# Семинарское задание: Создание нового приложения на Django

## Цель:

Изучить процесс создания нового приложения в Django, включая настройку моделей, представлений, маршрутов и форм, а также реализацию базового интерфейса с использованием HTML и CSS.

## Задание:

Создайте приложение, моделирующее систему управления библиотекой. Приложение должно поддерживать следующие функции:
- Управление книгами.
- Управление авторами.
- Управление категориями.
- Отображение списка книг и деталей каждой книги.

### Шаг 1: Создание проекта и приложения

1. Установите Django, если он еще не установлен:
 ```bash
 pip install django
 ```
2. Создайте новый проект Django:
 ```bash
 django-admin startproject library\_project
 cd library\_project
 ```
3. Создайте приложение для управления библиотекой:
 ```bash
 python manage.py startapp library
 ```
4. Зарегистрируйте приложение `library` в настройках проекта:
 В файле `settings.py` добавьте:
 ```python
 INSTALLED\_APPS = [
 # Другие приложения
 'library',
 ]
 ```

### Шаг 2: Настройка моделей

1. В файле `library/models.py` создайте следующие модели:
 - `Author`: модель для хранения информации об авторах книг (имя, биография).
 - `Category`: модель для категорий книг (например, "Наука", "Художественная литература").
 - `Book`: модель для книг, содержащая следующие поля:
 - `title` (название),
 - `author` (связь с моделью `Author`),
 - `category` (связь с моделью `Category`),
 - `description` (описание),
 - `published\_date` (дата публикации).

2. Пример модели:
 ```python
 from django.db import models

 class Author(models.Model):
 name = models.CharField(max\_length=200)
 biography = models.TextField()

 def \_\_str\_\_(self):
 return self.name

 class Category(models.Model):
 name = models.CharField(max\_length=100)

 def \_\_str\_\_(self):
 return self.name

 class Book(models.Model):
 title = models.CharField(max\_length=200)
 author = models.ForeignKey(Author, on\_delete=models.CASCADE)
 category = models.ForeignKey(Category, on\_delete=models.CASCADE)
 description = models.TextField()
 published\_date = models.DateField()

 def \_\_str\_\_(self):
 return self.title
 ```

3. Примените миграции:
 ```bash
 python manage.py makemigrations
 python manage.py migrate
 ```

### Шаг 3: Настройка панели администратора

1. Зарегистрируйте модели в `library/admin.py` для управления через панель администратора:
 ```python
 from django.contrib import admin
 from .models import Author, Category, Book

 admin.site.register(Author)
 admin.site.register(Category)
 admin.site.register(Book)
 ```

2. Создайте суперпользователя:
 ```bash
 python manage.py createsuperuser
 ```

3. Зайдите в панель администратора по адресу: `http://127.0.0.1:8000/admin` и добавьте несколько записей для авторов, категорий и книг.

### Шаг 4: Настройка представлений и маршрутов

1. В файле `library/views.py` создайте следующие представления:
 - `book\_list`: отображение списка всех книг.
 - `book\_detail`: отображение деталей конкретной книги.

2. Пример кода:
 ```python
 from django.shortcuts import render, get\_object\_or\_404
 from .models import Book

 def book\_list(request):
 books = Book.objects.all()
 return render(request, 'library/book\_list.html', {'books': books})

 def book\_detail(request, book\_id):
 book = get\_object\_or\_404(Book, id=book\_id)
 return render(request, 'library/book\_detail.html', {'book': book})
 ```

3. Настройте маршруты в `library/urls.py`:
 ```python
 from django.urls import path
 from . import views

 urlpatterns = [
 path('', views.book\_list, name='book\_list'),
 path('<int:book\_id>/', views.book\_detail, name='book\_detail'),
 ]
 ```

4. Подключите маршруты приложения в файле `library\_project/urls.py`:
 ```python
 from django.contrib import admin
 from django.urls import path, include

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

### Шаг 5: Создание шаблонов

1. Создайте папку `templates/library/` в приложении `library` и добавьте два файла:
 - `book\_list.html`: шаблон для отображения списка книг.
 - `book\_detail.html`: шаблон для отображения информации о книге.

2. Пример `book\_list.html`:
 ```html
 <!DOCTYPE html>
 <html>
 <head>
 <title>Список книг</title>
 </head>
 <body>
 <h1>Список книг</h1>
 <ul>
 {% for book in books %}
 <li>
 <a href="{{ book.id }}/">{{ book.title }}</a>
 - {{ book.author.name }}
 </li>
 {% endfor %}
 </ul>
 </body>
 </html>
 ```

3. Пример `book\_detail.html`:
 ```html
 <!DOCTYPE html>
 <html>
 <head>
 <title>{{ book.title }}</title>
 </head>
 <body>
 <h1>{{ book.title }}</h1>
 <p><strong>Автор:</strong> {{ book.author.name }}</p>
 <p><strong>Категория:</strong> {{ book.category.name }}</p>
 <p><strong>Описание:</strong> {{ book.description }}</p>
 <p><strong>Дата публикации:</strong> {{ book.published\_date }}</p>
 <a href="/">Вернуться к списку книг</a>
 </body>
 </html>
 ```

### Дополнительные задания:

1. Реализуйте поиск книг по названию или автору.
2. Создайте форму для добавления новых книг в базу данных.
3. Реализуйте функционал редактирования и удаления книг.

### Оценка выполнения:

1. \*\*Базовый уровень\*\*: Приложение работает, настроены модели, маршруты и шаблоны.
2. \*\*Средний уровень\*\*: Добавлены формы для добавления и редактирования данных.
3. \*\*Высокий уровень\*\*: Приложение полностью стилизовано, реализован поиск и другие дополнительные функции.