# Лабораторная работа: Работа с моделями в Django

## Цель работы:

Научиться создавать и работать с моделями в Django, а также изучить основные операции по созданию, чтению, обновлению и удалению записей в базе данных (CRUD-операции).

## Описание задания:

Создать простую модель в Django, настроить базу данных, выполнить миграции, создать несколько записей в базе данных через интерфейс администратора, а также реализовать функции для работы с моделями в представлениях.

## Инструменты:

- Установленная среда Python
- Установленный Django
- Текстовый редактор (например, PyCharm, Visual Studio Code)

## Ход работы:

### 1. Настройка базы данных:

В файле `settings.py` убедитесь, что базой данных по умолчанию является SQLite (или другая база данных, например, PostgreSQL, если она уже настроена):
```python
DATABASES = {
 'default': {
 'ENGINE': 'django.db.backends.sqlite3',
 'NAME': BASE\_DIR / 'db.sqlite3',
 }
}
```

### 2. Создание модели:

В приложении `myapp` откройте файл `models.py` и создайте модель `Book`, которая будет содержать информацию о книгах, такие как `title`, `author`, `published\_date`, и `isbn`:
```python
# myapp/models.py
from django.db import models

class Book(models.Model):
 title = models.CharField(max\_length=200)
 author = models.CharField(max\_length=100)
 published\_date = models.DateField()
 isbn = models.CharField(max\_length=13, unique=True)

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

### 3. Выполнение миграций:

Создайте миграцию для новой модели:
```bash
python manage.py makemigrations
```
Примените миграции для создания таблицы `Book` в базе данных:
```bash
python manage.py migrate
```

### 4. Добавление модели в админ-панель:

Зарегистрируйте модель `Book` в `admin.py` для управления записями через интерфейс администратора:
```python
# myapp/admin.py
from django.contrib import admin
from .models import Book

admin.site.register(Book)
```
Создайте суперпользователя для доступа в админ-панель:
```bash
python manage.py createsuperuser
```
Запустите сервер и войдите в админ-панель, перейдя по адресу `/admin/`, чтобы добавить несколько записей в модель `Book`.

### 5. Работа с моделью в представлениях:

Создайте представление для отображения списка всех книг:
```python
# myapp/views.py
from django.shortcuts import render
from .models import Book

def book\_list(request):
 books = Book.objects.all()
 return render(request, 'myapp/book\_list.html', {'books': books})
```
Создайте файл шаблона `book\_list.html` для отображения списка книг:
```html
<!-- templates/myapp/book\_list.html -->
<h1>Список книг</h1>
<ul>
 {% for book in books %}
 <li>{{ book.title }} by {{ book.author }} (ISBN: {{ book.isbn }})</li>
 {% endfor %}
</ul>
```
Добавьте маршрут для `book\_list` в `urls.py`:
```python
# myapp/urls.py
from django.urls import path
from . import views

urlpatterns = [
 path('books/', views.book\_list, name='book\_list'),
]
```

### 6. Операции CRUD в Django Shell:

Запустите интерактивную оболочку Django:
```bash
python manage.py shell
```
Создайте запись в модели `Book`:
```python
from myapp.models import Book
book = Book.objects.create(
 title="Пример книги",
 author="Автор",
 published\_date="2023-01-01",
 isbn="1234567890123"
)
```
Получите все записи:
```python
books = Book.objects.all()
for book in books:
 print(book.title)
```
Обновите запись:
```python
book.title = "Обновленное название"
book.save()
```
Удалите запись:
```python
book.delete()
```

### 7. Проверка работы:

Запустите сервер и откройте страницу `/books/` в браузере, чтобы увидеть список добавленных книг.

## Контрольные вопросы:

- Что такое модель в Django и для чего она используется?

- Как выполняются миграции в Django и для чего они нужны?

- Какие типы полей можно использовать в моделях Django?

- Как осуществить CRUD-операции с использованием Django ORM?

## Дополнительное задание:

Добавьте в модель `Book` новое поле `genre`, а также представление и шаблон для отображения подробной информации о книге, используя параметр `pk` в маршрутах URL.