Sure, let's create a simple example for CRUD (Create, Read, Update, Delete) operations in Django. We'll create a basic application to manage a list of books.

First, make sure you have Django installed. You can install it via pip:

```
pip install django
```

Then, let's create a new Django project and an app within it:

```bash
django-admin startproject library_project
cd library_project
python manage.py startapp books
```

Now, let's define our Book model in `books/models.py`:

```python
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    publication_year = models.IntegerField()

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

Then, we need to create and apply migrations:

```bash
python manage.py makemigrations
python manage.py migrate
```

Now, let's create views for CRUD operations in `books/views.py`:

```python
from django.shortcuts import render, redirect
from .models import Book
from .forms import BookForm

def book_list(request):
    books = Book.objects.all()
    return render(request, 'books/book_list.html', {'books': books})

def add_book(request):
    if request.method == 'POST':
        form = BookForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm()
    return render(request, 'books/book_form.html', {'form': form})

def edit_book(request, pk):
    book = Book.objects.get(pk=pk)
    if request.method == 'POST':
```

```
        form = BookForm(request.POST, instance=book)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm(instance=book)
    return render(request, 'books/book_form.html', {'form': form})

def delete_book(request, pk):
    book = Book.objects.get(pk=pk)
    book.delete()
    return redirect('book_list')
```

Next, let's create forms for adding and editing books in `books/forms.py`:

```python
from django import forms
from .models import Book

class BookForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = ['title', 'author', 'publication_year']
```

Now, let's create templates for listing, adding, and editing books in `books/templates/books/` directory.

`book_list.html`:

```html
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Book List</title>
</head>
<body>
    <h1>Book List</h1>
    <ul>
        {% for book in books %}
            <li>{{ book.title }} by {{ book.author }} ({{ book.publication_year }}) -
                <a href="{% url 'edit_book' book.pk %}">Edit</a> |
                <a href="{% url 'delete_book' book.pk %}">Delete</a>
            </li>
        {% endfor %}
    </ul>
    <a href="{% url 'add_book' %}">Add Book</a>
</body>
</html>
```

`book_form.html`:

```html
<!DOCTYPE html>
<html>
<head>
    <title>{% if form.instance.pk %}Edit Book{% else %}Add Book{% endif %}</title>
</head>
<body>
    <h1>{% if form.instance.pk %}Edit Book{% else %}Add Book{% endif %}</h1>
    <form method="post">
        {% csrf_token %}
```

```
        {{ form.as_p }}
        <button type="submit">Save</button>
    </form>
</body>
</html>
```

Lastly, let's define URLs in `books/urls.py`:

```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.book_list, name='book_list'),
    path('add/', views.add_book, name='add_book'),
    path('edit/<int:pk>/', views.edit_book, name='edit_book'),
    path('delete/<int:pk>/', views.delete_book, name='delete_book'),
]
```

And include these URLs in the main `urls.py` of your project:

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('books/', include('books.urls')),
]
```

```

Now you have a basic CRUD application for managing books in Django. You can start the development server using `python manage.py runserver` and visit `http://127.0.0.1:8000/books/` in your browser to see the application in action.
```