

Оглавление

Введение	2
Описание результата	3
Чек-лист: Модуль создание проекта:	13
Чек-лист: Модуль «Базы данных»	16
Чек-лист: Модуль «Тестирование»	20
Ссылка на гит.....	23
Заключение	24
Список литературы.....	Error! Bookmark not defined.

Введение

Данная работа представляет собой создание веб-приложения «Книжный магазин» с использованием фреймворка Django.

В магазине продаются книги различных жанров. Книги размещены по стеллажам, а опытный продавец знает, как подобрать книги по схожим тематикам, автору или издательству. Продавцу также приходится принимать новые поступления и размещать их на стеллажах. Покупатель в поиске конкретной книги может обратиться к продавцу за помощью. Информационная система предназначена для продавца.

Описание результата

Проект представляет собой архив для хранения книг, книги имеют авторов и распределены по полкам.

Модели данных сущностей представлены на рисунках 1-4.

```
class Author(models.Model):
    name = models.CharField(max_length=256, verbose_name="Имя")
    def __str__(self):
        return self.name
```

Рисунок 1 – Модель автора

```
class BaseBlog(models.Model):
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name="Добавлено"
    )

    class Meta:
        abstract = True
```

Рисунок 2 – Базовая модель

```
class Shelf(BaseBlog):
    title = models.CharField(max_length=256, verbose_name="Название")
    description = models.TextField(verbose_name="Описание")
    slug = models.SlugField(unique=True,
        verbose_name="Идентификатор",
        help_text="Идентификатор страницы для URL; "
        "разрешены символы латиницы, "
        "цифры, дефис и подчёркивание."
    )
    def __str__(self):
        return self.title

    class Meta:
        verbose_name = 'полка'
        verbose_name_plural = 'Полки'
```

Рисунок 3 – Модель полки

```

class Book(BaseBlog):
    title = models.CharField(max_length=256, verbose_name="Заголовок")
    text = models.TextField(verbose_name="Текст", )
    price = models.DecimalField(max_digits=10, decimal_places=2)
    amount = models.IntegerField(verbose_name="Остаток")
    author = models.ForeignKey(
        Author,
        on_delete=models.CASCADE,
        verbose_name="Автор книги",
        related_name="posts"
    )
    shelf = models.ForeignKey(
        Shelf,
        on_delete=models.SET_NULL,
        verbose_name="Полка",
        null=True
    )
    image = models.ImageField(
        upload_to="media/",
        null=True,
        verbose_name="Обложка",
        blank=True,
    )

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = 'книга'
        verbose_name_plural = 'Книги'

```

Рисунок 4 – Модель книги

На рисунках 5-11 представлены основные страницы проекта, подробные страницы книг и полок, а также страницы с добавлением книг, полок и авторов.

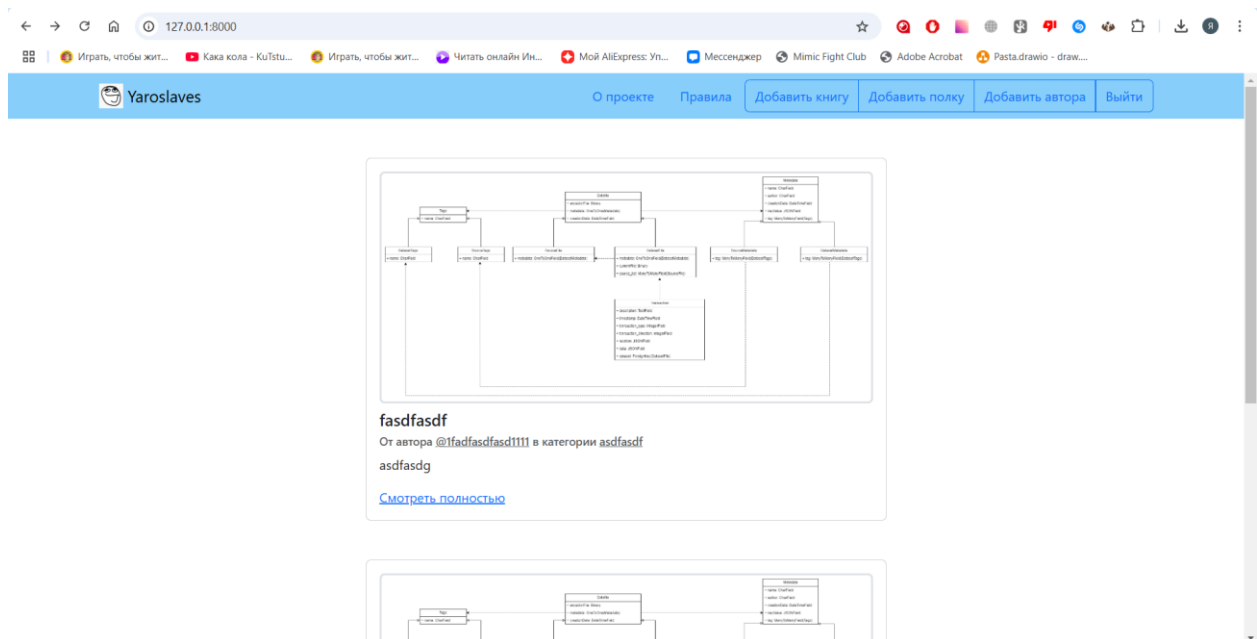


Рисунок 5 – Главная страница

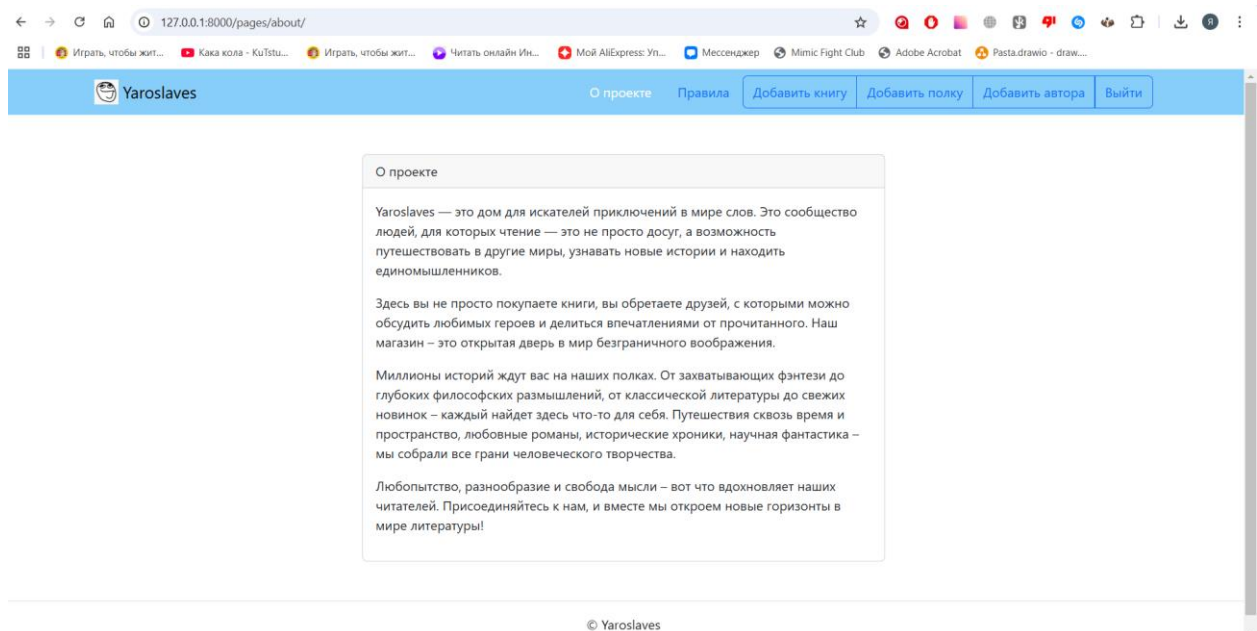


Рисунок 6 – О проекте

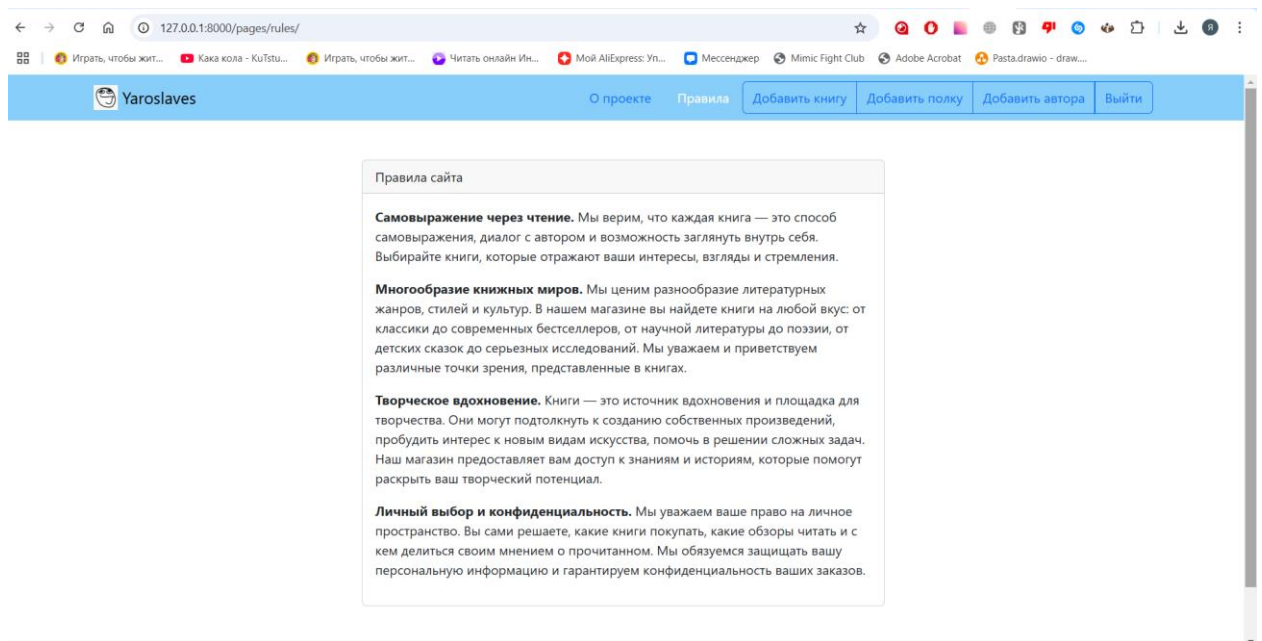


Рисунок 7 – Правила

A screenshot of the 'Страница добавления' (Add Page) form on the Yaroslaves website. The form is a white box with a grey border. It contains several input fields and a submit button. The fields are: 'Заголовок' (Title) with the value 'Заголовок', 'Текст' (Text) with the value 'Текст', 'Price' with the value 'Price', 'Остаток' (Remaining) with the value 'Остаток', 'Автор книги' (Book Author) with a dropdown menu showing '-----', 'Полка' (Shelf) with a dropdown menu showing '-----', and 'Обложка' (Cover) with a file selection button 'Выберите файл' and a status 'Файл не выбран'. At the bottom of the form is a blue button labeled 'Отправить' (Send).

Рисунок 8 – Страница добавления книги

Страница добавления

Название

Описание

Идентификатор

Идентификатор страницы для URL: разрешены символы латиницы, цифры, дефис и подчёркивание.

Отправить

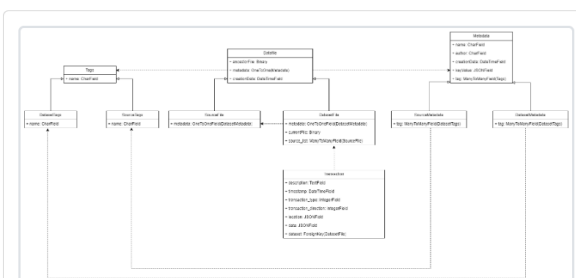
Рисунок 9 – Страница добавления полки

Страница добавления

Имя

Отправить

Рисунок 10 – Страница добавления автора



fasdfasdf

Цена: 22,00


Осталось: 2223

От автора [@1fadfasdfasd1111](#) на полке [asdfasdf](#)

asdfasd

[Отредактировать](#) [Удалить](#)

Рисунок 11 – Подробная страница книги

 Yaroslaves

[О проекте](#) [Правила](#) [Добавить книгу](#) [Добавить полку](#) [Добавить автора](#) [Выйти](#)

Редактирование книги

Заголовок

fasdfasdf

Текст

asdfasd

Price

22,00

Остаток

2223

Автор книги

1fadfasdfasd1111

Полка

asdfasdf

Обложка


Выберите файл

newhpcover-672x1024.png

На данный момент: [media/UIRS.drawio.png](#) ☐ Очистить

Отправить

Рисунок 12 – Страница редактирования книги

 Yaroslaves

[О проекте](#) [Правила](#) [Добавить книгу](#) [Добавить полку](#) [Добавить автора](#) [Выйти](#)

Удаление книги

| Планета Земля

g12g

g12g

Отправить

Рисунок 13 – Страница удаления книги

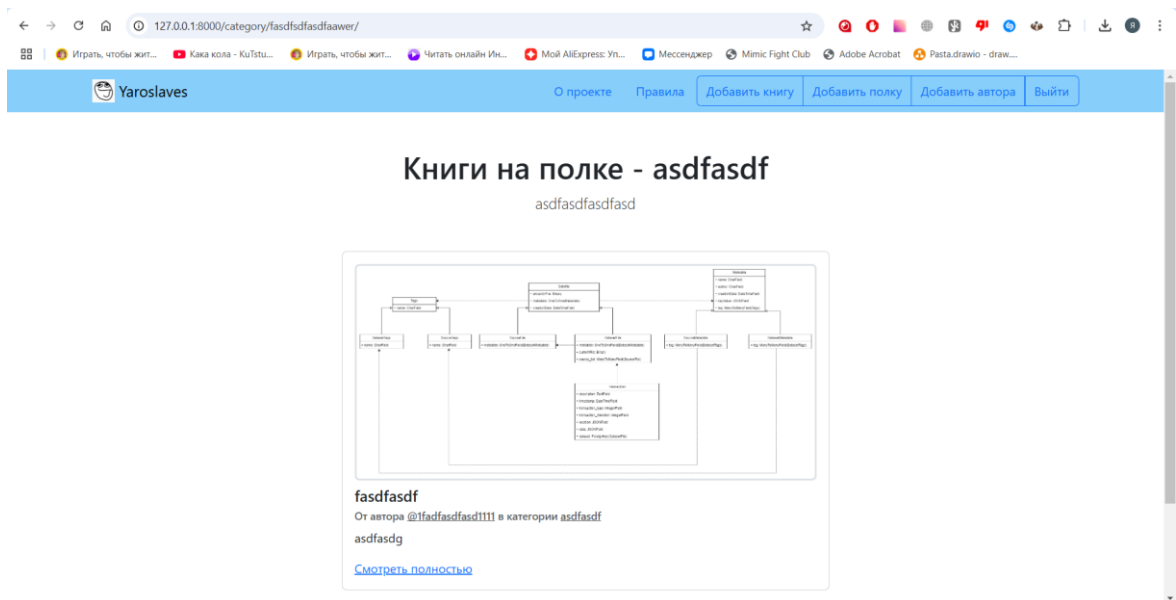


Рисунок 14 – Фильтрации по полке

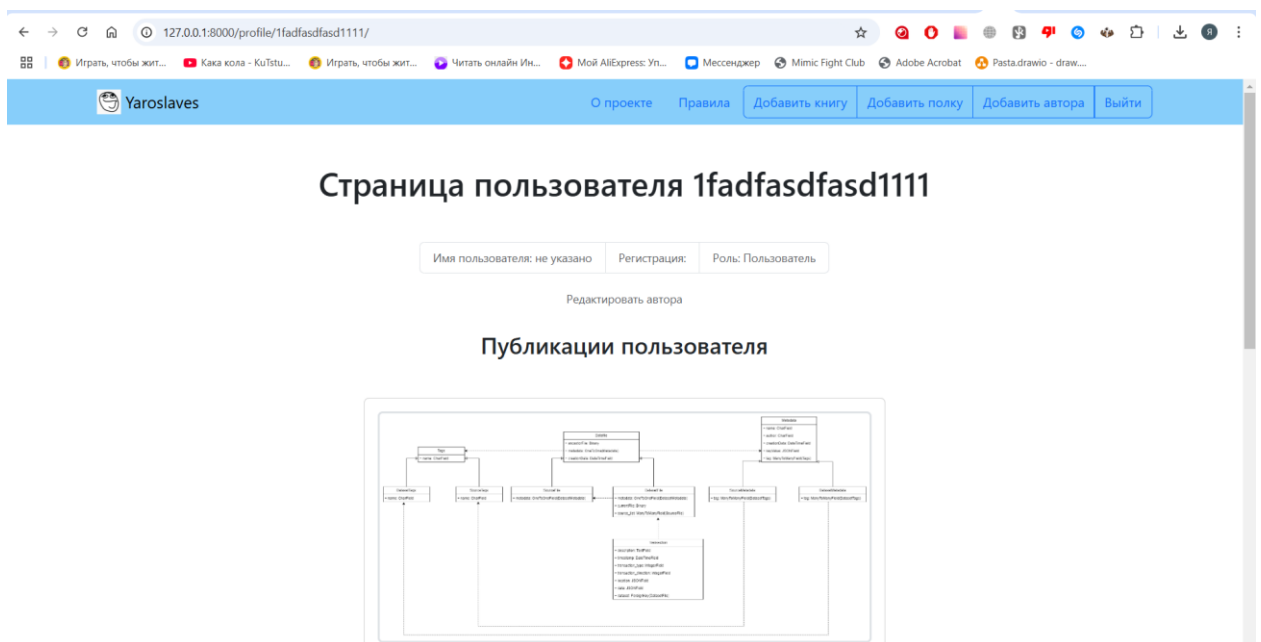


Рисунок 15 – Фильтрации по автору

Страницы добавления контента работают корректно:

Страница добавления

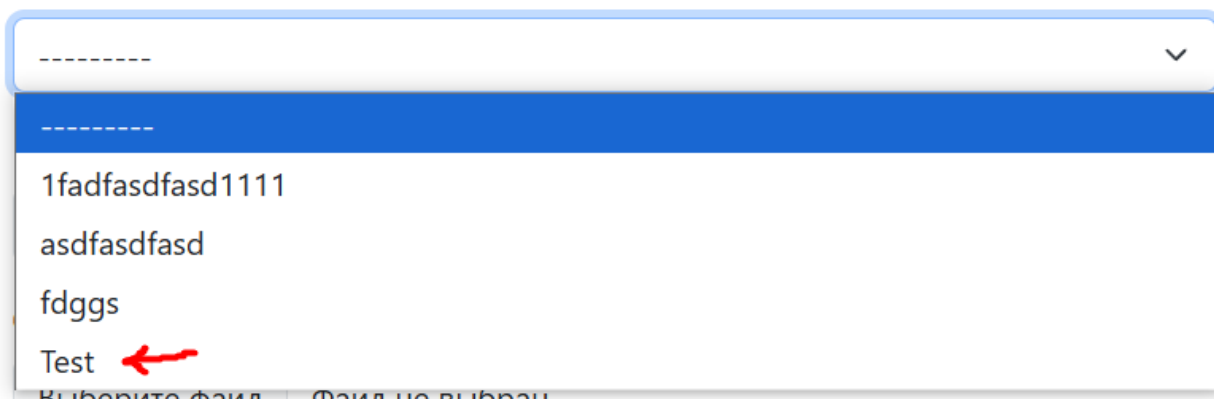
Имя

Test

Отправить

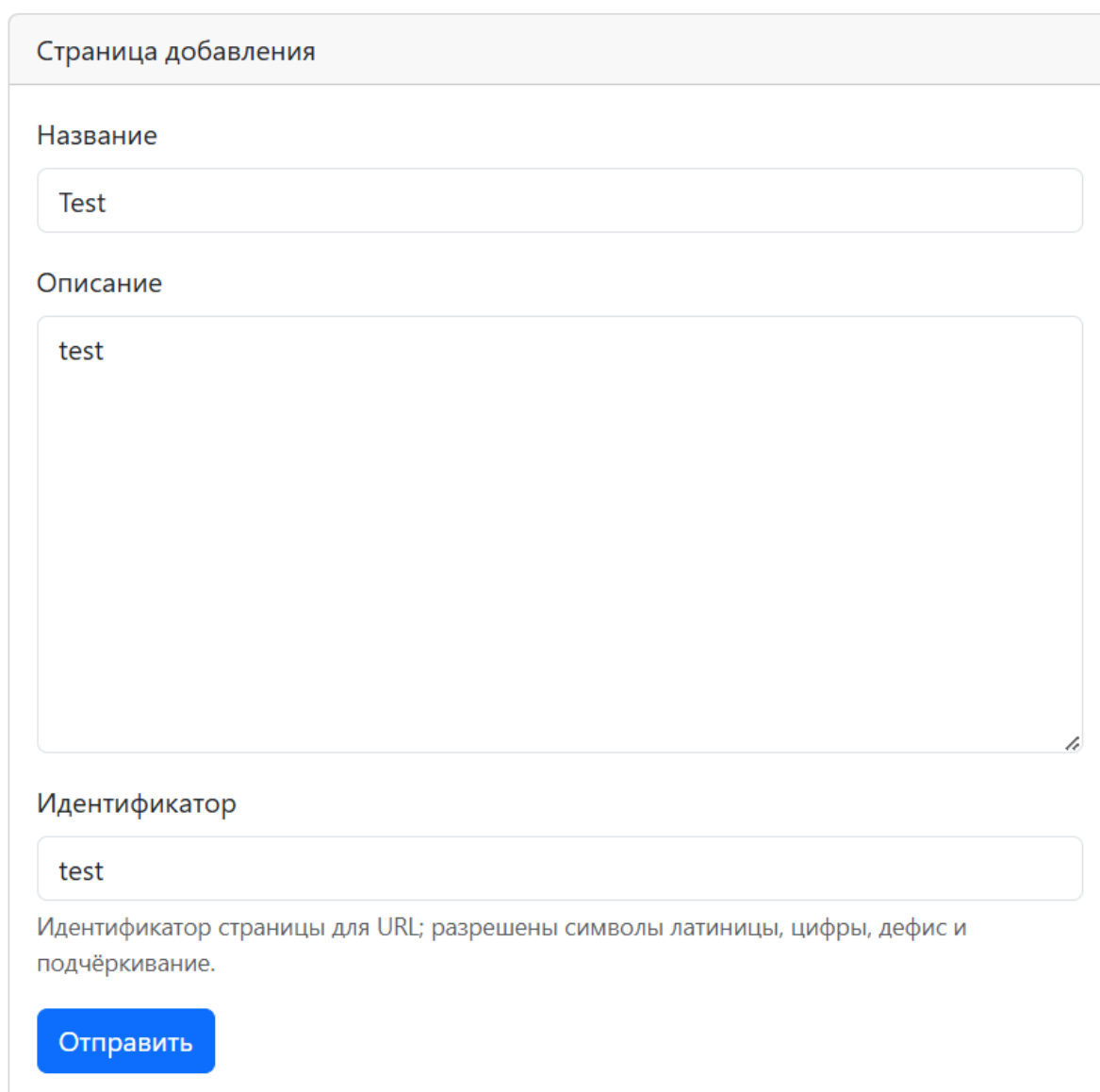
Рисунок 16 – Добавления автора, часть 1

Автор книги



A screenshot of a web interface showing a dropdown menu for selecting an author. The menu is open, displaying a list of author names. The first item is "1fadfasdfasd1111", followed by "asdfasdfasd", "fdggs", and "Test". A red arrow points to the "Test" option. The menu has a blue header bar and a white body. The text "Выберите файл" and "Файл не выбран" is visible at the bottom of the menu.

Рисунок 17 – Добавления автора, часть 2



A screenshot of a web form titled "Страница добавления" (Add Page). The form contains three input fields: "Название" (Name) with the value "Test", "Описание" (Description) with the value "test", and "Идентификатор" (Identifier) with the value "test". Below the identifier field, there is a note: "Идентификатор страницы для URL; разрешены символы латиницы, цифры, дефис и подчёркивание." (Page identifier for URL; Latin letters, digits, hyphen, and underscore are allowed). At the bottom of the form is a blue button labeled "Отправить" (Send).

Рисунок 18 – Добавления полки, часть 1

Полка

Mewq

asdfasdf

sdfg

Test

Рисунок 19 – Добавления полки, часть 2

Страница добавления

Заголовок

Test

Текст

test

Price

11

Остаток

11

Автор книги

Test

Полка

Test

Обложка

Выберите файл newhpcover-672x1024.png

Отправить

Рисунок 20 – Добавления книги, часть 1

Страница пользователя Test

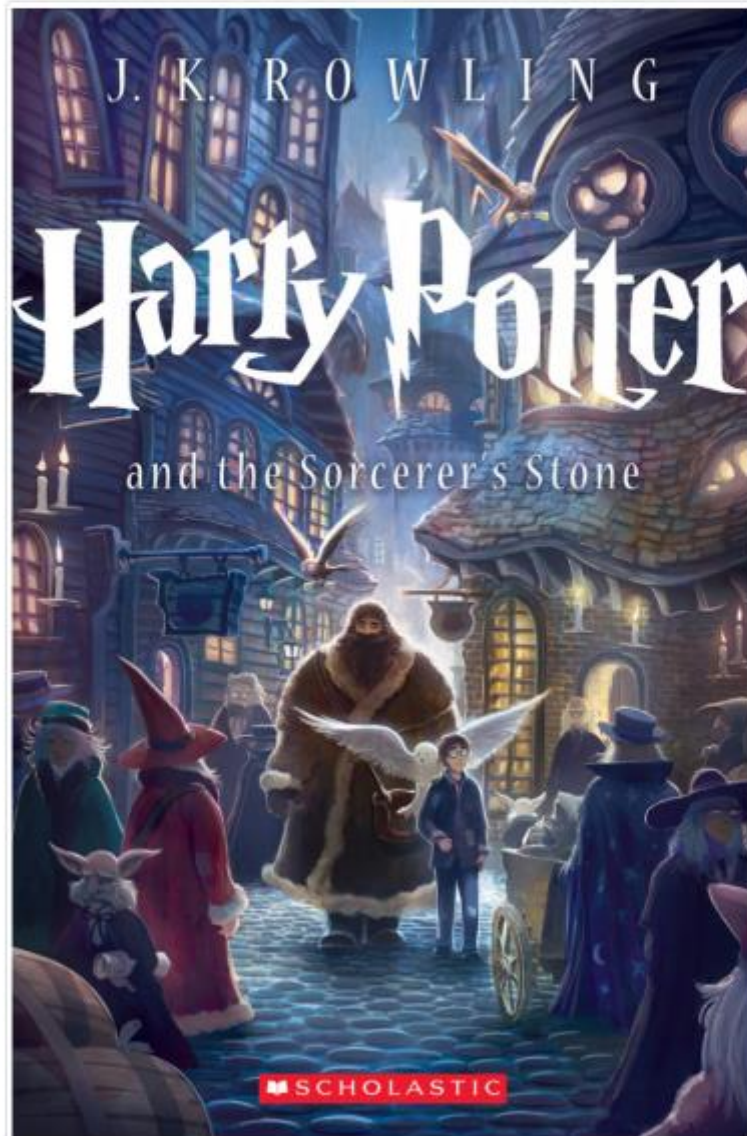
Имя пользователя: не указано

Регистрация:

Роль: Пользователь

[Редактировать автора](#)

Публикации пользователя



Test

От автора [@Test](#) в категории [Test](#)

test

[Скрыть рекламу](#)

Рисунок 21 – Добавления книги, часть 2

Чек-лист: Модуль создание проекта:

1. Обязательные критерии

- a. Все автотесты успешны – автотестов нет, требование неприменимо.
- b. Функция `index` (главная страница) отображает список книг с учетом пагинации и фильтрации по `amount__gte=1`. (рисунок 22)

```
def index(request):
    books = Book.objects.filter(
        amount__gte=1,
    )
    paginator = Paginator(books, 10)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, 'blog/index.html', {'page_obj': page_obj})
```

Рисунок 22 – Функция `index`

- c. Обработка ошибок: используется `get_object_or_404` для обработки случаев, когда объект не найден (рисунок 23).

```
def post_delete(request, post_id):
    book = get_object_or_404(Book, pk=post_id)
    instance = get_object_or_404(Book, id=post_id)
    form = PostForm(instance=book)
    context = {'form': form}
    if request.method == 'POST':
        instance.delete()
        return redirect('blog:index')
    return render(request, 'blog/create.html', context)
```

Рисунок 23 – Пример использования `get_object_or_404`

- d. Поле `slug` должно быть уникальным (рисунок 24).

```
class Shelf(BaseBlog):
    title = models.CharField(max_length=256, verbose_name="Название")
    description = models.TextField(verbose_name="Описание")
    slug = models.SlugField(unique=True,
                           verbose_name="Идентификатор",
                           help_text="Идентификатор страницы для URL; "
                                     "разрешены символы латиницы, "
                                     "цифры, дефис и подчёркивание.")

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

Рисунок 24 – Уникальность поля slug

- е. Внутри {название_проекта}/urls.py не должно быть маршрутов из приложения, кроме include (рисунок 25).

```
urlpatterns = [
    path('pages/', include('pages.urls')),
    path('admin/', admin.site.urls),
    path(
        'auth/registration/',
        CreateView.as_view(
            template_name='registration/registration_form.html',
            form_class=UserForm,
            success_url=reverse_lazy('blog:index'),
        ),
        name='registration',
    ),
    path('auth/', include('django.contrib.auth.urls')),
    path('', include('blog.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

handler404 = 'pages.views.page_not_found'
handler500 = 'pages.views.page_internal_server_error'
```

Рисунок 25 – urls

2. Желательные критерии

- а. Параметры в URL-адресах именованы содержательно (например, post_id, category_slug, username) (примеры на рисунках выше).

б. Код форматирован согласно PEP8 (рисунки 26-27).

```
<a class="nav-link {% if view_name == 'pages:rules' %} text-white {% endif %}" href="{% url 'pages:rules' %}" class="nav-link" >Правила</a>
</li>
{% if user.is_authenticated %}
<div class="btn-group" role="group" aria-label="Basic outlined example">
  <button type="button" class="btn btn-outline-primary"><a class="text-decoration-none text-reset" href="{% url 'blog:create_post' %}">Добавить книгу</a></button>
  <button type="button" class="btn btn-outline-primary"><a class="text-decoration-none text-reset" href="{% url 'blog:create_shelf' %}">Добавить полку</a></button>
  <button type="button" class="btn btn-outline-primary"><a class="text-decoration-none text-reset" href="{% url 'blog:create_author' %}">Добавить автора</a></button>
  <button type="button" class="btn btn-outline-primary"><a class="text-decoration-none text-reset" href="{% url 'logout' %}">Выйти</a></button>
</div>
{% else %}
<div class="btn-group" role="group" aria-label="Basic outlined example">
  <button type="button" class="btn btn-outline-primary"><a class="text-decoration-none text-reset" href="{% url 'login' %}">Войти</a></button>
</div>
{% endif %}
</ul>
{% endwith %}
</div>
```

Рисунок 26 – Пример HTML

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Book, Shelf, Author
from django.core.paginator import Paginator
from django.views import View
from blog.forms import UserEditForm, PostForm, ShelfForm
from django.contrib.auth.decorators import login_required

# Create your views here.
def index(request):
    books = Book.objects.filter(
        amount__gte=1,
    )
    paginator = Paginator(books, 10)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, 'blog/index.html', {'page_obj': page_obj})

def create_edit(request, book_id=None):
    book = get_object_or_404(
        Book,
```

Рисунок 27 – Пример оформления кода

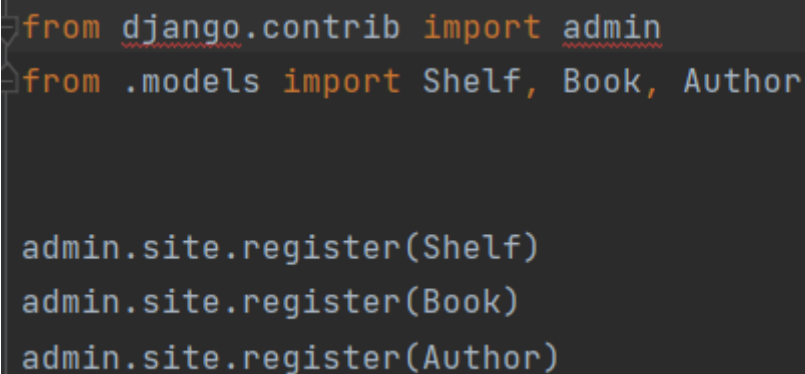
3. Факультативные критерии.

- а. При броске исключения после провала проверки параметра в функциях применять Django-класс `Http404`.
Выполняется использованием `get_object_or_404`.

Чек-лист: Модуль «Базы данных»

1. Обязательные критерии

- а. Все автотесты пройдены успешно – автотестов нет, требование неприменимо.
- б. Для всех моделей есть разделы в админке (рисунок 28).

A screenshot of a code editor showing Python code for Django model registration. The code is as follows:

```
from django.contrib import admin
from .models import Shelf, Book, Author

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

Рисунок 28 – регистрация моделей

- с. Модель `BaseBlog` является абстрактной, и не создает таблицу в БД – выполнено.
- д. В модели `Book` настроено поле `image` для загрузки изображений, и указана директория загрузки – выполнено (рисунок 29).


```

class Book(BaseBlog):
    title = models.CharField(max_length=256, verbose_name="Заголовок")
    text = models.TextField(verbose_name="Текст", )
    price = models.DecimalField(max_digits=10, decimal_places=2)
    amount = models.IntegerField(verbose_name="Остаток")
    author = models.ForeignKey(
        Author,
        on_delete=models.CASCADE,
        verbose_name="Автор книги",
        related_name="posts"
    )
    shelf = models.ForeignKey(
        Shelf,
        on_delete=models.SET_NULL,
        verbose_name="Полка",
        null=True
    )
    image = models.ImageField(
        upload_to="media/",
        null=True,
        verbose_name="Обложка",
        blank=True,
    )

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = 'книга'
        verbose_name_plural = 'Книги'

```

Рисунок 29– Модель book

f. ForeignKey с on_delete=models.CASCADE используется для автора, чтобы записи не «висели» при удалении автора. ForeignKey с on_delete=models.SET_NULL используется для полки, чтобы записи не «висели» при удалении полки – выполнено.

g. Модель BaseBlog использует абстрактный класс для общего поведения моделей – выполнено (рисунок 30).

```
class BaseBlog(models.Model):
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name="Добавлено"
    )
```

Рисунок 30 – Модель BaseBlog

2. Желательные критерии

а. Все поля у всех моделей содержат переводы через параметры verbose_name (рисунки 31-33).

```
class Author(models.Model):
    name = models.CharField(max_length=256, verbose_name="Имя")
    def __str__(self):
        return self.name

class BaseBlog(models.Model):
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name="Добавлено"
    )

    class Meta:
        abstract = True

class Shelf(BaseBlog):
    title = models.CharField(max_length=256, verbose_name="Название")
    description = models.TextField(verbose_name="Описание")
    slug = models.SlugField(unique=True,
        verbose_name="Идентификатор",
        help_text="Идентификатор страницы для URL; "
        "разрешены символы латиницы, "
        "цифры, дефис и подчёркивание.")
```

Рисунок 31 – Модели, часть 1

```

def __str__(self):
    return self.title

class Meta:
    verbose_name = 'полка'
    verbose_name_plural = 'Полки'

class Book(BaseBlog):
    title = models.CharField(max_length=256, verbose_name="Заголовок")
    text = models.TextField(verbose_name="Текст", )
    price = models.DecimalField(max_digits=10, decimal_places=2)
    amount = models.IntegerField(verbose_name="Остаток")
    author = models.ForeignKey(
        Author,
        on_delete=models.CASCADE,
        verbose_name="Автор книги",
        related_name="posts"
    )
    shelf = models.ForeignKey(
        Shelf,
        on_delete=models.SET_NULL,
        verbose_name="Полка",
        null=True
    )

```

Рисунок 32 – Модели, часть 2

```

image = models.ImageField(
    upload_to="media/",
    null=True,
    verbose_name="Обложка",
    blank=True,
)

def __str__(self):
    return self.title

class Meta:
    verbose_name = 'книга'
    verbose_name_plural = 'Книги'

```

Рисунок 33 – Модели, часть 3

- b. У каждой модели есть переводы в полях `verbose_name` и `verbose_name_plural` во вложенном классе `Meta` (рисунки 31-33).
- c. Использование `null=True` и `blank=True` для необязательных полей (рисунки 30-32).
- f. Всем моделям полезно добавить метод `def __str__(self):`. Он будет помогать при IDE-отладке и работе в админке (рисунки 31-33).
- j. Чтобы админка «говорила» с посетителем по-русски, нужно задать в `settings.py` настройку `LANGUAGE_CODE = 'ru-RU'` (рисунок 34).

```
LANGUAGE_CODE = 'ru-RU'
```

Рисунок 34 – Переключение на русский язык

Чек-лист: Модуль «Тестирование»

- a. Добавлены нестандартные страницы ошибок (рисунки 35-37).

```
{% extends "base.html" %}
{% block title %}Ошибка CSRF токена{% endblock %}
{% block content %}
    <h1>Ошибка CSRF токена. 403</h1>
    <a href="{% url 'blog:index' %}">Вернуться на главную</a>
{% endblock %}
```

Рисунок 35 – Ошибка 403

```
{% extends "base.html" %}
{% block title %}Страница не найдена{% endblock %}
{% block content %}
    <h1>Страница не найдена</h1>
    <p>Страницы с адресом {{ request.build_absolute_uri }} не существует!</p>
    <a href="{% url 'blog:index' %}">Вернуться на главную</a>
{% endblock %}
```

Рисунок 36 – Ошибка 404

```
{% extends "base.html" %}
{% block title %}Ошибка сервера{% endblock %}
{% block content %}
    <h1>Ошибка сервера</h1>
    <p>На сервере что-то пошло не так!</p>
    <a href="{% url 'blog:index' %}">Вернуться на главную</a>
{% endblock %}
```

Рисунок 37 – Ошибка 500

б. Подключены представления для работы с пользователем + представление для регистрации (рисунок 38).

Страница пользователя 1fadfasdfasd1111

Имя пользователя: не указано	Регистрация:	Роль: Пользователь
------------------------------	--------------	--------------------

[Редактировать автора](#)

Публикации пользователя

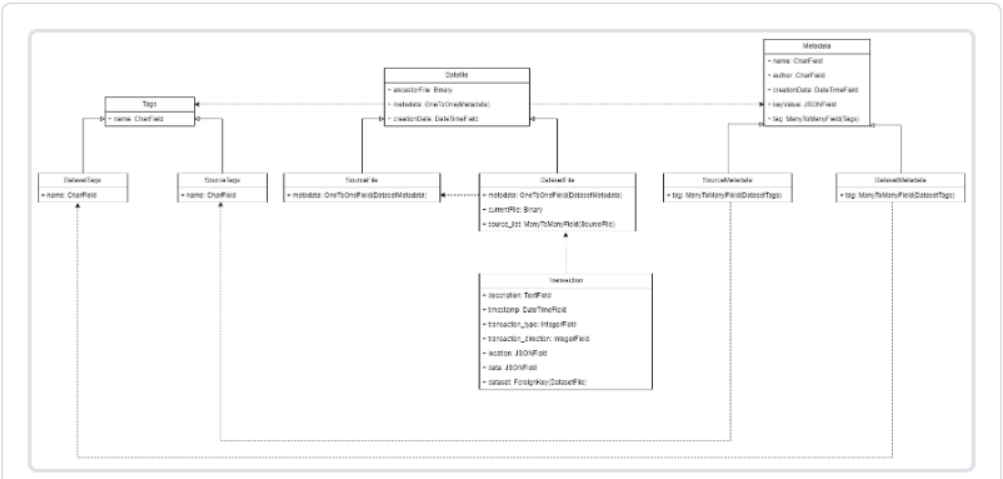


Рисунок 38 – Профиль автора

с. Добавлен постраничный вывод данных – выполнено (пример – рисунок 39).

```

def category_posts(request, category_slug):
    category = get_object_or_404(
        Shelf,
        slug=category_slug,
    )
    books = Book.objects.filter(
        amount__gte=1,
        shelf=category
    )
    paginator = Paginator(books, 10)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(
        request,
        'blog/category.html',
        {
            'page_obj': page_obj,
            'category': category
        }
    )

```

Рисунок 39 – Пример пагинации

d. Проект использует файловый бэкенд электронной почты в качестве заглушки (рисунок 40).

```

EMAIL_BACKEND = "django.core.mail.backends.filebased.EmailBackend"
EMAIL_FILE_PATH = BASE_DIR / 'sent_emails'

```

Рисунок 40 – Настройка почты

Ссылка на гит

<https://github.com/MrHumanis/Books>

Заключение

В рамках выполнения проекта, связанного с созданием информационной системы для книжного магазина, был приобретен практический опыт работы с фреймворком Django. Процесс разработки позволил углубить понимание принципов построения веб-приложений и освоить новые навыки, такие как управление файлами для организации данных о книгах, их авторах и жанрах. В результате была создана платформа, позволяющая продавцу эффективно управлять ассортиментом, включая добавление новых поступлений и поиск книг по различным критериям, что соответствует поставленным задачам и требованиям проекта.

