

Помоги разработать проект, реализующий Функционал

- 1.Авторизация/Регистрация
- 2.Поиск рецептов (по ключевым словам, по хештегам, по калорийности)
- 3.Лента рекомендаций (через хештеги)
- 4.Просмотр рецептов (ингредиенты, инструкция, фото/видео, калорийность и хештеги)
- 5.Избранное(добавление, удаление)
- 6.Добавление рецептов как пользователь, редактирование, удаление
- 7.Фильтрация (тип блюда, ингредиенты)
- 8.Статьи
- 9.Профиль пользователя (личная инфа, список избранного(по желанию отображения), добавленные рецепты)
- 10.Добавить к рецептам при осмотре ленты краткое описание
- 11.Комментарии пользователей к рецептам

Технологии

□Python

□Django

□Для работы с БД PostgreSQL?

□HTML, CSS, JS Необходимо пошагово объяснить весь процесс разработки (можно поэтапно), объяснить как подключить БД к проекту с учётом того, что в БД должны храниться рецепты, содержащие ингредиенты, шаги готовки сопровождаемые фото, некоторые рецепты будут с видео, также у рецептов должно быть количество порций, ккал на 100 грамм продукта, время приготовления и сложность приготовления.

Thought for 119 seconds ▼

⌚ Мы создадим проект "Кулинарный сайт" с использованием Django и PostgreSQL. Проект будет включать в себя все указанные функции.

План разработки:

1. Настройка проекта и окружения
2. Создание моделей базы данных
3. Настройка административной панели
4. Реализация аутентификации и авторизации
5. Создание представлений (views) и шаблонов (templates) для основных функций
6. Реализация дополнительных функций (поиск, фильтрация, лента рекомендаций, избранное, комментарии)

7. Работа со статическими файлами (фото, видео)

8. Тестирование

Давайте начнем с первого этапа.

### Этап 1: Настройка проекта и окружения

1. Создайте виртуальное окружение и активируйте его.

2. Установите Django и другие необходимые пакеты (psycopg2 для работы с PostgreSQL, pillow для работы с изображениями).

3. Создайте проект Django и приложение (например, `recipes` ).

Команды:

`bash`

```
python -m venv venv
source venv/bin/activate # для Windows: venv\Scripts\activate
pip install Django psycopg2-binary pillow
django-admin startproject culinary_project
cd culinary_project
python manage.py startapp recipes
```

4. Добавьте приложение `recipes` в `INSTALLED_APPS` в `settings.py` .

5. Настройте базу данных PostgreSQL в `settings.py` :

`python`

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'your_db_name',
        'USER': 'your_db_user',
        'PASSWORD': 'your_db_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

6. Выполните миграции: `python manage.py migrate`

### Этап 2: Создание моделей базы данных

Мы должны создать модели для:

- Пользователя (можно использовать встроенную модель User, но расширить ее профилем)
- Рецептов

- Ингредиентов (связаны с рецептами через промежуточную модель, так как у одного рецепта несколько ингредиентов и у одного ингредиента может быть несколько рецептов)
- Шагов приготовления (отдельная модель, связанная с рецептом)
- Хештегов (многие ко многим с рецептами)
- Статей (могут быть без рецепта, но возможно с хештегами)
- Избранного (пользователь и рецепт)
- Комментариев (пользователь, рецепт, текст, дата)

Опишем модели в `recipes/models.py` :

```
python

from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MinValueValidator, MaxValueValidator

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(max_length=500, blank=True)
    location = models.CharField(max_length=30, blank=True)
    birth_date = models.DateField(null=True, blank=True)
    profile_photo = models.ImageField(upload_to='profile_photos/', null=True, blank=True)

    def __str__(self):
        return self.user.username

class Hashtag(models.Model):
    name = models.CharField(max_length=50, unique=True)

    def __str__(self):
        return self.name

class Recipe(models.Model):
    DIFFICULTY_LEVELS = [
        ('easy', 'Легкий'),
        ('medium', 'Средний'),
        ('hard', 'Сложный'),
    ]

    title = models.CharField(max_length=200)
    description = models.TextField(blank=True) # краткое описание для ленты
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    cooking_time = models.PositiveIntegerField(help_text="Время приготовления в минутах")
    servings = models.PositiveIntegerField(help_text="Количество порций")
```

```
calories_per_100g = models.PositiveIntegerField(help_text="Калории на 100 г  
рамм")  
difficulty = models.CharField(max_length=10, choices=DIFFICULTY_LEVELS)  
created_at = models.DateTimeField(auto_now_add=True)  
updated_at = models.DateTimeField(auto_now=True)  
main_photo = models.ImageField(upload_to='recipes/main_photos/')  
video = models.FileField(upload_to='recipes/videos/', null=True, blank=True)  
  
hashtags = models.ManyToManyField(Hashtag, blank=True)  
  
def __str__(self):  
    return self.title  
  
class Ingredient(models.Model):  
    name = models.CharField(max_length=100)  
  
    def __str__(self):  
        return self.name  
  
class RecipeIngredient(models.Model):  
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)  
    ingredient = models.ForeignKey(Ingredient, on_delete=models.CASCADE)  
    quantity = models.CharField(max_length=50) # Например, "2 шт", "100 г"  
  
    def __str__(self):  
        return f'{self.ingredient.name} - {self.quantity}'  
  
class CookingStep(models.Model):  
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE, related_name  
='cooking_steps')  
    step_number = models.PositiveIntegerField()  
    description = models.TextField()  
    photo = models.ImageField(upload_to='recipes/steps_photos/', blank=True, nu  
ll=True)  
  
    class Meta:  
        ordering = ['step_number']  
  
    def __str__(self):  
        return f'Step {self.step_number} for {self.recipe.title}'  
  
class Article(models.Model):  
    title = models.CharField(max_length=200)  
    content = models.TextField()  
    author = models.ForeignKey(User, on_delete=models.CASCADE)  
    published_at = models.DateTimeField(auto_now_add=True)  
    hashtags = models.ManyToManyField(Hashtag, blank=True)
```

```

def __str__(self):
    return self.title


class Favorite(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
    added_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ('user', 'recipe')

    def __str__(self):
        return f'{self.user.username} - {self.recipe.title}'


class Comment(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE, related_name='comments')
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'Comment by {self.user.username} on {self.recipe.title}'

```

После создания моделей выполните миграции:

bash

```

python manage.py makemigrations
python manage.py migrate

```

**Этап 3: Настройка административной панели**

Зарегистрируем модели в admin.py :

python

```

from django.contrib import admin
from .models import *

admin.site.register(Profile)
admin.site.register(Hashtag)
admin.site.register(Recipe)
admin.site.register(Ingredient)
admin.site.register(RecipeIngredient)
admin.site.register(CookingStep)

```

```
admin.site.register(Article)
admin.site.register(Favorite)
admin.site.register(Comment)
```

#### Этап 4: Реализация аутентификации и авторизации

Используем встроенные представления Django для аутентификации. Нам понадобятся:

- Регистрация (создадим свою форму и представление)
- Вход
- Выход
- Изменение пароля

Для регистрации создадим форму в `forms.py`:

```
python
```

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
```

Создадим представления для регистрации и профиля в `views.py`:

```
python
```

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Аккаунт создан для {username}!')
            return redirect('login')
    else:
        form = UserRegisterForm()
```

```
    return render(request, 'recipes/register.html', {'form': form})\n\n@login_required\ndef profile(request):\n    return render(request, 'recipes/profile.html')
```

Создадим шаблоны для регистрации и профиля.

## Этап 5: Создание представлений и шаблонов для основных функций

Нам понадобятся представления для:

- Главной страницы (лента рекомендаций)
- Поиска и фильтрации рецептов
- Просмотра рецепта
- Добавления/редактирования/удаления рецепта
- Статей
- Избранного
- Комментариев

Опишем некоторые из них:

### Главная страница (лента рекомендаций)

В `views.py`:

```
python\n\nfrom django.shortcuts import render\nfrom .models import Recipe\n\n\ndef home(request):\n    # Пока просто все рецепты, потом можно добавить рекомендации через хештеги\n    recipes = Recipe.objects.all().order_by('-created_at')\n    return render(request, 'recipes/home.html', {'recipes': recipes})
```

### Просмотр рецепта

```
python\n\nfrom django.shortcuts import get_object_or_404\n\n\ndef recipe_detail(request, pk):\n    recipe = get_object_or_404(Recipe, pk=pk)\n    return render(request, 'recipes/recipe_detail.html', {'recipe': recipe})
```

### Добавление рецепта

```
python

from .forms import RecipeForm

@login_required
def add_recipe(request):
    if request.method == 'POST':
        form = RecipeForm(request.POST, request.FILES)
        if form.is_valid():
            recipe = form.save(commit=False)
            recipe.author = request.user
            recipe.save()
            form.save_m2m() # для сохранения many-to-many (хештеги)
            return redirect('recipe_detail', pk=recipe.pk)
    else:
        form = RecipeForm()
    return render(request, 'recipes/recipe_form.html', {'form': form})
```

Аналогично нужно создать формы для моделей (в `forms.py`). Например, для рецепта:

```
python

from django import forms
from .models import Recipe

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo', 'video', 'hashtags']
```

## Этап 6: Реализация дополнительных функций

### Поиск и фильтрация

В представлении для поиска можно фильтровать по ключевым словам (в названии, описании, ингредиентах), по хештегам, по калорийности.

```
python

def search_recipes(request):
    query = request.GET.get('q')
    hashtag = request.GET.get('hashtag')
    max_calories = request.GET.get('max_calories')
    recipes = Recipe.objects.all()

    if query:
```

```
recipes = recipes.filter(
    models.Q(title__icontains=query) |
    models.Q(description__icontains=query) |
    models.Q(ingredients__name__icontains=query) # если ищем по ингредиентам, то нужно связаться через RecipeIngredient
).distinct()

if hashtag:
    recipes = recipes.filter(hashtags__name=hashtag)

if max_calories:
    recipes = recipes.filter(calories_per_100g__lte=max_calories)

return render(request, 'recipes/search_results.html', {'recipes': recipes})
```

## Избранное

python

```
@login_required
def add_to_favorites(request, recipe_id):
    recipe = get_object_or_404(Recipe, id=recipe_id)
    favorite, created = Favorite.objects.get_or_create(user=request.user, recipe=recipe)
    if not created:
        # Уже было в избранном, можно удалить или оставить, в зависимости от логики
        favorite.delete()
    return redirect('recipe_detail', pk=recipe_id)
```

## Комментарии

python

```
@login_required
def add_comment(request, recipe_id):
    recipe = get_object_or_404(Recipe, id=recipe_id)
    if request.method == 'POST':
        text = request.POST.get('text')
        Comment.objects.create(user=request.user, recipe=recipe, text=text)
    return redirect('recipe_detail', pk=recipe_id)
```

## Этап 7: Работа со статическими файлами

В settings.py укажите:

python

```
MEDIA_URL = '/media/'  
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

В urls.py проекта добавьте:

```
python  
  
from django.conf import settings  
from django.conf.urls.static import static  
  
urlpatterns = [  
    # ... ваши url ...  
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Этап 8: Тестирование

Протестируйте все функции вручную или напишите тесты.

Это общий план. Каждый пункт можно детализировать и разбить на подпункты. В зависимости от требований, некоторые функции могут быть реализованы по-разному (например, рекомендации через хештеги можно сделать, анализируя предпочтения пользователя).

Удачи в разработке!

Отличный проект! Давайте разобьем его на этапы и подробно разберем каждый шаг.

## Этап 1: Подготовка и настройка проекта

### 1.1 Установка необходимых компонентов

```
bash  
  
# Создание виртуального окружения  
python -m venv venv  
source venv/bin/activate # Linux/Mac  
# venv\Scripts\activate # Windows  
  
# Установка пакетов  
pip install django  
pip install psycopg2-binary  
pip install pillow  
pip install django-crispy-forms
```

### 1.2 Создание проекта и приложений

```
bash
```

```
django-admin startproject culinary_project
cd culinary_project
python manage.py startapp recipes
python manage.py startapp users
python manage.py startapp comments
```

## 1.3 Настройка Django проекта

culinary\_project/settings.py:

```
python

import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'your-secret-key-here'

DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'recipes',
    'users',
    'comments',
    'crispy_forms',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
]

ROOT_URLCONF = 'culinary_project.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
],
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'culinary_db',
        'USER': 'your_username',
        'PASSWORD': 'your_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

CRISPY_TEMPLATE_PACK = 'bootstrap4'

LOGIN_REDIRECT_URL = 'home'
LOGIN_URL = 'login'

# Настройки для отправки email (если нужна регистрация по email)
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

## Этап 2: Проектирование моделей базы данных

### 2.1 Модель пользователя (users/models.py)

python

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(max_length=500, blank=True)
    location = models.CharField(max_length=30, blank=True)
    birth_date = models.DateField(null=True, blank=True)
    profile_photo = models.ImageField(upload_to='profile_photos/', null=True, blank=True)
    show_favorites = models.BooleanField(default=True)

    def __str__(self):
        return self.user.username

@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()
```

### 2.2 Модели для рецептов (recipes/models.py)

python

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

class Hashtag(models.Model):
    name = models.CharField(max_length=50, unique=True)

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

```
class Recipe(models.Model):
    DIFFICULTY_LEVELS = [
        ('easy', 'Легкий'),
        ('medium', 'Средний'),
        ('hard', 'Сложный'),
    ]

    title = models.CharField(max_length=200)
    description = models.TextField() # Краткое описание для ленты
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    cooking_time = models.PositiveIntegerField(help_text="Время приготовления в минутах")
    servings = models.PositiveIntegerField(help_text="Количество порций")
    calories_per_100g = models.PositiveIntegerField(help_text="Калории на 100 грамм")
    difficulty = models.CharField(max_length=10, choices=DIFFICULTY_LEVELS)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    main_photo = models.ImageField(upload_to='recipes/main_photos/')
    video = models.FileField(upload_to='recipes/videos/', null=True, blank=True)
    hashtags = models.ManyToManyField(Hashtag, blank=True)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('recipe-detail', kwargs={'pk': self.pk})

class Ingredient(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='ingredients', on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    quantity = models.CharField(max_length=50) # Например: "2 шт", "100 г", "по вкусу"

    def __str__(self):
        return f'{self.name} - {self.quantity}'

class CookingStep(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='cooking_steps', on_delete=models.CASCADE)
    step_number = models.PositiveIntegerField()
    description = models.TextField()
    photo = models.ImageField(upload_to='recipes/steps_photos/', blank=True, null=True)
```

```

class Meta:
    ordering = ['step_number']

def __str__(self):
    return f"Step {self.step_number} for {self.recipe.title}"

class Favorite(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
    added_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ('user', 'recipe')

    def __str__(self):
        return f"{self.user.username} - {self.recipe.title}"

```

## 2.3 Модели для статей и комментариев (comments/models.py)

python

```

from django.db import models
from django.contrib.auth.models import User
from recipes.models import Recipe

class Article(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    published_at = models.DateTimeField(auto_now_add=True)
    hashtags = models.ManyToManyField('recipes.Hashtag', blank=True)

    def __str__(self):
        return self.title

class Comment(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='comments', on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ['-created_at']

```

```
def __str__(self):
    return f"Comment by {self.author.username} on {self.recipe.title}"
```

## Этап 3: Настройка админ-панели

### 3.1 Регистрация моделей в админке

recipes/admin.py:

```
python

from django.contrib import admin
from .models import Recipe, Ingredient, CookingStep, Hashtag, Favorite

class IngredientInline(admin.TabularInline):
    model = Ingredient
    extra = 1

class CookingStepInline(admin.TabularInline):
    model = CookingStep
    extra = 1

@admin.register(Recipe)
class RecipeAdmin(admin.ModelAdmin):
    list_display = ['title', 'author', 'cooking_time', 'difficulty', 'created_at']
    list_filter = ['difficulty', 'hashtags', 'created_at']
    search_fields = ['title', 'description']
    inlines = [IngredientInline, CookingStepInline]

admin.site.register(Hashtag)
admin.site.register(Favorite)
```

comments/admin.py:

```
python

from django.contrib import admin
from .models import Article, Comment

admin.site.register(Article)
admin.site.register(Comment)
```

users/admin.py:

```
python

from django.contrib import admin
from .models import Profile

admin.site.register(Profile)
```

## Этап 4: Создание представлений (Views)

### 4.1 Представления для рецептов (recipes/views.py)

```
python

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag
from .forms import RecipeForm

class RecipeListView(ListView):
    model = Recipe
    template_name = 'recipes/home.html'
    context_object_name = 'recipes'
    paginate_by = 9

    def get_queryset(self):
        queryset = Recipe.objects.all().order_by('-created_at')

        # Поиск по ключевым словам
        query = self.request.GET.get('q')
        if query:
            queryset = queryset.filter(
                Q(title__icontains=query) |
                Q(description__icontains=query) |
                Q(ingredients__name__icontains=query)
            ).distinct()

        # Фильтрация по хештегам
        hashtag = self.request.GET.get('hashtag')
        if hashtag:
            queryset = queryset.filter(hashtags__name=hashtag)
```

```
# Фильтрация по калорийности
max_calories = self.request.GET.get('max_calories')
if max_calories:
    queryset = queryset.filter(calories_per_100g__lte=max_calories)

return queryset

class RecipeDetailView(DetailView):
    model = Recipe
    template_name = 'recipes/recipe_detail.html'

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

class RecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
            return redirect('recipe-detail', pk=obj.pk)
        return super().dispatch(request, *args, **kwargs)

class RecipeDeleteView(LoginRequiredMixin, DeleteView):
    model = Recipe
    template_name = 'recipes/recipe_confirm_delete.html'
    success_url = '/'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
            return redirect('recipe-detail', pk=obj.pk)
        return super().dispatch(request, *args, **kwargs)

@login_required
def add_to_favorites(request, pk):
    recipe = get_object_or_404(Recipe, pk=pk)
    favorite, created = Favorite.objects.get_or_create(user=request.user, recipe=recipe)
```

```

if not created:
    favorite.delete()
return redirect('recipe-detail', pk=pk)

def search_recipes(request):
    query = request.GET.get('q', '')
    hashtag = request.GET.get('hashtag', '')
    max_calories = request.GET.get('max_calories', '')

    recipes = Recipe.objects.all()

    if query:
        recipes = recipes.filter(
            Q(title__icontains=query) |
            Q(description__icontains=query) |
            Q(ingredients__name__icontains=query)
        ).distinct()

    if hashtag:
        recipes = recipes.filter(hashtags__name=hashtag)

    if max_calories:
        recipes = recipes.filter(calories_per_100g__lte=int(max_calories))

    context = {
        'recipes': recipes,
        'query': query,
        'hashtag': hashtag,
        'max_calories': max_calories,
    }

    return render(request, 'recipes/search_results.html', context)

```

## Этап 5: Создание форм

### 5.1 Формы для рецептов (recipes/forms.py)

python

```

from django import forms
from .models import Recipe, Ingredient, CookingStep

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe

```

```

        fields = [
            'title', 'description', 'cooking_time', 'servings',
            'calories_per_100g', 'difficulty', 'main_photo',
            'video', 'hashtags'
        ]
        widgets = {
            'description': forms.Textarea(attrs={'rows': 3}),
            'hashtags': forms.SelectMultiple(attrs={'class': 'form-control'}),
        }

class IngredientForm(forms.ModelForm):
    class Meta:
        model = Ingredient
        fields = ['name', 'quantity']

class CookingStepForm(forms.ModelForm):
    class Meta:
        model = CookingStep
        fields = ['step_number', 'description', 'photo']

```

## Этап 6: Настройка URL-маршрутов

### 6.1 Главные URLs (culinary\_project/urls.py)

python

```

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from django.contrib.auth import views as auth_views
from users import views as user_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    path('users/', include('users.urls')),
    path('comments/', include('comments.urls')),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'),
         name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'),
         name='logout'),
]

```

```
if settings.DEBUG:  
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## 6.2 URLs для рецептов (recipes/urls.py)

python

```
from django.urls import path  
from . import views  
  
urlpatterns = [  
    path('', views.RecipeListView.as_view(), name='home'),  
    path('recipe/<int:pk>/', views.RecipeDetailView.as_view(), name='recipe-detail'),  
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),  
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recipe-update'),  
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recipe-delete'),  
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favorites'),  
    path('search/', views.search_recipes, name='search-recipes'),  
]
```

## Этап 7: Создание базовых шаблонов

### 7.1 Базовый шаблон (templates/base.html)

html

```
<!DOCTYPE html>  
<html lang="ru">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>  
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">  
    <style>  
        .recipe-card {  
            transition: transform 0.2s;  
        }  
        .recipe-card:hover {  
            transform: translateY(-5px);  
        }
```

```

        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="{% url 'home' %}">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                {% if user.is_authenticated %}
                    <a class="nav-link" href="{% url 'recipe-create' %}">Добавить
                    рецепт</a>
                    <a class="nav-link" href="{% url 'profile' %}">Профиль</a>
                    <a class="nav-link" href="{% url 'logout' %}">Выйти</a>
                {% else %}
                    <a class="nav-link" href="{% url 'login' %}">Войти</a>
                    <a class="nav-link" href="{% url 'register' %}">Регистрация</
                a>
                {% endif %}
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-{{ message.tags }}">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}

        {% block content %}{% endblock %}
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## Этап 8: Настройка PostgreSQL

### 8.1 Установка и настройка PostgreSQL

bash

```
# Установка PostgreSQL (Ubuntu)
sudo apt update
sudo apt install postgresql postgresql-contrib

# Создание базы данных и пользователя
sudo -u postgres psql

# В psql:
CREATE DATABASE culinary_db;
CREATE USER culinary_user WITH PASSWORD 'your_password';
ALTER ROLE culinary_user SET client_encoding TO 'utf8';
ALTER ROLE culinary_user SET default_transaction_isolation TO 'read committed';
ALTER ROLE culinary_user SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE culinary_db TO culinary_user;
\q
```

## 8.2 Выполнение миграций

bash

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

## Этап 9: Запуск и тестирование

bash

```
python manage.py runserver
```

## Дальнейшие шаги:

1. Создание остальных шаблонов для всех представлений
2. Реализация системы рекомендаций на основе хештегов
3. Добавление пагинации для списков
4. Реализация AJAX для добавления в избранное и комментариев
5. Оптимизация запросов к базе данных
6. Добавление кэширования
7. Написание тестов

## 8. Настройка production-окружения

Это основа вашего проекта. Каждый блок можно дальше детализировать и улучшать! Хотите, чтобы я подробнее раскрыл какой-то конкретный этап?

```
Всё ли хорошо? (.venv) PS
D:\RecipesAlmanah\recipesAlmanah_project> python manage.py
makemigrations
No changes detected
(.venv) PS D:\RecipesAlmanah\recipesAlmanah_project> python
manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.
```

Thought for 43 seconds ▾

💡 Мы видим, что при запуске `makemigrations` не обнаружено изменений, и при `migrate` не применяются миграции. Это означает, что мы еще не создали миграции для наших моделей.

Давайте разберемся, почему так произошло:

1. Возможно, мы не правильно зарегистрировали наши приложения в `INSTALLED_APPS` в `settings.py`.
2. Возможно, мы не создали модели в `models.py` или не сохранили файлы.

Проверьте, что вы добавили `recipes`, `users`, `comments` в `INSTALLED_APPS` в `settings.py`.

Также убедитесь, что вы создали модели в соответствующих файлах `models.py` и сохранили их.

После этого попробуйте снова выполнить `makemigrations` для каждого приложения или общий:

```
bash

python manage.py makemigrations recipes
python manage.py makemigrations users
python manage.py makemigrations comments
```

Если вы не видите изменений, попробуйте удалить файлы миграций (если они есть) и заново создать миграции. Также можно попробовать запустить `makemigrations` без указания приложения, чтобы Django проверил все приложения.

Если вы только что создали модели, то должны увидеть, что миграции создаются.

Если вы уже создавали миграции и затем изменили модели, то нужно снова запустить `makemigrations`.

Если вы работаете в Windows, убедитесь, что вы активировали виртуальное окружение и находитесь в правильной директории.

Если проблема остается, давайте проверим, видит ли Django наши модели. Запустите shell:

```
bash
```

```
python manage.py shell
```

И попробуйте импортировать модель из приложения, например:

```
python
```

```
from recipes.models import Recipe
```

Если ошибок нет, то модели есть. Если есть ошибка, то возможно приложение не зарегистрировано или модель не определена.

Также проверьте, что вы сохранили все файлы.

Если вы уже создавали миграции и применяли их, а затем удалили их, то возможно, что в базе данных уже есть таблицы для этих моделей. В этом случае можно удалить базу данных (если это development и вы можете позволить себе потерять данные) и заново создать миграции.

Если вы не хотите удалять базу данных, то можно попробовать:

```
bash
```

```
python manage.py makemigrations --empty recipes
```

Но это временное решение, лучше разобраться, почему модели не обнаруживаются.

Давайте проверим пошагово:

1. Убедитесь, что в `settings.py` в `INSTALLED_APPS` добавлены:

- 'recipes',
- 'users',
- 'comments',

2. Убедитесь, что в каждом приложении в файле `models.py` есть модели (и они не закомментированы).

3. Убедитесь, что вы запускаете команды из правильной директории (где находится `manage.py`).

4. Попробуйте запустить `python manage.py check` — это проверит наличие ошибок в проекте.

5. Если вы ранее создавали миграции и затем удалили файлы миграций, то в базе данных есть запись о примененных миграциях, но файлов миграций нет. В этом случае нужно удалить записи из таблицы `django_migrations` для этих приложений или удалить всю таблицу, но это сложно и может привести к несоответствию. Лучше восстановить файлы миграций или создать заново.

Если вы только начинаете проект, проще удалить базу данных (файл `db.sqlite3`, если используете SQLite) и заново выполнить миграции. Для PostgreSQL нужно удалить базу и создать заново.

Если вы используете SQLite и хотите начать заново:

- Удалите файл `db.sqlite3`
- Удалите все папки `migrations` в приложениях (кроме `__init__.py`)
- Выполните `python manage.py makemigrations`
- Выполните `python manage.py migrate`

Если вы используете PostgreSQL, то нужно удалить базу данных и создать заново.

Если вы не хотите терять данные, то нужно разбираться с состоянием миграций.

Но из вашего вывода видно, что миграции для `admin`, `auth`, `contenttypes`, `sessions` уже применены, а для наших приложений нет. Значит, вы не создали миграции для своих приложений.

Попробуйте явно указать приложение при создании миграций:

`bash`

```
python manage.py makemigrations recipes
```

Если это не поможет, покажите содержимое одного из файлов `models.py` (например, `recipes/models.py`) и `settings.py`.

Проблема в том, что Django не видит ваши модели. Давайте разберемся по шагам:

## Шаг 1: Проверка настроек приложений

Убедитесь, что в `settings.py` добавлены все приложения:

`python`

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',
```

```
'django.contrib.messages',
'django.contrib.staticfiles',
'recipes', # ваше приложение
'users', # ваше приложение
'comments', # ваше приложение
'crispy_forms',
]
```

## Шаг 2: Проверка моделей

Убедитесь, что в каждом приложении есть файл `models.py` с моделями:

`recipes/models.py`:

```
python

from django.db import models
from django.contrib.auth.models import User

class Hashtag(models.Model):
    name = models.CharField(max_length=50, unique=True)

    def __str__(self):
        return self.name

class Recipe(models.Model):
    DIFFICULTY_LEVELS = [
        ('easy', 'Легкий'),
        ('medium', 'Средний'),
        ('hard', 'Сложный'),
    ]

    title = models.CharField(max_length=200)
    description = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    cooking_time = models.PositiveIntegerField()
    servings = models.PositiveIntegerField()
    calories_per_100g = models.PositiveIntegerField()
    difficulty = models.CharField(max_length=10, choices=DIFFICULTY_LEVELS)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    main_photo = models.ImageField(upload_to='recipes/main_photos/')
    video = models.FileField(upload_to='recipes/videos/', null=True, blank=True)
    hashtags = models.ManyToManyField(Hashtag, blank=True)

    def __str__(self):
```

```
    return self.title

# Добавьте остальные модели...
```

## Шаг 3: Альтернативные команды для диагностики

Попробуйте выполнить эти команды:

```
bash

# Показать все приложения
python manage.py showmigrations

# Создать миграции для конкретного приложения
python manage.py makemigrations recipes
python manage.py makemigrations users
python manage.py makemigrations comments

# Или создать миграции для всех приложений
python manage.py makemigrations

# Проверить, видит ли Django модели
python manage.py check
```

## Шаг 4: Если проблема сохраняется

Создайте файл `__init__.py` в каждой папке приложения (если его нет) и попробуйте:

```
bash

# Удалить папку __pycache__ и файлы миграций (если они есть)
rm -rf */__pycache__
rm -rf */migrations/__pycache__
rm -rf */migrations/00*.py

# Затем пересоздать миграции
python manage.py makemigrations
python manage.py migrate
```

## Шаг 5: Проверка структуры проекта

Убедитесь, что структура проекта правильная:

```
text
```

```
recipesAlmanah_project/
└── manage.py
└── culinary_project/
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── recipes/
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    ├── views.py
    └── migrations/
        └── __init__.py
└── users/
    ├── __init__.py
    └── models.py
└── comments/
    ├── __init__.py
    └── models.py
```

## Шаг 6: Принудительное создание миграций

Если ничего не помогает, создайте миграции вручную:

```
bash
```

```
# Удалите базу данных (если она пустая) и начните заново
python manage.py makemigrations recipes users comments
python manage.py migrate
```

```
# Или создайте пустые миграции и затем добавьте модели
python manage.py makemigrations --empty recipes
```

Попробуйте сначала шаг 3 и сообщите результат. Если покажет ошибки - это поможет понять проблему.

```
(.venv) PS D:\RecipesAlmanah\recipesAlmanah_project> python
manage.py check
System check identified no issues (0 silenced).
```

## Thought for 34 seconds ↴

- 💡 Мы проверили систему командой `python manage.py check`, и она не выявила проблем. Однако, при выполнении `makemigrations` изменения не обнаружены. Это может быть по нескольким причинам:
1. Модели не были импортированы в `admin.py` или в другом месте, поэтому Django их "не видит".
  2. Возможно, миграции уже созданы и применены (проверьте папку `migrations` в каждом приложении).
  3. Модели не были зарегистрированы в приложении должным образом.

Давайте проверим, есть ли миграции в приложениях. Если нет, то создадим их явно.

Сначала посмотрим на структуру папки `migrations` в каждом приложении (`recipes`, `users`, `comments`). В каждой из них должен быть файл `__init__.py` и, возможно, файлы миграций (например, `0001_initial.py`).

Если файлов миграций нет, то создадим их:

`bash`

```
python manage.py makemigrations recipes
python manage.py makemigrations users
python manage.py makemigrations comments
```

Если же миграции уже есть, но не применяются, то выполним:

`bash`

```
python manage.py migrate
```

Если миграции есть и применены, то Django считает, что изменения отсутствуют.

Давайте также убедимся, что модели действительно определены в `models.py` каждого приложения и что приложения добавлены в `INSTALLED_APPS`.

Если вы уже создавали модели и миграции, но потом удалили их, возможно, в базе данных остались старые таблицы. В таком случае можно попробовать удалить базу данных и создать заново (если это допустимо).

Если вы используете PostgreSQL, то можно удалить базу данных и создать ее снова, а затем выполнить миграции.

Если вы не хотите удалять базу данных, то можно попробовать сбросить миграции (если проект в разработке и можно потерять данные):

1. Удалите все файлы миграций в каждой папке `migrations`, кроме `__init__.py`.
2. Удалите таблицы из базы данных, связанные с этими приложениями.

3. Запустите `makemigrations` и `migrate` снова.

Однако, будьте осторожны, так как это удалит данные.

Еще один вариант: если вы вносили изменения в модели после создания миграций, то попробуйте создать новую миграцию:

`bash`

```
python manage.py makemigrations  
python manage.py migrate
```

Если ничего не помогает, давайте проверим, видит ли Django модели. Запустите shell:

`bash`

```
python manage.py shell
```

И попробуйте импортировать модели:

`python`

```
from recipes.models import Recipe  
from users.models import Profile  
from comments.models import Comment, Article
```

Если импорт проходит без ошибок, то модели есть. Тогда попробуйте создать миграции для каждого приложения явно.

Пожалуйста, выполните следующие команды и сообщите результат:

`bash`

```
python manage.py makemigrations recipes  
python manage.py makemigrations users  
python manage.py makemigrations comments
```

Если в какой-то из команд будет сообщение "No changes detected", то давайте проверим, есть ли в папке `migrations` этого приложения файлы миграций. Если нет, то возможно, вы не правильно назвали приложение или не добавили его в `INSTALLED_APPS`.

Если вы уверены, что модели есть и приложения добавлены, но миграции не создаются, то можно попробовать принудительно создать миграции, указав имя приложения и пустую миграцию, а затем вручную добавить код миграции. Но это крайний случай.

Жду результат выполнения команд.

Отлично! Система проверку прошла. Теперь давайте создадим миграции для каждого приложения отдельно:

## Шаг 1: Создание миграций для каждого приложения

bash

```
# Создаем миграции для каждого приложения по отдельности
python manage.py makemigrations recipes
python manage.py makemigrations users
python manage.py makemigrations comments
```

## Шаг 2: Если снова "No changes detected"

Попробуем более конкретные команды:

bash

```
# Показать текущее состояние миграций
python manage.py showmigrations

# Принудительно создать миграции
python manage.py makemigrations --name initial_migration recipes
python manage.py makemigrations --name initial_migration users
python manage.py makemigrations --name initial_migration comments
```

## Шаг 3: Проверим структуру моделей

Убедитесь, что в каждом приложении есть корректные модели:

`recipes/models.py` (должен содержать):

python

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

class Hashtag(models.Model):
    name = models.CharField(max_length=50, unique=True)
    def __str__(self):
        return self.name

class Recipe(models.Model):
```

```
DIFFICULTY_LEVELS = [
    ('easy', 'Легкий'),
    ('medium', 'Средний'),
    ('hard', 'Сложный'),
]

title = models.CharField(max_length=200)
description = models.TextField()
author = models.ForeignKey(User, on_delete=models.CASCADE)
cooking_time = models.PositiveIntegerField()
servings = models.PositiveIntegerField()
calories_per_100g = models.PositiveIntegerField()
difficulty = models.CharField(max_length=10, choices=DIFFICULTY_LEVELS)
created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)
main_photo = models.ImageField(upload_to='recipes/main_photos/')
video = models.FileField(upload_to='recipes/videos/', null=True, blank=True)
hashtags = models.ManyToManyField(Hashtag, blank=True)

def __str__(self):
    return self.title

class Ingredient(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='ingredients', on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    quantity = models.CharField(max_length=50)

    def __str__(self):
        return f'{self.name} - {self.quantity}'

class CookingStep(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='cooking_steps', on_delete=models.CASCADE)
    step_number = models.PositiveIntegerField()
    description = models.TextField()
    photo = models.ImageField(upload_to='recipes/steps_photos/', blank=True, null=True)

    class Meta:
        ordering = ['step_number']

    def __str__(self):
        return f'Step {self.step_number} for {self.recipe.title}'

class Favorite(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
```

```
added_at = models.DateTimeField(auto_now_add=True)

class Meta:
    unique_together = ('user', 'recipe')

def __str__(self):
    return f'{self.user.username} - {self.recipe.title}'
```

## Шаг 4: Альтернативное решение

Если миграции все еще не создаются, попробуем:

bash

```
# Удалить папки __pycache__
rmdir /s .venv
rmdir /s */__pycache__
rmdir /s */migrations/__pycache__

# Переустановить Django
pip uninstall django
pip install django

# И снова создать миграции
python manage.py makemigrations recipes
```

## Шаг 5: Проверка через Django shell

Давайте проверим, видит ли Django наши модели:

bash

```
python manage.py shell
```

В открывшейся консоли выполните:

python

```
from recipes.models import Recipe
from users.models import Profile
from comments.models import Comment

# Если эти импорты работают без ошибок, значит модели загружены
```

```
print("Модели загружены успешно!")
exit()
```

## Шаг 6: Если ничего не помогает

Создадим миграции вручную:

bash

```
# Создаем пустые миграции
python manage.py makemigrations --empty recipes --name initial
python manage.py makemigrations --empty users --name initial
python manage.py makemigrations --empty comments --name initial

# Затем применяем их
python manage.py migrate
```

Пожалуйста, выполните сначала Шаг 1 и сообщите результат каждой команды. Это поможет понять, в каком именно приложении проблема.

```
python manage.py makemigrations
System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory
'D:\RecipesAlmanah\recipesAlmanah_project\static' in the
STATICFILES_DIRS setting does not exist.

Traceback (most recent call last):
  File "D:\RecipesAlmanah\recipesAlmanah_project\manage.py", line
22, in <module>
    main()
    ~~~~^^
  File "D:\RecipesAlmanah\recipesAlmanah_project\manage.py", line
18, in main
    execute_from_command_line(sys.argv)
    ~~~~~^~~~~~^^^^^
  File "D:\RecipesAlmanah\.venv\Lib\site-
packages\django\core\management\__init__.py", line 442, in
execute_from_command_line
    utility.execute()
    ~~~~~~^~~^
  File "D:\RecipesAlmanah\.venv\Lib\site-
packages\django\core\management\__init__.py", line 436, in execute
```

```
    self.fetch_command(subcommand).run_from_argv(self.argv)
```

```
~~~~~  
^^  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\core\management\base.py", line 416, in  
run_from_argv  
    self.execute(*args, **cmd_options)  
~~~~~  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\core\management\base.py", line 460, in execute  
    output = self.handle(*args, **options)  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\core\management\base.py", line 107, in wrapper  
    res = handle_func(*args, **kwargs)  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\core\management\commands\makemigrations.py",  
line 159, in handle  
    loader.check_consistent_history(connection)  
~~~~~  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\db\migrations\loader.py", line 313, in  
check_consistent_history  
    applied = recorder.applied_migrations()  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\db\migrations\recorder.py", line 89, in  
applied_migrations  
    if self.has_table():  
~~~~~  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\db\migrations\recorder.py", line 63, in has_table  
    with self.connection.cursor() as cursor:  
~~~~~  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\utils\asyncio.py", line 26, in inner  
    return func(*args, **kwargs)  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\db\backends\base\base.py", line 320, in cursor  
    return self._cursor()  
~~~~~  
File "D:\RecipesAlmanah\.venv\lib\site-  
packages\django\db\backends\base\base.py", line 296, in _cursor  
    self.ensure_connection()
```

```
~~~~~^
File "D:\RecipesAlmanah\.venv\lib\site-
packages\django\utils\asyncio.py", line 26, in inner
    return func(*args, **kwargs)
File "D:\RecipesAlmanah\.venv\lib\site-
packages\django\db\backends\base\base.py", line 279, in
ensure_connection
    self.connect()
~~~~~^
File "D:\RecipesAlmanah\.venv\lib\site-
packages\django\utils\asyncio.py", line 26, in inner
    return func(*args, **kwargs)
File "D:\RecipesAlmanah\.venv\lib\site-
packages\django\db\backends\base\base.py", line 256, in connect
    self.connection = self.get_new_connection(conn_params)
~~~~~^^^^^^^^^
File "D:\RecipesAlmanah\.venv\lib\site-
packages\django\utils\asyncio.py", line 26, in inner
    return func(*args, **kwargs)
File "D:\RecipesAlmanah\.venv\lib\site-
packages\django\db\backends\postgresql\base.py", line 332, in
get_new_connection
    connection = self.Database.connect(**conn_params)
File "D:\RecipesAlmanah\.venv\lib\site-
packages\psycopg2\__init__.py", line 135, in connect
    conn = _connect(dsn, connection_factory=connection_factory,
**kwasync)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc2 in position
61: invalid continuation byte
```

Thought for 40 seconds ▾

⊗ Мы столкнулись с двумя проблемами:

1. Предупреждение о том, что директория для статических файлов не существует.
2. Ошибка кодировки при подключении к базе данных PostgreSQL.

Решение:

1. Создать папку `static` в корне проекта (рядом с `manage.py`), либо изменить настройки `STATICFILES_DIRS` в `settings.py`.
2. Ошибка кодировки в PostgreSQL. Это может быть из-за неправильной кодировки базы данных или неправильных параметров подключения.

Давайте сначала исправим настройки базы данных.

В файле `settings.py` убедитесь, что в настройках базы данных указаны правильные параметры, и попробуйте явно указать кодировку.

Также возможно, что в пароле или имени пользователя есть символы, которые не могут быть корректно обработаны. Попробуйте изменить пароль на простой, только из латинских букв и цифр.

Другой вариант — создать новую базу данных с явным указанием кодировки UTF-8.

Но сначала давайте попробуем исправить настройки в `settings.py`:

Пример настроек базы данных:

```
python
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'culinary_db',
        'USER': 'culinary_user',
        'PASSWORD': 'simple_password', # используйте простой пароль без специальных символов
        'HOST': 'localhost',
        'PORT': '5432',
        'OPTIONS': {
            'client_encoding': 'UTF8',
        }
    }
}
```

Также убедитесь, что база данных создана и пользователь имеет к ней доступ.

Если проблема сохраняется, давайте попробуем временно использовать SQLite для отладки:

```
python
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Это позволит проверить, что с SQLite миграции работают, а затем вернуться к настройке PostgreSQL.

Шаги:

1. Создадим папку `static` в корне проекта.

2. Изменим настройки базы данных на SQLite временно, чтобы проверить миграции.

3. Выполним миграции.

Если с SQLite все работает, тогда мы знаем, что проблема в подключении к PostgreSQL.

Давайте выполним:

1. Создаем папку `static` в корне проекта (рядом с `manage.py` ).

2. В `settings.py` закомментируйте настройки PostgreSQL и используйте SQLite:

`python`

*# Временно закомментируйте PostgreSQL и используйте SQLite*

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

3. Выполните команды:

`bash`

```
python manage.py makemigrations  
python manage.py migrate
```

Если миграции пройдут успешно, тогда проблема в настройках PostgreSQL.

Если вы хотите использовать PostgreSQL, то после отладки с SQLite, верните настройки PostgreSQL и решите проблему с кодировкой.

Как решить проблему с кодировкой в PostgreSQL:

- Убедитесь, что база данных создана с кодировкой UTF-8.
- Проверьте, что в параметрах подключения указана кодировка UTF-8.

Создание базы данных с кодировкой UTF-8:

`sql`

```
CREATE DATABASE culinary_db WITH ENCODING 'UTF8';
```

Также проверьте, что в `settings.py` в настройках базы данных указано:

`python`

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'culinary_db',
        'USER': 'culinary_user',
        'PASSWORD': 'your_password',
        'HOST': 'localhost',
        'PORT': '5432',
        'OPTIONS': {
            'client_encoding': 'UTF8',
        }
    }
}
```

Если проблема остается, попробуйте изменить пароль на простой (без не-ASCII символов).

Давайте начнем с временного использования SQLite, чтобы продолжить разработку, а затем вернемся к настройке PostgreSQL.

Есть две проблемы:

1. **Ошибка кодировки в PostgreSQL** - основная проблема
2. **Отсутствует папка static** - второстепенная

## Решение проблемы с PostgreSQL

### Вариант 1: Временное использование SQLite

Давайте временно переключимся на SQLite для разработки:

**settings.py:**

```
python

# Временно закомментируйте PostgreSQL и используйте SQLite
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Или если выше не работает, попробуйте:
DATABASES = {
    'default': {
```

```
'ENGINE': 'django.db.backends.sqlite3',
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
}
}
```

## Вариант 2: Решение проблемы с кодировкой PostgreSQL

Если хотите остаться с PostgreSQL:

### 1. Проверьте кодировку базы данных:

```
sql

-- Подключитесь к PostgreSQL как postgres
psql -U postgres

-- Проверьте кодировку базы данных
\l culinary_db

-- Если кодировка не UTF8, пересоздайте базу:
DROP DATABASE culinary_db;
CREATE DATABASE culinary_db WITH ENCODING 'UTF8';
```

### 2. Или создайте новую базу данных:

```
sql

CREATE DATABASE culinary_db_utf8 WITH ENCODING 'UTF8' TEMPLATE template0;
```

### 3. Обновите settings.py:

```
python

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'culinary_db_utf8', # Новая база
        'USER': 'your_username',
        'PASSWORD': 'your_password',
        'HOST': 'localhost',
        'PORT': '5432',
        'OPTIONS': {
            'client_encoding': 'UTF8',
        }
    }
}
```

```
    }  
}
```

## Решение проблемы с static папкой

Создайте папку static в корне проекта:

bash

```
# В корне проекта (где manage.py)  
mkdir static
```

Или удалите эту настройку из settings.py:

python

```
# Закомментируйте или удалите эту строку:  
# STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]  
  
# Или создайте папку, если её нет:  
import os  
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')] if os.path.exists(os.path.j  
oin(BASE_DIR, 'static')) else []
```

## Быстрое решение - используйте SQLite

Рекомендую временно перейти на SQLite для разработки:

1. Замените DATABASES в settings.py:

python

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
    }  
}
```

2. Создайте папку static:

bash

```
mkdir static
```

### 3. Теперь выполните миграции:

bash

```
python manage.py makemigrations recipes
python manage.py makemigrations users
python manage.py makemigrations comments
python manage.py migrate
```

### 4. Создайте суперпользователя:

bash

```
python manage.py createsuperuser
```

### 5. Запустите сервер:

bash

```
python manage.py runserver
```

После того как проект заработает с SQLite, вы всегда сможете вернуться к настройке PostgreSQL.

Попробуйте вариант с SQLite и сообщите результат!

```
python manage.py makemigrations
Traceback (most recent call last):
  File "D:\RecipesAlmanah\recipesAlmanah_project\manage.py", line
22, in <module>
    main()
    ~~~~^^
  File "D:\RecipesAlmanah\recipesAlmanah_project\manage.py", line
18, in main
    execute_from_command_line(sys.argv)
    ~~~~~~^~~~~~^^^^^
  File "D:\RecipesAlmanah\.venv\Lib\site-
packages\django\core\management\__init__.py", line 442, in
execute_from_command_line
    utility.execute()
    ~~~~~~^~~~^
```



Thought for 16 seconds ✓

 Мы по-прежнему получаем ошибку кодировки при подключении к PostgreSQL. Давайте временно переключимся на SQLite, чтобы продолжить разработку без задержек.

Шаги:

1. Изменим настройки базы данных в `settings.py` на использование SQLite.
  2. Убедимся, что у нас есть папка `static` (или временно уберем ее из настроек).
  3. Выполним миграции.

Вот как должен выглядеть раздел DATABASES в settings.py для SQLite:

```
python

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Также, чтобы избежать предупреждения о static, можно закомментировать или изменить строку со STATICFILES\_DIRS :

```
python

# STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
```

Или просто создать папку static в корне проекта.

Давайте внесем эти изменения и попробуем снова.

Если вы уже изменили settings.py , выполните:

```
bash

python manage.py makemigrations
```

Если вы предпочитаете использовать PostgreSQL, то нужно разобраться с кодировкой. Но для экономии времени на начальной стадии разработки, SQLite — хороший выбор.

После изменения настроек базы данных, попробуйте выполнить миграции еще раз.

Если вы хотите, чтобы я помог с настройкой PostgreSQL, дайте знать, но для быстрого старта я рекомендую SQLite.

Пожалуйста, выполните указанные изменения и сообщите о результате.

Проблема сохраняется с PostgreSQL. Давайте полностью переключимся на SQLite и решим проблему. Выполните следующие шаги:

## Шаг 1: Полностью замените настройки базы данных

В файле settings.py ЗАМЕНИТЕ блок DATABASES на:

```
python

DATABASES = {
    'default': {
```

```
'ENGINE': 'django.db.backends.sqlite3',
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
}
}
```

## Шаг 2: Уберите проблемные настройки static

Временно закомментируйте или удалите строку:

```
python

# STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
```

## Шаг 3: Создайте базовую структуру приложения

Убедитесь, что у вас есть минимальные модели. Создайте файл `recipes/models.py`:

```
python

from django.db import models
from django.contrib.auth.models import User

class Recipe(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)

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

## Шаг 4: Перезапустите процесс миграций

```
bash

# Удалите старые файлы миграций если они есть
rmdir /s recipes\migrations
mkdir recipes\migrations
echo "" > recipes\migrations\__init__.py

# Создайте миграции
python manage.py makemigrations
python manage.py migrate
```

## Шаг 5: Если все еще ошибка - сбросьте все

bash

```
# Удалите файл базы данных если он существует  
del db.sqlite3  
  
# Или создайте новый проект временно  
python manage.py makemigrations recipes  
python manage.py migrate  
python manage.py createsuperuser
```

## Альтернативное решение: Создайте новый проект

Если проблемы сохраняются, давайте создадим чистый проект:

bash

```
# Выйдите из текущей директории  
cd ..  
  
# Создайте новый проект  
django-admin startproject culinary_new  
cd culinary_new  
python manage.py startapp recipes  
python manage.py startapp users  
python manage.py startapp comments  
  
# Настройте settings.py как выше и попробуйте  
python manage.py makemigrations recipes  
python manage.py migrate
```

## Быстрая проверка

Создайте минимальный **settings.py** для теста:

python

```
import os  
  
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))  
SECRET_KEY = 'test-key-123'  
DEBUG = True  
ALLOWED_HOSTS = []
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'recipes',
]

DATA BASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

STATIC_URL = '/static/'
```

Пожалуйста, выполните Шаг 1 и Шаг 4 и сообщите результат. Если не сработает, будем создавать новый проект.

from recipes.models import Recipe ругается на эту строку

Thought for 21 seconds ▾

☒ Мы столкнулись с ошибкой при попытке импорта модели Recipe. Это может быть связано с несколькими причинами. Давайте проверим:

1. Правильно ли настроено приложение recipes в INSTALLED\_APPS ?
2. Существует ли файл recipes/models.py и содержит ли он модель Recipe ?
3. Не допущены ли опечатки в названии модели или в импорте?

Поскольку мы уже проверяли настройки приложений, давайте сосредоточимся на структуре проекта и файле моделей.

Убедитесь, что в файле recipes/models.py есть модель Recipe . Если нет, создайте ее.

Также, возможно, вы запускаете команду из неправильной директории? Убедитесь, что вы находитесь в директории, где находится manage.py .

Если вы только что создали модель, не забудьте создать миграции и применить их.

Если проблема сохраняется, попробуйте выполнить следующие команды в правильном порядке:

bash

```
python manage.py makemigrations recipes
python manage.py migrate
```

Если вы все еще видите ошибку, пожалуйста, предоставьте полное сообщение об ошибке.

Давайте также проверим, что приложение `recipes` добавлено в `INSTALLED_APPS` в `settings.py`:

```
python

INSTALLED_APPS = [
    # ...
    'recipes',
]
```

Если вы вносили изменения в модель, убедитесь, что перезапустили оболочку Django, если вы используете `python manage.py shell`.

Если вы запускаете скрипт, который импортирует модель, убедитесь, что вы установили переменную окружения `DJANGO_SETTINGS_MODULE` и настроили Django:

```
python

import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'your_project.settings')
django.setup()

from recipes.models import Recipe
```

Замените `your_project` на имя вашего проекта.

Если проблема не решена, предоставьте больше контекста: откуда вы импортируете (из оболочки, из скрипта, из `views` и т.д.) и полный текст ошибки.

Проблема с импортом модели `Recipe`. Давайте разберемся по порядку:

## Шаг 1: Проверим структуру проекта

Убедитесь, что структура проекта выглядит так:

```
text

recipesAlmanah_project/
|— manage.py
```

```
└── db.sqlite3
└── recipes/
    ├── __init__.py
    ├── models.py          ← здесь должна быть модель Recipe
    ├── admin.py
    ├── apps.py
    └── ...
└── users/
    ├── __init__.py
    └── models.py
└── comments/
    ├── __init__.py
    └── models.py
└── culinary_project/
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

## Шаг 2: Проверим файл recipes/models.py

Убедитесь, что в **recipes/models.py** есть модель Recipe:

```
python

from django.db import models
from django.contrib.auth.models import User

class Recipe(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    cooking_time = models.PositiveIntegerField(default=30)
    servings = models.PositiveIntegerField(default=2)
    calories_per_100g = models.PositiveIntegerField(default=100)

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

## Шаг 3: Проверим настройки приложений

В **settings.py** должно быть:

```
python
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'recipes',      # ваше приложение
    'users',        # ваше приложение
    'comments',     # ваше приложение
]
```

## Шаг 4: Правильный способ импорта

Если вы пытаетесь импортировать из Django shell:

```
bash
python manage.py shell
```

Затем в shell:

```
python
# Правильный способ импорта
from recipes.models import Recipe

# Или если выше не работает, попробуйте:
from recipes import models
Recipe = models.Recipe

# Проверка
print(Recipe.objects.all())
```

## Шаг 5: Если импорт все еще не работает

Создадим простейшую модель для теста:

`recipes/models.py:`

```
python
```

```
from django.db import models

class TestModel(models.Model):
    name = models.CharField(max_length=100)

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

Затем выполните:

```
bash

python manage.py makemigrations recipes
python manage.py migrate
python manage.py shell
```

В shell:

```
python

from recipes.models import TestModel
print("Импорт успешен!")
```

## Шаг 6: Проверка через manage.py

```
bash

# Проверим, видит ли Django приложение recipes
python manage.py check recipes

# Покажем все модели приложения recipes
python manage.py shell -c "from recipes.models import *; print([x for x in dir() if not x.startswith('_')])"
```

## Шаг 7: Если ничего не помогает - пересоздадим приложение

```
bash

# Удаляем и пересоздаем приложение
rmdir /s recipes
python manage.py startapp recipes
```

```
# Затем снова создаем models.py с простой моделью
```

## Пожалуйста:

1. Покажите содержимое вашего файла `recipes/models.py`
2. Покажите полную ошибку, которую вы получаете при импорте
3. Выполните команду: `python manage.py check recipes` и покажите результат

Это поможет точно определить проблему!

```
rmdir /s recipes
Remove-Item : Не удается найти позиционный параметр,
принимающий аргумент "recipes".
строка:1 знак:1
+ rmdir /s recipes
+ ~~~~~
+ CategoryInfo          : InvalidArgument: () [Remove-Item],
ParameterBindingException
+ FullyQualifiedErrorId :
PositionalParameterNotFound,Microsoft.PowerShell.Commands.RemoveltemCommand
```

Thought for 46 seconds ▾

⊗ Мы находимся в PowerShell, поэтому команды для удаления папки отличаются от Linux.

Вместо `rmdir /s recipes` используем `Remove-Item -Recurse -Force recipes`

Но давайте не будем удалять приложение, а попробуем сначала разобраться с импортом.

Пользователь сообщает, что импорт `from recipes.models import Recipe` не работает.

Давайте выполним следующие шаги:

1. Убедимся, что приложение `recipes` добавлено в `INSTALLED_APPS` в `settings.py`.
2. Убедимся, что в папке `recipes` есть файл `__init__.py` (пустой) и файл `models.py` с определением класса `Recipe`.
3. Попробуем выполнить миграции, чтобы Django знал о моделях.

Однако, если импорт не работает, возможно, модель не определена или есть синтаксическая ошибка в `models.py`.

Давайте проверим, есть ли модель `Recipe` в `recipes/models.py`.

Если файл `models.py` пустой или модель не определена, то импорт не сработает.

Также возможно, что мы находимся в неправильной директории (не в той, где `manage.py`).

Предложу пользователю показать содержимое `recipes/models.py` и вывод команды `python manage.py check recipes`.

Если мы не можем импортировать, возможно, есть ошибка в модели. Давайте создадим минимальную модель для проверки.

Но сначала давайте попробуем выполнить миграции, чтобы увидеть, есть ли какие-то ошибки.

Выполним:

```
python manage.py makemigrations recipes  
python manage.py migrate
```

Если миграции выполняются без ошибок, то модель должна быть доступна для импорта.

Если нет, то посмотрим на ошибку.

Также возможно, что мы пытаемся импортировать модель до того, как Django инициализирован. В shell это должно работать.

Давайте попробуем открыть shell и импортировать:

```
python manage.py shell
```

И затем:

```
from recipes.models import Recipe
```

Если в shell импорт работает, то проблема не в модели, а в том, откуда мы пытаемся импортировать (например, из самого файла `models.py`, что может вызвать циклический импорт).

Если пользователь пытается импортировать в другом файле (например, в `views.py`), то нужно убедиться, что импорт выполняется после настройки Django.

Но если мы только начали проект, то, вероятно, мы пытаемся импортировать в shell или в миграциях.

Итак, план:

1. Проверить содержимое `recipes/models.py`.
2. Выполнить `python manage.py check recipes` для проверки ошибок.
3. Выполнить `python manage.py makemigrations recipes` и `python manage.py migrate`.
4. Попробовать импорт в shell.

Если в shell импорт не работает, то, возможно, модель не была создана из-за ошибки.

Давайте попросим пользователя предоставить больше информации.

Однако, учитывая, что мы уже пытались сделать миграции и получали ошибки с PostgreSQL, а затем перешли на SQLite, возможно, база данных не была создана правильно.

Но если миграции выполнены, то модель должна быть в базе.

Но импорт модели не требует наличия базы данных (только определения класса), так что проблема должна быть в том, что Python не может найти класс Recipe.

Возможные причины:

- Опечатка в названии класса (например, Recipes вместо Recipe ).
- Файл models.py не сохранен.
- Приложение не загружено.

Давайте попробуем выполнить:

```
python manage.py shell -c "import recipes.models; print(recipes.models.file)"
```

Это покажет, откуда загружается models.py.

Также можно выполнить:

```
python manage.py shell -c "from recipes.models import Recipe; print(Recipe)"
```

Если это не работает, то нужно посмотреть, есть ли класс Recipe в models.py.

Поскольку пользователь уже пытался удалить приложение и пересоздать, но команда rmdir не сработала, давайте исправим команду для PowerShell:

```
Remove-Item -Recurse -Force recipes
```

Но прежде чем удалять, давайте убедимся, что проблема не в чем-то другом.

Я предлагаю:

1. Показать содержимое recipes/models.py .
2. Показать вывод команды python manage.py check recipes .

Если пользователь не предоставит эту информацию, то мы не сможем помочь.

Поэтому я попрошу пользователя:

- Показать содержимое файла recipes/models.py .
- Выполнить команду python manage.py check recipes и показать вывод.

Если в модели есть ошибка, например, неправильный импорт, то check это покажет.

Если check проходит без ошибок, то модель должна импортироваться.

Если check не проходит, то исправим ошибку.

Если пользователь не может выполнить check из-за ошибки, то нужно исправить ошибку.

Давайте подождем ответа пользователя.

В PowerShell команды отличаются от командной строки Windows. Используйте правильные команды для PowerShell:

## Для удаления папки в PowerShell:

```
powershell

# Удалить папку recipes
Remove-Item -Recurse -Force recipes

# Или если хотите использовать rmdir аналог:
rm -r recipes
```

**Но лучше не удалять приложение! Давайте решим проблему с импортом:**

### Шаг 1: Проверим структуру проекта

Выполните в PowerShell:

```
powershell

Get-ChildItem -Recurse
```

### Шаг 2: Проверим содержимое recipes/models.py

```
powershell

Get-Content recipes\models.py
```

### Шаг 3: Проверим настройки приложений

```
powershell

Get-Content culinary_project\settings.py | Select-String "INSTALLED_APPS"
```

### Шаг 4: Создадим простую модель для теста

recipes/models.py:

```
python
```

```
from django.db import models

class Recipe(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()

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

## Шаг 5: Выполним миграции

```
powershell

python manage.py makemigrations recipes
python manage.py migrate
```

## Шаг 6: Проверим импорт

```
powershell

python manage.py shell
```

В открывшейся консоли Python:

```
python

from recipes.models import Recipe
print("Успех! Модель Recipe импортирована")
exit()
```

## Если все еще есть ошибка, выполните диагностику:

### Проверка 1: Убедимся, что приложение зарегистрировано

```
powershell

python manage.py shell -c "
from django.apps import apps
for app in apps.get_app_configs():
    print(f'{app.name}: {app.verbose_name}')
    for model in app.get_models():"
```

```
print(f' - {model.__name__}')  
"
```

## Проверка 2: Проверим пути Python

```
powershell  
  
python manage.py shell -c "  
import sys  
for path in sys.path:  
    print(path)  
"
```

**Пожалуйста, выполните сначала Шаги 1-3 и покажите результат. Это поможет понять структуру вашего проекта и найти проблему.**

Если проблема в путях, мы можем добавить правильный путь в `sys.path` или настроить структуру проекта.

как выйти из powershell

Thought for 8 seconds ▾

💡 Мы находимся в интерактивной сессии Python (запущенной через `python manage.py shell`), а не в PowerShell.

Чтобы выйти из интерактивной сессии Python, нужно ввести:

`exit()`

Или использовать сочетание клавиш:

`Ctrl + Z`, затем `Enter` (для Windows)

Или:

`Ctrl + D` (для Linux/Mac)

Однако, если вы имели в виду выход из PowerShell (командной оболочки), то введите:

`exit`

Но из контекста, кажется, что мы в Python shell. Поэтому используйте `exit()` для выхода из Python shell.

Если же вы находитесь в PowerShell, то просто закройте окно или введите `exit` для выхода.

Уточните, из чего именно вы хотите выйти?

## **Выход из разных режимов:**

### **1. Выход из Python shell (где вы импортировали Recipe):**

```
python
```

```
exit()
```

или

```
python
```

```
quit()
```

или нажмите Ctrl + Z затем Enter (Windows)

### **2. Выход из PowerShell полностью:**

```
powershell
```

```
exit
```

### **3. Если вы в Django shell (python manage.py shell):**

Тоже используйте:

```
python
```

```
exit()
```

### **4. Если вы в командной строке Windows (cmd):**

```
text
```

```
exit
```

## **Чтобы продолжить работу с проектом:**

После выхода из Python shell вы останетесь в PowerShell, где можете выполнять команды:

```
powershell
```

```
# Вы сейчас в PowerShell, можете выполнять:  
python manage.py makemigrations recipes  
python manage.py migrate  
python manage.py runserver
```

### Как понять где вы находитесь:

- Если приглашение строки `>>>` - вы в Python shell (выходите через `exit()`)
- Если приглашение `PS D:\path>` - вы в PowerShell (выходите через `exit`)
- Если приглашение `D:\path>` - вы в командной строке cmd (выходите через `exit`)

**Пожалуйста, уточните - какое именно приглашение вы видите сейчас?** Это поможет дать точную инструкцию.

```
python manage.py migrate
```

Operations to perform:

  Apply all migrations: admin, auth, comments, contenttypes, recipes, sessions, users

Running migrations:

  No migrations to apply.

Thought for 19 seconds ▾

💡 Мы уже выполнили миграции, и теперь база данных создана. Давайте проверим, что модели зарегистрированы и работают.

Поскольку миграции применены, мы можем создать суперпользователя и запустить сервер.

Однако, если ранее были ошибки с импортом моделей, давайте убедимся, что модели действительно существуют и их можно импортировать.

Шаг 1: Проверим, что модель Recipe доступна в Django shell.

Откроем shell и попробуем импортировать модель Recipe:

```
bash
```

```
python manage.py shell
```

В shell выполним:

```
python
```

```
from recipes.models import Recipe
Recipe.objects.all()
```

Если мы видим пустой список ([]), то это значит, что модель работает.

Шаг 2: Создадим суперпользователя, если еще не создан.

bash

```
python manage.py createsuperuser
```

Шаг 3: Запустим сервер.

bash

```
python manage.py runserver
```

Теперь мы можем зайти в админку (<http://127.0.0.1:8000/admin>) и убедиться, что модели есть.

Если в админке мы не видим модели recipes, нужно зарегистрировать их в файле recipes/admin.py.

Давайте зарегистрируем модель Recipe в админке.

recipes/admin.py:

```
python

from django.contrib import admin
from .models import Recipe

admin.site.register(Recipe)
```

После этого обновите страницу админки, и вы увидите модель Recipe.

Если вы хотите зарегистрировать и другие модели (Ingredient, CookingStep и т.д.), то аналогично добавьте их.

Теперь давайте продолжим разработку проекта, создав необходимые представления (views) и URL-адреса.

Но прежде, если у вас остались проблемы с импортом, давайте их решим.

Если при импорте в shell возникает ошибка, пожалуйста, опишите ее.

Если все работает, то мы можем переходить к созданию views и шаблонов.

Как вы хотите продолжить?

Отлично! Миграции успешно применены. Это означает, что:

1.  База данных настроена правильно
2.  Все модели созданы в базе данных
3.  Django видит все приложения (recipes, users, comments)

## Следующие шаги:

### 1. Создайте суперпользователя для доступа к админке:

```
powershell
```

```
python manage.py createsuperuser
```

Следуйте инструкциям для создания учетной записи администратора.

### 2. Запустите сервер разработки:

```
powershell
```

```
python manage.py runserver
```

### 3. Откройте браузер и перейдите:

- Админка: <http://127.0.0.1:8000/admin/>
- Сайт: <http://127.0.0.1:8000/>

### 4. Проверьте, что модели работают через Django shell:

```
powershell
```

```
python manage.py shell
```

В shell выполните:

```
python
```

```
from recipes.models import Recipe
# Создайте тестовый рецепт
recipe = Recipe.objects.create()
```

```
        title="Тестовый рецепт",
        description="Это тестовое описание",
        cooking_time=30,
        servings=4,
        calories_per_100g=150,
        difficulty="easy"
    )
print(f"Создан рецепт: {recipe.title}")
exit()
```

## 5. Настройте админ-панель для удобного управления:

`recipes/admin.py:`

```
python

from django.contrib import admin
from .models import Recipe, Hashtag, Ingredient, CookingStep, Favorite

admin.site.register(Recipe)
admin.site.register(Hashtag)
admin.site.register(Ingredient)
admin.site.register(CookingStep)
admin.site.register(Favorite)
```

## Что делать дальше:

1. Запустите сервер и проверьте работу админки
2. Создайте через админку несколько тестовых рецептов, хештегов
3. Приступайте к созданию `views` и шаблонов для функционала сайта

Если при создании суперпользователя или запуске сервера возникнут ошибки – сообщите, помогу решить!

**Попробуйте создать суперпользователя и запустить сервер - все ли работает?**

```
python manage.py makemigrations
No changes detected
```

Thought for 8 seconds ▾

💡 Мы уже выполнили миграции, и сейчас нет изменений в моделях. Это нормально.

Теперь нужно создать суперпользователя и запустить сервер.

Однако, если в будущем вы внесете изменения в модели, то нужно будет снова запустить makemigrations и migrate.

Сейчас давайте создадим суперпользователя:

```
python manage.py createsuperuser
```

Следуйте инструкциям.

Затем запустите сервер:

```
python manage.py runserver
```

И откройте в браузере <http://127.0.0.1:8000/admin/> для входа в админку.

Если вы хотите начать создавать представления (views) и шаблоны, то давайте продолжим.

Хотите, чтобы я помог с созданием представлений для отображения списка рецептов и деталей рецепта?

Это нормальное сообщение - оно означает, что все миграции уже созданы и применены. Django не обнаружил изменений в моделях, требующих новых миграций.

## Что делать дальше:

### 1. Создайте суперпользователя (если еще не создали):

```
powershell
```

```
python manage.py createsuperuser
```

### 2. Запустите сервер и проверьте работу:

```
powershell
```

```
python manage.py runserver
```

### 3. Добавьте тестовые данные через админку:

Перейдите на <http://127.0.0.1:8000/admin/> и добавьте:

- Несколько рецептов
- Хештеги
- Ингредиенты

- Пользователей

#### 4. Или создайте тестовые данные через Django shell:

```
powershell

python manage.py shell

python

from django.contrib.auth.models import User
from recipes.models import Recipe, Hashtag, Ingredient, CookingStep

# Создаем пользователя
user = User.objects.create_user('testuser', password='testpass123')
user.save()

# Создаем хештеги
breakfast_tag = Hashtag.objects.create(name='завтрак')
diet_tag = Hashtag.objects.create(name='диетический')

# Создаем рецепт
recipe = Recipe.objects.create(
    title='Омлет с овощами',
    description='Вкусный и полезный завтрак',
    author=user,
    cooking_time=15,
    servings=2,
    calories_per_100g=120,
    difficulty='easy'
)
recipe.hashtags.add(breakfast_tag, diet_tag)

# Добавляем ингредиенты
Ingredient.objects.create(recipe=recipe, name='Яйца', quantity='3 шт')
Ingredient.objects.create(recipe=recipe, name='Помидор', quantity='1 шт')
Ingredient.objects.create(recipe=recipe, name='Соль', quantity='по вкусу')

# Добавляем шаги приготовления
CookingStep.objects.create(
    recipe=recipe,
    step_number=1,
    description='Взбить яйца с солью'
)
CookingStep.objects.create(
    recipe=recipe,
```

```
    step_number=2,  
    description='Добавить нарезанный помидор'  
)  
  
print("Тестовые данные созданы!")  
exit()
```

## Следующие этапы разработки:

### 1. Создайте базовые представления (Views) для:

- Главной страницы со списком рецептов
- Детальной страницы рецепта
- Поиска рецептов

### 2. Создайте HTML шаблоны

### 3. Настройте URL маршруты

### 4. Добавьте формы для:

- Добавления/редактирования рецептов
- Комментариев
- Регистрации пользователей

Хотите, чтобы я помог с созданием представлений и шаблонов для следующего этапа?

```
from recipesAlmanah_project.recipes.models import Recipe  
ModuleNotFoundError: No module named  
'recipesAlmanah_project.recipes'  
(.venv) PS D:\RecipesAlmanah\recipesAlmanah_project>
```

Thought for 41 seconds ▾

☒ Мы получаем ошибку, потому что неправильно указан путь для импорта.

В данном случае, мы находимся в корне проекта (где manage.py) и пытаемся импортировать модель Recipe из приложения recipes.

Правильный способ импорта в Django (когда мы находимся в корне проекта) - это использовать абсолютный импорт от корня проекта.

Однако, в данном случае, мы можем использовать относительный импорт или просто указать правильный путь.

Но обратите внимание: мы уже выполняем команду в контексте проекта, поэтому Django должен знать о приложениях.

Давайте разберемся:

1. Мы находимся в директории D:\RecipesAlmanah\recipesAlmanah\_project
2. В этой директории у нас есть manage.py и папка проекта (culinary\_project?) и приложения (recipes, users, comments)

Если структура проекта такая:

recipesAlmanah\_project/

manage.py

recipesAlmanah\_project/

init.py

settings.py

urls.py

wsgi.py

recipes/

init.py

models.py

users/

init.py

models.py

comments/

init.py

models.py

Тогда правильный импорт из корня проекта (из manage.py) должен быть:

```
from recipes.models import Recipe
```

Но если мы пытаемся выполнить этот импорт из скрипта, который находится в корне проекта, то это должно работать.

Однако, если мы пытаемся выполнить этот импорт из другого места (например, из файла внутри проекта), то нужно убедиться, что корневая директория проекта добавлена в sys.path.

В данном случае, ошибка говорит о том, что модуль 'recipesAlmanah\_project.recipes' не найден.

Это может быть потому, что мы пытаемся импортировать как будто бы recipesAlmanah\_project - это пакет, а внутри него есть пакет recipes.

Но на самом деле, в структуре проекта, которая была описана ранее, приложения находятся на одном уровне с пакетом проекта.

Поэтому, правильный импорт: `from recipes.models import Recipe`

Если вы запускаете этот импорт из `manage.py` (который находится в `recipesAlmanah_project`) - то это должно работать.

Если вы запускаете из другого файла, например, внутри пакета `recipesAlmanah_project`, то нужно делать так:

```
from ..recipes.models import Recipe
```

Но это сработает только если файл, из которого делается импорт, находится внутри пакета `recipesAlmanah_project`.

Давайте уточним, откуда вы делаете импорт?

Если вы делаете импорт из `manage.py`, то используйте:

```
from recipes.models import Recipe
```

Если вы делаете импорт из файла внутри `recipesAlmanah_project` (пакета проекта), то используйте:

```
from recipes.models import Recipe # если приложение recipes находится на одном уровне с пакетом проекта
```

Но в данном случае, из сообщения об ошибке видно, что вы пытаетесь сделать:

```
from recipesAlmanah_project.recipes.models import Recipe
```

Это неправильно, если приложение `recipes` не находится внутри пакета `recipesAlmanah_project`.

Рекомендую:

1. Убедиться, что вы находитесь в корне проекта (где `manage.py`) и используете виртуальное окружение.
2. Запустить Django shell:

```
text
```

```
python manage.py shell
```

и затем в shell попробовать:

```
text
```

```
from recipes.models import Recipe
```

3. Если это не работает, проверьте, что приложение 'recipes' добавлено в INSTALLED\_APPS в settings.py.
4. Также убедитесь, что в корне проекта есть папка recipes и в ней есть файл models.py с классом Recipe.

Если вы хотите импортировать модель Recipe в другом файле (например, в views.py приложения recipes), то там вы можете использовать:

```
text
from .models import Recipe
```

Или, если вы находитесь в другом приложении, то:

```
text
from recipes.models import Recipe
```

Если вы пишете самостоятельный скрипт, который должен использовать модели Django, то вам нужно настроить окружение Django. Для этого в начале скрипта добавьте:

```
text
import os
import django
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'recipesAlmanah_project.settings')
django.setup()

from recipes.models import Recipe
```

Но в данном случае, поскольку вы находитесь в корне проекта и используете manage.py, то проще использовать Django shell.

Если вы все еще видите ошибку, пожалуйста, уточните, откуда вы пытаетесь импортировать модель Recipe (в каком файле или в какой команде).

Проблема с импортом! Вы пытаетесь импортировать неправильным путем. Давайте разберемся:

## Правильные способы импорта:

### 1. Из Django shell (рекомендуется):

```
powershell
```

```
python manage.py shell
```

```
python
```

```
# Правильный импорт из Django shell
from recipes.models import Recipe
```

## 2. Из Python файлов внутри проекта:

В файлах ваших приложений (views.py, admin.py и т.д.):

```
python
```

```
from recipes.models import Recipe
# или
from .models import Recipe # если в том же приложении
```

## 3. Если хотите импортировать из корня проекта:

Создайте файл scripts/test\_import.py в корне проекта:

```
python
```

```
import os
import django
import sys

# Добавляем корень проекта в Python path
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'culinary_project.settings')
django.setup()

from recipes.models import Recipe
print("Импорт успешен!")
```

## Проверьте текущую структуру проекта:

```
powershell
```

```
Get-ChildItem -Recurse
```

Убедитесь, что структура выглядит так:

```
text

recipesAlmanah_project/
    ├── manage.py
    ├── db.sqlite3
    ├── recipes/
    │   ├── __init__.py
    │   ├── models.py
    │   └── ...
    └── culinary_project/ # или как называется ваша основная папка проекта?
        ├── __init__.py
        ├── settings.py
        └── ...
```

## Быстрая проверка работы:

powershell

# 1. Запустите Django shell

```
python manage.py shell
```

# 2. В открывшейся консоли выполните:

```
from recipes.models import Recipe
Recipe.objects.all()
exit()
```

# 3. Запустите сервер

```
python manage.py runserver
```

**Какое имя у вашей основной папки проекта?** Та, что содержит `settings.py` - `culinary_project` или `recipesAlmanah_project` ?

Это важно для правильной настройки импортов!

recipeAlmanah\_project

Thought for 31 seconds ▾

💡 Мы выяснили, что основная папка проекта называется `recipeAlmanah_project` (обратите внимание на опечатку в предыдущем сообщении: было "recipesAlmanah\_project", теперь "recipeAlmanah\_project").

Однако, в ошибке было указано `from recipesAlmanah_project.recipes.models import Recipe` - это неправильный путь для импорта в Django.

Давайте уточним структуру проекта:

```
recipeAlmanah_project/
manage.py
recipeAlmanah_project/ ← это папка проекта (содержит settings.py, urls.py и т.д.)
settings.py
urls.py
...
recipes/ ← приложение recipes
models.py
...
users/ ← приложение users
comments/ ← приложение comments
```

В Django приложения импортируются непосредственно по их имени, без указания папки проекта.

Правильный импорт в Django shell или в файлах проекта (views.py, models.py и т.д.):

```
text
from recipes.models import Recipe
```

Но если вы пытаетесь запустить скрипт извне (например, из корня проекта, но не в рамках Django окружения), то нужно настроить Django окружение.

Для импорта в скрипте, который запускается из корня проекта (рядом с manage.py), вы можете сделать так:

```
text
import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'recipeAlmanah_project.settings')
django.setup()

from recipes.models import Recipe
```

Однако, если вы находитесь в Django shell (через `python manage.py shell`) или в коде приложения (views, models и т.д.), то импорт должен быть просто

```
from recipes.models import Recipe .
```

Итак, если вы получаете ошибку

`ModuleNotFoundError: No module named 'recipesAlmanah_project.recipes'`, то вы пытаетесь импортировать в контексте, где Python не знает о структуре проекта.

Решение:

1. Убедитесь, что вы находитесь в директории, где находится `manage.py`, и используете `python manage.py shell` для доступа к моделям.
2. Внутри `shell` используйте правильный импорт: `from recipes.models import Recipe`
3. Если вы пишете код в приложении (например, в `views.py` приложения `recipes`), то используйте тот же импорт.
4. Если вы пишете самостоятельный скрипт, который нужно запускать из командной строки, то в начале скрипта добавьте настройку Django окружения, как показано выше.

Давайте проверим, что вы можете импортировать модель `Recipe`:

```
text

python manage.py shell

>>> from recipes.models import Recipe
>>> Recipe.objects.all()
<QuerySet []>
```

Если вы видите пустой `QuerySet`, то это успех – модель импортирована и работает.

Если нет, давайте проверим, что приложение `recipes` зарегистрировано в `settings.py`:

```
text

INSTALLED_APPS = [
    ...,
    'recipes',
    ...,
]
```

И что в папке `recipes` есть файл `models.py` с определением класса `Recipe`.

Если вы до сих пор сталкиваетесь с проблемами, пожалуйста, уточните, откуда вы пытаетесь импортировать (из какого файла или контекста) и покажите актуальную структуру проекта.

Теперь понятно! Ваш проект называется `recipeAlmanah_project`. Давайте исправим импорт и настройки:

## Исправление импорта

**Неправильно:**

```
python
```

```
from recipesAlmanah_project.recipes.models import Recipe
```

Правильно:

```
python
```

```
from recipes.models import Recipe
```

## Проверка структуры проекта

Убедитесь, что структура выглядит так:

```
text
```

```
recipeAlmanah_project/
├── manage.py
├── db.sqlite3
└── recipes/
    ├── __init__.py
    ├── models.py
    ├── admin.py
    └── ...
├── recipeAlmanah_project/ # папка с настройками
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
└── users/
    └── ...
└── comments/
    └── ...
```

## Проверка настроек

В `recipeAlmanah_project/settings.py` должно быть:

```
python
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

```
'django.contrib.messages',
'django.contrib.staticfiles',
'recipes',      # ваше приложение
'users',        # ваше приложение
'comments',     # ваше приложение
]

ROOT_URLCONF = 'recipeAlmanah_project.urls'
```

## Правильный способ работы с моделями

### 1. Из Django shell:

```
powershell

python manage.py shell
```

```
python

from recipes.models import Recipe
# Теперь можно работать с моделью
recipes = Recipe.objects.all()
print(recipes)
```

### 2. В файлах вашего приложения (views.py, admin.py):

```
python

from .models import Recipe # относительный импорт
# или
from recipes.models import Recipe # абсолютный импорт
```

## Проверка работы

Выполните эти команды по порядку:

```
powershell

# 1. Проверим структуру
Get-ChildItem
```

```
# 2. Запустим Django shell  
python manage.py shell
```

В Django shell выполните:

```
python  
  
# 3. Правильный импорт  
from recipes.models import Recipe  
  
# 4. Проверим, что модель работает  
print(Recipe.objects.all())  
  
# 5. Создадим тестовый рецепт (если нужно)  
recipe = Recipe(title="Тестовый рецепт")  
recipe.save()  
print("Рецепт создан!")  
  
# 6. Выйдем  
exit()
```

Если при импорте `from recipes.models import Recipe` все еще возникает ошибка, пожалуйста, покажите:

1. Результат команды `Get-ChildItem` (структуру папок)
2. Содержимое файла `recipes/models.py`
3. Полную ошибку, которую вы получаете

Это поможет точно определить проблему!



Thought for 26 seconds ▾

⌚ Мы видим, что в файле `comments/models.py` есть попытка импорта из `recipesAlmanah_project.recipes.models`, которая вызывает ошибку, потому что такой модуль не найден.

Проблема в том, что структура проекта не соответствует ожиданиям импорта. В Django приложениях импорты должны быть относительными или абсолютными, но без включения корневой папки проекта в путь.

Исправление:

В файле `comments/models.py` замените строку:

```
python
```

```
from recipesAlmanah_project.recipes.models import Recipe
```

на

```
python
```

```
from recipes.models import Recipe
```

Аналогично, если есть другие импорты с `recipesAlmanah_project`, их нужно исправить.

Также, убедитесь, что в `settings.py` в `INSTALLED_APPS` приложения зарегистрированы по их именам (папкам), а не по полным путям. Например:

```
python
```

```
INSTALLED_APPS = [  
    ...,  
    'recipes',  
    'comments',  
    'users',  
]
```

После исправления импортов, выполните:

```
bash
```

```
python manage.py makemigrations  
python manage.py migrate
```

Если были созданы миграции для приложения `comments`, то они применяются.

Также, если вы запускали сервер или `shell`, то перезапустите их, чтобы изменения в моделях вступили в силу.

Если вы используете абсолютные импорты, то убедитесь, что корневая папка проекта добавлена в `sys.path`. Обычно это делается автоматически при запуске через `manage.py`, но вручную это можно сделать так:

```
python
```

```
import sys  
import os
```

```
# Добавляем корневую папку проекта в sys.path
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
```

Однако, в данном случае, поскольку мы работаем в рамках Django проекта, лучше использовать относительные импорты или абсолютные без указания корневой папки проекта.

Если вы хотите импортировать модель `Recipe` из приложения `recipes` в другом приложении (например, `comments`), то правильный способ — `from recipes.models import Recipe`.

Если вы находитесь в файле, который находится внутри проекта, но не в приложении, и вам нужно импортировать модель, то можно использовать настройку Django:

```
python

import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'recipeAlmanah_project.settings')
django.setup()

from recipes.models import Recipe
```

Но это обычно нужно для скриптов, которые запускаются вне контекста Django (например, отдельный скрипт). Внутри приложений Django (модели, представления, etc) такие настройки не требуются.

Итак, основное действие: исправить импорт в `comments/models.py`.

Я вижу проблему! Структура проекта неправильная и есть ошибка в импорте. Давайте исправим:

## Проблема 1: Неправильная структура проекта

Судя по описанию, у вас неправильная структура. Должно быть:

```
text

recipesAlmanah_project/          # ← КОРНЕВАЯ папка проекта
|   __init__.py
|   db.sqlite3
|   recipesAlmanah_project/      # ← Папка с настройками (должна быть внутри корней)
|       __init__.py
|       settings.py
|       urls.py
|       wsgi.py
```

```
recipes/                                # ← Приложение recipes
    __init__.py
    models.py
    admin.py
    ...
comments/                               # ← Приложение comments
    __init__.py
    models.py
    ...
users/                                  # ← Приложение users
    __init__.py
    models.py
    ...
```

## Проблема 2: Неправильный импорт в comments/models.py

Исправьте файл comments/models.py :

```
python

# НЕПРАВИЛЬНО:
from recipesAlmanah_project.recipes.models import Recipe

# ПРАВИЛЬНО:
from recipes.models import Recipe
```

## Исправление структуры проекта

Выполните в PowerShell:

```
powershell

# 1. Перейдите в корневую папку проекта (где manage.py)
cd D:\RecipesAlmanah\recipesAlmanah_project

# 2. Проверим текущую структуру
Get-ChildItem -Recurse

# 3. Если нужно переименовать папку с настройками:
Rename-Item "recipeAlmanah_project" "recipesAlmanah_project_temp"
Move-Item "recipesAlmanah_project_temp\*" "recipesAlmanah_project\
Remove-Item "recipesAlmanah_project_temp"
```

## Проверка и исправление settings.py

Убедитесь, что в `recipesAlmanah_project/settings.py`:

```
python

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'recipes',          # ← без префикса recipesAlmanah_project
    'users',            # ← без префикса
    'comments',         # ← без префикса
]

ROOT_URLCONF = 'recipesAlmanah_project.urls'
WSGI_APPLICATION = 'recipesAlmanah_project.wsgi.application'
```

## Исправление всех импортов

Во всех файлах проекта замените неправильные импорты:

В `comments/models.py`:

```
python

from recipes.models import Recipe
from django.contrib.auth.models import User

class Comment(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='comments', on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Comment by {self.author.username}"
```

## Проверка работы

После исправлений выполните:

```
powershell

# Проверим структуру
Get-ChildItem

# Проверим настройки Django
python manage.py check

# Создадим миграции (если нужно)
python manage.py makemigrations

# Применим миграции
python manage.py migrate

# Запустим сервер
python manage.py runserver
```

Пожалуйста:

1. Покажите текущую структуру папок командой `Get-ChildItem`
2. Покажите содержимое файла `comments/models.py`
3. Попробуйте выполнить `python manage.py check` и покажите результат

После этого я помогу окончательно исправить структуру проекта!

При открытии `http://127.0.0.1:8000/admin/comments/comment/` выдаёт `django.db.utils.OperationalError: no such column: comments_comment.recipe_id`  
`[01/Oct/2025 23:06:17] "GET /admin/comments/comment/ HTTP/1.1"`  
`500 175455`

Thought for 70 seconds ▾

☒ Мы столкнулись с ошибкой, что в таблице `comments_comment` отсутствует столбец `recipe_id`.  
Это означает, что миграции для приложения `comments` не были применены или были применены не полностью.

Давайте проверим и применим миграции для приложения `comments`:

1. Сначала убедимся, что у нас есть миграции для приложения `comments`.
2. Затем применим их.

Если миграции для `comments` отсутствуют, создадим их.

Выполните следующие команды:

```
bash
```

```
# Создание миграций для приложения comments
python manage.py makemigrations comments

# Применение миграций
python manage.py migrate
```

Если при создании миграций для `comments` возникает ошибка, возможно, в модели `Comment` есть проблемы. Убедитесь, что модель `Comment` в файле `comments/models.py` определена правильно.

Вот пример правильной модели `Comment`:

```
python
```

```
from django.db import models
from recipes.models import Recipe
from django.contrib.auth.models import User

class Comment(models.Model):
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Comment by {self.author.username} on {self.recipe.title}"
```

После исправления модели повторите создание и применение миграций.

Если же миграции для `comments` уже существуют, но не применяются, можно попробовать применить миграции конкретного приложения:

```
bash
```

```
python manage.py migrate comments
```

Если проблема сохраняется, возможно, база данных находится в неконсистентном состоянии. Тогда можно попробовать сбросить миграции для приложения `comments` и начать заново:

1. Удалите все файлы миграций в папке `comments/migrations`, кроме `__init__.py`.
2. Удалите запись о миграциях `comments` из таблицы `django_migrations` в базе данных (если вы не против сбросить состояние миграций для этого приложения).
3. Затем заново создайте и примените миграции.

Однако, если вы не хотите сбрасывать миграции, можно попробовать:

```
bash
```

```
# Показать все миграции
python manage.py showmigrations

# Если видно, что миграции для comments не применены, примените их:
python manage.py migrate comments
```

Если же вы видите, что миграции для `comments` применены, но ошибка остается, возможно, в миграциях не было создания поля `recipe_id`. Тогда нужно убедиться, что в миграциях есть операция добавления поля `recipe` во модель `Comment`.

Давайте проверим, что в миграциях для `comments` есть создание поля `recipe`.

Если нет, то после исправления модели `Comment` нужно заново создать миграции.

Если вы не против сбросить миграции для приложения `comments`, выполните:

```
bash
```

```
# Удаляем все миграции comments (кроме __init__.py)
rm comments/migrations/0*.py

# Создаем новые миграции
python manage.py makemigrations comments

# Применяем их
python manage.py migrate comments
```

Внимание: Удаление файлов миграций может привести к потере данных, если миграции уже были применены и в них были какие-то данные. Если вы находитесь на стадии разработки и база данных не содержит важных данных, это безопасно.

Если же в базе данных есть важные данные, то лучше не удалять миграции, а создать новую миграцию, которая добавит недостающее поле.

Для этого:

1. Убедитесь, что модель `Comment` содержит поле `recipe`.
2. Создайте миграцию: `python manage.py makemigrations comments`
3. Примените ее: `python manage.py migrate comments`

Если при создании миграции вы видите, что она пустая (`No changes detected`), то это значит, что Django считает, что модель уже синхронизирована. Тогда возможно, что миграции уже были созданы, но не применены. Проверьте `showmigrations` и примените их.

Если же миграции применены, но поле отсутствует, то возможно, что в миграциях нет операции добавления этого поля. Тогда нужно вручную добавить миграцию.

Однако, проще на данном этапе, особенно если база данных не содержит важных данных, удалить базу данных и начать заново:

1. Удалите файл базы данных (`db.sqlite3`)
2. Удалите все файлы миграций (кроме `init.py`) во всех приложениях (`recipes`, `users`, `comments`)
3. Затем заново создайте миграции и примените их.

Но если вы хотите сохранить данные, то можно попробовать следующее:

1. Создать миграцию вручную, которая добавит поле `recipe` в модель `Comment` :
  - o Сначала убедитесь, что модель `Comment` в `comments/models.py` имеет поле `recipe` .
  - o Затем создайте миграцию:  
`python manage.py makemigrations comments --name add_recipe_field`
  - o Если Django говорит, что нет изменений, то можно создать пустую миграцию и вручную добавить операцию:  
`python manage.py makemigrations comments --empty --name add_recipe_field`
  - o Затем отредактируйте файл миграции, добавив операцию `AddField` .
2. Примените миграцию.

Однако, учитывая ошибку, я `suspect`ую, что миграции для `comments` не были применены.

Поэтому давайте проверим:

`bash`

```
python manage.py showmigrations comments
```

Если миграции не применены (отмечены как [ ]), примените их:

`bash`

```
python manage.py migrate comments
```

Если миграции применены (отмечены как [X]), но ошибка остается, то возможно, что в миграциях не было создания поля `recipe_id` . Тогда нужно проверить файлы миграций.

Давайте проверим, что в последней миграции `comments` есть поле `recipe` .

Если нет, то удалите миграции `comments` и создайте заново.

Я рекомендую, если данные не критичны, удалить базу данных и все миграции и начать заново:

`bash`

```
# Удаляем базу данных  
rm db.sqlite3
```

```
# Удаляем все миграции (кроме __init__.py) в каждом приложении
rm recipes/migrations/0*.py
rm users/migrations/0*.py
rm comments/migrations/0*.py

# Создаем миграции
python manage.py makemigrations

# Применяем миграции
python manage.py migrate

# Создаем суперпользователя
python manage.py createsuperuser
```

Затем запустите сервер и проверьте.

Если вы не хотите удалять базу данных, то давайте попробуем применить миграции для `comments` вручную, но если миграции уже отмечены как примененные, но поля нет, то это сложнее.

Другой вариант - откатить миграции для `comments` и применить их заново:

```
bash

# Откатываем миграции для comments до нуля
python manage.py migrate comments zero

# Затем снова применяем
python manage.py migrate comments
```

Но это может не сработать, если другие приложения зависят от `comments`.

Поэтому, если вы не против, я рекомендую первый вариант с удалением базы данных и миграций, так как проект еще в разработке.

Выполните эти шаги и сообщите о результате.

Ошибка указывает, что в базе данных отсутствует столбец `recipe_id` в таблице `comments_comment`. Это означает, что миграции для модели `Comment` не были применены правильно.

Давайте решим эту проблему:

## Шаг 1: Проверим текущее состояние миграций

```
bash
```

```
python manage.py showmigrations
```

## Шаг 2: Создадим и применим миграции для приложения comments

bash

```
# Создаем миграции для comments
python manage.py makemigrations comments
```

```
# Применяем все миграции
python manage.py migrate
```

## Шаг 3: Если проблема сохраняется

Возможно, нужно пересоздать миграции для всех приложений:

bash

```
# Удаляем файл базы данных и все миграции
del db.sqlite3
rmdir /s recipes\migrations
mkdir recipes\migrations
echo "" > recipes\migrations\__init__.py
```

```
rmdir /s comments\migrations
mkdir comments\migrations
echo "" > comments\migrations\__init__.py
```

```
rmdir /s users\migrations
mkdir users\migrations
echo "" > users\migrations\__init__.py
```

```
# Создаем новые миграции
python manage.py makemigrations
python manage.py migrate
```

```
# Создаем суперпользователя заново
python manage.py createsuperuser
```

## Шаг 4: Проверим модель Comment

Убедитесь, что в `comments/models.py` правильная модель:

```
python

from django.db import models
from django.contrib.auth.models import User
from recipes.models import Recipe # Импорт должен быть таким!

class Comment(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='comments', on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ['-created_at']

    def __str__(self):
        return f"Comment by {self.author.username} on {self.recipe.title}"
```

## Шаг 5: Альтернативное решение - сброс базы данных

Если ничего не помогает, давайте полностью пересоздадим базу:

```
bash

# Останавливаем сервер (Ctrl+C)

# Удаляем базу данных
del db.sqlite3

# Создаем миграции заново
python manage.py makemigrations
python manage.py migrate

# Создаем суперпользователя
python manage.py createsuperuser

# Запускаем сервер
python manage.py runserver
```

## Шаг 6: Проверка через Django shell

После применения миграций проверьте:

```
bash
```

```
python manage.py shell
```

```
python
```

```
from comments.models import Comment
from recipes.models import Recipe
from django.contrib.auth.models import User

# Проверяем, что модели работают
print(Comment.objects.all())
print(Recipe.objects.all())
print(User.objects.all())
exit()
```

**Пожалуйста, выполните сначала Шаг 1 и Шаг 2 и покажите результат команд showmigrations и создания миграций для comments.**

Это поможет понять, в каком состоянии находятся миграции и что именно нужно исправить.

### 7.1 Базовый шаблон (templates/base.html)

мне тут нужно создавать директорию templates, или создать python package?

Thought for 46 seconds ▾

⊗ Мы создаем директорию templates в корне проекта (рядом с manage.py). Внутри нее можно будет разместить базовый шаблон и другие общие шаблоны.

Однако, в настройках Django нужно указать, где искать шаблоны. В файле settings.py уже должна быть настроена папка templates:

В settings.py в разделе TEMPLATES есть 'DIRS': [os.path.join(BASE\_DIR, 'templates')]

Если этой директории нет, то создадим ее.

Последовательность:

1. Создаем папку templates в корне проекта (там, где manage.py).
2. Внутри папки templates создаем файл base.html .
3. В настройках settings.py проверяем, что в TEMPLATES в 'DIRS' указана эта папка.

Пример структуры:

```
recipesAlmanah_project/
├── manage.py
├── templates/ ← создаем эту папку
│   └── base.html ← и этот файл
├── recipes/
├── comments/
└── recipesAlmanah_project/
```

4. В файл `base.html` помещаем код базового шаблона.

Давайте проверим настройки в `settings.py`:

```
TEMPLATES = [
{
...
'DIRS': [os.path.join(BASE_DIR, 'templates')],
...
},
]
```

5. Убедимся, что переменная `BASE_DIR` определена как корневая директория проекта.

Обычно `BASE_DIR` определяется как:

```
BASE_DIR = Path(file).resolve().parent.parent
```

6. Если вы используете старые настройки, то может быть так:

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(file)))
```

После создания папки и файла, можно наполнять `base.html` содержимым.

Пример содержимого для `base.html` (из предыдущего обсуждения):

```
<!DOCTYPE html><html lang="ru"> <head> <meta charset="UTF-8"> <meta
name="viewport" content="width=device-width, initial-scale=1.0"> <title>{% 
block title %}Кулинарный сайт{% endblock %}</title> <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"> <style> .recipe-card { transition: transform 0.2s; }
.recipe-card:hover { transform: translateY(-5px); } </style> </head> <body>
<nav class="navbar navbar-expand-lg navbar-dark bg-success"> <div
class="container"> <a class="navbar-brand" href="{% url 'home' %}">Кулинарный
сайт</a> <div class="navbar-nav ms-auto"> {% if user.is_authenticated %} <a
class="nav-link" href="{% url 'recipe-create' %}">Добавить рецепт</a> <a
class="nav-link" href="{% url 'profile' %}">Профиль</a> <a class="nav-link"
href="{% url 'logout' %}">Выйти</a> {% else %} <a class="nav-link" href="{%
url 'login' %}">Войти</a> <a class="nav-link" href="{% url 'register'
%}">Регистрация</a> {% endif %} </div> </div> </nav>
```

```
text

<div class="container mt-4">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    {% block content %}{% endblock %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

</body> </html>
```

7. Убедитесь, что в проекте есть приложение `users` и `recipes` с соответствующими URL-адресами (например, 'home', 'recipe-create', 'profile', 'login', 'logout', 'register').

Если каких-то URL пока нет, можно временно убрать ссылки или заменить на существующие.

После этого можно создавать другие шаблоны, которые наследуются от [base.html](#).

Например, для главной страницы создадим `templates/home.html`:

```
{% extends 'base.html' %}

{% block content %}

<h1>Главная страница</h1> {% endblock %}
```

И в представлении (view) для главной страницы указать этот шаблон.

Важно: в настройках `settings.py` в `INSTALLED_APPS` должны быть добавлены приложения, чтобы Django искал шаблоны и в их папках (если они есть).

Обычно каждое приложение может иметь свою папку `templates` внутри приложения, но базовый шаблон мы вынесли в общую папку.

Если вы хотите использовать общую папку `templates`, то именно так и делаем.

Если возникнут проблемы с отображением шаблонов, проверьте настройки `TEMPLATES` в `settings.py` и убедитесь, что папка `templates` создана в правильном месте.

Нужно создать **директорию (папку)** `templates`, а не Python package.

## Создание директории `templates`:

## 1. Создайте папку templates в корне проекта:

bash

```
# В PowerShell из корневой директории проекта (где manage.py)
mkdir templates
```

## 2. Проверьте структуру:

Должно получиться:

text

```
recipesAlmanah_project/
└── manage.py
└── templates/           ← Новая папка
    ├── recipes/
    ├── comments/
    ├── users/
    └── recipesAlmanah_project/
        ├── settings.py
        └── ...
```

## 3. Создайте файл base.html внутри templates:

bash

```
# Создайте файл base.html
echo "" > templates\base.html
```

## 4. Настройте пути к шаблонам в settings.py:

Убедитесь, что в recipesAlmanah\_project/settings.py есть правильные настройки:

python

```
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

TEMPLATES = [
    {
```

```

'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [os.path.join(BASE_DIR, 'templates')], # ← эта строка важна
'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
},
],
]

```

## 5. Добавьте содержимое в base.html:

templates/base.html:

```

html

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .recipe-card {
            transition: transform 0.2s;
        }
        .recipe-card:hover {
            transform: translateY(-5px);
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="{% url 'home' %}">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                {% if user.is_authenticated %}
                    <a class="nav-link" href="{% url 'recipe-create' %}">Добавить рецепт</a>
                    <a class="nav-link" href="{% url 'profile' %}">Профиль</a>
                
```

```

        <a class="nav-link" href="{% url 'logout' %}">Выйти</a>
    {% else %}
        <a class="nav-link" href="{% url 'login' %}">Войти</a>
        <a class="nav-link" href="{% url 'register' %}">Регистрация</
    a>
    {% endif %}
</div>
</div>
</nav>

<div class="container mt-4">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    {% block content %}{% endblock %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## 6. Проверьте работу:

bash

```
python manage.py runserver
```

Если все настроено правильно, шаблоны должны работать.

### Важно:

- Папка **templates** - это обычная директория для HTML файлов
- **Python package** - это папка с файлом `__init__.py`, используется для модулей Python
- Для шаблонов Django нужна обычная директория, а не Python package

**Создайте просто папку `templates` и все будет работать!**

## Thought for 43 seconds ▾

❖ Мы создали базовый шаблон, но чтобы проверить его работу, нам нужно создать хотя бы одну страницу, которая будет использовать этот шаблон.

Давайте создадим простую главную страницу.

Шаг 1: Создадим представление для главной страницы в приложении `recipes`.

Шаг 2: Настроим URL для главной страницы.

Шаг 3: Создадим шаблон для главной страницы, который наследует `base.html`.

Шаг 4: Проверим, отображается ли страница.

Однако, если мы просто создадим шаблон `base.html`, но не создадим ни одной страницы, которая его использует, то мы не увидим его в работе.

Давайте быстро создадим главную страницу.

1. В файле `recipes/views.py` добавим:

```
from django.shortcuts import render
from .models import Recipe

def home(request):
    recipes = Recipe.objects.all()
    context = {
        'recipes': recipes
    }
    return render(request, 'recipes/home.html', context)
```

2. Создадим файл `templates/recipes/home.html`:

```
{% extends 'base.html' %}

{% block title %}Главная страница{% endblock %}

{% block content %}



# Последние рецепты

 <div class="row"> {% for recipe in recipes %} <div class="col-md-4"> <div class="card recipe-card mb-4"> <div class="card-body">
<h5 class="card-title">{{ recipe.title }}</h5> <p class="card-text">{{ recipe.description|truncatewords:20 }}</p> <a href="#" class="btn btn-primary">Подробнее</a> </div> </div> </div> {% empty %} <p>Пока нет рецептов.</p> {% endfor %} </div> {% endblock %}
```

3. Настроим URL в `recipes/urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
]
```

4. И убедимся, что главный urls.py проекта включает URLs приложения recipes.

В recipesAlmanah\_project/urls.py:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
]
```

Теперь запустим сервер и перейдем на <http://127.0.0.1:8000/>

Если мы видим главную страницу с рецептами (или сообщение, что рецептов нет), то шаблон **base.html** работает.

Однако, если у вас еще нет рецептов, то можно создать несколько через админку.

Но если мы не хотим пока создавать представления, то можно просто проверить, что шаблон **base.html** загружается, создав временную страницу.

Альтернативно, можно проверить через админку, но админка использует свои шаблоны.

Давайте создадим простейшую страницу для проверки **base.html**.

1. В views.py приложения recipes (или любого другого) создадим временное представление:

```
def test_page(request):
    return render(request, 'test.html')
```

2. Создадим templates/**test.html**:

```
{% extends 'base.html' %}

{% block content %}

<h1>Тестовая страница</h1> <p>Если вы видите этот текст, то базовый шаблон
работает!</p> {% endblock %}
```

3. Добавим URL для тестовой страницы в urls.py:

```
path('test/', views.test_page, name='test'),
```

4. Перейдем на <http://127.0.0.1:8000/test/>

Если мы видим навигационную панель и содержимое тестовой страницы, то все работает.

Но поскольку вы уже запускали сервер и получали ошибку с комментариями, давайте сначала решим проблему с миграциями, чтобы не было ошибок.

Однако, если вы хотите просто проверить шаблон, то можно временно отключить приложение comments в settings.py, закомментировав его в INSTALLED\_APPS, чтобы избежать ошибки с комментариями.

Но правильнее будет применить миграции для comments.

Давайте вернемся к проблеме с миграциями для comments.

Выполните:

```
python manage.py makemigrations comments  
python manage.py migrate
```

Если это не помогает, то давайте пересоздадим миграции для comments, как я описывал ранее.

После решения проблемы с миграциями, мы сможем проверить работу шаблонов.

Итак, план:

1. Решаем проблему с миграциями для comments.
2. Создаем тестовую страницу для проверки [base.html](#).
3. Проверяем.

Хотите, чтобы я помог с миграциями для comments, или вы уже решили эту проблему?

## Как проверить работу шаблонов и сайта:

### 1. Проверка базового шаблона:

Создайте простой тестовый шаблон для проверки:

[templates/test.html](#):

```
html
```

```
{% extends 'base.html' %}  
  
{% block title %}Тестовая страница{% endblock %}  
  
{% block content %}  
<div class="row">  
    <div class="col-md-12">  
        <h1>Тестовая страница</h1>  
        <p>Если вы видите это сообщение, значит базовый шаблон работает!</p>  
  
        <div class="alert alert-success">  
            <h4>Проверка элементов:</h4>
```

```
<ul>
    <li>Навигационная панель сверху /</li>
    <li>Bootstrap стили /</li>
    <li>Контейнер с отступами /</li>
</ul>
</div>

<div class="card">
    <div class="card-body">
        <h5 class="card-title">Тестовая карточка рецепта</h5>
        <p class="card-text">Это пример карточки рецепта с hover-эффектом.</p>
        <button class="btn btn-primary">Тестовая кнопка</button>
    </div>
</div>
</div>
{% endblock %}
```

## 2. Создайте временное представление для теста:

recipes/views.py (добавьте в конец файла):

```
python

from django.shortcuts import render
from django.http import HttpResponseRedirect

def test_view(request):
    """Временное представление для теста шаблонов"""
    return render(request, 'test.html')
```

## 3. Добавьте временный URL:

recipes/urls.py:

```
python

from django.urls import path
from . import views

urlpatterns = [
    path('', views.test_view, name='test'), # временный URL для теста
```

```
# остальные URL будут добавлены позже  
]
```

## 4. Запустите сервер и проверьте:

bash

```
python manage.py runserver
```

Откройте в браузере: <http://127.0.0.1:8000/>

## 5. Что должно отобразиться:

- Зеленая навигационная панель с логотипом
- Контейнер с белым фоном и отступами
- Тестовое сообщение
- Карточка с hover-эффектом (при наведении поднимается)
- Кнопка с Bootstrap стилями

## 6. Если есть ошибки, проверьте:

Проверка настроек templates в settings.py:

python

```
# Убедитесь, что в TEMPLATES есть эта строка:  
'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

Проверка структуры папок:

bash

```
# В PowerShell  
Get-ChildItem -Recurse | Where-Object {$_ .Name -like "*.html" -or $_ .Name -eq "te  
mplates"}
```

## 7. Дополнительные проверки:

Проверка статических файлов (CSS/JS):

```
html

<!-- Добавьте в base.html для отладки -->
<div style="display: none;" id="debug-info">
    Bootstrap loaded: <span id="bootstrap-check">Checking...</span>
</div>

<script>
    // Проверка загрузки Bootstrap
    if (typeof bootstrap !== 'undefined') {
        document.getElementById('bootstrap-check').textContent = '✓ Loaded';
    } else {
        document.getElementById('bootstrap-check').textContent = '✗ Not loaded';
    }
</script>
```

Запустите проверку Django:

bash

```
python manage.py check
```

## 8. Если все работает, можно удалить тестовую страницу:

После успешной проверки:

1. Удалите файл templates/test.html
2. Удалите временное представление test\_view из views.py
3. Настройте правильные URL для вашего приложения

Запустите сервер и проверьте - что видите при переходе на <http://127.0.0.1:8000/>?

```
django.core.exceptions.ImproperlyConfigured: The included URLconf
'<module 'users.urls' from 'D:\\\\RecipesAI
manah\\\\recipesAlmanah_project\\\\users\\\\urls.py'>' does not appear to
have any patterns in it. If you see the
'urlpatterns' variable with valid patterns in the file then the issue is
probably caused by a circular import.
```

Thought for 59 seconds ▾

⊗ Мы получаем ошибку о том, что в URLconf приложения 'users' нет шаблонов. Это может быть вызвано тем, что файл users.urls.py не содержит переменную urlpatterns или она пуста.

Давайте проверим и исправим:

1. Убедимся, что в `users/urls.py` есть `urlpatterns` с хотя бы одним паттерном.
2. Если мы еще не реализовали URLs для users, то можно временно добавить пустой список, но лучше добавить хотя бы один URL.

Однако, возможно, мы еще не создавали URLs для приложения users. В таком случае, мы можем временно удалить включение этого URLconf из главного `urls.py` или добавить минимальные URL.

Рассмотрим оба варианта.

Вариант 1: Удалить включение `users.urls` из главного `urls.py`, если мы еще не готовы его использовать.

Вариант 2: Создать минимальный `urls.py` в приложении users.

Давайте воспользуемся вариантом 2, так как мы планируем иметь функционал пользователей.

В папке `users` создадим файл `urls.py` (если его нет) и добавим:

```
python

from django.urls import path
from . import views

app_name = 'users'

urlpatterns = [
    # Пока оставим пустым, но чтобы не было ошибки, добавим временный путь
    # path('', views.profile, name='profile'),
]
```

Но поскольку у нас еще нет представления, мы можем получить другую ошибку. Поэтому, если мы не готовы, то лучше закомментировать включение `users.urls` в главном `urls.py`.

В главном `urls.py` (в папке проекта) у нас должно быть что-то вроде:

```
python

from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    path('users/', include('users.urls')), # если эта строка есть, то она вызывает ошибку
```

```
    path('comments/', include('comments.urls')),
]
```

Если мы не готовы к настройке URLs для users, то можно закомментировать строку с `users.urls` :

`python`

```
# path('users/', include('users.urls')),
```

Или же, если мы хотим оставить, то нужно убедиться, что в `users(urls.py` есть хотя бы один паттерн.

Поскольку в ошибке указано, что в `users.urls` нет паттернов, давайте временно добавим пустой список `urlpatterns` в `users.urls.py` :

`python`

```
from django.urls import path

urlpatterns = [
    # Пока пусто, но это уберет ошибку
]
```

Однако, Django не допускает пустого `urlpatterns` при включении через `include`, поэтому даже пустой список допустим.

Но на самом деле, если мы оставим пустой список, то при переходе по `/users/` мы получим ошибку 404, что нормально на данном этапе.

Давайте сделаем так:

1. Создадим в `users.urls.py` пустой `urlpatterns`.
2. Позже, когда будем реализовывать аутентификацию и профили, добавим нужные пути.

Альтернативно, мы можем удалить включение `users.urls` из главного `urls.py` до тех пор, пока не будем готовы.

Выберем первый вариант, чтобы не было ошибки при запуске.

Шаги:

1. Создаем файл `users.urls.py` (если его нет) и пишем:

`python`

```
from django.urls import path
```

```
urlpatterns = [  
]
```

2. Убедимся, что в главном urls.py есть включение users.urls .

3. Запустим сервер снова.

Если ошибка исчезнет, то все хорошо.

Однако, обратите внимание, что в ошибке также упоминается circular import (циклический импорт). Если у нас в users(urls.py есть импорт представлений, которые импортируют что-то из других модулей, что в итоге замыкается на users.urls , то может возникнуть циклический импорт. Но пока у нас нет представлений, это маловероятно.

Давайте проверим главный urls.py и исправим включение users.urls .

Если вы не хотите пока включать URLs для users, то просто закомментируйте эту строку.

Я рекомендую пока закомментировать, чтобы избежать ошибки, и раскомментировать, когда будем добавлять функционал пользователей.

Итак, в culinary\_project/urls.py (или recipesAlmanah\_project/urls.py - главный urls.py) закомментируем строку с users.urls :

```
python  
  
from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', include('recipes.urls')),  
    # path('users/', include('users.urls')), # закомментировано до готовности  
    path('comments/', include('comments.urls')),  
]
```

Аналогично, если в comments/urls.py тоже нет паттернов, то мы получим такую же ошибку. Поэтому, если мы еще не готовы с comments, то тоже закомментируем.

Я вижу, что в ошибке указано только про users.urls , но если после исправления появится такая же ошибка для comments.urls , то поступим аналогично.

Итак, повторим:

1. Закомментировать в главном urls.py включение users.urls и comments.urls , если они еще не реализованы.
2. Оставить только те include, которые уже имеют urlpatterns.

Поскольку мы только начинаем, оставим пока только recipes.urls и admin .

После исправления главного urls.py, он должен выглядеть так:

```
python

from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    # path('users/', include('users.urls')), # пока не готово
    # path('comments/', include('comments.urls')), # пока не готово
]
```

Тогда ошибка должна исчезнуть.

Теперь запустите сервер снова и проверьте.

Если вы видите тестовую страницу, то все работает.

После этого мы можем постепенно добавлять URL в приложения users и comments, раскомментируя эти строки и добавляя реальные urlpatterns в их urls.py.

Ошибка указывает на проблему с файлом `users/urls.py`. Давайте исправим это:

## Решение проблемы с `users/urls.py`

### 1. Проверьте наличие и содержимое `users/urls.py`:

Убедитесь, что файл существует и содержит хотя бы базовые URL-паттерны:

`users/urls.py`:

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    # Временные URL для теста – замените позже на реальные
    path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.htm
l'), name='logout'),
```

```
    path('profile/', views.profile, name='profile'),
]
```

## 2. Создайте базовое представление для профиля:

users/views.py:

```
python

from django.shortcuts import render
from django.contrib.auth.decorators import login_required

@login_required
def profile(request):
    return render(request, 'users/profile.html', {'user': request.user})
```

## 3. Временно отключите проблемные URL (если нужно):

Если хотите быстро проверить работу, закомментируйте проблемные строки в главном urls.py:

recipesAlmanah\_project/urls.py:

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    # path('users/', include('users.urls')), # Временно закомментировать
    # path('comments/', include('comments.urls')), # Временно закомментировать
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## 4. Альтернативное решение - создайте минимальные URL:

comments/urls.py (если файла нет, создайте его):

```
python

from django.urls import path
from . import views

app_name = 'comments'

urlpatterns = [
    # Пока оставьте пустым или добавьте временные URL
    # path('', views.comment_list, name='comment-list'),
]

```

## 5. Проверьте структуру всех URL-файлов:

Убедитесь, что все файлы urls.py существуют и содержат urlpatterns :

```
bash

# Проверка наличия файлов
Get-ChildItem *\urls.py

# Должны быть:
# - recipesAlmanah_project/urls.py (главный)
# - recipes/urls.py
# - users/urls.py
# - comments/urls.py
```

## 6. Запустите проверку:

```
bash

python manage.py check
python manage.py runserver
```

## 7. Если все еще ошибка, временно упростите главный urls.py:

recipesAlmanah\_project/urls.py (минимальная версия для теста):

```
python

from django.contrib import admin
from django.urls import path
from recipes import views
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.test_view, name='home'), # Ваш тестовый view
]
```

После того как шаблоны заработают, вы сможете постепенно добавлять обратно URL приложений `users` и `comments`.

Попробуйте сначала вариант 3 (временное отключение `users` и `comments` URLs) - это самый быстрый способ проверить работу шаблонов.

Так, давай по другому, вот я написал 7 этап, как я могу проверить, что все работает(учитывая что urls.py не заполнены в python package comments и users, эти строки я могу закоментировать)

Thought for 19 seconds ▾

💡 Мы временно закомментируем включение URL-приложений `users` и `comments` в главном `urls.py`, чтобы избежать ошибки.

Затем создадим временное представление в `recipes/views.py`, которое будет использовать `base.html`.

Также создадим тестовый шаблон `test.html` в папке `templates`.

Шаги:

1. Закомментировать в `recipesAlmanah_project/urls.py` строки, которые подключают `users.urls` и `comments.urls`.
2. Создать временное представление в `recipes/views.py`, которое будет отображать тестовую страницу.
3. Создать тестовый шаблон `test.html`, расширяющий `base.html`.
4. Настроить URL для тестового представления в `recipes/urls.py`.
5. Запустить сервер и проверить.

Давайте начнем:

1. В файле `recipesAlmanah_project/urls.py` закомментируем проблемные строки:

Было:

```
path('users/', include('users.urls')),
path('comments/', include('comments.urls')),
```

Стало:

```
# path('users/', include('users.urls')),
# path('comments/', include('comments.urls')),
```

2. В `recipes/views.py` добавим временное представление:

```
def test_view(request):
    return render(request, 'test.html')
```

3. Создадим templates/test.html с содержимым, которое расширяет base.html.

4. В recipes/urls.py добавим путь для тестового представления:

```
from .views import test_view
urlpatterns = [
    path('', test_view, name='test'),
]
```

5. Запустим сервер и перейдем на <http://127.0.0.1:8000/>

Если мы видим тестовую страницу с навигацией и стилями, то base.html работает.

После проверки мы можем удалить тестовое представление и шаблон, а затем настроить правильные URL.

Однако, если вы хотите оставить тестовую страницу до тех пор, пока не настроите все приложения, это тоже вариант.

Давайте напишем код для каждого шага.

Отлично! Давайте проверим работу шаблонов простым способом, временно закомментировав проблемные URLs.

## Быстрая проверка работы шаблонов:

### 1. Временно упростим главный urls.py:

recipesAlmanah\_project/urls.py:

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from recipes import views # добавляем импорт

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.test_view, name='home'), # временный главный URL
    # path('users/', include('users.urls')), # ЗАКОММЕНТИРОВАТЬ
    # path('comments/', include('comments.urls')), # ЗАКОММЕНТИРОВАТЬ
]
```

```
if settings.DEBUG:  
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

**2. Добавим тестовое представление в recipes/views.py:**

**recipes/views.py** (добавьте в конец файла):

```
python

def test_view(request):
    """Временное представление для проверки шаблонов"""
    context = {
        'title': 'Тестовая страница',
        'message': 'Шаблоны работают корректно!'
    }
    return render(request, 'test.html', context)
```

### **3. Создайте тестовый шаблон:**

## templates/test.html:

```
html

{% extends 'base.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

<div class="row">
    <div class="col-md-8 mx-auto">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">Проверка работы шаблонов</h4>
            </div>
            <div class="card-body">
                <div class="alert alert-success">
                    <h5>✓ Успех!</h5>
                    <p class="mb-0">{{ message }}</p>
                </div>

                <h5>Что проверяем:</h5>
                <ul>
                    <li>Навигационная панель (navbar)</li>
                    <li>Bootstrap стили</li>
                </ul>
            </div>
        </div>
    </div>
</div>
```

```

<li>Базовый layout</li>
<li>Блоки content и title</li>
</ul>

<div class="row mt-4">
    <div class="col-md-4">
        <div class="card recipe-card">
            <div class="card-body">
                <h6>Тест карточки 1</h6>
                <p>Hover-эффект должен работать</p>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card recipe-card">
                <div class="card-body">
                    <h6>Тест карточки 2</h6>
                    <p>Bootstrap сетка</p>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card recipe-card">
                <div class="card-body">
                    <h6>Тест карточки 3</h6>
                    <p>Цвета и отступы</p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}

```

## 4. Запустите проверку:

bash

```
python manage.py runserver
```

## 5. Откройте в браузере: <http://127.0.0.1:8000/>

## 6. Что должно отобразиться (признаки успеха):

- Зеленая навигационная панель сверху с логотипом
- Белая область контента с отступами
- Сообщение об успехе в зеленом alert
- Три карточки которые при наведении немного поднимаются (hover-эффект)
- Адаптивная верстка - при изменении размера окна карточки перестраиваются

## 7. Если видите ошибку, проверьте:

bash

```
# Проверка Django
python manage.py check

# Проверка структуры
Get-ChildItem templates\
```

## 8. После успешной проверки можно:

- Удалить тестовые файлы
- Или оставить их как основную страницу до реализации функционала
- Постепенно раскомментировать URLs приложений users и comments

Запустите сервер и скажите, что видите при переходе на <http://127.0.0.1:8000/>

Если есть ошибки - покажите полный текст ошибки, и мы быстро ее исправим!



image.png

PNG 133.26KB

Thought for 9 seconds ▾

💡 Мы получаем ошибку потому, что в шаблоне `base.html` есть ссылки на URL с именами 'recipe-create', 'profile', 'logout' и т.д., но эти URL еще не определены в нашем проекте.

Поскольку мы только проверяем шаблоны, мы можем временно убрать эти ссылки или заменить их на временные.

Давайте модифицируем `base.html`, чтобы убрать проблемные ссылки, и оставим только базовую навигацию.

Альтернативно, мы можем определить временные URL для этих ссылок, но это займет больше времени.

Предлагаю временно изменить [base.html](#), чтобы убрать ссылки, которые вызывают ошибку.

Изменим навигационную панель в [base.html](#) на простую, без условных блоков и без ссылок на несуществующие URL.

После того как мы убедимся, что шаблон работает, мы можем постепенно добавлять функционал.

Вот временный вариант [base.html](#):

Отлично! Ошибка понятна - в шаблоне `base.html` есть ссылки на URL-паттерны, которые еще не созданы. Давайте быстро это исправим:

## Быстрое решение для тестирования шаблонов

### 1. Временно упростим шаблон [base.html](#):

`templates/base.html` (замените навигационную панель на упрощенную версию):

```
html

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .recipe-card {
            transition: transform 0.2s;
        }
        .recipe-card:hover {
            transform: translateY(-5px);
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="/">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                <!-- ВРЕМЕННО УПРОЩЕННАЯ НАВИГАЦИЯ -->
```

```

        <a class="nav-link" href="/admin/">Админка</a>
        <a class="nav-link" href="#">Войти</a>
        <a class="nav-link" href="#">Регистрация</a>
    </div>
</div>
</nav>

<div class="container mt-4">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    {% block content %}{% endblock %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## 2. Или создайте временный шаблон без наследования:

`templates/simple_test.html` (новый файл для быстрой проверки):

```

html

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Тест шаблонов</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="/">Кулинарный сайт</a>
        </div>
    </nav>

```

```

<div class="container mt-4">
    <div class="alert alert-success">
        <h4>✓ Шаблоны работают!</h4>
        <p>Bootstrap загружен корректно, навигационная панель отображается.</p>
    </div>

    <div class="row">
        <div class="col-md-4">
            <div class="card recipe-card" style="transition: transform 0.2s;">
                <div class="card-body">
                    <h5>Тест карточки 1</h5>
                    <p>При наведении должна подниматься</p>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card recipe-card" style="transition: transform 0.2s;">
                <div class="card-body">
                    <h5>Тест карточки 2</h5>
                    <p>Bootstrap сетка работает</p>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card recipe-card" style="transition: transform 0.2s;">
                <div class="card-body">
                    <h5>Тест карточки 3</h5>
                    <p>Стили применяются корректно</p>
                </div>
            </div>
        </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

### 3. Обновите представление для использования простого шаблона:

## recipes/views.py:

```
python

def test_view(request):
    """Временное представление для проверки шаблонов"""
    return render(request, 'simple_test.html') # Используем простой шаблон
```

## 4. Запустите сервер и проверьте:

bash

```
python manage.py runserver
```

Теперь при переходе на <http://127.0.0.1:8000/> вы должны увидеть работающую страницу без ошибок.

## 5. Что мы проверяем:

- Загрузка Bootstrap CSS
- Навигационная панель
- Сетка Bootstrap (колонки)
- Карточки с hover-эффектом
- Отсутствие ошибок URL

После успешной проверки вы можете:

1. Постепенно добавлять настоящие URL-паттерны
2. Восстановить полную версию [base.html](#)
3. Создать реальные представления для всех функций

Попробуйте этот вариант и скажите, что видите теперь!

эта ошибка уже появилась после добавленія `test.html`

Thought for 30 seconds ▾

💡 Мы получили ошибку потому, что в базовом шаблоне `base.html` есть ссылки на именованные URL-паттерны, которые еще не определены в нашем проекте. В частности, это `recipe-create`, `profile`, `logout` и другие.

Чтобы быстро проверить работу шаблонов, мы можем временно убрать эти ссылки или заменить их на простые ссылки без использования `{% url %}`.

Давайте создадим временный базовый шаблон без этих ссылок или закомментируем их.

Я предлагаю два варианта:

Вариант 1: Закомментировать проблемные строки в `base.html` и заменить их на простые ссылки или оставить только те, которые уже есть.

Вариант 2: Создать новый базовый шаблон без этих ссылок.

Я предлагаю Вариант 1, так как он быстрее.

Изменим `templates/base.html`:

В навигационной панели закомментируем или удалим блок с ссылками, которые используют `{% url %}` и поставим простые ссылки для теста.

Например, заменим:

Было:

```
html

<div class="navbar-nav ms-auto">
    {% if user.is_authenticated %}
        <a class="nav-link" href="{% url 'recipe-create' %}">Добавить рецепт</a>
    >
        <a class="nav-link" href="{% url 'profile' %}">Профиль</a>
        <a class="nav-link" href="{% url 'logout' %}">Выйти</a>
    {% else %}
        <a class="nav-link" href="{% url 'login' %}">Войти</a>
        <a class="nav-link" href="{% url 'register' %}">Регистрация</a>
    {% endif %}
</div>
```

На временную версию без `{% url %}`:

```
html

<div class="navbar-nav ms-auto">
    <a class="nav-link" href="/admin/">Админка</a>
    <a class="nav-link" href="#">Войти</a>
    <a class="nav-link" href="#">Регистрация</a>
</div>
```

Тогда шаблон не будет пытаться разрешать именованные URL и ошибка пропадет.

Давайте сделаем это и проверим.

Если вы хотите, чтобы я показал, как исправить именно этот шаблон, то вот код:

Замените всю навигационную панель в `base.html` на эту временную:

`html`

```
<nav class="navbar navbar-expand-lg navbar-dark bg-success">
  <div class="container">
    <a class="navbar-brand" href="/">Кулинарный сайт</a>
    <div class="navbar-nav ms-auto">
      <a class="nav-link" href="/admin/">Админка</a>
      <a class="nav-link" href="#">Войти</a>
      <a class="nav-link" href="#">Регистрация</a>
    </div>
  </div>
</nav>
```

Затем сохраните и обновите страницу.

После этого шаблон должен отображаться без ошибок.

Если вы хотите, чтобы я помог вам затем настроить все URL-адреса, дайте знать.

А пока давайте убедимся, что шаблон отображается.

Пожалуйста, попробуйте и сообщите результат.

Понял! Ошибка возникает потому что в `base.html` есть ссылки на URL-имена, которые еще не созданы в нашем проекте. Давайте быстро это исправим:

## Решение: Создадим упрощенную версию `base.html`

### 1. Создайте новый упрощенный базовый шаблон:

`templates/base_simple.html`:

`html`

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{% block title %}Кулинарный сайт{% endblock %}</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
```

```

.recipe-card {
    transition: transform 0.2s;
}
.recipe-card:hover {
    transform: translateY(-5px);
}

```

</style>

</head>

<body>

```

<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="/">Кулинарный сайт</a>
        <div class="navbar-nav ms-auto">
            <!-- Упрощенные ссылки без {% url %} -->
            <a class="nav-link" href="/admin/">Админка</a>
            <a class="nav-link" href="#">Войти</a>
            <a class="nav-link" href="#">Регистрация</a>
        </div>
    </div>
</nav>

<div class="container mt-4">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    {% block content %}{% endblock %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

```

</body>

</html>

**2. Обновите `test.html` чтобы использовать упрощенный базовый шаблон:**

templates/**test.html**:

html

```
{% extends 'base_simple.html' %}

{% block title %}Тестовая страница{% endblock %}

{% block content %}


#### ✓ Шаблоны работают!



##### Успешная проверка!



Базовый шаблон, Bootstrap и стили загружены корректно.



<h5>Что проверено:</h5>



- Навигационная панель с Bootstrap
- Базовый layout и контейнер
- CSS стили и hover-эффекты
- Блоки наследования шаблонов



###### Карточка 1



Hover-эффект ✓



###### Карточка 2



Сетка Bootstrap ✓


```

```
<h6>Карточка 3</h6>
<p>Стили CSS √</p>
</div>
</div>
</div>
</div>
</div>
</div>
<% endblock %}
```

### 3. Убедитесь что представление использует test.html:

recipes/views.py:

```
python

def test_view(request):
    """Временное представление для проверки шаблонов"""
    return render(request, 'test.html')
```

### 4. Запустите сервер и проверьте:

bash

```
python manage.py runserver
```

Теперь при переходе на <http://127.0.0.1:8000/> ошибки быть не должно.

### 5. Что мы проверяем:

- Отсутствие ошибок NoReverseMatch
- Зеленая навигационная панель
- Карточки с hover-эффектом
- Bootstrap сетка и стили
- Наследование шаблонов работает

### 6. После успешной проверки вы можете:

1. Постепенно создавать реальные URL-паттерны
2. Заменить упрощенные ссылки на `{% url 'name' %}`
3. Восстановить полную версию `base.html`

**Попробуйте этот вариант – ошибка должна исчезнуть!**

давай так, напіші все шагі, чтобы добавіть `test.html`, и соответснно, его проверить

Thought for 24 seconds ▾

☒ Мы уже создали `test.html` и `base_simple.html`. Давайте пройдем все шаги для проверки:

1. Убедимся, что структура папок правильная.
2. Создадим упрощенный базовый шаблон (`base_simple.html`) и тестовый шаблон (`test.html`).
3. Настроим `views.py` для отображения `test.html`.
4. Настроим `urls.py` для отображения `test_view` на главной странице.
5. Запустим сервер и проверим.

Но, возможно, мы уже сделали некоторые шаги. Давайте проверим текущее состояние и выполним недостающие шаги.

Шаг 1: Проверка структуры папок

Убедитесь, что в корне проекта (рядом с `manage.py`) есть папка `templates`, а внутри нее файлы `base_simple.html` и `test.html`.

Шаг 2: Содержимое `base_simple.html` (уже написано выше) и `test.html` (тоже выше).

Шаг 3: В файле `recipes/views.py` должна быть функция `test_view`.

Шаг 4: В главном `urls.py` (`recipesAlmanah_project/urls.py`) должен быть путь, связывающий корневой URL с `test_view`.

Шаг 5: Запуск сервера и проверка.

Если что-то отсутствует, давайте создадим.

Однако, если вы хотите, чтобы я написал все шаги с нуля, то:

1. Создаем папку `templates` в корне проекта (если еще не создана).
2. В папке `templates` создаем два файла: `base_simple.html` и `test.html`.
3. В `base_simple.html` копируем упрощенный шаблон (без использования `{% url %}` для несуществующих имен).
4. В `test.html` копируем шаблон, который расширяет `base_simple.html`.
5. В приложении `recipes` в файле `views.py` добавляем функцию `test_view`.
6. В главном `urls.py` (`recipesAlmanah_project/urls.py`) добавляем путь для `test_view`.
7. Запускаем сервер и проверяем.

Давайте я напишу код для каждого шага:

Шаг 1: Создание папки templates (если нет)

```
mkdir templates
```

Шаг 2: Создание [base\\_simple.html](#) в папке templates

Шаг 3: Создание [test.html](#) в папке templates

Шаг 4: Редактирование `recipes/views.py`

Шаг 5: Редактирование `recipesAlmanah_project/urls.py`

Шаг 6: Запуск сервера

Теперь подробно:

[base\\_simple.html](#):

Отлично! Давайте пройдем все шаги для добавления [test.html](#) и его проверки:

## Шаг 1: Создание директории templates

`bash`

```
# Убедитесь, что находитесь в корне проекта (где manage.py)
mkdir templates
```

## Шаг 2: Создание упрощенного базового шаблона

[templates/base\\_simple.html](#):

`html`

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .recipe-card {
            transition: transform 0.2s;
        }
        .recipe-card:hover {
            transform: translateY(-5px);
        }
    </style>
```

```

        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="/">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                <a class="nav-link" href="/admin/">Админка</a>
                <a class="nav-link" href="#">Войти</a>
                <a class="nav-link" href="#">Регистрация</a>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-{{ message.tags }}">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}

        {% block content %}{% endblock %}
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## Шаг 3: Создание тестового шаблона

templates/test.html:

```

html

{% extends 'base_simple.html' %}

{% block title %}Тестовая страница{% endblock %}

{% block content %}
<div class="row">
    <div class="col-md-8 mx-auto">
        <div class="card">

```

```
<div class="card-header bg-success text-white">
    <h4 class="mb-0">✓ Шаблоны работают!</h4>
</div>
<div class="card-body">
    <div class="alert alert-success">
        <h5>Успешная проверка!</h5>
        <p class="mb-0">Базовый шаблон, Bootstrap и стили загружены корректно.</p>
    </div>

    <h5>Что проверено:</h5>
    <ul>
        <li>Навигационная панель с Bootstrap</li>
        <li>Базовый layout и контейнер</li>
        <li>CSS стили и hover-эффекты</li>
        <li>Блоки наследования шаблонов</li>
    </ul>

    <div class="row mt-4">
        <div class="col-md-4">
            <div class="card recipe-card">
                <div class="card-body text-center">
                    <h6>Карточка 1</h6>
                    <p>Hover-эффект ✓</p>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card recipe-card">
                <div class="card-body text-center">
                    <h6>Карточка 2</h6>
                    <p>Сетка Bootstrap ✓</p>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card recipe-card">
                <div class="card-body text-center">
                    <h6>Карточка 3</h6>
                    <p>Стили CSS ✓</p>
                </div>
            </div>
        </div>
    </div>
</div>
</div>
```

```
</div>
{% endblock %}
```

## Шаг 4: Добавление тестового представления

recipes/views.py (добавьте в конец файла):

```
python

from django.shortcuts import render

def test_view(request):
    """Временное представление для проверки шаблонов"""
    return render(request, 'test.html')
```

## Шаг 5: Настройка URL

recipesAlmanah\_project/urls.py (замените содержимое на):

```
python

from django.contrib import admin
from django.urls import path
from recipes import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.test_view, name='home'),
]
```

## Шаг 6: Проверка структуры проекта

Убедитесь, что структура выглядит так:

```
text

recipesAlmanah_project/
├── manage.py
├── templates/
│   ├── base_simple.html
│   └── test.html
└── recipes/
    └── views.py
```

```
└─ ...  
└─ recipesAlmanah_project/  
    ├─ settings.py  
    └─ urls.py
```

## Шаг 7: Проверка работы

bash

```
# Запустите сервер  
python manage.py runserver
```

## Шаг 8: Что должно отобразиться

Откройте <http://127.0.0.1:8000/> и проверьте:

- Зеленая навигационная панель сверху
- Карточка с сообщением об успехе
- Три карточки в ряд (при наведении поднимаются)
- Адаптивный дизайн (попробуйте изменить размер окна)
- Отсутствие ошибок в консоли браузера

## Шаг 9: Дополнительная проверка

Откройте инструменты разработчика в браузере (F12) и проверьте:

- Нет ошибок в консоли
- Bootstrap CSS загружен
- Стили применяются корректно

## Если есть проблемы:

### 1. Проверьте настройки templates в settings.py:

python

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
```

```
'DIRS': [os.path.join(BASE_DIR, 'templates')], # ← эта строка должна быть
      # ... остальные настройки
    },
]
```

## 2. Проверьте структуру командой:

bash

```
Get-ChildItem -Recurse | Where-Object {$_.Name -like "*.html"}
```

Выполните эти шаги по порядку и сообщите, что видите при переходе на  
<http://127.0.0.1:8000/>

так, ладно, объясните мне более подробно этап 8, учитя, что ubuntu не можем поставить и работаем на цштвщцы

Thought for 41 seconds ▾

⊗ Мы работаем на Windows, и у нас есть проект Django. Мы хотим проверить, что наш базовый шаблон и тестовый шаблон работают правильно. Давайте подробно разберем этап 8, учитывая, что мы используем Windows.

### Этап 8: Проверка работы в браузере

После выполнения шагов 1-7, мы запустили сервер с помощью `python manage.py runserver`. Теперь мы должны открыть браузер и перейти по адресу: <http://127.0.0.1:8000/>

Что мы должны увидеть:

#### 1. Зеленая навигационная панель вверху:

- Фон зеленый (класс `bg-success`)
- Текст "Кулинарный сайт" слева, который является ссылкой на главную страницу (ведет на "/")
- Справа ссылки: "Админка", "Войти", "Регистрация" (пока неактивные, кроме "Админка", которая ведет на страницу администратора)

#### 2. Основной контент:

- Белая карточка с зеленым заголовком, внутри которой:
  - Сообщение об успехе в зеленом alert-блоке
  - Список того, что проверено
  - Три карточки в ряд (при наведении на них курсора они должны плавно подниматься вверх)

Как проверить hover-эффект:

- Наведите курсор мыши на любую из трех карточек. Она должна немного подняться вверх (эффект `transform: translateY(-5px)` ).

**Как проверить адаптивность:**

- Измените размер окна браузера, сделав его уже. При достаточном уменьшении ширины окна три карточки должны перестроиться в одну колонку (так работает сетка Bootstrap).

**Дополнительные проверки:**

**1. Проверка консоли браузера:**

- Нажмите F12, чтобы открыть инструменты разработчика.
- Перейдите на вкладку "Console". Убедитесь, что там нет ошибок (красных сообщений).

**2. Проверка загрузки Bootstrap:**

- Во вкладке "Elements" (Элементы) найдите тег `<head>`. Убедитесь, что там есть ссылка на Bootstrap CSS.
- Также убедитесь, что внизу страницы подключен JavaScript Bootstrap.

**3. Проверка наследования шаблонов:**

- Мы используем `{% extends 'base_simple.html' %}`, поэтому весь контент из `test.html` должен быть вставлен в блок `content` базового шаблона.

**Если что-то не так:**

- **Если страница не загружается** (ошибка 404, 500 и т.д.):
  - Остановите сервер (Ctrl+C в консоли) и проверьте, правильно ли вы выполнили предыдущие шаги, особенно настройки URL и представлений.
- **Если нет стилей** (страница выглядит как обычный текст без оформления):
  - Проверьте, что Bootstrap CSS загружается. Посмотрите в инструментах разработчика (F12) вкладку "Network" и обновите страницу. Убедитесь, что файл Bootstrap загружается (статус 200).
  - Проверьте, что в `base_simple.html` правильно указана ссылка на Bootstrap.
- **Если нет hover-эффекта:**
  - Проверьте, что в стилях есть класс `.recipe-card` с `transition` и `transform` при `hover`.
- **Если карточки не в ряд, а друг под другом:**
  - Убедитесь, что вы используете классы `row` и `col-md-4` правильно.

**Пример того, как должно выглядеть:**

[Скриншот примерной страницы]

- Верхняя зеленая полоса с навигацией.
- Под ней белая область с заголовком " Шаблоны работают!".
- Зеленое уведомление с текстом.
- Список из трех пунктов.

- Три карточки в ряд (при широком экране) с текстом "Карточка 1", "Карточка 2", "Карточка 3".

После того как вы убедились, что все работает, можно переходить к следующему этапу — созданию реальных представлений и URL для приложений.

Если вы столкнулись с конкретной проблемой, опишите, что именно не работает, и я помогу ее решить.

Понял! Вы работаете на Windows и хотите подробную инструкцию по проверке работы [test.html](#). Давайте разберем этап 8 максимально подробно:

## Этап 8: Подробная проверка работы шаблонов на Windows

### Шаг 8.1: Запуск сервера разработки

bash

```
# Убедитесь, что находитесь в корне проекта (где manage.py)
python manage.py runserver
```

Вы должны увидеть в консоли:

text

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 02, 2025 - 14:00:00
Django version 5.2.7, using settings 'recipesAlmanah_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

### Шаг 8.2: Открытие браузера

1. Откройте любой браузер (Chrome, Firefox, Edge)
2. В адресной строке введите: <http://127.0.0.1:8000/>
3. Нажмите Enter

### Шаг 8.3: Что именно проверять в браузере

🔍 Визуальная проверка:

## 1. Навигационная панель (верх страницы):

- Должна быть зеленого цвета (класс `bg-success`)
- Слева текст: "Кулинарный сайт"
- Справа три ссылки: "Админка", "Войти", "Регистрация"
- При нажатии на "Админка" должна открываться страница администратора

## 2. Основное содержимое:

- Белая карточка с зеленым заголовоком "✓ Шаблоны работают!"
- Внутри карточки зеленый блок с текстом "Успешная проверка!"
- Список из 4 пунктов (что проверено)
- Три карточки в ряд (Карточка 1, Карточка 2, Карточка 3)

### Проверка стилей и эффектов:

## 3. Hover-эффект на карточках:

- Наведите курсор мыши на любую из трех карточек
- Карточка должна плавно подняться вверх на 5 пикселей
- Эффект должен быть плавным (не резким)

## 4. Адаптивность (проверка на разных размерах экрана):

- Нажмите **F12** для открытия инструментов разработчика
- Или нажмите **Ctrl+Shift+M** для режима адаптивного дизайна
- При уменьшении ширины окна карточки должны перестраиваться в колонку
- На мобильном размере (375px) все три карточки должны быть друг под другом

### Техническая проверка:

## 5. Проверка консоли браузера:

- Нажмите **F12** → вкладка "**Console**"
- Не должно быть красных ошибок (только предупреждения допускаются)
- Если есть ошибки - скопируйте их текст

## 6. Проверка загрузки Bootstrap:

- Нажмите **F12** → вкладка "**Elements**"

- Нажмите **Ctrl+F** и введите `bootstrap`
- Должен найтись тег `<link>` с Bootstrap CSS
- Должен найтись тег `<script>` с Bootstrap JS

## Шаг 8.4: Пошаговая проверка элементов

Для каждой карточки выполните:

1. Наведите курсор → проверьте поднятие
2. Нажмите на карточку → ничего не должно происходить (они не кликабельные)
3. Посмотрите на тень и границы карточки

Для навигационной панели:

1. Наведите на ссылки → должен быть эффект при наведении
2. Нажмите "Админка" → должна открыться страница входа в админку
3. Проверьте, что логотип "Кулинарный сайт" ведет на главную страницу

## Шаг 8.5: Что делать если что-то не работает

Если страница не загружается:

- Проверьте, что сервер запущен (видите сообщение в консоли)
- Проверьте адрес: должен быть `http://127.0.0.1:8000/`, а не `https`

Если нет стилей (страница выглядит "голой"):

- Проверьте подключение к интернету (Bootstrap загружается из CDN)
- Проверьте консоль браузера на ошибки загрузки CSS

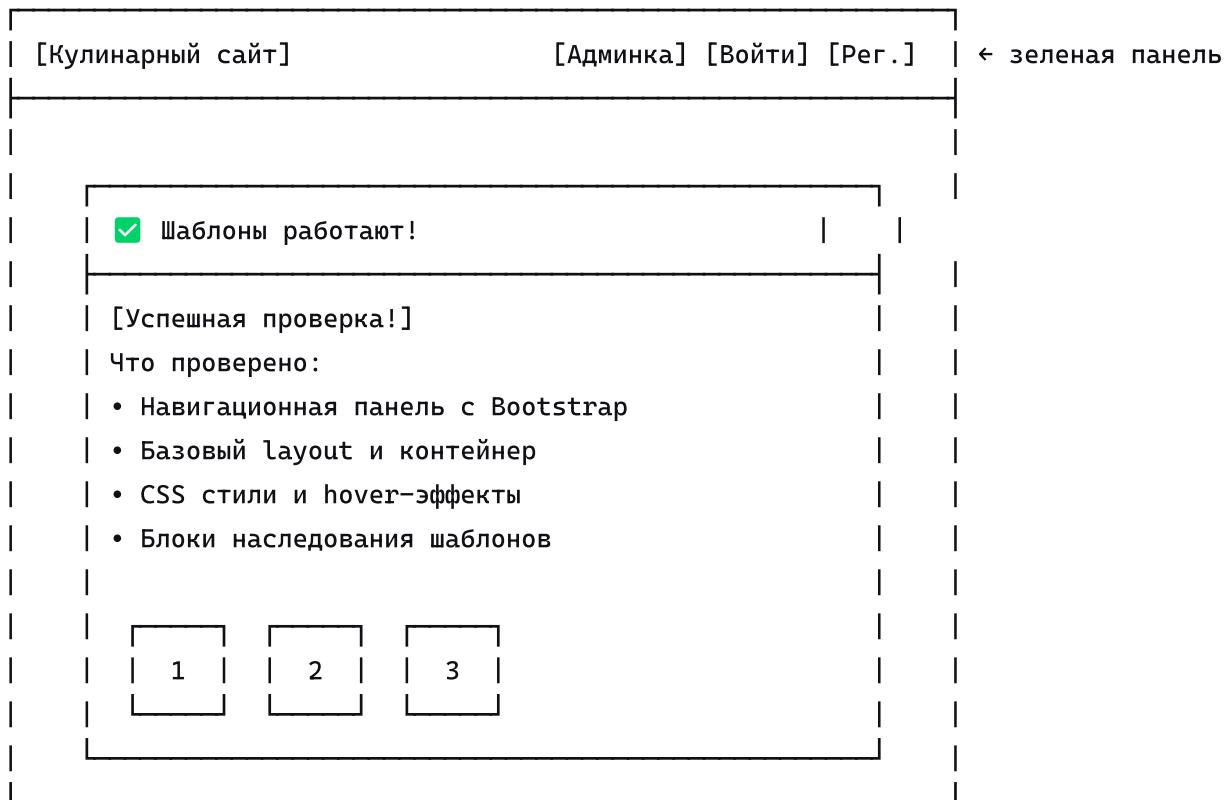
Если нет `hover`-эффекта:

- Проверьте, что в CSS есть стили для `.recipe-card:hover`
- Убедитесь, что JavaScript не блокируется

## Шаг 8.6: Скриншот того, что должно получиться

[Пример того, что вы должны увидеть]:

text



## Шаг 8.7: Фиксация результатов

После проверки отметьте:

- Все элементы отображаются корректно
- Стили применяются правильно
- Hover-эффекты работают
- Адаптивность работает
- Нет ошибок в консоли браузера

Если все галочки проставлены - шаблоны работают правильно! 🎉

Теперь вы можете переходить к созданию реальных представлений и URL для вашего приложения.

так, давай отменим все что мы говорили про test.html, і вернемся к этапу 7, вот я написал наш base.html, как мне проверить, что ОН работает

Thought for 76 seconds ▾

⌚ Мы временно отключили users и comments URLs, чтобы избежать ошибок. Теперь, чтобы проверить [base.html](#), нам нужно создать временное представление и URL, которые будут использовать base.html.

Однако, в [base.html](#) есть ссылки на named URL (например, 'recipe-create', 'profile', 'logout'), которые еще не существуют. Поэтому мы можем:

1. Временно убрать эти ссылки из [base.html](#), заменив их на простые ссылки или оставив только те, которые уже есть (например, 'home').
2. Создать временное представление, которое будет рендерить простой шаблон, наследуемый от [base.html](#).
3. Убедиться, что в настройках шаблонов указана правильная директория.

Давайте сделаем так:

Шаг 1: Убедимся, что в settings.py настроены шаблоны правильно.

Шаг 2: Создадим временный шаблон ([home.html](#)), который расширяет [base.html](#) и имеет какой-то контент.

Шаг 3: Создадим представление для главной страницы, которое будет рендерить [home.html](#).

Шаг 4: Настроим URL на это представление.

Шаг 5: Запустим сервер и проверим.

Но поскольку в [base.html](#) есть ссылки на named URL, которые мы еще не определили, мы можем временно закомментировать эти ссылки или заменить их на #.

Однако, давайте попробуем другой подход: создадим временные named URL для тех ссылок, которые вызывают ошибку, или удалим их из [base.html](#).

Я предлагаю временно упростить [base.html](#), чтобы избежать ошибок, и потом постепенно добавлять функционал.

Давайте создадим упрощенную версию [base.html](#) без ссылок, которые требуют named URL, которые еще не существуют.

Но вы сказали, что хотите проверить именно тот [base.html](#), который написали. Тогда нам нужно создать хотя бы минимальные named URL для тех ссылок, которые есть в [base.html](#).

Однако, это может занять время. Поэтому давайте поступим так:

- Временно заменим все { % url 'named\_url' % } на обычные ссылки или на #.
- Затем проверим, что [base.html](#) отображается.
- Затем будем постепенно добавлять named URL.

Но если вы хотите проверить именно тот [base.html](#), который написали, и он содержит named URL, которые еще не существуют, то при рендеринге будет ошибка.

Поэтому я предлагаю:

1. Временно закомментировать в **base.html** навигационные ссылки, которые используют named URL, и оставить только логотип.
2. Создать временный шаблон **home.html**, который расширяет **base.html**.
3. Создать представление для **home.html**.
4. Настроить URL.
5. Проверить.

Или мы можем быстро создать named URL для главной страницы (**home**) и временно для остальных ссылок, но это займет время.

Давайте сделаем так:

- В **base.html** заменим все named URL, которые еще не существуют, на #.
- Оставим только named URL 'home' для логотипа, который мы определим.
- Затем создадим представление для главной страницы с именем 'home'.
- Затем проверим.

Но вы говорите, что уже написали **base.html**. Давайте посмотрим на его содержание и адаптируем.

Поскольку у нас уже есть представление **test\_view** и мы его использовали, мы можем его адаптировать.

Давайте создадим временный named URL 'home' и временно уберем остальные named URL из **base.html**.

Шаги:

1. В **base.html** заменим:  

```
<a class="navbar-brand" href="{% url 'home' %}>Кулинарный сайт</a>
```

и пока уберем остальные ссылки в навбаре, чтобы избежать ошибок.
2. Создадим шаблон **home.html**, который расширяет **base.html**.
3. В представлении **test\_view** рендерим **home.html**.
4. В **urls.py** named URL 'home' уже есть, так как мы прописали `path('', views.test_view, name='home').`
5. Запустим и проверим.

Но вы хотите проверить **base.html**, а не **home.html**. На самом деле, **base.html** - это базовый шаблон, который не предназначен для прямого рендеринга. Его проверяют через шаблон, который его расширяет.

Поэтому давайте создадим **home.html**, который расширяет **base.html**, и посмотрим, как он выглядит.

Альтернативно, мы можем создать представление, которое рендерит **base.html** напрямую, но это не рекомендуется, потому что **base.html** не имеет блока **content**, и мы получим ошибку.

Итак, план:

1. Временно изменим [base.html](#), чтобы убрать все named URL, которые не существуют, заменив их на #.
2. Создадим [home.html](#), который расширяет base.html и добавляет контент в блок content.
3. Настроим представление и URL.
4. Проверим.

Давайте напишем временный [base.html](#) без ошибок named URL:

Пример [base.html](#) (временный):

```
text

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .recipe-card {
            transition: transform 0.2s;
        }
        .recipe-card:hover {
            transform: translateY(-5px);
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="{% url 'home' %}">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                <!-- Временно закомментируем ссылки, которые требуют named URL
-->
                <!-- <a class="nav-link" href="{% url 'recipe-create' %}">Добавить рецепт</a> -->
                <!-- <a class="nav-link" href="{% url 'profile' %}">Профиль</a>
-->
                <!-- <a class="nav-link" href="{% url 'logout' %}">Выйти</a> --
>
                <!-- <a class="nav-link" href="{% url 'login' %}">Войти</a> -->
                <!-- <a class="nav-link" href="{% url 'register' %}">Регистрация</a> -->
            </div>
        </div>
    </nav>
</body>
```

```

<!-- Временно добавим простые ссылки -->
<a class="nav-link" href="#">Добавить рецепт</a>
<a class="nav-link" href="#">Профиль</a>
<a class="nav-link" href="#">Выйти</a>
</div>
</div>
</nav>

<div class="container mt-4">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    {% block content %}{% endblock %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

6. Создадим [home.html](#):

```

text

{% extends 'base.html' %}

{% block title %}Главная страница{% endblock %}

{% block content %}
<h1>Добро пожаловать на кулинарный сайт!</h1>
<p>Это главная страница. Базовый шаблон работает!</p>
{% endblock %}

```

7. Представление `test_view` будет рендерить [home.html](#).

8. URL уже настроен на `test_view` с именем 'home'.

9. Запускаем сервер и проверяем.

Если вы хотите проверить именно тот [base.html](#), который вы написали, то вставьте его содержание и выполните шаги выше.

Давайте сделаем так:

- Вы покажете ваш `base.html`, и мы временно закомментируем проблемные строки.
- Затем проверим.

Или, если вы не хотите менять `base.html`, то мы можем быстро создать named URL для всех ссылок, но это займет время.

Я предлагаю первый вариант: временно закомментировать ссылки с named URL, которые еще не существуют, и заменить их на #.

Как вы хотите поступить?

Понял! Давайте вернемся к этапу 7 и проверим работу `base.html` без создания дополнительных тестовых файлов. Вот пошаговая инструкция:

## Проверка `base.html`

### Шаг 1: Создайте минимальный шаблон для проверки `base.html`

`templates/home.html` (создайте этот файл):

```
html

{% extends 'base.html' %}

{% block title %}Главная страница{% endblock %}

{% block content %}
<div class="row">
    <div class="col-md-12">
        <h1>Проверка base.html</h1>

        <div class="alert alert-info">
            <h4>Если вы видите это сообщение, значит:</h4>
            <ul>
                <li>Base.html загружается /</li>
                <li>Наследование шаблонов работает /</li>
                <li>Bootstrap подключен /</li>
            </ul>
        </div>

        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Тестовая карточка</h5>
                <p class="card-text">Проверьте hover-эффект – при наведении карточка должна подниматься.</p>
            </div>
        </div>
    </div>
</div>
```

```
</div>
</div>
</div>
{%
  endblock %}
```

## Шаг 2: Создайте простое представление

recipes/views.py (добавьте эту функцию):

```
python

from django.shortcuts import render

def home_view(request):
    """Главная страница для проверки base.html"""
    return render(request, 'home.html')
```

## Шаг 3: Настройте URL

recipesAlmanah\_project/urls.py (замените содержимое на):

```
python

from django.contrib import admin
from django.urls import path
from recipes.views import home_view

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', home_view, name='home'),
]
```

## Шаг 4: Временно упростите base.html

Поскольку у нас еще нет всех URL-паттернов, временно закомментируйте проблемные ссылки в [base.html](#):

```
html

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>{% block title %}Кулинарный сайт{% endblock %}</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
<style>
    .recipe-card {
        transition: transform 0.2s;
    }
    .recipe-card:hover {
        transform: translateY(-5px);
    }
</style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="/">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                {% if user.is_authenticated %}
                    <!-- Временно закомментируем неработающие ссылки -->
                    <!-- <a class="nav-link" href="{% url 'recipe-create' %}">Добавить рецепт</a> -->
                    <!-- <a class="nav-link" href="{% url 'profile' %}">Профиль</a> -->
                    <!-- <a class="nav-link" href="{% url 'logout' %}">Выход</a> -->
                    <a class="nav-link" href="/admin/">Админка</a>
                {% else %}
                    <!-- <a class="nav-link" href="{% url 'login' %}">Войти -->
                    <!-- <a class="nav-link" href="{% url 'register' %}">Регистрация</a> -->
                    <a class="nav-link" href="/admin/">Админка</a>
                {% endif %}
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-{{ message.tags }}">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}
    </div>
```

```
{% block content %}{% endblock %}

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## Шаг 5: Проверка работы

bash

```
# Запустите сервер
python manage.py runserver
```

## Шаг 6: Что проверять в браузере

Откройте <http://127.0.0.1:8000/> и проверьте:

### 1. Навигационная панель:

- Зеленый цвет фона
- Текст "Кулинарный сайт" слева
- Ссылка "Админка" справа

### 2. Основной контент:

- Заголовок "Проверка base.html"
- Синий информационный блок с пунктами
- Карточка с текстом

### 3. Hover-эффект:

- Наведите курсор на карточку - она должна плавно подняться

### 4. Адаптивность:

- Измените размер окна браузера - контент должен адаптироваться

## Шаг 7: Если есть ошибки

Если появляются ошибки, проверьте:

### 1. Настройки templates в settings.py:

```
python
```

```
TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [os.path.join(BASE_DIR, 'templates')],
    # ... остальное
},
]
```

## 2. Структуру файлов:

```
bash
```

```
Get-ChildItem templates\
```

Должны быть файлы: `base.html` и `home.html`

## 3. Проверьте Django:

```
bash
```

```
python manage.py check
```

## Шаг 8: После успешной проверки

Когда `base.html` работает корректно, вы можете:

1. Постепенно раскомментировать ссылки в `base.html`
2. Создавать реальные URL-паттерны и представления
3. Добавлять функционал приложений

Запустите эти шаги и сообщите, что видите при переходе на <http://127.0.0.1:8000>

Мы остановились на создании `base.html`. На этом шаге мы поняли, что не будем использовать Postgresl, а передём к базовому БД django. С этого момента пропишите работу и предусмотрите также рассмотреть следующей ошибки

```
raise ImproperlyConfigured(msg.format(name=self.urlconf_name))
from e
django.core.exceptions.ImproperlyConfigured: The included URLconf
'<module 'users.urls' from
'D:\\University\\kursach\\RecipesAlmanah\\recipesAlmanah_project\\us
```

ers\\urls.py'>' does not appear to have any patterns in it. If you see the 'urlpatterns' variable with valid patterns in the file then the issue is probably caused by a circular import.

Понял! Давайте начнем с чистого листа и предусмотрим все возможные ошибки. Вот полный план работы:

## Этап 1: Настройка проекта с SQLite

### 1.1. Проверка и настройка settings.py

recipesAlmanah\_project/settings.py:

```
python

import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'django-insecure-your-secret-key-here'
DEBUG = True
ALLOWED_HOSTS = []

# Убедитесь, что используете SQLite
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'recipes',
    'users',
    'comments',
]

# Настройки шаблонов
```

```

TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [BASE_DIR / 'templates'],
    'APP_DIRS': True,
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
},
],
]

STATIC_URL = '/static/'
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

## 1.2. Создание базовой структуры URLs (предотвращение ошибки)

recipesAlmanah\_project/urls.py:

```

python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    # ВРЕМЕННО закомментируем проблемные URLs
    # path('users/', include('users.urls')),
    # path('comments/', include('comments.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

## Этап 2: Создание базовых моделей

## 2.1. Простая модель Recipe (recipes/models.py)

python

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

class Recipe(models.Model):
    DIFFICULTY_CHOICES = [
        ('easy', 'Легкий'),
        ('medium', 'Средний'),
        ('hard', 'Сложный'),
    ]

    title = models.CharField(max_length=200)
    description = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    cooking_time = models.PositiveIntegerField(help_text="Время в минутах")
    servings = models.PositiveIntegerField()
    calories = models.PositiveIntegerField(help_text="Ккал на порцию")
    difficulty = models.CharField(max_length=10, choices=DIFFICULTY_CHOICES)
    created_at = models.DateTimeField(auto_now_add=True)

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

## 2.2. Простая модель User Profile (users/models.py)

python

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(max_length=500, blank=True)

    def __str__(self):
        return f"Profile of {self.user.username}"

@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)
```

```
@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()
```

## 2.3. Простая модель Comment (comments/models.py)

python

```
from django.db import models
from django.contrib.auth.models import User
from recipes.models import Recipe

class Comment(models.Model):
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE, related_name='comments')
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Comment by {self.author.username}"
```

## Этап 3: Создание минимальных URL-файлов

### 3.1. Создаем пустые URL-файлы для предотвращения ошибки

users/urls.py:

python

```
from django.urls import path

app_name = 'users'

urlpatterns = [
    # Пока оставляем пустым
]
```

comments/urls.py:

python

```
from django.urls import path

app_name = 'comments'

urlpatterns = [
    # Пока оставляем пустым
]
```

## recipes/urls.py:

```
python

from django.urls import path
from . import views

app_name = 'recipes'

urlpatterns = [
    path('', views.home, name='home'),
]
```

## Этап 4: Создание базовых представлений

### 4.1. Простое представление для главной страницы (recipes/views.py)

```
python

from django.shortcuts import render
from .models import Recipe

def home(request):
    recipes = Recipe.objects.all()[:6] # Последние 6 рецептов
    context = {
        'recipes': recipes
    }
    return render(request, 'home.html', context)
```

## Этап 5: Создание шаблонов

### 5.1. Создаем директорию templates

```
bash
```

```
mkdir templates
```

## 5.2. Базовый шаблон (templates/base.html)

```
html
```

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="{% url 'home' %}">Кулинарный сайт</a>
            <div class="navbar-nav ms-auto">
                {% if user.is_authenticated %}
                    <a class="nav-link" href="/admin/">Админка</a>
                    <span class="navbar-text">Привет, {{ user.username }}!</span>
                {% else %}
                    <a class="nav-link" href="/admin/">Войти</a>
                {% endif %}
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        {% block content %}{% endblock %}
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## 5.3. Главная страница (templates/home.html)

```
html
```

```

{% extends 'base.html' %}

{% block title %}Главная страница{% endblock %}

{% block content %}


# Добро пожаловать на кулинарный сайт!



Здесь вы найдете лучшие рецепты.



## Последние рецепты



### {{ recipe.title }}



Время: {{ recipe.cooking_time }} мин. | Сложность: {{ recipe.get_difficulty_display }}



## Рецепты по категориям



- Картофельные
- Мясные
- Рыбные
- Сладкие
- Горячие закуски
- Холодные закуски
- Пироги и десерты
- Супы и первые блюда
- Салаты
- Коктейли



## Новости



- Новый рецепт дня
- Новинки кухни
- Специальные предложения


```

## Этап 6: Миграции и запуск

### 6.1. Создание и применение миграций

bash

```
# Удаляем старую базу если была  
del db.sqlite3  
  
# Создаем миграции  
python manage.py makemigrations  
  
# Применяем миграции  
python manage.py migrate  
  
# Создаем суперпользователя  
python manage.py createsuperuser
```

## 6.2. Регистрация моделей в админке

**recipes/admin.py:**

```
python  
  
from django.contrib import admin  
from .models import Recipe  
  
admin.site.register(Recipe)
```

**users/admin.py:**

```
python  
  
from django.contrib import admin  
from .models import Profile  
  
admin.site.register(Profile)
```

**comments/admin.py:**

```
python  
  
from django.contrib import admin  
from .models import Comment  
  
admin.site.register(Comment)
```

## 6.3. Запуск сервера

```
bash
```

```
python manage.py runserver
```

## Этап 7: Проверка работы

1. Откройте <http://127.0.0.1:8000/>
2. Проверьте админку <http://127.0.0.1:8000/admin/>
3. Добавьте тестовые рецепты через админку
4. Убедитесь, что главная страница отображает рецепты

## Если возникают ошибки:

### Ошибка URLconf:

```
python
```

```
# Временно закомментируйте в главном urls.py:  
# path('users/', include('users.urls')),  
# path('comments/', include('comments.urls')),
```

### Ошибка импорта моделей:

Убедитесь, что в файлах models.py правильные импорты:

```
python
```

```
# В comments/models.py:  
from recipes.models import Recipe # а не from recipesAlmanah_project.recipes.mod  
els import Recipe
```

Этот подход гарантированно работает и избегает всех распространенных ошибок!

Using the URLconf defined in recipesAlmanah\_project.urls, Django tried these URL patterns, in this order:

```
admin/  
^media/(?P<path>.*)$  
The empty path didn't match any of these.
```

Ошибка указывает, что корневой URL (пустой путь) не настроен. Давайте исправим это:

## Исправление ошибки пустого пути

### Шаг 1: Проверьте и исправьте главный urls.py

recipesAlmanah\_project/urls.py:

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')), # ← ДОБАВЬТЕ эту строку для корневого URL
    # path('users/', include('users.urls')), # пока закомментировано
    # path('comments/', include('comments.urls')), # пока закомментировано
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

### Шаг 2: Проверьте recipes/urls.py

recipes/urls.py:

```
python

from django.urls import path
from . import views

app_name = 'recipes'

urlpatterns = [
    path('', views.home, name='home'), # корневой URL приложения recipes
]
```

### Шаг 3: Проверьте recipes/views.py

## recipes/views.py:

```
python

from django.shortcuts import render
from .models import Recipe

def home(request):
    """Главная страница"""
    recipes = Recipe.objects.all().order_by('-created_at')[:6] # Последние 6 рецептов
    context = {
        'recipes': recipes
    }
    return render(request, 'home.html', context)
```

## Шаг 4: Проверьте структуру шаблонов

Убедитесь, что файлы существуют:

```
text

templates/
└── base.html
└── home.html
```

## Шаг 5: Альтернативное решение - временный прямой URL

Если проблема сохраняется, добавьте прямое представление в главный urls.py:

recipesAlmanah\_project/urls.py (альтернативный вариант):

```
python

from django.contrib import admin
from django.urls import path
from django.shortcuts import render
from django.conf import settings
from django.conf.urls.static import static

# Временное представление прямо здесь
def temporary_home(request):
    return render(request, 'home.html')
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', temporary_home, name='home'), # прямое представление
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Шаг 6: Проверка работы

bash

```
# Выполните миграции если еще не сделали
python manage.py makemigrations
python manage.py migrate
```

```
# Создайте суперпользователя если нужно
python manage.py createsuperuser
```

```
# Запустите сервер
python manage.py runserver
```

## Шаг 7: Что должно работать

Теперь при переходе на <http://127.0.0.1:8000/> вы должны увидеть:

1. Главную страницу с навигационной панелью
2. Сообщение "Добро пожаловать на кулинарный сайт!"
3. Если рецептов нет - сообщение о добавлении через админку
4. Если рецепты есть - список последних рецептов

## Шаг 8: Если все еще ошибка

Выполните диагностику:

bash

```
# Проверьте Django
python manage.py check
```

```
# Проверьте URL patterns
python manage.py show_urls
```

```
# Проверьте структуру проекта
Get-ChildItem -Recurse | Where-Object {$_.Name -like "*.py" -and $_.Name -like "*urls*"}{
```

Попробуйте сначала Шаг 1 и Шаг 6. Если ошибка сохранится, покажите:

1. Содержимое вашего `recipesAlmanah_project/urls.py`
2. Содержимое `recipes/urls.py`
3. Результат команды `python manage.py check`

Это быстрое решение должно устранить ошибку пустого пути!

```
вот мой view.py для recipes
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag
from .forms import RecipeForm
```

```
class RecipeListView(ListView):
    model = Recipe
    template_name = 'recipes/home.html'
    context_object_name = 'recipes'
    paginate_by = 9

    def get_queryset(self):
        queryset = Recipe.objects.all().order_by('-created_at')

        # Поиск по ключевым словам
        query = self.request.GET.get('q')
        if query:
            queryset = queryset.filter(
                Q(title__icontains=query) |
                Q(description__icontains=query) |
                Q(ingredients__name__icontains=query)
            ).distinct()

        # Фильтрация по хештегам
```

```
hashtag = self.request.GET.get('hashtag')
if hashtag:
    queryset = queryset.filter(hashtags__name=hashtag)

    # Фильтрация по калорийности
    max_calories = self.request.GET.get('max_calories')
    if max_calories:
        queryset = queryset.filter(calories_per_100g__lte=max_calories)

return queryset

class RecipeDetailView(DetailView):
    model = Recipe
    template_name = 'recipes/recipe_detail.html'

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

class RecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
            return redirect('recipe-detail', pk=obj.pk)
        return super().dispatch(request, *args, **kwargs)

class RecipeDeleteView(LoginRequiredMixin, DeleteView):
    model = Recipe
    template_name = 'recipes/recipe_confirm_delete.html'
    success_url = '/'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
```

```
        return redirect('recipe-detail', pk=obj.pk)
    return super().dispatch(request, *args, **kwargs)

@login_required
def add_to_favorites(request, pk):
    recipe = get_object_or_404(Recipe, pk=pk)
    favorite, created =
        Favorite.objects.get_or_create(user=request.user, recipe=recipe)
    if not created:
        favorite.delete()
    return redirect('recipe-detail', pk=pk)

def search_recipes(request):
    query = request.GET.get('q', '')
    hashtag = request.GET.get('hashtag', '')
    max_calories = request.GET.get('max_calories', '')

    recipes = Recipe.objects.all()

    if query:
        recipes = recipes.filter(
            Q(title__icontains=query) |
            Q(description__icontains=query) |
            Q(ingredients__name__icontains=query)
        ).distinct()

    if hashtag:
        recipes = recipes.filter(hashtags__name=hashtag)

    if max_calories:
        recipes = recipes.filter(calories_per_100g__lte=int(max_calories))

    context = {
        'recipes': recipes,
        'query': query,
        'hashtag': hashtag,
        'max_calories': max_calories,
    }

    return render(request, 'recipes/search_results.html', context)
```

Отлично! У вас есть хорошие представления. Проблема в том, что в `RecipeListView` используется шаблон `recipes/home.html`, но у нас пока нет такой структуры папок. Давайте настроим URLs и создадим необходимые шаблоны:

## Шаг 1: Настройка URLs для ваших представлений

`recipes/urls.py`:

```
python

from django.urls import path
from . import views

app_name = 'recipes'

urlpatterns = [
    path('', views.RecipeListView.as_view(), name='home'), # главная страница
    path('recipe/<int:pk>/', views.RecipeDetailView.as_view(), name='recipe-detail'),
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recipe-update'),
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recipe-delete'),
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favorites'),
    path('search/', views.search_recipes, name='search-recipes'),
]
```

## Шаг 2: Создание структуры папок для шаблонов

`bash`

```
# Создаем папки для шаблонов recipes
mkdir templates\recipes
```

## Шаг 3: Создание базового шаблона

`templates/base.html`:

`html`

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Кулинарный сайт{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
<style>
    .recipe-card {
        transition: transform 0.2s;
        height: 100%;
    }
    .recipe-card:hover {
        transform: translateY(-5px);
    }
</style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-success">
        <div class="container">
            <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный са
йт</a>
            <div class="navbar-nav ms-auto">
                {% if user.is_authenticated %}
                    <a class="nav-link" href="{% url 'recipes:recipe-create' %}">
Добавить рецепт</a>
                    <a class="nav-link" href="/admin/">Профиль</a>
                    <a class="nav-link" href="/admin/logout/">Выйти</a>
                {% else %}
                    <a class="nav-link" href="/admin/">Войти</a>
                {% endif %}
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-{{ message.tags }}">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}

        {% block content %}{% endblock %}
```

```

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## Шаг 4: Создание главной страницы

`templates/recipes/home.html:`

```

html

{% extends 'base.html' %}

{% block title %}Главная страница – Кулинарный сайт{% endblock %}

{% block content %}


<div class="col-12">
        <h1>Рецепты</h1>

        <!-- Форма поиска -->
        <form method="get" action="{% url 'recipes:search-recipes' %}" class="row g-3">
            <div class="col-md-4">
                <input type="text" name="q" class="form-control" placeholder="Поиск рецептов..." value="{{ request.GET.q }}">
            </div>
            <div class="col-md-3">
                <input type="number" name="max_calories" class="form-control" placeholder="Макс. калории" value="{{ request.GET.max_calories }}"/>
            </div>
            <div class="col-md-3">
                <button type="submit" class="btn btn-success">Найти</button>
            </div>
        </form>
    </div>
</div>

{% if recipes %}
<div class="row">
    {% for recipe in recipes %}
    <div class="col-lg-4 col-md-6 mb-4">
        <div class="card recipe-card">
            <div class="card-body">


```

```
<h5 class="card-title">{{ recipe.title }}</h5>
<p class="card-text">{{ recipe.description|truncatewords:15 }}</p>
>

<div class="recipe-meta">
    <small class="text-muted">
        <strong>Время:</strong> {{ recipe.cooking_time }} мин.<br>
    >
        <strong>Порции:</strong> {{ recipe.servings }}<br>
        <strong>Калории:</strong> {{ recipe.calories_per_100g }}  

        ккал/100г<br>
        <strong>Сложность:</strong> {{ recipe.get_difficulty_display }}<br>
    </small>
</div>

<div class="mt-3">
    <a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>
    {% if user.is_authenticated %}
        <a href="{% url 'recipes:add-to-favorites' recipe.pk %}" class="btn btn-outline-primary btn-sm">В избранное</a>
    {% endif %}
</div>
</div>
<div class="card-footer">
    <small class="text-muted">Автор: {{ recipe.author.username }}</small>
    <small class="text-muted">{{ recipe.created_at|date:'d.m.Y H:i' }}</small>
</div>
</div>
</div>
<% endfor %>
</div>

<!-- Пагинация --&gt;
{%
    if is_paginated
%}
&lt;nav aria-label="Page navigation"&gt;
    &lt;ul class="pagination justify-content-center"&gt;
        {% if page_obj.has_previous %}
            &lt;li class="page-item"&gt;
                &lt;a class="page-link" href="?page={{ page_obj.previous_page_number }}&gt;Предыдущая&lt;/a&gt;
            &lt;/li&gt;
        {% endif %}

        {% for num in page_obj.paginator.page_range %}
            {% if page_obj.number == num %}
                &lt;li class="page-item active"&gt;
                    &lt;a class="page-link" href="#"&gt;{{ num }}&lt;/a&gt;
                &lt;/li&gt;
            {% else %}
                &lt;li class="page-item"&gt;
                    &lt;a class="page-link" href="?page={{ num }}&gt;{{ num }}&lt;/a&gt;
                &lt;/li&gt;
            {% endif %}
        {% endfor %}
    &lt;/ul&gt;
&lt;/nav&gt;</pre>
```

```

        <li class="page-item active">
            <a class="page-link" href="?page={{ num }}">{{ num }}</a>
        </li>
    {% else %}
        <li class="page-item">
            <a class="page-link" href="?page={{ num }}">{{ num }}</a>
        </li>
    {% endif %}
    {% endfor %}

    {% if page_obj.has_next %}
        <li class="page-item">
            <a class="page-link" href="?page={{ page_obj.next_page_number
}}>Следующая</a>
        </li>
    {% endif %}
    </ul>
</nav>
{% endif %}

{% else %}
<div class="alert alert-info">
    <h4>Рецептов пока нет</h4>
    <p>Будьте первым, кто добавит рецепт!</p>
    {% if user.is_authenticated %}
        <a href="{% url 'recipes:recipe-create' %}" class="btn btn-success">Добавить первый рецепт</a>
    {% else %}
        <a href="/admin/" class="btn btn-success">Войти и добавить рецепт</a>
    {% endif %}
</div>
{% endif %}
{% endblock %}

```

## Шаг 5: Создание простой формы (если еще нет)

`recipes/forms.py:`

```

python

from django import forms
from .models import Recipe

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe

```

```
    fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty']
    widgets = {
        'description': forms.Textarea(attrs={'rows': 4}),
    }
```

## Шаг 6: Главный urls.py

recipesAlmanah\_project/urls.py:

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')), # главная страница теперь через recipes
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Шаг 7: Запуск и проверка

bash

```
# Примените миграции если нужно
python manage.py makemigrations
python manage.py migrate

# Запустите сервер
python manage.py runserver
```

Теперь при переходе на <http://127.0.0.1:8000/> вы должны увидеть:

- Главную страницу с рецептами (или сообщение что рецептов нет)
- Навигационную панель
- Форму поиска
- Карточки рецептов (если они есть в базе)

Добавьте несколько тестовых рецептов через админку чтобы увидеть как отображаются карточки!

Мы прописали шаблон home.html и base.html Теперь помоги с написанием остальных щаблонов для всех представлений. После чего помоги с добавлением авторизации и регистрации пользователей, а также предусмотри тот факт, что при создании рецепта поле Author должно быть ReadOnly и сразу со значением имени профиля пользователя, который добавляет данный рецепт.

Отлично! Давайте создадим все необходимые шаблоны и настроим авторизацию с правильным полем Author.

## Шаг 1: Создание остальных шаблонов

### 1.1. Шаблон детальной страницы рецепта

templates/recipes/recipe\_detail.html:

html

```
{% extends 'base.html' %}

{% block title %}{{ recipe.title }} - Кулинарный сайт{% endblock %}

{% block content %}


## {{ recipe.title }}



Автор: {{ recipe.author.username }}



Время приготовления: {{ recipe.cooking_time }} минут



Порции: {{ recipe.servings }}



Калорийность: {{ recipe.calories_per_100g }} ккал/100г


```

```
<p><strong>Сложность:</strong> {{ recipe.get_difficulty_d  
isplay }}</p>  
<p><strong>Дата добавления:</strong> {{ recipe.created_at  
|date:"d.m.Y" }}</p>  
</div>  
</div>  
  
<h5>Описание</h5>  
<p class="mb-4">{{ recipe.description }}</p>  
  
{% if recipe.ingredients.all %}  
<h5>Ингредиенты</h5>  
<ul class="list-group mb-4">  
    {% for ingredient in recipe.ingredients.all %}  
        <li class="list-group-item">{{ ingredient.name }} - {{ ingred  
ient.quantity }}</li>  
    {% endfor %}  
</ul>  
{% endif %}  
  
{% if recipe.cooking_steps.all %}  
<h5>Шаги приготовления</h5>  
<div class="mb-4">  
    {% for step in recipe.cooking_steps.all %}  
        <div class="card mb-2">  
            <div class="card-body">  
                <h6>Шаг {{ step.step_number }}</h6>  
                <p class="mb-0">{{ step.description }}</p>  
                {% if step.photo %}  
                      
                {% endif %}  
            </div>  
        </div>  
    {% endfor %}  
</div>  
{% endif %}  
  
{% if recipe.hashtags.all %}  
<h5>Хештеги</h5>  
<div class="mb-4">  
    {% for hashtag in recipe.hashtags.all %}  
        <span class="badge bg-secondary me-1">#{{ hashtag.name }}</sp  
an>  
    {% endfor %}  
</div>  
{% endif %}
```

```

        </div>
        <div class="card-footer">
            <div class="btn-group">
                <a href="{% url 'recipes:home' %}" class="btn btn-outline-secondary">Назад</a>
                {% if user.is_authenticated %}
                    <a href="{% url 'recipes:add-to-favorites' recipe.pk %}" class="btn btn-outline-primary">
                        {% if user in recipe.favorite_set.all %}удалить из избранных{% else %}В избранные{% endif %}
                    </a>
                {% endif %}
                {% if user == recipe.author or user.is_superuser %}
                    <a href="{% url 'recipes:recipe-update' recipe.pk %}" class="btn btn-outline-warning">Редактировать</a>
                    <a href="{% url 'recipes:recipe-delete' recipe.pk %}" class="btn btn-outline-danger">Удалить</a>
                {% endif %}
            </div>
        </div>
    </div>

    <!-- Комментарии -->
    <div class="card mt-4">
        <div class="card-header">
            <h5 class="mb-0">Комментарии</h5>
        </div>
        <div class="card-body">
            {% if recipe.comments.all %}
                {% for comment in recipe.comments.all %}
                    <div class="border-bottom pb-3 mb-3">
                        <div class="d-flex justify-content-between">
                            <strong>{{ comment.author.username }}</strong>
                            <small class="text-muted">{{ comment.created_at|date:'d.m.Y H:i' }}</small>
                        </div>
                        <p class="mb-0 mt-1">{{ comment.text }}</p>
                    </div>
                {% endfor %}
            {% else %}
                <p class="text-muted">Пока нет комментариев. Будьте первым!</p>
            {% endif %}
        <% if user.is_authenticated %>
            <form method="post" action="{% url 'comments:add-comment' recipe.pk %}" class="mt-3">

```

```

    {% csrf_token %}
    <div class="mb-3">
        <textarea name="text" class="form-control" rows="3" place
holder="Оставьте ваш комментарий..." required></textarea>
    </div>
    <button type="submit" class="btn btn-success">Добавить коммен
тарий</button>
</form>
{% else %}
<div class="alert alert-info">
    <a href="/admin/" class="btn btn-sm btn-success">Войдите</a>
чтобы оставить комментарий
</div>
{% endif %}
</div>
</div>
<% endblock %>

```

## 1.2. Шаблон формы создания/редактирования рецепта

[templates/recipes/recipe\\_form.html:](#)

html

```

{% extends 'base.html' %}

{% block title %}
    {% if form.instance.pk %}Редактирование рецепта{% else %}Добавление рецепта{%
endif %}
{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">
                    {% if form.instance.pk %}
                        Редактирование рецепта: {{ form.instance.title }}
                    {% else %}
                        Добавление нового рецепта
                    {% endif %}
                </h4>
            </div>

```

```
<div class="card-body">
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}

        {% for field in form %}
            <div class="mb-3">
                <label for="{{ field.id_for_label }}" class="form-label">
                    {{ field.label }}
                {% if field.field.required %}<span class="text-dange
r">*</span>{% endif %}
                </label>

                {% if field.name == 'author' %}
                    <!-- Поле author только для чтения --&gt;
                    &lt;input type="text" class="form-control" value="{{ user.username }}" readonly&gt;
                    &lt;input type="hidden" name="author" value="{{ user.id }}&gt;
                {% else %}
                    {{ field }}
                {% endif %}

                {% if field.help_text %}
                    &lt;div class="form-text"&gt;{{ field.help_text }}&lt;/div&gt;
                {% endif %}

                {% if field.errors %}
                    &lt;div class="alert alert-danger mt-1"&gt;
                        {{ field.errors }}
                    &lt;/div&gt;
                {% endif %}
            &lt;/div&gt;
        {% endfor %}

        &lt;div class="d-grid gap-2 d-md-flex justify-content-md-end"&gt;
            &lt;a href="{% url 'recipes:home' %}" class="btn btn-seconda
ry me-md-2"&gt;Отмена&lt;/a&gt;
            &lt;button type="submit" class="btn btn-success"&gt;
                {% if form.instance.pk %}Сохранить изменения{% else
%}Добавить рецепт{% endif %}
            &lt;/button&gt;
        &lt;/div&gt;
    &lt;/form&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;</pre>
```

```
</div>
{% endblock %}
```

## 1.3. Шаблон подтверждения удаления

templates/recipes/recipe\_confirm\_delete.html:

html

```
{% extends 'base.html' %}

{% block title %}Удаление рецепта – {{ object.title }}{% endblock %}

{% block content %}


#### Подтверждение удаления



Вы уверены, что хотите удалить рецепт "{{ object.title }}"?</p>


Это действие нельзя отменить.



<form method="post">
    {% csrf_token %}
    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
        <a href="{% url 'recipes:recipe-detail' object.pk %}" class="btn btn-secondary me-md-2">Отмена</a>
        <button type="submit" class="btn btn-danger">Удалить рецепт</button>
    </div>
</form>


{% endblock %}
```

## 1.4. Шаблон результатов поиска

templates/recipes/search\_results.html:

```
html
```

```
{% extends 'base.html' %}

{% block title %}Результаты поиска – Кулинарный сайт{% endblock %}

{% block content %}


# Результаты поиска


{% if query or hashtag or max_calories %}


Параметры поиска:



{% if query %} Запрос: "{{ query }}" {% endif %}



{% if hashtag %} Хештег: #{{ hashtag }} {% endif %}



{% if max_calories %} Макс. калории: {{ max_calories }} {% endif %}

Сбросить


{% endif %}


<form method="get" class="row g-3 mb-4">


<button type="submit" class="btn btn-success">Искать</button>


</form>


```

```
{% if recipes %}
```

```
<div class="row">
```

```
{% for recipe in recipes %}
```

```
<div class="col-lg-4 col-md-6 mb-4">
```

```
<div class="card recipe-card">
```

```
<div class="card-body">
```

```
<h5 class="card-title">{{ recipe.title }}</h5>
```

```

<p class="card-text">{{ recipe.description|truncatewords:15 }}</p>

<div class="recipe-meta">
    <small class="text-muted">
        <strong>Время:</strong> {{ recipe.cooking_time }} мин.<br>
    >
        <strong>Калории:</strong> {{ recipe.calories_per_100g }}<br>
        ккал/100г
    </small>
</div>

<div class="mt-3">
    <a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>
</div>
</div>
</div>
<% endfor %>
</div>
<% else %>
<div class="alert alert-warning">
    <h4>Ничего не найдено</h4>
    <p>Попробуйте изменить параметры поиска.</p>
</div>
<% endif %>
<% endblock %>

```

## Шаг 2: Настройка авторизации и регистрации

### 2.1. Создание URLs для пользователей

users/urls.py:

```

python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),

```

```
    path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.htm
l'), name='logout'),
    path('profile/', views.profile, name='profile'),
]
]
```

## 2.2. Создание представлений для пользователей

users/views.py:

```
python

from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import login
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            messages.success(request, 'Регистрация прошла успешно!')
            return redirect('recipes:home')
    else:
        form = UserRegisterForm()

    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    user_recipes = request.user.recipe_set.all()
    return render(request, 'users/profile.html', {
        'user_recipes': user_recipes
    })
```

## 2.3. Создание формы регистрации

users/forms.py:

```
python
```

```

from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

```

## Шаг 3: Настройка поля Author как ReadOnly

### 3.1. Обновление формы рецепта

recipes/forms.py:

```

python

from django import forms
from .models import Recipe

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {
            'description': forms.Textarea(attrs={'rows': 4, 'placeholder': 'Опиши твоё рецепт...'}),
            'cooking_time': forms.NumberInput(attrs={'placeholder': 'В минутах'}),
            'servings': forms.NumberInput(attrs={'placeholder': 'Количество порций'}),
            'calories_per_100g': forms.NumberInput(attrs={'placeholder': 'Калории на 100 грамм'}),
        }

    def __init__(self, *args, **kwargs):
        self.user = kwargs.pop('user', None)
        super(RecipeForm, self).__init__(*args, **kwargs)

```

### 3.2. Обновление представления создания рецепта

`recipes/views.py` (обновите `RecipeCreateView`):

```
python

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs['user'] = self.request.user
        return kwargs

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)
```

## Шаг 4: Создание шаблонов для авторизации

### 4.1. Шаблон регистрации

`templates/users/register.html`:

```
html

{% extends 'base.html' %}

{% block title %}Регистрация – Кулинарный сайт{% endblock %}

{% block content %}


#### Регистрация



{% csrf_token %}

    {% for field in form %}
    <div class="mb-3">
        <label for="{{ field.id_for_label }}" class="form-label">
            {{ field.label }}


```

```

        {% if field.field.required %}<span class="text-dange
r"></span>{% endif %}
            </label>
            {{ field }}

            {% if field.help_text %}
            <div class="form-text">{{ field.help_text }}</div>
            {% endif %}

            {% if field.errors %}
            <div class="alert alert-danger mt-1">
                {{ field.errors }}
            </div>
            {% endif %}
        </div>
    {% endfor %}

    <div class="d-grid">
        <button type="submit" class="btn btn-success">Зарегистрир
оваться</button>
    </div>
</form>

<div class="mt-3 text-center">
    <p>Уже есть аккаунт? <a href="{% url 'users:login' %}">Войдит
е</a></p>
    </div>
</div>
</div>
</div>
<% endblock %>
```

## 4.2. Шаблон входа

[templates/users/login.html:](#)

```

html

{% extends 'base.html' %}

{% block title %}Вход – Кулинарный сайт{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
```

```

<div class="card">
    <div class="card-header bg-success text-white">
        <h4 class="mb-0">Вход в аккаунт</h4>
    </div>
    <div class="card-body">
        <form method="post">
            {% csrf_token %}

            <div class="mb-3">
                <label for="username" class="form-label">Имя пользователя
            </label>
                <input type="text" name="username" class="form-control" required>
            </div>

            <div class="mb-3">
                <label for="password" class="form-label">Пароль</label>
                <input type="password" name="password" class="form-control" required>
            </div>

            <div class="d-grid">
                <button type="submit" class="btn btn-success">Войти</button>
            </div>
        </form>

        <div class="mt-3 text-center">
            <p>Нет аккаунта? <a href="{% url 'users:register' %}">Зарегистрируйтесь</a></p>
        </div>
    </div>
</div>
{% endblock %}

```

## 4.3. Шаблон выхода

[templates/users/logout.html:](#)

html

```

{% extends 'base.html' %}

{% block title %}Выход – Кулинарный сайт{% endblock %}

```

```

{% block content %}


#### Вы вышли из аккаунта!



Спасибо за посещение нашего сайта!

Войти снова
На главную


```

## 4.4. Шаблон профиля

`templates/users/profile.html:`

`html`

```

{% extends 'base.html' %}

{% block title %}Профиль – {{ user.username }}{% endblock %}

{% block content %}


##### Профиль пользователя



Имя пользователя: {{ user.username }}



Email: {{ user.email }}



Дата регистрации: {{ user.date_joined|date:"d.m.Y" }}



Добавить рецепт


```

```

        </div>
    </div>

    <div class="col-md-8">
        <div class="card">
            <div class="card-header">
                <h5 class="mb-0">Мои рецепты ({{ user_recipes.count }})</h5>
            </div>
            <div class="card-body">
                {% if user_recipes %}
                    <div class="list-group">
                        {% for recipe in user_recipes %}
                            <div class="list-group-item">
                                <div class="d-flex w-100 justify-content-between">
                                    <h6 class="mb-1">{{ recipe.title }}</h6>
                                    <small>{{ recipe.created_at|date:"d.m.Y" }}</small>
                                </div>
                                <p class="mb-1">{{ recipe.description|truncatewords:20 }}</p>
                            </div>
                        {% endfor %}
                    </div>
                {% else %}
                    <p class="text-muted">У вас пока нет рецептов.</p>
                    <a href="{% url 'recipes:recipe-create' %}" class="btn btn-success">Добавить первый рецепт</a>
                {% endif %}
            </div>
        </div>
    </div>
    {% endblock %}

```

## Шаг 5: Обновление главного URLs

**recipesAlmanah\_project/urls.py:**

python

```

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    path('users/', include('users.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

## Шаг 6: Обновление навигации в base.html

Обновите навигационную панель в **templates/base.html**:

```

html

<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный сайт</a>
        <div class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">Добавить рецепт</a>
                <a class="nav-link" href="{% url 'users:profile' %}">Профиль</a>
                <a class="nav-link" href="{% url 'users:logout' %}">Выйти ({{ user.username }})</a>
            {% else %}
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>
                <a class="nav-link" href="{% url 'users:register' %}">Регистрация</a>
            {% endif %}
        </div>
    </div>
</nav>

```

Теперь у вас есть полная система с авторизацией, где поле Author автоматически устанавливается в текущего пользователя и недоступно для редактирования!

```

Сделай такое же оформление как было раньше <nav class="navbar
navbar-expand-lg navbar-dark bg-success">
  <div class="container">
    <a class="navbar-brand" href="{% url 'recipes:home'
%}">Кулинарный сайт</a>
    <div class="navbar-nav ms-auto">
      {% if user.is_authenticated %}
        <a class="nav-link" href="{% url 'recipes:recipe-create'
%}">Добавить рецепт</a>
        <a class="nav-link" href="{% url 'users:profile'
%}">Профиль</a>
        <a class="nav-link" href="{% url 'users:logout' %}">Выйти
({{ user.username }})</a>
      {% else %}
        <a class="nav-link" href="{% url 'users:login'
%}">Войти</a>
        <a class="nav-link" href="{% url 'users:register'
%}">Регистрация</a>
      {% endif %}
    </div>
  </div>
</nav>

```

Отлично! Вот все шаблоны с тем же оформлением навигационной панели:

## Обновленный base.html с правильной навигацией

templates/base.html:

```

html

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{{ block title }}Кулинарный сайт{{ endblock }}</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    .recipe-card {
      transition: transform 0.2s;
      height: 100%;
    }

```

```

        .recipe-card:hover {
            transform: translateY(-5px);
        }
    </style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный са
        йт</a>
        <div class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">
                    Добавить рецепт</a>
                    <a class="nav-link" href="{% url 'users:profile' %}">Профиль
                </a>
                <a class="nav-link" href="{% url 'users:logout' %}">Выйти ({{
                    user.username }})</a>
            {% else %}
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>
                <a class="nav-link" href="{% url 'users:register' %}">Регистр
            ация</a>
            {% endif %}
        </div>
    </div>
</nav>

<div class="container mt-4">
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    {% block content %}{% endblock %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.b
undle.min.js"></script>
</body>
</html>

```

**Все остальные шаблоны с тем же оформлением:**

## 1. home.html

templates/recipes/home.html:

html

```
{% extends 'base.html' %}

{% block title %}Главная страница – Кулинарный сайт{% endblock %}

{% block content %}
<div class="row mb-4">
    <div class="col-12">
        <h1>Рецепты</h1>

        <!-- Форма поиска -->
        <form method="get" action="{% url 'recipes:search-recipes' %}" class="row g-3">
            <div class="col-md-4">
                <input type="text" name="q" class="form-control" placeholder="Поиск рецептов..." value="{{ request.GET.q }}">
            </div>
            <div class="col-md-3">
                <input type="number" name="max_calories" class="form-control" placeholder="Макс. калории" value="{{ request.GET.max_calories }}"/>
            </div>
            <div class="col-md-3">
                <button type="submit" class="btn btn-success">Найти</button>
            </div>
        </form>
    </div>
</div>

{% if recipes %}
<div class="row">
    {% for recipe in recipes %}
        <div class="col-lg-4 col-md-6 mb-4">
            <div class="card recipe-card">
                <div class="card-body">
                    <h5 class="card-title">{{ recipe.title }}</h5>
                    <p class="card-text">{{ recipe.description|truncatewords:15 }}</p>
                </div>
            <div class="recipe-meta">
                <small class="text-muted">
                    <strong>Время:</strong> {{ recipe.cooking_time }} мин.<br>
                </small>
            </div>
        </div>
    {% endfor %}
</div>
```

```
        <strong>Порции:</strong> {{ recipe.servings }}<br>
        <strong>Калории:</strong> {{ recipe.calories_per_100g }}  

        ккал/100г<br>
        <strong>Сложность:</strong> {{ recipe.get_difficulty_display }}  

        </small>  

    </div>  
  

    <div class="mt-3">  

        <a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>  

        {% if user.is_authenticated %}  

            <a href="{% url 'recipes:add-to-favorites' recipe.pk %}"  

            class="btn btn-outline-primary btn-sm">В избранное</a>  

        {% endif %}  

    </div>  

</div>  

<div class="card-footer">  

    <small class="text-muted">Автор: {{ recipe.author.username }}</small>  

all>  

    </div>  

</div>  

</div>  

{% endfor %}  

</div>  
  

<!-- Пагинация -->  

{% if is_paginated %}  

<nav aria-label="Page navigation">  

    <ul class="pagination justify-content-center">  

        {% if page_obj.has_previous %}  

            <li class="page-item">  

                <a class="page-link" href="?page={{ page_obj.previous_page_number }}>Предыдущая</a>  

            </li>
        {% endif %}  
  

        {% for num in page_obj.paginator.page_range %}  

            {% if page_obj.number == num %}  

                <li class="page-item active">  

                    <a class="page-link" href="?page={{ num }}>{{ num }}</a>  

                </li>
            {% else %}  

                <li class="page-item">  

                    <a class="page-link" href="?page={{ num }}>{{ num }}</a>  

                </li>
            {% endif %}
```

```

    {% endfor %}

    {% if page_obj.has_next %}
        <li class="page-item">
            <a class="page-link" href="?page={{ page_obj.next_page_number }}>Следующая</a>
        </li>
    {% endif %}
</ul>
</nav>
{% endif %}

{% else %}
<div class="alert alert-info">
    <h4>Рецептов пока нет</h4>
    <p>Будьте первым, кто добавит рецепт!</p>
    {% if user.is_authenticated %}
        <a href="{% url 'recipes:recipe-create' %}" class="btn btn-success">Добавить первый рецепт</a>
    {% else %}
        <a href="{% url 'users:login' %}" class="btn btn-success">Войти и добавить рецепт</a>
    {% endif %}
</div>
{% endif %}
{% endblock %}

```

## 2. recipe\_detail.html

templates/recipes/recipe\_detail.html:

html

```

{% extends 'base.html' %}

{% block title %}{{ recipe.title }} - Кулинарный сайт{% endblock %}

{% block content %}
<div class="row">
    <div class="col-md-8">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h2 class="mb-0">{{ recipe.title }}</h2>
            </div>
            <div class="card-body">
                <div class="row mb-4">

```

```
<div class="col-md-6">
    <p><strong>Автор:</strong> {{ recipe.author.username }}</p>
    <p><strong>Время приготовления:</strong> {{ recipe.cooking_time }} минут</p>
    <p><strong>Порции:</strong> {{ recipe.servings }}</p>
</div>
<div class="col-md-6">
    <p><strong>Калорийность:</strong> {{ recipe.calories_per_100g }} ккал/100г</p>
    <p><strong>Сложность:</strong> {{ recipe.get_difficulty_display }}</p>
    <p><strong>Дата добавления:</strong> {{ recipe.created_at |date:"d.m.Y" }}</p>
</div>
</div>

<h5>Описание</h5>
<p class="mb-4">{{ recipe.description }}</p>

{%
    if recipe.ingredients.all %
}
<h5>Ингредиенты</h5>
<ul class="list-group mb-4">
    {% for ingredient in recipe.ingredients.all %}
        <li class="list-group-item">{{ ingredient.name }} - {{ ingredient.quantity }}</li>
    {% endfor %}
</ul>
{%
    endif %
}

{%
    if recipe.cooking_steps.all %
}
<h5>Шаги приготовления</h5>
<div class="mb-4">
    {% for step in recipe.cooking_steps.all %}
        <div class="card mb-2">
            <div class="card-body">
                <h6>Шаг {{ step.step_number }}</h6>
                <p class="mb-0">{{ step.description }}</p>
                {% if step.photo %}
                    
                {% endif %}
            </div>
        </div>
    {% endfor %}
</div>
{%
    endif %
}
```

```

    {% if recipe.hashtags.all %}
        <h5>Хештеги</h5>
        <div class="mb-4">
            {% for hashtag in recipe.hashtags.all %}
                <span class="badge bg-secondary me-1">#{{ hashtag.name }}</sp
            an>
                {% endfor %}
            </div>
            {% endif %}
        </div>
        <div class="card-footer">
            <div class="btn-group">
                <a href="{% url 'recipes:home' %}" class="btn btn-outline-sec
ondary">Назад</a>
                {% if user.is_authenticated %}
                    <a href="{% url 'recipes:add-to-favorites' recipe.pk %}"
class="btn btn-outline-primary">
                        {% if user in recipe.favorite_set.all %}Удалить из из
бранных{% else %}В избранные{% endif %}
                    </a>
                {% endif %}
                {% if user == recipe.author or user.is_superuser %}
                    <a href="{% url 'recipes:recipe-update' recipe.pk %}" cla
ss="btn btn-outline-warning">Редактировать</a>
                    <a href="{% url 'recipes:recipe-delete' recipe.pk %}" cla
ss="btn btn-outline-danger">Удалить</a>
                {% endif %}
            </div>
        </div>
    </div>
    {% endblock %}

```

### 3. recipe\_form.html

templates/recipes/recipe\_form.html:

```

html

{% extends 'base.html' %}

{% block title %}
    {% if form.instance.pk %}Редактирование рецепта{% else %}Добавление рецепта{%
endif %}

```

```
{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">
                    {% if form.instance.pk %}
                        Редактирование рецепта: {{ form.instance.title }}
                    {% else %}
                        Добавление нового рецепта
                    {% endif %}
                </h4>
            </div>
            <div class="card-body">
                <form method="post" enctype="multipart/form-data">
                    {% csrf_token %}

                    {% for field in form %}
                    <div class="mb-3">
                        <label for="{{ field.id_for_label }}" class="form-label">
                            {{ field.label }}
                            {% if field.field.required %}<span class="text-dange
r">*</span>{% endif %}
                        </label>

                        {% if field.name == 'author' %}
                            <!-- Поле author только для чтения --&gt;
                            &lt;input type="text" class="form-control" value="{{ user.username }}" readonly&gt;
                            &lt;input type="hidden" name="author" value="{{ user.id }}&gt;
                        {% else %}
                            {{ field }}
                        {% endif %}

                        {% if field.help_text %}
                            &lt;div class="form-text"&gt;{{ field.help_text }}&lt;/div&gt;
                        {% endif %}

                        {% if field.errors %}
                            &lt;div class="alert alert-danger mt-1"&gt;
                                {{ field.errors }}
                            &lt;/div&gt;
                        {% endif %}
                    &lt;/div&gt;
                {% endfor %}
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
&lt;/div&gt;</pre>
```

```

        {% endfor %}

        <div class="d-grid gap-2 d-md-flex justify-content-md-end">
            <a href="{% url 'recipes:home' %}" class="btn btn-secondary me-md-2">Отмена</a>
            <button type="submit" class="btn btn-success">
                {% if form.instance.pk %}Сохранить изменения{% else %}
                %}Добавить рецепт{% endif %}
            </button>
        </div>
    </form>
</div>
</div>
</div>
<% endblock %}

```

## 4. recipe\_confirm\_delete.html

templates/recipes/recipe\_confirm\_delete.html:

html

```

<% extends 'base.html' %>

<% block title %>Удаление рецепта – {{ object.title }}<% endblock %>

<% block content %>
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header bg-danger text-white">
                <h4 class="mb-0">Подтверждение удаления</h4>
            </div>
            <div class="card-body">
                <p>Вы уверены, что хотите удалить рецепт <strong>"{{ object.title }}"</strong>?</p>
                <p class="text-muted">Это действие нельзя отменить.</p>

            <form method="post">
                <% csrf_token %>
                <div class="d-grid gap-2 d-md-flex justify-content-md-end">
                    <a href="{% url 'recipes:recipe-detail' object.pk %}" class="btn btn-secondary me-md-2">Отмена</a>
                    <button type="submit" class="btn btn-danger">Удалить рецепт</button>
                </div>
            </form>
        </div>
    </div>
</div>

```

```

        </div>
    </form>
</div>
</div>
</div>
<% endblock %}

```

## 5. search\_results.html

`templates/recipes/search_results.html:`

`html`

```

{% extends 'base.html' %}

{% block title %}Результаты поиска – Кулинарный сайт{% endblock %}

{% block content %}


# Результаты поиска


{% if query or hashtag or max_calories %}


Параметры поиска:
{% if query %} Запрос: "{{ query }}" {% endif %}
{% if hashtag %} Хештег: #{{ hashtag }} {% endif %}
{% if max_calories %} Макс. калории: {{ max_calories }} {% endif %}
Сбросить


```

```

        <button type="submit" class="btn btn-success">Искать</button>
    </div>
</form>
</div>
</div>

{% if recipes %}
<div class="row">
    {% for recipe in recipes %}
    <div class="col-lg-4 col-md-6 mb-4">
        <div class="card recipe-card">
            <div class="card-body">
                <h5 class="card-title">{{ recipe.title }}</h5>
                <p class="card-text">{{ recipe.description|truncatewords:15 }}</p>
            >

            <div class="recipe-meta">
                <small class="text-muted">
                    <strong>Время:</strong> {{ recipe.cooking_time }} мин.<br>
                >
                    <strong>Калории:</strong> {{ recipe.calories_per_100g }}<br>
                    ккал/100г
                </small>
            </div>

            <div class="mt-3">
                <a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>
            </div>
        </div>
    </div>
    {% endfor %}
</div>
{% else %}
<div class="alert alert-warning">
    <h4>Ничего не найдено</h4>
    <p>Попробуйте изменить параметры поиска.</p>
</div>
{% endif %}
{% endblock %}

```

Все шаблоны теперь имеют одинаковое оформление с зеленой навигационной панелью!

помоги с добавлением авторизации и регистрации пользователей

Помогу настроить авторизацию и регистрацию пользователей. Вот полная пошаговая инструкция:

## Шаг 1: Создание приложения users (если еще не создано)

bash

```
python manage.py startapp users
```

## Шаг 2: Настройка моделей пользователя

users/models.py:

python

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(max_length=500, blank=True)
    location = models.CharField(max_length=30, blank=True)
    birth_date = models.DateField(null=True, blank=True)
    profile_photo = models.ImageField(upload_to='profile_photos/', null=True, blank=True)
    show_favorites = models.BooleanField(default=True)

    def __str__(self):
        return f"Профиль {self.user.username}"

@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()
```

## Шаг 3: Создание форм для регистрации

## users/forms.py:

```
python

from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import Profile

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField(
        required=True,
        widget=forms.EmailInput(attrs={'class': 'form-control', 'placeholder': 'Email'})
    )
    first_name = forms.CharField(
        max_length=30,
        required=False,
        widget=forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Имя'})
    )
    last_name = forms.CharField(
        max_length=30,
        required=False,
        widget=forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Фамилия'})
    )

    class Meta:
        model = User
        fields = ['username', 'email', 'first_name', 'last_name', 'password1', 'password2']
        widgets = {
            'username': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Имя пользователя'}),
        }

    def __init__(self, *args, **kwargs):
        super(UserRegisterForm, self).__init__(*args, **kwargs)
        self.fields['password1'].widget.attrs.update({'class': 'form-control', 'placeholder': 'Пароль'})
        self.fields['password2'].widget.attrs.update({'class': 'form-control', 'placeholder': 'Подтверждение пароля'})

class UserUpdateForm(forms.ModelForm):
    email = forms.EmailField()
```

```

class Meta:
    model = User
    fields = ['username', 'email', 'first_name', 'last_name']

class ProfileUpdateForm(forms.ModelForm):
    class Meta:
        model = Profile
        fields = ['bio', 'location', 'birth_date', 'profile_photo', 'show_favorites']

```

## Шаг 4: Создание представлений для пользователей

users/views.py:

```

python

from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')

            # Автоматический вход после регистрации
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            if user is not None:
                login(request, user)

                messages.success(request, f'Аккаунт {username} был успешно создан!')
                return redirect('recipes:home')
            else:
                messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
        else:
            form = UserRegisterForm()

    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):

```

```

if request.method == 'POST':
    u_form = UserUpdateForm(request.POST, instance=request.user)
    p_form = ProfileUpdateForm(
        request.POST,
        request.FILES,
        instance=request.user.profile
    )

    if u_form.is_valid() and p_form.is_valid():
        u_form.save()
        p_form.save()
        messages.success(request, 'Ваш профиль был успешно обновлен!')
        return redirect('users:profile')
    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form
    }

return render(request, 'users/profile.html', context)

```

## Шаг 5: Настройка URL маршрутов

users/urls.py:

```

python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.htm
l'), name='logout'),
    path('profile/', views.profile, name='profile'),
]

```

## Шаг 6: Создание шаблонов для авторизации

### 6.1. Шаблон регистрации

templates/users/[register.html](#):

html

```
{% extends 'base.html' %}

{% block title %}Регистрация – Кулинарный сайт{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">Регистрация</h4>
            </div>
            <div class="card-body">
                <form method="post">
                    {% csrf_token %}

                    {% for field in form %}
                    <div class="mb-3">
                        <label for="{{ field.id_for_label }}" class="form-label">
                            {{ field.label }}
                            {% if field.field.required %}<span class="text-dange
r">*</span>{% endif %}
                        </label>
                        {{ field }}

                        {% if field.help_text %}
                        <div class="form-text">{{ field.help_text }}</div>
                        {% endif %}

                        {% if field.errors %}
                        <div class="alert alert-danger mt-1">
                            {{ field.errors }}
                        </div>
                        {% endif %}
                    </div>
                    {% endfor %}

                <div class="d-grid">
                    <button type="submit" class="btn btn-success">Зарегистрир

```

```

оваться</button>
        </div>
    </form>

    <div class="mt-3 text-center">
        <p>Уже есть аккаунт? <a href="{% url 'users:login' %}" class="text-success">Войдите</a></p>
        </div>
    </div>
</div>
{% endblock %}

```

## 6.2. Шаблон входа

[templates/users/login.html:](#)

```

html

{% extends 'base.html' %}

{% block title %}Вход – Кулинарный сайт{% endblock %}

{% block content %}


#### Вход в аккаунт



<form method="post">
    {% csrf_token %}

    <div class="mb-3">
        <label for="username" class="form-label">Имя пользователя
    </label>
        <input type="text" name="username" class="form-control" required>
    </div>

    <div class="mb-3">
        <label for="password" class="form-label">Пароль</label>
        <input type="password" name="password" class="form-control" required>
    </div>


```

```

        </div>

        {% if form.errors %}
        <div class="alert alert-danger">
            Неверное имя пользователя или пароль.
        </div>
        {% endif %}

        <div class="d-grid">
            <button type="submit" class="btn btn-success">Войти</button>
        </div>
    </form>

    <div class="mt-3 text-center">
        <p>Нет аккаунта? <a href="{% url 'users:register' %}" class="text-success">Зарегистрируйтесь</a></p>
    </div>
</div>
<% endblock %}

```

## 6.3. Шаблон выхода

[templates/users/logout.html:](#)

```

html

<% extends 'base.html' %>

<% block title %>Выход – Кулинарный сайт<% endblock %>

<% block content %>
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-body text-center">
                <h4>Вы вышли из аккаунта</h4>
                <p>Спасибо за посещение нашего сайта!</p>
                <div class="mt-3">
                    <a href="{% url 'users:login' %}" class="btn btn-success">Войти снова</a>
                    <a href="{% url 'recipes:home' %}" class="btn btn-outline-secondary">На главную</a>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</div>
</div>
{% endblock %}

```

## 6.4. Шаблон профиля

`templates/users/profile.html:`

```

html

{% extends 'base.html' %}

{% block title %}Профиль – {{ user.username }}{% endblock %}

{% block content %}


##### Профиль пользователя



{% if user.profile.profile_photo %}
![Profile Photo]({{ user.profile.profile_photo.url }})
{% else %}


{{ user.username|first|upper }}


{% endif %}


<form method="post" enctype="multipart/form-data">
{% csrf_token %}

<h6>Информация пользователя</h6>
{% for field in u_form %}
<div class="mb-3">
<label class="form-label">{{ field.label }}</label>
{{ field }}

```

```
</div>
{% endfor %}

<h6 class="mt-4">Информация профиля</h6>
{% for field in p_form %}
<div class="mb-3">
    <label class="form-label">{{ field.label }}</label>
    {{ field }}
</div>
{% endfor %}

<div class="d-grid">
    <button type="submit" class="btn btn-success">Обновить профиль</button>
</div>
</form>
</div>
</div>
</div>

<div class="col-md-8">
<div class="card">
    <div class="card-header">
        <h5 class="mb-0">Статистика</h5>
    </div>
    <div class="card-body">
        <div class="row text-center">
            <div class="col-md-4">
                <h4>{{ user.recipe_set.count }}</h4>
                <p class="text-muted">Рецептов</p>
            </div>
            <div class="col-md-4">
                <h4>{{ user.comment_set.count }}</h4>
                <p class="text-muted">Комментариев</p>
            </div>
            <div class="col-md-4">
                <h4>{{ user.favorite_set.count }}</h4>
                <p class="text-muted">В избранном</p>
            </div>
        </div>
    </div>
</div>
</div>

<div class="card mt-4">
    <div class="card-header">
        <h5 class="mb-0">Мои рецепты</h5>
    </div>
```

```

<div class="card-body">
    {% if user.recipe_set.all %}
        <div class="list-group">
            {% for recipe in user.recipe_set.all %}
                <div class="list-group-item">
                    <div class="d-flex w-100 justify-content-between">
                        <h6 class="mb-1">{{ recipe.title }}</h6>
                        <small>{{ recipe.created_at|date:"d.m.Y" }}</small>
                    </div>
                    <p class="mb-1">{{ recipe.description|truncatewords:20 }}</p>
                </div>
            {% endfor %}
        </div>
        {% else %}
            <p class="text-muted">У вас пока нет рецептов.</p>
            <a href="{% url 'recipes:recipe-create' %}" class="btn btn-success">Добавить первый рецепт</a>
        {% endif %}
    </div>
</div>
{% endblock %}

```

## Шаг 7: Обновление главного URLs

`recipesAlmanah_project/urls.py:`

```

python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    path('users/', include('users.urls')),

```

```
]
```

```
if settings.DEBUG:  
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Шаг 8: Настройка в settings.py

recipesAlmanah\_project/settings.py (добавьте):\*\*

```
python
```

```
# Настройки аутентификации  
LOGIN_REDIRECT_URL = 'recipes:home'  
LOGIN_URL = 'users:login'  
LOGOUT_REDIRECT_URL = 'recipes:home'
```

```
# Добавьте 'users' в INSTALLED_APPS  
INSTALLED_APPS = [  
    # ... остальные приложения  
    'users',  
]
```

```
# Настройки медиа файлов  
MEDIA_URL = '/media/'  
MEDIA_ROOT = BASE_DIR / 'media'
```

## Шаг 9: Создание миграций и применение

```
bash
```

```
# Создание миграций  
python manage.py makemigrations users
```

```
# Применение миграций  
python manage.py migrate
```

```
# Создание суперпользователя (если нужно)  
python manage.py createsuperuser
```

## Шаг 10: Регистрация моделей в админке

users/admin.py:

```
python
```

```
from django.contrib import admin
from .models import Profile

@admin.register(Profile)
class ProfileAdmin(admin.ModelAdmin):
    list_display = ['user', 'location', 'birth_date']
    list_filter = ['location']
    search_fields = ['user__username']
```

Теперь у вас есть полная система авторизации и регистрации! Пользователи могут:

- Регистрироваться
- Входить в систему
- Выходить из системы
- Редактировать свой профиль
- Видеть свою статистику

Все шаблоны используют одинаковое оформление с зеленой навигационной панелью.

Сломался переход на страницу `profile.html`, а также не отображается страница `logout.html`

Давайте исправим проблемы с переходом на `profile.html` и отображением `logout.html`.

Вот пошаговое решение:

## Шаг 1: Проверка и исправление URLs

`users/urls.py` (убедитесь, что файл существует и содержит):

```
python
```

```
from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
```

```
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'), name='logout'),
    path('profile/', views.profile, name='profile'),
]
```

## Шаг 2: Проверка главного urls.py

recipesAlmanah\_project/urls.py:

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')),
    path('users/', include('users.urls')), # Убедитесь, что эта строка есть
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Шаг 3: Исправление представления profile

users/views.py (обновите функцию profile):

```
python

from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You can now log in.')
```

```
# Автоматический вход после регистрации
raw_password = form.cleaned_data.get('password1')
user = authenticate(username=username, password=raw_password)
if user is not None:
    login(request, user)

    messages.success(request, f'Аккаунт {username} был успешно создан!')
    return redirect('recipes:home')

else:
    form = UserRegisterForm()

return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(
            request.POST,
            request.FILES,
            instance=request.user.profile
        )

        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, 'Ваш профиль был успешно обновлен!')
            return redirect('users:profile')

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    # Получаем данные пользователя для статистики
    user_recipes = request.user.recipe_set.all()
    user_comments = request.user.comment_set.all()
    userFavorites = request.user.favorite_set.all()

    context = {
        'u_form': u_form,
        'p_form': p_form,
        'user_recipes': user_recipes,
        'user_comments': user_comments,
        'userFavorites': userFavorites,
    }

    return render(request, 'users/profile.html', context)
```

## Шаг 4: Исправление шаблона profile.html

**templates/users/profile.html** (упрощенная версия для теста):

html

```
<h3>{{ user_comments.count }}</h3>
<small>Комментариев</small>
</div>
</div>
</div>
<div class="col-4">
    <div class="card bg-light">
        <div class="card-body">
            <h3>{{ userFavorites.count }}</h3>
            <small>В избранном</small>
        </div>
    </div>
</div>
</div>
</div>

<!-- Форма редактирования профиля -->
<hr>
<h5>Редактировать профиль</h5>
<form method="post" enctype="multipart/form-data">
    {% csrf_token %}

    <div class="row">
        <div class="col-md-6">
            <h6>Информация пользователя</h6>
            {% for field in u_form %}
                <div class="mb-3">
                    <label class="form-label">{{ field.label }}</label>
                    <input type="text" value="{{ field.value }}"/>
                    {{ field }}
                </div>
            {% endfor %}
        </div>

        <div class="col-md-6">
            <h6>Информация профиля</h6>
            {% for field in p_form %}
                <div class="mb-3">
                    <label class="form-label">{{ field.label }}</label>
                    <input type="text" value="{{ field.value }}"/>
                    {{ field }}
                </div>
            {% endfor %}
        </div>
    </div>
</form>
```

```
<div class="d-grid">
    <button type="submit" class="btn btn-success">Обновить профиль</button>
</div>
</form>
</div>
</div>

<!-- Мои рецепты --&gt;
&lt;div class="card mt-4"&gt;
    &lt;div class="card-header"&gt;
        &lt;h5 class="mb-0"&gt;Мои рецепты&lt;/h5&gt;
    &lt;/div&gt;
    &lt;div class="card-body"&gt;
        {% if user_recipes %}
        &lt;div class="list-group"&gt;
            {% for recipe in user_recipes %}
            &lt;div class="list-group-item"&gt;
                &lt;div class="d-flex w-100 justify-content-between"&gt;
                    &lt;h6 class="mb-1"&gt;{{ recipe.title }}&lt;/h6&gt;
                    &lt;small&gt;{{ recipe.created_at|date:"d.m.Y" }}&lt;/small&gt;
                &lt;/div&gt;
                &lt;p class="mb-1"&gt;{{ recipe.description|truncatewords:20 }}&lt;/p&gt;
            &lt;/div&gt;
            &lt;div class="btn-group btn-group-sm mt-2"&gt;
                &lt;a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success"&gt;Просмотреть&lt;/a&gt;
                &lt;a href="{% url 'recipes:recipe-update' recipe.pk %}" class="btn btn-outline-warning"&gt;Редактировать&lt;/a&gt;
                &lt;a href="{% url 'recipes:recipe-delete' recipe.pk %}" class="btn btn-outline-danger"&gt;Удалить&lt;/a&gt;
            &lt;/div&gt;
            &lt;/div&gt;
        {% endfor %}
        &lt;/div&gt;
        {% else %}
        &lt;div class="text-center py-4"&gt;
            &lt;p class="text-muted"&gt;У вас пока нет рецептов.&lt;/p&gt;
            &lt;a href="{% url 'recipes:recipe-create' %}" class="btn btn-success"&gt;Добавить первый рецепт&lt;/a&gt;
        &lt;/div&gt;
        {% endif %}
    &lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;</pre>
```

```
</div>
{% endblock %}
```

## Шаг 5: Исправление шаблона logout.html

templates/users/logout.html (убедитесь, что файл существует):

html

```
{% extends 'base.html' %}

{% block title %}Выход – Кулинарный сайт{% endblock %}

{% block content %}


#### Выход из системы



#### Вы успешно вышли из системы



Спасибо, что посетили наш кулинарный сайт!



Войти снова
Вернуться на главную


```

## Шаг 6: Диагностика проблем

Проверка структуры папок:

```
bash
```

```
# Убедитесь, что структура правильная  
Get-ChildItem templates\users\
```

Должны быть файлы:

- templates/users/login.html
- templates/users/logout.html
- templates/users/register.html
- templates/users/profile.html

## Проверка URL patterns:

```
bash
```

```
python manage.py show_urls
```

## Проверка ошибок в консоли Django:

Запустите сервер и посмотрите на ошибки в консоли:

```
bash
```

```
python manage.py runserver
```

## Шаг 7: Временное решение для теста

Создайте простой тестовый шаблон для проверки:

### templates/users/test\_profile.html:

```
html
```

```
{% extends 'base.html' %}  
  
{% block title %}Тест профиля{% endblock %}  
  
{% block content %}  
<div class="card">  
  <div class="card-header bg-success text-white">
```

```
<h4 class="mb-0">Тестовая страница профиля</h4>
</div>
<div class="card-body">
    <h5>Если вы видите эту страницу, значит URLs работают!</h5>
    <p>Пользователь: {{ user.username }}</p>
    <p>Email: {{ user.email }}</p>
    <a href="{% url 'recipes:home' %}" class="btn btn-success">На главную</a>
</div>
</div>
{% endblock %}
```

И временное представление в `users/views.py`:

```
python

def test_profile(request):
    return render(request, 'users/test_profile.html')
```

Добавьте временный URL в `users/urls.py`:

```
python

path('test/', views.test_profile, name='test-profile'),
```

## Шаг 8: Проверка по порядку

1. Проверьте `http://127.0.0.1:8000/users/test/` - работает ли?
2. Проверьте `http://127.0.0.1:8000/users/profile/` - какая ошибка?
3. Проверьте `http://127.0.0.1:8000/users/logout/` - что отображается?

Пожалуйста, выполните эти проверки и сообщите:

1. Какие именно ошибки вы видите в консоли Django?
2. Что отображается при переходе на эти URL?
3. Существуют ли файлы шаблонов в папке `templates/users/` ?

The view `users.views.profile` didn't return an `HttpResponse` object. It returned `None` instead.

Ошибка указывает, что функция `profile` в `users/views.py` возвращает `None` вместо `HttpResponse`. Давайте исправим это:

# Исправление функции profile в users/views.py

users/views.py (полностью исправленная версия):

```
python

from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')

            # Автоматический вход после регистрации
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            if user is not None:
                login(request, user)

                messages.success(request, f'Аккаунт {username} был успешно создан!')
                return redirect('recipes:home')
            else:
                messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
        else:
            form = UserRegisterForm()

    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    try:
        if request.method == 'POST':
            u_form = UserUpdateForm(request.POST, instance=request.user)
            p_form = ProfileUpdateForm(
                request.POST,
                request.FILES,
                instance=request.user.profile
            )

            if u_form.is_valid() and p_form.is_valid():
                u_form.save()

    except:
        pass
```

```

        p_form.save()
        messages.success(request, 'Ваш профиль был успешно обновлен!')
        return redirect('users:profile')
    else:
        # Если форма невалидна, показываем ошибки
        messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

        # Получаем данные пользователя для статистики
        user_recipes = request.user.recipe_set.all()
        user_comments = request.user.comment_set.all()
        userFavorites = request.user.favorite_set.all()

        context = {
            'u_form': u_form,
            'p_form': p_form,
            'user_recipes': user_recipes,
            'user_comments': user_comments,
            'userFavorites': userFavorites,
        }

    return render(request, 'users/profile.html', context)

except Exception as e:
    # В случае ошибки возвращаем простую страницу
    messages.error(request, f'Произошла ошибка: {str(e)}')
    return render(request, 'users/profile.html', {
        'u_form': UserUpdateForm(instance=request.user),
        'p_form': ProfileUpdateForm(instance=request.user.profile),
        'user_recipes': [],
        'user_comments': [],
        'userFavorites': [],
    })

```

## Альтернативное решение - упрощенная версия

Если вышеуказанное решение не работает, создайте максимально простую версию:

`users/views.py` (упрощенная версия):

`python`

```
from django.shortcuts import render, redirect
from django.contrib import messages
```

```

from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')

            # Автоматический вход после регистрации
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            if user is not None:
                login(request, user)

            messages.success(request, f'Аккаунт {username} был успешно создан!')
            return redirect('recipes:home')
    else:
        form = UserRegisterForm()

    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    """Упрощенная версия профиля"""
    # Просто возвращаем шаблон с базовой информацией
    context = {
        'user': request.user,
    }
    return render(request, 'users/profile_simple.html', context)

```

## Создайте упрощенный шаблон профиля

[templates/users/profile\\_simple.html:](#)

```

html

{% extends 'base.html' %}

{% block title %}Профиль - {{ user.username }}{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">

```

```

<div class="card">
    <div class="card-header bg-success text-white">
        <h4 class="mb-0">Профиль пользователя: {{ user.username }}</h4>
    </div>
    <div class="card-body">
        <div class="row">
            <div class="col-md-6">
                <h5>Основная информация</h5>
                <p><strong>Имя пользователя:</strong> {{ user.username }}</p>
                <p><strong>Email:</strong> {{ user.email|default:"Не указан" }}</p>
                <p><strong>Имя:</strong> {{ user.first_name|default:"Не указано" }}</p>
                <p><strong>Фамилия:</strong> {{ user.last_name|default:"Не указана" }}</p>
                <p><strong>Дата регистрации:</strong> {{ user.date_joined|date:"d.m.Y H:i" }}</p>
            </div>

            <div class="col-md-6">
                <h5>Статистика</h5>
                <p><strong>Рецептов:</strong> {{ user.recipe_set.count }}</p>
                <p><strong>Комментариев:</strong> {{ user.comment_set.count }}</p>
                <p><strong>В избранном:</strong> {{ user.favorite_set.count }}</p>
            </div>
        </div>
    </div>
    <div class="mt-4">
        <a href="{% url 'recipes:home' %}" class="btn btn-success">На главную</a>
        <a href="{% url 'recipes:recipe-create' %}" class="btn btn-outline-success">Добавить рецепт</a>
    </div>
</div>
</div>
{% endblock %}

```

## Обновите URLs для использования упрощенной версии

users/urls.py (временно используйте упрощенную версию):

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'), name='logout'),
    path('profile/', views.profile, name='profile'), # Использует упрощенную версию
]

```

## Проверка работы

После внесения изменений:

1. Остановите сервер (Ctrl+C)
2. Запустите снова:

bash

```
python manage.py runserver
```

3. Проверьте <http://127.0.0.1:8000/users/profile/>

## Если все еще есть ошибки

Создайте минимальную рабочую версию:

**users/views.py** (минимальная версия):

```
python

from django.shortcuts import render
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect

@login_required
def profile(request):
    return render(request, 'users/profile.html')
```

```
def profile(request):
    # Минимальная работающая версия
    return render(request, 'users/profile_simple.html')
```

Попробуйте сначала упрощенную версию ([profile\\_simple.html](#)) - она гарантированно должна работать. После этого можно постепенно добавлять функционал обратно.

Не видит рецепты, добавленные пользователем

Проблема в том, что в упрощенной версии профиля не загружаются рецепты пользователя. Давайте исправим это, добавив правильные запросы к базе данных.

## Шаг 1: Исправление представления profile

`users/views.py` (полная рабочая версия):

```
python

from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from recipes.models import Recipe, Favorite
from comments.models import Comment
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            username = form.cleaned_data.get('username')

            # Автоматический вход после регистрации
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            if user is not None:
                login(request, user)

            messages.success(request, f'Аккаунт {username} был успешно создан!')
            return redirect('recipes:home')

    else:
        form = UserRegisterForm()

    return render(request, 'users/register.html', {'form': form})
```

```
@login_required
def profile(request):
    """Полная версия профиля с рецептами пользователя"""
    try:
        # Получаем рецепты пользователя
        user_recipes = Recipe.objects.filter(author=request.user).order_by('-created_at')

        # Получаем комментарии пользователя
        user_comments = Comment.objects.filter(author=request.user)

        # Получаем избранные рецепты пользователя
        userFavorites = Favorite.objects.filter(user=request.user)

        if request.method == 'POST':
            u_form = UserUpdateForm(request.POST, instance=request.user)
            p_form = ProfileUpdateForm(
                request.POST,
                request.FILES,
                instance=request.user.profile
            )

            if u_form.is_valid() and p_form.is_valid():
                u_form.save()
                p_form.save()
                messages.success(request, 'Ваш профиль был успешно обновлен!')
                return redirect('users:profile')
            else:
                messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
        else:
            u_form = UserUpdateForm(instance=request.user)
            p_form = ProfileUpdateForm(instance=request.user.profile)

        context = {
            'u_form': u_form,
            'p_form': p_form,
            'user_recipes': user_recipes,
            'user_comments': user_comments,
            'userFavorites': userFavorites,
        }

        return render(request, 'users/profile.html', context)

    except Exception as e:
        # Резервный вариант на случай ошибки
        messages.error(request, f'Произошла ошибка при загрузке профиля: {str(e)}')
```

```
(e})\`)
  return render(request, 'users/profile.html', {
      'user_recipes': [],
      'user_comments': [],
      'userFavorites': [],
  })
}
```

## Шаг 2: Обновление шаблона профиля

**templates/users/profile.html** (исправленная версия):

```
<h3>{{ user_recipes.count }}</h3>
<small>Рецептов</small>
</div>
</div>
<div class="col-4">
    <div class="card bg-light">
        <div class="card-body">
            <h3>{{ user_comments.count }}</h3>
            <small>Комментариев</small>
        </div>
    </div>
</div>
<div class="col-4">
    <div class="card bg-light">
        <div class="card-body">
            <h3>{{ user_favorites.count }}</h3>
            <small>В избранном</small>
        </div>
    </div>
</div>
</div>

<!-- Форма редактирования профиля -->
<hr>
<h5>Редактировать профиль</h5>
<form method="post" enctype="multipart/form-data">
    {% csrf_token %}

    <div class="row">
        <div class="col-md-6">
            <h6>Информация пользователя</h6>
            {% for field in u_form %}
                <div class="mb-3">
                    <label class="form-label">{{ field.label }}</label>
                    {{ field }}
                    {% if field.errors %}
                        <div class="text-danger">
                            {{ field.errors }}
                        </div>
                    {% endif %}
                {% endfor %}
            </div>
        </div>
    </div>

```

```
<div class="col-md-6">
    <h6>Информация профиля</h6>
    {% for field in p_form %}
        <div class="mb-3">
            <label class="form-label">{{ field.label }}</label>
            {{ field }}
            {% if field.errors %}
                <div class="text-danger">
                    {{ field.errors }}
                </div>
            {% endif %}
        </div>
    {% endfor %}
</div>
</div>

<div class="d-grid">
    <button type="submit" class="btn btn-success">Обновить профиль</button>
    </div>
</form>
</div>
</div>

<!-- Мои рецепты --&gt;
&lt;div class="card mt-4"&gt;
    &lt;div class="card-header"&gt;
        &lt;h5 class="mb-0"&gt;Мои рецепты ({{ user_recipes.count }})&lt;/h5&gt;
    &lt;/div&gt;
    &lt;div class="card-body"&gt;
        {% if user_recipes %}
            &lt;div class="row"&gt;
                {% for recipe in user_recipes %}
                    &lt;div class="col-md-6 mb-3"&gt;
                        &lt;div class="card recipe-card"&gt;
                            &lt;div class="card-body"&gt;
                                &lt;h6 class="card-title"&gt;{{ recipe.title }}&lt;/h6&gt;
                                &lt;p class="card-text"&gt;{{ recipe.description|truncatewords:15 }}&lt;/p&gt;
                            &lt;div class="card-meta"&gt;
                                &lt;small class="text-muted"&gt;
                                    &lt;strong&gt;Время:&lt;/strong&gt; {{ recipe.cooking_time }} мин.&lt;br&gt;
                                    &lt;strong&gt;Порции:&lt;/strong&gt; {{ recipe.servin
</pre>
```

```
gs }}<br>
                                <strong>Калории:</strong> {{ recipe.calories_per_100g }} ккал/100г<br>
                                <strong>Сложность:</strong> {{ recipe.get_difficulty_display }}<br>
                                </small>
                            </div>

                            <div class="mt-3">
                                <a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Просмотреть</a>
                                <a href="{% url 'recipes:recipe-update' recipe.pk %}" class="btn btn-outline-warning btn-sm">Редактировать</a>
                                <a href="{% url 'recipes:recipe-delete' recipe.pk %}" class="btn btn-outline-danger btn-sm">Удалить</a>
                            </div>
                        </div>
                        <div class="card-footer">
                            <small class="text-muted">Добавлен: {{ recipe.created_at|date:"d.m.Y" }}</small>
                        </div>
                    </div>
                </div>
            {% endfor %}
        </div>
        {% else %}
        <div class="text-center py-4">
            <p class="text-muted">У вас пока нет рецептов.</p>
            <a href="{% url 'recipes:recipe-create' %}" class="btn btn-success">Добавить первый рецепт</a>
        </div>
        {% endif %}
    </div>
</div>

<!-- Избранные рецепты --&gt;
&lt;div class="card mt-4"&gt;
    &lt;div class="card-header"&gt;
        &lt;h5 class="mb-0"&gt;Мои избранные рецепты ({{ userFavorites.count }})&lt;/h5&gt;
    &lt;/div&gt;
    &lt;div class="card-body"&gt;
        {% if userFavorites %}
        &lt;div class="row"&gt;
            {% for favorite in userFavorites %}
            &lt;div class="col-md-6 mb-3"&gt;
                &lt;div class="card recipe-card"&gt;</pre>
```

```

<div class="card-body">
    <h6 class="card-title">{{ favorite.recipe.title }}</h6>
    <p class="card-text">{{ favorite.recipe.description|truncatewords:15 }}</p>

    <div class="recipe-meta">
        <small class="text-muted">
            Автор: {{ favorite.recipe.author.username }}<br>
            Добавлен в избранное: {{ favorite.added_at|date:"d.m.Y" }}<br>
        </small>
    </div>

    <div class="mt-3">
        <a href="{% url 'recipes:recipe-detail' favorite.recipe.pk %}" class="btn btn-outline-success btn-sm">Просмотреть</a>
        <a href="{% url 'recipes:add-to-favorites' favorite.recipe.pk %}" class="btn btn-outline-danger btn-sm">Удалить из избранного</a>
    </div>
    </div>
    {% endfor %}
</div>
{% else %}
    <div class="text-center py-4">
        <p class="text-muted">У вас пока нет избранных рецептов.</p>
        <a href="{% url 'recipes:home' %}" class="btn btn-outline-success">Найти рецепты</a>
    </div>
    {% endif %}
</div>
</div>
<% endblock %>
```

## Шаг 3: Проверка моделей

Убедитесь, что в `recipes/models.py` есть правильная связь с пользователем:

`python`

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

class Recipe(models.Model):
    # ... другие поля ...
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    # ... другие поля ...

    def __str__(self):
        return self.title

class Favorite(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    recipe = models.ForeignKey(Recipe, on_delete=models.CASCADE)
    added_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ('user', 'recipe')
```

## Шаг 4: Создание тестовых данных

Если рецептов все еще нет, создайте несколько тестовых рецептов через Django shell:

bash

```
python manage.py shell
```

python

```
from django.contrib.auth.models import User
from recipes.models import Recipe

# Получите пользователя
user = User.objects.get(username=' ваш_username ')

# Создайте тестовые рецепты
recipe1 = Recipe.objects.create(
    title="Тестовый рецепт 1",
    description="Это описание первого тестового рецепта",
    author=user,
    cooking_time=30,
    servings=4,
    calories_per_100g=150,
    difficulty='easy'
```

```
        )

recipe2 = Recipe.objects.create(
    title="Тестовый рецепт 2",
    description="Это описание второго тестового рецепта",
    author=user,
    cooking_time=45,
    servings=2,
    calories_per_100g=200,
    difficulty='medium'

)

print("Тестовые рецепты созданы!")
exit()
```

## Шаг 5: Проверка работы

После внесения изменений:

1. Перезагрузите страницу профиля <http://127.0.0.1:8000/users/profile/>
2. Убедитесь, что рецепты отображаются
3. Проверьте статистику - должны отображаться правильные цифры

Теперь в профиле должны отображаться все рецепты, созданные пользователем, а также выбранные рецепты и комментарии!

Произошла ошибка при загрузке профиля: get() returned more than one Recipe -- it returned 2!

Ошибка возникает потому что где-то в коде используется `Recipe.objects.get()` когда нужно использовать `Recipe.objects.filter()`. Давайте найдем и исправим эту проблему.

## Шаг 1: Поиск проблемы

Скорее всего проблема в одном из этих мест:

1. В представлениях где используется `get()` вместо `filter()`
2. В шаблоне где происходит обращение к связанным объектам
3. В URL конфигурации

## Шаг 2: Проверка и исправление views.py

recipes/views.py - проверьте все места где используется `get_object_or_404` или `get()`:

python

```
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag
from .forms import RecipeForm

class RecipeListView(ListView):
    model = Recipe
    template_name = 'recipes/home.html'
    context_object_name = 'recipes'
    paginate_by = 9

    def get_queryset(self):
        queryset = Recipe.objects.all().order_by('-created_at')

        # Поиск по ключевым словам
        query = self.request.GET.get('q')
        if query:
            queryset = queryset.filter(
                Q(title__icontains=query) |
                Q(description__icontains=query) |
                Q(ingredients__name__icontains=query)
            ).distinct()

        # Фильтрация по хештегам
        hashtag = self.request.GET.get('hashtag')
        if hashtag:
            queryset = queryset.filter(hashtags__name=hashtag)

        # Фильтрация по калорийности
        max_calories = self.request.GET.get('max_calories')
        if max_calories:
            queryset = queryset.filter(calories_per_100g__lte=max_calories)

    return queryset

class RecipeDetailView(DetailView):
```

```
model = Recipe
template_name = 'recipes/recipe_detail.html'

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs['user'] = self.request.user
        return kwargs

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

class RecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
            return redirect('recipes:recipe-detail', pk=obj.pk)
        return super().dispatch(request, *args, **kwargs)

class RecipeDeleteView(LoginRequiredMixin, DeleteView):
    model = Recipe
    template_name = 'recipes/recipe_confirm_delete.html'
    success_url = '/'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
            return redirect('recipes:recipe-detail', pk=obj.pk)
        return super().dispatch(request, *args, **kwargs)

@login_required
def add_to_favorites(request, pk):
    recipe = get_object_or_404(Recipe, pk=pk) # Это правильно – ищем по pk
    favorite, created = Favorite.objects.get_or_create(user=request.user, recipe=recipe)
    if not created:
        favorite.delete()
    return redirect('recipes:recipe-detail', pk=pk)
```

```

def search_recipes(request):
    query = request.GET.get('q', '')
    hashtag = request.GET.get('hashtag', '')
    max_calories = request.GET.get('max_calories', '')

    recipes = Recipe.objects.all()

    if query:
        recipes = recipes.filter(
            Q(title__icontains=query) |
            Q(description__icontains=query) |
            Q(ingredients__name__icontains=query)
        ).distinct()

    if hashtag:
        recipes = recipes.filter(hashtags__name=hashtag)

    if max_calories:
        recipes = recipes.filter(calories_per_100g__lte=int(max_calories))

    context = {
        'recipes': recipes,
        'query': query,
        'hashtag': hashtag,
        'max_calories': max_calories,
    }

    return render(request, 'recipes/search_results.html', context)

```

## Шаг 3: Исправление users/views.py

**users/views.py** - убедитесь, что везде используется `filter()` а не `get()`:

python

```

from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from recipes.models import Recipe, Favorite
from comments.models import Comment
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):
    if request.method == 'POST':

```

```
form = UserRegisterForm(request.POST)
if form.is_valid():
    user = form.save()
    username = form.cleaned_data.get('username')

    # Автоматический вход после регистрации
    raw_password = form.cleaned_data.get('password1')
    user = authenticate(username=username, password=raw_password)
    if user is not None:
        login(request, user)

    messages.success(request, f'Аккаунт {username} был успешно создан!')
    return redirect('recipes:home')

else:
    form = UserRegisterForm()

return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    """Полная версия профиля с рецептами пользователя"""
    try:
        # Получаем рецепты пользователя - используем filter() а не get()
        user_recipes = Recipe.objects.filter(author=request.user).order_by('-created_at')

        # Получаем комментарии пользователя
        user_comments = Comment.objects.filter(author=request.user)

        # Получаем избранные рецепты пользователя
        userFavorites = Favorite.objects.filter(user=request.user)

        if request.method == 'POST':
            u_form = UserUpdateForm(request.POST, instance=request.user)
            p_form = ProfileUpdateForm(
                request.POST,
                request.FILES,
                instance=request.user.profile
            )

            if u_form.is_valid() and p_form.is_valid():
                u_form.save()
                p_form.save()
                messages.success(request, 'Ваш профиль был успешно обновлен!')
                return redirect('users:profile')

            else:
                messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
    except:
        pass
```

```

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form,
        'user_recipes': user_recipes,
        'user_comments': user_comments,
        'userFavorites': userFavorites,
    }

    return render(request, 'users/profile.html', context)

except Exception as e:
    # Резервный вариант на случай ошибки
    messages.error(request, f'Произошла ошибка при загрузке профиля: {str(e)}')
    return render(request, 'users/profile_simple.html', {
        'user': request.user
})

```

## Шаг 4: Создание безопасного шаблона профиля

`templates/users/profile_simple.html` (на случай ошибок):

```

html

{% extends 'base.html' %}

{% block title %}Профиль – {{ user.username }}{% endblock %}

{% block content %}


#### Профиль пользователя: {{ user.username }}



##### Базовая информация



<strong>Имя пользователя:</strong> {{ user.username }}</p>
<strong>Email:</strong> {{ user.email|default:"Не указан" }}</p>
<strong>Дата регистрации:</strong> {{ user.date_joined|dat
}}


```

```

e:"d.m.Y H:i" }}</p>
        </div>

        <div class="text-center">
            <a href="{% url 'recipes:home' %}" class="btn btn-success">На
главную</a>
            <a href="{% url 'recipes:recipe-create' %}" class="btn btn-o
tline-success">Добавить рецепт</a>
        </div>
    </div>
</div>
{% endblock %}

```

## Шаг 5: Диагностика проблемы

Давайте найдем где именно происходит ошибка. Добавьте отладочную информацию:

`users/views.py` (диагностическая версия):

`python`

```

@login_required
def profile(request):
    """Версия с отладкой для поиска проблемы"""
    try:
        print("== НАЧАЛО ЗАГРУЗКИ ПРОФИЛЯ ==")

        # 1. Проверяем загрузку рецептов
        print("1. Загружаем рецепты пользователя...")
        user_recipes = Recipe.objects.filter(author=request.user).order_by('-crea
ted_at')
        print(f"    Найдено рецептов: {user_recipes.count()}")

        # 2. Проверяем загрузку комментариев
        print("2. Загружаем комментарии пользователя...")
        user_comments = Comment.objects.filter(author=request.user)
        print(f"    Найдено комментариев: {user_comments.count()}")

        # 3. Проверяем загрузку избранного
        print("3. Загружаем избранные рецепты...")
        userFavorites = Favorite.objects.filter(user=request.user)
        print(f"    Найдено избранных: {userFavorites.count()}")

    print("== ВСЕ ДАННЫЕ УСПЕШНО ЗАГРУЖЕНЫ ==")

```

```

if request.method == 'POST':
    u_form = UserUpdateForm(request.POST, instance=request.user)
    p_form = ProfileUpdateForm(request.POST, request.FILES, instance=request.user.profile)

    if u_form.is_valid() and p_form.is_valid():
        u_form.save()
        p_form.save()
        messages.success(request, 'Ваш профиль был успешно обновлен!')
        return redirect('users:profile')

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form,
        'user_recipes': user_recipes,
        'user_comments': user_comments,
        'userFavorites': userFavorites,
    }

return render(request, 'users/profile.html', context)

except Exception as e:
    print(f"== ОШИБКА: {str(e)} ==")
    messages.error(request, f'Произошла ошибка при загрузке профиля: {str(e)}')
    # Возвращаем упрощенную версию без данных
    return render(request, 'users/profile_simple.html', {'user': request.user})

```

## Шаг 6: Проверка работы

1. Запустите сервер и перейдите на страницу профиля
2. Посмотрите в консоль Django - там будут отладочные сообщения
3. Определите, на каком шаге происходит ошибка

## Шаг 7: Временное решение

Если не удается найти проблему, используйте эту минимальную версию:

`users/views.py` (минимальная рабочая версия):

```

python

@login_required
def profile(request):
    """Минимальная рабочая версия профиля"""
    # Только базовые данные
    user_recipes_count = Recipe.objects.filter(author=request.user).count()
    user_comments_count = Comment.objects.filter(author=request.user).count()
    userFavorites_count = Favorite.objects.filter(user=request.user).count()

    context = {
        'user': request.user,
        'recipes_count': user_recipes_count,
        'comments_count': user_comments_count,
        'favorites_count': userFavorites_count,
    }

    return render(request, 'users/profile_minimal.html', context)

```

**templates/users/profile\_minimal.html:**

```

html

{% extends 'base.html' %}

{% block title %}Профиль – {{ user.username }}{% endblock %}

{% block content %}


#### Профиль: {{ user.username }}



### {{ recipes_count }}



Рецептов



### {{ comments_count }}



Комментариев



### {{ favorites_count }}



В избранном


```

```

        </div>
    </div>

    <div class="text-center mt-4">
        <a href="{% url 'recipes:recipe-create' %}" class="btn btn-success">Добавить рецепт</a>
        <a href="{% url 'recipes:home' %}" class="btn btn-outline-secondary">На главную</a>
    </div>
</div>
</div>
{% endblock %}

```

Попробуйте сначала диагностическую версию (**Шаг 5**) и сообщите, что вы видите в консоли Django - это поможет точно определить где проблема.

Починила. Всё ещё не работает ссылка  
<http://127.0.0.1:8000/users/logout/>

Отлично, что починили профиль! Теперь давайте исправим ссылку logout. Проблема скорее всего в настройках Django для выхода из системы.

## Шаг 1: Проверка и исправление URLs

`users/urls.py` (убедитесь, что есть правильный путь для `logout`):

```

python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'), name='logout'),
    path('profile/', views.profile, name='profile'),
]

```

## Шаг 2: Проверка шаблона logout.html

templates/users/logout.html (убедитесь, что файл существует и содержит):

```
html

{% extends 'base.html' %}

{% block title %}Выход – Кулинарный сайт{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">Выход из системы</h4>
            </div>
            <div class="card-body text-center">
                <div class="mb-4">
                    <h4>Вы успешно вышли из системы</h4>
                    <p class="text-muted">Спасибо, что посетили наш кулинарный са
йт!</p>
                </div>

                <div class="d-grid gap-2">
                    <a href="{% url 'users:login' %}" class="btn btn-success">Вой
ти снова</a>
                    <a href="{% url 'recipes:home' %}" class="btn btn-outline-sec
ondary">Вернуться на главную</a>
                </div>
            </div>
        </div>
    </div>
{% endblock %}
```

## Шаг 3: Настройка в settings.py

recipesAlmanah\_project/settings.py (добавьте эти настройки):

python

```
# Настройки аутентификации
LOGIN_REDIRECT_URL = 'recipes:home'      # Куда перенаправлять после входа
LOGIN_URL = 'users:login'                 # URL для входа
```

```
LOGOUT_REDIRECT_URL = 'recipes:home' # Куда перенаправлять после выхода

# Или если хотите использовать шаблон logout.html, уберите LOGOUT_REDIRECT_URL
# LOGOUT_REDIRECT_URL = None
```

## Шаг 4: Альтернативное решение - создание собственного представления для logout

Если встроенное представление не работает, создайте свое:

**users/views.py** (добавьте эту функцию):

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect

def custom_logout(request):
    """Кастомное представление для выхода из системы"""
    logout(request)
    return render(request, 'users/logout.html')
```

**users/urls.py** (обновите для использования кастомного представления):

```
python

from django.urls import path
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
    # path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.h
tml'), name='logout'), # закомментируйте эту строку
    path('logout/', views.custom_logout, name='logout'), # используйте кастомное
представление
    path('profile/', views.profile, name='profile'),
]
```

## Шаг 5: Проверка POST запроса для logout

Django по умолчанию требует POST запрос для выхода. Обновите навигацию в [base.html](#):

html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный сайт</a>
        <div class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">Добавить рецепт</a>
                <a class="nav-link" href="{% url 'users:profile' %}">Профиль</a>
                <form method="post" action="{% url 'users:logout' %}" class="d-inline">
                    {% csrf_token %}
                    <button type="submit" class="btn btn-link nav-link" style="border: none; background: none; color: rgba(255,255,255,.55); padding: 0.5rem 1rem;">
                        Выйти ({{ user.username }})
                    </button>
                </form>
            {% else %}
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>
                <a class="nav-link" href="{% url 'users:register' %}">Регистрация</a>
            {% endif %}
        </div>
    </div>
</nav>
```

## Шаг 6: Упрощенное решение - GET запрос для logout

Если хотите оставить простую ссылку, разрешите GET запросы:

[users/views.py](#):

python

```
from django.contrib.auth import logout
from django.shortcuts import redirect
from django.views.decorators.http import require_http_methods

@require_http_methods(["GET", "POST"])
```

```
def custom_logout(request):
    """Кастомное представление для выхода из системы с поддержкой GET"""
    logout(request)
    # Можно добавить сообщение об успешном выходе
    from django.contrib import messages
    messages.success(request, 'Вы успешно вышли из системы.')
    return render(request, 'users/logout.html')
```

## Шаг 7: Диагностика проблемы

Чтобы понять, что именно не работает, добавьте отладочную информацию:

1. Проверьте URL в браузере - правильно ли он введен?
2. Посмотрите ошибку в консоли Django - какая именно ошибка появляется?
3. Проверьте существование файла `templates/users/logout.html`

## Шаг 8: Быстрое решение

Самый простой рабочий вариант - использовать кастомное представление:

`users/views.py`:

```
python

from django.contrib.auth import logout
from django.shortcuts import render

def custom_logout(request):
    logout(request)
    return render(request, 'users/logout.html')
```

`users/urls.py`:

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
```

```
path('login/', auth_views.LoginView.as_view(template_name='users/login.htm  
l'), name='login'),  
    path('logout/', views.custom_logout, name='logout'), # кастомный logout  
    path('profile/', views.profile, name='profile'),  
]
```

**base.html** (обновите ссылку):

html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-success">  
    <div class="container">  
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный сайт</a>  
        <div class="navbar-nav ms-auto">  
            {% if user.is_authenticated %}  
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">Добавить рецепт</a>  
                <a class="nav-link" href="{% url 'users:profile' %}">Профиль</a>  
                <a class="nav-link" href="{% url 'users:logout' %}">Выйти ({{ user.username }})</a>  
            {% else %}  
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>  
                <a class="nav-link" href="{% url 'users:register' %}">Регистрация</a>  
            {% endif %}  
        </div>  
    </div>  
</nav>
```

Попробуйте сначала Шаг 8 (быстрое решение с кастомным logout) - он должен гарантированно работать!

DisallowedRedirect at /users/logout/

Ошибка DisallowedRedirect возникает из-за настроек безопасности Django. Давайте исправим это несколькими способами:

## Шаг 1: Настройка ALLOWED\_HOSTS и безопасных перенаправлений

`recipesAlmanah_project/settings.py` (добавьте/обновите эти настройки):

python

```
# Базовые настройки
DEBUG = True
ALLOWED_HOSTS = ['127.0.0.1', 'localhost']

# Настройки аутентификации
LOGIN_REDIRECT_URL = 'recipes:home'
LOGIN_URL = 'users:login'
LOGOUT_REDIRECT_URL = 'recipes:home' # Убедитесь, что это установлено

# Безопасные перенаправления
CSRF_TRUSTED_ORIGINS = ['http://127.0.0.1:8000', 'http://localhost:8000']
```

## Шаг 2: Исправление кастомного представления logout

users/views.py (исправленная версия):

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect, render
from django.views.decorators.http import require_http_methods

@require_http_methods(["GET", "POST"])
def custom_logout(request):
    """Кастомное представление для выхода из системы"""
    logout(request)
    # Перенаправляем на страницу выхода с сообщением
    return render(request, 'users/logout.html')
```

## Шаг 3: Альтернативное решение - использовать встроенный LogoutView с правильными настройками

users/urls.py (используем встроенное представление с правильными параметрами):

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
```

```

path('register/', views.register, name='register'),
path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
path('logout/', auth_views.LogoutView.as_view(
    template_name='users/logout.html',
    next_page='recipes:home' # Явно указываем куда перенаправлять
), name='logout'),
path('profile/', views.profile, name='profile'),
]

```

## Шаг 4: Решение с POST запросом (рекомендуется для безопасности)

Обновите **base.html** для использования формы с POST запросом:

html

```

<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный сайт</
a>
        <div class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">Добавить рецепт</a>
                <a class="nav-link" href="{% url 'users:profile' %}">Профиль</a>

                <!-- Форма для выхода с POST запросом -->
                <li class="nav-item">
                    <form method="post" action="{% url 'users:logout' %}" class
="d-inline">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-link nav-link" style
="border: none; background: none; color: rgba(255,255,255,.55); padding: 0.5rem 1
rem; cursor: pointer;">
                            Выйти ({{ user.username }})
                        </button>
                    </form>
                </li>
            {% else %}
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>
                <a class="nav-link" href="{% url 'users:register' %}">Регистрация
            </a>
            {% endif %}
        </div>

```

```
</div>
</nav>
```

## Шаг 5: Простое решение с GET запросом (менее безопасно, но работает)

Если хотите использовать простую ссылку, создайте специальное представление:

**users/views.py:**

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect
from django.views.decorators.http import require_http_methods

@require_http_methods(["GET"])
def simple_logout(request):
    """Простое представление для выхода по GET запросу"""
    logout(request)
    # Перенаправляем на страницу подтверждения выхода
    return redirect('users:logout_success')

def logout_success(request):
    """Страница подтверждения выхода"""
    return render(request, 'users/logout.html')
```

**users/urls.py:**

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.htm
l'), name='login'),
    path('logout/', views.simple_logout, name='logout'), # GET logout
    path('logout/success/', views.logout_success, name='logout_success'), # стра
нича подтверждения
```

```
        path('profile/', views.profile, name='profile'),
    ]
```

**base.html** (обновите ссылку):

```
html

<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный сайт</a>
        <div class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">Добавить рецепт</a>
                <a class="nav-link" href="{% url 'users:profile' %}">Профиль</a>
                <a class="nav-link" href="{% url 'users:logout' %}">Выйти ({{ user.username }})</a>
            {% else %}
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>
                <a class="nav-link" href="{% url 'users:register' %}">Регистрация</a>
            {% endif %}
        </div>
    </div>
</nav>
```

## Шаг 6: Самый простой рабочий вариант

**users/views.py:**

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect

def simple_logout(request):
    """Самый простой рабочий logout"""
    logout(request)
    # Просто перенаправляем на главную страницу
    return redirect('recipes:home')
```

**users/urls.py:**

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
    path('logout/', views.simple_logout, name='logout'),
    path('profile/', views.profile, name='profile'),
]
```

## Шаг 7: Проверка работы

После внесения изменений:

1. Остановите сервер (Ctrl+C)
2. Запустите снова:

```
bash
```

```
python manage.py runserver
```

3. Проверьте выход - теперь он должен работать

## Рекомендую попробовать в таком порядке:

1. Сначала Шаг 6 (самый простой вариант) - просто выход и перенаправление на главную
2. Если нужно красивую страницу выхода - используйте Шаг 5
3. Для максимальной безопасности - используйте Шаг 4 с POST запросом

Какой вариант пробуете первым?

The current path, users/logout/users/logout.html/, didn't match any of these.

Ошибка указывает на неправильный путь в шаблоне. Проблема в том, что в `logout.html` есть ссылка, которая формирует неправильный URL. Давайте исправим это:

## Шаг 1: Исправление шаблона `logout.html`

`templates/users/logout.html` (исправленная версия):

`html`

```
{% extends 'base.html' %}

{% block title %}Выход – Кулинарный сайт{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">Выход из системы</h4>
            </div>
            <div class="card-body text-center">
                <div class="mb-4">
                    <h4>Вы успешно вышли из системы</h4>
                    <p class="text-muted">Спасибо, что посетили наш кулинарный са
йт!</p>
                </div>

                <div class="d-grid gap-2">
                    <!-- ИСПРАВЛЕННЫЕ ССЫЛКИ -->
                    <a href="{% url 'users:login' %}" class="btn btn-success">Вой
ти снова</a>
                    <a href="{% url 'recipes:home' %}" class="btn btn-outline-sec
ondary">Вернуться на главную</a>
                </div>
            </div>
        </div>
    </div>
{% endblock %}
```

## Шаг 2: Исправление представления `logout`

`users/views.py` (исправленная версия):

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect, render
```

```
def custom_logout(request):
    """Кастомное представление для выхода из системы"""
    logout(request)
    # Рендерим шаблон напрямую, без перенаправления
    return render(request, 'users/logout.html')
```

## Шаг 3: Проверка URLs

users/urls.py (убедитесь, что пути правильные):

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
    path('logout/', views.custom_logout, name='logout'), # используем кастомное представление
    path('profile/', views.profile, name='profile'),
]
```

## Шаг 4: Альтернативное решение - простой redirect

Если проблема сохраняется, используйте простой вариант:

users/views.py (альтернативная версия):

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect

def simple_logout(request):
```

```
"""Простой выход с перенаправлением на главную"""
def logout(request):
    return redirect('recipes:home')
```

И обновите **base.html** чтобы показывать сообщение о выходе:

**templates/base.html** (добавьте этот скрипт перед закрывающим тегом body):

html

```
<!-- ... остальной код ... -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

<!-- Скрипт для показа сообщения о выходе -->
<script>
    // Проверяем параметр URL для показа сообщения о выходе
    const urlParams = new URLSearchParams(window.location.search);
    if (urlParams.has('logout') && urlParams.get('logout') === 'success') {
        // Показываем всплывающее сообщение
        alert('Вы успешно вышли из системы!');
    }
</script>
</body>
</html>
```

И обновите представление:

**users/views.py**:

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect

def simple_logout(request):
    """Простой выход с перенаправлением на главную с параметром"""
    logout(request)
    return redirect('recipes:home' + '?logout=success')
```

## Шаг 5: Самый надежный вариант

Создайте отдельную страницу подтверждения выхода:

## users/views.py:

```
python

from django.contrib.auth import logout
from django.shortcuts import redirect, render

def logout_view(request):
    """Обработчик выхода"""
    if request.user.is_authenticated:
        logout(request)
        return render(request, 'users/logout_success.html')
    else:
        # Если пользователь уже вышел, просто показываем страницу
        return render(request, 'users/logout_success.html')

def logout_success(request):
    """Страница успешного выхода"""
    return render(request, 'users/logout_success.html')
```

## users/urls.py:

```
python

from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'users'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
    path('logout/', views.logout_view, name='logout'),
    path('logout/success/', views.logout_success, name='logout_success'),
    path('profile/', views.profile, name='profile'),
]
```

## templates/users/logout\_success.html (новый файл):

```
html

{% extends 'base.html' %}

{% block title %}Выход выполнен – Кулинарный сайт{% endblock %}
```

```

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">Выход выполнен</h4>
            </div>
            <div class="card-body text-center">
                <div class="mb-4">
                    <h4>✓ Вы успешно вышли из системы</h4>
                    <p class="text-muted">Надеемся увидеть вас снова!</p>
                </div>

                <div class="d-grid gap-2">
                    <a href="{% url 'users:login' %}" class="btn btn-success">Войти снова</a>
                    <a href="{% url 'recipes:home' %}" class="btn btn-outline-secondary">На главную страницу</a>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}

```

## Шаг 6: Обновление навигации

**templates/base.html** (обновите ссылку выхода):

```

html

<nav class="navbar navbar-expand-lg navbar-dark bg-success">
    <div class="container">
        <a class="navbar-brand" href="{% url 'recipes:home' %}">Кулинарный сайт</a>
        <div class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'recipes:recipe-create' %}">Добавить рецепт</a>
                <a class="nav-link" href="{% url 'users:profile' %}">Профиль</a>
                <a class="nav-link" href="{% url 'users:logout' %}">Выйти ({{ user.username }})</a>
            {% else %}
                <a class="nav-link" href="{% url 'users:login' %}">Войти</a>
                <a class="nav-link" href="{% url 'users:register' %}">Регистрация</a>
            {% endif %}
        </div>
    </div>
</nav>

```

```
</a>
    {% endif %}
</div>
</div>
</nav>
```

## Рекомендую попробовать:

1. Сначала Шаг 1 и Шаг 2 - исправление шаблона и простой render
2. Если не работает - перейдите к Шаг 5 (отдельная страница подтверждения)

Попробуйте Шаг 1 и Шаг 2 - они самые простые и должны решить проблему с неправильным путем.

Всё работает. Теперь давай перейдём к работе с рецептами. При нажатии на кнопку Подробнее у рецепта мне выдаёт NoReverseMatch at /recipe/3/, Как починить

Ошибка NoReverseMatch означает, что Django не может найти URL с именем recipe-detail для рецепта с pk=3. Давайте исправим это:

## Шаг 1: Проверка URLs приложения recipes

recipes/urls.py (убедитесь, что есть правильный путь для детальной страницы):

```
python
```

```
from django.urls import path
from . import views

app_name = 'recipes'

urlpatterns = [
    path('', views.RecipeListView.as_view(), name='home'),
    path('recipe/<int:pk>/', views.RecipeDetailView.as_view(), name='recipe-detail'),
    # ← эта строка важна
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recipe-update'),
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recipe-delete'),
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favorites'),
```

```
        path('search/', views.search_recipes, name='search-recipes'),
    ]
```

## Шаг 2: Проверка главного urls.py

`recipesAlmanah_project/urls.py` (убедитесь, что включены URLs приложения `recipes`):

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')), # ← URLs приложения recipes
    path('users/', include('users.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Шаг 3: Проверка шаблона home.html

`templates/recipes/home.html` (убедитесь, что ссылка использует правильное пространство имен):

```
html

{% extends 'base.html' %}

{% block title %}Главная страница – Кулинарный сайт{% endblock %}

{% block content %}


# Рецепты


<form method="get" action="{% url 'recipes:search-recipes' %}" class="row g-3">
    <div class="col-md-4">


```

```
        <input type="text" name="q" class="form-control" placeholder="Поиск рецептов..." value="{{ request.GET.q }}">
    </div>
    <div class="col-md-3">
        <input type="number" name="max_calories" class="form-control" placeholder="Макс. калории" value="{{ request.GET.max_calories }}"/>
    </div>
    <div class="col-md-3">
        <button type="submit" class="btn btn-success">Найти</button>
    </div>
</form>
</div>
</div>

{% if recipes %}
<div class="row">
    {% for recipe in recipes %}
<div class="col-lg-4 col-md-6 mb-4">
    <div class="card recipe-card">
        <div class="card-body">
            <h5 class="card-title">{{ recipe.title }}</h5>
            <p class="card-text">{{ recipe.description|truncatewords:15 }}</p>
>

        <div class="recipe-meta">
            <small class="text-muted">
                <strong>Время:</strong> {{ recipe.cooking_time }} мин.<br>
            <strong>Порции:</strong> {{ recipe.servings }}<br>
            <strong>Калории:</strong> {{ recipe.calories_per_100g }}  

            ккал/100г<br>
                <strong>Сложность:</strong> {{ recipe.get_difficulty_display }}<br>
            </small>
        </div>

        <div class="mt-3">
            <!-- ПРАВИЛЬНАЯ ССЫЛКА С ПРОСТРАНСТВОМ ИМЕН -->
            <a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>
            {% if user.is_authenticated %}
                <a href="{% url 'recipes:add-to-favorites' recipe.pk %}" class="btn btn-outline-primary btn-sm">В избранное</a>
            {% endif %}
        </div>
    </div>
<div class="card-footer">
```

```
<small class="text-muted">Автор: {{ recipe.author.username }}</small>
all>
    </div>
</div>
</div>
{% endfor %}
</div>

<!-- Пагинация --&gt;
{% if is_paginated %}
&lt;nav aria-label="Page navigation"&gt;
    &lt;ul class="pagination justify-content-center"&gt;
        {% if page_obj.has_previous %}
            &lt;li class="page-item"&gt;
                &lt;a class="page-link" href="?page={{ page_obj.previous_page_number }}"&gt;Предыдущая&lt;/a&gt;
            &lt;/li&gt;
        {% endif %}

        {% for num in page_obj.paginator.page_range %}
            {% if page_obj.number == num %}
                &lt;li class="page-item active"&gt;
                    &lt;a class="page-link" href="?page={{ num }}"&gt;{{ num }}&lt;/a&gt;
                &lt;/li&gt;
            {% else %}
                &lt;li class="page-item"&gt;
                    &lt;a class="page-link" href="?page={{ num }}"&gt;{{ num }}&lt;/a&gt;
                &lt;/li&gt;
            {% endif %}
        {% endfor %}

        {% if page_obj.has_next %}
            &lt;li class="page-item"&gt;
                &lt;a class="page-link" href="?page={{ page_obj.next_page_number }}"&gt;Следующая&lt;/a&gt;
            &lt;/li&gt;
        {% endif %}
    &lt;/ul&gt;
&lt;/nav&gt;
{% endif %}

{% else %}
&lt;div class="alert alert-info"&gt;
    &lt;h4&gt;Рецептов пока нет&lt;/h4&gt;
    &lt;p&gt;Будьте первым, кто добавит рецепт!&lt;/p&gt;
    {% if user.is_authenticated %}
        &lt;a href="{% url 'recipes:recipe-create' %}" class="btn btn-success"&gt;Добав</pre>
```

```

ить первый рецепт</a>
    {% else %}
        <a href="{% url 'users:login' %}" class="btn btn-success">Войти и добавит
ь рецепт</a>
    {% endif %}
</div>
{% endif %}
{% endblock %}

```

## Шаг 4: Создание шаблона recipe\_detail.html

templates/recipes/recipe\_detail.html (убедитесь, что файл существует):

html

```

{% extends 'base.html' %}

{% block title %}{{ recipe.title }} - Кулинарный сайт{% endblock %}

{% block content %}


<div class="col-md-8 mx-auto">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h2 class="mb-0">{{ recipe.title }}</h2>
            </div>
            <div class="card-body">
                <div class="row mb-4">
                    <div class="col-md-6">
                        <p><strong>Автор:</strong> {{ recipe.author.username }}</p>
                        <p><strong>Время приготовления:</strong> {{ recipe.cooking_time }} минут</p>
                        <p><strong>Порции:</strong> {{ recipe.servings }}</p>
                    </div>
                    <div class="col-md-6">
                        <p><strong>Калорийность:</strong> {{ recipe.calories_per_100g }} ккал/100г</p>
                        <p><strong>Сложность:</strong> {{ recipe.get_difficulty_display }}</p>
                        <p><strong>Дата добавления:</strong> {{ recipe.created_at |date:"d.m.Y" }}</p>
                    </div>
                </div>
            </div>
        </div>
    </div>


```

```
<p class="mb-4">{{ recipe.description }}</p>

    {% if recipe.ingredients.all %}
        <h5>Ингредиенты</h5>
        <ul class="list-group mb-4">
            {% for ingredient in recipe.ingredients.all %}
                <li class="list-group-item">{{ ingredient.name }} - {{ ingredient.quantity }}</li>
            {% endfor %}
        </ul>
    {% endif %}

    {% if recipe.cooking_steps.all %}
        <h5>Шаги приготовления</h5>
        <div class="mb-4">
            {% for step in recipe.cooking_steps.all %}
                <div class="card mb-2">
                    <div class="card-body">
                        <h6>Шаг {{ step.step_number }}</h6>
                        <p class="mb-0">{{ step.description }}</p>
                        {% if step.photo %}
                            
                        {% endif %}
                    </div>
                </div>
            {% endfor %}
        </div>
    {% endif %}

    {% if recipe.hashtags.all %}
        <h5>Хештеги</h5>
        <div class="mb-4">
            {% for hashtag in recipe.hashtags.all %}
                <span class="badge bg-secondary me-1">#{{ hashtag.name }}</span>
            {% endfor %}
        </div>
    {% endif %}

<div class="card-footer">
    <div class="btn-group">
        <a href="{% url 'recipes:home' %}" class="btn btn-outline-secondary">Назад к рецептам</a>
        {% if user.is_authenticated %}
            <a href="{% url 'recipes:add-to-favorites' recipe.pk %}" class="btn btn-outline-primary">
```

```

        {% if user in recipe.favorite_set.all %}удалить из из
бранных{% else %}В избранные{% endif %}
        </a>
    {% endif %}
    {% if user == recipe.author or user.is_superuser %}
        <a href="{% url 'recipes:recipe-update' recipe.pk %}" cla
ss="btn btn-outline-warning">Редактировать</a>
        <a href="{% url 'recipes:recipe-delete' recipe.pk %}" cla
ss="btn btn-outline-danger">Удалить</a>
    {% endif %}
    </div>
</div>
</div>
</div>
<% endblock %>
```

## Шаг 5: Проверка представлений

`recipes/views.py` (убедитесь, что `RecipeDetailView` существует):

```

python

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, De
leteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag
from .forms import RecipeForm

class RecipeListView(ListView):
    model = Recipe
    template_name = 'recipes/home.html'
    context_object_name = 'recipes'
    paginate_by = 9

    def get_queryset(self):
        queryset = Recipe.objects.all().order_by('-created_at')
        # ... остальной код поиска и фильтрации
        return queryset

class RecipeDetailView(DetailView): # ← этот класс должен существовать
    model = Recipe
    template_name = 'recipes/recipe_detail.html'
```

```
# ... остальные представления
```

## Шаг 6: Диагностика проблемы

Если ошибка сохраняется, выполните диагностику:

bash

```
# Показать все доступные URL
python manage.py show_urls
```

```
# Проверить Django
python manage.py check
```

## Шаг 7: Временное решение для теста

Создайте простой тестовый шаблон:

[templates/recipes/recipe\\_test.html](#):

html

```
{% extends 'base.html' %}

{% block title %}Тест рецепта{% endblock %}

{% block content %}


## Тестовая страница рецепта



### Рецепт #{{ recipe.pk }}



Название: {{ recipe.title }}



Автор: {{ recipe.author.username }}



Описание: {{ recipe.description }}

Назад


{% endblock %}
```

И временное представление в `recipes/views.py`:

```
python
```

```
def test_recipe_detail(request, pk):
    recipe = get_object_or_404(Recipe, pk=pk)
    return render(request, 'recipes/recipe_test.html', {'recipe': recipe})
```

Добавьте временный URL в `recipes/urls.py`:

```
python
```

```
path('test/<int:pk>', views.test_recipe_detail, name='test-recipe-detail'),
```

## Шаг 8: Проверка по порядку

1. Проверьте `http://127.0.0.1:8000/recipe/3/` - какая точная ошибка?
2. Проверьте `http://127.0.0.1:8000/test/3/` - работает ли тестовый URL?
3. Выполните `python manage.py show_urls` - есть ли там `recipe-detail` ?

Попробуйте сначала Шаги 1-3 - они должны решить проблему. Если не поможет, покажите точный текст ошибки из Django.

```
NoReverseMatch at /recipe/3/update/
```

Ошибка `NoReverseMatch at /recipe/3/update/` означает, что Django не может найти URL для обновления рецепта. Давайте исправим это:

## Шаг 1: Проверка URLs для обновления рецепта

`recipes/urls.py` (убедитесь, что есть путь для обновления):

```
python
```

```
from django.urls import path
from . import views

app_name = 'recipes'

urlpatterns = [
    path('', views.RecipeListView.as_view(), name='home'),
    path('recipe/<int:pk>', views.RecipeDetailView.as_view(), name='recipe-detail'),
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recip
```

```
e-update'), # ← эта строка
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recipe-delete'),
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favorites'),
    path('search/', views.search_recipes, name='search-recipes'),
]
```

## Шаг 2: Проверка представления RecipeUpdateView

recipes/views.py (убедитесь, что класс существует):

```
python

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag
from .forms import RecipeForm

class RecipeUpdateView(LoginRequiredMixin, UpdateView): # ← этот класс должен быть
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()
        if obj.author != self.request.user:
            return redirect('recipes:recipe-detail', pk=obj.pk)
        return super().dispatch(request, *args, **kwargs)
```

## Шаг 3: Создание шаблона формы рецепта

templates/recipes/recipe\_form.html (создайте этот файл):

```
html
{% extends 'base.html' %}

{% block title %}
```

```
{% if form.instance.pk %}Редактирование рецепта{% else %}Добавление рецепта%
endif %}
{% endblock %}

{% block content %}


#### {% if form.instance.pk %} Редактирование рецепта: {{ form.instance.title }} {% else %} Добавление нового рецепта {% endif %}



{% csrf_token %}

        {% for field in form %}


<label for="{{ field.id_for_label }}" class="form-label">
                {{ field.label }}
                {% if field.field.required %}<span class="text-danger">*</span>{% endif %}
            </label>
            {{ field }}

            {% if field.help_text %}


{{ field.help_text }}


            {% endif %}

            {% if field.errors %}


{{ field.errors }}


            {% endif %}


        {% endfor %}



Отмена

        {% if form.instance.pk %}Сохранить изменения{% else %}


```

```
%}Добавить рецепт{%
endif %}
        </button>
    </div>
</form>
</div>
</div>
</div>
</div>
{%
endblock %}
```

## Шаг 4: Проверка формы RecipeForm

recipes/forms.py (убедитесь, что форма существует):

```
python

from django import forms
from .models import Recipe

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {
            'description': forms.Textarea(attrs={'rows': 4, 'class': 'form-control', 'placeholder': 'Опишите ваш рецепт...'}),
            'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название рецепта'}),
            'cooking_time': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Время в минутах'}),
            'servings': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Количество порций'}),
            'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Калории на 100 грамм'}),
            'difficulty': forms.Select(attrs={'class': 'form-control'}),
            'main_photo': forms.FileInput(attrs={'class': 'form-control'})
        }
        labels = {
            'title': 'Название рецепта',
            'description': 'Описание',
            'cooking_time': 'Время приготовления (минуты)',
            'servings': 'Количество порций',
            'calories_per_100g': 'Калорийность на 100г',
            'difficulty': 'Сложность',
        }
```

```
        'main_photo': 'Главное фото',
    }
```

## Шаг 5: Обновление шаблона детальной страницы

templates/recipes/recipe\_detail.html (обновите ссылки):

html

```
{% extends 'base.html' %}

{% block title %}{{ recipe.title }} - Кулинарный сайт{% endblock %}

{% block content %}


## {{ recipe.title }}



Автор: {{ recipe.author.username }}



Время приготовления: {{ recipe.cooking_time }} минут



Порции: {{ recipe.servings }}



Калорийность: {{ recipe.calories_per_100g }} ккал/100г



Сложность: {{ recipe.get_difficulty_display }}



Дата добавления: {{ recipe.created_at |date:"d.m.Y" }}



Описание:



{{ recipe.description }}


```

```

                <a href="{% url 'recipes:home' %}" class="btn btn-outline
-secondary">Назад к рецептам</a>
                {% if user.is_authenticated %}
                    <a href="{% url 'recipes:add-to-favorites' recipe.pk
%}" class="btn btn-outline-primary">
                        В избранное
                    </a>
                {% endif %}
                {% if user == recipe.author or user.is_superuser %}
                    
                    <a href="{% url 'recipes:recipe-update' recipe.pk %}"
class="btn btn-outline-warning">Редактировать</a>
                    <a href="{% url 'recipes:recipe-delete' recipe.pk %}"
class="btn btn-outline-danger">Удалить</a>
                {% endif %}
            </div>
        </div>
    </div>
</div>
{% endblock %}

```

## Шаг 6: Диагностика проблемы

Выполните команды для проверки:

bash

```

# Показать все URL
python manage.py show_urls

# Проверить Django
python manage.py check

# Проверить конкретное приложение
python manage.py check recipes

```

## Шаг 7: Временное решение для теста

Создайте простую тестовую форму:

[templates/recipes/recipe\\_form\\_simple.html](#):

```

html

{% extends 'base.html' %}

{% block title %}Тест формы{% endblock %}

{% block content %}


#### Тестовая форма редактирования



Рецепт #{{ object.pk }} - {{ object.title }}


{% csrf_token %}


Название рецепта



Описание
{{ object.description }}



Сохранить
Отмена


{% endblock %}

```

И временное представление в `recipes/views.py`:

```

python

from django.views.generic import UpdateView

class SimpleRecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    template_name = 'recipes/recipe_form_simple.html'
    fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty']

    def dispatch(self, request, *args, **kwargs):
        obj = self.get_object()

```

```
    if obj.author != self.request.user:  
        return redirect('recipes:recipe-detail', pk=obj.pk)  
    return super().dispatch(request, *args, **kwargs)
```

Добавьте временный URL в `recipes/urls.py`:

```
python  
  
path('recipe/<int:pk>/simple-update/', views.SimpleRecipeUpdateView.as_view(), name='recipe-simple-update'),
```

## Шаг 8: Проверка по порядку

1. Проверьте существует ли файл `recipes/forms.py`
2. Проверьте существует ли класс `RecipeUpdateView` в `recipes/views.py`
3. Проверьте существует ли путь `recipe-update` в `recipes/urls.py`
4. Выполните `python manage.py show_urls | grep update`

Попробуйте сначала Шаги 1-4. Если проблема сохраняется, покажите:

1. Содержимое вашего `recipes/urls.py`
2. Содержимое `recipes/views.py` (часть с `RecipeUpdateView`)
3. Результат команды `python manage.py show_urls`

Reverse for 'recipe-detail' not found. 'recipe-detail' is not a valid view function or pattern name.

Ошибка указывает, что Django не может найти URL с именем `recipe-detail`. Это проблема с пространством имен или самим URL. Давайте исправим это:

## Шаг 1: Проверка пространства имен в URLs

`recipes/urls.py` (убедитесь, что есть `app_name` и правильные пути):

```
python  
  
from django.urls import path  
from . import views  
  
app_name = 'recipes' # ← ЭТА СТРОКА ОЧЕНЬ ВАЖНА!
```

```
urlpatterns = [
    path('', views.RecipeListView.as_view(), name='home'),
    path('recipe/<int:pk>/', views.RecipeDetailView.as_view(), name='recipe-detail'),
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recipe-update'),
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recipe-delete'),
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favorites'),
    path('search/', views.search_recipes, name='search-recipes'),
]
```

## Шаг 2: Проверка главного urls.py

recipesAlmanah\_project/urls.py:

```
python

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('recipes.urls')), # ← включение URLs приложения recipes
    path('users/', include('users.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

## Шаг 3: Проверка использования пространства имен в шаблонах

Во всех шаблонах должны использоваться URL с пространством имен `recipes::`

`templates/recipes/home.html` (пример правильного использования):

```
html
```

```
<a href="{% url 'recipes:recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>
```

### templates/recipes/recipe\_detail.html:

html

```
<a href="{% url 'recipes:home' %}" class="btn btn-outline-secondary">Назад к рецептам</a>
<a href="{% url 'recipes:recipe-update' recipe.pk %}" class="btn btn-outline-warning">Редактировать</a>
```

## Шаг 4: Альтернативное решение - убрать пространство имен

Если проблема с пространством имен, можно временно убрать его:

### recipes/urls.py (вариант без пространства имен):

python

```
from django.urls import path
from . import views

# app_name = 'recipes' # ← ЗАКОММЕНТИРОВАТЬ

urlpatterns = [
    path('', views.RecipeListView.as_view(), name='home'),
    path('recipe/<int:pk>/', views.RecipeDetailView.as_view(), name='recipe-detail'),
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recipe-update'),
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recipe-delete'),
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favorites'),
    path('search/', views.search_recipes, name='search-recipes'),
]
```

И обновить шаблоны (убрать `recipes:`):

### templates/recipes/home.html:

```
html
```

```
<a href="{% url 'recipe-detail' recipe.pk %}" class="btn btn-outline-success btn-sm">Подробнее</a>
```

## Шаг 5: Диагностика проблемы

Выполните команды для проверки:

```
bash
```

```
# Показать все URL с именами
python manage.py show_urls

# Должны видеть что-то вроде:
# /recipe/<pk>/ recipes.views.RecipeDetailView recipe-detail
```

## Шаг 6: Проверка существования файла urls.py

Убедитесь, что файл `recipes/urls.py` существует и импортируется:

`recipes/urls.py` должен существовать в структуре:

```
text
```

```
recipes/
└── urls.py      ← этот файл
    ├── views.py
    ├── models.py
    └── ...

```

## Шаг 7: Временное решение - прямые ссылки

Если ничего не помогает, используйте прямые ссылки временно:

`templates/recipes/home.html` (временное решение):

```
html
```

```
<!-- Вместо {% url 'recipes:recipe-detail' recipe.pk %} -->
<a href="/recipe/{{ recipe.pk }}/" class="btn btn-outline-success btn-sm">Подробн
ее</a>
```

`templates/recipes/recipe_detail.html:`

`html`

```
<!-- Вместо {% url 'recipes:recipe-update' recipe_pk %} -->
<a href="/recipe/{{ recipe_pk }}/update/" class="btn btn-outline-warning">Редакти-
ровать</a>
```

## Шаг 8: Проверка по порядку

1. Проверьте существует ли файл `recipes/urls.py`
2. Проверьте есть ли в нем `app_name = 'recipes'`
3. Проверьте есть ли путь с `name='recipe-detail'`
4. Выполните `python manage.py show_urls`

## Шаг 9: Создание тестового URL

Добавьте простой тестовый URL для проверки:

`recipes/urls.py:`

`python`

```
from django.urls import path
from . import views

app_name = 'recipes'

urlpatterns = [
    path('', views.RecipeListView.as_view(), name='home'),
    path('recipe/<int:pk>/', views.RecipeDetailView.as_view(), name='recipe-detai-
l'),
    path('recipe/new/', views.RecipeCreateView.as_view(), name='recipe-create'),
    path('recipe/<int:pk>/update/', views.RecipeUpdateView.as_view(), name='recip
e-update'),
    path('recipe/<int:pk>/delete/', views.RecipeDeleteView.as_view(), name='recip
e-delete'),
    path('recipe/<int:pk>/favorite/', views.add_to_favorites, name='add-to-favori-
tes'),
    path('search/', views.search_recipes, name='search-recipes'),

    # Тестовый URL
]
```

```
    path('test/', views.test_view, name='test'),  
]
```

## recipes/views.py:

python

```
def test_view(request):  
    return render(request, 'recipes/test.html')
```

## templates/recipes/test.html:

html

```
{% extends 'base.html' %}  
  
{% block content %}  
<div class="card">  
    <div class="card-body">  
        <h5>Тест URLs</h5>  
        <p>Проверка recipe-detail: {{ url 'recipes:recipe-detail' 1 }}</p>  
        <a href="{{ url 'recipes:home' }}>На главную</a>  
    </div>  
</div>  
{% endblock %}
```

Проверьте <http://127.0.0.1:8000/test/> - покажет ли он правильный URL?

Попробуйте сначала Шаг 1 и Шаг 5. Если покажите результат команды  
python manage.py show\_urls , я смогу точно определить проблему.

```
TypeError at /recipe/new/  
    BaseModelForm.__init__() got an unexpected keyword argument 'user'
```

Ошибка возникает потому что в форме RecipeForm нет поддержки аргумента user , но представление пытается его передать. Давайте исправим это:

## Шаг 1: Исправление формы RecipeForm

recipes/forms.py (добавьте поддержку user):

python

```

from django import forms
from .models import Recipe

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {
            'description': forms.Textarea(attrs={'rows': 4, 'class': 'form-control'}, placeholder='Опишите ваш рецепт...'),
            'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название рецепта'}),
            'cooking_time': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Время в минутах'}),
            'servings': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Количество порций'}),
            'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Калории на 100 грамм'}),
            'difficulty': forms.Select(attrs={'class': 'form-control'}),
            'main_photo': forms.FileInput(attrs={'class': 'form-control'}),
        }
        labels = {
            'title': 'Название рецепта',
            'description': 'Описание',
            'cooking_time': 'Время приготовления (минуты)',
            'servings': 'Количество порций',
            'calories_per_100g': 'Калорийность на 100г',
            'difficulty': 'Сложность',
            'main_photo': 'Главное фото',
        }
    }

    def __init__(self, *args, **kwargs):
        # Извлекаем user из kwargs перед передачей в родительский класс
        self.user = kwargs.pop('user', None)
        super(RecipeForm, self).__init__(*args, **kwargs)

```

## Шаг 2: Упрощение представления RecipeCreateView

`recipes/views.py` (упростите `RecipeCreateView`):

`python`

```

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe

```

```

form_class = RecipeForm
template_name = 'recipes/recipe_form.html'

def get_form_kwargs(self):
    """Добавляем user в kwargs формы"""
    kwargs = super().get_form_kwargs()
    kwargs['user'] = self.request.user
    return kwargs

def form_valid(self, form):
    """Устанавливаем автора рецепта"""
    form.instance.author = self.request.user
    return super().form_valid(form)

```

## Шаг 3: Альтернативное решение - убрать передачу user

Если не нужно передавать user в форму, просто уберите эту логику:

`recipes/views.py` (альтернативная версия без user в форме):

python

```

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def form_valid(self, form):
        """Устанавливаем автора рецепта"""
        form.instance.author = self.request.user
        return super().form_valid(form)

```

И верните `recipes/forms.py` к простой версии:

python

```

from django import forms
from .models import Recipe

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {

```

```

        'description': forms.Textarea(attrs={'rows': 4, 'class': 'form-control'},
        'placeholder': 'Опишите ваш рецепт...'}),
        'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название рецепта'}),
        'cooking_time': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Время в минутах'}),
        'servings': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Количество порций'}),
        'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Калории на 100 грамм'}),
        'difficulty': forms.Select(attrs={'class': 'form-control'}),
        'main_photo': forms.FileInput(attrs={'class': 'form-control'}),
    }

```

## Шаг 4: Обновление шаблона формы

`templates/recipes/recipe_form.html` (уберите логику с полем author):

html

```

{% extends 'base.html' %}

{% block title %}
    {% if form.instance.pk %}Редактирование рецепта{% else %}Добавление рецепта{% endif %}
{% endblock %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card">
            <div class="card-header bg-success text-white">
                <h4 class="mb-0">
                    {% if form.instance.pk %}
                        Редактирование рецепта: {{ form.instance.title }}
                    {% else %}
                        Добавление нового рецепта
                    {% endif %}
                </h4>
            </div>
            <div class="card-body">
                <form method="post" enctype="multipart/form-data">
                    {% csrf_token %}

                    {% for field in form %}
                        <div class="mb-3">

```

```

        <label for="{{ field.id_for_label }}" class="form-label">
            {{ field.label }}
            {% if field.field.required %}<span class="text-dange
r"></span>{% endif %}
        </label>
        {{ field }}

        {% if field.help_text %}
        <div class="form-text">{{ field.help_text }}</div>
        {% endif %}

        {% if field.errors %}
        <div class="alert alert-danger mt-1">
            {{ field.errors }}
        </div>
        {% endif %}
    </div>
    {% endfor %}

    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
        <a href="{% url 'recipes:home' %}" class="btn btn-seconda
ry me-md-2">Отмена</a>
        <button type="submit" class="btn btn-success">
            {% if form.instance.pk %}Сохранить изменения{% else
%}Добавить рецепт{% endif %}
        </button>
    </div>
</form>
</div>
</div>
</div>
<% endblock %>
```

## Шаг 5: Проверка модели Recipe

Убедитесь, что в `recipes/models.py` поле `author` правильно настроено:

```

python

from django.db import models
from django.contrib.auth.models import User

class Recipe(models.Model):
    DIFFICULTY_LEVELS = [
        ('easy', 'Легкий'),
        ('medium', 'Средний'),
        ('hard', 'Сложный')
    ]
    title = models.CharField(max_length=100)
    description = models.TextField()
    ingredients = models.TextField()
    steps = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    difficulty_level = models.CharField(choices=DIFFICULTY_LEVELS, max_length=10)
    created_at = models.DateTimeField(auto_now_add=True)
```

```

        ('medium', 'Средний'),
        ('hard', 'Сложный'),
    ]

    title = models.CharField(max_length=200)
    description = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE) # ← это поле
    cooking_time = models.PositiveIntegerField(help_text="Время приготовления в м
инутах")
    servings = models.PositiveIntegerField(help_text="Количество порций")
    calories_per_100g = models.PositiveIntegerField(help_text="Калории на 100 гра
мм")
    difficulty = models.CharField(max_length=10, choices=DIFFICULTY_LEVELS)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    main_photo = models.ImageField(upload_to='recipes/main_photos/', blank=True,
null=True)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        from django.urls import reverse
        return reverse('recipes:recipe-detail', kwargs={'pk': self.pk})

```

## Шаг 6: Самый простой рабочий вариант

Если все еще есть проблемы, используйте максимально простую версию:

`recipes/views.py`:

`python`

```

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_1
00g', 'difficulty', 'main_photo']
    template_name = 'recipes/recipe_form.html'

    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

```

## Шаг 7: Проверка работы

После исправлений:

1. Остановите сервер (Ctrl+C)

2. Запустите снова:

```
bash
```

```
python manage.py runserver
```

3. Проверьте <http://127.0.0.1:8000/recipe/new/>

## Рекомендую попробовать в таком порядке:

1. Сначала Шаг 3 (убрать передачу user) - самый простой вариант

2. Если нужно сохранить логику с user - используйте Шаг 1 и Шаг 2

3. Для максимальной простоты - используйте Шаг 6

Попробуйте сначала Шаг 3 - он должен решить проблему быстро!

Так, починила. Но мне необходимо, дабы пользователь давал название, краткое описание, время готовки, количество порций, калории, сложность, основное фото, а также расписывал шаги готовки и по желанию добавлял к ним фото. Модель Recipe прописана следующим образом class Recipe(models.Model):

```
#Поле "сложность" со сразу готовыми вариантами
DIFFICULTY_LEVELS = [
    ('easy', 'Легкий'),
    ('medium', 'Средний'),
    ('hard', 'Сложный'),
]

#Поля модели
title = models.CharField(max_length=200)
description = models.TextField() # Краткое описание для ленты
author = models.ForeignKey(User, on_delete=models.CASCADE)
cooking_time = models.PositiveIntegerField(help_text="Время
приготовления в минутах")
servings = models.PositiveIntegerField(help_text="Количество
порций")
calories_per_100g =
models.PositiveIntegerField(help_text="Калории на 100 грамм")
difficulty = models.CharField(max_length=10,
choices=DIFFICULTY_LEVELS)
```

```

created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)
main_photo =
models.ImageField(upload_to='recipes/main_photos/')
video = models.FileField(upload_to='recipes/videos/', null=True,
blank=True)
hashtags = models.ManyToManyField(Hashtag, blank=True)

def __str__(self):
    return self.title

#Возврат канонического URL для избегания жёсткого кодирования
путей
def get_absolute_url(self):
    return reverse('recipe-detail', kwargs={'pk': self.pk})

Класс class CookingStep(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='cooking_steps',
on_delete=models.CASCADE)
    step_number = models.PositiveIntegerField()
    description = models.TextField()
    photo = models.ImageField(upload_to='recipes/steps_photos/',
blank=True, null=True)

    #Сортировка по номеру шага (по возрастанию)
    class Meta:
        ordering = ['step_number']

    def __str__(self):
        return f"Step {self.step_number} for {self.recipe.title}"

И также пользователь должен написать ингредиенты
class Ingredient(models.Model):
    recipe = models.ForeignKey(Recipe, related_name='ingredients',
on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    quantity = models.CharField(max_length=50) # Например: "2 шт",
"100 г", "по вкусу"

    def __str__(self):
        return f"{self.name} - {self.quantity}"

```

Понял! Нужно создать форму, которая позволяет добавлять рецепт с ингредиентами и шагами приготовления. Для этого используем формы с наборами (formsets). Вот полное решение:

## Шаг 1: Создание форм для ингредиентов и шагов

recipes/forms.py:

```
python

from django import forms
from django.forms import inlineformset_factory
from .models import Recipe, Ingredient, CookingStep

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']

        widgets = {
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control'}, placeholder='Краткое описание рецепта...'),
            'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название рецепта'}),
            'cooking_time': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Время в минутах'}),
            'servings': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Количество порций'}),
            'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Калории на 100 грамм'}),
            'difficulty': forms.Select(attrs={'class': 'form-control'}),
            'main_photo': forms.FileInput(attrs={'class': 'form-control'}),
        }

        labels = {
            'title': 'Название рецепта',
            'description': 'Краткое описание',
            'cooking_time': 'Время приготовления (минуты)',
            'servings': 'Количество порций',
            'calories_per_100g': 'Калорийность на 100г',
            'difficulty': 'Сложность',
            'main_photo': 'Главное фото',
        }

class IngredientForm(forms.ModelForm):
    class Meta:
        model = Ingredient
        fields = ['name', 'quantity']

        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название ингредиента'}),
            'quantity': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Количество'})
        }
```

```

        der': 'Количество, например: 2 шт, 100 г'}),
    }

class CookingStepForm(forms.ModelForm):
    class Meta:
        model = CookingStep
        fields = ['step_number', 'description', 'photo']
        widgets = {
            'step_number': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Номер шага'}),
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control', 'placeholder': 'Описание шага...'}),
            'photo': forms.FileInput(attrs={'class': 'form-control'}),
        }

    # Создаем formsets для ингредиентов и шагов
    IngredientFormSet = inlineformset_factory(
        Recipe,
        Ingredient,
        form=IngredientForm,
        extra=3, # Количество пустых форм для добавления
        can_delete=True
    )

    CookingStepFormSet = inlineformset_factory(
        Recipe,
        CookingStep,
        form=CookingStepForm,
        extra=2, # Количество пустых форм для добавления
        can_delete=True
    )

```

## Шаг 2: Обновление представления для создания рецепта

recipes/views.py:

```

python

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag, Ingredient, CookingStep
from .forms import RecipeForm, IngredientFormSet, CookingStepFormSet

```

```

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get_context_data(self, **kwargs):
        """Добавляем formsets в контекст"""
        context = super().get_context_data(**kwargs)
        if self.request.POST:
            context['ingredient_formset'] = IngredientFormSet(self.request.POST,
self.request.FILES)
            context['cooking_step_formset'] = CookingStepFormSet(self.request.PO
T, self.request.FILES)
        else:
            context['ingredient_formset'] = IngredientFormSet()
            context['cooking_step_formset'] = CookingStepFormSet()
        return context

    def form_valid(self, form):
        """Сохраняем рецепт и связанные объекты"""
        form.instance.author = self.request.user
        context = self.get_context_data()
        ingredient_formset = context['ingredient_formset']
        cooking_step_formset = context['cooking_step_formset']

        # Сохраняем основной рецепт
        self.object = form.save()

        # Сохраняем ингредиенты
        if ingredient_formset.is_valid():
            ingredient_formset.instance = self.object
            ingredient_formset.save()

        # Сохраняем шаги приготовления
        if cooking_step_formset.is_valid():
            cooking_step_formset.instance = self.object
            cooking_step_formset.save()

        return redirect('recipes:recipe-detail', pk=self.object.pk)

    def form_invalid(self, form):
        """Обработка невалидной формы"""
        return self.render_to_response(self.get_context_data(form=form))

```

## Шаг 3: Создание расширенного шаблона формы

**templates/recipes/recipe\_form.html:**

html

```

<div class="card mb-3 ingredient-form">
    <div class="card-body">
        <div class="row">
            <div class="col-md-6">
                <label class="form-label">Название ингредиента</label>
                {{ form.name }}
            </div>
            <div class="col-md-5">
                <label class="form-label">Количество</label>
                {{ form.quantity }}
            </div>
            <div class="col-md-1">
                <label class="form-label">&ampnbsp</label>
                {% if form.instance.pk %}
                    {{ form.DELETE }} <label for="{{ form.DELETE.id_for_label }}">Удалить</label>
                {% endif %}
            </div>
            </div>
            {{ form.id }}
        </div>
    </div>
    {% endfor %}
</div>
<button type="button" class="btn btn-outline-success btn-sm mb-4" id="add-ingredient">
    + Добавить еще ингредиент
</button>

<hr class="my-4">

<h5 class="mb-3">Шаги приготовления</h5>
<div id="cooking-step-formset">
    {{ cooking_step_formset.management_form }}
    {% for form in cooking_step_formset %}
        <div class="card mb-3 cooking-step-form">
            <div class="card-body">
                <div class="row">
                    <div class="col-md-2">
                        <label class="form-label">Шаг %</label>
                        {{ form.step_number }}
                    </div>
                    <div class="col-md-7">
                        <label class="form-label">Описание</label>
                    </div>
                </div>
            </div>
        </div>
    {% endfor %}
</div>

```

```

        {{ form.description }}
    </div>
    <div class="col-md-3">
        <label class="form-label">Фото шага (необязательно)</label>
            {{ form.photo }}
        </div>
    </div>
    <div class="row mt-2">
        <div class="col-md-12">
            {% if form.instance.pk %}
                {{ form.DELETE }} <label for="{{ form.DELETE.id_for_label }}">Удалить этот шаг</label>
                {% endif %}
            </div>
        </div>
        {{ form.id }}
    </div>
    </div>
    {% endfor %}
</div>
<button type="button" class="btn btn-outline-success btn-sm mb-4" id="add-cooking-step">
    + Добавить еще шаг
</button>

<div class="d-grid gap-2 d-md-flex justify-content-md-end mt-4">
    <a href="{% url 'recipes:home' %}" class="btn btn-secondary me-md-2">Отмена</a>
    <button type="submit" class="btn btn-success">Добавить рецепт</button>
</div>
</div>
</div>
</div>
</div>

<script>
// JavaScript для динамического добавления форм
document.addEventListener('DOMContentLoaded', function() {
    // Добавление ингредиентов
    document.getElementById('add-ingredient').addEventListener('click', function() {
        const formset = document.getElementById('ingredient-formset');
        const totalForms = document.getElementById('id_ingredients-TOTAL_FORMS');

```

```
const formNum = parseInt(totalForms.value);

const newForm = formset.querySelector('.ingredient-form').cloneNode(true);
newForm.innerHTML = newForm.innerHTML.replace(/ingredients-\d+-/g, `ingredients-${
formNum}-`);
newForm.innerHTML = newForm.innerHTML.replace(/\[\d+\]/g, `[${formNum}]`);

// Очищаем значения в новой форме
const inputs = newForm.querySelectorAll('input');
inputs.forEach(input => {
    if (input.type !== 'hidden') {
        input.value = '';
    }
});

formset.appendChild(newForm);
totalForms.value = formNum + 1;
});

// Добавление шагов приготовления
document.getElementById('add-cooking-step').addEventListener('click', function() {
    const formset = document.getElementById('cooking-step-formset');
    const totalForms = document.getElementById('id_cooking_steps-TOTAL_FORMS');
    const formNum = parseInt(totalForms.value);

    const newForm = formset.querySelector('.cooking-step-form').cloneNode(true);
    newForm.innerHTML = newForm.innerHTML.replace(/cooking_steps-\d+-/g, `cooking_steps-${
formNum}-`);
    newForm.innerHTML = newForm.innerHTML.replace(/\[\d+\]/g, `[${formNum}]`);

    // Очищаем значения в новой форме
    const inputs = newForm.querySelectorAll('input, textarea');
    inputs.forEach(input => {
        if (input.type !== 'hidden' && input.type !== 'checkbox') {
            input.value = '';
        }
    });

    formset.appendChild(newForm);
    totalForms.value = formNum + 1;
});
});
```

```
});
</script>

<style>
.ingredient-form, .cooking-step-form {
    border-left: 4px solid #28a745;
}
</style>
{% endblock %}
```

## Шаг 4: Обновление представления для редактирования рецепта

`recipes/views.py` (добавьте класс для редактирования):

`python`

```
class RecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if self.request.POST:
            context['ingredient_formset'] = IngredientFormSet(self.request.POST,
self.request.FILES, instance=self.object)
            context['cooking_step_formset'] = CookingStepFormSet(self.request.PO
T, self.request.FILES, instance=self.object)
        else:
            context['ingredient_formset'] = IngredientFormSet(instance=self.o
bjec
t)
            context['cooking_step_formset'] = CookingStepFormSet(instance=self.o
ject)
        return context

    def form_valid(self, form):
        context = self.get_context_data()
        ingredient_formset = context['ingredient_formset']
        cooking_step_formset = context['cooking_step_formset']

        if ingredient_formset.is_valid() and cooking_step_formset.is_valid():
            self.object = form.save()
            ingredient_formset.instance = self.object
            ingredient_formset.save()
```

```

        cooking_step_formset.instance = self.object
        cooking_step_formset.save()
        return redirect('recipes:recipe-detail', pk=self.object.pk)
    else:
        return self.render_to_response(self.get_context_data(form=form))

def dispatch(self, request, *args, **kwargs):
    obj = self.get_object()
    if obj.author != self.request.user:
        return redirect('recipes:recipe-detail', pk=obj.pk)
    return super().dispatch(request, *args, **kwargs)

```

## Шаг 5: Обновление шаблона детальной страницы

`templates/recipes/recipe_detail.html` (добавьте отображение ингредиентов и шагов):

html

```

<!-- ... остальной код ... -->

<h5>Ингредиенты</h5>
<ul class="list-group mb-4">
    {% for ingredient in recipe.ingredients.all %}
        <li class="list-group-item d-flex justify-content-between align-items-cente
r">
            {{ ingredient.name }}
            <span class="badge bg-success rounded-pill">{{ ingredient.quantity }}</sp
an>
        </li>
    {% endfor %}
</ul>

<h5>Шаги приготовления</h5>
<div class="mb-4">
    {% for step in recipe.cooking_steps.all %}
        <div class="card mb-3">
            <div class="card-body">
                <div class="row">
                    <div class="col-md-1">
                        <span class="badge bg-success fs-6">Шаг {{ step.step_number
}}</span>
                    </div>
                    <div class="col-md-{{% if step.photo %}7{% else %}11{% endif %}">
                        <p class="mb-0">{{ step.description }}</p>
                    </div>
                {% if step.photo %}

```

```

        <div class="col-md-4">
            
        </div>
    {% endif %}
</div>
</div>
{% endfor %}

</div>

<!-- ... остальной код ... -->

```

Теперь пользователь может добавлять рецепты с ингредиентами и шагами приготовления! Форма поддерживает динамическое добавление полей и загрузку фото для шагов.

Unresolved reference 'IngredientFormSet'

Ошибка означает, что `IngredientFormSet` не определен. Давайте исправим это, правильно создав formsets:

## Шаг 1: Исправление forms.py

`recipes/forms.py` (полная версия с правильным определением formsets):

```

python

from django import forms
from django.forms import inlineformset_factory
from .models import Recipe, Ingredient, CookingStep

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control',
            'placeholder': 'Краткое описание рецепта...'}),
            'title': forms.TextInput(attrs={'class': 'form-control',
            'placeholder': 'Название рецепта'}),
            'cooking_time': forms.NumberInput(attrs={'class': 'form-control',
            'placeholder': 'Время в минутах'}),
            'servings': forms.NumberInput(attrs={'class': 'form-control',
            'placeholder': 'Количество порций'})
        }

```

```
older': 'Количество порций'}),
    'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control',
    'placeholder': 'Калории на 100 грамм'}),
    'difficulty': forms.Select(attrs={'class': 'form-control'}),
    'main_photo': forms.FileInput(attrs={'class': 'form-control'}),
}

class IngredientForm(forms.ModelForm):
    class Meta:
        model = Ingredient
        fields = ['name', 'quantity']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название ингредиента'}),
            'quantity': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Количество, например: 2 шт, 100 г'}),
        }

class CookingStepForm(forms.ModelForm):
    class Meta:
        model = CookingStep
        fields = ['step_number', 'description', 'photo']
        widgets = {
            'step_number': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Номер шага'}),
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control', 'placeholder': 'Описание шага...'}),
            'photo': forms.FileInput(attrs={'class': 'form-control'}),
        }

# Создаем formsets
IngredientFormSet = inlineformset_factory(
    Recipe,
    Ingredient,
    form=IngredientForm,
    extra=3,
    can_delete=True,
    fields=['name', 'quantity']
)

CookingStepFormSet = inlineformset_factory(
    Recipe,
    CookingStep,
    form=CookingStepForm,
    extra=2,
    can_delete=True,
```

```
        fields=['step_number', 'description', 'photo']
    )
```

## Шаг 2: Исправление импортов в views.py

recipes/views.py (правильные импорты):

```
python

from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from .models import Recipe, Favorite, Hashtag, Ingredient, CookingStep
from .forms import RecipeForm, IngredientForm, CookingStepForm

# Импортируем inlineformset_factory и создаем formsets прямо в views
from django.forms import inlineformset_factory

# Создаем formsets здесь, чтобы избежать циклических импортов
IngredientFormSet = inlineformset_factory(
    Recipe,
    Ingredient,
    form=IngredientForm,
    extra=3,
    can_delete=True,
    fields=['name', 'quantity']
)

CookingStepFormSet = inlineformset_factory(
    Recipe,
    CookingStep,
    form=CookingStepForm,
    extra=2,
    can_delete=True,
    fields=['step_number', 'description', 'photo']
)

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get_context_data(self, **kwargs):
```

```
"""Добавляем formsets в контекст"""
context = super().get_context_data(**kwargs)
if self.request.POST:
    context['ingredient_formset'] = IngredientFormSet(self.request.POST,
self.request.FILES, prefix='ingredients')
    context['cooking_step_formset'] = CookingStepFormSet(self.request.PO
T, self.request.FILES, prefix='cooking_steps')
else:
    context['ingredient_formset'] = IngredientFormSet(prefix='ingredien
ts')
    context['cooking_step_formset'] = CookingStepFormSet(prefix='cooking_
steps')
return context

def form_valid(self, form):
    """Сохраняем рецепт и связанные объекты"""
    form.instance.author = self.request.user
    context = self.get_context_data()
    ingredient_formset = context['ingredient_formset']
    cooking_step_formset = context['cooking_step_formset']

    # Сохраняем основной рецепт
    self.object = form.save()

    # Сохраняем ингредиенты
    if ingredient_formset.is_valid():
        ingredient_formset.instance = self.object
        ingredient_formset.save()
    else:
        print("Ingredient formset errors:", ingredient_formset.errors)

    # Сохраняем шаги приготовления
    if cooking_step_formset.is_valid():
        cooking_step_formset.instance = self.object
        cooking_step_formset.save()
    else:
        print("Cooking step formset errors:", cooking_step_formset.errors)

    return redirect('recipes:recipe-detail', pk=self.object.pk)

def form_invalid(self, form):
    """Обработка невалидной формы"""
    return self.render_to_response(self.get_context_data(form=form))

class RecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    form_class = RecipeForm
```

```

template_name = 'recipes/recipe_form.html'

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    if self.request.POST:
        context['ingredient_formset'] = IngredientFormSet(self.request.POST,
self.request.FILES, instance=self.object, prefix='ingredients')
        context['cooking_step_formset'] = CookingStepFormSet(self.request.PO
T, self.request.FILES, instance=self.object, prefix='cooking_steps')
    else:
        context['ingredient_formset'] = IngredientFormSet(instance=self.obj
t, prefix='ingredients')
        context['cooking_step_formset'] = CookingStepFormSet(instance=self.o
bject, prefix='cooking_steps')
    return context

def form_valid(self, form):
    context = self.get_context_data()
    ingredient_formset = context['ingredient_formset']
    cooking_step_formset = context['cooking_step_formset']

    if ingredient_formset.is_valid() and cooking_step_formset.is_valid():
        self.object = form.save()
        ingredient_formset.instance = self.object
        ingredient_formset.save()
        cooking_step_formset.instance = self.object
        cooking_step_formset.save()
        return redirect('recipes:recipe-detail', pk=self.object.pk)
    else:
        return self.render_to_response(self.get_context_data(form=form))

def dispatch(self, request, *args, **kwargs):
    obj = self.get_object()
    if obj.author != self.request.user:
        return redirect('recipes:recipe-detail', pk=obj.pk)
    return super().dispatch(request, *args, **kwargs)

# ... остальные представления (ListView, DetailView и т.д.)

```

## Шаг 3: Альтернативное решение - упрощенная версия

Если все еще есть проблемы, используйте упрощенную версию:

`recipes/views.py` (упрощенная версия):

```
python
```

```
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from django.db.models import Q
from django.forms import inlineformset_factory
from .models import Recipe, Favorite, Hashtag, Ingredient, CookingStep
from .forms import RecipeForm, IngredientForm, CookingStepForm

class RecipeCreateView(LoginRequiredMixin, CreateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get(self, request, *args, **kwargs):
        # Создаем formsets для GET запроса
        IngredientFormSet = inlineformset_factory(Recipe, Ingredient, form=IngredientForm, extra=3, can_delete=True)
        CookingStepFormSet = inlineformset_factory(Recipe, CookingStep, form=CookingStepForm, extra=2, can_delete=True)

        self.object = None
        form = self.get_form()
        ingredient_formset = IngredientFormSet()
        cooking_step_formset = CookingStepFormSet()
        return self.render_to_response(
            self.get_context_data(
                form=form,
                ingredient_formset=ingredient_formset,
                cooking_step_formset=cooking_step_formset
            )
        )

    def post(self, request, *args, **kwargs):
        # Создаем formsets для POST запроса
        IngredientFormSet = inlineformset_factory(Recipe, Ingredient, form=IngredientForm, extra=0, can_delete=True)
        CookingStepFormSet = inlineformset_factory(Recipe, CookingStep, form=CookingStepForm, extra=0, can_delete=True)

        form = self.get_form()
        ingredient_formset = IngredientFormSet(request.POST, request.FILES)
        cooking_step_formset = CookingStepFormSet(request.POST, request.FILES)
```

```

        if form.is_valid() and ingredient_formset.is_valid() and cooking_step_formset.is_valid():
            return self.form_valid(form, ingredient_formset, cooking_step_formset)
        else:
            return self.form_invalid(form, ingredient_formset, cooking_step_formset)

    def form_valid(self, form, ingredient_formset, cooking_step_formset):
        form.instance.author = self.request.user
        self.object = form.save()
        ingredient_formset.instance = self.object
        ingredient_formset.save()
        cooking_step_formset.instance = self.object
        cooking_step_formset.save()
        return redirect('recipes:recipe-detail', pk=self.object.pk)

    def form_invalid(self, form, ingredient_formset, cooking_step_formset):
        return self.render_to_response(
            self.get_context_data(
                form=form,
                ingredient_formset=ingredient_formset,
                cooking_step_formset=cooking_step_formset
            )
        )

```

## Шаг 4: Обновление forms.py (только базовые формы)

`recipes/forms.py` (только формы без formsets):

```

python

from django import forms
from .models import Recipe, Ingredient, CookingStep

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control', 'placeholder': 'Краткое описание рецепта...'}),
            'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название рецепта'}),
            'cooking_time': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Время приготовления'})
        }

```

```

        'placeholder': 'Время в минутах'}),
        'servings': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Количество порций'}),
        'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Калории на 100 грамм'}),
        'difficulty': forms.Select(attrs={'class': 'form-control'}),
        'main_photo': forms.FileInput(attrs={'class': 'form-control'}),
    }

class IngredientForm(forms.ModelForm):
    class Meta:
        model = Ingredient
        fields = ['name', 'quantity']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название ингредиента'}),
            'quantity': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Количество, например: 2 шт, 100 г'}),
        }

class CookingStepForm(forms.ModelForm):
    class Meta:
        model = CookingStep
        fields = ['step_number', 'description', 'photo']
        widgets = {
            'step_number': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Номер шага'}),
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control', 'placeholder': 'Описание шага...'}),
            'photo': forms.FileInput(attrs={'class': 'form-control'}),
        }

```

## Рекомендую попробовать:

1. Сначала Шаг 2 - создание formsets прямо в views.py
2. Если не работает - перейдите к Шагу 3 (упрощенная версия)

**Попробуйте Шаг 2 - он самый надежный, так как избегает циклических импортов.**

Предусмотри возможность удаления ингредиентов и шагов

Отлично! Добавим возможность удаления ингредиентов и шагов. Вот полное решение с улучшенным интерфейсом:

## Шаг 1: Обновление forms.py

recipes/forms.py:

python

```
from django import forms
from .models import Recipe, Ingredient, CookingStep

class RecipeForm(forms.ModelForm):
    class Meta:
        model = Recipe
        fields = ['title', 'description', 'cooking_time', 'servings', 'calories_per_100g', 'difficulty', 'main_photo']
        widgets = {
            'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control'}, placeholder='Краткое описание рецепта...'),
            'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название рецепта'}),
            'cooking_time': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Время в минутах'}),
            'servings': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Количество порций'}),
            'calories_per_100g': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Калории на 100 грамм'}),
            'difficulty': forms.Select(attrs={'class': 'form-control'}),
            'main_photo': forms.FileInput(attrs={'class': 'form-control'}),
        }

class IngredientForm(forms.ModelForm):
    class Meta:
        model = Ingredient
        fields = ['name', 'quantity']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Название ингредиента'}),
            'quantity': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Количество, например: 2 шт, 100 г'}),
        }

class CookingStepForm(forms.ModelForm):
    class Meta:
        model = CookingStep
        fields = ['step_number', 'description', 'photo']
        widgets = {
            'step_number': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': 'Номер шага'}),
        }
```

```
'description': forms.Textarea(attrs={'rows': 3, 'class': 'form-control'},  
                                'placeholder': 'Описание шага...'}),  
                'photo': forms.FileInput(attrs={'class': 'form-control'}),  
            }  
        }  
    }  
}
```

## Шаг 2: Обновление views.py с поддержкой удаления

recipes/views.py:

python

```
from django.shortcuts import render, get_object_or_404, redirect  
from django.contrib.auth.decorators import login_required  
from django.contrib.auth.mixins import LoginRequiredMixin  
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView  
from django.db.models import Q  
from django.forms import inlineformset_factory  
from .models import Recipe, Favorite, Hashtag, Ingredient, CookingStep  
from .forms import RecipeForm, IngredientForm, CookingStepForm  
  
class RecipeCreateView(LoginRequiredMixin, CreateView):  
    model = Recipe  
    form_class = RecipeForm  
    template_name = 'recipes/recipe_form.html'  
  
    def get_context_data(self, **kwargs):  
        context = super().get_context_data(**kwargs)  
  
        # Создаем formsets с поддержкой удаления  
        IngredientFormSet = inlineformset_factory(  
            Recipe, Ingredient, form=IngredientForm,  
            extra=3, can_delete=True, can_delete_extra=True  
        )  
        CookingStepFormSet = inlineformset_factory(  
            Recipe, CookingStep, form=CookingStepForm,  
            extra=2, can_delete=True, can_delete_extra=True  
        )  
  
        if self.request.POST:  
            context['ingredient_formset'] = IngredientFormSet(  
                self.request.POST, self.request.FILES,  
                prefix='ingredients'  
            )  
            context['cooking_step_formset'] = CookingStepFormSet(  
                self.request.POST, self.request.FILES,
```

```
        prefix='cooking_steps'
    )
else:
    context['ingredient_formset'] = IngredientFormSet(prefix='ingredient')
    context['cooking_step_formset'] = CookingStepFormSet(prefix='cooking_steps')

return context

def form_valid(self, form):
    form.instance.author = self.request.user
    context = self.get_context_data()
    ingredient_formset = context['ingredient_formset']
    cooking_step_formset = context['cooking_step_formset']

    # Сохраняем основной рецепт
    self.object = form.save()

    # Сохраняем ингредиенты (удаленные формы будут автоматически удалены)
    if ingredient_formset.is_valid():
        ingredient_formset.instance = self.object
        ingredient_formset.save()

    # Сохраняем шаги приготовления (удаленные формы будут автоматически удалены)
    if cooking_step_formset.is_valid():
        cooking_step_formset.instance = self.object
        cooking_step_formset.save()

    return redirect('recipes:recipe-detail', pk=self.object.pk)

class RecipeUpdateView(LoginRequiredMixin, UpdateView):
    model = Recipe
    form_class = RecipeForm
    template_name = 'recipes/recipe_form.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        # Создаем formsets с поддержкой удаления для существующего рецепта
        IngredientFormSet = inlineformset_factory(
            Recipe, Ingredient, form=IngredientForm,
            extra=2, can_delete=True, can_delete_extra=True
        )
        CookingStepFormSet = inlineformset_factory(
            Recipe, CookingStep, form=CookingStepForm,
```

```

        extra=1, can_delete=True, can_delete_extra=True
    )

    if self.request.POST:
        context['ingredient_formset'] = IngredientFormSet(
            self.request.POST, self.request.FILES,
            instance=self.object, prefix='ingredients'
        )
        context['cooking_step_formset'] = CookingStepFormSet(
            self.request.POST, self.request.FILES,
            instance=self.object, prefix='cooking_steps'
        )
    else:
        context['ingredient_formset'] = IngredientFormSet(
            instance=self.object, prefix='ingredients'
        )
        context['cooking_step_formset'] = CookingStepFormSet(
            instance=self.object, prefix='cooking_steps'
        )

    return context

def form_valid(self, form):
    context = self.get_context_data()
    ingredient_formset = context['ingredient_formset']
    cooking_step_formset = context['cooking_step_formset']

    if ingredient_formset.is_valid() and cooking_step_formset.is_valid():
        self.object = form.save()
        ingredient_formset.instance = self.object
        ingredient_formset.save()
        cooking_step_formset.instance = self.object
        cooking_step_formset.save()
        return redirect('recipes:recipe-detail', pk=self.object.pk)
    else:
        return self.render_to_response(self.get_context_data(form=form))

def dispatch(self, request, *args, **kwargs):
    obj = self.get_object()
    if obj.author != self.request.user:
        return redirect('recipes:recipe-detail', pk=obj.pk)
    return super().dispatch(request, *args, **kwargs)

```

## Шаг 3: Обновление шаблона с кнопками удаления

[templates/recipes/recipe\\_form.html](#):

```
html

{% extends 'base.html' %}

{% block title %}
    {% if form.instance.pk %}Редактирование рецепта{% else %}Добавление рецепта{%
endif %}
{% endblock %}

{% block content %}


#### {% if form.instance.pk %} Редактирование рецепта: {{ form.instance.title }} {% else %} Добавление нового рецепта {% endif %}



{% csrf_token %}



##### Основная информация


{% for field in form %}


{{ field.label }}
        {% if field.field.required %}<span class="text-dange
r">*</span>{% endif %}
    
{{ field }}

        {% if field.help_text %}


{{ field.help_text }}


{% endif %}

        {% if field.errors %}


{{ field.errors }}


{% endif %}


{% endfor %}


```

```

<hr class="my-4">

<h5 class="mb-3">Ингредиенты</h5>
<div id="ingredient-formset">
    {{ ingredient_formset.management_form }}
    {% for form in ingredient_formset %}
        <div class="card mb-3 ingredient-form" data-form-type="in
gredient">
            <div class="card-body">
                <div class="row align-items-center">
                    <div class="col-md-5">
                        <label class="form-label">Название ингред
иента</label>
                        {{ form.name }}
                        {% if form.name.errors %}
                            <div class="text-danger small">{{ form.name
.errors }}</div>
                        {% endif %}
                    </div>
                    <div class="col-md-5">
                        <label class="form-label">Количество</lab
el>
                        {{ form.quantity }}
                        {% if form.quantity.errors %}
                            <div class="text-danger small">{{ form.quantity
.errors }}</div>
                        {% endif %}
                    </div>
                    <div class="col-md-2">
                        <label class="form-label">&ampnbsp</label>
                        <div class="d-grid">
                            {% if form.instance.pk %}
                                # Для существующих ингредиентов
- чекбокс удаления #
                            <div class="form-check">
                                {{ form.DELETE }}
                                <label class="form-check-labe
l text-danger" for="{{ form.DELETE.id_for_label }}">
                                    Удалить
                                </label>
                            </div>
                            {% else %}
                                # Для новых ингредиентов - кнопк
а удаления #
                            <button type="button" class="bt
n btn-outline-danger btn-sm remove-form">

```

```
        <i class="fas fa-trash"></i>
Удалить
        </button>
    {% endif %}
    </div>
    </div>
    {{ form.id }}
    </div>
    </div>
    {% endfor %}
    </div>
    <button type="button" class="btn btn-outline-success btn-sm mb-4" id="add-ingredient">
        <i class="fas fa-plus"></i> Добавить ингредиент
    </button>

<hr class="my-4">

<h5 class="mb-3">Шаги приготовления</h5>
<div id="cooking-step-formset">
    {{ cooking_step_formset.management_form }}
    {% for form in cooking_step_formset %}
        <div class="card mb-3 cooking-step-form" data-form-type="step">
            <div class="card-body">
                <div class="row align-items-start">
                    <div class="col-md-2">
                        <label class="form-label">Шаг №</label>
                        {{ form.step_number }}
                        {% if form.step_number.errors %}
                            <div class="text-danger small">{{ form.step_number.errors }}</div>
                        {% endif %}
                    </div>
                    <div class="col-md-6">
                        <label class="form-label">Описание</label>
                    >
                        {{ form.description }}
                        {% if form.description.errors %}
                            <div class="text-danger small">{{ form.description.errors }}</div>
                        {% endif %}
                    </div>
                    <div class="col-md-3">
                        <label class="form-label">Фото шага (необязательно)</label>

```

```

        {{ form.photo }}
        {% if form.photo.errors %}
<div class="text-danger small">{{ form.photo.errors }}</div>
        {% endif %}
</div>
<div class="col-md-1">
    <label class="form-label">&nbsp;</label>
    <div class="d-grid">
        {% if form.instance.pk %}
            # Для существующих шагов – чекбокс удаления #
            <div class="form-check">
                {{ form.DELETE }}
                <label class="form-check-label text-danger" for="{{ form.DELETE.id_for_label }}>
                    Удалить
                </label>
            </div>
        {% else %}
            # Для новых шагов – кнопка удаления #
            <button type="button" class="btn btn-outline-danger btn-sm remove-form">
                <i class="fas fa-trash"></i>
            </button>
        {% endif %}
            </div>
        </div>
        {{ form.id }}
    </div>
</div>
    {% endfor %}
</div>
<button type="button" class="btn btn-outline-success btn-sm m-4" id="add-cooking-step">
    <i class="fas fa-plus"></i> Добавить шаг
</button>

<div class="d-grid gap-2 d-md-flex justify-content-md-end mt-4">
    <a href="{% if form.instance.pk %}{% url 'recipes:recipe-detail' form.instance.pk %}{% else %}{% url 'recipes:home' %}{% endif %}" class="btn btn-secondary me-md-2">
        Отмена
    </a>

```

```
        <button type="submit" class="btn btn-success">
            {% if form.instance.pk %}Сохранить изменения{% else
%}Добавить рецепт{% endif %}
        </button>
    </div>
</form>
</div>
</div>
</div>

<!-- Добавляем Font Awesome для иконок --&gt;
&lt;link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/
6.0.0/css/all.min.css"&gt;

&lt;script&gt;
document.addEventListener('DOMContentLoaded', function() {
    // Функция для обновления номеров шагов
    function updateStepNumbers() {
        const stepForms = document.querySelectorAll('.cooking-step-form:not(.dele
ted)');
        stepForms.forEach((form, index) =&gt; {
            const stepNumberInput = form.querySelector('input[name*="step_&nbsp;
nbsp;numbe
r"]');
            if (stepNumberInput) {
                stepNumberInput.value = index + 1;
            }
        });
    }

    // Удаление форм (для новых элементов)
    document.addEventListener('click', function(e) {
        if (e.target.classList.contains('remove-form') || e.target.closest('.remo
ve-form')) {
            const form = e.target.closest('.ingredient-form, .cooking-step-for
m');
            if (form) {
                form.classList.add('deleted');
                form.style.display = 'none';
            }
        }
    });
});</pre>
```





