button>

</div>

</form>

</div>

</div>

</div>

</div>

{% endblock %}

### Formulario de Pregunta (quiz/templates/quiz/question\_form.html)

{% extends "base.html" %}

{% block title %}Añadir Pregunta{% endblock %}

{% block content %}

<div class="row">

<div class="col-md-10 offset-md-1">

<div class="card">

<div class="card-header">

<h2>Añadir Pregunta a: {{ exam.title }}</h2>

</div>

<div class="card-body">

<form method="post">

{% csrf\_token %}

<div class="mb-3">

<label for="{{ question\_form.text.id\_for\_label }}" class="form-label">Texto de la pregunta</label>

{{ question\_form.text }}

{% if question\_form.text.errors %}

<div class="text-danger">{{ question\_form.text.errors }}</div>

{% endif %}

</div>

<h4 class="mt-4 mb-3">Opciones de respuesta</h4>

<div class="alert alert-info">

Marca la casilla "Es correcta" en la opción que sea la respuesta correcta.

</div>

{{ formset.management\_form }}

<div id="options-container">

{% for choice\_form in formset %}

<div class="card mb-3">

<div class="card-body">

<div class="row">

<div class="col-md-9">

<label class="form-label">Opción {{ forloop.counter }}</label>

{{ choice\_form.text }}

</div>

<div class="col-md-3">

<div class="form-check mt-2">

{{ choice\_form.is\_correct }}

<label class="form-check-label" for="{{ choice\_form.is\_correct.id\_for\_label }}">

Es correcta

</label>

</div>

</div>

</div>

{{ choice\_form.id }}

</div>

</div>

{% endfor %}

</div>

<div class="d-flex justify-content-between mt-4">

<a href="{% url 'exam\_detail' exam.id %}" class="btn btn-secondary">Cancelar</a>

<div>

<button type="submit" name="add\_another" class="btn btn-info">Guardar y añadir otra</button>

<button type="submit" class="btn btn-primary">Guardar y terminar</button>

</div>

</div>

</form>

</div>

</div>

</div>

</div>

{% endblock %}

{% block extra\_js %}

<script>

*// Script para asegurar que solo una opción sea marcada como correcta*

document.addEventListener('DOMContentLoaded', function() {

const checkboxes = document.querySelectorAll('input[type=checkbox]');

checkboxes.forEach(function(checkbox) {

checkbox.addEventListener('change', function() {

if (this.checked) {

*// Desmarcar las demás opciones*

checkboxes.forEach(function(otherCheckbox) {

if (otherCheckbox !== checkbox) {

otherCheckbox.checked = false;

}

});

}

});

});

});

</script>

{% endblock %}

## Paso 10: Ejecutar el Servidor

python3 manage.py runserver

Ahora puedes acceder a:

* Lista de exámenes: <http://127.0.0.1:8000/>
* Crear examen: <http://127.0.0.1:8000/exam/create/>

## Paso 11: Registrar los Modelos en el Admin (opcional)

Editar quiz/admin.py:

from django.contrib import admin

from .models import Exam, Question, Choice

class ChoiceInline(admin.TabularInline):

"""Inline admin for choices"""

model = Choice

extra = 4

class QuestionInline(admin.StackedInline):

"""Inline admin for questions"""

model = Question

extra = 1

@admin.register(Exam)

class ExamAdmin(admin.ModelAdmin):

"""Admin configuration for exams"""

list\_display = ('title', 'created\_date')

inlines = [QuestionInline]

@admin.register(Question)

class QuestionAdmin(admin.ModelAdmin):

"""Admin configuration for questions"""

list\_display = ('text', 'exam')

inlines = [ChoiceInline]

## Flujo de Trabajo de la Aplicación

1. **Crear un examen**:
   * Accede a la página principal y haz clic en "Crear Examen"
   * Completa el título y descripción
   * Haz clic en "Continuar"
2. **Añadir preguntas al examen**:
   * Después de crear el examen, serás redirigido al formulario de preguntas
   * Escribe el texto de la pregunta
   * Agrega las opciones de respuesta
   * Marca la casilla "Es correcta" en la opción que sea la respuesta correcta
   * Haz clic en "Guardar y añadir otra" para añadir más preguntas o "Guardar y terminar" para finalizar
3. **Ver la lista de exámenes**:
   * Accede a la página principal para ver todos los exámenes
   * Haz clic en "Ver Detalles" para explorar un examen específico
4. **Ver detalle de un examen**:
   * Revisa todas las preguntas del examen
   * Expande cada pregunta para ver sus opciones
   * Las opciones correctas se destacan en verde

## Conclusiones

En esta guía, has aprendido a:

1. **Crear una aplicación Django básica** con modelos para exámenes, preguntas y opciones.
2. **Implementar relaciones entre modelos** usando ForeignKey para establecer conexiones entre exámenes, preguntas y opciones.
3. **Crear formularios** para la entrada de datos, incluyendo formsets para manejar múltiples opciones de respuesta.
4. **Desarrollar vistas** para las tres funcionalidades principales:
   * Listado de exámenes
   * Detalle de un examen
   * Creación de exámenes y preguntas
5. **Diseñar plantillas** utilizando Bootstrap para una interfaz sencilla pero funcional.

Esta aplicación demuestra cómo Django facilita la creación de aplicaciones web con modelos de datos relacionados. Los conceptos aprendidos aquí pueden aplicarse a proyectos más complejos, añadiendo más funcionalidades según sea necesario.

## Siguientes Pasos

Si deseas ampliar esta aplicación, aquí hay algunos retos interesantes:

1. **Mejorar la gestión de exámenes**:
   * Implementar la edición de exámenes y preguntas
   * Añadir funcionalidad para eliminar preguntas
   * Permitir reordenar las preguntas mediante drag-and-drop
2. **Crear un sistema de juego**:
   * Desarrollar una vista para "jugar" el examen
   * Implementar un temporizador para limitar el tiempo de respuesta
   * Calcular y mostrar puntuaciones basadas en respuestas correctas
   * Generar un resumen de resultados al finalizar
3. **Añadir características sociales**:
   * Implementar un sistema de usuarios
   * Permitir compartir exámenes con otros usuarios
   * Crear rankings de puntuaciones

## Propuestas de Aplicaciones Adicionales

Para seguir practicando con Django, te propongo desarrollar estas aplicaciones complementarias:

### 1. Aplicación de Categorías (categories)

Crea una aplicación separada para gestionar categorías de exámenes:

python3 manage.py startapp categories

Esta aplicación podría:

* Definir un modelo Category con campos como nombre, descripción e icono
* Relacionar categorías con exámenes (relación muchos a muchos)
* Permitir filtrar exámenes por categoría
* Mostrar estadísticas de exámenes por categoría

### 2. Aplicación de Estadísticas (stats)

Desarrolla una aplicación para analizar el rendimiento:

python3 manage.py startapp stats

Esta aplicación podría:

* Registrar los intentos de los usuarios en los exámenes
* Calcular estadísticas como porcentaje de aciertos, tiempo promedio, etc.
* Generar gráficos de rendimiento
* Identificar preguntas con mayor índice de fallos

Implementar estas aplicaciones te permitirá practicar conceptos más avanzados de Django como relaciones entre modelos, queries complejas y gestión de datos entre múltiples aplicaciones.

**Desarrollo**

**Reto:**

* Desarrollar con CSS puro.
* Con apoyo de la AI realizar endpoints y consumirlos desde un frontend desarrollado con HTML, CSS y JS vanilla o React.

**Incrementos**

* **Repositorio (0-5)**
* **Código fuente (0-5)**
* **Informe (-10-0)**
* **Exposición (0-10)**

**Repositorio:**

MENDOZA YAMPI, ITALO ANDRE: <https://github.com/syderkkk/quiz_app>

RIVERA MUNAREZ, JOSEP DANTON: <https://github.com/JosepRivera/dae-laboratorio-3>

**Proyecto funcionando:**  
Patrón de fondo

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**Caso de prueba Creación de un examen (Italo Mendoza):**

Formulario:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Agregación de preguntas y sus opciones:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Confirmación:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Pagina principal y base de datos actualizada con el examen creado:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

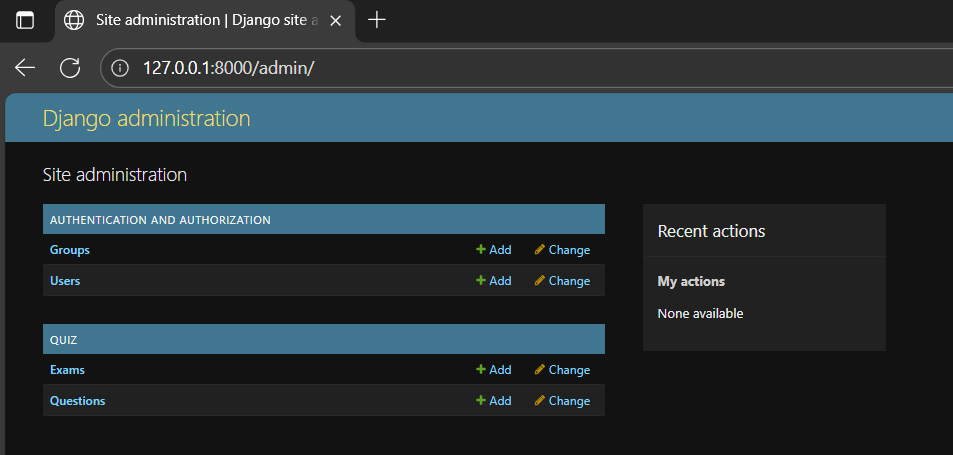
**Captura de la estructura del proyecto:**

Texto

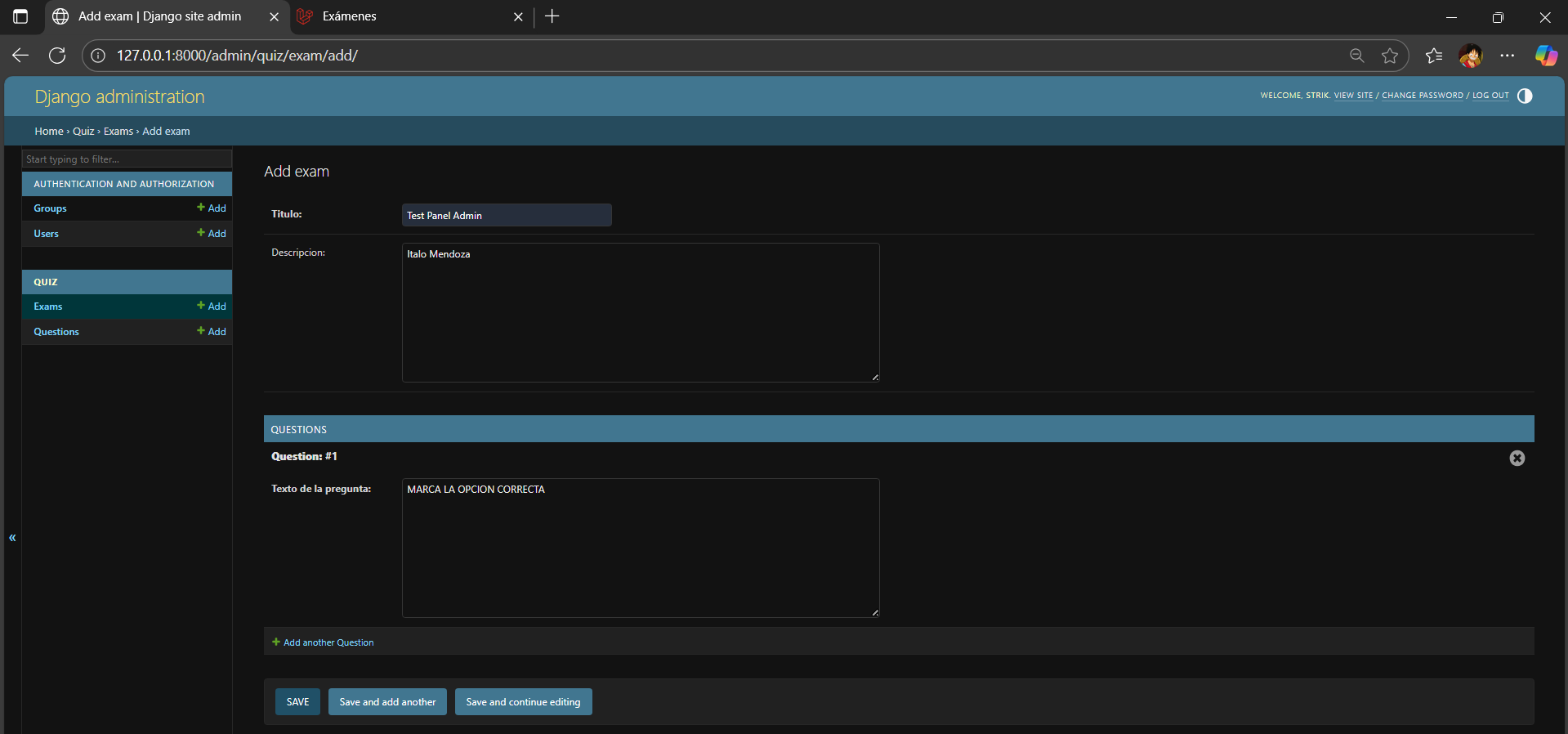
Descripción generada automáticamente

**Creación de un examen desde el Admin Panel:**

Panel del administrador:

****

Formulario para la creación del examen:

****

Mensaje de confirmación de la creación del exámen:

**Interfaz de usuario gráfica, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

Página principal y base de datos actualizada después del examen creado desde el panel de administrador:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

**Enlace video:**

**MENDOZA YAMPI, ITALO ANDRE:** [**https://youtu.be/6CtZy1f1kVw**](https://youtu.be/6CtZy1f1kVw)

**RIVERA MUNAREZ, JOSEP DANTON:** <https://drive.google.com/file/d/1pe4GLRzFIPE_lg7x3K5OjLqZS9ptbrEm/view?usp=drive_link>

**Conclusiones**

* Se logró crear una aplicación básica en Django que permite manejar exámenes, preguntas y opciones de respuesta.
* Se aplicó la relación entre modelos, lo que permitió que exámenes, preguntas y respuestas estuvieran conectados de manera correcta.
* Se usaron formularios y formsets para ingresar datos, lo que facilitó la creación de preguntas con sus opciones.