Самостоятельная работа №2

Техническое задание : реализовать полный `CruD` функционал для небольшого маркетплейса.
Описание: ***пользователем*** сервиса будет выступать `Store` (магазин), каждый магазин должен уметь размещать товары на странице, обновлять и удалять их.

Функционал: реализовать простейший функционал на основе следующих моделей.

Аналог пользовательской модели

Вместо пользовательской модели в проекте используем модель `Store`:

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

class Store(AbstractUser):
    inn = models.TextField(null=True, blank=True) # ИНН юридического лица магазина
    year_open = models.PositiveIntegerField(null=True, blank=True) # Год открытия
магазина
```

Поле ***username*** считаем аналогом названия магазина, ***email*** - электронный адрес магазина, ***password*** - пароль для входа в аккаунт магазина (поля ***username***, ***email***, ***password*** прикрепляются при наследовании от `AbstractUser`).

Описание в панели администратора: в панели администратора информация про магазин должна выводиться в виде: `username`, `email`, `inn`, `year_open`.

Магазин (пользователь) умеет менять и сбрасывать пароль

Реализовать функционал изменения и сброса пароля для аккаунта магазина (`password_change`, `password_reset`).

Магазин (пользователь) уммет размещать, редактировать и удалять товары

Модель товара `Item`:

```
from django.db import models
from django.contrib.auth import get_user_model
from django.urls import reverse

# Create your models here.
class Item(models.Model):
    title = models.CharField(max_length=255) # Название товара
    price = models.FloatField() # Цена единицы товара
    amount = models.PositiveIntegerField() # Количество единиц товара на складе
магазина
    date = models.DateTimeField(auto_now_add=True) # Дата публикации товара в
сервисе
```

```
    owner = models.ForeignKey(get_user_model(), on_delete=models.CASCADE) #
Магазин-собственник товара


    def __str__(self):
        return self.title


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

Время определяется по МСК : часовой пояс `Europe/Moscow`

Реализовать функционал ***CrUD*** для оборота товаров:

```
/items/ - доступ к списку всех товаров в сервисе (все товары всех магазинов,
список)
/items/new/ - создание нового товара (LoginRequired ссылка)
/items/<int:pk>/detail/ - детальный обзор товара (LoginRequired ссылка)
/items/<int:pk>/edit/ - редактирование товара (LoginRequired ссылка)
/items/<int:pk>/delete/ - удаление товара (LoginRequired ссылка)
```

Отображение товара

Отображение товара на веб-страницах (как на общем, так и на детальном рассмотрении) должно включать в себя ***название товара*** , ***количество на складе***, ***дату и время публикации в сервисе***, ***собственника(магазин, выложивший товар)***, ***цену единицы товара***.

Редактирование товара

Редактировать можно только поля ***название товара*** , ***количество на складе***,***цену единицы товара***. Время публикации и собственника менять нельзя.

При создании товара

Собственником товара выбирается текущей залогиненный магазин.

Шаблоны

Реализация шаблонов остается на ваше усмотрение. Главное условие - все шаблоны механизмов `password_reset` и `password_change` не должны совпадать с стандартными шаблонами (в которых выводится панель администратора).

Решение прислать в чат преподавателю, в виде ссылки на `github` репозиторий с исходным кодом проекта.