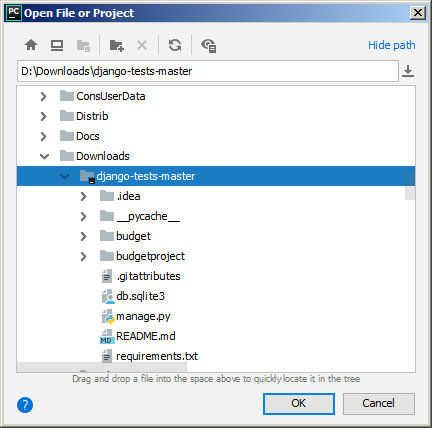
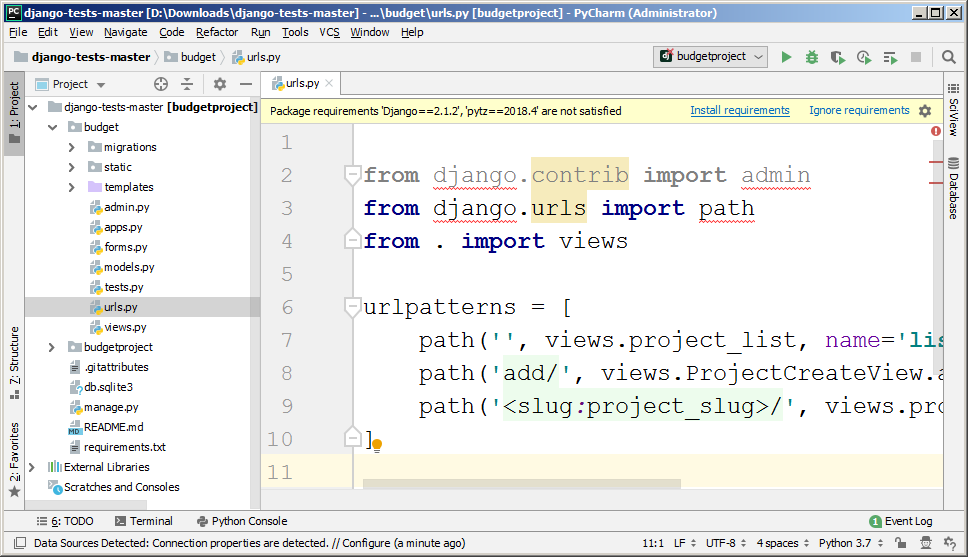
1. Скачайте и распакуйте проект для тестирования:

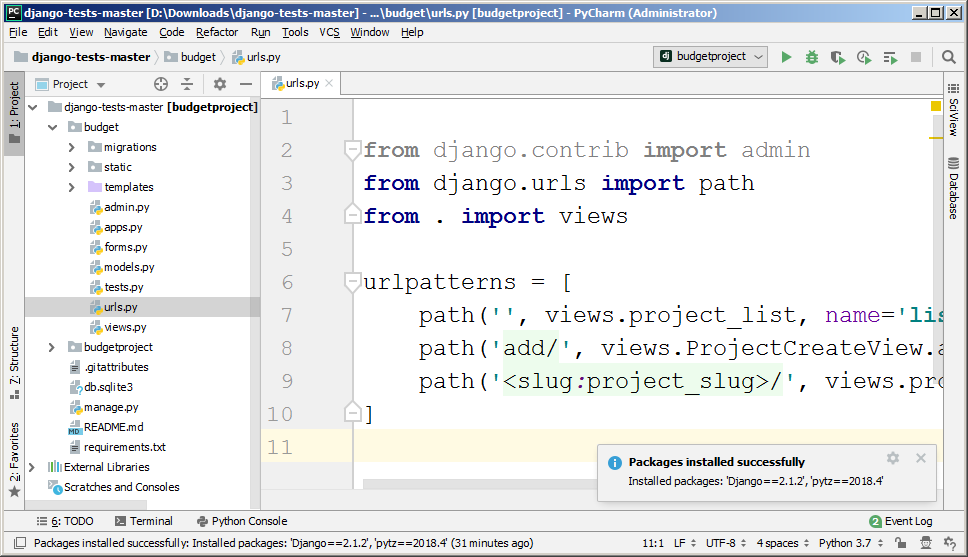
https://github.com/Burmakova/django-tests

2. Откройте проект в PyCharm:



3. Откройте любой Python file, доустановите требуемые пакеты:





4. Скачанное веб-приложение предназначено для управления бюджетом проектов. Запустите веб-приложение и познакомьтесь с ним: добавьте 2 проекта, в каждом проекте по 2 категории расходов, по 2 расхода каждой категории.

5. Создайте в папке budget папку **tests** для тестов и в ней первый Python file для тестирования urls.py – назовем **test\_urls.py**.

6. Создайте в папке budget/tests Python file с именем **\_\_init\_\_.py** для возможности запуска тестов.

7. Напишите в файле test\_urls.py класс с тремя тестами для трех разновидностей url, поддерживаемых веб-приложением:

from django.test import SimpleTestCase

from django.urls import reverse, resolve

from budget.views import project\_list, project\_detail, ProjectCreateView

# класс - набор тестов

class TestUrls(SimpleTestCase):

# тест 1:

# главная страница со списком проектов

# генерируется функцией project\_list?

def test\_list\_url\_resolves(self):

# в функцию reverse передается значение атрибута name,

# указанное в urls.py

url = reverse('list')

# сравнение функции, используемой для обработки полученного url,

# c функцией project\_list

self.assertEquals(resolve(url).func, project\_list)

# тест 2:

# страница добавления проекта

# генерируется классом ProjectCreateView?

def test\_add\_url\_resolves(self):

url = reverse('add')

# сравнение класса, используемого для обработки полученного url,

# c классом ProjectCreateView

self.assertEquals(resolve(url).func.view\_class, ProjectCreateView)

# тест 3:

# страница проекта с бюджетом и списком расходов

# генерируется функцией project\_detail?

def test\_detail\_url\_resolves(self):

# подстановка произвольного аргумента 'some-slug'

# в качестве названия проекта

# (не обязательно существующего проекта -

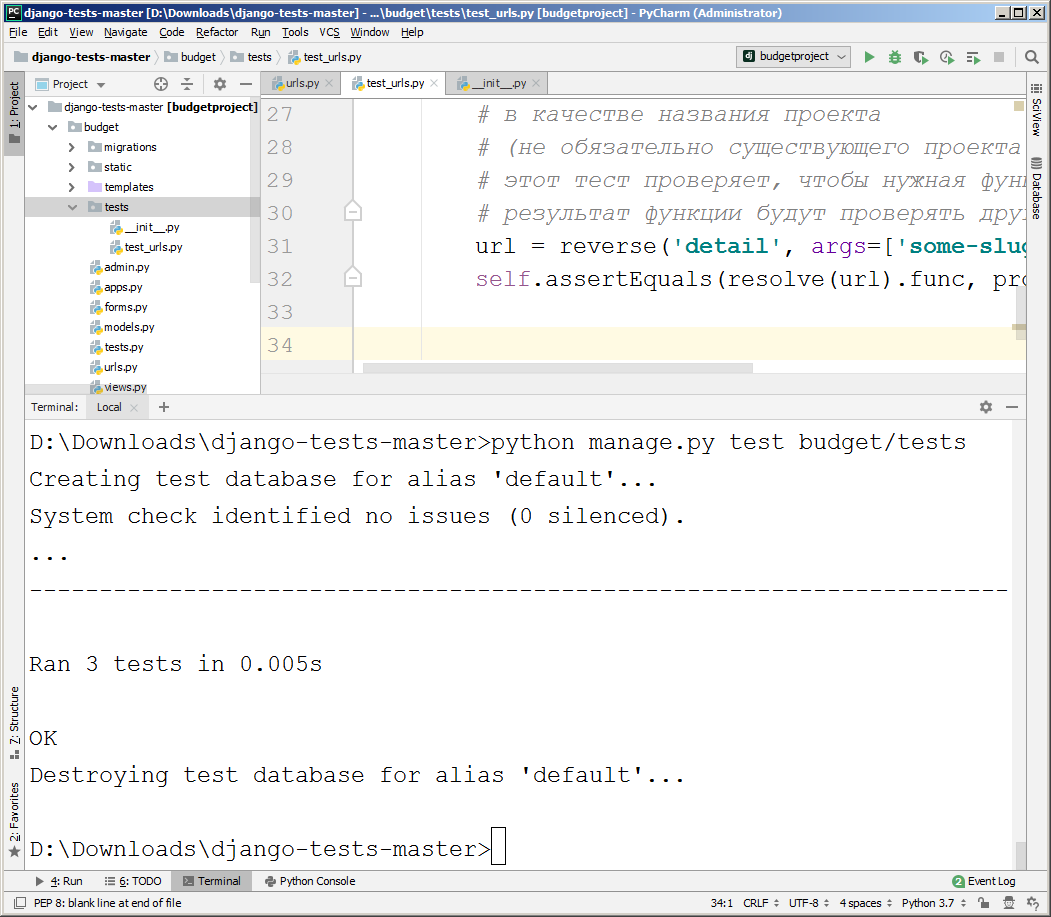
# этот тест проверяет, чтобы нужная фунция просто вызвалась,

# результат функции будут проверять другие тесты)

url = reverse('detail', args=['some-slug'])

self.assertEquals(resolve(url).func, project\_detail)

8. Запустите тесты в терминале командой **python manage.py test budget/tests**, убедитесь в их успешном выполнении:



9. Создайте в папке tests новый Python file для тестирования views.py – назовем **test\_views.py.**

10. Напишите в файле test\_views.py класс с двумя тестами для тестирования главной страницы и страницы с описанием проекта:

from django.test import TestCase, Client

from django.urls import reverse

# класс - набор тестов

class TestViews(TestCase):

# настройка, выполняемая перед запуском тестов:

# подготовка эмулятора браузера (Client) и url-ов,

def setUp(self):

self.client = Client()

self.list\_url = reverse('list')

# будем искать проект с именем project1

self.detail\_url = reverse('detail', args=['project1'])

# тест 1:

# главная страница со списком проектов

# генерируется успешно (код 200) и

# на основе шаблона budget/project-list.html?

def test\_project\_list\_GET(self):

# обращение эмулятора браузера к странице со списком проектов

response = self.client.get(self.list\_url)

self.assertEquals(response.status\_code, 200)

self.assertTemplateUsed(response, 'budget/project-list.html')

# тест 2:

# страница со списком расходов в проекте

# генерируется успешно (код 200) и

# на основе шаблона budget/project-detail.html?

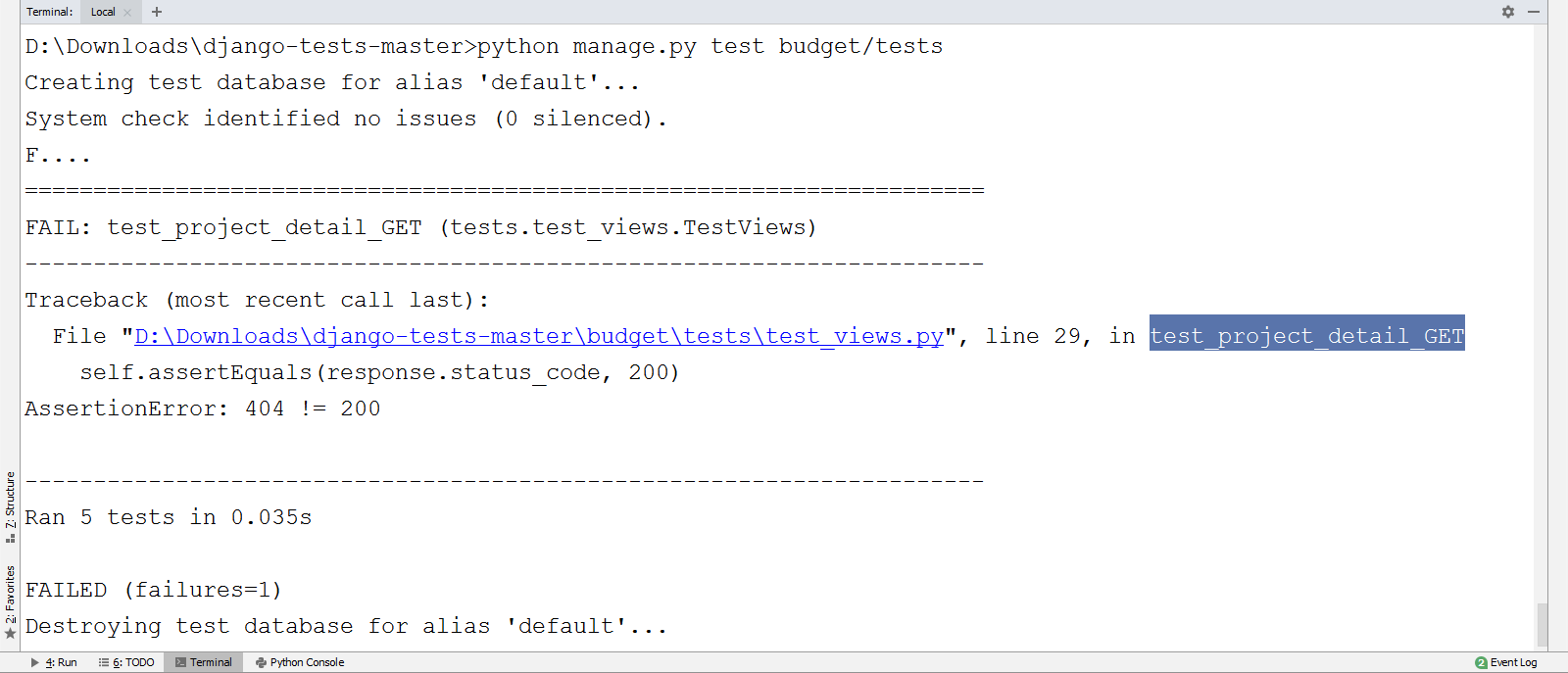
def test\_project\_detail\_GET(self):

response = self.client.get(self.detail\_url)

self.assertEquals(response.status\_code, 200)

self.assertTemplateUsed(response, 'budget/project-detail.html')

11. Запустите тесты в терминале командой **python** **manage.py test budget/tests**, убедитесь в возникновении ошибки в тесте test\_project\_detail\_GET – проект с именем project1 не найден, о чем свидетельствует код 404:



12. Для прохождения теста следует создавать тестируемый проект в базе данных каждый раз при запуске тестов. Для этого следует дополнить метод setUp:

def setUp(self):

self.client = Client()

self.list\_url = reverse('list')

# будем искать проект с именем project1

self.detail\_url = reverse('detail', args=['project1'])

# вставка проекта с именем project1 в БД

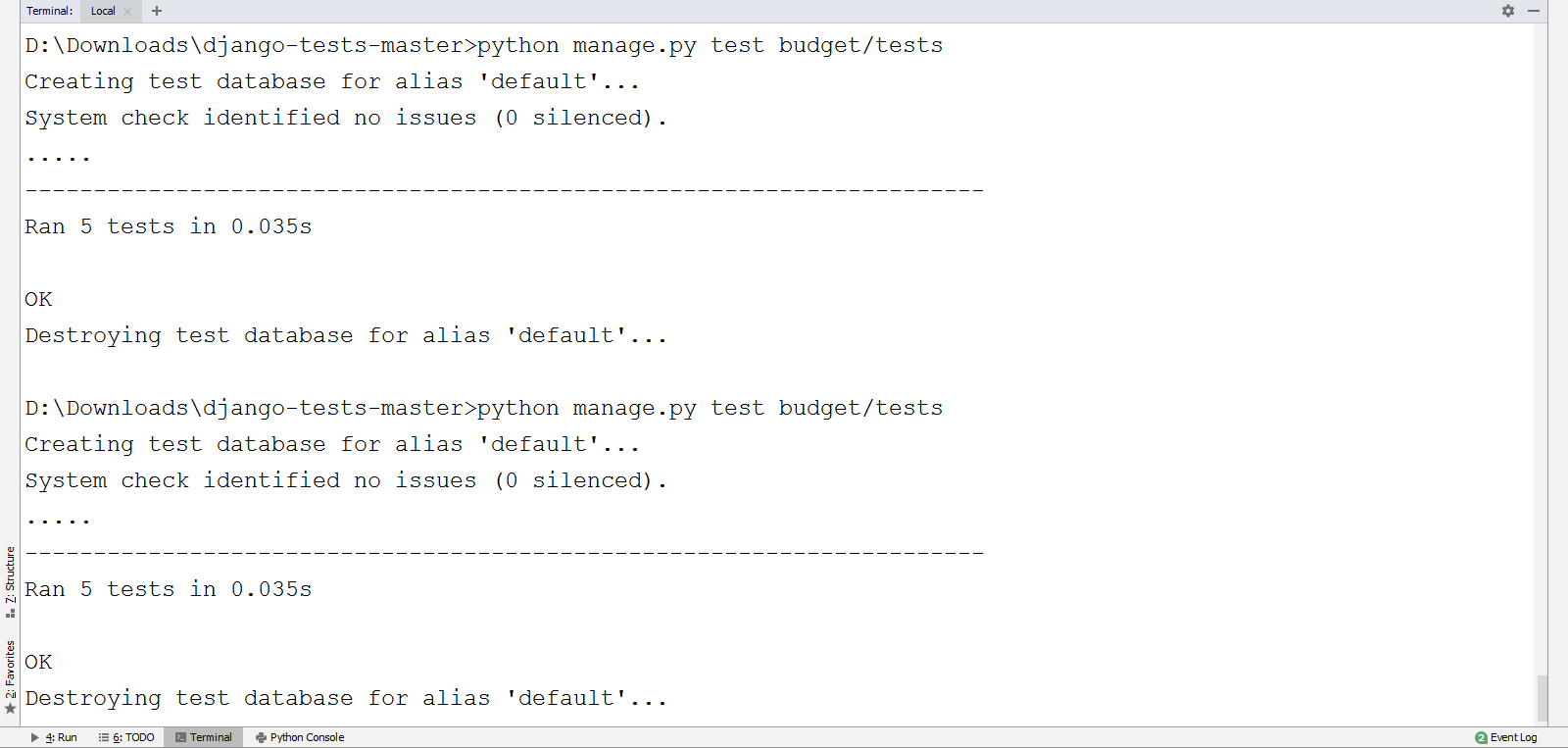
self.project1 = Project.objects.create(

name='project1',

budget=10000

)

13. Запустите тесты в терминале командой **python manage.py test budget/tests** 2 раза – убедитесь, что тесты проходят успешно и что создаваемые при каждом запуске проекты с именем project1 не конфликтуют друг с другом. Это говорит о том, что для каждого теста (каждой функции) создается своя копия БД и для нее вызывается метод setUp:



14. Добавьте в класс еще один тест на добавление расхода в проект:

# тест 3:

# отправка на страницу со списком расходов в проекте

# POST-запроса на добавление нового расхода заканчивается

# переадресацией (код 302) на отображение страницы и

# наличием нового расхода в БД?

def test\_project\_detail\_POST\_add\_expense(self):

# создание в тестовом проекте категории для расходов

Category.objects.create(project=self.project1, name='development')

# отправка на страницу со списком расходов в проекте

# POST-запроса на добавление нового расхода

response = self.client.post(self.detail\_url, {

'title': 'expensel',

'amount': 1000,

'category': 'development'

})

self.assertEquals(response.status\_code, 302)

self.assertEquals(

self.project1.expenses.first().title,

'expensel'

)

15. Запустите тесты в терминале командой **python manage.py test budget/tests**, убедитесь в их успешном выполнении.

16. Добавьте в класс еще один тест на попытку добавления расхода без указания каких-либо сведений о нем – ожидается, что попытка не приведет к увеличению числа строк в таблице расходов:

# тест 4:

# отправка на страницу со списком расходов в проекте

# POST-запроса БЕЗ ДАННЫХ заканчивается

# переадресацией (код 302) на отображение страницы и

# нулевым количеством расходов в БД?

def test\_project\_detail\_POST\_no\_data(self):

response = self.client.post(self.detail\_url)

self.assertEquals(response.status\_code, 302)

self.assertEquals(self.project1.expenses.count(), 0)

17. Запустите тесты в терминале командой python manage.py test budget/tests, убедитесь в их успешном выполнении.

18. Добавьте в класс еще один тест на попытку удаления расхода – поскольку для теста будет использоваться своя независимая копия базы данных, сперва расход потребуется создать:

# тест 5:

# отправка на страницу со списком расходов в проекте

# DELETE-запроса с id расхода заканчивается

# успешным выполнением без вывода данных (код 204) и

# нулевым количеством расходов в БД?

def test\_project\_detail\_DELETE\_deletes\_expense(self):

# создание категории через модель

category1 = Category.objects.create(

project=self.project1,

name='development'

)

# создание расхода через модель

Expense.objects.create(

project=self.project1,

title='expensel',

amount=1000,

category=category1

)

# отправка DELETE-запроса эмулятором браузера

response = self.client.delete(

self.detail\_url,

json.dumps({"id": 1})

)

# проверка результатов

self.assertEquals(response.status\_code, 204)

self.assertEquals(self.project1.expenses.count(), 0)

18. Запустите тесты в терминале командой python manage.py test budget/tests, убедитесь в их успешном выполнении.

19. Добавьте тест с попыткой удаления расхода без указания его идентификатора:

# тест 6:

# отправка на страницу со списком расходов в проекте

# DELETE-запроса БЕЗ id расхода заканчивается

# сообщением об отсутствии страницы (код 404) и

# сохранением одного расхода в БД?

def test\_project\_detail\_DELETE\_no\_id(self):

category1 = Category.objects.create(

project=self.project1,

name='development'

)

Expense.objects.create(

project=self.project1,

title='expensel',

amount=1000,

category=category1

)

# отправка DELETE-запроса БЕЗ id расхода

response = self.client.delete(self.detail\_url)

# проверка результатов

self.assertEquals(response.status\_code, 404)

self.assertEquals(self.project1.expenses.count(), 1)

20. Запустите тесты в терминале командой python manage.py test budget/tests, убедитесь в их успешном выполнении.