Финальный проект

Путь Джеффа Бизона близится к завершению и теперь ему необходимо собрать полный функционал сервиса по продаже книг и журналов! В данном задании вам необходимо реализовать полный функционал для корректного взаимодействия с сервисом по размещению книг и журналов.

Модели

В проекте используется 3 модели с следующими типами реляций:

- Модель `Book` (книги) связана с `RegularUser` связью `Many-to-One`.
- Модель `Journal` (журналы) связана с `RegularUser` связью `Many-to-One`.
- При удалении пользователя должны удаляться все реляции с его книгами и журналами.

Описание модели *RegularUser*

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.
class RegularUser(AbstractUser):
    phone_number = models.TextField(null=False, blank=False) # Номер телефона
пользователя
    postal_code = models.TextField(null = False, blank=False) # Почтовый индекс
пользователя
    age = models.PositiveIntegerField(null=True, blank=True) # Возраст
пользователя
    sex = models.BooleanField(null=True, blank=True) # Пол пользователя
```

Описание модели *Book*

```
from django.db import models
from django.contrib.auth import get_user_model # отдает модель RegularUser
from django.urls import reverse

class Book(models.Model):
    book_title = models.CharField(max_length=250) # Название книги
    book_author = models.CharField(max_length=200) # Автор книги
    rating = models.IntegerField() # Рейтинг книги
    date = models.DateTimeField(auto_now_add=True) # Дата размещения на сервисе
    owner = models.ForeignKey(get_user_model(), on_delete=models.CASCADE) #
Пользователь, разместивший книгу


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

```
    def get_absolute_url(self):
        return reverse("detail_book", args=[str(self.id)])
```

Описание модели *Journal*

```
from django.db import models
from django.contrib.auth import get_user_model # отдает модель RegularUser
from django.urls import reverse

class Journal(models.Model):
    journal_title = models.CharField(max_length=250) # Название журнала
    redaction = models.CharField(max_length=100) # Редактор журнала
    pages_amount = models.IntegerField() # Количество страниц журнала
    date = models.DateTimeField(auto_now_add=True) # Дата размещения на сервисе
    owner = models.ForeignKey(get_user_model(), on_delete=models.CASCADE) #
Пользователь, разместивший журнал


    def __str__(self):
        return self.journal_title


    def get_absolute_url(self):
        return reverse("detail_journal", args=[str(self.id)])
```

Учесть

При регистрации пользователь в сервисе обязан заполнять поля `username`, `password` , а также `email` , `phone_number` и `postal_code`, для этого сделайте необходимые правки на этапе создания форм взаимодействия с моделью.

Отображение пользователя в панели администратора

В панели администратора информация о пользователе должна находиться в соответствии со списком: `username`, `phone_number`, `email`, `postal_code`.

Аккаунтинг

Для корректной работы пользователя в сервисе необходимо наделить юзера базовым функционалом:

- `/user/login` - страница логина
- `/user/signup/` - страница регистрации
- `/user/logout/` - ссылка для разлогинивания
- `/user/password_change/` и `/user/password_change/done/` для смены пароля
- `/user/password_reset/` + все сопутствующие `URL` ссылки для реализации механизма сброса пароля.

Сброс пароля реализовать при помощи выброса ссылки в командную строку (терминал).

Book CrUd

Для корректной работы с книгами необходимо реализовать поддержку следующих `URL` запросов:

- ***Доступ***: `/books/` ***Действие***: страница, содержащая список книг (в произвольном порядке). Про каждую книгу требуется указать: ***Название***(кликабельная ссылка, отправляющая на страницу с детальной информацией про данную книгу), ***Автор книги*** , ***Рейтинг***, ***Дата и время добавления на сервис***, ***Собственник, разместивший книгу***. Страница доступна только аутентифицированным пользователям. Никнейм запроса: `books`.

- ***Доступ***: `/books/new/` ***Действие***: страница создания новой книги. При создании книги необходимо указать: ***Название***, ***Автор книги*** , ***Рейтинг***. Книги на сервис может добавлять только аутентифицированный (залогиненный) пользователь. Никнейм: `new_book`.

- ***Доступ***: `/books/<int:pk>/detail/` ***Действие***: страница с детальной информацией про книгу. Содержит ***Название***, ***Автор книги*** , ***Рейтинг***, ***Дата и время добавления на сервис***, ***Собственник, разместивший книгу***. А также ссылки для перехода на `books` , `edit_book` и `delete_book`. Страница доступна только аутентифицированным пользователям. ***Никнейм***: `detail_book`.

- ***Доступ***: `/books/<int:pk>/delete/` ***Действие***: страница удаления текущей книги. Содержит на себе только кнопку подтверждения удаления. Страница доступна только аутентифицированным пользователям. ***Никнейм***: `delete_book`.

- ***Доступ***: `/books/<int:pk>/edit/` ***Действие***: страница редактирования текущей книги. Содержит поля редактирования ***Название***,***Автор книги*** , ***Рейтинг***. Так же присутсвует ссылка на `books`. Страница доступна только аутентифицированным пользователям. ***Никнейм***: `edit_book`.

Journal CrUD

Для корректной работы с журналами необходимо реализовать поддержку следующих `URL` запросов:

- ***Доступ***: `/journals/` ***Действие***: страница, содержащая список журналов (в произвольном порядке). Про каждый журнал требуется указать: ***Название журнала***(кликабельная ссылка, отправляющая на страницу с детальной информацией про данный журнал), ***Редактор*** , ***Количество страниц***, ***Дата и время добавления на сервис***, ***Собственник, разместивший журнал***. Страница доступна только аутентифицированным пользователям. Никнейм запроса: `journals`.

- ***Доступ***: `/journals/new/` ***Действие***: страница создания нового журнала. При создании журнала необходимо указать: ***Название журнала***, ***Редактора*** , ***Количество страниц***. Журналы на сервис может добавлять только аутентифицированный (залогиненный) пользователь. Никнейм: `new_journal`.

- ***Доступ***: `/journals/<int:pk>/detail/` ***Действие***: страница с детальной информацией про журнал. Содержит ***Название журнала***, ***Редактора журнала*** , ***Количество страниц***, ***Дата и время добавления на сервис***, ***Собственник, разместивший журнал***. А также ссылки для перехода на `journals` , `edit_journal` и `delete_journal`. Страница доступна только аутентифицированным пользователям. ***Никнейм***: `detail_journal`.

- ***Доступ***: `/journals/<int:pk>/delete/` ***Действие***: страница удаления текущего журнала. Содержит на себе только кнопку подтверждения удаления. Страница доступна только аутентифицированным пользователям. ***Никнейм***: `delete_journal`.

- ***Доступ***: `/journals/<int:pk>/edit/` ***Действие***: страница редактирования текущего журнала. Содержит поля редактирования ***Название журнала***,***Редактор журнала*** , ***Количество страниц***. Так же присутсвует ссылка на `journals`. Страница доступна только аутентифицированным пользователям. ***Никнейм***: `edit_journal`.

Домашняя страница и страница информации

Для красивого интерфейса необходио реализовать 2 страницы с адресами:

- `/` домашняя страница. Содержит на себе ссылки (большие кнопки по центру веб- страницы) на `books` и `journals`. Доступна для всех. ***Никнейм***: `home`.
- `/info/` станица с информацией про разработчика. Доступна для всех. Содержит значки (`icons`) 2-ух соц-сетей (`Facebook`, `twitter`) (пусть для простоты они ведут на страницу регистрации, но там могут быть запрятаны ссылки, например, на ваши аккаунты). А также иконка `github` с сылкой на ваше решение 😃

Часовой пояс приложения

Часовой пояс выбираем `Europe/Moscow`

Стилизация шаблонов и веб форма

Для стилизации веб-форм используем `django-crispy-forms` с адаптером под `bootstrap4` . Для стилизации общего фона возьмем за основу https://getbootstrap.com/docs/4.0/examples/album/

Тесты

В каждом приложении должны присутствовать тесты для проверки валидности рабочего функционала. Минимальный набор:

- Тесты доступа по прямой ссылке, например `/books/new/`
- Тесты доступа по никнейму `new_book`
- Отображение правильного шаблона
- Если используется база данных в приложении , то тесты валидного отображения представителей модели (тест контента)

Дополнительно

Данный пункт необязательный, но для тех, кто хочет ближе познакомиться с грамотной разработкой - внедрите `CI/CD` сервис в свой репозиторий (хороший пример `Travis CI`) ***Ссылка на туториал*** https://docs.travis-ci.com/user/languages/python/

Документация

Необходимо создать простейшую документацию для своего приложения. Оформить в виде `README.md` файла. Файл должен содержать информацию:

- как установить проект локально
- как подключить все зависимости
- какие `url` ссылки за что отвечают
- описание базового функционала всего сервиса (описание в целом)

Виртуальное окружение

Обязательно наличие `Pipfile` и `Pipfile.lock` со всеми зависимостями проекта

Решение

Решение разместить на `github` и прислать ссылку на репозиторий преподавателю по адресу `evlasov@specialist.ru`