# Проект: Платформа для обмена вещами (бартерная система)

**Описание задачи:** Разработать монолитное веб-приложение на Django для организации обмена вещами между пользователями. Пользователи смогут размещать объявления о товарах для обмена, просматривать чужие объявления и отправлять предложения на обмен. Приложение должно предоставлять удобный веб-интерфейс и, при необходимости, REST API для работы с объявлениями и обменными предложениями.

## Функциональные требования

### Создание объявления:

#### Входные данные:

* Пользователь (user\_id или данные авторизации через стандартную систему Django).
* Заголовок объявления (title).
* Описание товара (description).
* URL изображения (image\_url) – опционально.
* Категория товара (category).
* Состояние товара (например, «новый», «б/у»).
* **Автоматизация:**
* Генерация уникального идентификатора (id) для объявления (автоинкрементное поле).
* Автоматическая фиксация даты публикации (created\_at).
* **Вывод:** Подтверждение создания объявления с его данными.

1. **Редактирование объявления:**

* Возможность обновления полей (title, description, image\_url, category, condition).
* Ограничение: только автор объявления может редактировать запись.
* Обработка ошибок: уведомление, если объявление не найдено или пользователь не является автором.

1. **Удаление объявления:**

* Удаление объявления по его уникальному идентификатору.
* Корректная обработка ошибок (например, при отсутствии объявления с указанным id).

1. **Поиск и фильтрация объявлений:**

* Поиск по ключевым словам в заголовке и описании.
* Фильтрация по категории и состоянию товара.
* Реализация пагинации для возврата ограниченного числа результатов за один запрос.

1. **Обмен предложениями:**

* **Создание предложения обмена:**
* Пользователь отправляет предложение обмена, указывая:
* id объявления, инициирующего предложение (ad\_sender\_id).
* id объявления получателя (ad\_receiver\_id).
* Комментарий (comment).
* Автоматическая установка статуса предложения: «ожидает».
* **Обновление предложения:**
* Возможность изменения статуса предложения (например, «принята» или «отклонена»).
* **Просмотр предложений:**
* Фильтрация по отправителю, получателю или статусу.

1. **Отображение объявлений:**

* Веб-страница или API для получения списка всех объявлений с основными данными:
* id, user, title, description, image\_url, category, condition, created\_at.

## Технологический стек

* **Язык:** Python 3.8+
* **Веб-фреймворк:** Django 4+ (с использованием встроенной ORM)
* **Шаблонизация:** Django Templates для создания HTML-страниц
* **База данных:** SQLite или PostgreSQL (на выбор)
* **REST API:** Django REST Framework (опционально, для расширения функционала)
* **Документация:**
* README.md с инструкциями по установке, настройке и запуску проекта
* Автоматическая документация API (при использовании Django REST Framework)
* **Тестирование:** Unittest или Pytest для проверки ключевых функций приложения

## Структура проекта

* **apps:** Создать отдельное приложение (например, ads) для управления объявлениями и предложениями обмена.
* **models:** Определение моделей:
* **Ad:** Модель объявления со всеми необходимыми полями (id, user, title, description, image\_url, category, condition, created\_at).
* **ExchangeProposal:** Модель для предложений обмена (id, ad\_sender, ad\_receiver, comment, status, created\_at).
* **views:** Реализация представлений для CRUD-операций, поиска, фильтрации и обработки обменных предложений.
* **forms:** (При использовании HTML-интерфейса) формы для создания/редактирования объявлений и предложений.
* **urls:** Маршрутизация запросов к соответствующим представлениям.
* **templates:** Шаблоны Django для отображения веб-страниц.
* **tests:** Модуль с тестами для проверки основных функций (создание, редактирование, удаление, поиск).

## Алгоритм реализации и анализ

### Инициализация проекта:

* Создать новый проект Django и приложение (например, ads).
* Настроить виртуальное окружение и установить зависимости (Django, Django REST Framework — если используется, и необходимые библиотеки для работы с БД).

#### Моделирование данных (models.py):

##### Модель Ad:

* Поля: id (PK, автоинкремент), user (ForeignKey к встроенной модели User), title, description, image\_url, category, condition, created\_at (auto\_now\_add).

##### Модель ExchangeProposal:

* Поля: id, ad\_sender (ForeignKey к Ad), ad\_receiver (ForeignKey к Ad), comment, status (ChoiceField: «ожидает», «принята», «отклонена»), created\_at.

#### Создание форм и сериализаторов:

* Для HTML-интерфейса – реализовать формы на основе Django Forms для создания и редактирования объявлений.
* При использовании Django REST Framework – создать сериализаторы (AdSerializer, ProposalSerializer) для валидации входящих/исходящих данных.

#### Реализация представлений (views.py):

##### Создание объявления:

* Обработчик формы (или API-представление) принимает данные, валидирует их, создаёт объект Ad, устанавливает created\_at автоматически.

##### Редактирование объявления:

* Представление для обновления данных объявления с проверкой авторства.

##### Удаление объявления:

* Обработчик удаления записи по id с проверкой существования.

##### Поиск и фильтрация:

* Представление (или API эндпоинт) для поиска по ключевым словам в title/description, фильтрации по category и condition, с реализацией пагинации.

#### Работа с предложениями обмена:

* Создание, обновление статуса и просмотр предложений (фильтрация по отправителю, получателю или статусу).

### Маршрутизация (urls.py):

* Настроить URL-маршруты для каждого представления (создание, редактирование, удаление, поиск объявлений и работа с предложениями).

## Тестирование и документация:

* Написать тесты для проверки ключевых функций (создание, редактирование, удаление, поиск объявлений и обработка предложений) с использованием встроенной системы тестирования Django или Pytest.
* Подготовить README.md с подробными инструкциями по установке, миграциям, запуску сервера и запуску тестов.