

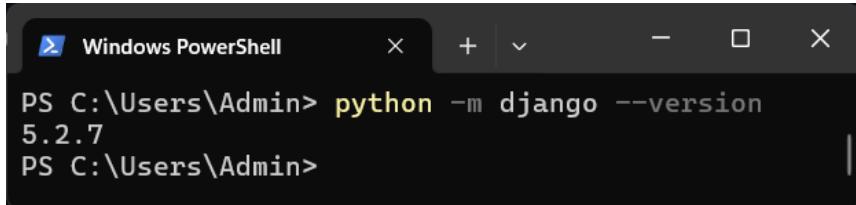
Введение в Django

Часть 1

Создание проекта и первого приложения (“polls”), написание базовой view, подключение URL, запуск dev-сервера.

1. Установка django

```
python -m pip install Django
```

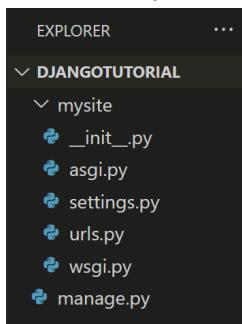


```
PS C:\Users\Admin> python -m django --version
5.2.7
PS C:\Users\Admin>
```

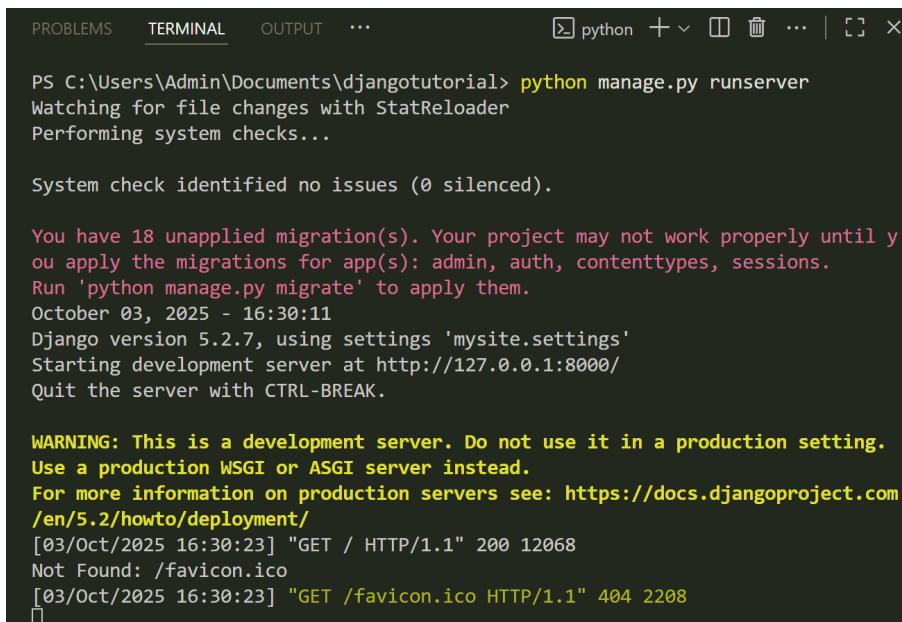
2. mkdir djangotutorial

```
3. django-admin startproject mysite djangotutorial
```

4. Получилось:



5. Сервер разработки



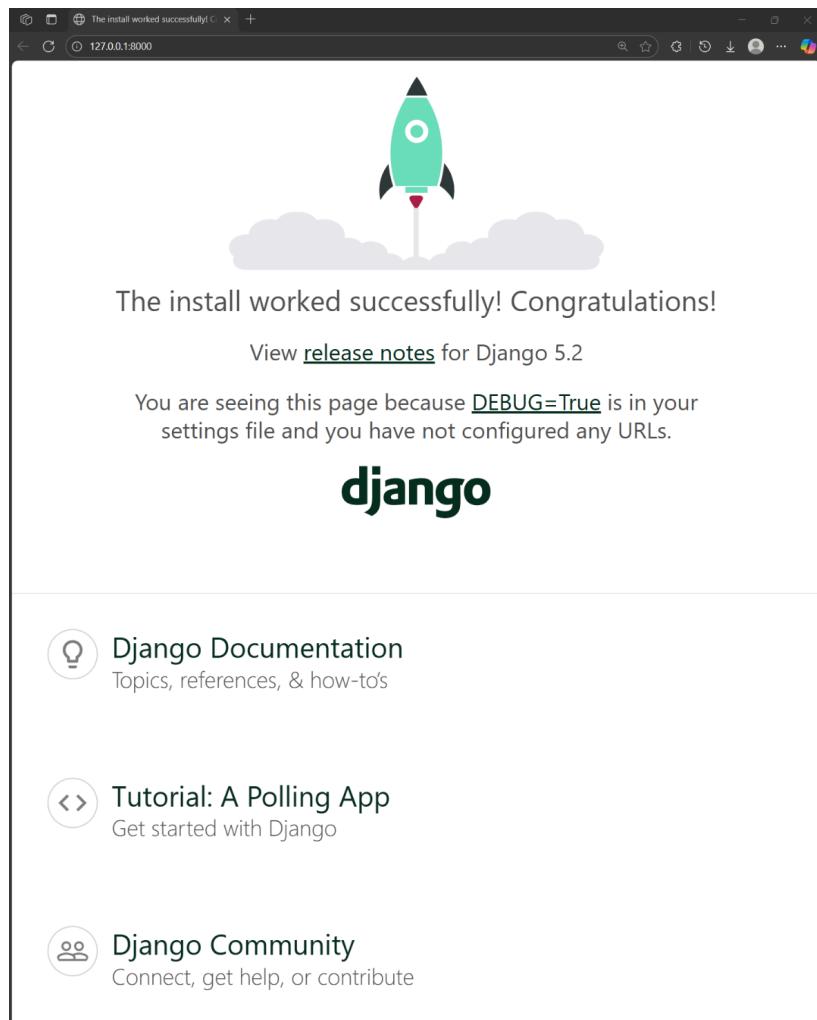
```
PROBLEMS TERMINAL OUTPUT ...
python + v ... | [] x
PS C:\Users\Admin\Documents\djangotutorial> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

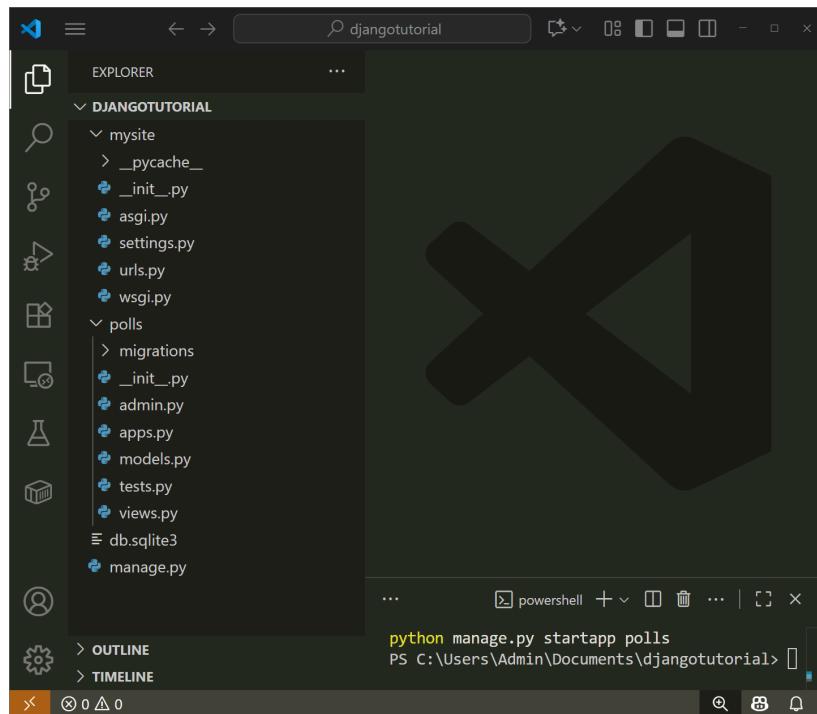
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 03, 2025 - 16:30:11
Django version 5.2.7, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting.
Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[03/Oct/2025 16:30:23] "GET / HTTP/1.1" 200 12068
Not Found: /favicon.ico
[03/Oct/2025 16:30:23] "GET /favicon.ico HTTP/1.1" 404 2208
```

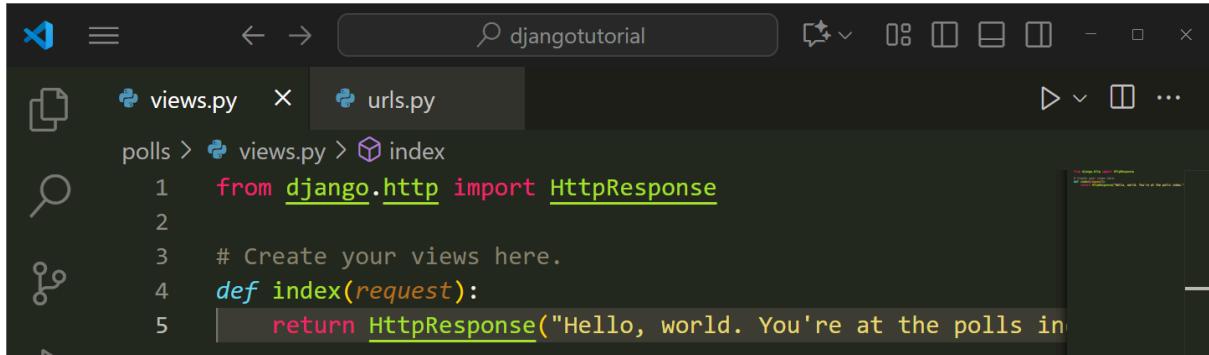
6. <http://127.0.0.1:8000/>



7. Создание приложения опросы



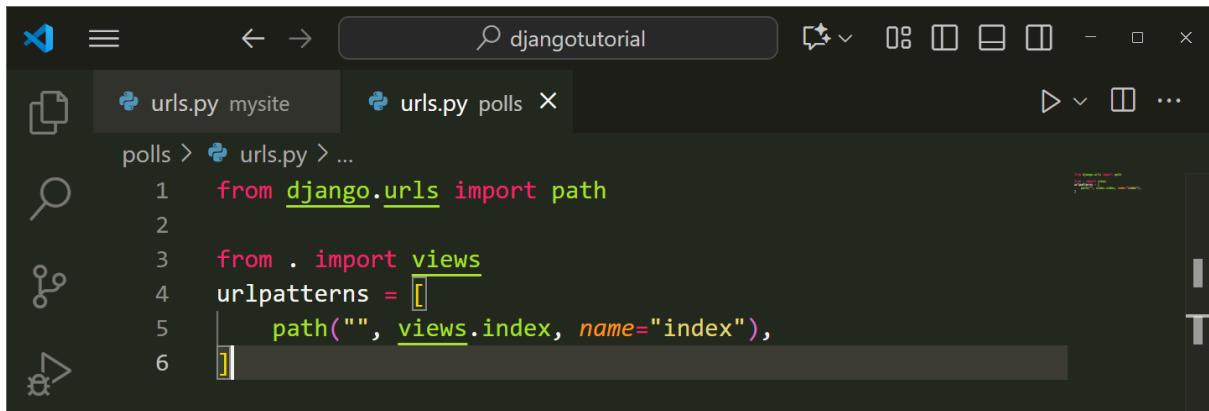
8. views.py



Screenshot of VS Code showing the contents of `views.py` in the `polls` application. The code defines a single view function `index` that returns a `HttpResponse` with the message "Hello, world. You're at the polls index".

```
from django.http import HttpResponse
# Create your views here.
def index(request):
    return HttpResponse("Hello, world. You're at the polls index")
```

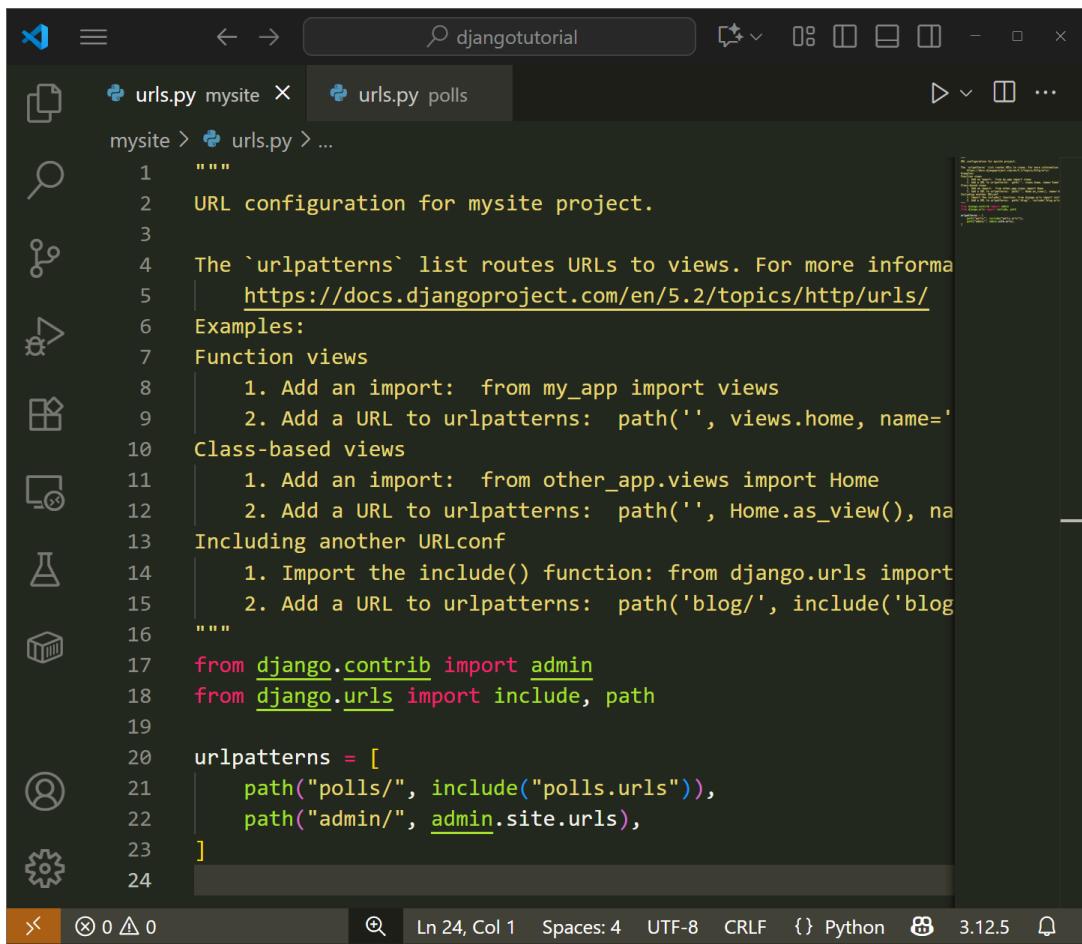
9. urls.py



Screenshot of VS Code showing the contents of `urls.py` in the `polls` application. It includes a single URL pattern that maps the root path to the `index` view defined in `views.py`.

```
from django.urls import path
from . import views
urlpatterns = [
    path("", views.index, name="index"),
```

10. urls.py



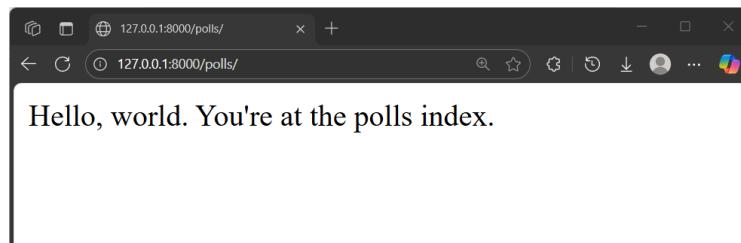
Screenshot of VS Code showing the contents of `urls.py` in the `mysite` project. This file serves as the main URL configuration for the project, containing examples for both function-based and class-based views, as well as instructions for including other URLconf files like `polls.urls` and `admin.site.urls`.

```
"""
URL configuration for mysite project.

The `urlpatterns` list routes URLs to views. For more information
see https://docs.djangoproject.com/en/5.2/topics/http/urls/
Examples:
Function views
1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path('', views.home, name='home')
Class-based views
1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
1. Import the include() function: from django.urls import include
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path("polls/", include("polls.urls")),
    path("admin/", admin.site.urls),
]
```

11. Результат python manage.py runserver



Часть 2

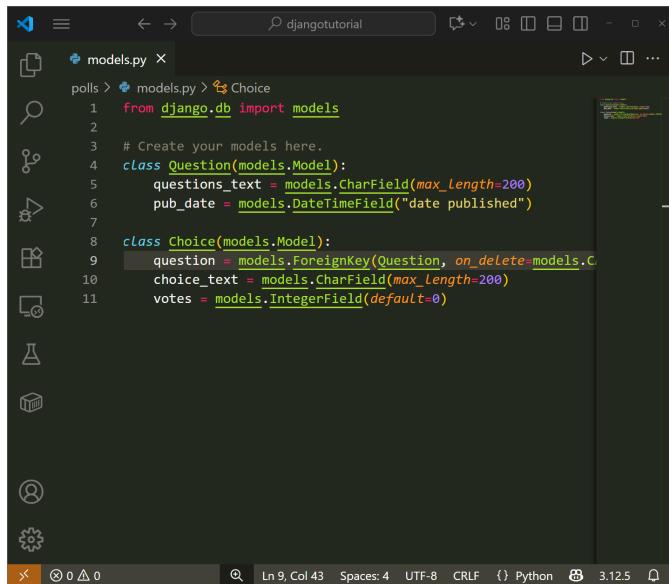
Работа с моделями и базой данных, использование API ORM, добавление приложения в админку

12. python manage.py migrate

```
PS C:\Users\Admin\Documents\djangotutorial> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
PS C:\Users\Admin\Documents\djangotutorial>
```

Создание моделей

13. Модель



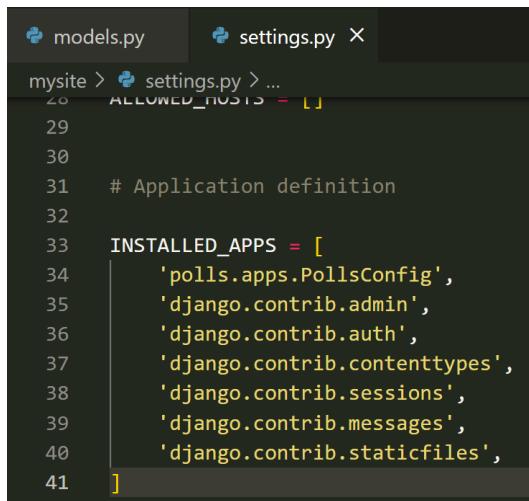
models.py

```
from django.db import models

# Create your models here.
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField("date published")

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

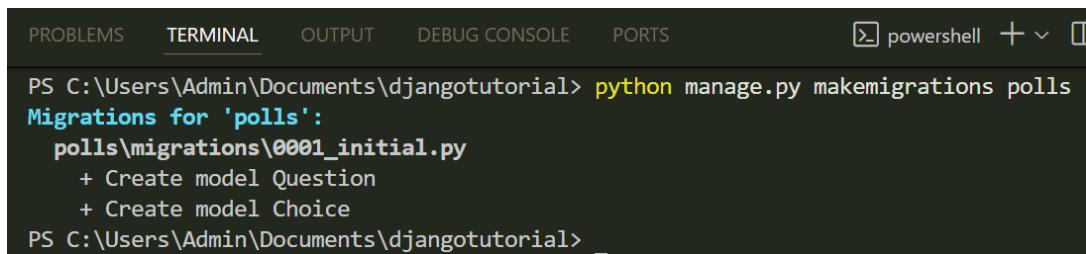
14. Добавление приложения в проект



settings.py

```
mysite > settings.py > ...
20     ALLOWED_HOSTS = []
21
22
23
24
25
26
27
28
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'polls.apps.PollsConfig',
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41 ]
```

15. Миграция базы данных



```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powershell +
```

```
PS C:\Users\Admin\Documents\djangotutorial> python manage.py makemigrations polls
Migrations for 'polls':
  polls\migrations\0001_initial.py
    + Create model Question
    + Create model Choice
PS C:\Users\Admin\Documents\djangotutorial>
```

16. sq|migrate

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powershell + × [ ] ×

● PS C:\Users\Admin\Documents\.djangotutorial> python manage.py sqlmigrate polls 0001
BEGIN;
--
-- Create model Question
--
CREATE TABLE "polls_question" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "questions_text" varchar(200) NOT NULL, "pub_date" datetime NOT NULL);
--
-- Create model Choice
--
CREATE TABLE "polls_choice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "choice_text" varchar(200) NOT NULL, "votes" integer NOT NULL, "question_id" bigint NOT NULL REFERENCES "polls_question" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE INDEX "polls_choice_question_id_c5b4b260" ON "polls_choice" ("question_id");
COMMIT;
```

17. Синхронизация изменений

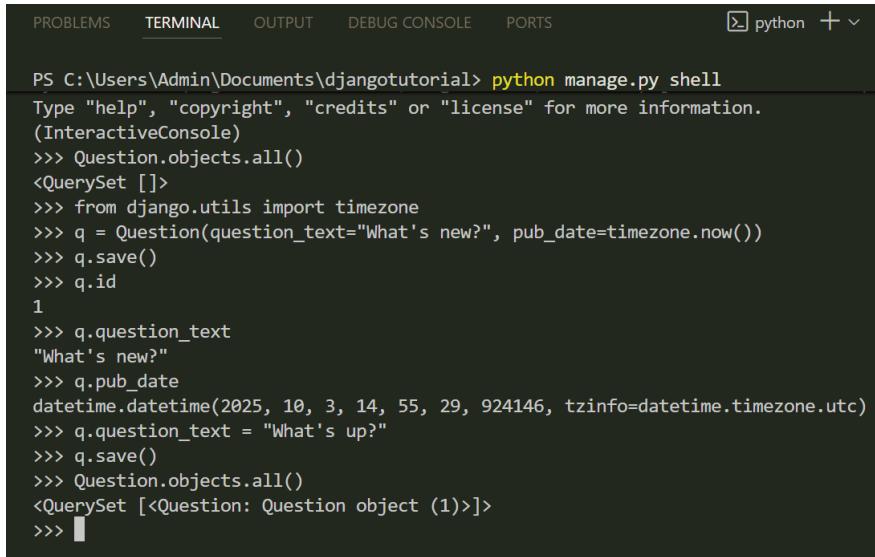
```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS power

("id") DEFERRABLE INITIALLY DEFERRED);

COMMIT;
PS C:\Users\Admin\Documents\djngotutorial> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK
PS C:\Users\Admin\Documents\djngotutorial>
```

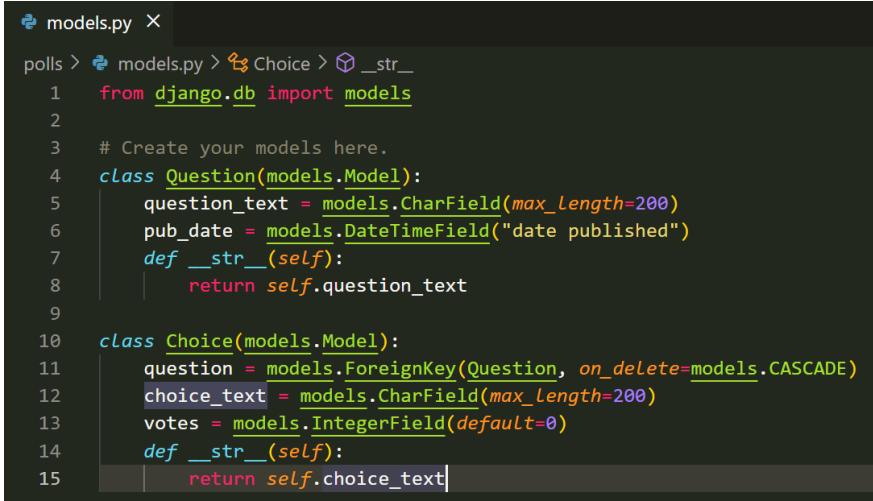
API

18. Модели в консоли



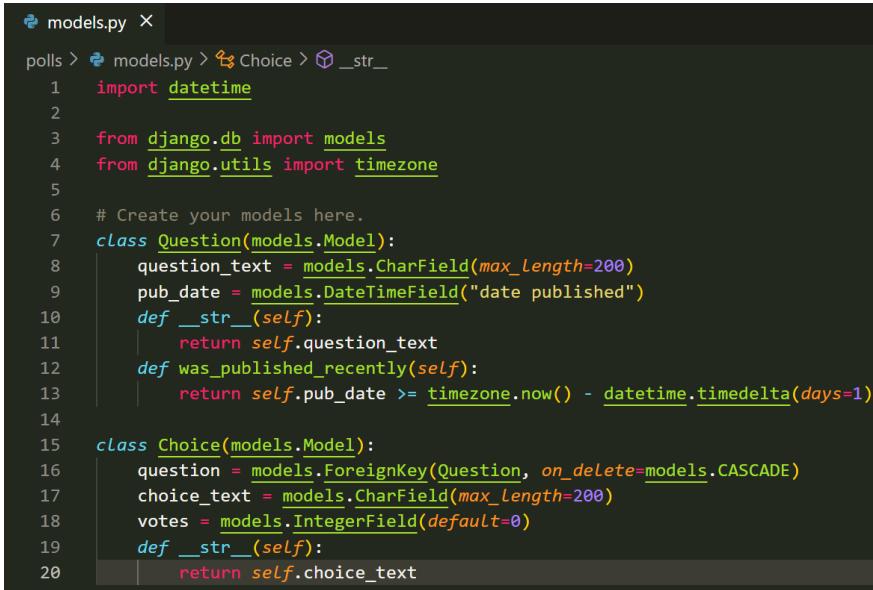
```
PS C:\Users\Admin\Documents\.djangotutorial> python manage.py shell
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> Question.objects.all()
<QuerySet []>
>>> from django.utils import timezone
>>> q = Question(question_text="What's new?", pub_date=timezone.now())
>>> q.save()
>>> q.id
1
>>> q.question_text
"What's new?"
>>> q.pub_date
datetime.datetime(2025, 10, 3, 14, 55, 29, 924146, tzinfo=datetime.timezone.utc)
>>> q.question_text = "What's up?"
>>> q.save()
>>> Question.objects.all()
<QuerySet [Question: Question object (1)]>
>>> 
```

19. Добавление двух методов к модели



```
polls > models.py > Choice > __str__
1  from django.db import models
2
3  # Create your models here.
4  class Question(models.Model):
5      question_text = models.CharField(max_length=200)
6      pub_date = models.DateTimeField("date published")
7      def __str__(self):
8          return self.question_text
9
10 class Choice(models.Model):
11     question = models.ForeignKey(Question, on_delete=models.CASCADE)
12     choice_text = models.CharField(max_length=200)
13     votes = models.IntegerField(default=0)
14     def __str__(self):
15         return self.choice_text| 
```

20. Добавление пользовательского метода



```
polls > models.py > Choice > __str__
1  import datetime
2
3  from django.db import models
4  from django.utils import timezone
5
6  # Create your models here.
7  class Question(models.Model):
8      question_text = models.CharField(max_length=200)
9      pub_date = models.DateTimeField("date published")
10     def __str__(self):
11         return self.question_text
12     def was_published_recently(self):
13         return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
14
15 class Choice(models.Model):
16     question = models.ForeignKey(Question, on_delete=models.CASCADE)
17     choice_text = models.CharField(max_length=200)
18     votes = models.IntegerField(default=0)
19     def __str__(self):
20         return self.choice_text| 
```

21. В консоли

```
PS C:\Users\Admin\Documents\djangotutorial> python manage.py shell
8 objects imported automatically (use -v 2 for details).

Ctrl click to launch VS Code Native REPL
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug 6 2024, 20:45:27) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> Question.objects.all()
<QuerySet [<Question: What's up?>]>
>>> Question.objects.filter(id=1)
<QuerySet [<Question: What's up?>]>
>>> Question.objects.filter(question_text__startswith="What")
<QuerySet [<Question: What's up?>]>
>>> from django.utils import timezone
>>> current_year = timezone.now().year
>>> Question.objects.get(pub_date__year=current_year)
<Question: What's up?>
>>> Question.objects.get(id=2)
Traceback (most recent call last):
  File "C:\Program Files\Python312\Lib\code.py", line 90, in runcode
    exec(code, self.locals)
  File "<console>", line 1, in <module>
  File "C:\Users\Admin\AppData\Roaming\Python\Python312\site-packages\django\db\models\manager.py",
line 87, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
                                                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Admin\AppData\Roaming\Python\Python312\site-packages\django\db\models\query.py", l
ine 633, in get
    raise self.model.DoesNotExist(
polls.models.Question.DoesNotExist: Question matching query does not exist.
>>> Question.objects.get(pk=1)
<Question: What's up?>
>>> q = Question.objects.get(pk=1)
>>> q.was_published_recently()
True
>>> q = Question.objects.get(pk=1)
>>> q.choice_set.all()
<QuerySet []>
>>> q.choice_set.create(choice_text="Not much", votes=0)
<Choice: Not much>
>>> q.choice_set.create(choice_text="The sky", votes=0)
<Choice: The sky>
>>> c = q.choice_set.create(choice_text="Just hacking again", votes=0)
>>> c.question
<Question: What's up?>
>>> q.choice_set.all()
<QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>
>>> q.choice_set.count()
3
>>> Choice.objects.filter(question__pub_date__year=current_year)
<QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>
>>> c = q.choice_set.filter(choice_text__startswith="Just hacking")
>>> c.delete()
(1, {'polls.Choice': 1})
>>> 
```

Django Admin

22. Создание пользователя с правами администратора

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powershell +
```

- PS C:\Users\Admin\Documents\djangotutorial> python manage.py createsuperuser
Username (leave blank to use 'admin'): admin
Email address: dmitry [REDACTED]
Password:
Password (again):
Superuser created successfully.
- PS C:\Users\Admin\Documents\djangotutorial> []

Password: djangoav7mi

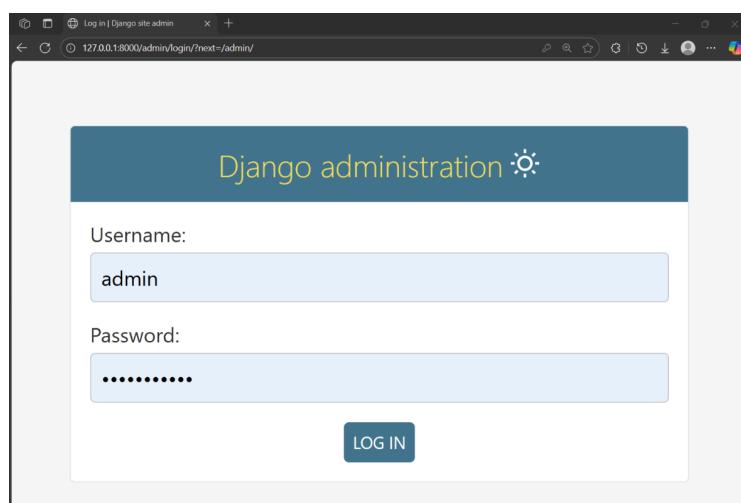
23. Запуск сервера разработки

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS python + □
```

- PS C:\Users\Admin\Documents\djangotutorial> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 03, 2025 - 21:13:09
Django version 5.2.7, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use
or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en
ent/
[03/Oct/2025 21:13:15] "GET /admin/ HTTP/1.1" 302 0
[03/Oct/2025 21:13:15] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 4173
[03/Oct/2025 21:13:15] "GET /static/admin/css/base.css HTTP/1.1" 200 22120
[03/Oct/2025 21:13:15] "GET /static/admin/css/dark_mode.css HTTP/1.1" 200 2808
[03/Oct/2025 21:13:15] "GET /static/admin/js/theme.js HTTP/1.1" 200 1653
[03/Oct/2025 21:13:15] "GET /static/admin/css/nav_sidebar.css HTTP/1.1" 200 2810
[03/Oct/2025 21:13:15] "GET /static/admin/css/login.css HTTP/1.1" 200 951
[03/Oct/2025 21:13:15] "GET /static/admin/js/nav_sidebar.js HTTP/1.1" 200 3063
[03/Oct/2025 21:13:15] "GET /static/admin/css/responsive.css HTTP/1.1" 200 16565
[]



24. Сайт администратора

The screenshot shows the Django administration interface. At the top, it says "Django administration" and "WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT". Below this is a sidebar titled "Site administration" with sections for "AUTHENTICATION AND AUTHORIZATION" (Groups and Users), "Recent actions", and "My actions" (None available).

25. Приложение для опросов изменяющееся в админке

```
polls > admin.py
1  from django.contrib import admin
2
3  from .models import Question
4  # Register your models here.
5  admin.site.register(Question)
```

26. Результат

The screenshot shows the Django administration interface after the code change. It includes the "POLL" section with the "Questions" model, which has "Add" and "Change" buttons. The rest of the interface is identical to the first screenshot.

Баранов Д.А. ИВТ 2.1

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Polls > Questions

Select question to change

ADD QUESTION +

Action: ----- Go

0 of 1 selected

QUESTION

What's up?

1 question

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Polls > Questions > What's up?

The question "What's up?" was changed successfully. You may edit it again below.

Change question

What's up?

HISTORY

Question text:

What's up?

Date published:

Date: 2025-10-03 Today |

Time: 18:27:32 Now |

Note: You are 3 hours ahead of server time.

SAVE

Save and add another

Save and continue editing

Delete

The screenshot shows the Django administration interface. The title bar says "Django administration". Below it, a navigation bar includes "WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT". The main content area shows the path "Home > Polls > Questions > What's up? > History". The title of the page is "Change history: What's up?". A table lists one entry: "Oct. 3, 2025, 6:27 p.m." by "admin" with the action "Changed Date published.". There is also a link "1 entry".

Часть 3

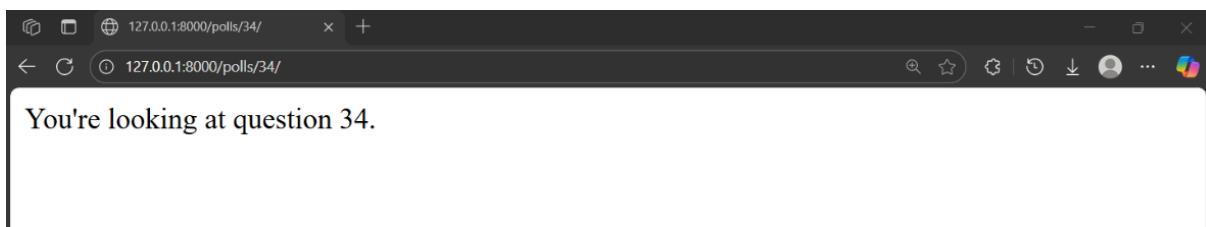
Добавление более сложных views, работа с шаблонами, обработка ошибок 404, использование get_object_or_404, именование URL.

27. views

```
polls > views.py > ...
1  from django.http import HttpResponseRedirect
2
3  # Create your views here.
4  def index(request):
5      return HttpResponseRedirect("Hello, world. You're at the polls index.")
6
7  def detail(request, question_id):
8      return HttpResponseRedirect("You're looking at question %s." % question_id)
9
10 def results(request, question_id):
11     response = "You're looking at the results of question %s."
12     return HttpResponseRedirect(response % question_id)
13
14 def vote(request, question_id):
15     return HttpResponseRedirect("You're voting on question %s." % question_id)
```

28. Подключение новых представлений к модулю

```
polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4  urlpatterns = [
5      path("", views.index, name="index"),
6      path("<int:question_id>/", views.detail, name="detail"),
7      path("<int:question_id>/results/", views.results, name="results"),
8      path("<int:question_id>/vote/", views.vote, name="votes"),
9  ]
```



29. views

```
polls > views.py > vote
1  from django.http import HttpResponseRedirect
2
3  from .models import Question
4
5  # Create your views here.
6  def index(request):
7      latest_question_list = Question.objects.order_by("-pub_date")[:5]
8      output = ", ".join([q.question_text for q in latest_question_list])
9      return HttpResponseRedirect(output)
10
11 def detail(request, question_id):
12     return HttpResponseRedirect("You're looking at question %s." % question_id)
13
14 def results(request, question_id):
15     response = "You're looking at the results of question %s."
16     return HttpResponseRedirect(response % question_id)
17
18 def vote(request, question_id):
19     return HttpResponseRedirect("You're voting on question %s." % question_id)
```

30. Шаблон

```
polls > templates > polls > index.html > ...
1  {% if latest_question_list %}
2      <ul>
3          {% for question in latest_question_list %}
4              <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a>
5          </li>
6          {% endfor %}
7      </ul>
8  {% else %}
9      <p>No polls are available.</p>
10  {% endif %}
```

31. Загрузка шаблона

```
polls > views.py > vote
1  from django.http import HttpResponseRedirect
2  from django.template import loader
3
4  from .models import Question
5
6  # Create your views here.
7  def index(request):
8      latest_question_list = Question.objects.order_by("-pub_date")[:5]
9      template = loader.get_template("polls/index.html")
10     context = {"latest_question_list": latest_question_list}
11     return HttpResponseRedirect(template.render(context, request))
12
13 def detail(request, question_id):
14     return HttpResponseRedirect("You're looking at question %s." % question_id)
15
16 def results(request, question_id):
17     response = "You're looking at the results of question %s."
18     return HttpResponseRedirect(response % question_id)
19
20 def vote(request, question_id):
21     return HttpResponseRedirect("You're voting on question %s." % question_id)
```



32. Использование render

```
views.py  x
polls > views.py > vote
1  from django.http import HttpResponseRedirect
2  from django.shortcuts import render
3
4  from .models import Question
5
6  # Create your views here.
7  def index(request):
8      latest_question_list = Question.objects.order_by("-pub_date")[:5]
9      context = {"latest_question_list": latest_question_list}
10     return render(request, "polls/index.html", context)
11
12 def detail(request, question_id):
13     return HttpResponseRedirect("You're looking at question %s." % question_id)
14
15 def results(request, question_id):
16     response = "You're looking at the results of question %s."
17     return HttpResponseRedirect(response % question_id)
18
19 def vote(request, question_id):
20     return HttpResponseRedirect("You're voting on question %s." % question_id)
```

33. Сообщение об ошибке 404

```
views.py  x
polls > views.py > vote
1  from django.http import HttpResponseRedirect
2  from django.http import Http404
3  from django.shortcuts import render
4
5  from .models import Question
6
7  # Create your views here.
8  def index(request):
9      latest_question_list = Question.objects.order_by("-pub_date")[:5]
10     context = {"latest_question_list": latest_question_list}
11     return render(request, "polls/index.html", context)
12
13 def detail(request, question_id):
14     try:
15         question = Question.objects.get(pk=question_id)
16     except Question.DoesNotExist:
17         raise Http404("Question does not exist")
18     return render(request, "polls/detail.html", {"question": question})
19
20 def results(request, question_id):
21     response = "You're looking at the results of question %s."
22     return HttpResponseRedirect(response % question_id)
23
24 def vote(request, question_id):
25     return HttpResponseRedirect("You're voting on question %s." % question_id)
```

34. Шаблон

```
detail.html X
polls > templates > polls > detail.html
1  {{ question }}
```

35. views

```
detail.html views.py X
polls > views.py > vote
1  from django.shortcuts import get_object_or_404, render
2
3  from .models import Question
4
5  # Create your views here.
6  def index(request):
7      latest_question_list = Question.objects.order_by("-pub_date")[:5]
8      context = {"latest_question_list": latest_question_list}
9      return render(request, "polls/index.html", context)
10
11  def detail(request, question_id):
12      question = get_object_or_404(Question, pk=question_id)
13      return render(request, "polls/detail.html", {"question": question})
14
15  def results(request, question_id):
16      response = "You're looking at the results of question %s."
17      return HttpResponse(response % question_id)
18
19  def vote(request, question_id):
20      return HttpResponse("You're voting on question %s." % question_id)
```

36. Система шаблонов

```
detail.html X
polls > templates > polls > detail.html > ul
1  <h1>{{ question.question_text }}</h1>
2  <ul>
3  {% for choice in question.choice_set.all %}
4      <li>{{ choice.choice_text }}</li>
5  {% endfor %}
6  </ul>
```

37. Удаление жестко заданных URL в шаблонах

```
index.html X
polls > templates > polls > index.html > ...
1  {% if latest_question_list %}
2      <ul>
3          {% for question in latest_question_list %}
4              <li><a href="{% url 'detail' question.id %}">{{ question.question_text }}</a>
5          </li>
6          {% endfor %}
7      </ul>
8  {% else %}
9      <p>No polls are available.</p>
10  {% endif %}
```

```
urls.py X ...
polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4  urlpatterns = [
5      path("", views.index, name="index"),
6      path("specifics/<int:question_id>/", views.detail, name="detail"),
7      path("<int:question_id>/results/", views.results, name="results"),
8      path("<int:question_id>/vote/", views.vote, name="votes"),
9  ]
```

38. Пространство имен URL-адресов

```
urls.py X
polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4  app_name = "polls"
5  urlpatterns = [
6      path("", views.index, name="index"),
7      path("<int:question_id>/", views.detail, name="detail"),
8      path("<int:question_id>/results/", views.results, name="results"),
9      path("<int:question_id>/vote/", views.vote, name="vote"),
10 ]

```

39. Навести указатель мыши на детальный вид в пространстве имен

```
index.html X
polls > templates > polls > index.html > ...
1  {% if latest_question_list %}
2      <ul>
3          {% for question in latest_question_list %}
4              <li><a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}</a></li>
5          {% endfor %}
6      </ul>
7  {% else %}
8      <p>No polls are available.</p>
9  {% endif %}
10 
```

Часть 4

Формы и взаимодействие фронтенда и бэкенда, более продвинутые view, включая generic (классы).

40. Обновление шаблона

```
detail.html X
polls > templates > polls > detail.html > form
1  <form action="{% url 'polls:vote' question.id %}" method="post">
2  {% csrf_token %}
3  <fieldset>
4      <legend><h1>{{ question.question_text }}</h1></legend>
5      {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
6      {% for choice in question.choice_set.all %}
7          <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}">
8          <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br>
9      {% endfor %}
10 
```

41. Обновление views

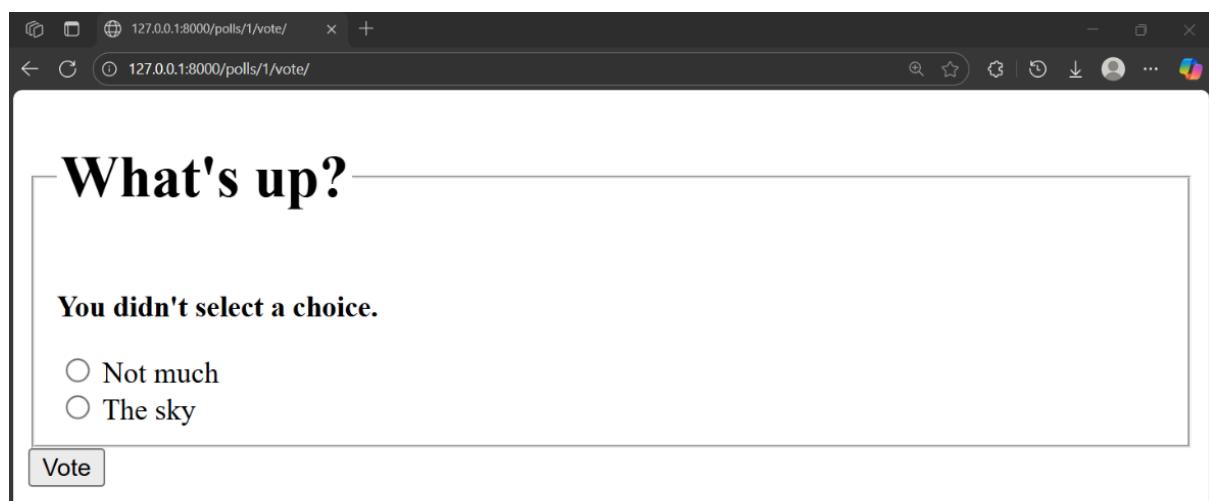
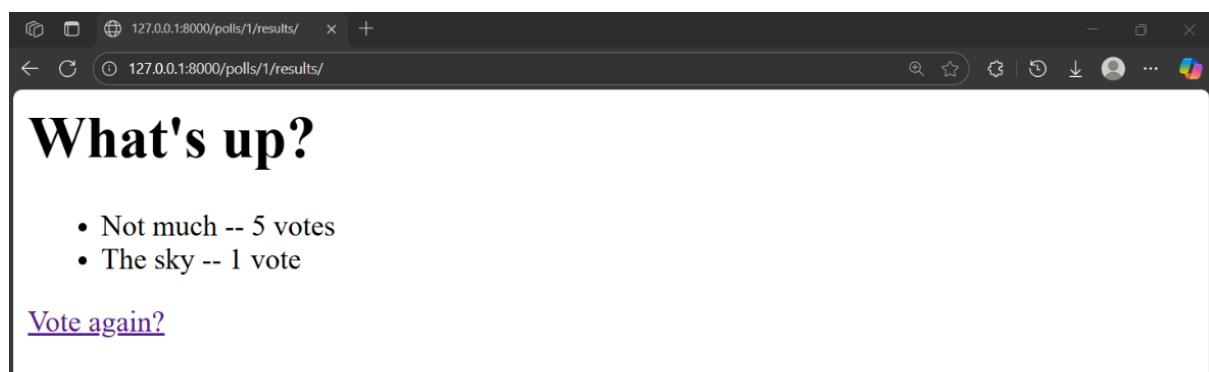
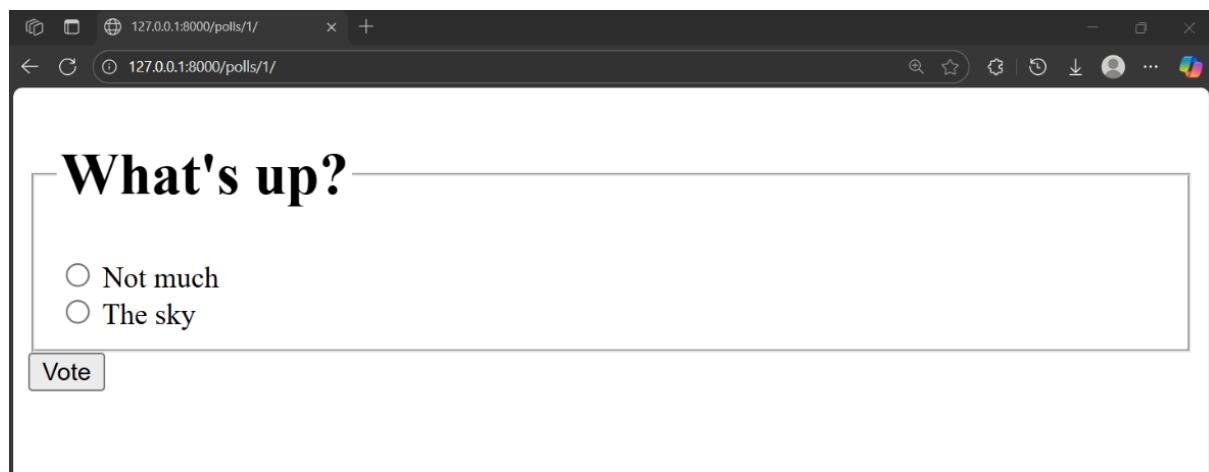
```
views.py ×
polls > views.py > vote
1  from django.db.models import F
2  from django.http import HttpResponseRedirect, HttpResponseRedirect
3  from django.shortcuts import get_object_or_404, render
4  from django.urls import reverse
5
6  from .models import Choice, Question
7
8  # Create your views here.
9  def index(request):
10     latest_question_list = Question.objects.order_by("-pub_date")[:5]
11     context = {"latest_question_list": latest_question_list}
12     return render(request, "polls/index.html", context)
13
14  def detail(request, question_id):
15     question = get_object_or_404(Question, pk=question_id)
16     return render(request, "polls/detail.html", {"question": question})
17
18  def results(request, question_id):
19     response = "You're looking at the results of question %s."
20     return HttpResponseRedirect(response % question_id)
21
22  def vote(request, question_id):
23     question = get_object_or_404(Question, pk=question_id)
24     try:
25         selected_choice = question.choice_set.get(pk=request.POST["choice"])
26     except (KeyError, Choice.DoesNotExist):
27         return render(
28             request,
29             "polls/detail.html",
30             {
31                 "question": question,
32                 "error_message": "You didn't select a choice.",
33             },
34         )
35     else:
36         selected_choice.votes = F("votes") + 1
37         selected_choice.save()
38     return HttpResponseRedirect(reverse("polls:results", args=(question.id,)))
```

42. Представление перенаправляет на результаты для вопроса

```
def results(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    return render(request, "polls/results.html", {"question": question})
```

43. results.html

```
results.html ×
polls > templates > polls > results.html > a
1  <h1>{{ question.question_text }}</h1>
2
3  <ul>
4      {% for choice in question.choice_set.all %}
5          <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{ choice.votes|pluralize }}</li>
6      {% endfor %}
7  </ul>
8  <a href="{% url 'polls:detail' question.id %}">Vote again?</a>
```



44. Изменение URLconf

```
urls.py  X
polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4
5  app_name = "polls"
6  urlpatterns = [
7      path("", views.IndexView.as_view(), name="index"),
8      path("<int:pk>/", views.DetailView.as_view(), name="detail"),
9      path("<int:pk>/results/", views.ResultsView.as_view(), name="results"),
10     path("<int:question_id>/vote/", views.vote, name="vote"),
11 ]
```

45. Изменение представлений

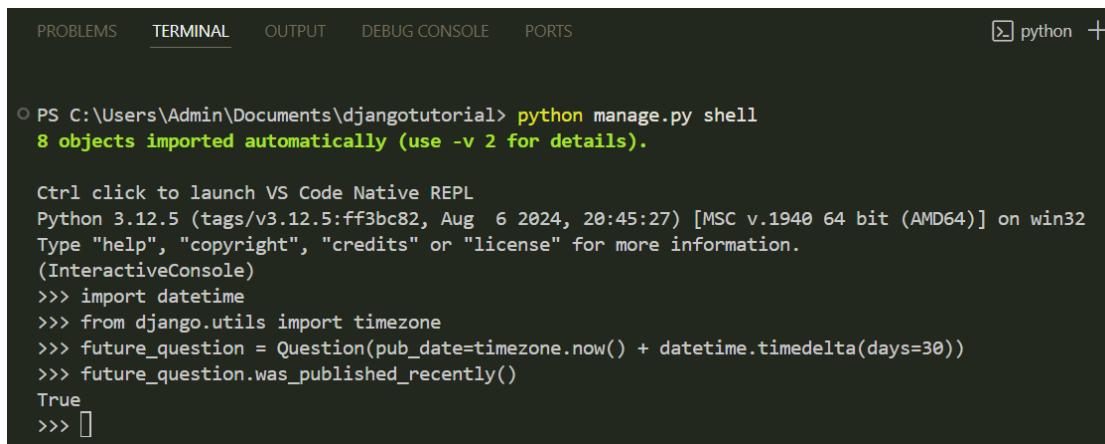


```
views.py  X
polls > views.py > vote
1  from django.db.models import F
2  from django.http import HttpResponseRedirect
3  from django.shortcuts import get_object_or_404, render
4  from django.urls import reverse
5  from django.views import generic
6
7  from .models import Choice, Question
8
9  # Create your views here.
10 class IndexView(generic.ListView):
11     template_name = "polls/index.html"
12     context_object_name = "latest_question_list"
13
14     def get_queryset(self):
15         """Return the last five published questions."""
16         return Question.objects.order_by("-pub_date")[:5]
17
18 class DetailView(generic.DetailView):
19     model = Question
20     template_name = "polls/detail.html"
21
22 class ResultsView(generic.DetailView):
23     model = Question
24     template_name = "polls/results.html"
25
26 def vote(request, question_id):
27     question = get_object_or_404(Question, pk=question_id)
28     try:
29         selected_choice = question.choice_set.get(pk=request.POST["choice"])
30     except (KeyError, Choice.DoesNotExist):
31         return render(
32             request,
33             "polls/detail.html",
34             {
35                 "question": question,
36                 "error_message": "You didn't select a choice.",
37             },
38         )
39     else:
40         selected_choice.votes = F("votes") + 1
41         selected_choice.save()
42         return HttpResponseRedirect(reverse("polls:results", args=(question.id,)))
```

Часть 5

Тестирование приложения, введение юнит-тестов

46. Выявление бага



PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS python +

PS C:\Users\Admin\Documents\djngotutorial> python manage.py shell
8 objects imported automatically (use -v 2 for details).

Ctrl click to launch VS Code Native REPL
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug 6 2024, 20:45:27) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> import datetime
>>> from django.utils import timezone
>>> future_question = Question(pub_date=timezone.now() + datetime.timedelta(days=30))
>>> future_question.was_published_recently()
True
>>> []

47. Создание теста для выявления ошибки

```
tests.py  X
polls > tests.py > QuestionModelTests > test_was_published_recently_with_future_question
1 import datetime
2
3 from django.test import TestCase
4 from django.utils import timezone
5
6 from .models import Question
7
8 # Create your tests here.
9 class QuestionModelTests(TestCase):
10     def test_was_published_recently_with_future_question(self):
11         """
12             was_published_recently() returns False for questions whose pub_date is in the future.
13         """
14         time = timezone.now() + datetime.timedelta(days=30)
15         future_question = Question(pub_date=time)
16         self.assertIs(future_question.was_published_recently(), False)
```

48. Проведение тестов

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powershell + □ ─
PS C:\Users\Admin\Documents\djngotutorial> python manage.py test polls
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
F
=====
FAIL: test_was_published_recently_with_future_question (polls.tests.QuestionModelTests.test_was_published_recently_with_future_question)
was_published_recently() returns False for questions whose pub_date is in the future.

Traceback (most recent call last):
  File "C:\Users\Admin\Documents\djngotutorial\polls\tests.py", line 16, in test_was_published_recently_with_future_question
    self.assertIs(future_question.was_published_recently(), False)
AssertionError: True is not False

-----
Ran 1 test in 0.003s

FAILED (failures=1)
Destroying test database for alias 'default'...
PS C:\Users\Admin\Documents\djngotutorial> [ ]
```

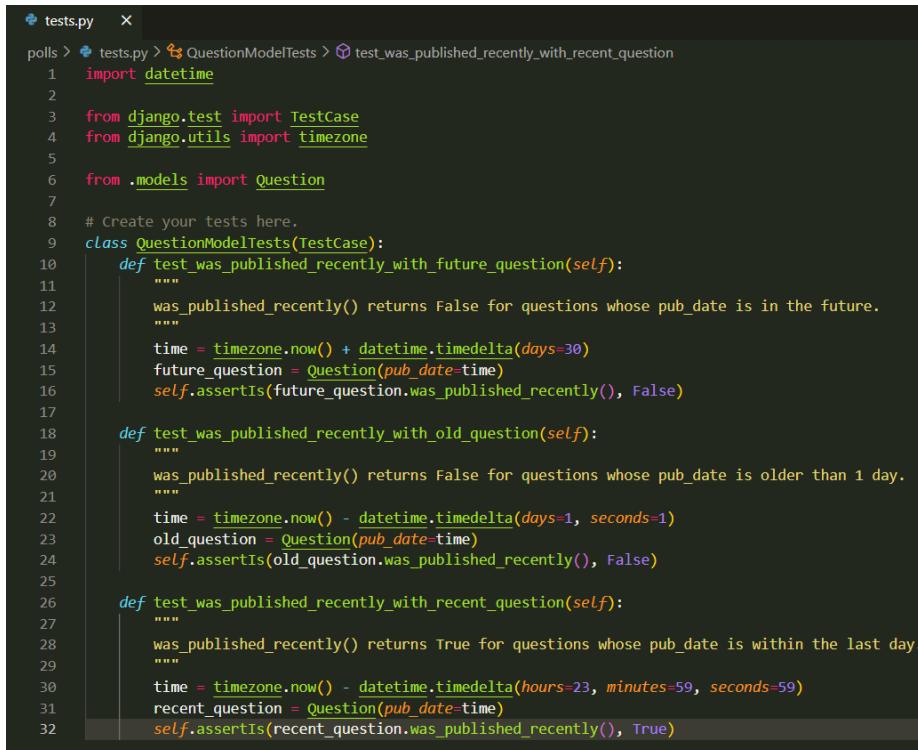
49. Исправление ошибки

```
def was_published_recently(self):
    now = timezone.now()
    return now - datetime.timedelta(days=1) <= self.pub_date <= now
```

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powershell
PS C:\Users\Admin\Documents\djngotutorial> python manage.py test polls
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
-----
Ran 1 test in 0.001s

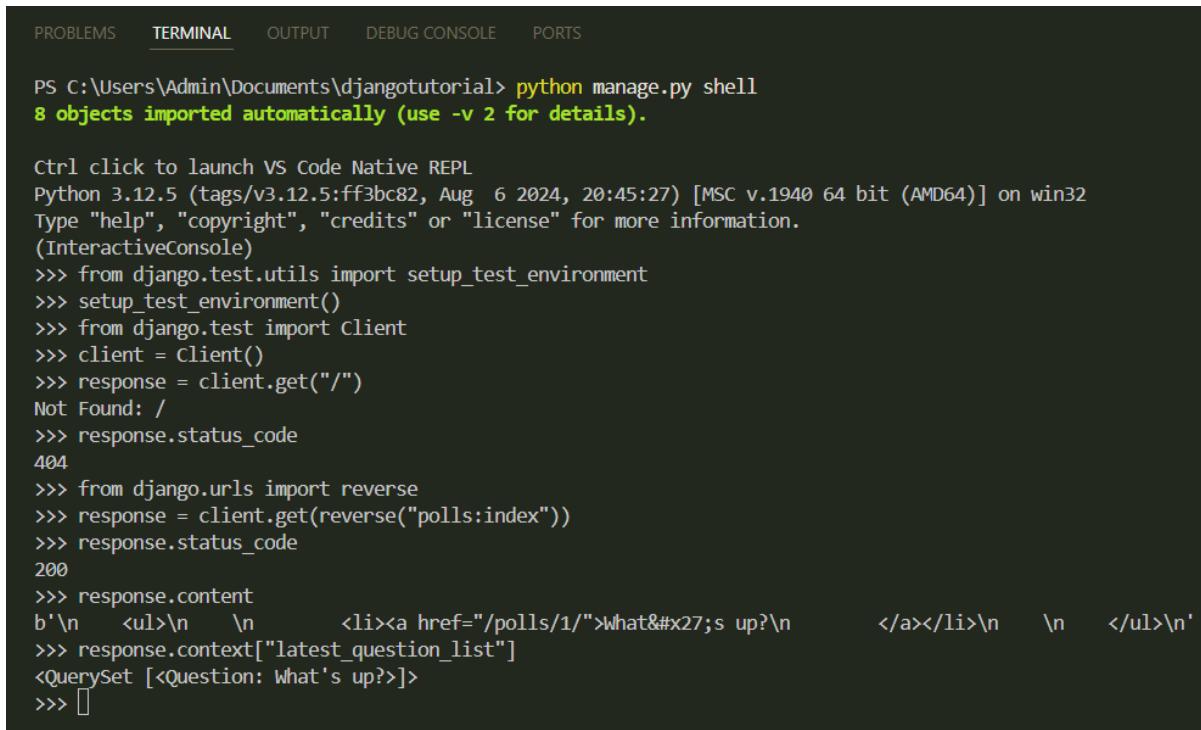
OK
Destroying test database for alias 'default'...
PS C:\Users\Admin\Documents\djngotutorial> [ ]
```

50. Более комплексные испытания



```
tests.py  X
polls > tests.py > QuestionModelTests > test_was_published_recently_with_recent_question
1 import datetime
2
3 from django.test import TestCase
4 from django.utils import timezone
5
6 from .models import Question
7
8 # Create your tests here.
9 class QuestionModelTests(TestCase):
10     def test_was_published_recently_with_future_question(self):
11         """
12             was_published_recently() returns False for questions whose pub_date is in the future.
13         """
14         time = timezone.now() + datetime.timedelta(days=30)
15         future_question = Question(pub_date=time)
16         self.assertIs(future_question.was_published_recently(), False)
17
18     def test_was_published_recently_with_old_question(self):
19         """
20             was_published_recently() returns False for questions whose pub_date is older than 1 day.
21         """
22         time = timezone.now() - datetime.timedelta(days=1, seconds=1)
23         old_question = Question(pub_date=time)
24         self.assertIs(old_question.was_published_recently(), False)
25
26     def test_was_published_recently_with_recent_question(self):
27         """
28             was_published_recently() returns True for questions whose pub_date is within the last day.
29         """
30         time = timezone.now() - datetime.timedelta(hours=23, minutes=59, seconds=59)
31         recent_question = Question(pub_date=time)
32         self.assertIs(recent_question.was_published_recently(), True)
```

51. Тестовый клиент Django



PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS

```
PS C:\Users\Admin\Documents\.djangoproject> python manage.py shell
8 objects imported automatically (use -v 2 for details).

Ctrl click to launch VS Code Native REPL
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug 6 2024, 20:45:27) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.test.utils import setup_test_environment
>>> setup_test_environment()
>>> from django.test import Client
>>> client = Client()
>>> response = client.get("/")
Not Found: /
>>> response.status_code
404
>>> from django.urls import reverse
>>> response = client.get(reverse("polls:index"))
>>> response.status_code
200
>>> response.content
b'\n    <ul>\n        \n            <li><a href="/polls/1/">What's up?\n                </a></li>\n        \n    </ul>\n'
>>> response.context["latest_question_list"]
<QuerySet [<Question: What's up?>]>
>>> []
```

52. Улучшение view get_queryset(self)

```
views.py  X
polls > views.py > DetailView
1  from django.db.models import F
2  from django.http import HttpResponseRedirect
3  from django.shortcuts import get_object_or_404, render
4  from django.urls import reverse
5  from django.views import generic
6  from django.utils import timezone
7
8  from .models import Choice, Question
9
10 # Create your views here.
11 class IndexView(generic.ListView):
12     template_name = "polls/index.html"
13     context_object_name = "latest_question_list"
14
15     def get_queryset(self):
16         """
17             Return the last five published questions (not including those set to be published in the future).
18         """
19         return Question.objects.filter(pub_date__lte=timezone.now()).order_by("-pub_date")[:5]
```

53. Тестирование нового представления

```
tests.py  X
polls > tests.py > QuestionIndexViewTests > test_two_past_questions
33     self.assertEqual(recent_question.was_published_recently(), True)
34
35     def create_question(question_text, days):
36         """
37             Create a question with the given `question_text` and published the
38             given number of `days` offset to now (negative for questions published
39             in the past, positive for questions that have yet to be published).
40         """
41         time = timezone.now() + datetime.timedelta(days=days)
42         return Question.objects.create(question_text=question_text, pub_date=time)
43
44     class QuestionIndexViewTests(TestCase):
45         def test_no_questions(self):
46             """
47                 If no questions exist, an appropriate message is displayed.
48             """
49             response = self.client.get(reverse("polls:index"))
50             self.assertEqual(response.status_code, 200)
51             self.assertContains(response, "No polls are available.")
52             self.assertQuerySetEqual(response.context["latest_question_list"], [])
53
54         def test_past_question(self):
55             """
56                 Questions with a pub_date in the past are displayed on the
57                 index page.
58             """
59             question = create_question(question_text="Past question.", days=-30)
60             response = self.client.get(reverse("polls:index"))
61             self.assertQuerySetEqual(
62                 response.context["latest_question_list"],
63                 [question],
64             )
65
66         def test_future_question(self):
67             """
68                 Questions with a pub_date in the future aren't displayed on
69                 the index page.
70             """
71             create_question(question_text="Future question.", days=30)
72             response = self.client.get(reverse("polls:index"))
73             self.assertContains(response, "No polls are available.")
74             self.assertQuerySetEqual(response.context["latest_question_list"], [])
75
76         def test_future_question_and_past_question(self):
77             """
78                 Even if both past and future questions exist, only past questions
79                 are displayed.
80             """
81             question = create_question(question_text="Past question.", days=-30)
82             create_question(question_text="Future question.", days=30)
83             response = self.client.get(reverse("polls:index"))
84             self.assertQuerySetEqual(
85                 response.context["latest_question_list"],
86                 [question],
87             )
88
89         def test_two_past_questions(self):
90             """
91                 The questions index page may display multiple questions.
92             """
93             question1 = create_question(question_text="Past question 1.", days=-30)
94             question2 = create_question(question_text="Past question 2.", days=-5)
95             response = self.client.get(reverse("polls:index"))
96             self.assertQuerySetEqual(
97                 response.context["latest_question_list"],
98                 [question2, question1],
```

54. Тестирование DetailView

```
21  class DetailView(generic.DetailView):
22      model = Question
23      template_name = "polls/detail.html"
24
25      def get_queryset(self):
26          """
27              Excludes any questions that aren't published yet.
28          """
29          return Question.objects.filter(pub_date__lte=timezone.now())
```

55. Тесты

```
tests.py  X
polls > tests.py > QuestionDetailViewTests > test_past_question
100
101  class QuestionDetailViewTests(TestCase):
102      def test_future_question(self):
103          """
104              The detail view of a question with a pub_date in the future
105              returns a 404 not found.
106          """
107          future_question = create_question(question_text="Future question.", days=5)
108          url = reverse("polls:detail", args=(future_question.id,))
109          response = self.client.get(url)
110          self.assertEqual(response.status_code, 404)
111
112      def test_past_question(self):
113          """
114              The detail view of a question with a pub_date in the past
115              displays the question's text.
116          """
117          past_question = create_question(question_text="Past Question.", days=-5)
118          url = reverse("polls:detail", args=(past_question.id,))
119          response = self.client.get(url)
120          self.assertContains(response, past_question.question_text)
```

56. Результаты тестов

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS

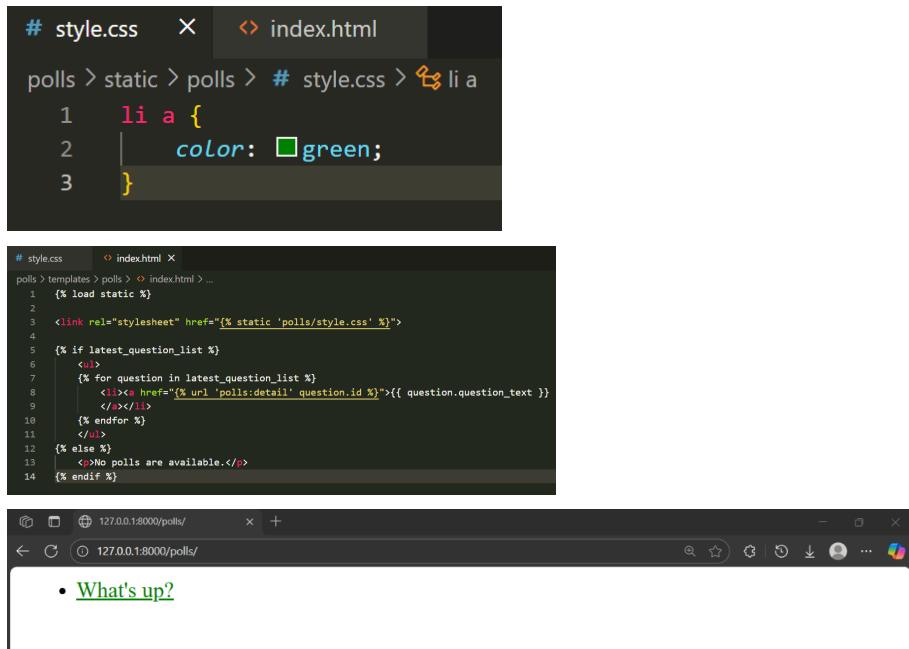
PS C:\Users\Admin\Documents\djangotutorial> python manage.py test polls
Found 10 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 10 tests in 0.060s

OK
Destroying test database for alias 'default'...
D PS C:\Users\Admin\Documents\djangotutorial>
```

Часть 6

Статические файлы (CSS, JavaScript, изображения) и как их обслуживать

57. Настройка внешнего вида приложения



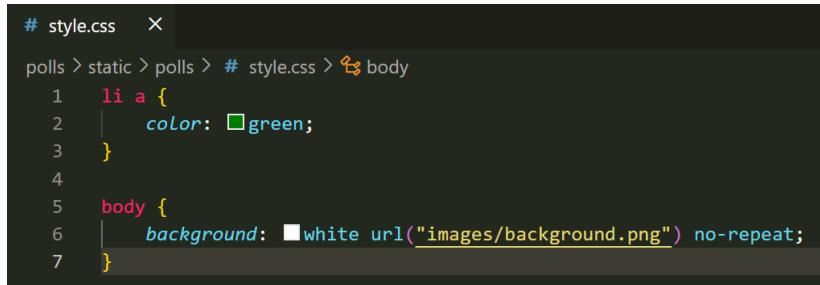
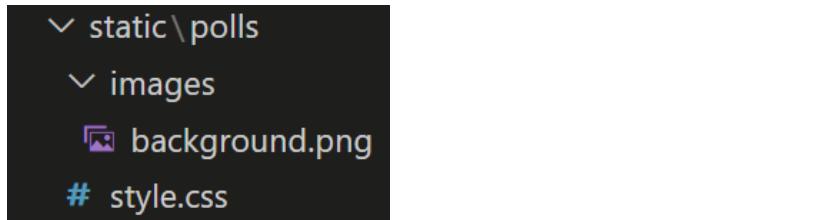
```
# style.css    X  index.html

polls > static > polls > # style.css > li a
1   li a {
2     color: green;
3   }
```

```
# style.css    X  index.html
polls > templates > polls > index.html > ...
1  {% load static %}
2
3  <link rel="stylesheet" href="{% static 'polls/style.css' %}">
4
5  {% if latest_question_list %}
6    <ul>
7      {% for question in latest_question_list %}
8        <li><a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}</a></li>
9      {% endfor %}
10   </ul>
11   {% else %}
12     <p>No polls are available.</p>
13   {% endif %}
```



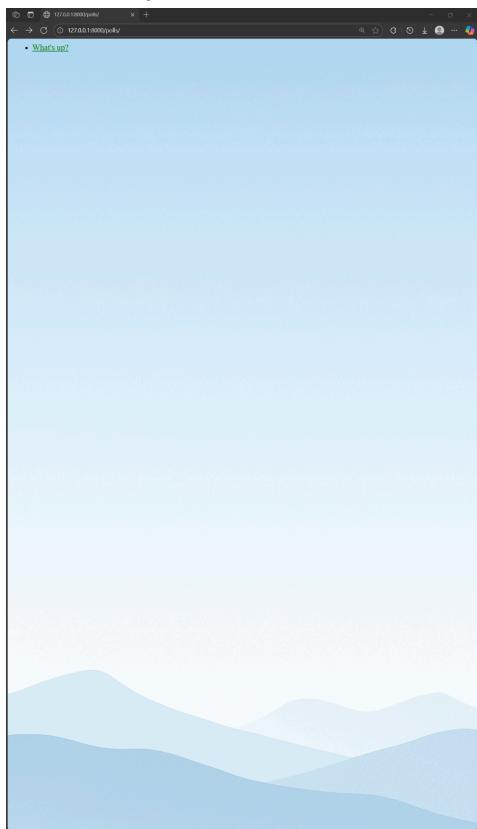
58. Добавление фонового изображения



```
# style.css    X

polls > static > polls > # style.css > body
1   li a {
2     color: green;
3   }
4
5   body {
6     background: white url("images/background.png") no-repeat;
7   }
```

59. Результат



Часть 7

Настройка и кастомизация админ-интерфейса

60. Настройка формы администратора

```
admin.py  X
polls > admin.py > ...
1   from django.contrib import admin
2
3   from .models import Question
4
5   # Register your models here.
6
7   class QuestionAdmin(admin.ModelAdmin):
8       fields = ["pub_date", "question_text"]
9
10  admin.site.register(Question, QuestionAdmin)
```

The screenshot shows the Django administration interface. The title bar says "Django administration" and "WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT". The URL in the address bar is "127.0.0.1:8000/admin/polls/question/1/change/". The main content area shows a poll entry with the title "What's up?". It includes fields for "Date published" (set to 2025-10-03) and "Time" (set to 18:27:32). A note at the bottom says "Note: You are 3 hours ahead of server time.". Below the form, there are four action buttons: "SAVE" (dark blue), "Save and add another" (medium blue), "Save and continue editing" (light blue), and "Delete" (red).

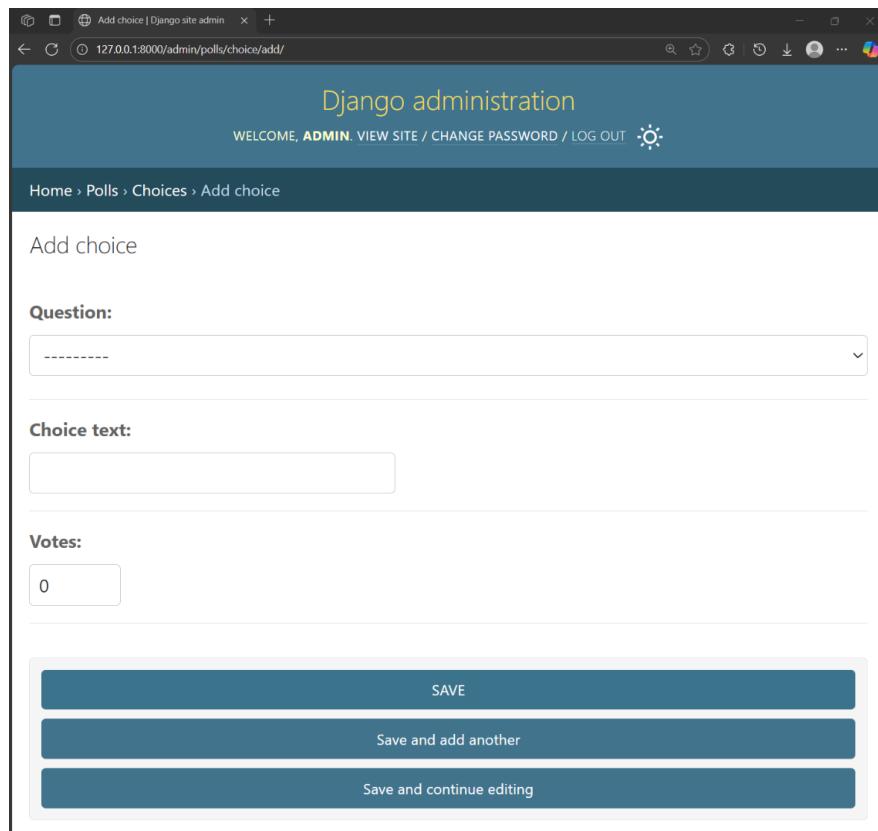
61. Разделение формы вверх в наборы полей

```
admin.py  X
polls > admin.py > ...
1  from django.contrib import admin
2
3  from .models import Question
4
5  # Register your models here.
6
7  class QuestionAdmin(admin.ModelAdmin):
8      fieldsets = [
9          (None, {"fields": ["question_text"]}),
10         ("Date information", {"fields": ["pub_date"]}),
11     ]
12
13 admin.site.register(Question, QuestionAdmin)
```

The screenshot shows the Django administration interface for a poll titled "What's up?". The "Question text:" field contains "What's up?". The "Date published:" field shows "2025-10-03" and the "Time published:" field shows "18:27:32". A note at the bottom states "Note: You are 3 hours ahead of server time." Below the form are four action buttons: "SAVE" (dark blue), "Save and add another" (medium blue), "Save and continue editing" (light blue), and "Delete" (red).

62. Добавление связанных объектов

```
polls > admin.py > ...
1  from django.contrib import admin
2
3  from .models import Choice, Question
4
5  # Register your models here.
6
7  class QuestionAdmin(admin.ModelAdmin):
8      fieldsets = [
9          (None, {"fields": ["question_text"]}),
10         ("Date information", {"fields": ["pub_date"]}),
11     ]
12
13 admin.site.register(Choice)
```



63. Обновление

```
admin.py X
polls > admin.py > ...
1  from django.contrib import admin
2
3  from .models import Choice, Question
4
5  # Register your models here.
6
7  class ChoiceInline(admin.StackedInline):
8      model = Choice
9      extra = 3
10
11 class QuestionAdmin(admin.ModelAdmin):
12     fieldsets = [
13         (None, {"fields": ["question_text"]}),
14         ("Date information", {"fields": ["pub_date"], "classes": ["collapse"]}),
15     ]
16     inlines = [ChoiceInline]
17
18 admin.site.register(Question, QuestionAdmin)
```

Баранов Д.А. ИВТ 2.1

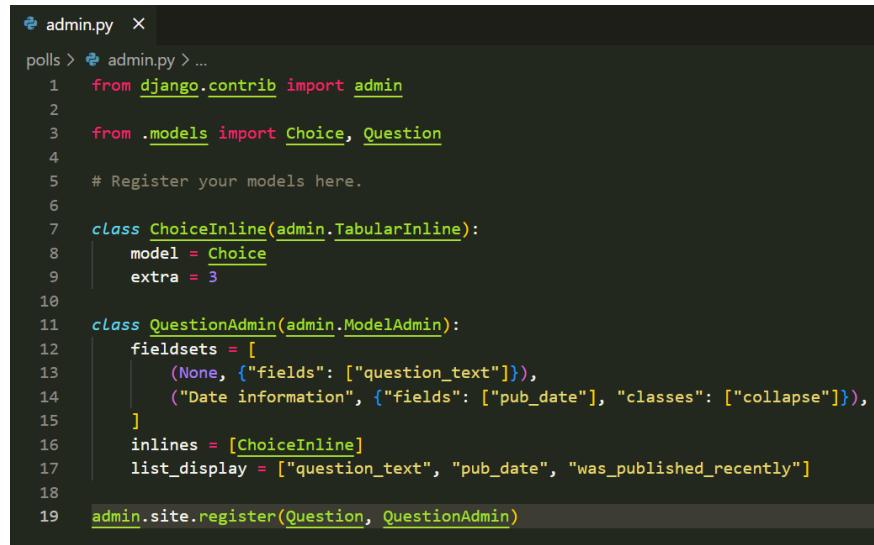
The screenshot shows the Django administration interface for adding a new poll question. On the left, there's a sidebar with 'AUTHENTICATION AND AUTHORIZATION' and 'POLLS' sections. The 'POLLS' section has a 'Questions' item with a '+ Add' button. The main area is titled 'Add question' and contains a 'Question text:' input field. Below it is a 'Date information' section. The next section is 'CHOICES', which contains three entries: 'Choice: #1', 'Choice: #2', and 'Choice: #3'. Each choice has a 'Choice text:' input field and a 'Votes:' input field set to '0'. There's also a link '+ Add another Choice'. At the bottom are 'SAVE', 'Save and add another', and 'Save and continue editing' buttons.

64. Смена на табличный формат

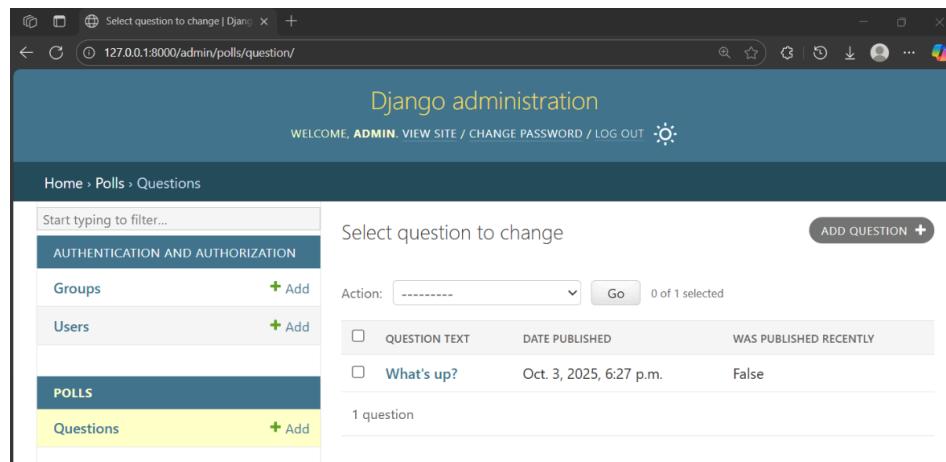
```
7     class ChoiceInline(admin.TabularInline):
```

This screenshot shows the same 'Add question' page after changing the inline form to a tabular format. The 'CHOICES' section now displays a table with columns for 'CHOICE TEXT', 'VOTES', and 'DELETE?'. Three rows are present, each with an empty 'CHOICE TEXT' field and a 'VOTES' field set to '0'. Each row has a delete icon ('X') in the 'DELETE?' column. A link '+ Add another Choice' is at the bottom. The other parts of the interface remain the same as in the first screenshot.

65. Настройка списка изменений администратора



```
admin.py
1 from django.contrib import admin
2
3 from .models import Choice, Question
4
5 # Register your models here.
6
7 class ChoiceInline(admin.TabularInline):
8     model = Choice
9     extra = 3
10
11 class QuestionAdmin(admin.ModelAdmin):
12     fieldsets = [
13         (None, {"fields": ["question_text"]}),
14         ("Date information", {"fields": ["pub_date"], "classes": ["collapse"]}),
15     ]
16     inlines = [ChoiceInline]
17     list_display = ["question_text", "pub_date", "was_published_recently"]
18
19 admin.site.register(Question, QuestionAdmin)
```

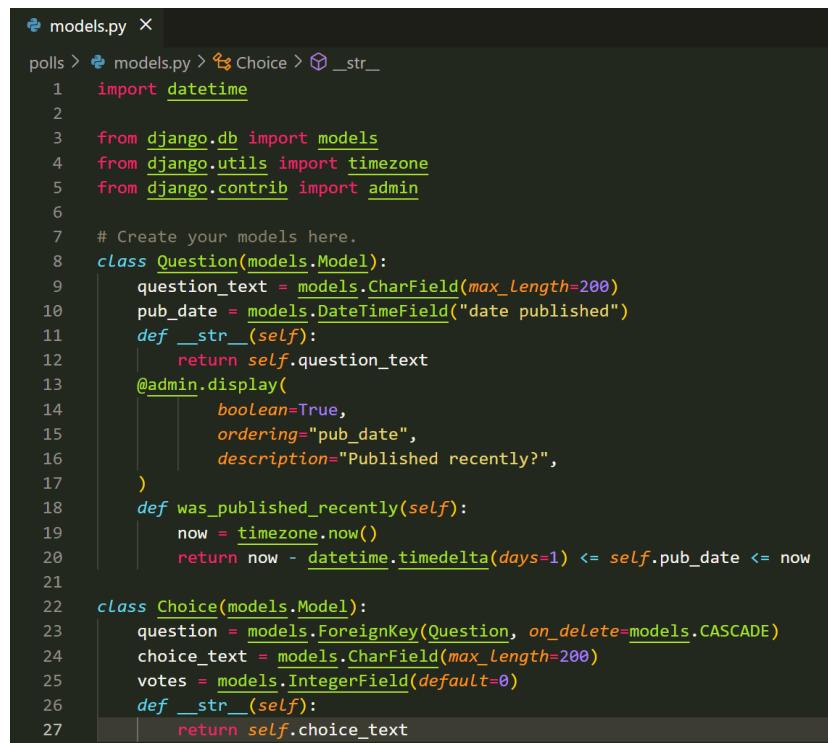


The screenshot shows the Django administration interface for the 'Questions' model. The left sidebar has 'Questions' highlighted in yellow. The main area displays a single question entry:

QUESTION TEXT	DATE PUBLISHED	WAS PUBLISHED RECENTLY
What's up?	Oct. 3, 2025, 6:27 p.m.	False

Below the table, it says '1 question'.

66. Использование декоратора



```
models.py
1 import datetime
2
3 from django.db import models
4 from django.utils import timezone
5 from django.contrib import admin
6
7 # Create your models here.
8 class Question(models.Model):
9     question_text = models.CharField(max_length=200)
10    pub_date = models.DateTimeField("date published")
11    def __str__(self):
12        return self.question_text
13    @admin.display(
14        boolean=True,
15        ordering="pub_date",
16        description="Published recently?",
17    )
18    def was_published_recently(self):
19        now = timezone.now()
20        return now - datetime.timedelta(days=1) <= self.pub_date <= now
21
22 class Choice(models.Model):
23     question = models.ForeignKey(Question, on_delete=models.CASCADE)
24     choice_text = models.CharField(max_length=200)
25     votes = models.IntegerField(default=0)
26     def __str__(self):
27         return self.choice_text
```

67. Добавление list_filter

```
admin.py < admin.py > ...
1  from django.contrib import admin
2
3  from .models import Choice, Question
4
5  # Register your models here.
6
7  class ChoiceInline(admin.TabularInline):
8      model = Choice
9      extra = 3
10
11 class QuestionAdmin(admin.ModelAdmin):
12     fieldsets = [
13         (None, {"fields": ["question_text"]}),
14         ("Date information", {"fields": ["pub_date"], "classes": ["collapse"]}),
15     ]
16     inlines = [ChoiceInline]
17     list_display = ["question_text", "pub_date", "was_published_recently"]
18     list_filter = ["pub_date"]
19
20 admin.site.register(Question, QuestionAdmin)
```

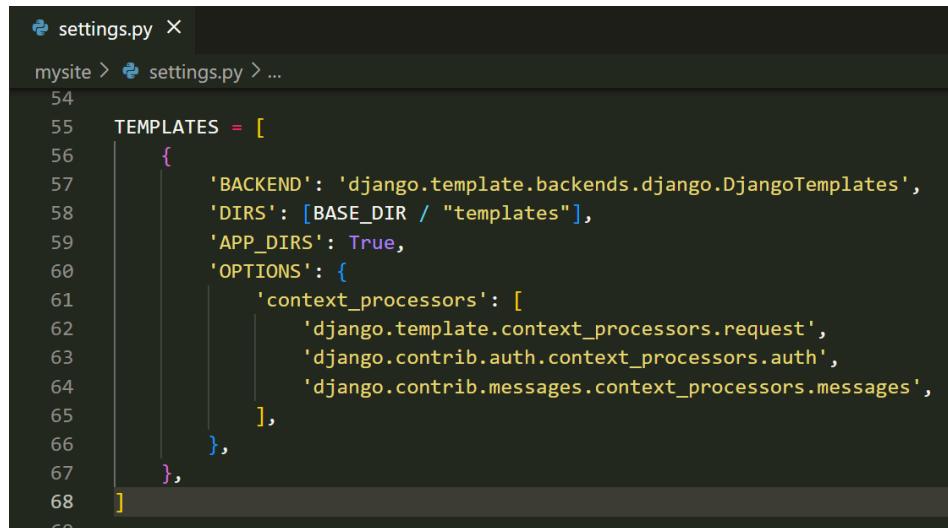
The screenshot shows the Django administration interface for the 'Questions' model. On the left, there's a sidebar with a search bar and links for 'Groups', 'Users', and 'POLLs'. Under 'POLLs', the 'Questions' model is selected. In the main content area, there's a title 'Select question to change', a 'FILTER' sidebar with 'Show counts' and date range options, and a table listing one question: 'What's up?' published on Oct. 3, 2025, at 6:27 p.m.

68. Добавление окна поиска в верхней части списка изменений

```
list_filter = [ pub_date ]
search_fields = ["question_text"]
```

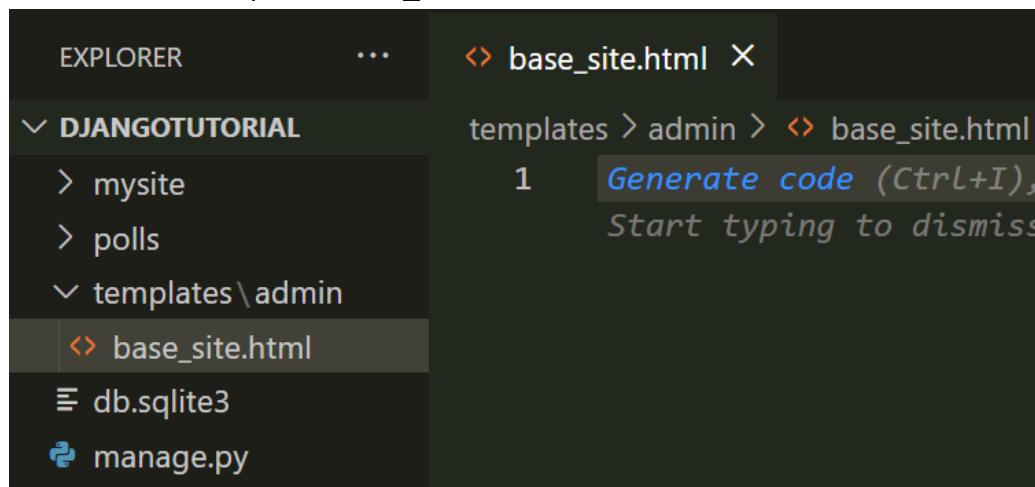
The screenshot shows the same Django administration interface as before, but with a search bar added to the top-left of the list view. The search bar contains a placeholder 'Start typing to filter...' and a 'Search' button. The rest of the interface remains the same, including the sidebar and the list of questions.

69. Настройка внешнего вида администратора
Настройка шаблонов проекта

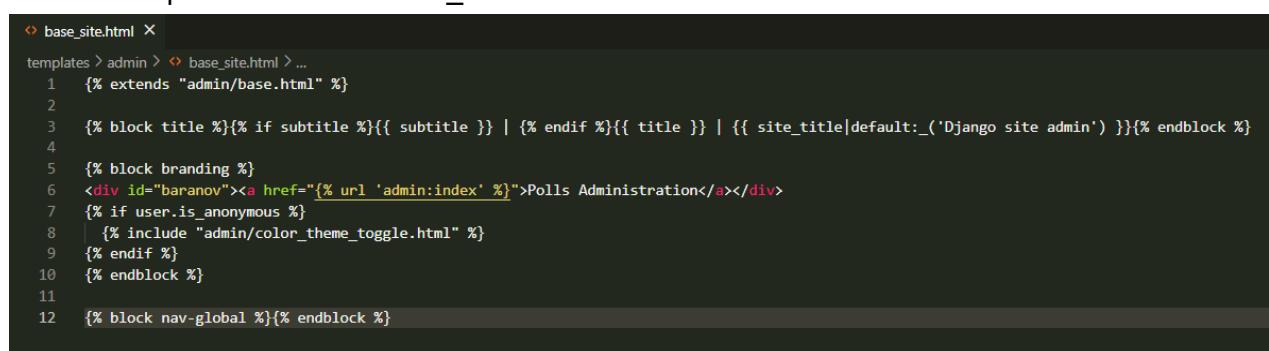


```
settings.py X
mysite > settings.py > ...
54
55     TEMPLATES = [
56         {
57             'BACKEND': 'django.template.backends.django.DjangoTemplates',
58             'DIRS': [BASE_DIR / "templates"],
59             'APP_DIRS': True,
60             'OPTIONS': {
61                 'context_processors': [
62                     'django.template.context_processors.request',
63                     'django.contrib.auth.context_processors.auth',
64                     'django.contrib.messages.context_processors.messages',
65                 ],
66             },
67         },
68     ]
69
```

70. Создание файла base_site.html

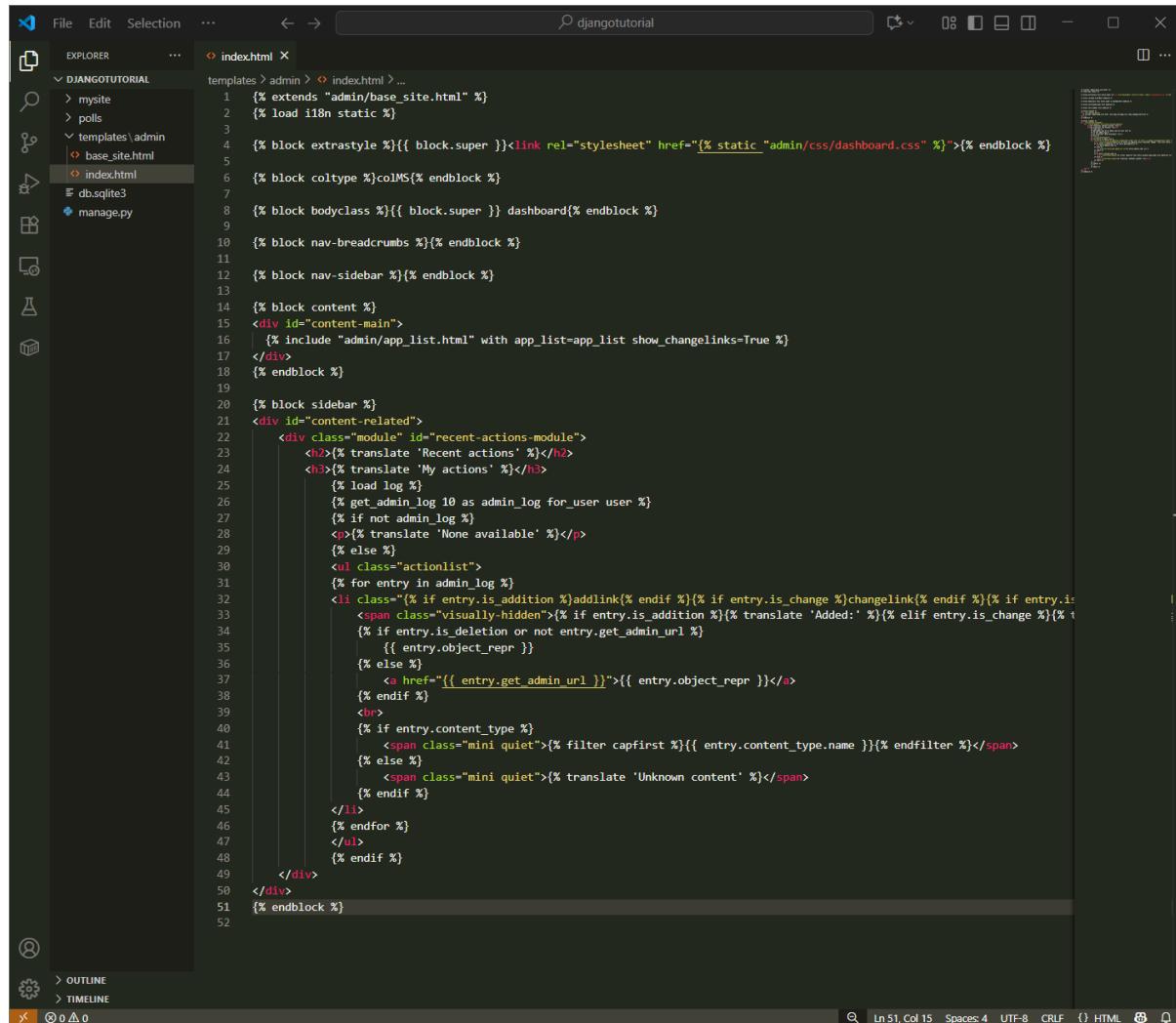


71. Настройка шаблона base_site.html



```
base_site.html X
templates > admin > base_site.html > ...
1  {% extends "admin/base.html" %}
2
3  {% block title %}{% if subtitle %}{{ subtitle }} | {% endif %}{{ title }} | {{ site_title|default:_('Django site admin') }}{% endblock %}
4
5  {% block branding %}
6      <div id="baranov"><a href="{% url 'admin:index' %}">Polls Administration</a></div>
7  {% if user.is_anonymous %}
8      | {% include "admin/color_theme_toggle.html" %}
9  {% endif %}
10  {% endblock %}
11
12  {% block nav-global %}{% endblock %}
```

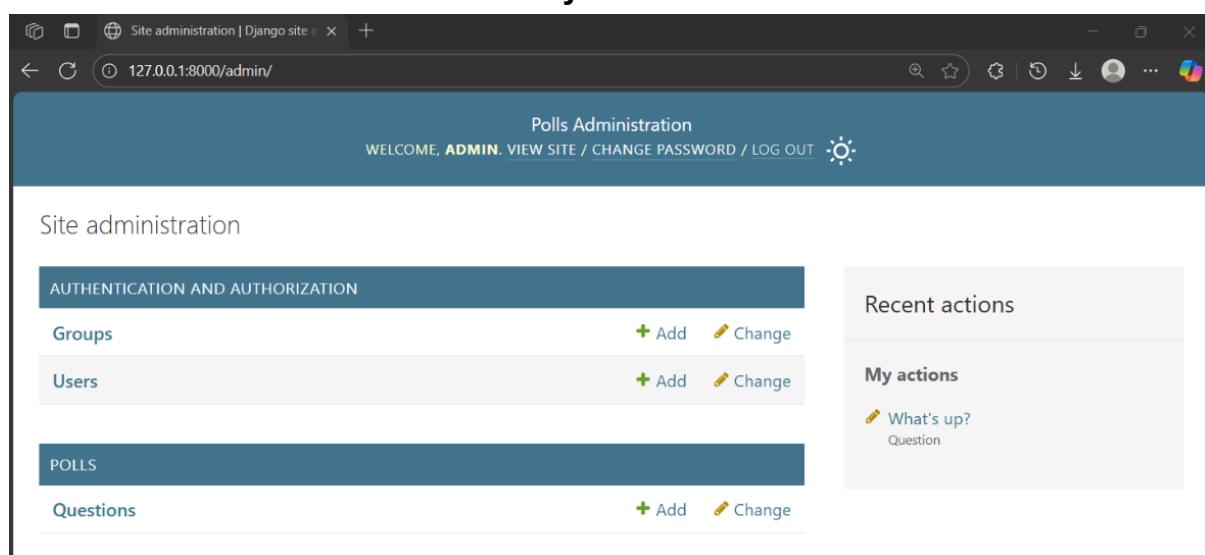
72. Настройка шаблона index.html



The screenshot shows the Visual Studio Code interface with the Django tutorial project open. The left sidebar shows the file structure under 'DJANGOTUTORIAL'. The main editor area displays the 'index.html' template. The template code includes various Django template tags like {% extends %}, {% block %}, and {% for %}. It also includes HTML code for a sidebar with recent actions and user management. The status bar at the bottom shows the file is 51 lines long, in column 15, with 4 spaces, in UTF-8 encoding, and has CRLF line endings.

```
File Edit Selection ... < > djangotutorial
EXPLORER
DJANGOTUTORIAL
  > mysite
  > polls
  > templates\admin
    > base_site.html
    > index.html
  db.sqlite3
  manage.py
index.html
templates\admin>index.html>...
1  {% extends "admin/base_site.html" %}
2  {% load i18n static %}
3
4  {% block extrastyle %}{% block.super %}<link rel="stylesheet" href="{% static "admin/css/dashboard.css" %}">{% endblock %}
5
6  {% block coltype %}colMS{% endblock %}
7
8  {% block bodyclass %}{% block.super %} dashboard{% endblock %}
9
10  {% block nav-breadcrumbs %}{% endblock %}
11
12  {% block nav-sidebar %}{% endblock %}
13
14  {% block content %}
15  <div id="content-main">
16  |  {% include "admin/app_list.html" with app_list=app_list show_changelinks=True %}
17  </div>
18  {% endblock %}
19
20  {% block sidebar %}
21  <div id="content-related">
22  <div class="module" id="recent-actions-module">
23  <h2>{% translate 'Recent actions' %}</h2>
24  <h3>{% translate 'My actions' %}</h3>
25  |  {% load log %}
26  |  {% get_admin_log 10 as admin_log for user user %}
27  |  {% if not admin_log %}
28  |  <p>{% translate 'None available' %}</p>
29  |  {% else %}
30  |  <ul class="actionlist">
31  |  {% for entry in admin_log %}
32  |  <li class="{"% if entry.is_addition %}addlink{"% endif %}"{"% if entry.is_change %}changalink{"% endif %}"{"% if entry.is_deletion %}deletelink{"% endif %}">
33  |  <span class="visually-hidden">{% if entry.is_addition %}{% translate 'Added:' %}{% elif entry.is_change %}{% translate 'Changed:' %}{% elif entry.is_deletion %}{% translate 'Deleted:' %}{% endif %}</span>
34  |  |  {{ entry.object_repr }}
35  |  |  {% else %}
36  |  |  <a href="{{ entry.get_admin_url }}">{{ entry.object_repr }}</a>
37  |  |  {% endif %}
38  |  |  <br>
39  |  |  {% if entry.content_type %}
40  |  |  <span class="mini quiet">{{ filter capfirst }}{{ entry.content_type.name }}{{ endfilter }}</span>
41  |  |  {% else %}
42  |  |  <span class="mini quiet">{% translate 'Unknown content' %}</span>
43  |  |  {% endif %}
44  |  |  {% endif %}
45  |  </li>
46  |  {% endfor %}
47  |  </ul>
48  |  {% endif %}
49  </div>
50  </div>
51  {% endblock %}
52
```

Результаты:



The screenshot shows a web browser displaying the Django admin site at '127.0.0.1:8000/admin/'. The top navigation bar says 'Polls Administration' and includes links for 'WELCOME, ADMIN.', 'VIEW SITE / CHANGE PASSWORD / LOG OUT'. The main content area is titled 'Site administration'. It features two main sections: 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'POLLS' (Questions). Below these are 'Recent actions' and 'My actions' sections, which both currently show a single entry: 'What's up? Question'.

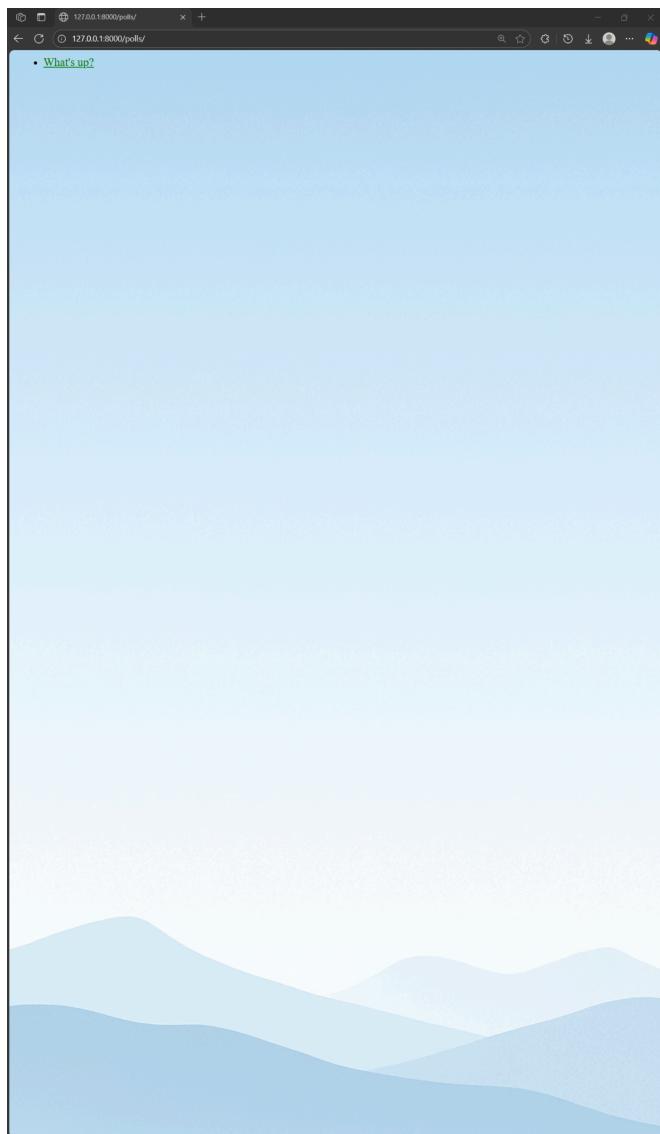
The screenshot displays two views of the Django Admin interface for a 'Polls' application.

Polls Administration (Top View):

- Left Sidebar:** Shows 'AUTHENTICATION AND AUTHORIZATION' sections for 'Groups' and 'Users', and a 'POLL' section for 'Questions'.
- Main Content:** A search bar and a table listing one poll question titled 'What's up?'. The table columns include 'QUESTION TEXT', 'DATE PUBLISHED', and 'PUBLISH' status.
- Right Sidebar:** Includes a 'FILTER' section with options like 'Show counts', 'By date published', and date ranges ('Any date', 'Today', 'Past 7 days', 'This month', 'This year').

Change Question View (Bottom View):

- Left Sidebar:** Same as the top view.
- Main Content:** A 'Change question' form for the poll titled 'What's up?'. It includes a 'Question text' field containing 'What's up?' and a 'Date information' section.
- Choices Section:** A table showing poll choices with their text, votes, and delete options. The choices are:
 - 'Not much' (7 votes)
 - 'The sky' (4 votes)
 - (empty choice) (0 votes)
 - (empty choice) (0 votes)
 - (empty choice) (0 votes)
- Buttons:** At the bottom are buttons for 'SAVE', 'Save and add another', 'Save and continue editing', and 'Delete'.



127.0.0.1:8000/polls/1/

What's up?

Not much
 The sky

Vote

