Тестовое задание на примере каталога \ магазина товаров, простое, во многом минимальное по функционалу приложение, с базовой очевидной логикой, без требований и подводных камней.  
  
Цель продемонстрировать в ширину, охват понимания возможностей django, и стилистику написания кода.  
  
Модели  
- каталог (иерархическое дерево категорий)  
- товар связанный с категорией каталога, например с изображением и «спецификацией» (Filefield)  
- корзина \ заказ набор товаров, изменение статусов заказов, завершение  
- промо абстракция для объединения между собой товаров в “акцию”, с дополнительными условиями.  
  
Представления Class / Function Based View  
  
Бизнес логику вынести в модели, или в отдельные модули на уровне моделей. Минимизировать ее во view.    
  
Пара форм, с нестандартной инициализацией или валидацией.  
  
Написать миграцию данных, на «несуществующую проблему», и обратную миграцию.  
  
Добавить модели в админку, подобрать кастомные представления (в списке), фильтры, поиск, сортировку — по смыслу моделей. Например для заказа полезно будет кол-во товаров, для категории товары внутри, а для товара действующая акция / акции на товар.  
  
Далее как бонусы, в свободной форме для демонстрации возможностей, жирным выделил наиболее интересные с точки зрения django.  
  
1. Подготовить базовый деплой приложения, чтобы можно было запустить ее одной командой. (Мы используем pip + venv, плюсом будет docker compose)  
2. Для изображений предусмотреть хранение обрезанных пережатых вариантов для превью.  
3. Написать несколько юнит тестов (pytest), на самые базовые функции (например расчет суммы заказа \ корзины)  
**4. Продублировать функционал представлений в API, (django drf).**   
**5. Добавить пагинацию, сортировку, фильтры, добавить кастомные фильтры, «не из коробки», например категории только с товарами или только заказы с акционными товарами.**   
6. Добавить пару тестов для API  
7. В моделях, представлениях, сериализаторах использовать миксины для повторяющегося кода, например дата создания, флаг  пометки удаления вынести в миксины.  
**8. Написать кастомный менеджер моделей, например возвращающий товар с лучшей ценой в категории.**   
9. Попробовать сделать поиск на базе джанги, за который будет не стыдно (может быть, даже подключить elasticsearch)  
10. Переписать встроенные методы, например BaseModel.save() (  
**11. Написать импорт / синхронизацию даннных (например товары), из произвольной sql таблицы товаров**  
**12. Добавить команду для выполнения через manage.py, например синхронизации товаров.**   
13. Добавить команду синхронизации данных по api, например курса валют с открытого источника (не имеет значения, на самом деле какого).  
**14. Добавить сигналы, например, отправку писем при формировании заказа вынести в сигнал (фактическая отправка не нужна)**  
15. Добавить celery задание, на обновление курсов валют, чтобы фоном регулярно выполнялось.  
16. Написать асинхронный вызов, например формировать документа xls / pdf (можно и пустого), допустим карточки товара. или экспорта каталога в zip, повесить его на произвольный url.  
  
Тайп нотации, docstring, на собственный код приветствуются.  
  
Джанго кукикаттер приветствуется  
<https://github.com/pydanny/cookiecutter-django>  
  
Фронт на rest api будет большим плюсом для демонстрации работы  
  
+ linux bash, docker, git (рекомендую ознакомиться с концепцией github flow),  
+ а также в целом с современным окружением питона — среды, пакетные менеджеры, линтеры, форматтеры  
<https://mitelman.engineering/blog/python-best-practice/automating-python-best-practices-for-a-new-project/>  
в пайчарме, большинство функционала этого уже есть

Обязательная публикация изменений в git репозиторий, ссылку на который ожидаем увидеть в ответном письме.