Баранов Д.А. ИВТ 2.1

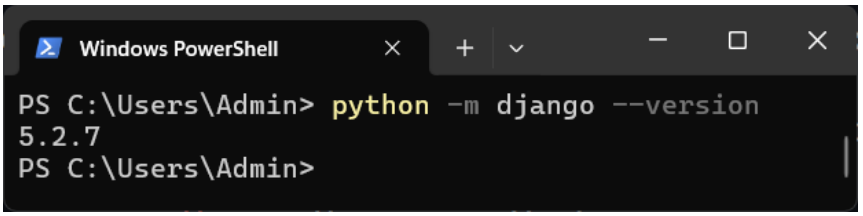
Введение в Django

Часть 1

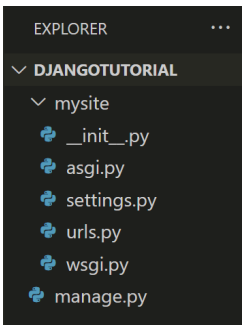
Создание проекта и первого приложения (“polls”), написание базовой view, подключение URL, запуск dev-сервера.

1. Установка django

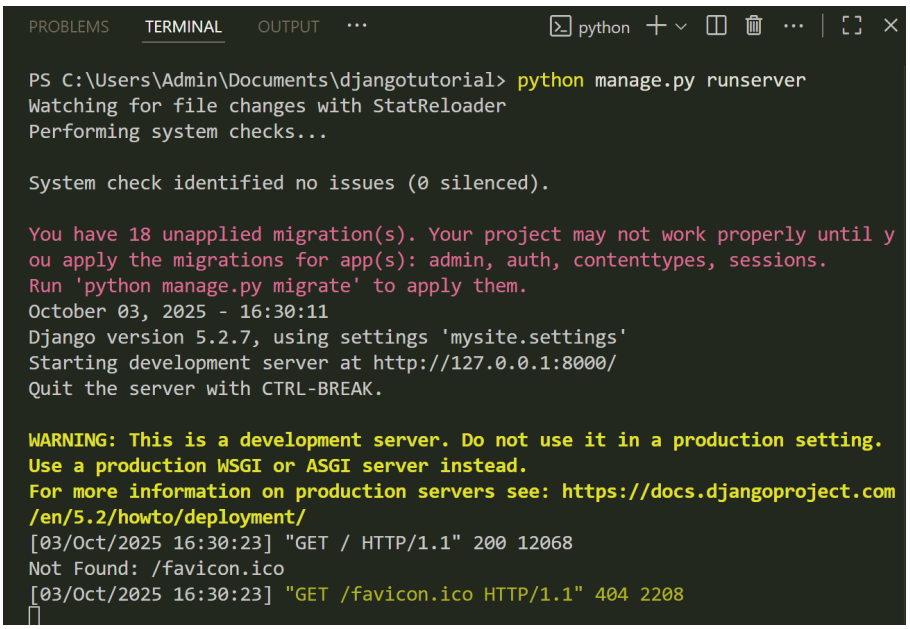
   python -m pip install Django



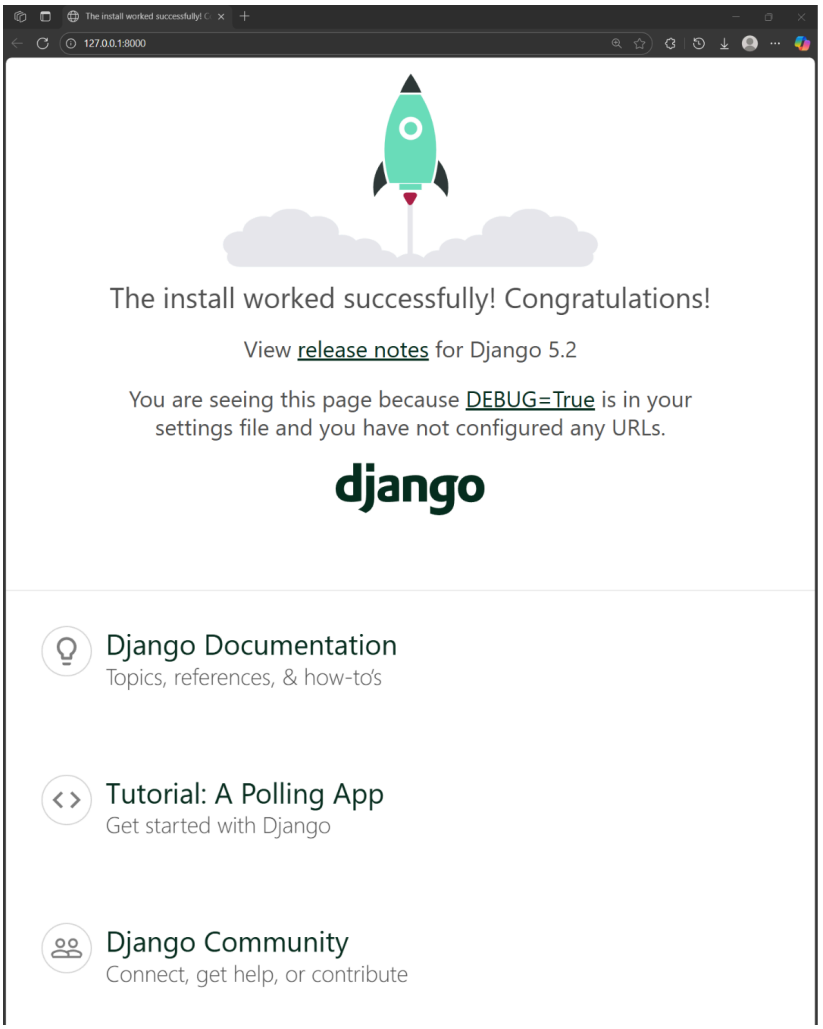
2. mkdir djangotutorial
3. `django-admin startproject mysite djangotutorial`
4. Получилось:



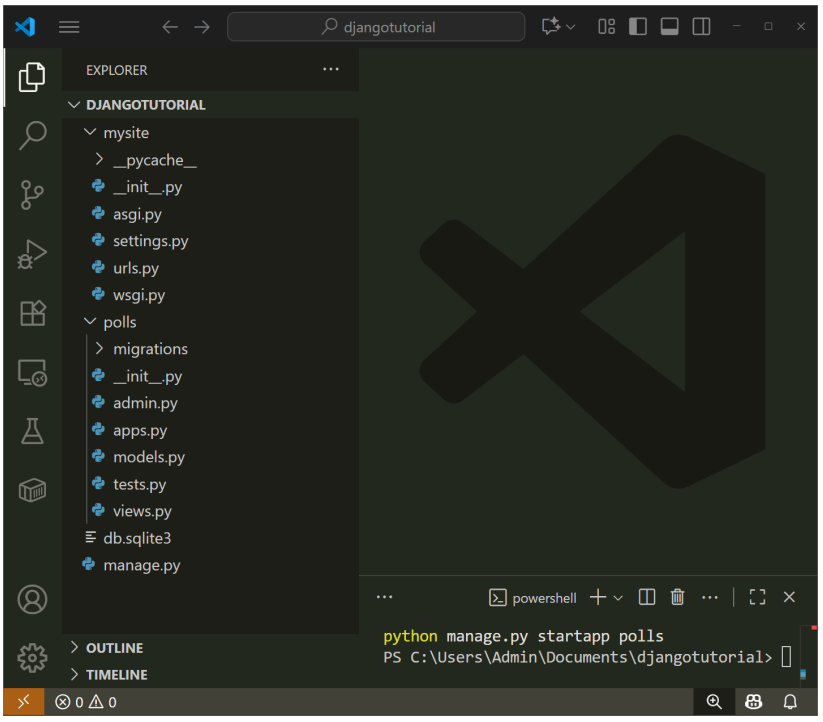
5. Сервер разработки



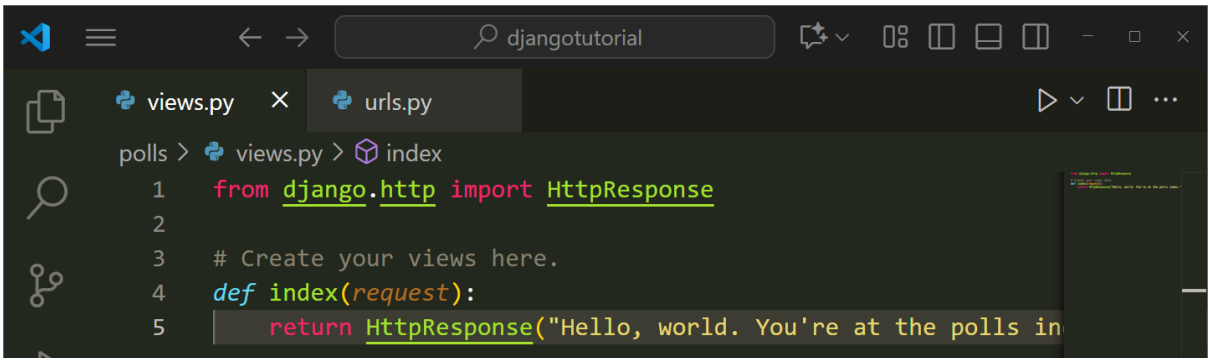
6. http://127.0.0.1:8000/



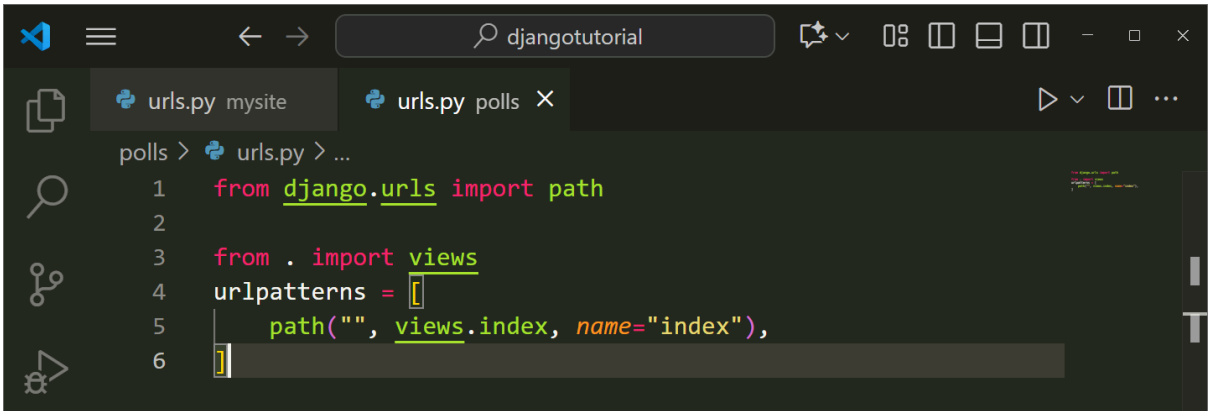
7. Создание приложения опросы



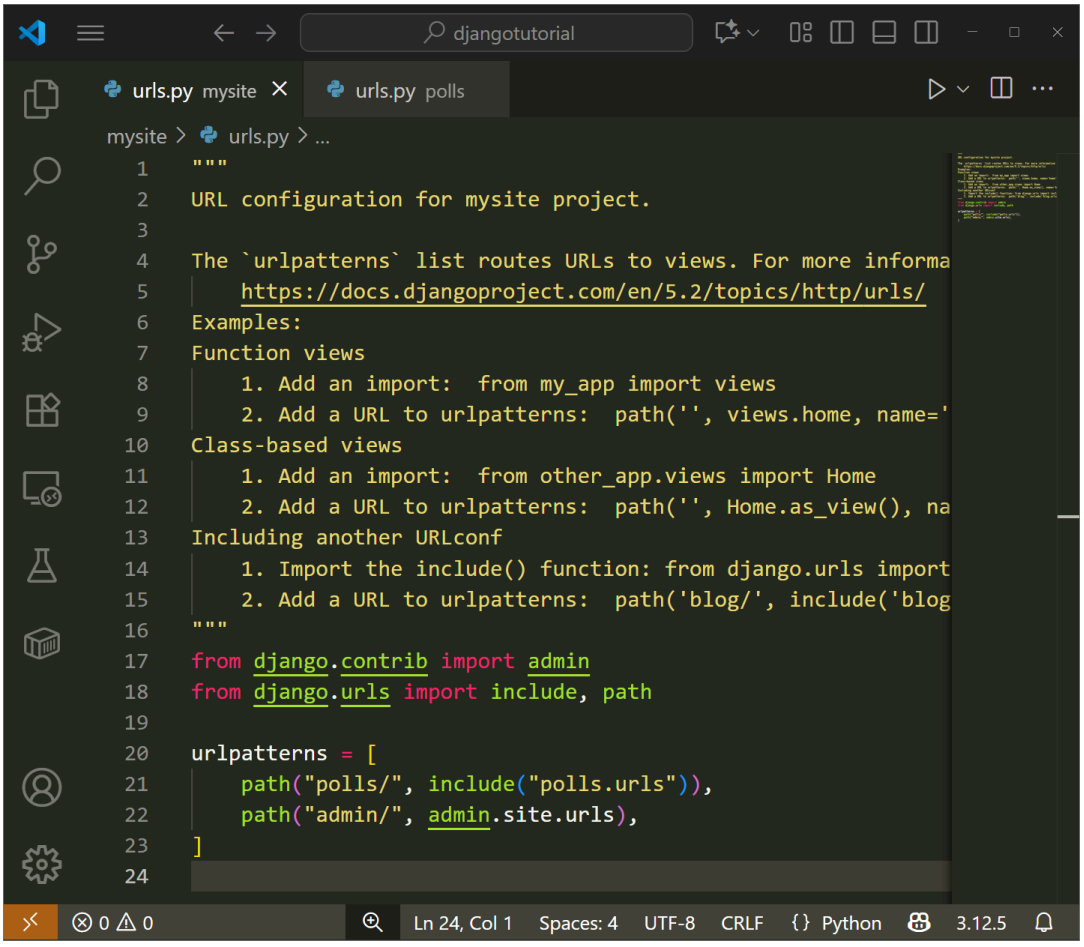
8. views.py



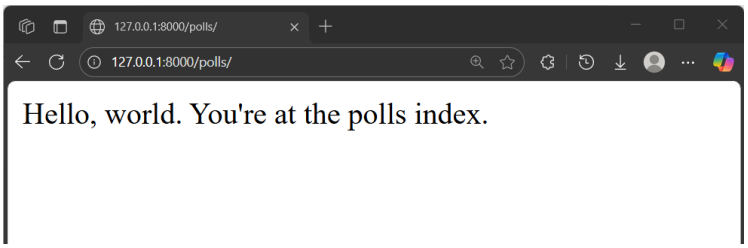
9. urls.py



10. urls.py



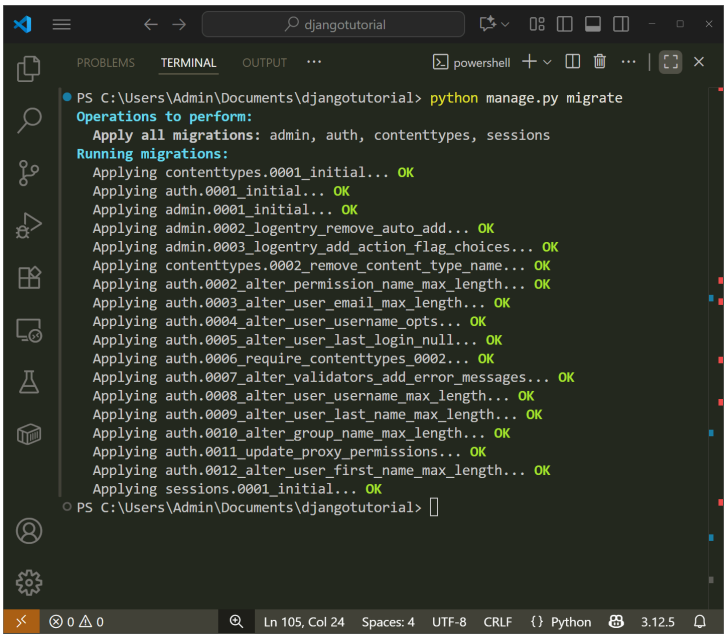
11. Результат `python manage.py runserver`



Часть 2

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

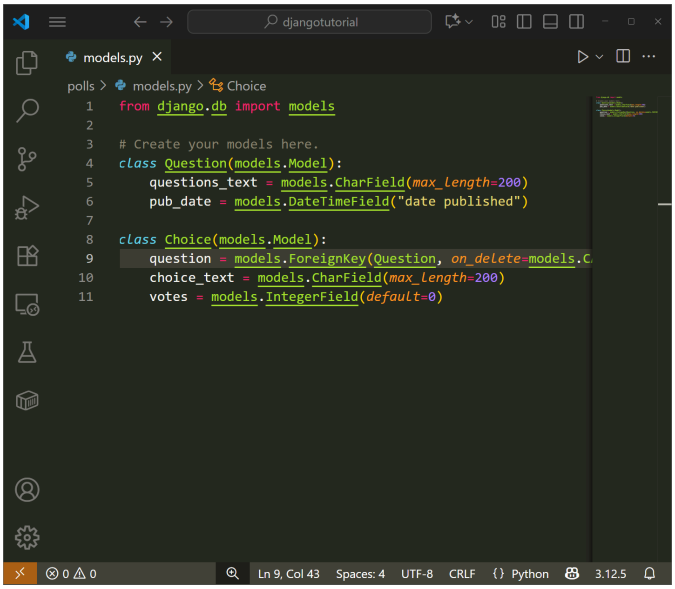
12. `python manage.py migrate`



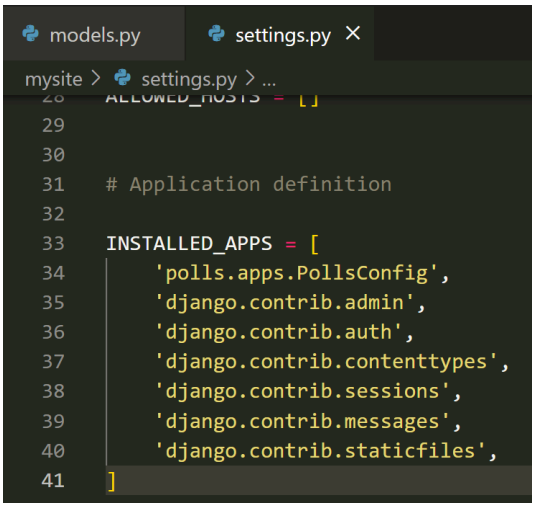
Баранов Д.А. ИВТ 2.1

Создание моделей

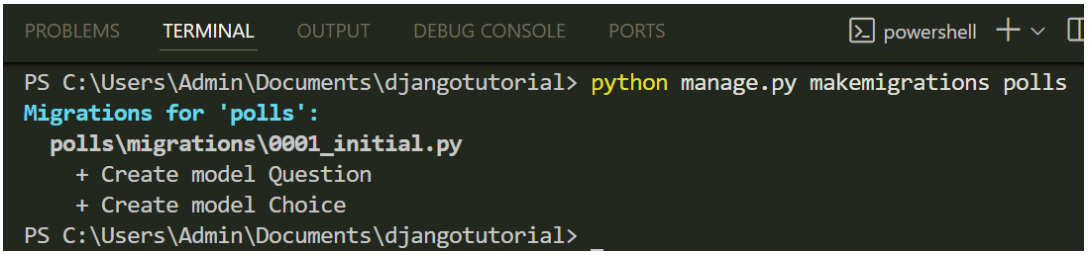
13. Модель



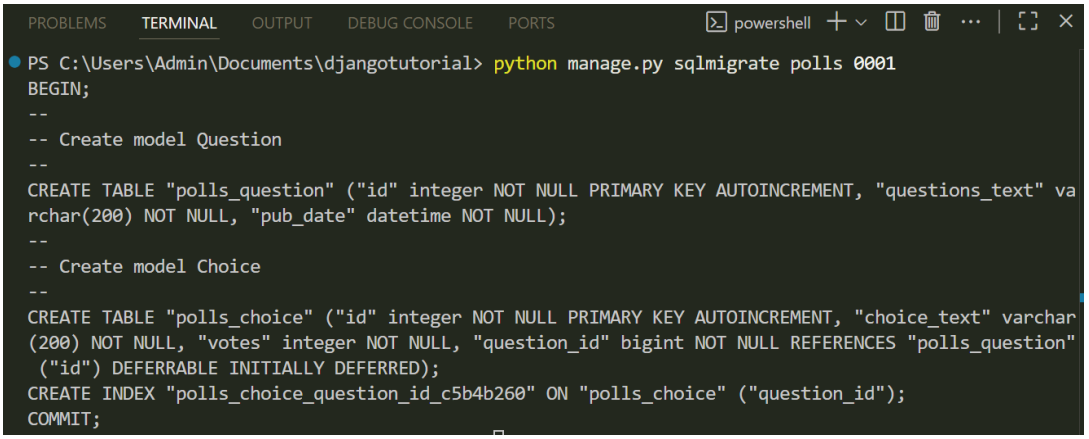
14. Добавление приложения в проект



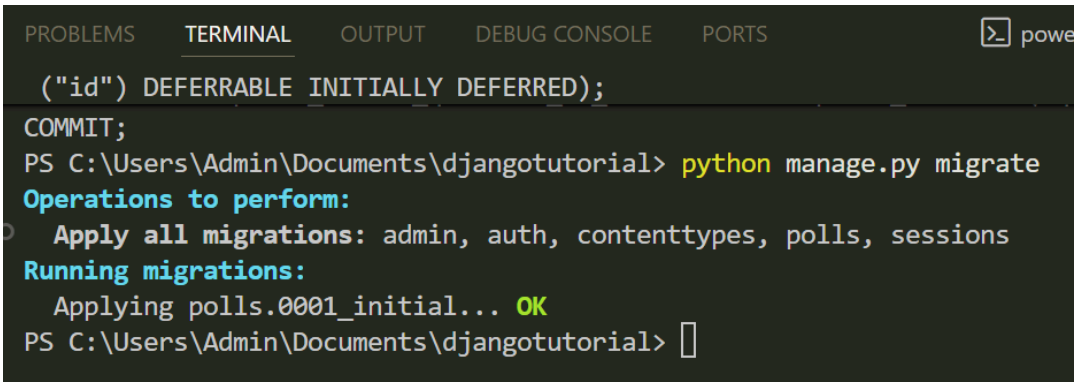
15. Миграция базы данных



16. sqlmigrate



17. Синхронизация изменений



API

18. Модели в консоли

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS                python

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

models.py

polls > models.py > Choice > __str__

```python
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")
    def __str__(self):
        return self.question_text

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

20. Добавление пользовательского метода

models.py

polls > models.py > Choice > __str__

```python
import datetime

from django.db import models
from django.utils import timezone

# Create your models here.
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField("date published")
    def __str__(self):
        return self.question_text
    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)

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

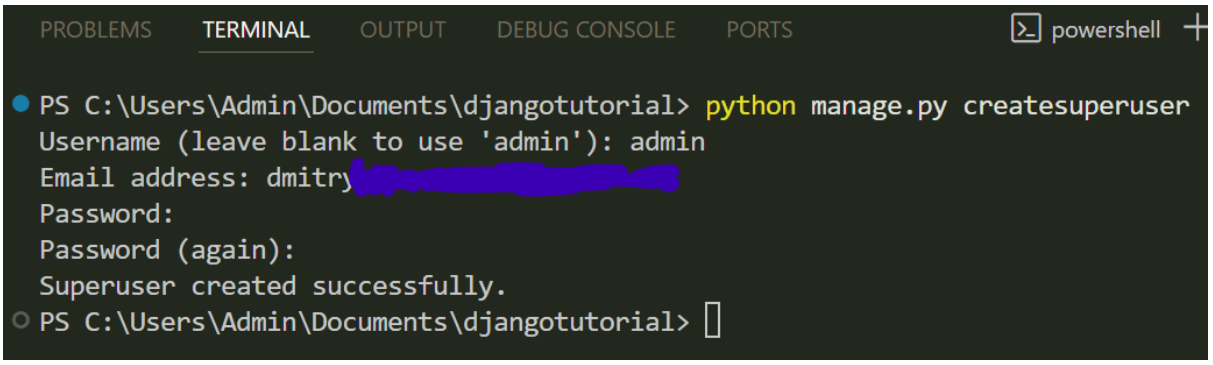
21. В консоли

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS                    python

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})
>>>
```

Баранов Д.А. ИВТ 2.1

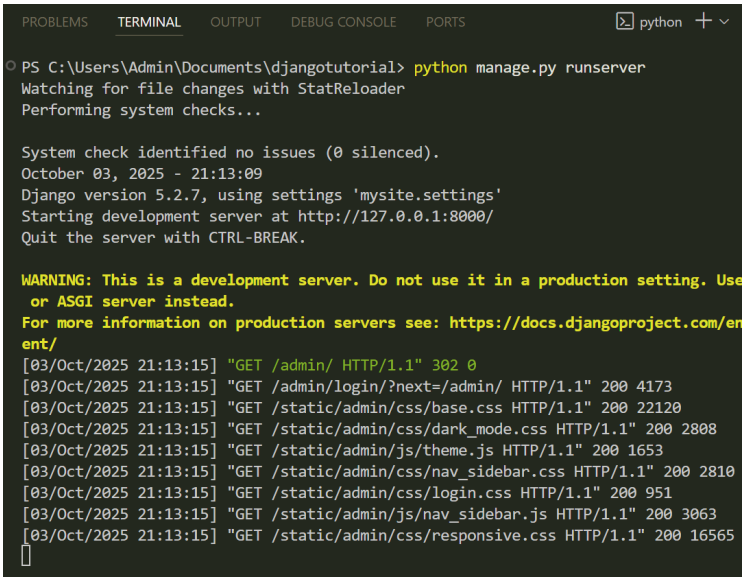
Django Admin

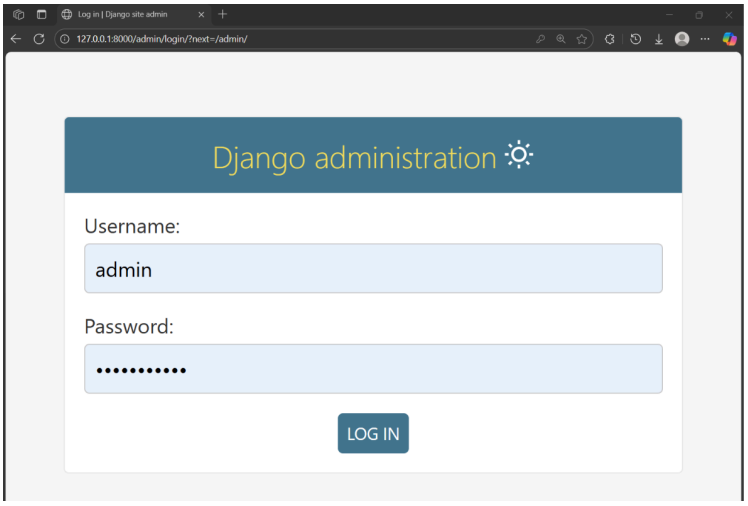
22. Создание пользователя с правами администратора



Password: djangoav7mi

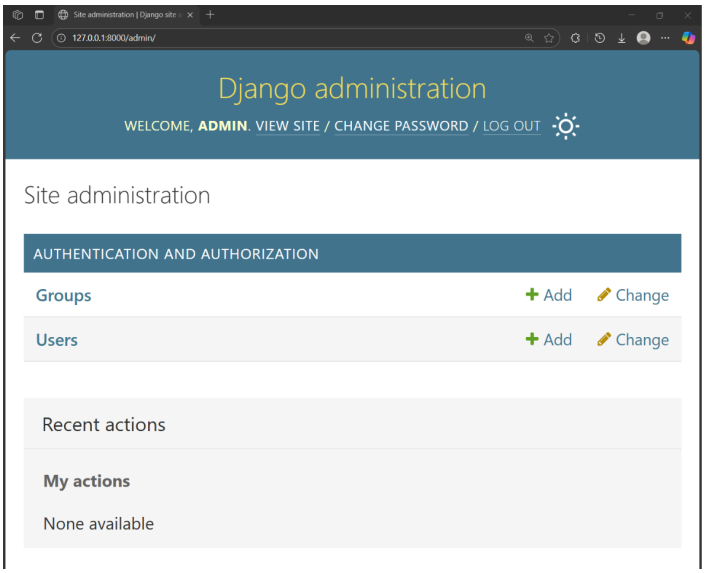
23. Запуск сервера разработки



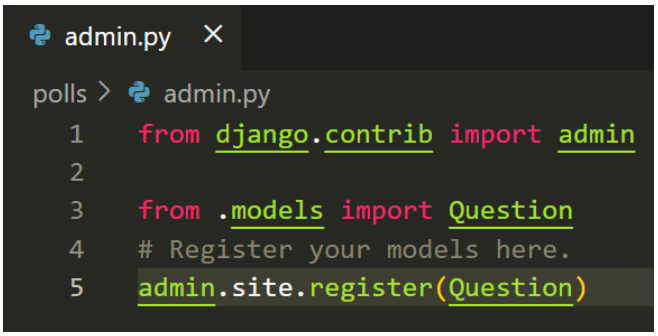


Баранов Д.А. ИВТ 2.1

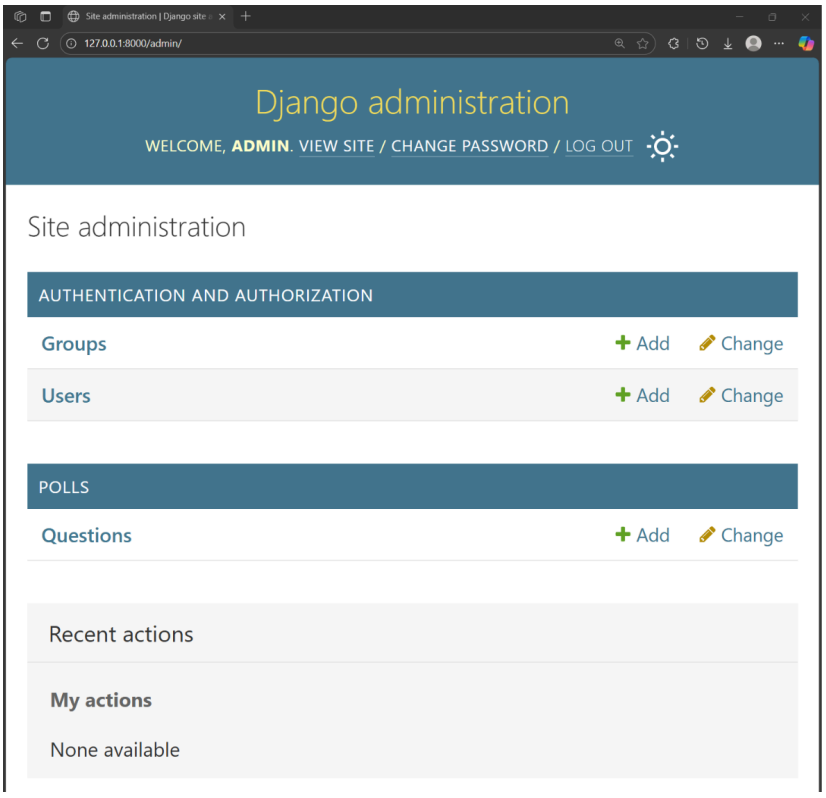
24. Сайт администратора



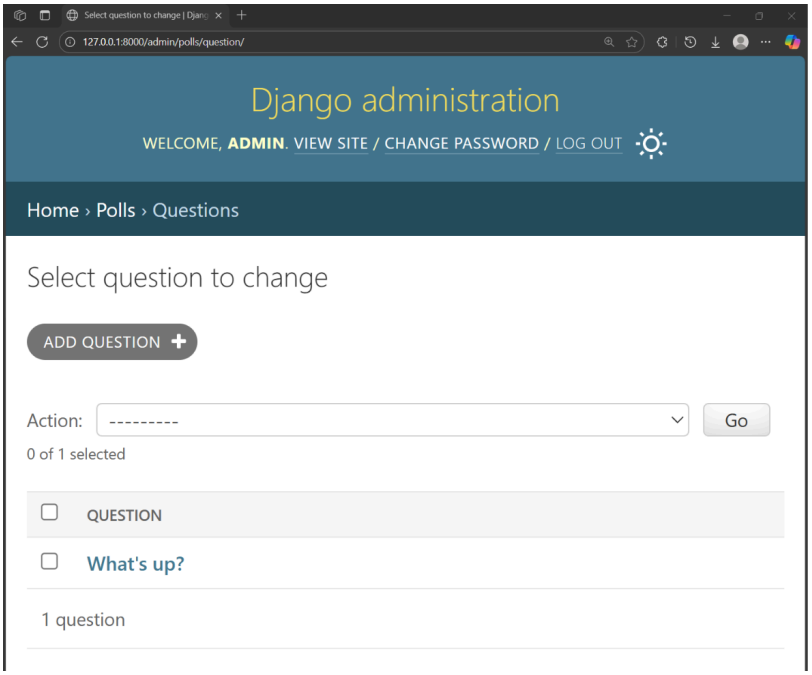
25. Приложение для опросов изменяемое в админке

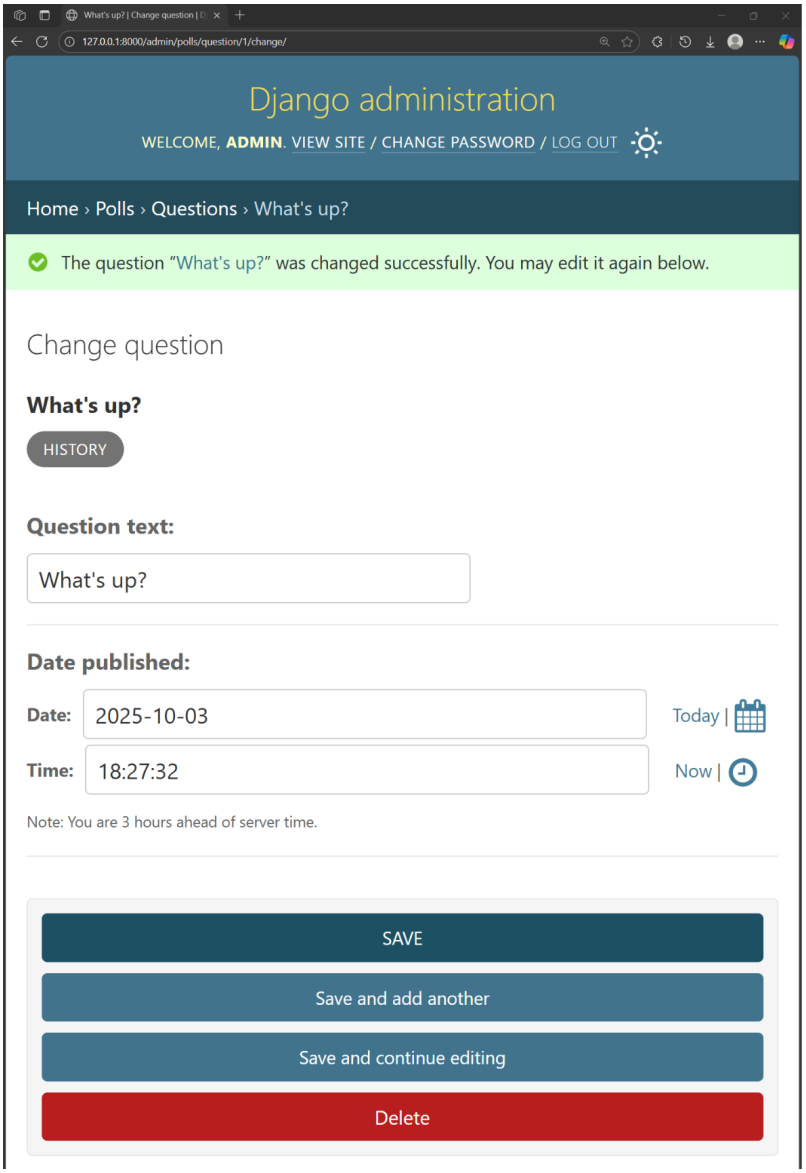


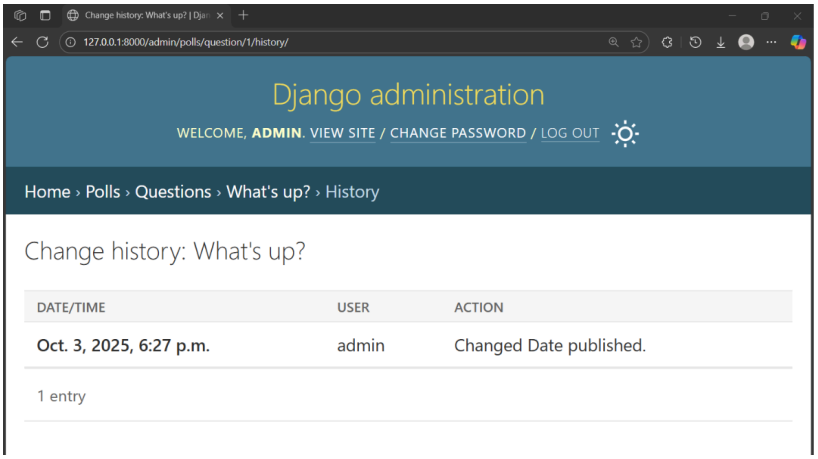
26. Результат



Баранов Д.А. ИВТ 2.1







Часть 3

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

27. views

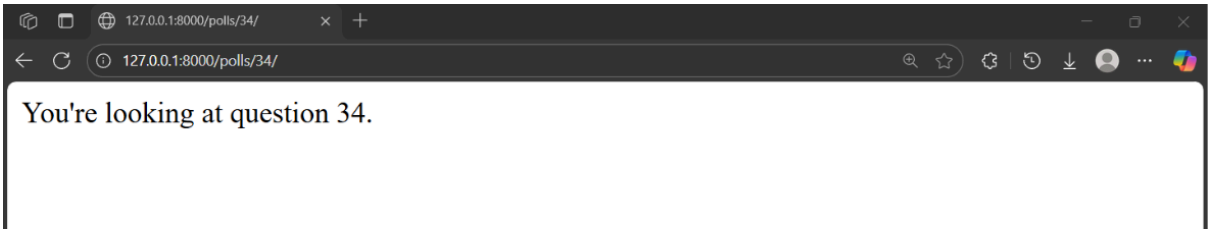
views.py

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

28. Подключение новых представлений к модулю

urls.py

```
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

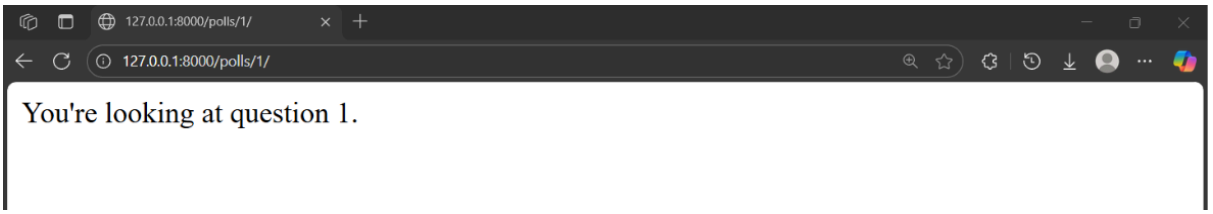
```python
from django.http import HttpResponse

from .models import Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    output = ", ".join([q.question_text for q in latest_question_list])
    return HttpResponse(output)

def detail(request, question_id):
    return HttpResponse("You're looking at question %s." % question_id)

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)
```

30. Шаблон

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a>
</li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

31. Загрузка шаблона

```python
from django.http import HttpResponse
from django.template import loader

from .models import Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    template = loader.get_template("polls/index.html")
    context = {"latest_question_list": latest_question_list}
    return HttpResponse(template.render(context, request))

def detail(request, question_id):
    return HttpResponse("You're looking at question %s." % question_id)

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)
```



32. Использование render

views.py

polls > views.py > vote

```python
from django.http import HttpResponse
from django.shortcuts import render

from .models import Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    context = {"latest_question_list": latest_question_list}
    return render(request, "polls/index.html", context)

def detail(request, question_id):
    return HttpResponse("You're looking at question %s." % question_id)

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)
```

33. Сообщение об ошибке 404

views.py

polls > views.py > vote

```python
from django.http import HttpResponse
from django.http import Http404
from django.shortcuts import render

from .models import Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    context = {"latest_question_list": latest_question_list}
    return render(request, "polls/index.html", context)

def detail(request, question_id):
    try:
        question = Question.objects.get(pk=question_id)
    except Question.DoesNotExist:
        raise Http404("Question does not exist")
    return render(request, "polls/detail.html", {"question": question})

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)
```

34. Шаблон

detail.html

```
{{ question }}
```

35. views

views.py

```python
from django.shortcuts import get_object_or_404, render

from .models import Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    context = {"latest_question_list": latest_question_list}
    return render(request, "polls/index.html", context)

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

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    return HttpResponse("You're voting on question %s." % question_id)
```

36. Система шаблонов

detail.html

```
<h1>{{ question.question_text }}</h1>
<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }}</li>
{% endfor %}
</ul>
```

37. Удаление жестко заданных URL в шаблонах

index.html

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="{% url 'detail' question.id %}">{{ question.question_text }}</a>
</li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

urls.py

```python
from django.urls import path

from . import views
urlpatterns = [
    path("", views.index, name="index"),
    path("specifics/<int:question_id>/", views.detail, name="detail"),
    path("<int:question_id>/results/", views.results, name="results"),
    path("<int:question_id>/vote/", views.vote, name="votes"),
]
```

38. Пространство имен URL-адресов

urls.py

```python
from django.urls import path

from . import views
app_name = "polls"
urlpatterns = [
    path("", views.index, name="index"),
    path("<int:question_id>/", views.detail, name="detail"),
    path("<int:question_id>/results/", views.results, name="results"),
    path("<int:question_id>/vote/", views.vote, name="vote"),
]
```

39. Навести указатель мыши на детальный вид в пространстве имен

index.html

```html
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}
        </a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

Часть 4

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

40. Обновление шаблона

detail.html

```html
<form action="{% url 'polls:vote' question.id %}" method="post">
{% csrf_token %}
<fieldset>
    <legend><h1>{{ question.question_text }}</h1></legend>
    {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
    {% for choice in question.choice_set.all %}
        <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}">
        <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br>
    {% endfor %}
</fieldset>
<input type="submit" value="Vote">
</form>
```

41. Обновление views

views.py

```python
from django.db.models import F
from django.http import HttpResponse, HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse

from .models import Choice, Question

# Create your views here.
def index(request):
    latest_question_list = Question.objects.order_by("-pub_date")[:5]
    context = {"latest_question_list": latest_question_list}
    return render(request, "polls/index.html", context)

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

def results(request, question_id):
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST["choice"])
    except (KeyError, Choice.DoesNotExist):
        return render(
            request,
            "polls/detail.html",
            {
                "question": question,
                "error_message": "You didn't select a choice.",
            },
        )
    else:
        selected_choice.votes = F("votes") + 1
        selected_choice.save()
        return HttpResponseRedirect(reverse("polls:results", args=(question.id,)))
```

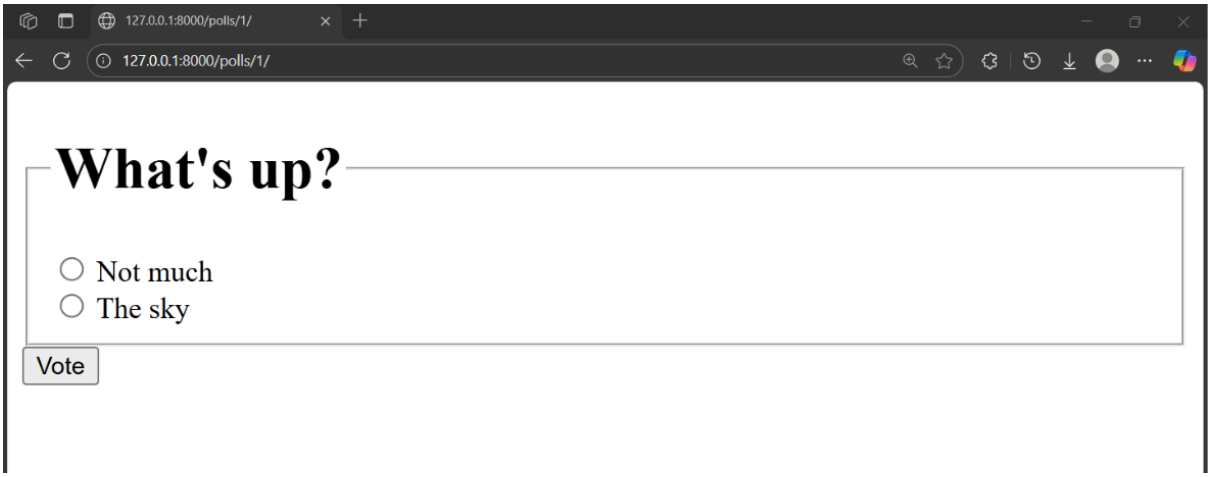
42. Представление перенаправляет на результаты для вопроса

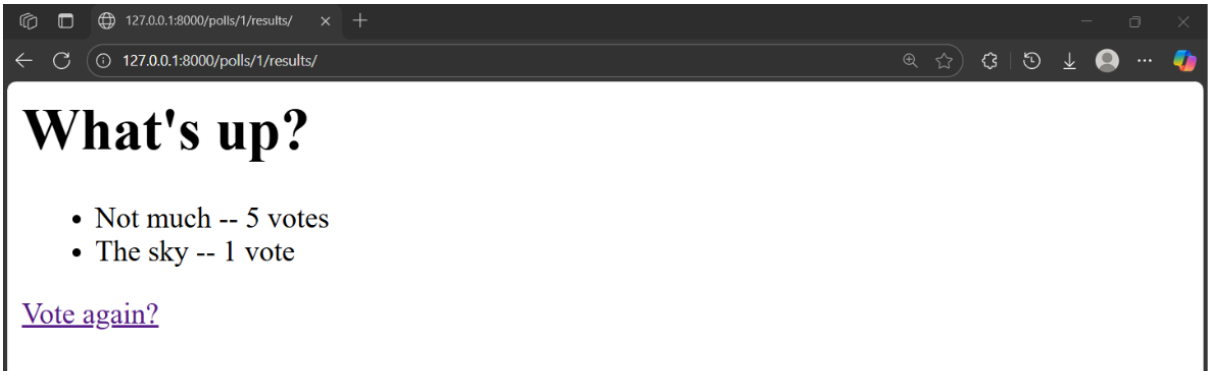
```python
def results(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    return render(request, "polls/results.html", {"question": question})
```

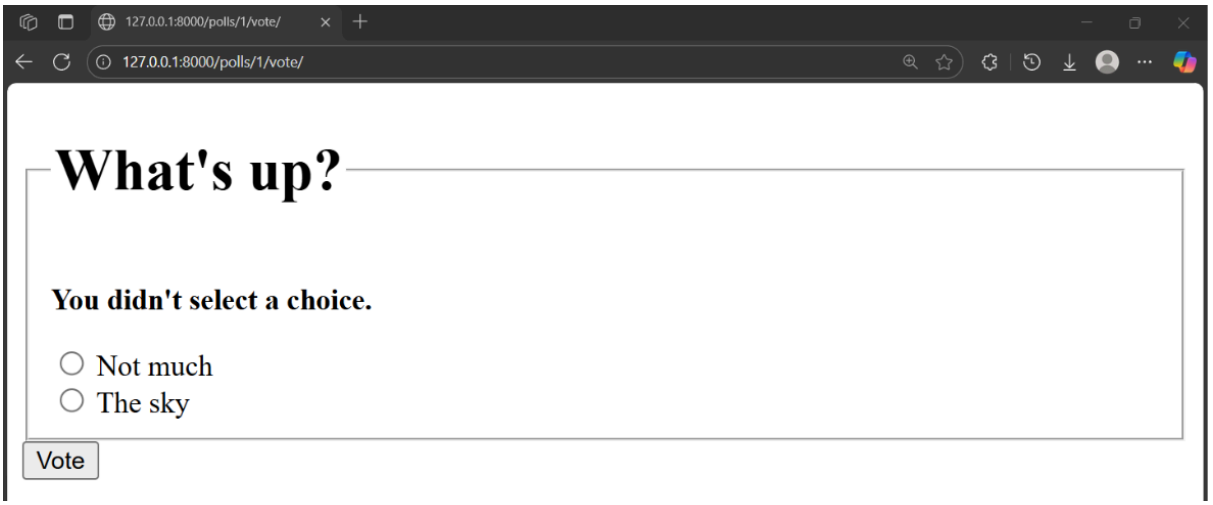
43. results.html

results.html

```html
<h1>{{ question.question_text }}</h1>

<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{ choice.votes|pluralize }}</li>
{% endfor %}
</ul>
<a href="{% url 'polls:detail' question.id %}">Vote again?</a>
```







44. Изменение URLconf

urls.py

polls > urls.py > ...

```python
from django.urls import path

from . import views

app_name = "polls"
urlpatterns = [
    path("", views.IndexView.as_view(), name="index"),
    path("<int:pk>/", views.DetailView.as_view(), name="detail"),
    path("<int:pk>/results/", views.ResultsView.as_view(), name="results"),
    path("<int:question_id>/vote/", views.vote, name="vote"),
]
```

45. Изменение представлений

views.py

polls > views.py > vote

```python
from django.db.models import F
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse
from django.views import generic

from .models import Choice, Question

# Create your views here.
class IndexView(generic.ListView):
    template_name = "polls/index.html"
    context_object_name = "latest_question_list"

    def get_queryset(self):
        """Return the last five published questions."""
        return Question.objects.order_by("-pub_date")[:5]

class DetailView(generic.DetailView):
    model = Question
    template_name = "polls/detail.html"

class ResultsView(generic.DetailView):
    model = Question
    template_name = "polls/results.html"

def vote(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST["choice"])
    except (KeyError, Choice.DoesNotExist):
        return render(
            request,
            "polls/detail.html",
            {
                "question": question,
                "error_message": "You didn't select a choice.",
            },
        )
    else:
        selected_choice.votes = F("votes") + 1
        selected_choice.save()
        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\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)
>>> 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

polls > tests.py > QuestionModelTests > test_was_published_recently_with_future_question

```python
import datetime

from django.test import TestCase
from django.utils import timezone

from .models import Question

# Create your tests here.
class QuestionModelTests(TestCase):
    def test_was_published_recently_with_future_question(self):
        """
        was_published_recently() returns False for questions whose pub_date is in the future.
        """
        time = timezone.now() + datetime.timedelta(days=30)
        future_question = Question(pub_date=time)
        self.assertIs(future_question.was_published_recently(), False)
```

48. Проведение тестов

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powershell

```
PS C:\Users\Admin\Documents\djangotutorial> 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\djangotutorial\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\djangotutorial>
```

49. Исправление ошибки

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

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS powersh

```
PS C:\Users\Admin\Documents\djangotutorial> 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\djangotutorial>
```

50. Более комплексные испытания

tests.py

polls > tests.py > QuestionModelTests > test_was_published_recently_with_recent_question

```python
import datetime

from django.test import TestCase
from django.utils import timezone

from .models import Question

# Create your tests here.
class QuestionModelTests(TestCase):
    def test_was_published_recently_with_future_question(self):
        """
        was_published_recently() returns False for questions whose pub_date is in the future.
        """
        time = timezone.now() + datetime.timedelta(days=30)
        future_question = Question(pub_date=time)
        self.assertIs(future_question.was_published_recently(), False)

    def test_was_published_recently_with_old_question(self):
        """
        was_published_recently() returns False for questions whose pub_date is older than 1 day.
        """
        time = timezone.now() - datetime.timedelta(days=1, seconds=1)
        old_question = Question(pub_date=time)
        self.assertIs(old_question.was_published_recently(), False)

    def test_was_published_recently_with_recent_question(self):
        """
        was_published_recently() returns True for questions whose pub_date is within the last day.
        """
        time = timezone.now() - datetime.timedelta(hours=23, minutes=59, seconds=59)
        recent_question = Question(pub_date=time)
        self.assertIs(recent_question.was_published_recently(), True)
```

51. Тестовый клиент Django

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS

```
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)
>>> 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&#x27;s up?\n        </a></li>\n    \n    </ul>\n'
>>> response.context["latest_question_list"]
<QuerySet [<Question: What's up?>]>
>>>
```

Баранов Д.А. ИВТ 2.1

52. Улучшение view get_queryset(self)

views.py

polls > views.py > DetailView

```python
from django.db.models import F
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse
from django.views import generic
from django.utils import timezone

from .models import Choice, Question

# Create your views here.
class IndexView(generic.ListView):
    template_name = "polls/index.html"
    context_object_name = "latest_question_list"

    def get_queryset(self):
        """
        Return the last five published questions (not including those set to be published in the future).
        """
        return Question.objects.filter(pub_date__lte=timezone.now()).order_by("-pub_date")[:5]
```

53. Тестирование нового представления

tests.py

polls > tests.py > QuestionIndexViewTests > test_two_past_questions

```python
        self.assertIs(recent_question.was_published_recently(), True)

def create_question(question_text, days):
    """
    Create a question with the given `question_text` and published the
    given number of `days` offset to now (negative for questions published
    in the past, positive for questions that have yet to be published).
    """
    time = timezone.now() + datetime.timedelta(days=days)
    return Question.objects.create(question_text=question_text, pub_date=time)

class QuestionIndexViewTests(TestCase):
    def test_no_questions(self):
        """
        If no questions exist, an appropriate message is displayed.
        """
        response = self.client.get(reverse("polls:index"))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "No polls are available.")
        self.assertQuerySetEqual(response.context["latest_question_list"], [])

    def test_past_question(self):
        """
        Questions with a pub_date in the past are displayed on the
        index page.
        """
        question = create_question(question_text="Past question.", days=-30)
        response = self.client.get(reverse("polls:index"))
        self.assertQuerySetEqual(
            response.context["latest_question_list"],
            [question],
        )

    def test_future_question(self):
        """
        Questions with a pub_date in the future aren't displayed on
        the index page.
        """
        create_question(question_text="Future question.", days=30)
        response = self.client.get(reverse("polls:index"))
        self.assertContains(response, "No polls are available.")
        self.assertQuerySetEqual(response.context["latest_question_list"], [])

    def test_future_question_and_past_question(self):
        """
        Even if both past and future questions exist, only past questions
        are displayed.
        """
        question = create_question(question_text="Past question.", days=-30)
        create_question(question_text="Future question.", days=30)
        response = self.client.get(reverse("polls:index"))
        self.assertQuerySetEqual(
            response.context["latest_question_list"],
            [question],
        )

    def test_two_past_questions(self):
        """
        The questions index page may display multiple questions.
        """
        question1 = create_question(question_text="Past question 1.", days=-30)
        question2 = create_question(question_text="Past question 2.", days=-5)
        response = self.client.get(reverse("polls:index"))
        self.assertQuerySetEqual(
            response.context["latest_question_list"],
            [question2, question1],
        )
```

54. Тестирование DetailView

```python
class DetailView(generic.DetailView):
    model = Question
    template_name = "polls/detail.html"

    def get_queryset(self):
        """
        Excludes any questions that aren't published yet.
        """
        return Question.objects.filter(pub_date__lte=timezone.now())
```

55. Тесты

tests.py

polls > tests.py > QuestionDetailViewTests > test_past_question

```python
class QuestionDetailViewTests(TestCase):
    def test_future_question(self):
        """
        The detail view of a question with a pub_date in the future
        returns a 404 not found.
        """
        future_question = create_question(question_text="Future question.", days=5)
        url = reverse("polls:detail", args=(future_question.id,))
        response = self.client.get(url)
        self.assertEqual(response.status_code, 404)

    def test_past_question(self):
        """
        The detail view of a question with a pub_date in the past
        displays the question's text.
        """
        past_question = create_question(question_text="Past Question.", days=-5)
        url = reverse("polls:detail", args=(past_question.id,))
        response = self.client.get(url)
        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'...
PS C:\Users\Admin\Documents\djangotutorial>
```

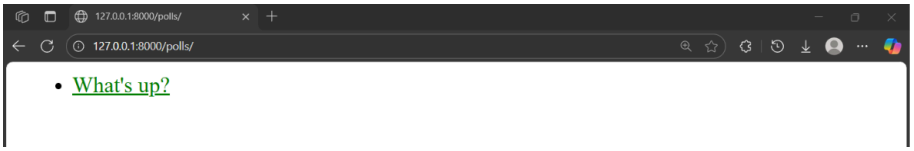
Часть 6

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

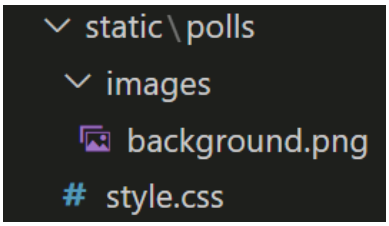
57. Настройка внешнего вида приложения

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

```
# style.css   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 }}
9           </a></li>
10      {% endfor %}
11      </ul>
12  {% else %}
13      <p>No polls are available.</p>
14  {% endif %}
```

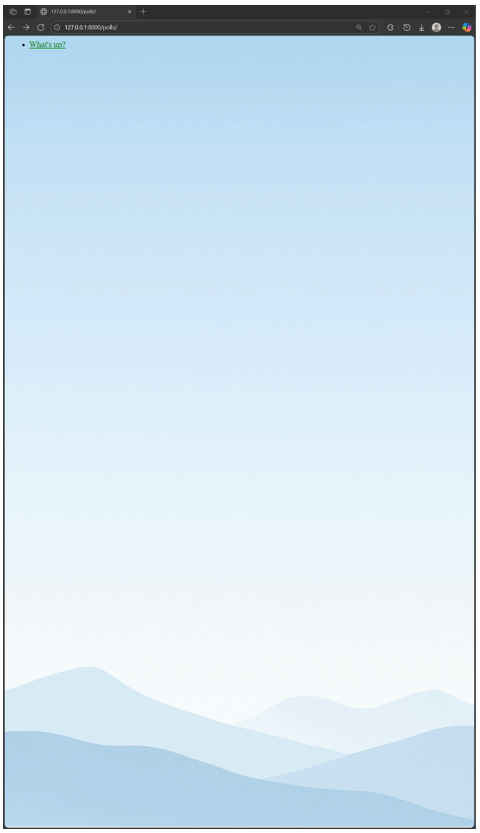


58. Добавление фонового изображения




```
# style.css  ×
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

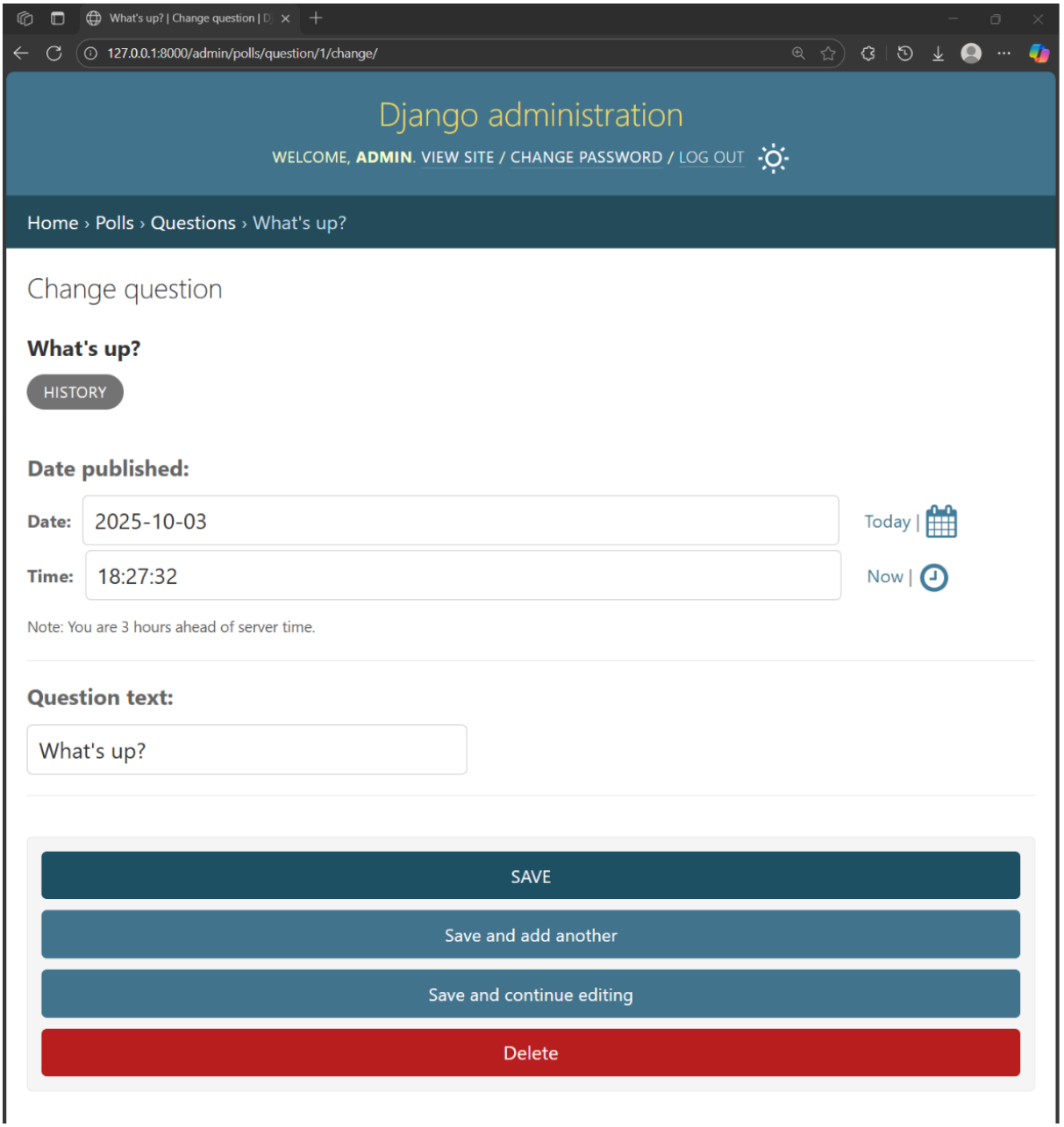
60. Настройка формы администратора

admin.py

polls > admin.py > ...

```python
from django.contrib import admin

from .models import Question

# Register your models here.

class QuestionAdmin(admin.ModelAdmin):
    fields = ["pub_date", "question_text"]

admin.site.register(Question, QuestionAdmin)
```

Баранов Д.А. ИВТ 2.1



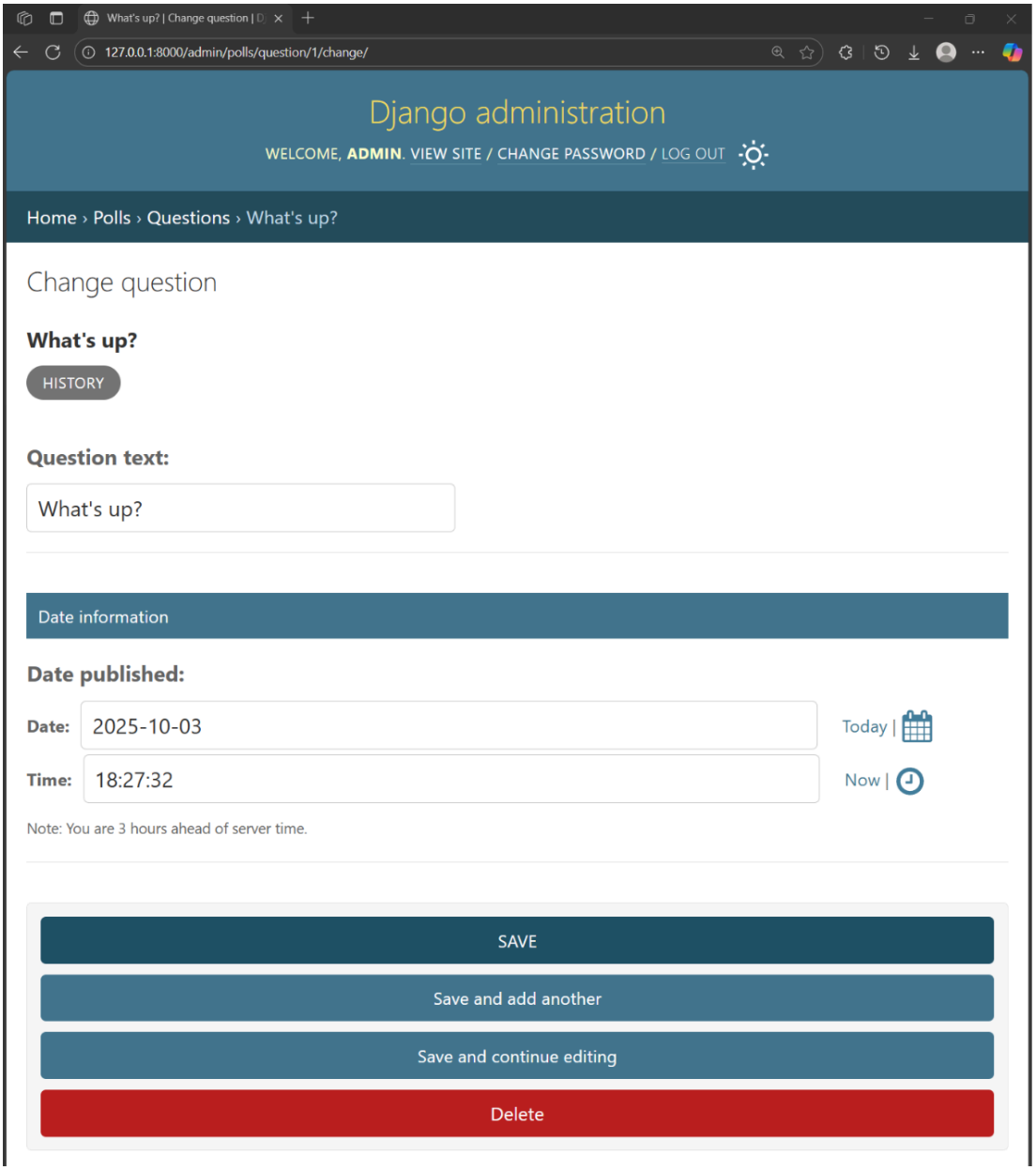
61. Разделение формы вверх в наборы полей

admin.py

polls > admin.py > ...

```python
from django.contrib import admin

from .models import Question

# Register your models here.

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {"fields": ["question_text"]}),
        ("Date information", {"fields": ["pub_date"]}),
    ]

admin.site.register(Question, QuestionAdmin)
```

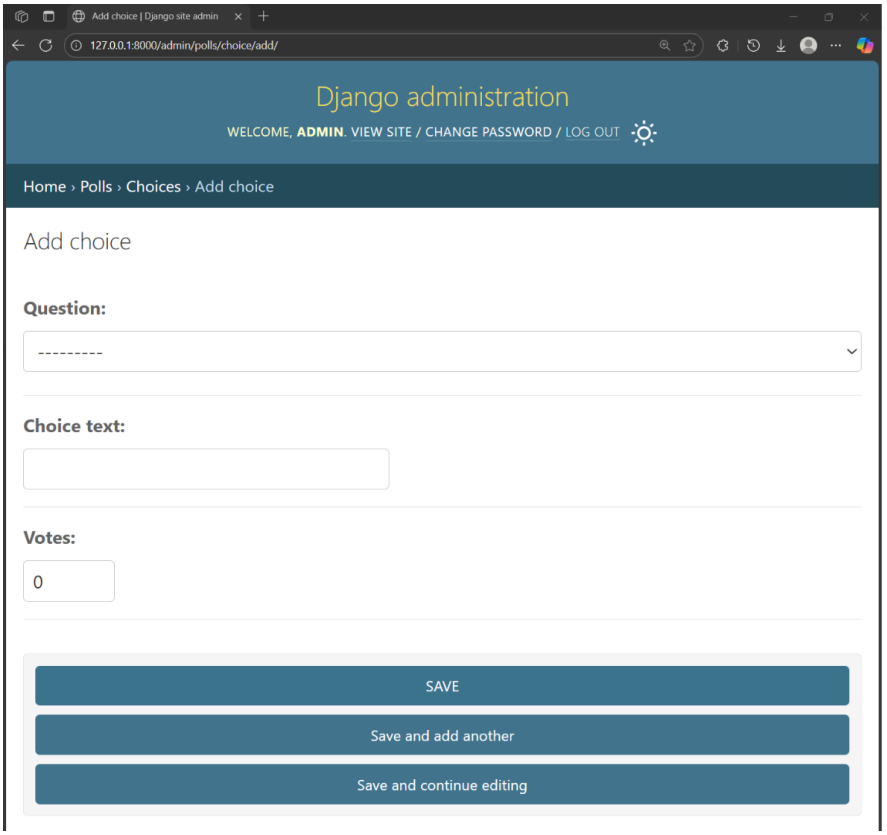
Баранов Д.А. ИВТ 2.1



62. Добавление связанных объектов

admin.py

```
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)
```

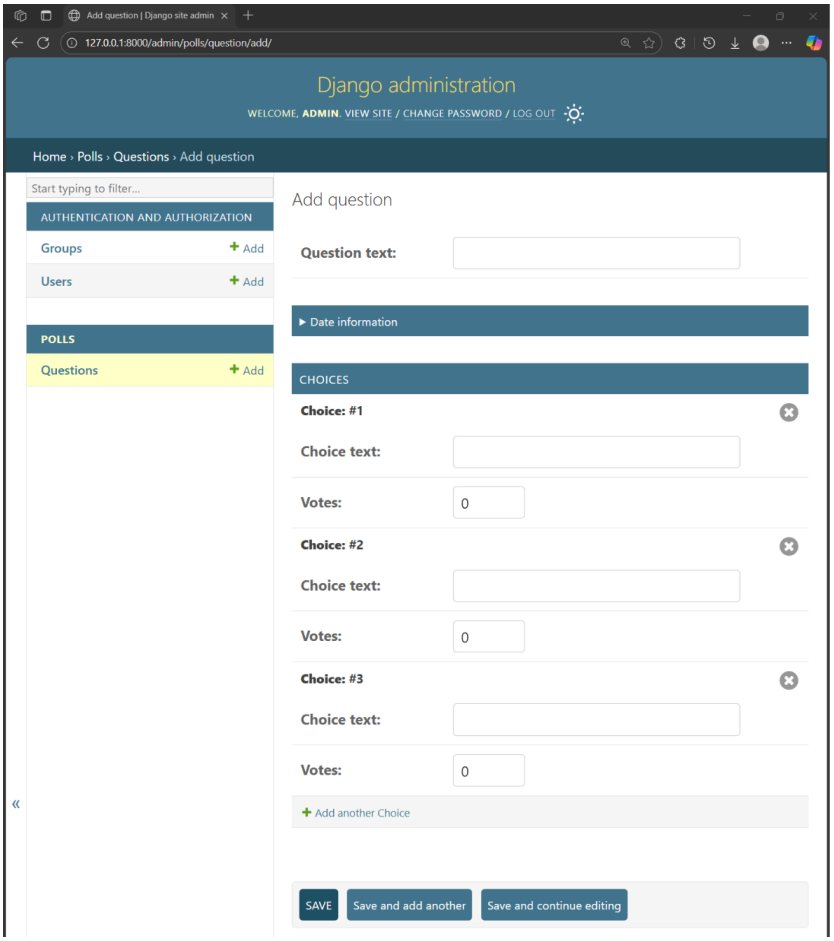


63. Обновление

admin.py

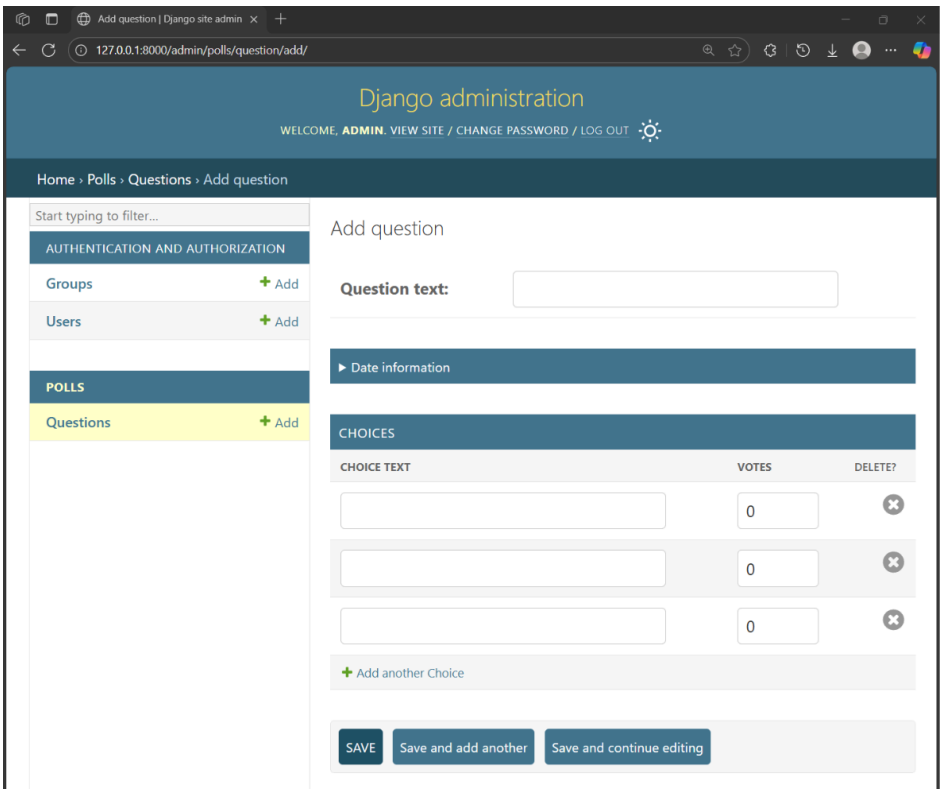
```python
from django.contrib import admin

from .models import Choice, Question

# Register your models here.

class ChoiceInline(admin.StackedInline):
    model = Choice
    extra = 3

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {"fields": ["question_text"]}),
        ("Date information", {"fields": ["pub_date"], "classes": ["collapse"]}),
    ]
    inlines = [ChoiceInline]

admin.site.register(Question, QuestionAdmin)
```

Баранов Д.А. ИВТ 2.1



64. Смена на табличный формат

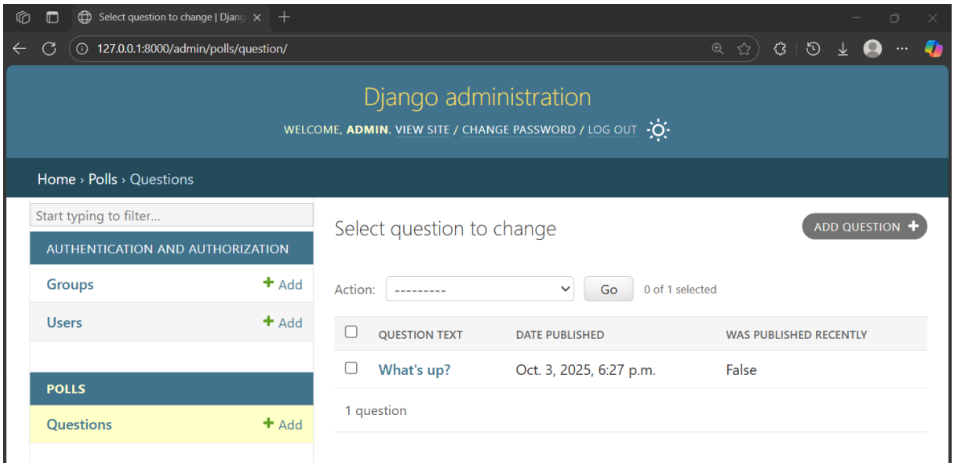
```
7   class ChoiceInline(admin.TabularInline):
```



65. Настройка списка изменений администратора

admin.py

```python
from django.contrib import admin

from .models import Choice, Question

# Register your models here.

class ChoiceInline(admin.TabularInline):
    model = Choice
    extra = 3

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {"fields": ["question_text"]}),
        ("Date information", {"fields": ["pub_date"], "classes": ["collapse"]}),
    ]
    inlines = [ChoiceInline]
    list_display = ["question_text", "pub_date", "was_published_recently"]

admin.site.register(Question, QuestionAdmin)
```



66. Использование декоратора

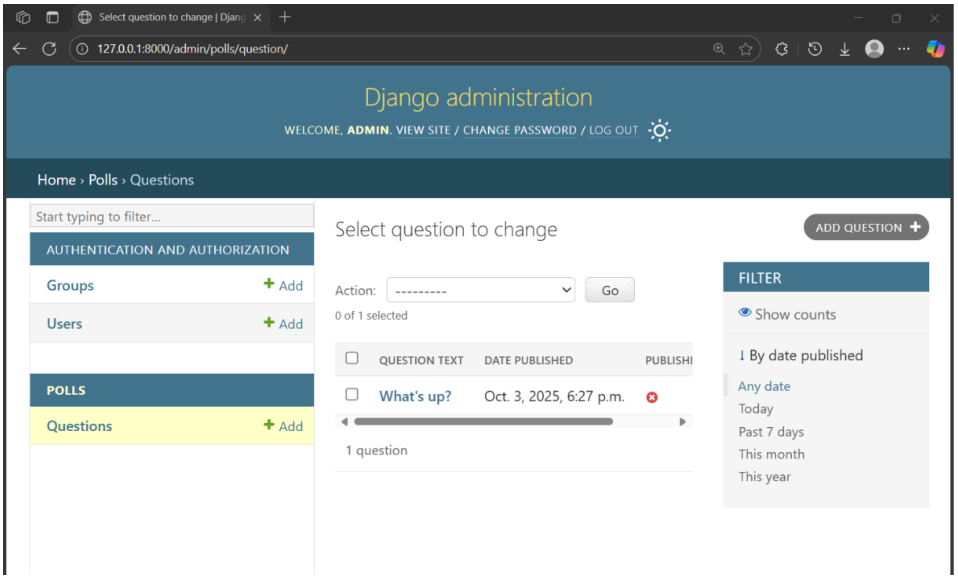
models.py

```python
import datetime

from django.db import models
from django.utils import timezone
from django.contrib import admin

# Create your models here.
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField("date published")
    def __str__(self):
        return self.question_text
    @admin.display(
            boolean=True,
            ordering="pub_date",
            description="Published recently?",
    )
    def was_published_recently(self):
        now = timezone.now()
        return now - datetime.timedelta(days=1) <= self.pub_date <= now

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

Баранов Д.А. ИВТ 2.1

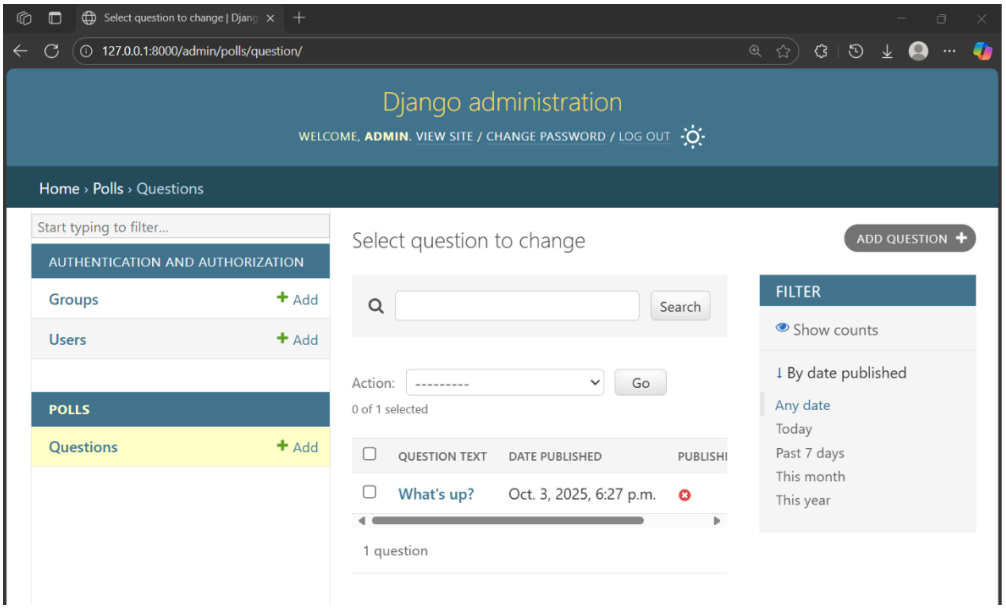
67. Добавление list_filter

```python
from django.contrib import admin

from .models import Choice, Question

# Register your models here.

class ChoiceInline(admin.TabularInline):
    model = Choice
    extra = 3

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {"fields": ["question_text"]}),
        ("Date information", {"fields": ["pub_date"], "classes": ["collapse"]}),
    ]
    inlines = [ChoiceInline]
    list_display = ["question_text", "pub_date", "was_published_recently"]
    list_filter = ["pub_date"]

admin.site.register(Question, QuestionAdmin)
```



68. Добавление окна поиска в верхней части списка изменений

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



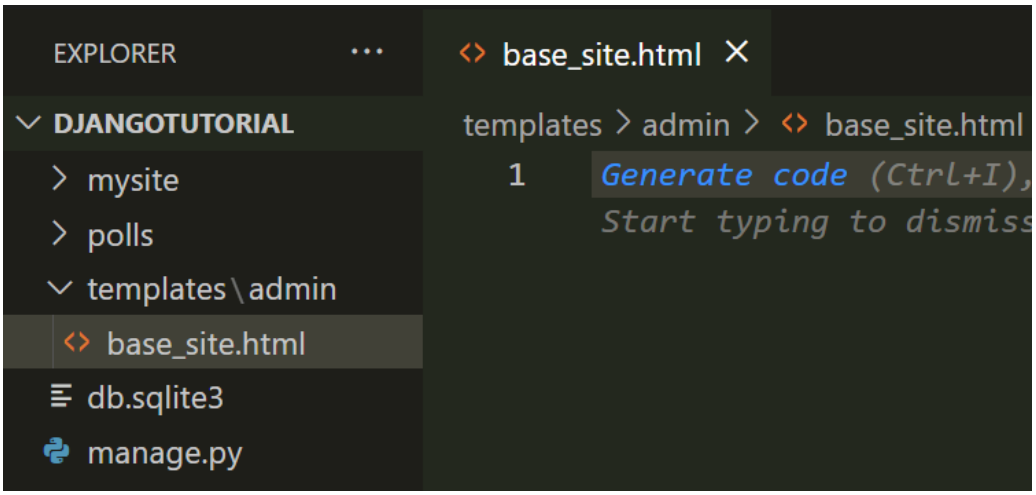
69. Настройка внешнего вида администратора

Настройка шаблонов проекта

settings.py

```
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  ]
```

70. Создание файла base_site.html



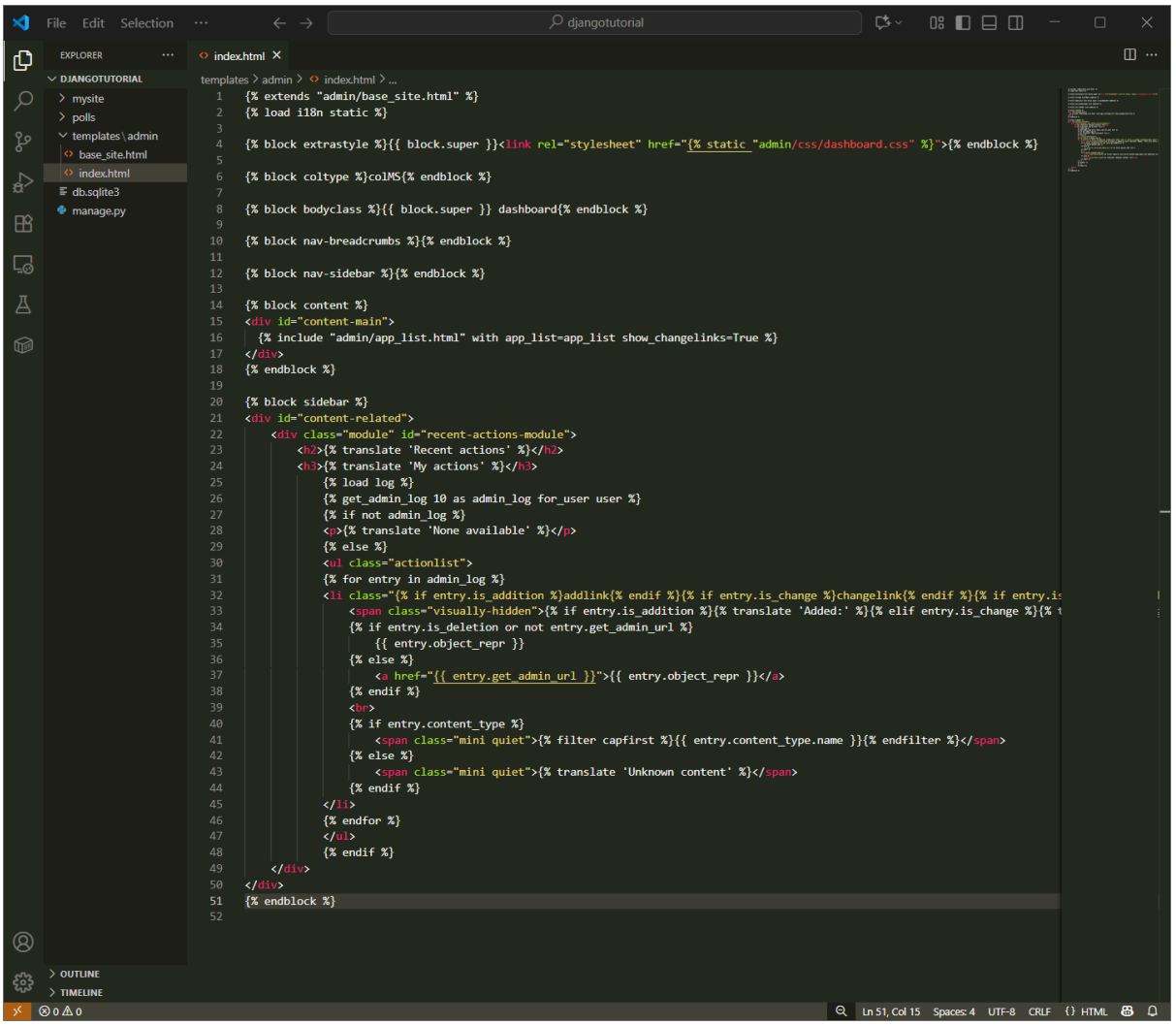
71. Настройка шаблона base_site.html

base_site.html

```
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 %}
```

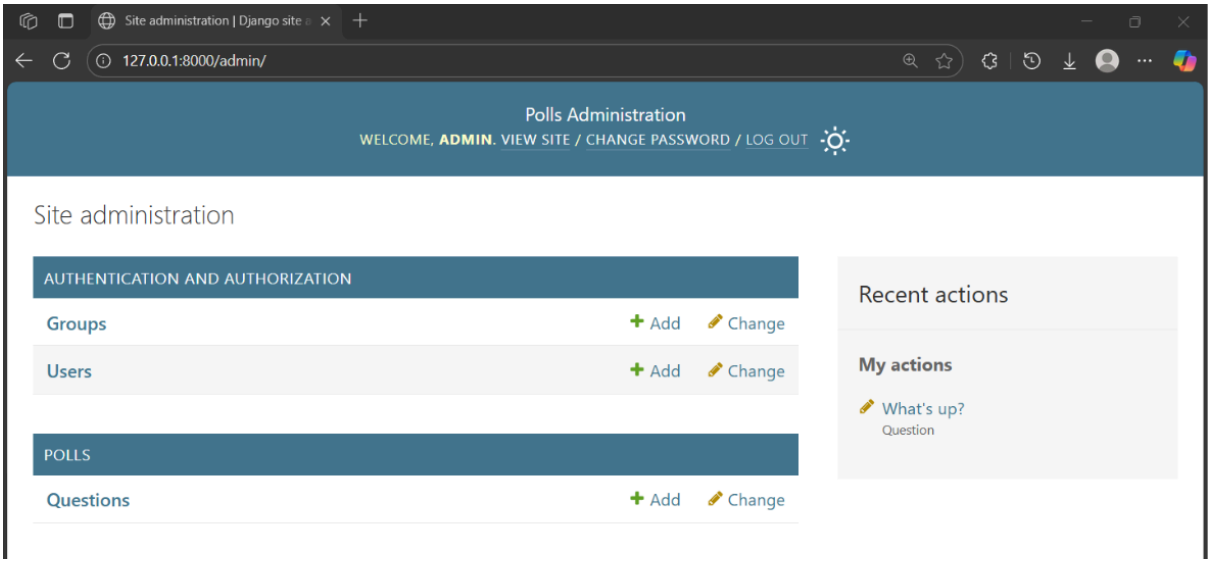
Баранов Д.А. ИВТ 2.1

72. Настройка шаблона index.html

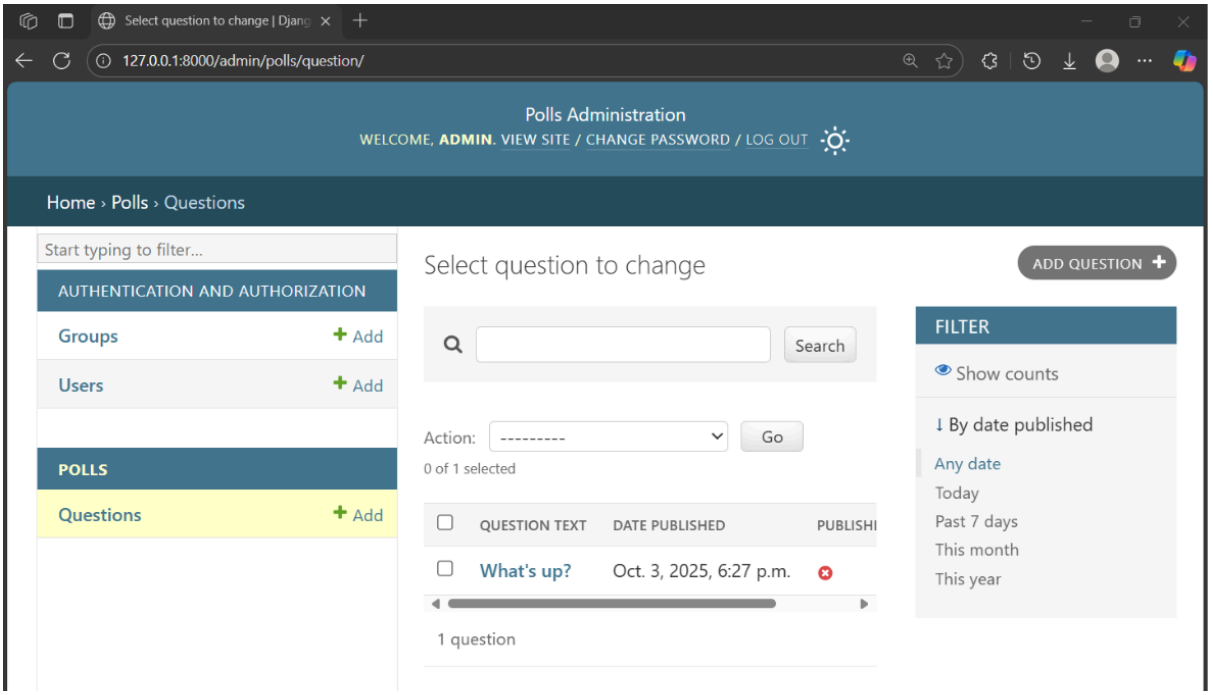


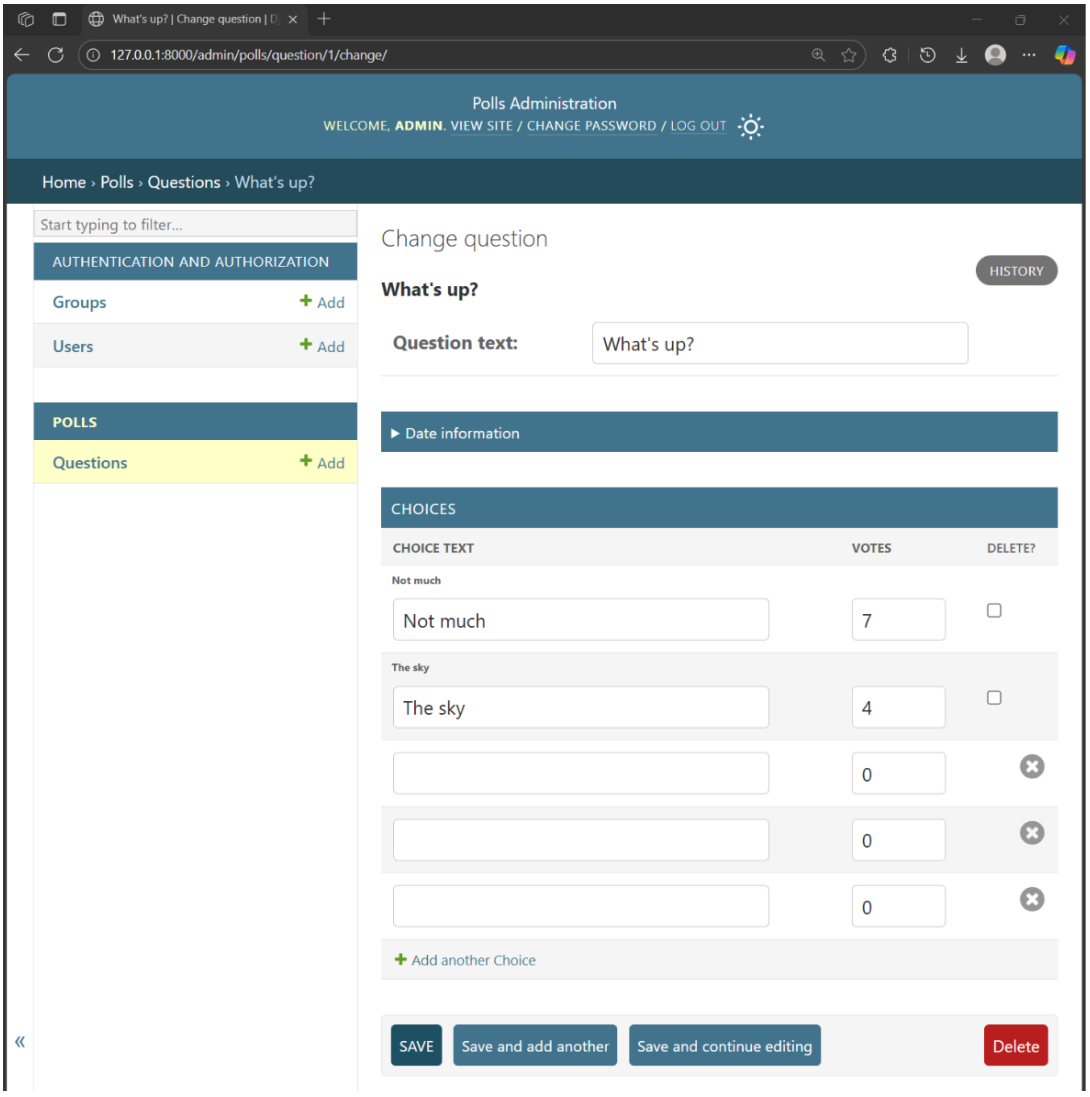
```
{% extends "admin/base_site.html" %}
{% load i18n static %}

{% block extrastyle %}{{ block.super }}<link rel="stylesheet" href="{% static "admin/css/dashboard.css" %}">{% endblock %}

{% block coltype %}colMS{% endblock %}

{% block bodyclass %}{{ block.super }} dashboard{% endblock %}

{% block nav-breadcrumbs %}{% endblock %}

{% block nav-sidebar %}{% endblock %}

{% block content %}
<div id="content-main">
  {% include "admin/app_list.html" with app_list=app_list show_changelinks=True %}
</div>
{% endblock %}

{% block sidebar %}
<div id="content-related">
    <div class="module" id="recent-actions-module">
        <h2>{% translate 'Recent actions' %}</h2>
        <h3>{% translate 'My actions' %}</h3>
            {% load log %}
            {% get_admin_log 10 as admin_log for_user user %}
            {% if not admin_log %}
            <p>{% translate 'None available' %}</p>
            {% else %}
            <ul class="actionlist">
            {% for entry in admin_log %}
            <li class="{% if entry.is_addition %}addlink{% endif %}{% if entry.is_change %}changelink{% endif %}{% if entry.is
                <span class="visually-hidden">{% if entry.is_addition %}{% translate 'Added:' %}{% elif entry.is_change %}{%
                {% if entry.is_deletion or not entry.get_admin_url %}
                    {{ entry.object_repr }}
                {% else %}
                    <a href="{{ entry.get_admin_url }}">{{ entry.object_repr }}</a>
                {% endif %}
                <br>
                {% if entry.content_type %}
                    <span class="mini quiet">{% filter capfirst %}{{ entry.content_type.name }}{% endfilter %}</span>
                {% else %}
                    <span class="mini quiet">{% translate 'Unknown content' %}</span>
                {% endif %}
            </li>
            {% endfor %}
            </ul>
            {% endif %}
    </div>
</div>
{% endblock %}
```

Результаты:



Баранов Д.А. ИВТ 2.1





Баранов Д.А. ИВТ 2.1