СТиВПП 3 - Проект каталог. База данных

Выполненные работы

В рамках задания была реализована основная модель SSD с полями, описывающими характеристики товара:

- Артикул (sku)
- Производитель (brand)
- Модель (model)
- Ёмкость (capacity_gb)
- Интерфейс (interface)
- Форм-фактор (form_factor)
- Скорость чтения и записи (read_speed, write_speed)
- Гарантия (warranty_years)
- Цена (price)
- Наличие на складе (in_stock)

Для полей interface и form_factor реализованы отдельные модели (Interface, FormFactor) со связью **один-ко-многим** (ForeignKey с on_delete=models.PROTECT), как требовалось по заданию.

Модель SSD содержит ограничения по типу данных и уникальности, например sku — уникальное поле, числовые характеристики реализованы через PositiveIntegerField и DecimalField.

Настройка административной панели

Для всех моделей была настроена админка:

- Отображение ключевых полей в списке
- Фильтрация по интерфейсу, форм-фактору и наличию
- Поиск по артикулу, бренду и модели
- Разбиение формы редактирования на логические блоки

Это упростило процесс добавления и редактирования записей через панель администратора.

Изменение представлений

Домашняя страница (HomeView) реализована на основе ListView, и отображает список SSD-дисков с фильтрацией по:

- Поисковому запросу
- Интерфейсу
- Диапазону цен

Представление для детального описания (SSDDetailView) отображает полную информацию о выбранном товаре.

В представлении реализована передача параметров запроса в шаблон для сохранения состояния фильтров при пагинации.

Чек-лист требований заказчика

Требование	Примечание
Создать модели для хранения информации о товарах	Модель SSD содержит все необходимые поля
Поля типа «один из» вынесены в отдельные таблицы	Реализованы модели Interface и FormFactor
Настроена административная панель Django	Все модели добавлены в админку с удобным интерфейсом
Представления используют данные из базы	Используются ListView и DetailView, реализована фильтрация
Прописаны ограничения по типам и допустимым значениям полей	Все поля строго типизированы
Связи один-ко-многим реализованы	Поля interface и form_factor — внешние ключи
Обновлены шаблоны, при необходимости	Шаблоны адаптированы под использование связанных моделей

Изменение проекта

1. Модели (catalog/models.py)

Созданы SSD, Interface и FormFactor. Поля interface и form_factor заменены на ForeignKey для соответствия требованию об отдельной таблице и связи один-ко-многим. Добавлены ограничения на поля (уникальность, типы данных).

```
from django.db import models


class Interface(models.Model):
    name = models.CharField(max_length=20, unique=True,
verbose_name="Интерфейс")

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

```
class FormFactor(models.Model):
    name = models.CharField(max_length=20, unique=True, verbose_name="Форм-
фактор")

    def __str__(self):
        return self.name


class SSD(models.Model):
    sku = models.CharField(max_length=50, unique=True,
verbose_name="Артикул")
    brand = models.CharField(max_length=100, verbose_name="Производитель")
    model = models.CharField(max_length=100, verbose_name="Модель")
    capacity_gb = models.PositiveIntegerField(verbose_name="Ёмкость (ГБ)")
    interface = models.ForeignKey(Interface, on_delete=models.PROTECT,
verbose_name="Интерфейс")
    form_factor = models.ForeignKey(FormFactor, on_delete=models.PROTECT,
verbose_name="Форм-фактор")
    read_speed = models.PositiveIntegerField(verbose_name="Скорость чтения
(МБ/с)")
    write_speed = models.PositiveIntegerField(verbose_name="Скорость записи
(МБ/с)")
    warranty_years = models.PositiveIntegerField(null=True, blank=True,
verbose_name="Гарантия (лет)")
    price = models.DecimalField(max_digits=10, decimal_places=2,
verbose_name="Цена")
    in_stock = models.BooleanField(default=True, verbose_name="В наличии")

    def __str__(self):
        return f"{self.brand} {self.model}"
```

Изменения:

- Созданы модели *Interface* и *FormFactor* для полей «один из».

- Поля *interfac* и *form_factor* в модели SSD заменены на ForeignKey с on_delete=models.PROTECT, чтобы предотвратить удаление связанных записей.

- Ограничения на поля сохранены: unique=True для sku, PositiveIntegerField для числовых полей, DecimalField для цены.

2. Админка (catalog/admin.py)

Все модели зарегистрированы с настройками для удобного управления данными.

```
from django.contrib import admin

from .models import SSD, Interface, FormFactor
```

```python
@admin.register(Interface)
class InterfaceAdmin(admin.ModelAdmin):
    list_display = ('name',)


@admin.register(FormFactor)
class FormFactorAdmin(admin.ModelAdmin):
    list_display = ('name',)


@admin.register(SSD)
class SSDAdmin(admin.ModelAdmin):
    list_display = ('brand', 'model', 'sku', 'price', 'in_stock')
    list_filter = ('interface', 'form_factor', 'in_stock')
    search_fields = ('brand', 'model', 'sku')
    fieldsets = (
        ('Основное', {
            'fields': ('sku', 'brand', 'model', 'price')
        }),
        ('Характеристики', {
            'fields': ('capacity_gb', 'interface', 'form_factor')
        }),
        ('Дополнительно', {
            'fields': ('warranty_years', 'in_stock')
        }),
    )
```

Изменения:

- Добавлены InterfaceAdmin и FormFactorAdmin для управления интерфейсами и форм-факторами.

3. Представления (catalog/views.py)

Представления: HomeView и SSDDetailView обновлены для работы с базой данных. В HomeView исправлена фильтрация по interface с учетом новой структуры моделей.

```python
from django.db.models import Q
from django.views.generic import ListView, DetailView, TemplateView

from .models import SSD


class HomeView(ListView):
    model = SSD
    template_name = "catalog/home.html"
    context_object_name = "ssd_list"
    paginate_by = 48

    def get_queryset(self):
        queryset = super().get_queryset()
```

```
        self.params = self.request.GET.copy()

        search_query = self.params.get('search')
        interface = self.params.get('interface')
        min_price = self.params.get('min_price')
        max_price = self.params.get('max_price')

        if search_query:
            queryset = queryset.filter(
                Q(brand__icontains=search_query) |
                Q(model__icontains=search_query) |
                Q(sku__icontains=search_query)
            )
        if interface:
            queryset = queryset.filter(interface__name=interface)
        if min_price:
            queryset = queryset.filter(price__gte=min_price)
        if max_price:
            queryset = queryset.filter(price__lte=max_price)

        return queryset

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['params'] = self.params
        return context


class SSDDetailView(DetailView):
    model = SSD
    template_name = "catalog/detail.html"


class AboutView(TemplateView):
    template_name = "catalog/about.html"
```

Изменения:

- В HomeView фильтр по interface изменен на interface__name, так как теперь это поле ссылается на модель Interface.

4. Генерация тестовых данных (catalog/management/commands/seed_catalog.py)

Команда seed_catalog адаптирована для создания экземпляров Interface и FormFactor, а затем генерации SSD с ссылками на них.

```
from decimal import Decimal
from django.core.management.base import BaseCommand
from django_seed import Seed
from faker import Faker
from catalog.models import SSD, Interface, FormFactor
```

```
class Command(BaseCommand):
    help = "Генерирует тестовые данные для SSD"
    fake = Faker(['en_US'])

    def add_arguments(self, parser):
        parser.add_argument('--number', type=int, default=50,
help='Количество SSD для генерации')
        parser.add_argument('--clear', action='store_true', help='Очистить
базу перед генерацией')

    def handle(self, *args, **options):
        if options['clear']:
            SSD.objects.all().delete()
            Interface.objects.all().delete()
            FormFactor.objects.all().delete()
            self.stdout.write(self.style.SUCCESS("База данных очищена"))

        # Создаем интерфейсы
        interfaces = ['SATA', 'NVMe']
        for interface in interfaces:
            Interface.objects.get_or_create(name=interface)

        # Создаем форм-факторы
        form_factors = ['2.5"', 'M.2']
        for form_factor in form_factors:
            FormFactor.objects.get_or_create(name=form_factor)

        seeder = Seed.seeder()
        seeder.faker = self.fake

        brands = ["Samsung", "Western Digital", "Kingston", "Crucial",
"SanDisk", "Intel", "Seagate"]
        models = ["EVO", "Pro", "Blue", "Green", "Ultra", "Plus", "Extreme"]

        seeder.add_entity(SSD, options['number'], {
            'sku': lambda x:
f"SSD{seeder.faker.unique.random_number(digits=8)}",
            'brand': lambda x: seeder.faker.random_element(brands),
            'model': lambda x: f"{seeder.faker.random_element(models)}
{seeder.faker.random_element(['Pro', 'Plus', 'Ultra', 'X', 'Lite'])}"[:50],
            'capacity_gb': lambda x: seeder.faker.random_element([250, 500,
1000, 2000, 4000]),
            'interface': lambda x: Interface.objects.order_by('?').first(),
            'form_factor': lambda x:
FormFactor.objects.order_by('?').first(),
            'read_speed': lambda x: seeder.faker.random_int(500, 7000),
            'write_speed': lambda x: seeder.faker.random_int(400, 6000),
            'warranty_years': lambda x: seeder.faker.random_int(1, 5),
            'price': lambda x: Decimal(seeder.faker.random_int(2000, 30000)
+ Decimal(seeder.faker.random_int(0, 99)) / 100),
            'in_stock': lambda x:
seeder.faker.boolean(chance_of_getting_true=75),
        })
```

```
        seeder.execute()
        self.stdout.write(self.style.SUCCESS(f"Успешно создано
{options['number']} SSD-дисков"))
```

Изменения:

- Добавлено создание объектов Interface и FormFactor перед генерацией SSD.

- Поля interface и form_factor теперь ссылаются на случайные экземпляры из соответствующих моделей.

5. Шаблоны

Шаблоны home.html и detail.html остаются функциональными, так как interface и form_factor автоматически преобразуются в строки благодаря методу __str__ в моделях Interface и FormFactor. Однако в home.html нужно обновить фильтр интерфейса, чтобы он соответствовал значениям name из модели Interface:

Остальные шаблоны (base.html, about.html) не требуют изменений.