## Стратегия разработки "Tornado Todd" за 5 часов

**Главное правило:** Не стремитесь к идеалу. Цель — создать работающий каркас с 1-2 ключевыми функциями, который можно будет протестировать.

---

## Фаза 1: Подготовка бури (0:00 - 1:00)

**Цель:** Подготовить рабочее окружение и каркас проекта.

- **Шаг 1 (15 мин):** Установка Django и создание проекта.
  - `pip install django`
  - `django-admin startproject tornadotodd .` (точка важна!)
  - `python manage.py startapp todo`
  - Добавить `'todo'` в `INSTALLED_APPS` в `settings.py`.
- **Шаг 2 (30 мин):** Создание и применение базовой модели.
  - В `todo/models.py` создать модель `Task` всего с тремя полями: `user` (ForeignKey), `title` (CharField), `created_date` (DateTimeField(auto_now_add=True)).
  - `python manage.py makemigrations`
  - `python manage.py migrate`
- **Шаг 3 (15 мин):** Настройка суперпользователя и первый запуск.
  - `python manage.py createsuperuser` (логин/пароль: admin/admin)
  - `python manage.py runserver`
  - **Результат фазы:** Проверить, что сервер запускается и вы можете зайти в админку (`/admin`).

💡 **Критерий успеха фазы 1:** Админка Django открывается, и вы видите стандартную страницу.

---

## Фаза 2: Вихрь функций (1:00 - 2:00)

**Цель:** Реализовать core-логику: создание и отображение задач.

- **Шаг 1 (30 мин):** Создаем самые простые представления (Views).
  - В `todo/views.py` создать два view:
    1. `task_list`: просто выводит `Task.objects.all()`.
    2. `add_task`: обрабатывает простую POST-форму для добавления новой задачи.
- **Шаг 2 (30 мин):** Прописываем базовые URL-маршруты.

    - В `urls.py` проекта подключить URL-ы приложения `todo`.
    - В `todo/urls.py` прописать два пути: `''` (список задач) и `'add/'` (добавление задачи).
- **Результат фазы:** По адресу `http://127.0.0.1:8000/` должна открываться пустая страница (задач ещё нет), а в админке можно создать задачу и она должна там отобразиться.

💡 **Критерий успеха фазы 2:** Страница со списком задач не выдаёт ошибку 404 или 500.

---

### Фаза 3: Глаз бури (Интерфейс) (2:00 - 3:00)

**Цель:** Создать минималистичные HTML-шаблоны, чтобы было с чем взаимодействовать.

- **Шаг 1 (45 мин):** Создание базовых шаблонов.
    - Создать папку `templates/todo/`.
    - Создать `base.html` с основной разметкой HTML5. Подключить Bootstrap CSS по CDN (это займёт 1 строку кода и даст хороший вид).
    - Создать `task_list.html`, который наследует от `base.html` и в цикле выводит список задач (пока без возможности отметить как выполненную или удалить).
    - Добавить на страницу простейшую форму (`<form method="POST">`) для добавления новой задачи.
- **Шаг 2 (15 мин):** Интеграция views с шаблонами.
    - Вернуть рендеринг шаблонов из ваших view-функций.
- **Результат фазы:** Вы можете через простую форму на странице добавить новую задачу, и она моментально появляется в списке ниже. **Это момент истины!**

💡 **Критерий успеха фазы 3:** Функционал "Добавить задачу" работает через браузер без использования админки.

---

### Фаза 4: Приземление данных (3:00 - 4:00)

**Цель:** Привязать задачи к конкретному пользователю и добавить выход из системы.

- **Шаг 1 (30 мин):** Включить систему аутентификации Django.
    - В `settings.py` добавить `LOGIN_REDIRECT_URL = '/'` и `LOGOUT_REDIRECT_URL = '/'`.
    - В главный `urls.py` добавить `path('accounts/', include('django.contrib.auth.urls'))`.
- **Шаг 2 (30 мин):** Фильтрация задач по пользователю.

    - Модифицировать view `task_list`: вместо `Task.objects.all()` использовать `Task.objects.filter(user=request.user)`.
    - В view `add_task` при сохранении новой задачи привязывать её к текущему пользователю: `new_task.user = request.user`.
- **Результат фазы:** Вы регистрируете двух пользователей через админку, заходите под каждым и видите, что их списки задач изолированы друг от друга.

💡 **Критерий успеха фазы 4:** Задачи, созданные одним пользователем, не видны другому.

---

### Фаза 5: Развёртывание и документирование (4:00 - 5:00)

**Цель:** Закоммитить всё в GitHub и написать README.

- **Шаг 1 (15 мин):** Создать `.gitignore` для Python/Django и инициализировать Git-репозиторий.
- **Шаг 2 (15 мин):** Создать файл `requirements.txt` (`pip freeze > requirements.txt`).
- **Шаг 3 (20 мин):** Комиты и пуш в GitHub.
    - `git add .`
    - `git commit -m "Минимально рабочая версия Tornado Todd"`
    - Создать репозиторий на GitHub и запушить код.
- **Шаг 4 (10 мин):** Скопировать и слегка адаптировать текст для `README.md` (который мы придумали ранее), чтобы он соответствовал тому, что вы сделали на самом деле.

💡 **Критерий успеха фазы 5:** Код находится в публичном репозитории на GitHub с вменяемым README.

---

**Итог:** Через 5 часов у вас будет работающий прототип, полностью готовый к этапу тестирования из вашего задания. Вы сможете тестировать:

1. Функциональность (добавление задачи).
2. Надёжность (что будет, если отправить пустую форму?).
3. Безопасность (попытка доступа к данным другого пользователя через URL).
4. Удобство использования.

Удачи! Этот план амбициозный, но выполнимый. 🌀