

Informe de Laboratorio 08

Tema: Django: Relaciones de uno a muchos, muchos a muchos y impresión de pdf y email

Nota

Estudiante	Escuela	Asignatura
Fernando Miguel Garambel Marín fgarambel@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Laboratorio de Programación Web 2 Semestre: III Código: 1701212

Laboratorio	Tema	Duración
08	Django: Relaciones de uno a muchos, muchos a muchos y impresión de pdf y emails	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 24 de mayo 2024	Al 14 de junio 2024

1. Objetivos

- Implementar en una aplicación en Django el manejo de Bases de datos.
- Utilizar una tabla y relacionarla con muchas tablas.
- Utilizar muchas tablas y relacionarlas con muchas tablas.
- Implementar el envío de emails y la impresión de pdfs desde una aplicación Django

2. Actividades

- Crear un proyecto en Django
- Siga los pasos de los videos para poder implementar la aplicación de Relación de Uno a muchos en una Base de Datos, muchos a muchos, impresión de pdfs y envío de emails.
- Use git y haga los commits necesarios para manejar correctamente la aplicación.

3. Ejercicio Propuestos

- Deberán replicar la actividad de los videos donde se trabaja con Relacion de uno a muchos, de muchos a muchos, impresión de pdfs y envío de emails; adecuándolo desde un proyecto en blanco Django.
- Para ello crear una carpeta dentro del proyecto github colaborativo con el docente, e informar el link donde se encuentra.
- Eres libre de agregar CSS para decorar tu trabajo.
- Ya sabes que el trabajo con Git es obligatorio. Revisa los videos entregados.

4. Equipos, materiales y temas utilizados

- Sistema operativo de 64 bits, procesador basado en x64.
- Latex.
- git version 2.41.0.windows.1
- Cuenta en GitHub con el correo institucional.

5. URL Github, Video

- URL del Repositorio GitHub para clonar o recuperar.
- https://github.com/FernandoGarambelM/Lab08_Django_relaciones.git
- URL para el video flipgrid
- <https://flip.com/s/pvZBG1NhvCRn>

6. Replicando la actividad del video

- Primero ingresamos al entorno virtual

Listing 1: Activar el ambiente virtual

```
test\Scripts\activate
```

- Creamos los modelos que vamos a usar

Listing 2: Código de models.py

```
from django.db import models

# Create your models here.
class Language(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Framework(models.Model):
    name = models.CharField(max_length=10)
    language = models.ForeignKey(Language, on_delete=models.CASCADE)

    def __str__(self):
        return self.name

class Movie(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Character(models.Model):
    name = models.CharField(max_length=10)
    movies = models.ManyToManyField(Movie)

    def __str__(self):
        return self.name
```

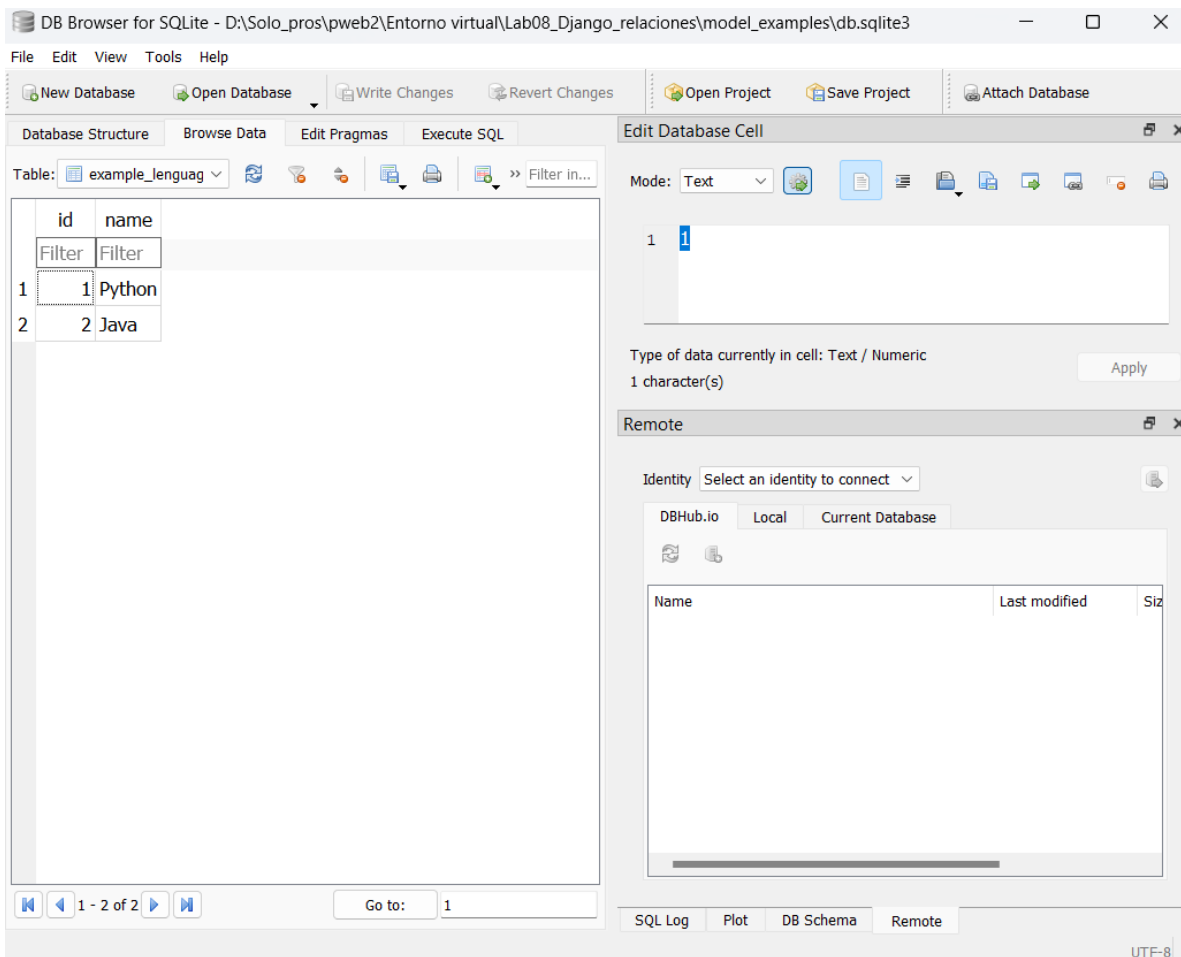
- Es importante aclarar las relaciones que se crearon en estos modelos:
- One-to-Many (Uno a muchos)
- Entre Framework y Language hay una relación uno a muchos. Cada framework está asociado con un único lenguaje, pero un lenguaje puede tener múltiples frameworks. Esto se logra mediante la clave foránea (ForeignKey) en el modelo Framework.
- Many-to-Many (Muchos a muchos)
- Entre Character y Movie hay una relación muchos a muchos. Un personaje puede aparecer en múltiples películas y una película puede tener múltiples personajes. Esto se logra mediante el campo ManyToManyField en el modelo Character.

- En resumen: Language tiene una lista de Frameworks asociados, Framework pertenece a un solo Language, Character puede estar en varias Movies y viceversa, Movie puede tener varios Characters y viceversa, Estas relaciones permiten modelar de manera eficiente la estructura y las interconexiones de los datos.

Listing 3: Ingresamos a shell de python

```
python manage.py shell
```

- Ahi ingresaremos los datos para las tablas en base de datos
- Ahora veremos las bases de datos



The screenshot shows the DB Browser for SQLite application. The main window displays the 'example_lenguag' table with the following data:

id	name
1	Python
2	Java

The interface also includes a menu bar (File, Edit, View, Tools, Help), a toolbar with various database operations, and a right-hand panel for editing database cells. The 'Remote' section shows connection options like DBHub.io, Local, and Current Database.

DB Browser for SQLite - D:\Solo_pros\pweb2\Entorno virtual\Lab08_Django_relaciones\model_examples\db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: example_framework

	id	name	language_id
	Filter	Filter	Filter
1	1	Django	1
2	2	Flask	1
3	3	Bottle	1
4	4	Spring	2

1 - 4 of 4

Go to: 1

Mode: Text

1

Type of data currently in cell: Text / Numeric
1 character(s)

Apply

Remote

Identity Select an identity to connect

DBHub.io Local Current Database

Name	Last modified	Size
------	---------------	------

SQL Log Plot DB Schema Remote

UTF-8

DB Browser for SQLite - D:\Solo_pros\pweb2\Entorno virtual\Lab08_Django_relaciones\model_examples\db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: example_movie

	id	name
Filter	Filter	
1	1	Avengers
2	2	Civil War
3	3	Thor: Dark World
4	4	Winter Soldier

1 - 4 of 4

Go to: 1

Mode: Text

1

Type of data currently in cell: Text / Numeric
1 character(s)

Apply

Remote

Identity Select an identity to connect

DBHub.io Local Current Database

Name	Last modified	Size
------	---------------	------

SQL Log Plot DB Schema Remote

UTF-8

DB Browser for SQLite - D:\Solo_pros\pweb2\Entorno virtual\Lab08_Django_relaciones\model_examples\db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: example_characters

id	name
1	Captain America
2	Thor

1 2

Go to: 1

Mode: Text

1

Type of data currently in cell: Text / Numeric
1 character(s)

Apply

Remote

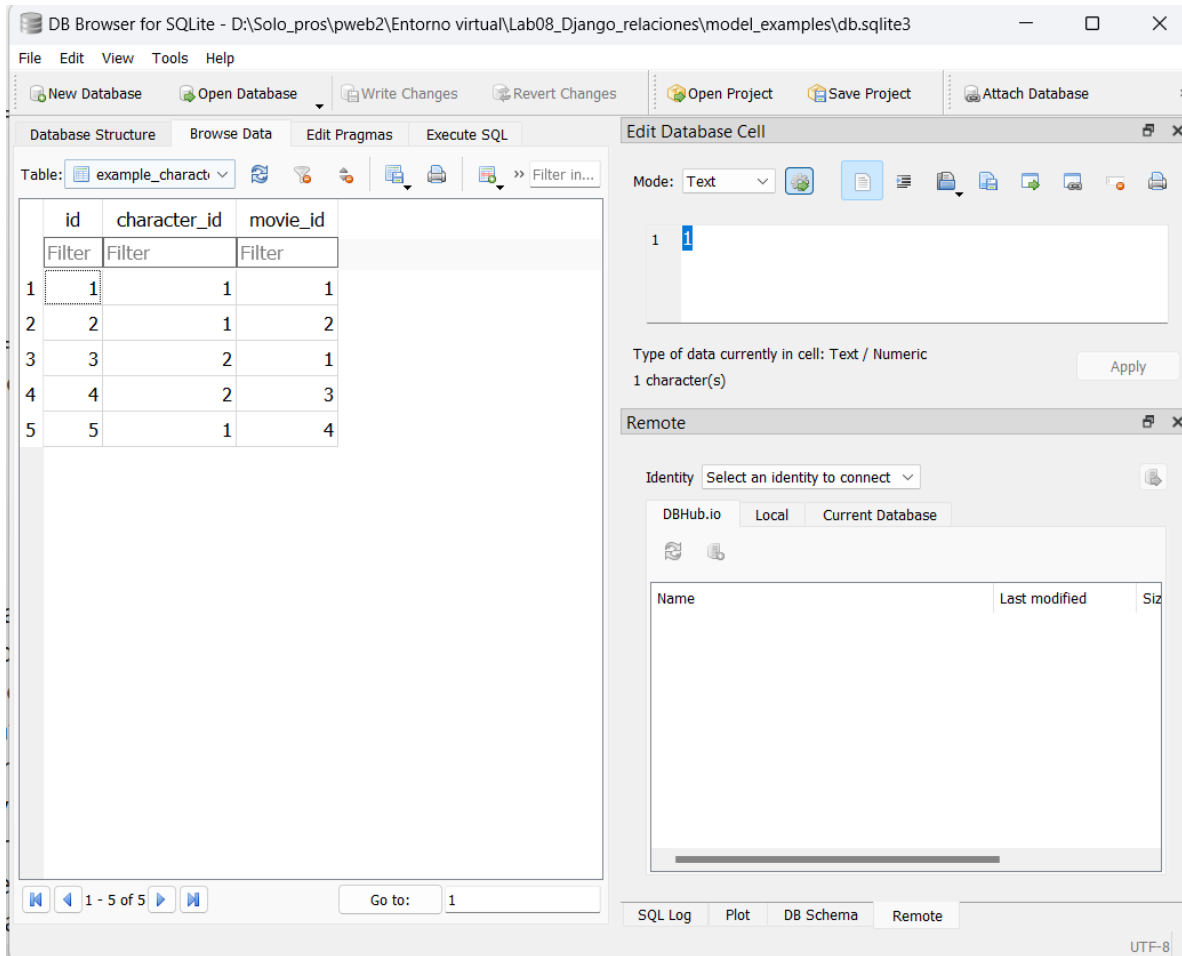
Identity Select an identity to connect

DBHub.io Local Current Database

Name	Last modified	Size
------	---------------	------

SQL Log Plot DB Schema Remote

UTF-8



Listing 4: Script para crear un superusuario

```
#!/bin/sh

# Variables de entorno para el superusuario
DJANGO_SUPERUSER_USERNAME=admin
DJANGO_SUPERUSER_EMAIL=admin@example.com
DJANGO_SUPERUSER_PASSWORD=123456

# Ejecutar el comando createsuperuser sin interaccion
python manage.py shell -c "from django.contrib.auth import get_user_model; User =
    get_user_model(); User.objects.create_superuser('$DJANGO_SUPERUSER_USERNAME',
    '$DJANGO_SUPERUSER_EMAIL', '$DJANGO_SUPERUSER_PASSWORD') if not
    User.objects.filter(username='$DJANGO_SUPERUSER_USERNAME').exists() else
    print('Superusuario ya existe.')"

```

- Realizamos migraciones para enlazar la base de datos

Listing 5: Código para realizar migraciones


```
python manage.py makemigrations
python manage.py migrations
```

- Después de seguir estos pasos la plantilla ya funciona por completo

7. Adecuar la plantilla a un proyecto en blanco de Django

- Creamos el modelo destination con los datos requeridos

Listing 6: Código de models.py

```
1 from django.db import models
2
3 # Create your models here.
4
5 class Destination(models.Model):
6
7     name = models.CharField(max_length=100)
8     img = models.ImageField(upload_to='pics')
9     desc = models.TextField()
10    price = models.IntegerField()
11    offer = models.BooleanField(default=False)
```

8. Crear formularios de Añadir Destinos Turísticos, Modificar, Listar y Eliminar Destinos.

- Luego creamos los forms

Listing 7: Código de forms.py

```
1 from django import forms
2 from .models import Destination
3
4 class DestinosTuristicosForm(forms.ModelForm):
5     class Meta:
6         model = Destination
7         fields = ['name', 'img', 'desc', 'price', 'offer']
```

- Luego creamos views.py

Listing 8: Código de views.py

```
1 from django.shortcuts import render, redirect, get_object_or_404
2 from .models import Destination
3 from .forms import DestinosTuristicosForm
4 # Create your views here.
5
6 def index(request):
7
```

```
8     dests = Destination.objects.all()
9
10    return render(request, "index.html", {'dests': dests})
11
12    def gestionar_destinos(request, id=None):
13        if request.method == 'POST':
14            if id:
15                destino = get_object_or_404(Destination, id=id)
16                form = DestinosTuristicosForm(request.POST, request.FILES, instance=destino)
17                if 'modificar' in request.POST:
18                    if form.is_valid():
19                        form.save()
20                elif 'eliminar' in request.POST:
21                    destino.delete()
22            else:
23                form = DestinosTuristicosForm(request.POST, request.FILES)
24                if form.is_valid():
25                    form.save()
26                return redirect('gestionar_destinos')
27
28        else:
29            if id:
30                destino = get_object_or_404(Destination, id=id)
31                form = DestinosTuristicosForm(instance=destino)
32            else:
33                form = DestinosTuristicosForm()
34
35        destinos = Destination.objects.all()
36        return render(request, 'gestionar_destinos.html', {
37            'form': form,
38            'destinos': destinos,
39            'id': id,
40        })
```

- Luego agregamos urls.py para que Django pueda leerlos

Listing 9: Código de urls.py

```
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("", views.index, name="index"),
7     path('gestionar/', views.gestionar_destinos, name="gestionar_destinos"),
8     path('gestionar/<int:id>/', views.gestionar_destinos, name="modificar_destino"),
9 ]
```

- Usamos la plantilla de travello, específicamente la siguiente parte para listar los destinos turísticos con if y for

Listing 10: Pedazo de html donde se usa for e if

```
{% for dest in dests %}
```

```

<!-- Destination -->
<div class="destination_item">
  <div class="destination_image">
    

    {% if dest.offer %}
    <div class="spec_offer text-center"><a href="#">Special
      Offer</a></div>
    {% endif %}
  </div>
  <div class="destination_content">
    <div class="destination_title"><a
      href="destinations.html">{{dest.name}}</a>
    </div>
    <div class="destination_subtitle">
      <p>{{dest.desc}}</p>
    </div>
    <div class="destination_price">From ${{dest.price}}</div>
  </div>
</div>

{% endfor %}

```

- Se crea en templates gestionar_destinos.html

Listing 11: Html de gestionar_destinos.html

```

{% load static %}
<!DOCTYPE html>
<html>
<head>
  <title>Gestionar Destinos</title>
  <link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}">
</head>
<body>
  <h1>Gestionar Destinos Turisticos</h1>

  <!-- Formulario de Anadir/Modificar -->
  <form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" name="modificar" {% if not id %}disabled{% endif %}>
      Modificar</button>
    <button type="submit" name="eliminar" {% if not id %}disabled{% endif %}>
      Eliminar</button>
    <button type="submit" name="anadir" {% if id %}disabled{% endif %}>Anadir</button>
  </form>

  <h2>Lista de Destinos</h2>
  <ul>
    {% for destino in destinos %}
      <li>
        <h2>{{ destino.name }}</h2>
        <p>{{ destino.desc }}</p>
        <p>Precio: {{ destino.price }}</p>
      </li>
    {% endfor %}
  </ul>

```

```
<p>{% if destino.offer %}En oferta{% endif %}</p>
<a href="{% url 'modificar_destino' destino.id %}">Seleccionar</a>
</li>
{% endfor %}
</ul>

</body>
</html>
```

- Para el registro y login de usuarios hacemos los mismo pasos para crear la app y ahora mostramos solo views por que es lo mas importante

Listing 12: Código de views.py de accounts

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.models import User, auth

# Create your views here.
def login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = auth.authenticate(username=username, password=password)

        if user is not None:
            if user.is_superuser:
                auth.login(request, user)
                return redirect("/gestionar")
            else:
                auth.login(request, user)
                return redirect("/")
        else:
            messages.info(request, 'invalid credentials')
            return redirect('login')

    else:
        return render(request, 'login.html')

def register(request):

    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        username = request.POST['username']
        password1 = request.POST['password1']
        password2 = request.POST['password2']
        email = request.POST['email']

        if password1==password2:
            if User.objects.filter(username=username).exists():
                messages.info(request, 'Username Taken')
                return redirect('register')
            elif User.objects.filter(email=email).exists():
```

```
        messages.info(request, 'Email Taken')
        return redirect('register')
    else:
        user = User.objects.create_user(username=username, password=password1,
                                         email=email, first_name=first_name, last_name=last_name)
        user.save();
        print('user created')
        return redirect('login')

    else:
        messages.info(request, 'password not matching..')
        return redirect('register')
    return redirect('/')

else:
    return render(request, 'register.html')

def logout(request):
    auth.logout(request)
    return redirect('/')
```

- Aquí se procesan los datos obtenidos en login y register

Listing 13: Código de login.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset='utf-8'>
5     <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6     <title>Page Title</title>
7     <meta name='viewport' content='width=device-width, initial-scale=1'>
8     <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
9     <script src='main.js'></script>
10 </head>
11 <body>
12     <form action="login" method="post">
13         {% csrf_token %}
14         <input type="text" name="username" placeholder="username"><br>
15         <input type="password" name="password" placeholder="password"><br>
16         <input type="Submit">
17     </form>
18
19
20     <div>
21         {% for message in messages %}
22         <h3> {{message}} </h3>
23         {% endfor %}
24     </div>
25 </body>
26 </html>
```

Listing 14: Código de register.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6   <title>Registration</title>
7   <meta name='viewport' content='width=device-width, initial-scale=1'>
8   <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
9   <script src='main.js'></script>
10 </head>
11 <body>
12   <form action="register" method="post">
13     {% csrf_token %}
14
15     <input type="text" name="first_name" placeholder="First Name"><br>
16     <input type="text" name="last_name" placeholder="Last Name"><br>
17     <input type="text" name="username" placeholder="Username"><br>
18     <input type="email" name="email" placeholder="email"><br>
19     <input type="password" name="password1" placeholder="Password"><br>
20     <input type="password" name="password2" placeholder="Confirm Password"><br>
21     <input type="Submit">
22
23   </form>
24
25   <div>
26     {% for message in messages %}
27     <h3> {{message}} </h3>
28     {% endfor %}
29   </div>
30
31 </body>
32 </html>
```

9. Resultado

- Lista de los destinos turisticos

SIMPLY AMAZING PLACES

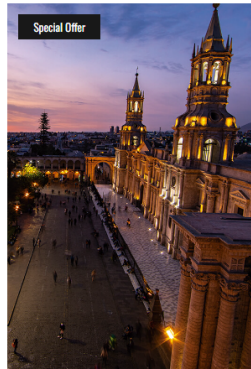
Popular Destinations



Cañon del Colca

Ubicado dentro del Valle del Colca, el Cañón es uno de los mayores destinos turísticos no solo de Arequipa, sino también del Perú. Su fácil acceso «un bus o auto particular desde la ciudad hacia el distrito de Chivay, no toma más de 3 horas y media». Aquí se desarrollan actividades variadas que la hacen tan requerida por los turistas interesados en los deportes de aventura, de la cultura ancestral y los amantes de la naturaleza.

From 570



Centro Historico de Arequipa

Situada a las faldas de un volcán, es conocida como la Ciudad Blanca. Sus calles son muestra del arte colonial y resguardan grandes atractivos.

From 55



Volcán Chachani

El Chachani es un volcán situado a 55 km al norte de la ciudad de Arequipa, en el sur del Perú. Pese a su considerable altura, la nieve es muy escasa en su cumbre, debido a la extrema sequedad atmosférica reinante en la vertiente occidental de los Andes de esa región.

From 5500

- Vista de los forms para gestionar los diferentes destinos turisticos

Gestionar Destinos Turísticos

Name:

Img:

Ningún archivo seleccionado

Desc:

Price:

Offer:

☐

Lista de Destinos

•

Cañón del Colca

Ubicado dentro del Valle del Colca, el Cañón es uno de los mayores destinos turísticos no solo de Arequipa, sino también del Perú. Su fácil acceso -un bus o auto particular desde la ciudad hacia el distrito de Chivay, no toma más de 3 horas y media-. Aquí se desarrollan actividades variadas que la hacen tan requerida por los turistas interesados en los deportes de aventura, de la cultura ancestral y los amantes de la naturaleza.

Precio: 70

[Seleccionar](#)

•

Centro Historico de Arequipa

Situada a las faldas de un volcán, es conocida como la Ciudad Blanca. Sus calles son muestra del arte colonial y resguardan grandes atractivos.

Precio: 5

En oferta

[Seleccionar](#)

•

Volcán Chachani

El Chachani es un volcán situado a 55 km al norte de la ciudad de Arequipa, en el sur del Perú. Pese a su considerable altura, la nieve es muy escasa en su cumbre, debido a la extrema sequedad atmosférica reinante en la vertiente occidental de los Andes de esa región.

Precio: 500

[Seleccionar](#)

10. Referencias

- <https://docs.djangoproject.com/es/3.2/>
- <https://docs.djangoproject.com/es/3.2/ref/models/fields/#field-types>