

Московский государственный технический
университет им. Н. Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Отчет по домашнему

заданию

«Разработка прототипа веб-
приложения с использованием
фреймворка Django

Выполнил:

студент группы РТ5-51

Робертс Д.А.

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2021 г.

Полученное задание:

Цель домашнего задания: изучение возможностей создания прототипа веб-приложения на основе базы данных с использованием фреймворка Django.

На основе результатов лабораторных работ 4 и 5, создайте прототип веб-приложения с использованием фреймворка Django на основе базы данных, реализующий концепцию master/detail. Прототип должен содержать:

Две модели, связанные отношением один-ко-многим.

Стандартное средство администрирования Django позволяет редактировать данные моделей. Желательно настроить русификацию ввода и редактирования данных.

Веб-приложение формирует отчет в виде отдельного view/template, отчет выводит HTML-страницу, содержащую связанные данные из двух моделей.

Для верстки шаблонов используется фреймворк Bootstrap, или аналогичный фреймворк по желанию студента.

Текст программы:

Файл models.py

```
from django.db import models

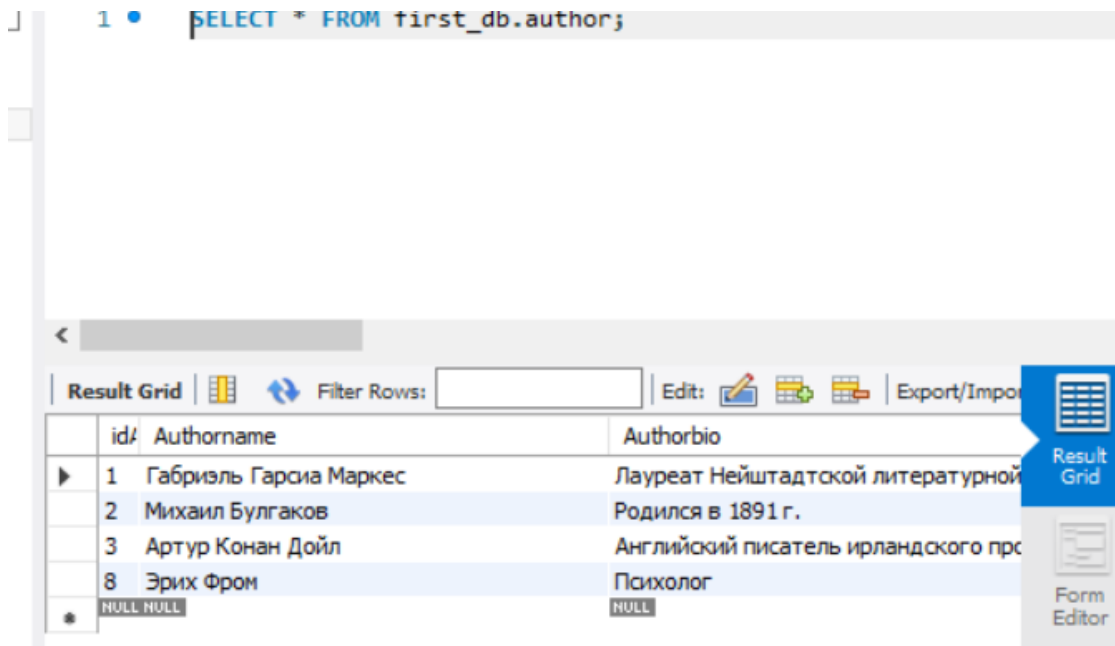
class Author(models.Model):
    idauthor = models.AutoField(db_column='idAuthor', primary_key=True) #
    # Field name made lowercase.
    authorname = models.CharField(db_column='Authorname', unique=True,
max_length=60) # Field name made lowercase.
    authorbio = models.CharField(db_column='Authorbio', max_length=100,
blank=True, null=True) # Field name made lowercase.

    class Meta:
        managed = True
        db_table = 'author'

class Book(models.Model):
    name = models.CharField(max_length=30, blank=True, null=True)
    dicription = models.CharField(max_length=255, blank=True, null=True)
    bookauthor = models.ForeignKey(Author, models.DO_NOTHING,
db_column='bookauthor', blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'book'
```

Данные хранятся на локальном сервере системы mySQL

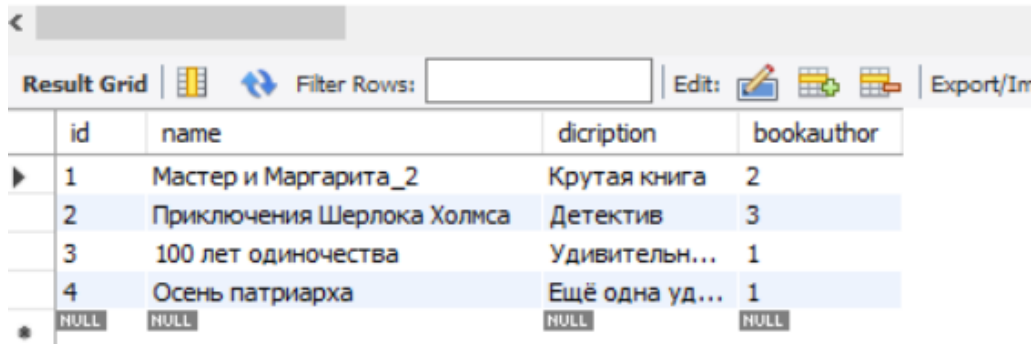


The screenshot shows a database management interface. At the top, a SQL query is entered: `SELECT * FROM first_db.author;`. Below the query, the results are displayed in a table with three columns: `id`, `Authorname`, and `Authorbio`. The results are as follows:

id	Authorname	Authorbio
1	Габриэль Гарсиа Маркес	Лауреат Нейштадтской литературной
2	Михаил Булгаков	Родился в 1891 г.
3	Артур Конан Дойл	Английский писатель ирландского про
8	Эрих Фром	Психолог
NULL	NULL	NULL

The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. On the right side, there are buttons for 'Result Grid' and 'Form Editor'.

```
1
2 • SELECT * FROM first_db.book;
```



The screenshot shows a database management interface with a 'Result Grid' tab. The grid contains 4 rows of data from the 'first_db.book' table. The columns are 'id', 'name', 'dcription', and 'bookauthor'. The data is as follows:

id	name	dcription	bookauthor
1	Мастер и Маргарита_2	Крутая книга	2
2	Приключения Шерлока Холмса	Детектив	3
3	100 лет одиночества	Удивительн...	1
4	Осень патриарха	Ещё одна уд...	1

Подключение бд к сайту в файле settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': "first_db",
        'USER': "dbuser",
        'PASSWORD': "123",
        'HOST': "localhost",
        'PORT': 3306, # Стандартный порт MySQL
        'OPTIONS': {'charset': 'utf8'},
        'TEST_CHARSET': 'utf8',
    }
}
```

Файл urls.py

```
from django.urls import path
from doge import views
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('', views.bookList, name='books_url'),
    path('delete_Book/<int:id>', views.delete_Book),
    path('edit/<int:id>', views.edit),
    path('create_book/', views.create_book),
    path('delete_Author/<int:idauthor>', views.delete_Author),
    path('edit_Author/<int:idAuthor>', views.edit_Author),
    path('create/', views.create),
]
urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
urlpatterns+=staticfiles_urlpatterns()
```

Файл views.py

```

from django.shortcuts import render
from django.http import HttpResponse
from django.shortcuts import render, redirect, get_object_or_404
from doge.models import Book
from datetime import date
from doge.models import Author
from django.http import HttpResponseRedirect
from django.http import HttpResponseNotFound

def bookList(request):
    return render(request, 'doge/books.html', {'data': {
        'current_date': date.today(),
        'books': Book.objects.all(),
        'dogs': Doge.objects.all(),
        'cats': Kit.objects.all(),
        'author': Author.objects.all()
    }})

def GetBook(request, id):
    return render(request, 'doge/book.html', {'data': {
        'current_date': date.today(),
        'book': Book.objects.filter(id=id)[0],
        'author': Author.objects.filter()
    }})

def delete_Book(request, id):
    book = get_object_or_404(Book, id=id)
    try:
        book.delete()
        return HttpResponseRedirect("/")
    except Book.DoesNotExist:
        return HttpResponseNotFound("<h2>file not found</h2>")

def delete_Author(request, idauthor):
    author = get_object_or_404(Author, idauthor=idauthor)
    try:
        author.delete()
        return HttpResponseRedirect("/")
    except Author.DoesNotExist:
        return HttpResponseNotFound("<h2>file not found</h2>")

def create(request):
    if request.method == "POST":
        dl = Author(authorname = request.POST.get("Authorname"),
,authorbio=request.POST.get("Authorbio"))

        dl.save()
        return HttpResponseRedirect("/")

def create_book(request):
    if request.method == "POST":
        book = Book(name=request.POST.get("name"),
dicription=request.POST.get("dicription"), bookauthor =
Author.objects.get(idauthor=request.POST.get("bookauthorr")))
        book.save()
        return HttpResponseRedirect("/")

def edit(request, id):
    try:

```

```

        book = Book.objects.get(id=id)
        if request.method == "POST":
            book.name = request.POST.get("name")
            book.dicription = request.POST.get("discription")
            book.bookauthor.idAuthor =
Author.objects.get(idauthor=request.POST.get("bookauthorr"))
            book.save()
            return HttpResponseRedirect("/")
        else:
            return render(request, "doge/edit.html", {"book": book})
    except Book.DoesNotExist:
        return HttpResponseRedirect("<h2>file not found</h2>")

def edit_Author(request, idAuthor):
    try:
        author = Author.objects.get(idauthor=idAuthor)
        if request.method == "POST":
            author.authorname = request.POST.get("authorname")
            author.authorbio = request.POST.get("authorbio")
            author.save()
            return HttpResponseRedirect("/")
        else:
            return render(request, "doge/edit_author.html", {"author":
author})
    except Book.DoesNotExist:
        return HttpResponseRedirect("<h2>file not found</h2>")

```

Экранные формы с примерами выполнения программы:

Страница состоит из нескольких блоков:

1) Книги

The screenshot shows a web application interface for managing books. The page is titled "Домашнее задание" (Homework) and displays a list of books under the heading "Книги". The books listed are "Мастер и Маргарита 2", "Приключения Шерлока Холмса", "100 лет одиночества", and "Осень патриарха". Each book entry has two buttons: "Изменить »" (Edit) and "Удалить »" (Delete). Below the list of books is a form titled "Добавление новой книги" (Add new book) with input fields for "Введите название книги" (Enter book title), "Введите описание" (Enter description), and "Введите ФИО автора" (Enter author's full name), followed by a "Сохранить" (Save) button. At the bottom, there is a section titled "Авторы" (Authors) with links to "Габриэль Гарсиа Маркес", "Михаил Булгаков", and "Артур Конан Дойл". The browser's address bar shows the URL "127.0.0.1:8000".

При нажатии на название откроется страница с подробной информацией о книге:

The screenshot shows a web application interface displaying detailed information about a book. The page is titled "Книга" (Book) and shows the following information: "Название: Приключения Шерлока Холмса" (Title: Adventures of Sherlock Holmes), "Описание: Детектив" (Description: Detective), and "Автор: Артур Конан Дойл" (Author: Arthur Conan Doyle). The browser's address bar shows the URL "127.0.0.1:8000".

При нажатии изменить происходит переход на форму для изменения информации о книге:

Книга

Главная Об авторе

Введите название книги
Мастер и Маргарита_2

Введите описание книги

Введите ФИО автора
2

Сохранить

При нажатии кнопки удалить происходит удаление записи.

Если ввести данные в форму «Добавление новой книги» и нажать сохранить, то новая книга появится на странице.

2) Авторы

← PyCharm → 127.0.0.1:8000 Домашнее задание 215 отзывов

Авторы

Габриэль Гарсиа Маркес

Изменить »

Удалить »

Михаил Булгаков

Изменить »

Удалить »

Артур Конан Дойл

Изменить »

Удалить »

Эрих Фром

Изменить »

Удалить »

Добавление нового автора

Введите ФИО автора

Введите информацию об авторе

Сохранить

Произведения Габриэль Гарсиа Маркес:

Возможности аналогичны возможностям в предыдущем блоке.

3) Отображение книг по авторам:

Произведения Габриэль Гарсиа Маркес:

- 100 лет одиночества
- Осень патриарха

Произведения Михаил Булгаков:

- Мастер и Маргарита_2

Произведения Артур Конан Дойл:

- Приключения Шерлока Холмса

Произведения Эрих Фром: