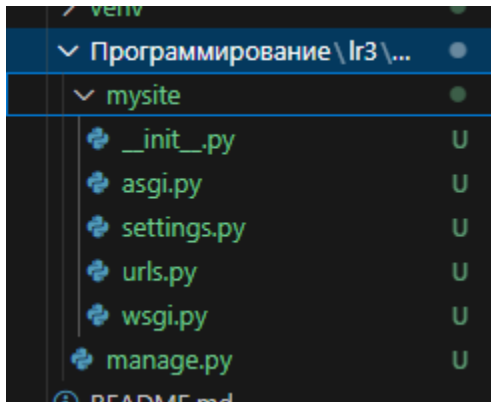


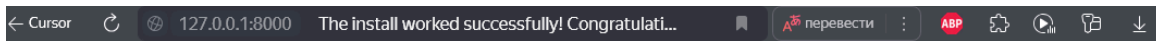
Часть 1: Запросы и ответы

С помощью команды создадим проект

```
nastya@DESKTOP-4RHVHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/lr3
$ django-admin startproject mysite
(venv)
```



С помощью python manage.py запустим проект и увидим стартовую страницу



The install worked successfully! Congratulations!

View [release notes](#) for Django 5.1

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

django

Создадим व्युшुकु, которая будет представлением главной страницы

```
rgpu_7sem > Программирование > lr3 > mysite > polls > views.py > ...
1  from django.http import HttpResponse
2
3
4  def index(request):
5      return HttpResponse("Hello, world. You're at the polls index.")
6
```

Настроим путь для приложения

```
1  JS tasks.js      main.py convex_text 4      setup.py 1
rgpu_7sem > Программирование > lr3 > mysite > polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6      path("", views.index, name="index"),
7  ]
```

И пути всего проекта, подключающее пути приложения polls

```
rgpu_7sem > Программирование > lr3 > mysite > mysite > urls.py > ...
1  from django.contrib import admin
2  from django.urls import include, path
3
4  urlpatterns = [
5      path("polls/", include("polls.urls")),
6      path("admin/", admin.site.urls),
7  ]
```

Запустим и увидим, что получилось

```
← ↻ 127.0.0.1:8000 127.0.0.1:8000/polls/ 📖 🗑️ перевести ⋮
Hello, world. You're at the polls index.
```

Часть 2: Модели и сайт администратора

Поставим время на Российское (МСК)

```
TIME_ZONE = 'Europe/Moscow'  
| Ctrl+L to chat, Ctrl+K to generate  
USE_TZ = True
```

Выполним миграции (модели из подключенных приложений)

```
$ 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  
(venv)
```

Создадим свои модели

```
rgpu_7sem > Программирование > lr3 > mysite > polls > models.py > Choice  
1  from django.db import models  
2  
3  
4  class Question(models.Model):  
5      question_text = models.CharField(max_length=200)  
6      pub_date = models.DateTimeField("date published")  
7  
8  
9  class Choice(models.Model):  
10     question = models.ForeignKey(Question, on_delete=models.CASCADE)  
11     choice_text = models.CharField(max_length=200)  
12     votes = models.IntegerField(default=0)
```

Создадим для них миграции

```
nastya@DESKTOP-4RHHK2 MINGW64 /e/Repos
$ python manage.py makemigrations
Migrations for 'polls':
  polls\migrations\0001_initial.py
    + Create model Question
    + Create model Choice
(venv)
nastya@DESKTOP-4RHHK2 MINGW64 /e/Repos
```

```
nastya@DESKTOP-4RHHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/1r3/mysite
$ python manage.py sqlmigrate polls 0001
BEGIN;
--
-- Create model Question
--
CREATE TABLE "polls_question" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "question_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" big
int NOT NULL REFERENCES "polls_question" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE INDEX "polls_choice_question_id_c5b4b260" ON "polls_choice" ("question_id");
COMMIT;
(venv)
nastya@DESKTOP-4RHHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/1r3/mysite
$
```

```
nastya@DESKTOP-4RHHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирован
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK
(venv)
```

В терминале проведем операции с этими моделями

```
nastya@DESKTOP-4RHHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/1r3/mysite
$ python manage.py shell
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from polls.models import Choice, Question
>>> 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(2024, 10, 10, 13, 49, 25, 164685, tzinfo=datetime.timezone.utc)
>>> q.question_text = "What's up?"
>>> q.save()
>>> Question.objects.all()
<QuerySet [<Question: Question object (1)>]>
>>>
```

Добавим магический метод `str`, чтобы явно отображалось описание модели

```
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
```

Добавим метод `was_published_recently`

```
sem > Программирование > lr3 > mysite > polls > models.py > ...
from django.db import models
from django.utils import timezone
import datetime

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
```

```

(venv)
nastya@DESKTOP-4RHVHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/1r3/mysite
$ python manage.py shell
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from polls.models import Choice, Question
>>> 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 "<console>", line 1, in <module>
  File "E:\Repositories\rgpu_7sem\venv\Lib\site-packages\django\db\models\manager.py", line 87, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
           ~~~~~~^~~~~~
  File "E:\Repositories\rgpu_7sem\venv\Lib\site-packages\django\db\models\query.py", line 649, 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",
...
...
KeyboardInterrupt
>>> 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>]>
>>> 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})
>>>

```

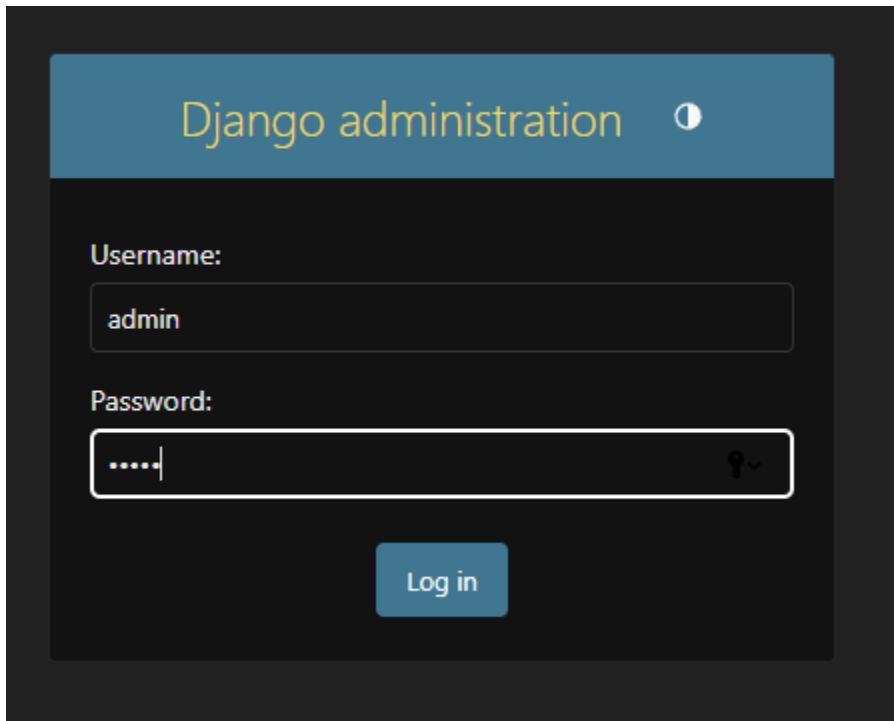
Создадим админа

```

$ python manage.py createsuperuser
Username (leave blank to use 'nastya'): admin
Email address: admin@mail.ru
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(venv)

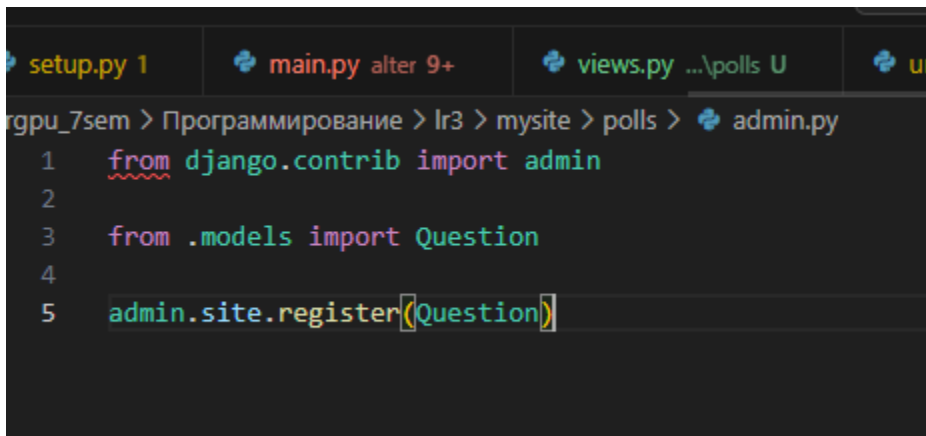
```

Залогинимся



The image shows the Django administration login interface. At the top, there is a blue header with the text "Django administration" and a small circular icon. Below the header, the page has a dark background. There are two input fields: "Username:" with the value "admin" and "Password:" with masked characters ".....". A "Log in" button is located below the password field.

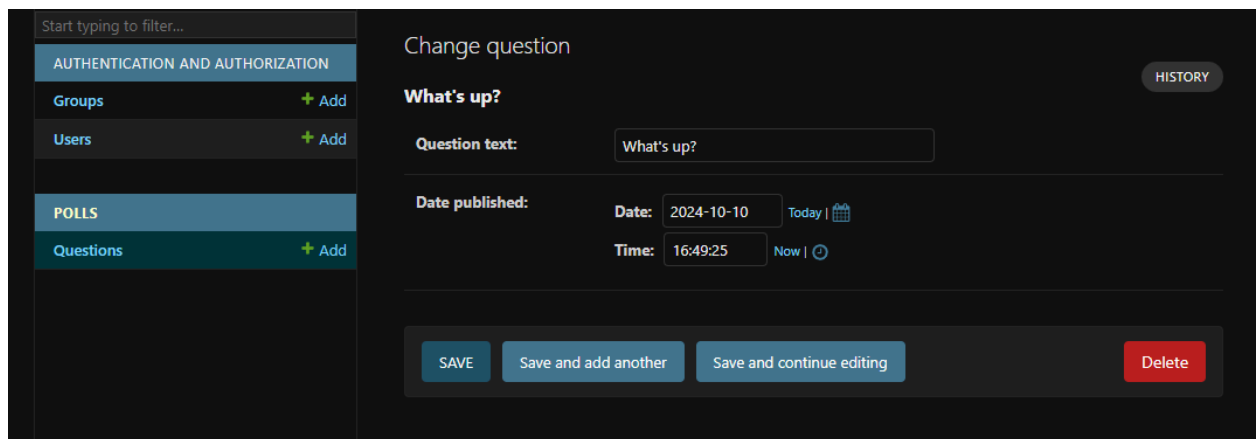
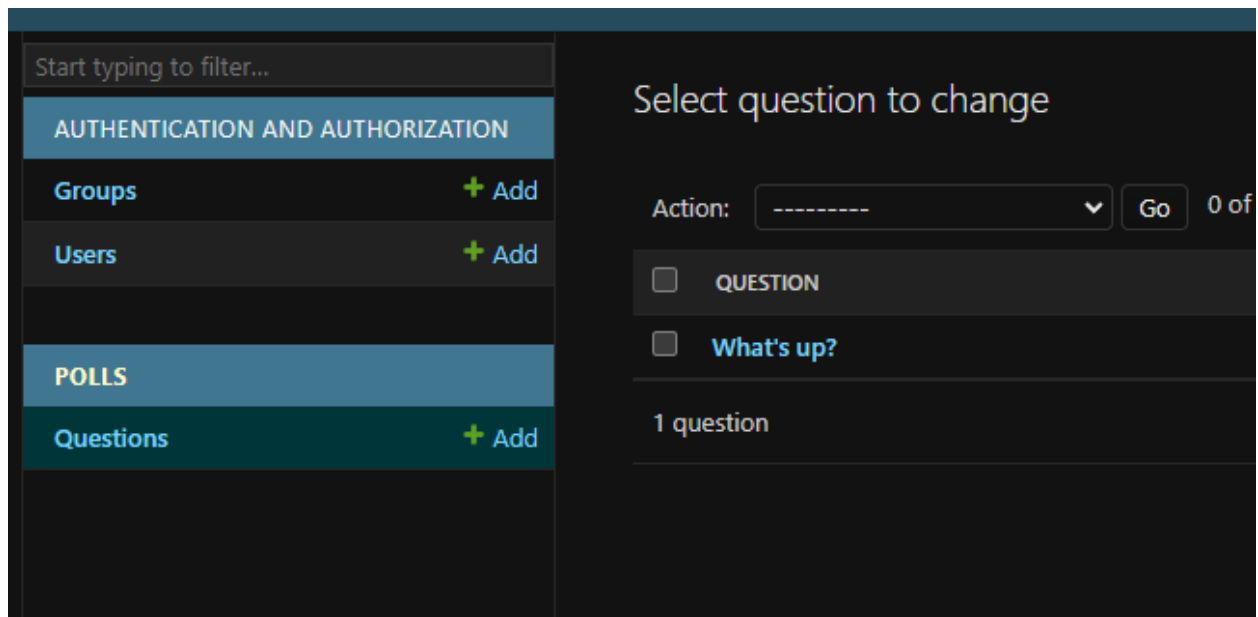
В админке регистрируем модель Вопросов



The image shows a code editor with a dark theme. The file explorer at the top shows files: setup.py, main.py, views.py, and ur. The main editor area shows the content of admin.py, with the following code:

```
1 from django.contrib import admin
2
3 from .models import Question
4
5 admin.site.register(Question)
```

И в админке появилась эта модель



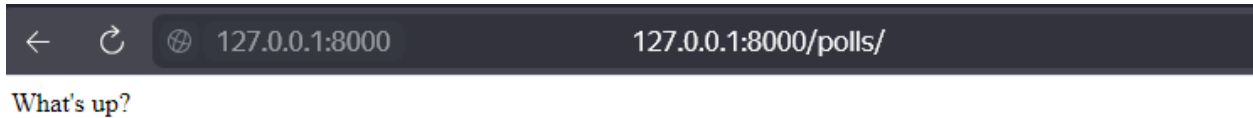
Часть 3: Views and templates

В приложении добавим пути

```
gpu_7sem > Программирование > lr3 > mysite > polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6      # ex: /polls/
7      path("", views.index, name="index"),
8      # ex: /polls/5/
9      path("<int:question_id>/", views.detail, name="detail"),
10     # ex: /polls/5/results/
11     path("<int:question_id>/results/", views.results, name="results"),
12     # ex: /polls/5/vote/
13     path("<int:question_id>/vote/", views.vote, name="vote"),
14 ]
```

И представления для каждого пути

```
gpu_7sem > Программирование > lr3 > mysite > polls > views.py > ...
1  from django.http import HttpResponse
2
3  from .models import Question
4  | Ctrl+L to chat, Ctrl+K to generate
5
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 HttpResponse(output)
10
11
12  def detail(request, question_id):
13      return HttpResponse("You're looking at question %s." % question_id)
14
15
16  def results(request, question_id):
17      response = "You're looking at the results of question %s."
18      return HttpResponse(response % question_id)
19
20
21  def vote(request, question_id):
22      return HttpResponse("You're voting on question %s." % question_id)
```



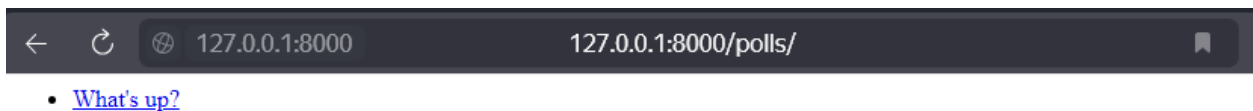
Добавим html

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

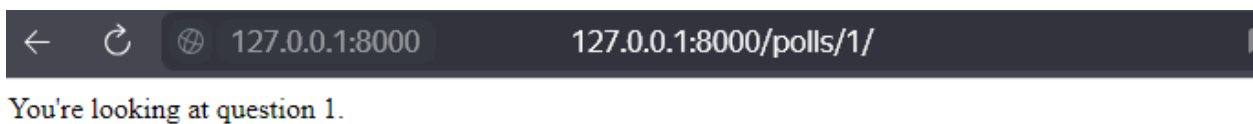
from .models import Question

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):
```



По ссылке конкретного голосования будет такая надпись по конкретному id



```
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})
```

← ↻ 127.0.0.1:8000 Page not found at /polls/456545/

Page not found (404)

Question does not exist

Request Method: GET
Request URL: http://127.0.0.1:8000/polls/456545/
Raised by: polls.views.detail

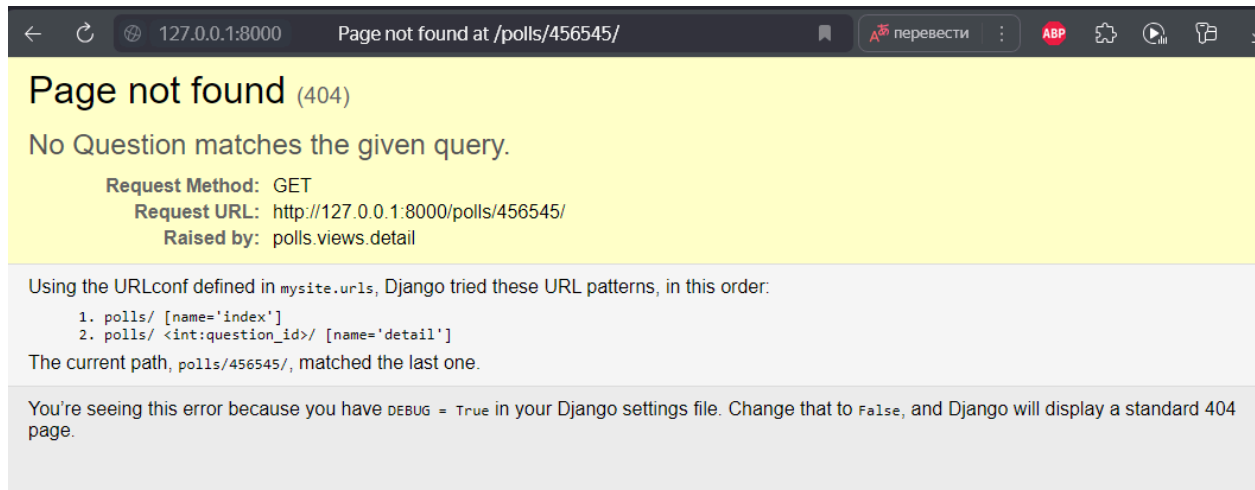
Using the URLconf defined in `mysite.urls`, Django tried these URL patterns, in this order:

1. `polls/` [name='index']
2. `polls/ <int:question_id>/` [name='detail']

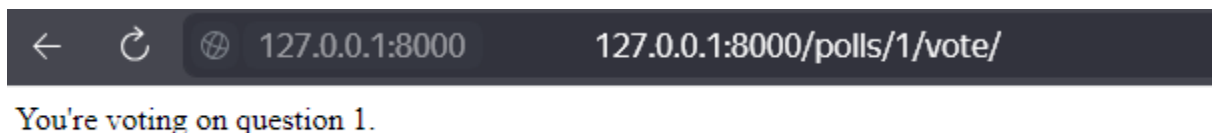
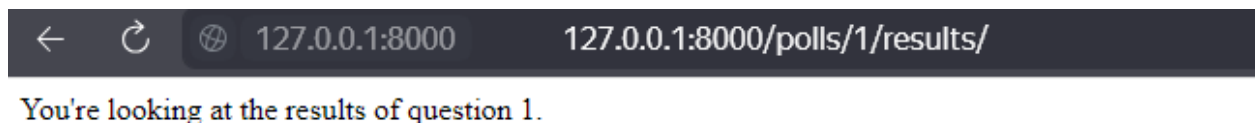
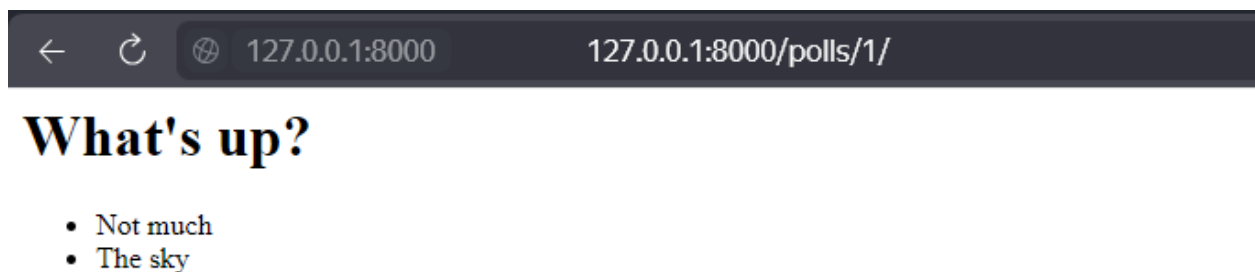
The current path, `polls/456545/`, matched the last one.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`.

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



```
u_7sem > Программирование > lr3 > mysite > polls > templates > pc
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>
```

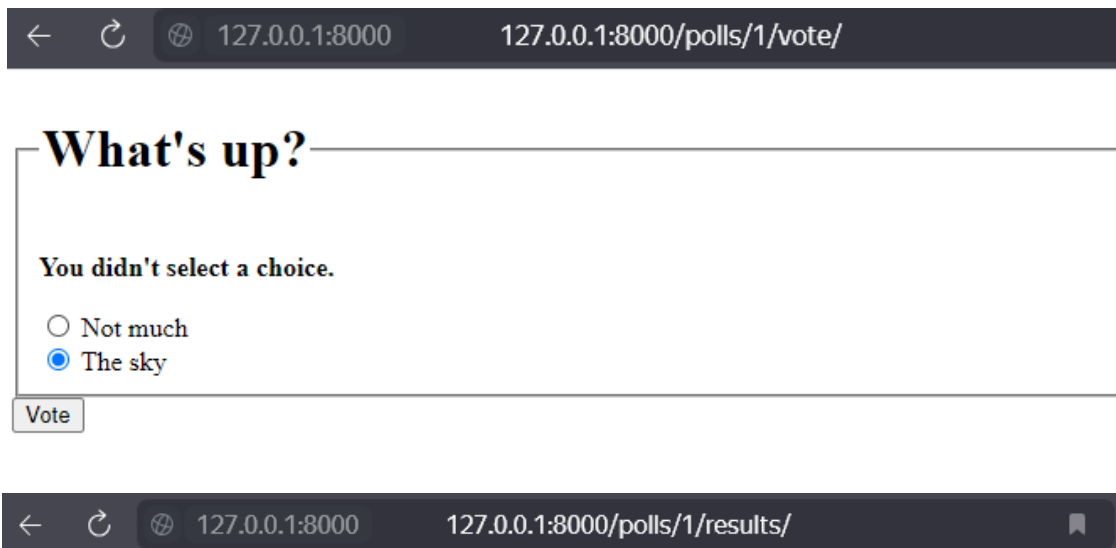


Часть 4. Forms and generic views

Создадим форму в views.py

```
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,)))
```

Увидим результат:



← ↻ 127.0.0.1:8000 127.0.0.1:8000/polls/1/vote/

What's up?

You didn't select a choice.

☐ Not much

☒ The sky

Vote

← ↻ 127.0.0.1:8000 127.0.0.1:8000/polls/1/results/

What's up?

- Not much -- 0 votes
- The sky -- 1 vote

[Vote again?](#)

В Джанго можно использовать классы (дженерики) вместо функций. Код становится короче, тк он уже включает базовые функции

```
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
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
19 class DetailView(generic.DetailView):
20     model = Question
21     template_name = "polls/detail.html"
22
23
24 class ResultsView(generic.DetailView):
25     model = Question
26     template_name = "polls/results.html"
27
28
```

Часть5: Testing

Воспроизведем баг в терминале

```
$ python manage.py shell
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [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
>>> from polls.models import Question
>>> future_question = Question(pub_date=timezone.now() + datetime.timedelta(days=30))
>>> future_question.was_published_recently()
True
>>> 
```

Напишем тест и увидим, что он не прошел

```
rgpu_7sem > Программирование > lr3 > mysite > polls > tests.py > ...
3  from django.test import TestCase
4  from django.utils import timezone
5
6  from .models import Question
7
8
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
13         is in the future.
14         """
15         time = timezone.now() + datetime.timedelta(days=30)
16         future_question = Question(pub_date=time)
17         self.assertIs(future_question.was_published_recently(), False)
18
```

PROBLEMS 91 OUTPUT DEBUG CONSOLE TERMINAL PORTS

(venv)
nastya@DESKTOP-4R4MHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/lr3/mysite
\$ 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

Traceback (most recent call last):
 File "E:\Repositories\rgpu_7sem\Программирование\lr3\mysite\polls\tests.py", line 17, 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 0.001s

FAILED (failures=1)
Destroying test database for alias 'default'...
(venv)
nastya@DESKTOP-4R4MHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/lr3/mysite
\$

Исправим функцию

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

Тест прошел успешно

```
$ 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'...  
(venv)  
nastya@DESKTOP-4R4MHK2 MINGW64 /e/Repositories/rgpu_7sem/Программирование/lr3/mysite  
$
```

Напишем еще тесты

```
7         self.assertIs(future_question.was_published_recently(), False)
8
9     def test_was_published_recently_with_old_question(self):
10         """
11         was_published_recently() returns False for questions whose pub_date
12         is older than 1 day.
13         """
14         time = timezone.now() - datetime.timedelta(days=1, seconds=1)
15         old_question = Question(pub_date=time)
16         self.assertIs(old_question.was_published_recently(), False)
17
18     def test_was_published_recently_with_recent_question(self):
19         """
20         was_published_recently() returns True for questions whose pub_date
21         is within the last day.
22         """
23         time = timezone.now() - datetime.timedelta(hours=23, minutes=59, seconds=59)
24         recent_question = Question(pub_date=time)
25         self.assertIs(recent_question.was_published_recently(), True)
```

Все 5 прошли успешно

```
PROBLEMS 93 OUTPUT DEBUG CONSOLE TERMINAL PORTS
System check identified no issues (0 silenced).
.....
-----
Ran 5 tests in 0.019s

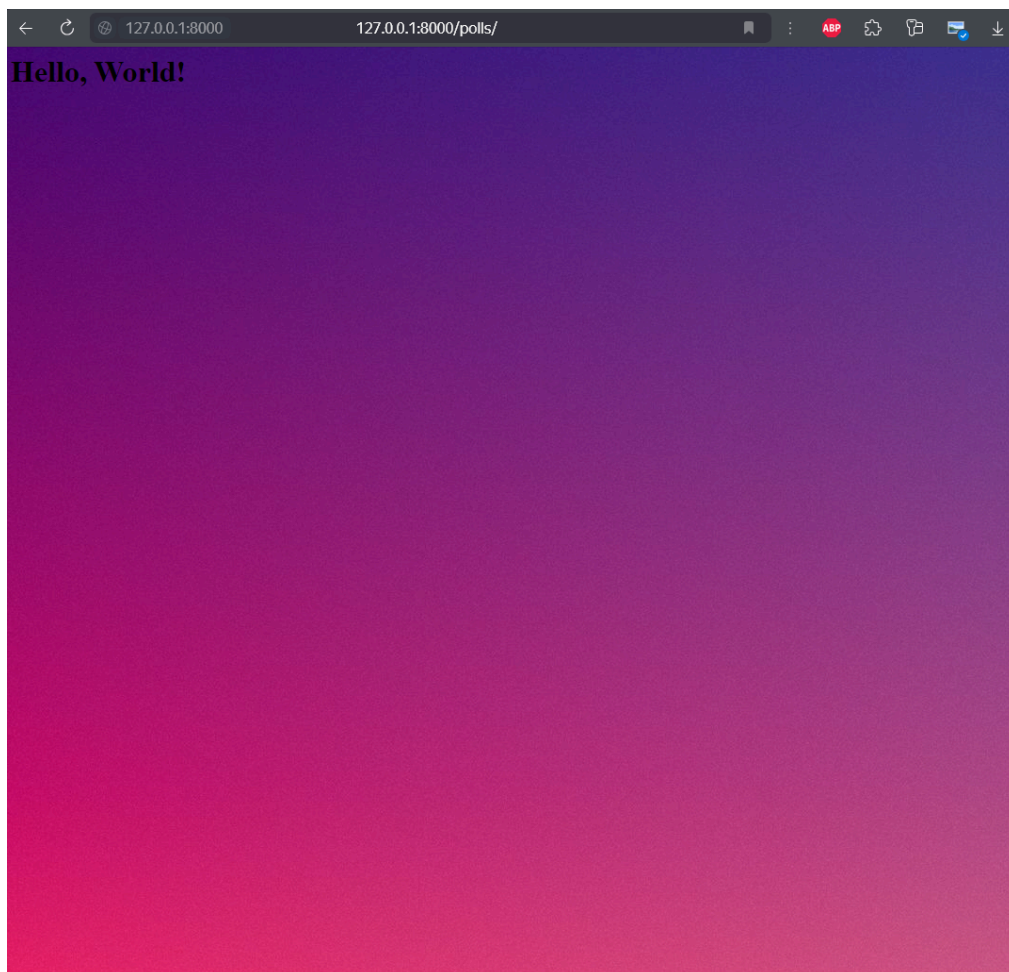
OK
Destroying test database for alias 'default'...
(venv)
root@DESKTOP-4PLM4K3: ~/Documents/Python/Projects/Project1
```


Часть 6: Static files

В джанге можно подключить статику, чтобы изменить стиль страниц (добавим фон)

```
views.py ... \polls 3, U  <> detail.html  <> index.html  background.png  <
polls_7sem > Программирование > lr3 > mysite > polls > static > polls > # style.css > body
1  body {
2  |    background: white url("images/background.png") no-repeat;
3  }
```

```
polls_7sem > Программирование > lr3 > mysite > polls > templates > polls > <> index.html
1  {% load static %}
2
3  <link rel="stylesheet" href="{% static 'polls/style.css' %}">
4  <h1>Hello, World!</h1>
```



Часть 7: Customizing the admin site

Админку можно кастомизировать: добавить отдельные заголовки, фильтрацию и тд.

```
rgpu_7sem > Программирование > lr3 > mysite > polls > admin.py > ...
1  from django.contrib import admin
2
3  from .models import Question
4
5
6  class QuestionAdmin(admin.ModelAdmin):
7      fieldsets = [
8          (None, {"fields": ["question_text"]}),
9          ("Date information", {"fields": ["pub_date"]}),
10     ]
11
12
13  admin.site.register(Question, QuestionAdmin)
```


Change question


What's up?

HISTORY

Question text:

Date information

Date published: Date: Today | 

Time: Now | 

grip / sent / программирование / it's / mysite / polls / admin.py / ...

```

1  from django.contrib import admin
2
3  from .models import Choice, Question
4
5
6  class ChoiceInline(admin.StackedInline):
7      model = Choice
8      extra = 3
9
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
19 admin.site.register(Question, QuestionAdmin)

```

Add question

Question text:

► Date information

CHOICES

Choice: #1



Choice text:

Votes:

Choice: #2



Choice text:

Votes:

Choice: #3



Choice text:

Votes:

+ Add another Choice

```
class ChoiceInline(admin.TabularInline):  
    def __init__(self, *args, **kwargs):
```

Question text:

► Date information

CHOICES

CHOICE TEXT

VOTES

DELETE?



+ Add another Choice