1. Для выставления оценок пользователями **добавим новую модель Mark в riddles/models.py**:

**class** Mark(models.Model):  
 riddle = models.ForeignKey(  
 Riddle,  
 verbose\_name=**'Загадка'**,  
 on\_delete=models.CASCADE)  
 author = models.ForeignKey(  
 User,  
 verbose\_name=**'Пользователь'**)  
 mark = models.IntegerField(  
 verbose\_name=**'Оценка'**)  
 pub\_date = models.DateTimeField(  
 **'Дата оценки'**,  
 default=timezone.now)

1. Создадим миграцию – код на языке SQL для создания в БД таблицы, соответствующей нашей модели – напишем **в терминале**:

**python manage.py makemigrations riddles**

1. Выполним сгенерированный код на языке SQL – напишем **в терминале**:

**python manage.py migrate**

1. Дадим администраторам возможность подчищать созданную в БД таблицу – **допишем в существующий файл riddles/admin.py:**

**from** .models **import** Mark  
...  
**admin.site.register(Mark)**

1. Для добавления объектов созданного класса в БД **допишем новую функцию в существующем файле riddles/views.py:**

*# сообщения***from** .models **import** Mark

**...**

**def** post\_mark(request, riddle\_id):  
 msg = Mark()  
 msg.author = request.user  
 msg.riddle = get\_object\_or\_404(Riddle, pk=riddle\_id)  
 msg.mark = request.POST[**'mark'**]  
 msg.pub\_date = datetime.now()  
 msg.save()  
 **return** HttpResponseRedirect(app\_url+str(riddle\_id))

1. **Привяжем созданную функцию к соответствующему URL, на который будем присылать сообщения, в riddles/urls.py:**

urlpatterns = [  
 url(**r'^$'**, views.index, name=**'index'**),  
 url(**r'^([0-9]+)/$'**, views.detail, name=**'detail'**),  
 url(**r'^([0-9]+)/answer/$'**, views.answer, name=**'answer'**),

...  
 ***# отправка оценки* url(r'^([0-9]+)/post\_mark/$',  
 views.post\_mark,  
 name='post\_mark'),**  
]

1. Оценивать загадку должны иметь возможность только те, кто ее еще не оценивал. Для тех, кто поставил свою оценку, должна отображаться эта оценка. Для проверки выставленности оценки и выборки из БД этой оценки и средней по всем пользователям оценки загадки **допишем новые параметры в список параметров функции render в функции detail в существующем файле riddles/views.py:**

***# оценки*from .models import Mark  
*# вычисление среднего,  
# например, средней оценки*from django.db.models import Avg**

*...*

*# страница загадки со списком ответов***def** detail(request, riddle\_id):  
 error\_message = **None  
 if "error\_message" in** request.GET:  
 error\_message = request.GET[**"error\_message"**]  
 **return** render(  
 request,  
 **"answer.html"**,  
 {  
 **"riddle"**: get\_object\_or\_404(  
 Riddle, pk=riddle\_id),  
 **"error\_message"**: error\_message,  
 **"latest\_messages"**:  
 Message.objects  
 .filter(chat\_id=riddle\_id)  
 .order\_by(**'-pub\_date'**)[:5],  
 ***# кол-во оценок, выставленных пользователем* "already\_rated\_by\_user":  
 Mark.objects  
 .filter(author\_id=request.user.id)  
 .count(),  
 *# оценка текущего пользователя* "user\_rating":  
 Mark.objects  
 .filter(author\_id=request.user.id)  
 .filter(riddle\_id=riddle\_id)  
 .aggregate(Avg('mark'))  
 ["mark\_\_avg"],  
 *# средняя по всем пользователям оценка* "avg\_mark":  
 Mark.objects  
 .filter(riddle\_id=riddle\_id)  
 .aggregate(Avg('mark'))  
 ["mark\_\_avg"]**  
 }  
 )

1. Чтобы авторизованные пользователи, которые еще не оставляли оценку текущей загадке, могли ее оценить, **допишем новый фрагмент в шаблон** **riddles/templates/answer.html перед фрагментом с сообщениями (<h4>Chat</h4>):**

<**hr**>  
<**h4**>Rating</**h4**>  
 <**p**>Average rating:  
 <**span id="avg\_mark\_span"**>  
 {{ **avg\_mark** }}  
 </**span**>  
 </**p**>  
{% **if user**.**is\_authenticated** %}  
 {% **if already\_rated\_by\_user** == 0%}  
 <**form action="post\_mark/" method="post"**>  
 {% **csrf\_token** %}  
 <**div class="row"**>  
 <**div class="col-sm text-center"**>  
 <**input name="mark"  
 type="radio"  
 value="1"**/> 1  
 </**div**>  
 <**div class="col-sm text-center"**>  
 <**input name="mark"  
 type="radio"  
 value="2"**/> 2  
 </**div**>  
 <**div class="col-sm text-center"**>  
 <**input name="mark"  
 type="radio"  
 value="3"**/> 3  
 </**div**>  
 <**div class="col-sm text-center"**>  
 <**input name="mark"  
 type="radio"  
 value="4"**/> 4  
 </**div**>  
 <**div class="col-sm text-center"**>  
 <**input name="mark"  
 type="radio"  
 value="5"**/> 5  
 </**div**>  
 <**div class="col-sm text-center"**>  
 <**input class="btn btn-success"  
 type="submit"  
 value="Send mark"**/>  
 </**div**>  
 </**div**>  
 </**form**>  
 {% **else** %}  
 Your rating: {{ **user\_rating** }}  
 {% **endif** %}  
  
{% **endif** %}  
<**hr**>  
<**h4**>Chat</**h4**>

...

1. **Запустим приложение, авторизуемся, зайдем на страницу загадки, оценим загадку, убедимся, что оценить второй раз не сможем – вместо выбора новой оценки будет отображаться наша старая оценка.**
2. Плохо, средняя оценка обновляется только вместе со страницей – на посещаемом ресурсе новые загадки могут очень быстро набирать рейтинг – поэтому **допишем функцию, возвращающую среднюю оценку загадки в формате JSON, в файл riddles/views.py:**

**def** get\_mark(request, riddle\_id):  
 res = Mark.objects\  
 .filter(riddle\_id=riddle\_id)\  
 .aggregate(Avg(**'mark'**))  
  
 **return** JsonResponse(json.dumps(res), safe=**False**)

1. **Допишем URL, запросы к которому будут обрабатываться этой функцией, в urls.py:**

urlpatterns = [  
 url(**r'^$'**, views.index, name=**'index'**),  
 url(**r'^([0-9]+)/$'**, views.detail, name=**'detail'**),  
 url(**r'^([0-9]+)/answer/$'**, views.answer, name=**'answer'**),  
 *...*  
 *# отправка оценки* url(**r'^([0-9]+)/post\_mark/$'**,  
 views.post\_mark,  
 name=**'post\_mark'**),  
 ***# средняя оценка* url(r'^([0-9]+)/get\_mark/$',  
 views.get\_mark,  
 name='get\_mark'),**  
]

1. Допишем отправку асинхронных запросов каждые 5 сек и вывод ответов на них на страницу **в шаблон riddles/templates/answer.py в существующий тег <script> в конце:**

<**script**>  
**function** *send\_msg\_update\_request*(){  
 **$**.get(**"msg\_list/"**,{},*parse\_msg\_update\_response*);  
 **$.get("get\_mark/",{},*parse\_mark\_update\_response*);**  
}

//старая функция  
**function** *parse\_msg\_update\_response*(data, success){  
*...*  
}  
**function *parse\_mark\_update\_response*(data, success){res = *JSON*.parse(data);  
 $("#avg\_mark\_span").html(res["mark\_\_avg"].toFixed(1));  
}**  
*setInterval*(*send\_msg\_update\_request*, 5000);  
</**script**>

1. **Заходим в двух вкладках браузера на страницу одной и той же неоцененной загадки, ставим оценку на одной вкладке и смотрим, обновится ли средняя оценка на другой. Откройте консоль разработчика на F12 и посмотрите на вкладке Network, как отправляются запросы и что приходит в ответ (нажав на любой из запросов в списке).**