1. Для обеспечения возможности удобной работы с нашим приложением из других приложений добавим так называемый программный интерфейс для передачи состояния нашего приложения – содержания таблиц БД (так называемый REST API). Первое, что поможет нам в этом – **установка пакета django-tastypie в используемый в проекте интерпретатор через меню «File»-«Settings...».**
2. **Для работы установленного пакета добавим его название в список установленных приложений в файле settings.py:**

INSTALLED\_APPS = [  
 **'riddles.apps.RiddlesConfig'**,  
 **'django.contrib.admin'**,  
 **'django.contrib.auth'**,  
 **'django.contrib.contenttypes'**,  
 **'django.contrib.sessions'**,  
 **'django.contrib.messages'**,  
 **'django.contrib.staticfiles'**,  
 **'crispy\_forms'**,  
 *# для API* **'tastypie'**,  
]

1. Для работы со строками таблиц БД через API создадим 2 новых класса, наследующих класс tastypie.resources.ModelResource и использующих наши модели Riddle и Option, **в новом файле riddles/api.py:**

*# для API***from** tastypie.resources **import** ModelResource  
**from** riddles.models **import** Riddle, Option   
*# для безопасности API*  
**from** tastypie.authentication **import** BasicAuthentication  
**from** tastypie.authorization **import** DjangoAuthorization  
**from** tastypie.serializers **import** Serializer  
  
  
*# класс для управления загадками***class** RiddleResource(ModelResource):  
 *# описание* **class** Meta:  
 *# набор данных - все загадки из БД* queryset = Riddle.objects.all()  
 *# имя, которое нужно указывать в URL после /api/* resource\_name = **'riddle'** *# требуем указать логин и пароль в заголовках* authentication = BasicAuthentication()  
 *# права на добавление-обновление-удаление  
 # выдаем на основании логина и пароля - только админам* authorization = DjangoAuthorization()  
 *# стандартный класс для сохранения* serializer = Serializer()

*# класс для управления вариантами ответов***class** OptionResource(ModelResource):  
 *# ID загадки, к которой относится вариант ответа  
 # (переменной из модели), по умолчанию не выводится  
 # из-за того, что это внешний ключ* riddle\_id = fields.IntegerField(**'riddle\_id'**)

*# описание* **class** Meta:  
 *# набор данных - все пользователи из БД* queryset = Option.objects.all()  
 *# имя, которое нужно указывать в URL после /api/* resource\_name = **'option'** *# требуем прохождения авторизации* authentication = BasicAuthentication()  
 *# права на добавление-обновление-удаление  
 # выдаем на основании логина и пароля – только админам* authorization = DjangoAuthorization()  
 *# стандартный класс для сохранения* serializer = Serializer()

1. Для привязки созданных классов к URL-ам **допишем в существующем файле riddles/urls.py:**

*# старые import***from** django.conf.urls **import** re\_path  
**from** . **import** views  
*# новые import для API***from** django.conf.urls **import** include  
**from** riddles.api **import** RiddleResource, OptionResource  
**from** tastypie.api **import** Api  
  
*# api\_name будет указываться в URLах  
# перед resource\_name из api.py*api = Api(api\_name=**'api'**)  
api.register(RiddleResource())  
api.register(OptionResource())  
  
*# старое*app\_name = **'riddles'**urlpatterns = [  
 *# ...* re\_path(**r'^unsubscribe/$'**,

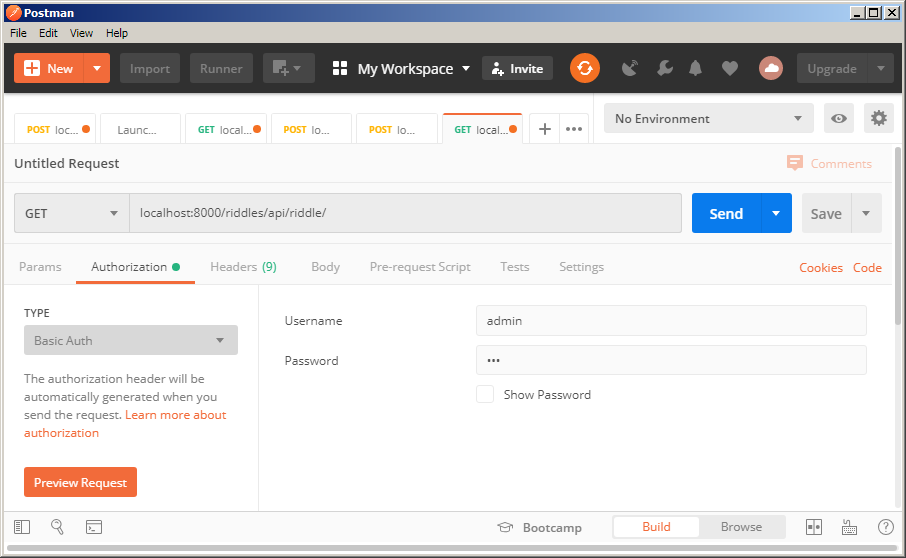
views.unsubscribe,

name=**'unsubscribe'**),  
 *# для API* re\_path(**r'^'**, include(api.urls)),  
]

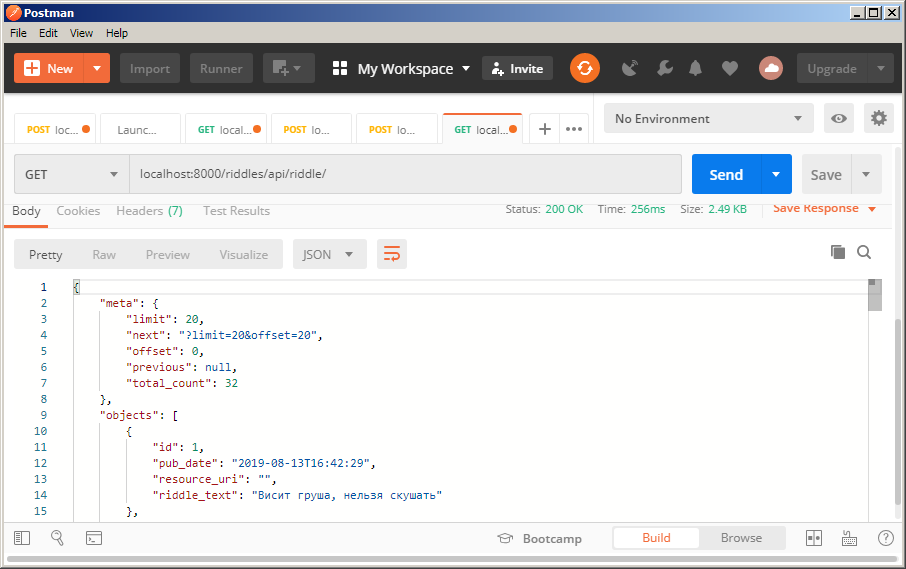
1. **Для тестирования API обычного браузера недостаточно – требуется использовать специальные программы, например, Postman – скачаем и установим его:**

https://www.postman.com/downloads/

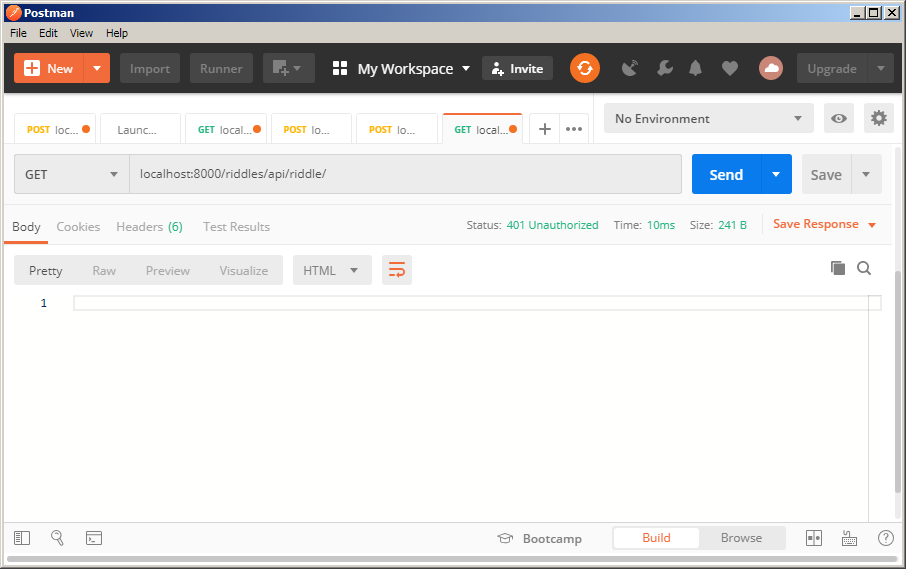
1. **Запустим приложение в PyCharm, запустим Postman, зайдем на вкладку Authorization, выберем тип аутентификации (передачи логина и пароля) «Basic Auth» (т.е. с помощью стандартных заголовков в HTTP-запросе), введем логин и пароль администратора:**

****

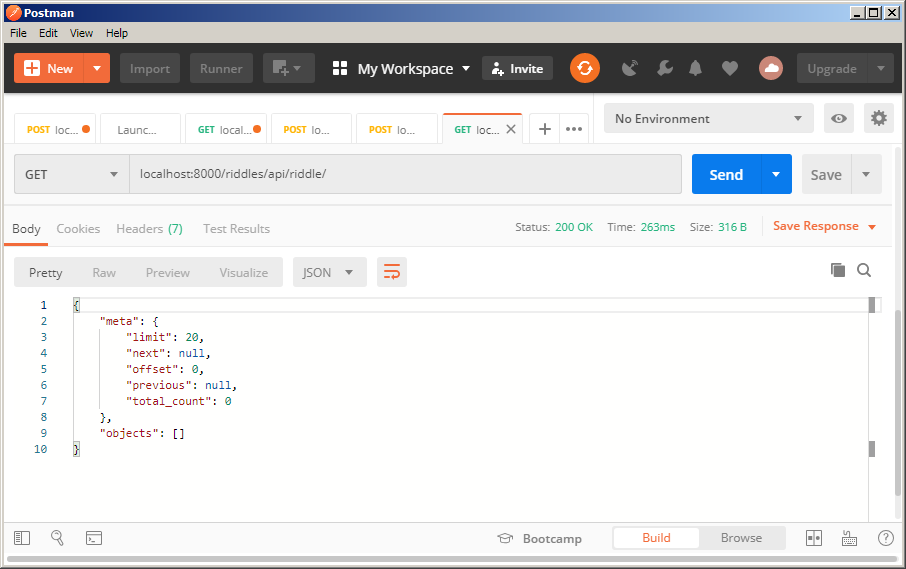
1. **Отправим GET-запрос на выборку всех загадок в Postman на адрес localhost:8000/riddles/api/riddle/ нажав синюю кнопку «Send», убедимся, что ответ имеет статус «200 ОК» и в ответе есть загадки:**

****

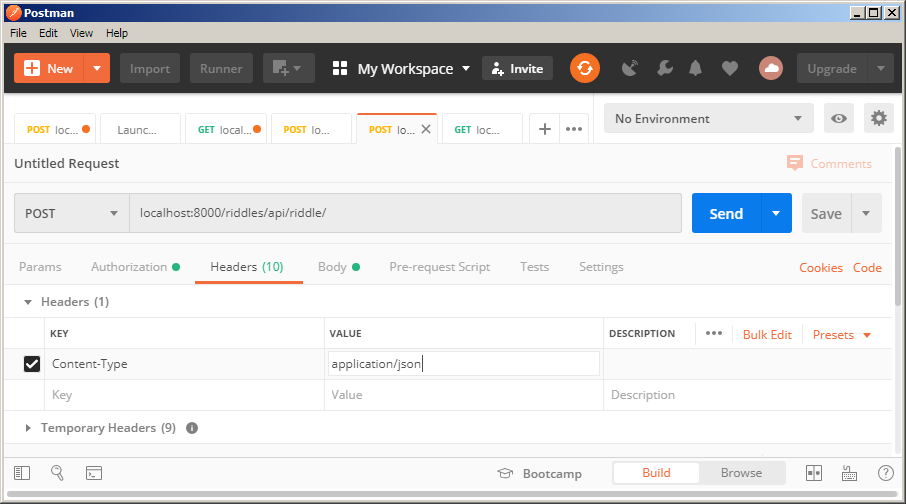
1. **Убедимся, что при отправке запросов с неправильным логином, паролем или типом аутентификации ответ придет пустым и со статусом «401 Unauthorized» – за это отвечает используемый в классах объект authentication = BasicAuthentication():**

****

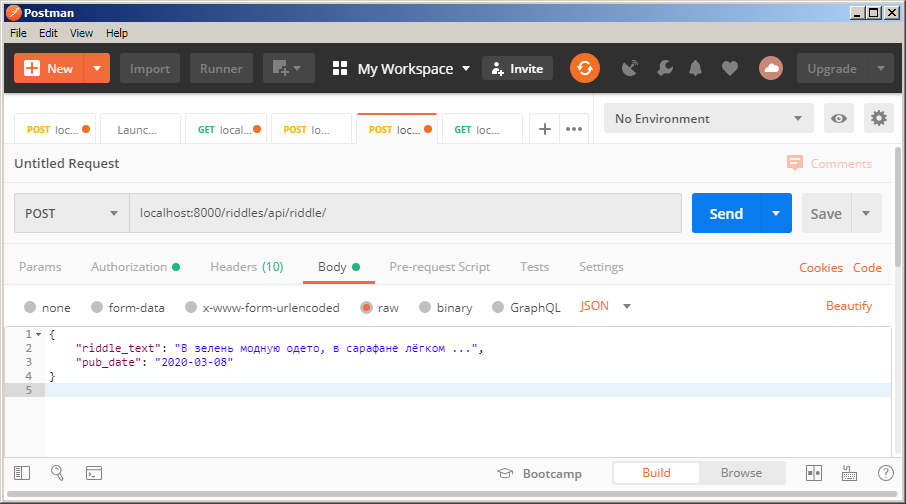
1. **Убедимся, что при отправке запросов с логином и паролем простого пользователя ответ придет с кодом «200 ОК», но без загадок, т.е. простым пользователям не разрешается использовать API, за это отвечает объект   
   authorization = DjangoAuthorization():**

****

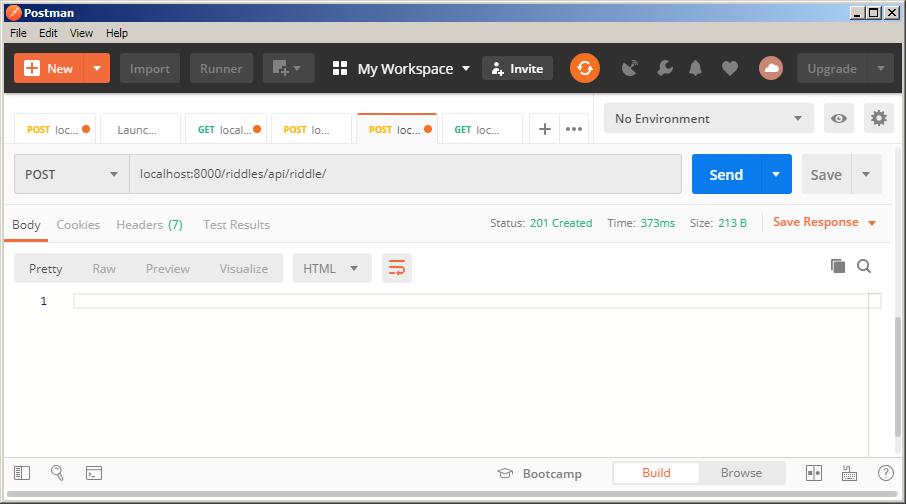
1. Наш API позволяет не только считывать объекты, но и записывать благодаря объекту serializer = Serializer(). Попробуем создать новую загадку с помощью API. **Для этого в Postman выберем метод POST, URL останется тем же (в этом преимущество API – не надо придумывать новые URL для каждой операции), на вкладке Headers добавим заголовок «Content-Type» со значением «application/json»:**

****

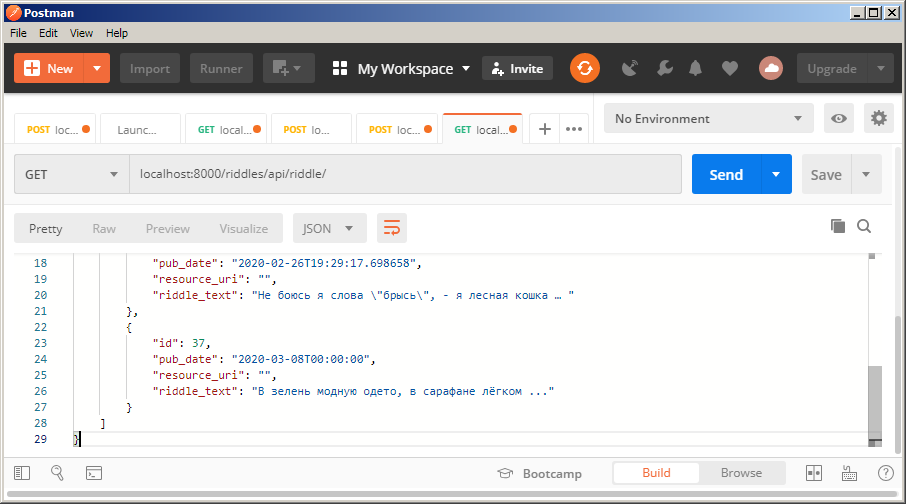
1. **Саму добавляемую загадку и дату добавления напишем на вкладке Body в режиме «raw» в формате JSON, который по умолчанию используется в пакете django-tastypie:**

****

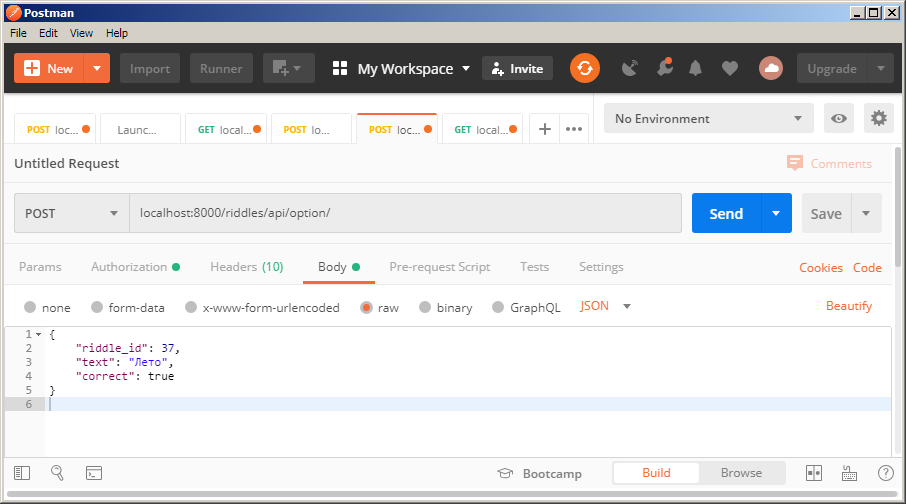
1. **Нажмем кнопку «Send», убедимся, что ответ пришел пустым и со статусом «201 Created»:**

****

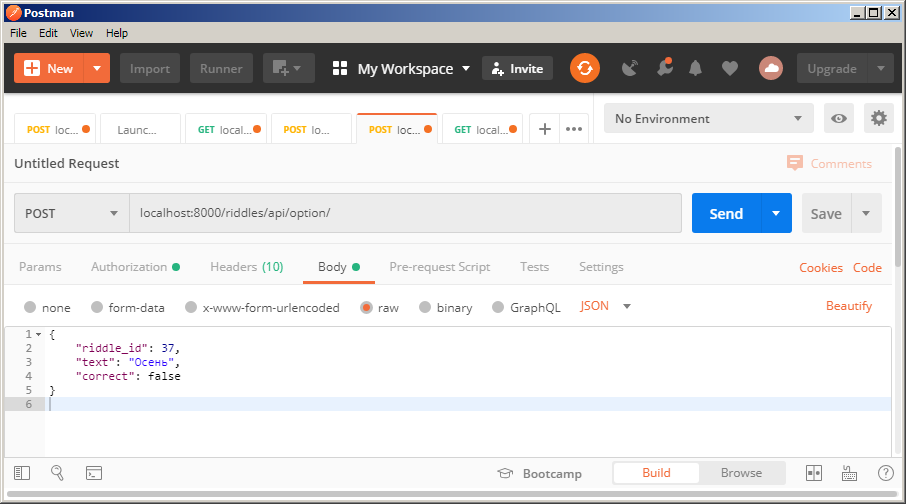
1. **Убедимся в появлении загадки с помощью старого GET-запроса, заодно узнаем ID, присвоенный новой загадке:**

****

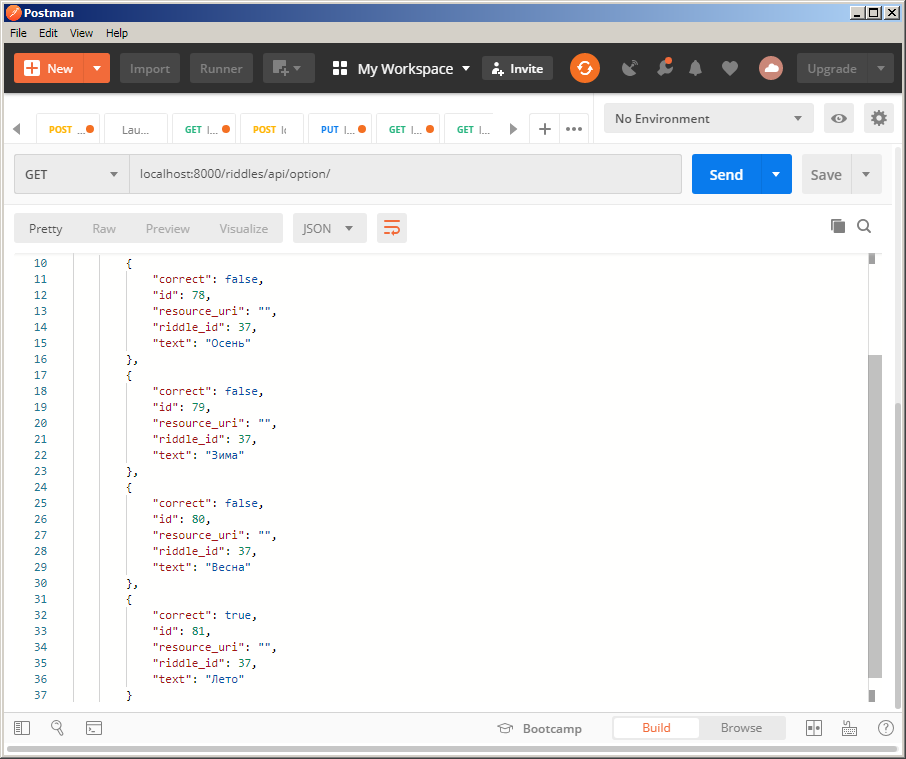
1. **Создадим POST-запрос к URL localhost:8000/riddles/api/option/ на добавление правильного варианта ответа к созданной загадке, за основу возьмем из истории POST-запрос на добавление загадки, чтобы сохранить параметры аутентификации и заголовок «Content-Type», поменяем только URL и Body, при отправке запроса должен прийти ответ с кодом «201 Created»:**

****

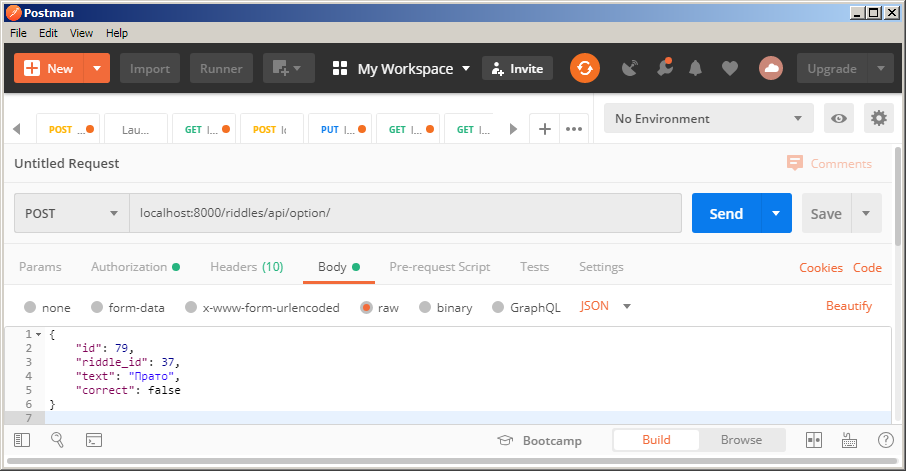
1. **Аналогично добавьте другие времена года в качестве неправильных вариантов:**

****

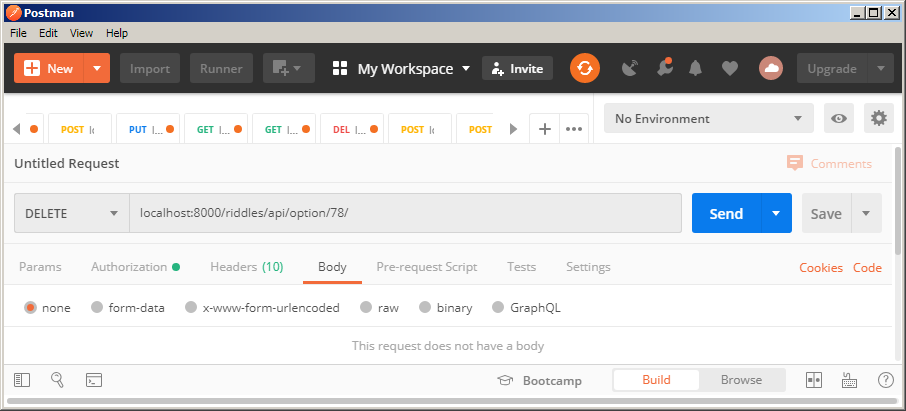
1. **Убедитесь в успешном добавлении вариантов с помощью GET-запроса к тому же URL, познакомьтесь с назначенными ID:**

****

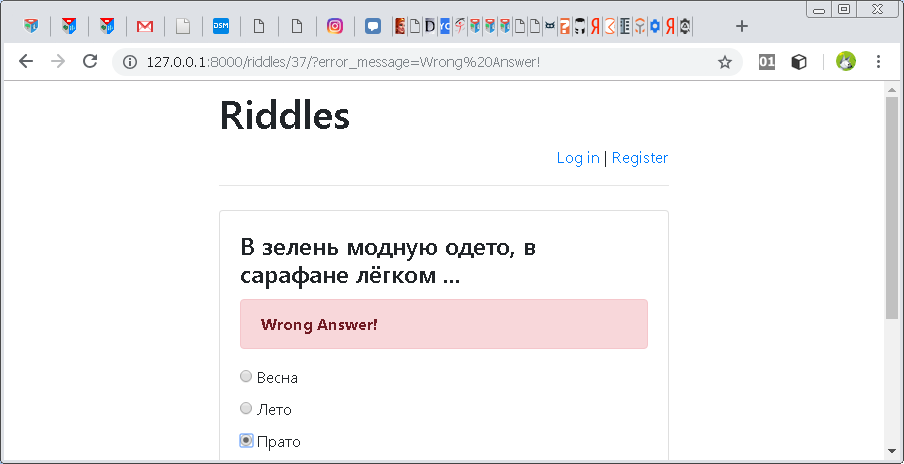
1. Вариант «Зима» не подходит ни по рифме, ни по смыслу. **Поэтому заменим его на более реалистичный вариант «прАто» (поле по-итальянски) с помощью POST-запроса с указанием присвоенного при прошлом POST-запросе ID, ожидаем кода ответа «201 Created»:**

****

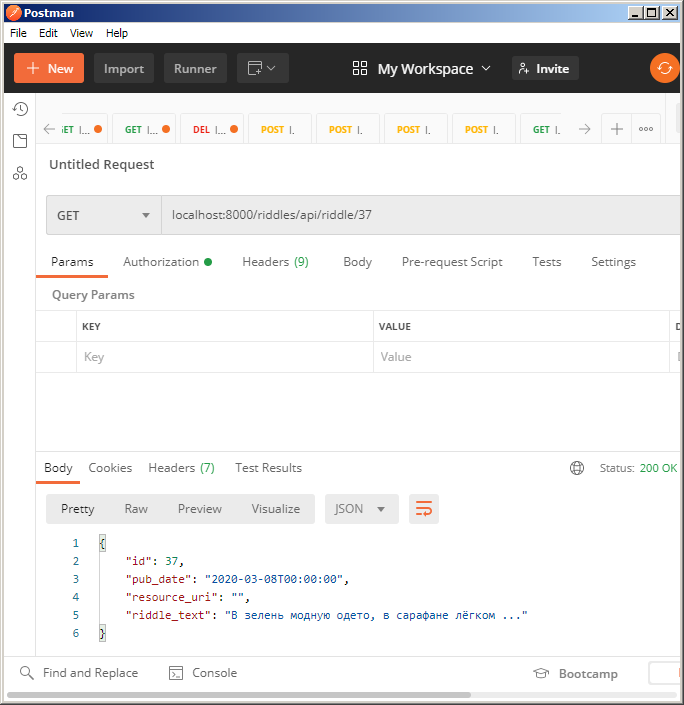
1. Вариант «Осень» тоже не подходит ни по рифме, ни по смыслу. **Поэтому удалим его с помощью DELETE-запроса с указанием присвоенного при POST-запросе ID в URL, ожидаем кода ответа «204 No content». Внимание! Будьте осторожны! Отправка DELETE-запроса к старому URL удаляет все варианты ответов всех загадок! Не проверяйте – опасно!**

****

1. **Убедитесь в корректном отображении загадки и вариантов ответа с помощью браузера:**



**20.\* Для отбора конкретной загадки по идентификатору, например, для размещенном сайте, следует использовать GET-запрос к URL localhost:8000/riddles/api/riddle/<ID загадки>:**



**21.\*** Для разрешения отбора ответов на конкретную загадку следует задать значение поля filtering в классе Meta внутри OptionResource **в конце файла riddles/api.py:**

class Meta:

# набор данных - все пользователи из БД

queryset = Option.objects.all()

# имя, которое нужно указывать в URL после /api/

resource\_name = 'option'

# требуем прохождения авторизации

authentication = BasicAuthentication()

# права на добавление-обновление-удаление

# выдаем на основании логина и пароля - только админам

authorization = DjangoAuthorization()

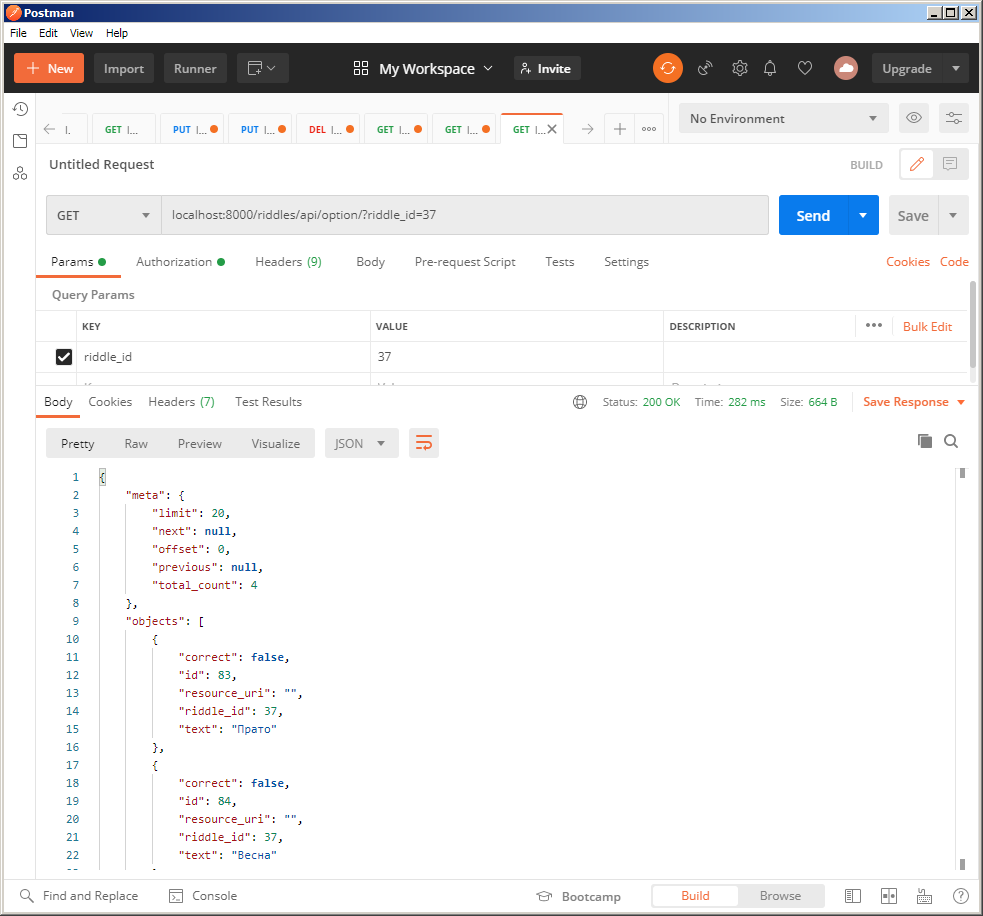
# стандартный класс для сохранения

serializer = Serializer()

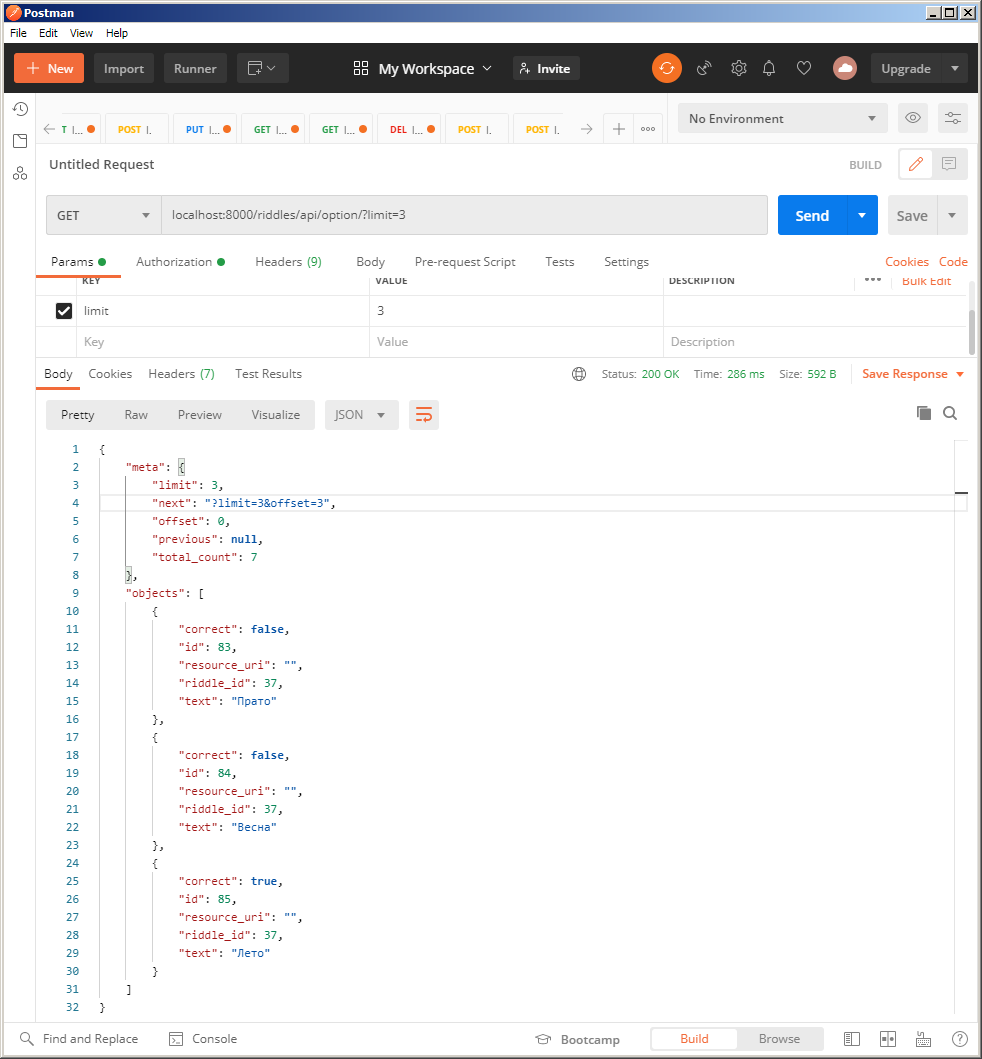
**# разрешаем отбор ответов по ID загадки**

**filtering = {"riddle\_id": ["exact", "in"]}**

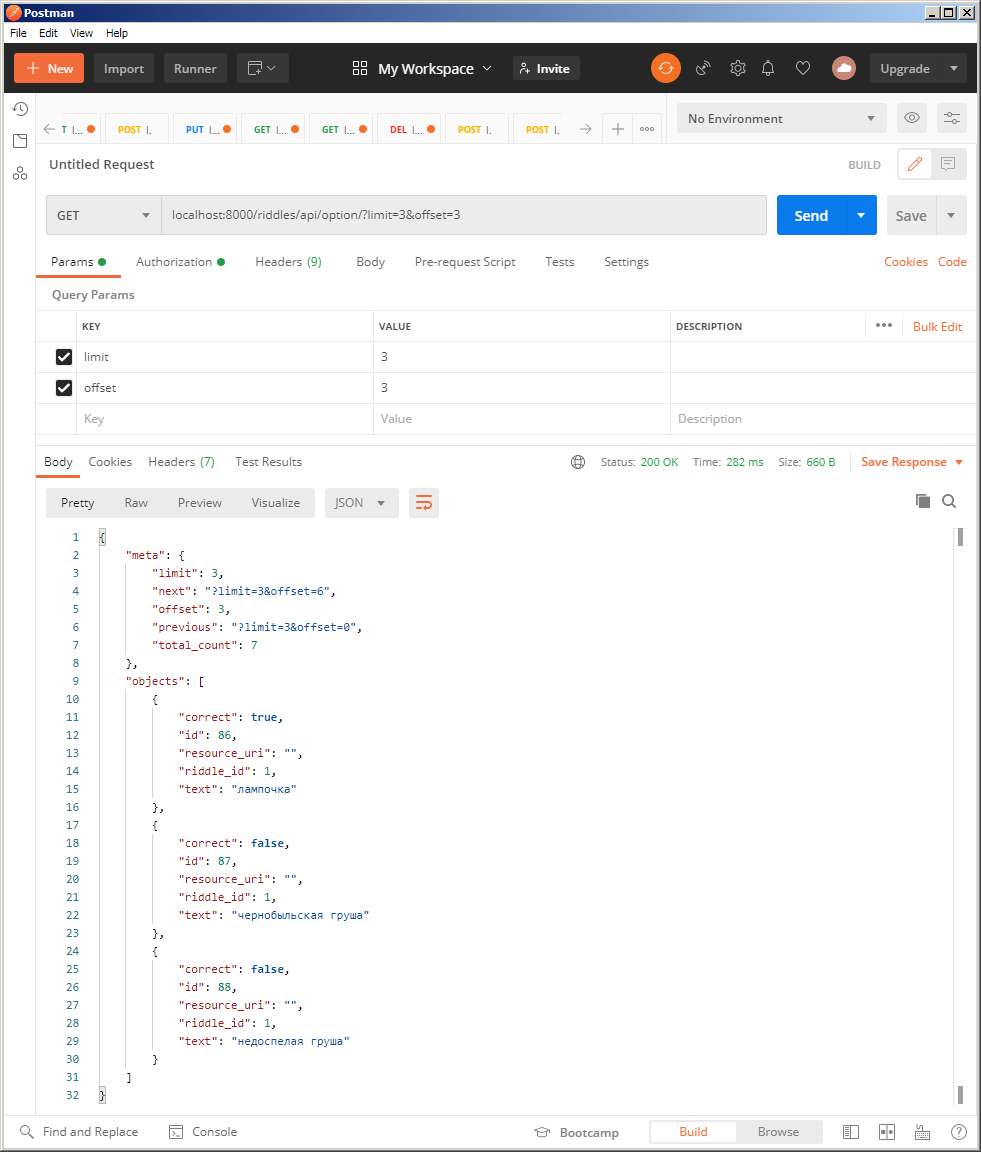
**22.\* Для отбора ответов на конкретную загадку следует использовать GET-запрос к URL localhost:8000/riddles/api/option/?riddle\_id=<ID загадки>:**



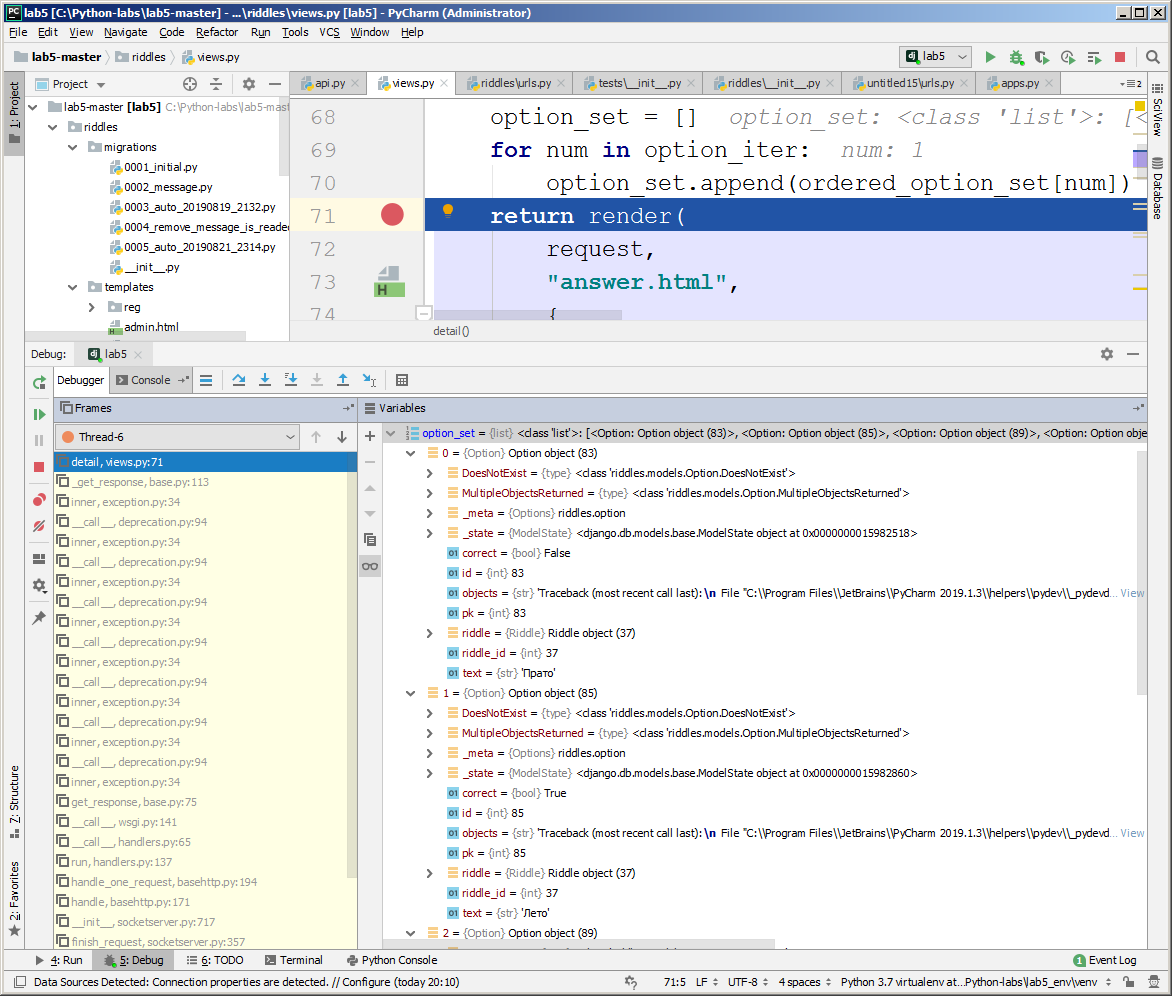
**23.\* Для разбиения длинных списков на страницы используются параметры GET-запросов limit и offset. Для отбора первых трех элементов списка ответов следует использовать GET-запрос к URL localhost:8000/riddles/api/option/?limit=3:**

****

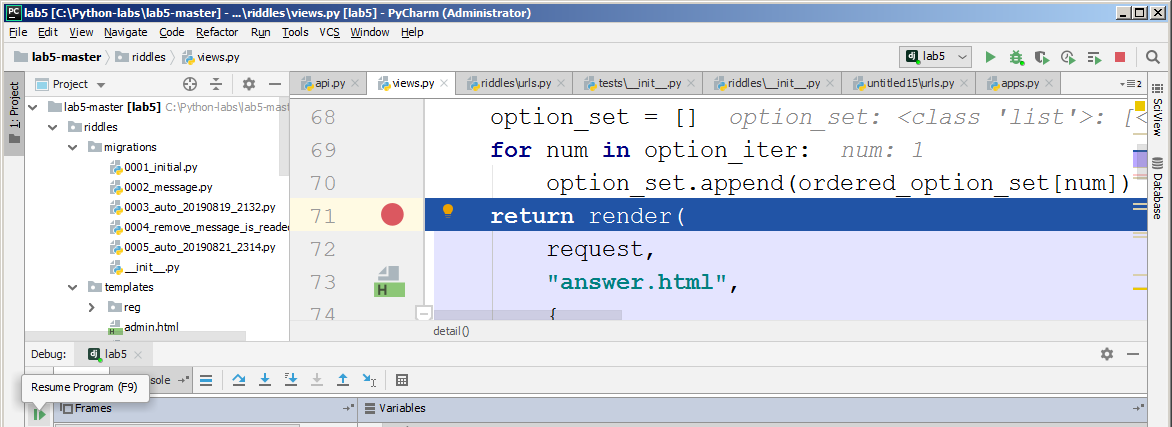
**24.\* Набор параметров для получения следующей страницы из трех элементов указан в поле meta.next:**

****

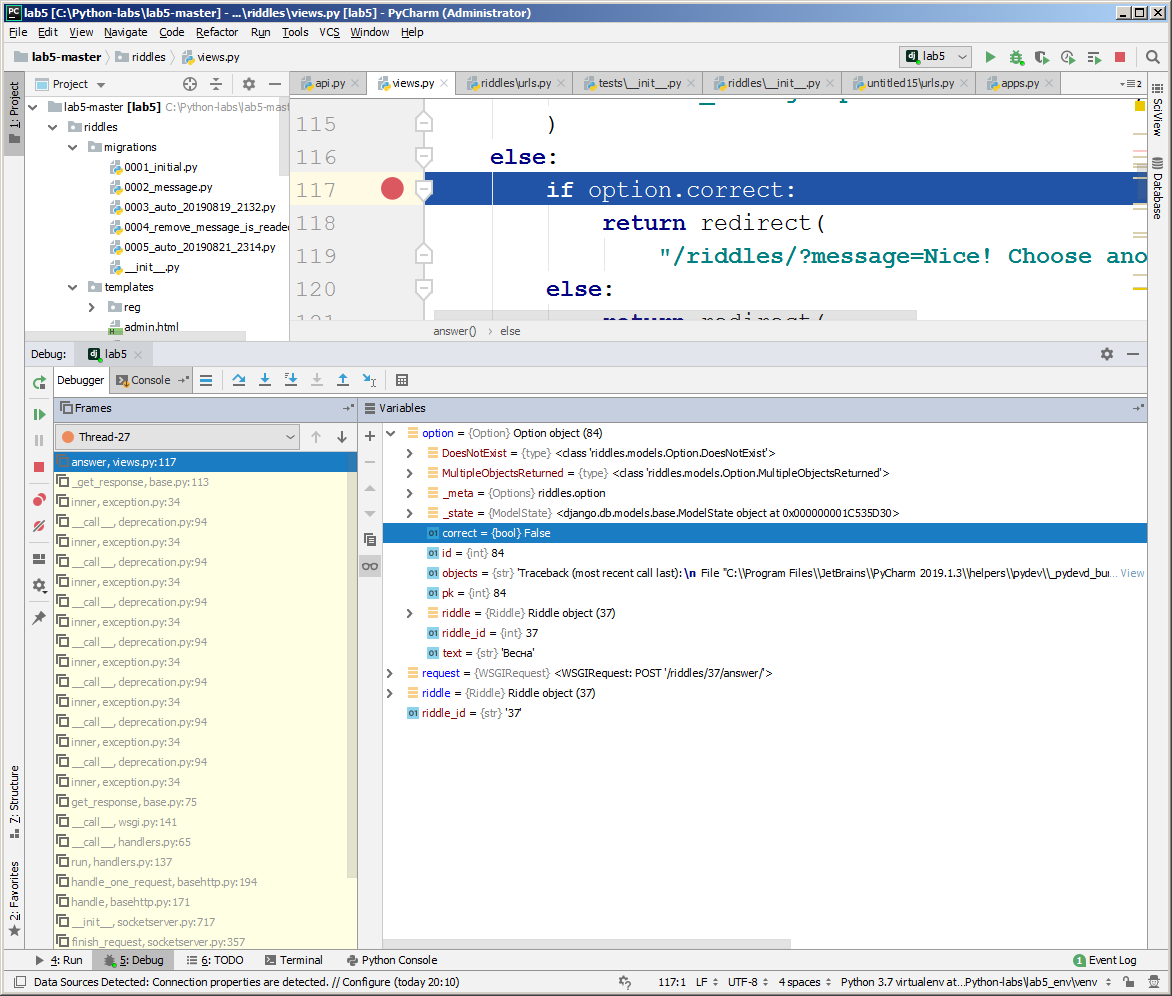
**25.\* Для исправления ошибок в работе программ нужно понимать, какие значения имеют переменные на конкретном этапе выполнения той или иной функции. Для получения значений переменных следует установить точки остановки (Breakpoints) щелчком на поле слева от строки кода, следующей за строками, в которых изменяется значение интересующих переменных. После расстановки точек останова следует запустить программу в режиме отладки (Debug), нажав вместо пиктограммы с треугольником на пиктограмму с жуком. Затем после запуска программы следует выполнить действия, которые приведут к вызову фрагментов кода, в которых были установлены точки остановки. При достижении точки остановки программа (в случае с веб-приложением – страница браузера) зависнет, позволяя разработчику переключиться в окно PyCharm и увидеть текущие значения переменных. Например, рассмотрим список выбранных из БД вариантов ответа на загадку в функции detail в файле riddles/views.py:**

****

**Для продолжения отладки следует нажать кнопку Resume Program (F9).**

****

**26.\* Для понимания, какая из ветвей кода будет выполняться после проверки условия, можно поставить точку остановки на условный оператор:**

****