1. Для добавления новых загадок администраторами вместе с вариантами ответа **добавим новый шаблон riddles/templates/admin.html**:

{% **extends 'base.html'** %}  
{% **block main** %}  
 {% **if user**.**is\_staff** %}  
 {% **if message** %}  
 <**p class="alert alert-success"**>  
 <**strong**>{{ **message** }}</**strong**>  
 </**p**>  
 {% **endif** %}  
 <**div class="card mb-3"**>  
 <**div class="card-body"**>  
 <**form action="../post\_riddle/" method="post"**>  
 {% **csrf\_token** %}  
 <**h5**>Add riddle:</**h5**>  
 <**div class="mb-3"**>  
 <**input class="form-control"  
 placeholder="Riddle text"  
 name="text"  
 required**/>  
 </**div**>  
 <**div class="card mb-3"**>  
 <**div class="card-body" id="options\_new"**>  
 <**div class="mb-3"**>  
 <**input class="form-control"  
 placeholder="Option 1 (correct)"  
 name="option1"  
 required**/>  
 </**div**>  
 <**div class="mb-3"**>  
 <**input class="form-control"  
 placeholder="Option 2"  
 name="option2"  
 required**/>  
 </**div**>  
 <**script**>  
 **var *option\_counter*** = 2;  
 </**script**>  
 </**div**>  
 </**div**>  
 <**div class="mb-3"**>  
 <**button class="btn btn-success"  
 onclick="***addOption*(**'options\_new'**, 0)**"  
 type="button"**>  
 + Add option  
 </**button**>  
 <**button class="btn btn-success"**>  
 Save  
 </**button**>  
 </**div**>  
 <**input type="hidden"  
 name="riddle\_id"  
 value="0"**/>  
 </**form**>  
 </**div**>  
 </**div**>  
 <**div class="card mb-3"**>  
 <**div class="card-body"**>  
 <**h4 class="card-title"**>Available Riddles</**h4**>  
 {% **if latest\_riddles** %}  
 <**ul**>  
 {% **for riddle in latest\_riddles** %}  
 <**li**>  
 <**a href="/riddles/**{{ **riddle**.**id** }}**/"**>  
 {{ **riddle**.**riddle\_text** }}  
 </**a**>  
 </**li**>  
 {% **endfor** %}  
 </**ul**>  
 {% **else** %}  
 <**p**>No riddles are available right now.</**p**>  
 {% **endif** %}  
 </**div**>  
 </**div**>  
  
 <**script**>  
 *//добавление блока для нового ответа* **function** *addOption*(target, pos\_in\_array){  
 **var** option=***document***.createElement(**"div"**);  
 ***option\_counter*** += 1;  
 option.**className**=**"input-group mb-3"**;  
 option.**innerHTML**=  
 **'<input class="form-control"'** +  
 **' placeholder="Option '** +   
 ***option\_counter*** + **'"'** +   
 **' name="option'** + ***option\_counter*** + **'"/>'** +  
 **'<div class="input-group-append">'** +  
 **' <button class="btn btn-danger"'** +  
 **' type="button"'** +  
 **' onclick="removeParent('** +  
 **' this.parentNode)">'** +  
 **' Delete'** +  
 **' </button>'** +  
 **'</div>'**;  
 ***document***.getElementById(target).appendChild(option);  
 }  
 *//удаление блока ответа с поправкой номеров других ответов* **function** *removeParent*(child){  
 **var** optionList=child.**parentNode**.**parentNode**;  
 **var** currentOption=child.**parentNode**;  
 optionList.*removeChild*(currentOption);  
 **for**(**var** i=3; i<optionList.**children**.**length**; i+=1){  
 **var** opt=optionList.**children**[i].**children**[0];  
 opt.**name**=**"option"**+(i);  
 opt.**placeholder**=**"Option "**+(i);  
 }  
 ***option\_counter*** -= 1;  
 }  
 </**script**>  
 {% **else** %}  
 <**h4**>Access denied.</**h4**>  
 {% **endif** %}  
{% **endblock** %}

1. Для отображения администраторской панели добавления загадок **допишем новую функцию, аналогичную функции index, в существующем файле riddles/views.py:**

**def** admin(request):  
 message = **None  
 if "message" in** request.GET:  
 message = request.GET[**"message"**]  
 *# создание HTML-страницы по шаблону admin.html  
 # с заданными параметрами latest\_riddles и message* **return** render(  
 request,  
 **"admin.html"**,  
 {  
 **"latest\_riddles"**:  
 Riddle.objects.order\_by(**'-pub\_date'**)[:5],  
 **"message"**: message,  
 }  
 )

1. **Привяжем созданную функцию к соответствующему URL, на который будет заходить администратор, в riddles/urls.py:**

urlpatterns = [  
 *# ...*   
 re\_path(**r'^([0-9]+)/get\_mark/$'**,  
 views.get\_mark,  
 name=**'get\_mark'**),

re\_path(**r'^admin/$'**, views.admin, name=**'admin'**),   
]

1. Просто отобразить пустую форму добавления новой загадки недостаточно – необходимо реализовать обработку отправляемых заполненных форм. Для этого **допишем новую функцию в существующем файле riddles/views.py:**

**def** post\_riddle(request):  
 *# защита от добавления загадок неадминистраторами* author = request.user  
 **if not** (author.is\_authenticated **and** author.is\_staff):  
 **return** HttpResponseRedirect(app\_url+**"admin"**)  
 *# добавление загадки* rid = Riddle()  
 rid.riddle\_text = request.POST[**'text'**]  
 rid.pub\_date = datetime.now()  
 rid.save()  
 *# добавление вариантов ответа* i = 1 *# нумерация вариантов на форме начинается с 1*  
 *# количество вариантов неизвестно,  
 # поэтому ожидаем возникновение исключения,  
 # когда варианты кончатся* **try**:  
 **while** request.POST[**'option'**+str(i)]:  
 opt = Option()  
 opt.riddle = rid  
 opt.text = request.POST[**'option'**+str(i)]  
 opt.correct = (i == 1)  
 opt.save()  
 i += 1  
 *# это ожидаемое исключение,  
 # при котором ничего делать не надо* **except**:  
 **pass  
  
 return** HttpResponseRedirect(app\_url+str(rid.id))

1. **Привяжем созданную функцию к соответствующему URL, на который будут отправляться заполненные формы, в riddles/urls.py:**

urlpatterns = [  
 *# ...*

re\_path(**r'^admin/$'**, views.admin, name=**'admin'**),

re\_path(**r'^post\_riddle/$'**,   
 views.post\_riddle,   
 name=**’post\_riddle’**),  
]

1. **Запустим приложение, зайдем на адрес http://127.0.0.1:8000/riddles/admin/ , добавим 2 загадки с вариантами ответа, ответим на них правильно. Обратите внимание, что правильный вариант всегда оказывается первым в списке вариантов ответа.**
2. Неинтересно, когда правильный вариант при выборе ответа всегда оказывается первым из-за того, что стоит первым в форме добавления загадки! Реализуем перестановку вариантов ответа с помощью **модуля numpy – установим его к используемому интерпретатору через меню File – Settings.**
3. **Переделаем старую функцию detail с помощью установленного модуля numpy в файле riddles/views.py:**

**import** numpy

*# ...*

*# 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"**]  
 *# формируем список ответов* ordered\_option\_set = \  
 list(Option.objects.filter(riddle\_id=riddle\_id))  
 *# формируем случайный порядок номеров ответов* option\_iter = \  
 numpy.random.permutation(len(ordered\_option\_set))  
 *# формируем новый список, в который выписываем ответы  
 # в сформированном случайном порядке* option\_set = []  
 **for** num **in** option\_iter:  
 option\_set.append(ordered\_option\_set[num])  
 **return** render(  
 request,  
 **"answer.html"**,  
 {  
 *# передаем список ответов в случайном порядке*  
 **"option\_set"**: option\_set,  
  
 **"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)  
 .filter(riddle\_id=riddle\_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. **Подставим сформированный список ответов в случайном порядке в форму ответа в шаблоне riddles/templates/answer.html:**

...

<**form action="answer/" method="post"**>  
 {% **csrf\_token** %}  
 {% **for option in option\_set** %}  
 <**input type="radio"  
 name="option"  
 id="option**{{ **forloop**.**counter** }}**"  
 value="**{{ **option**.**id** }}**"**/>  
 <**label for="option**{{ **forloop**.**counter** }}**"**>  
 {{ **option**.**text** }}  
 </**label**><**br**>  
 {% **endfor** %}  
 <**input type="submit" value="Answer"**/>  
</**form**>

...

1. **Запустим приложение, зайдем на страницу любой загадки, несколько раз обновим страницу в браузере. Обратите внимание, что при каждом обновлении страницы порядок вариантов ответа изменяется.**
2. Администраторам неудобно набирать адрес панели добавления загадок в адресной строке – поэтому **допишем ссылку на панель добавления загадки, которая будет появляться только у администраторов, после ссылок Log out и Change password в шаблоне riddles/templates/base.html:**

...

<**p class="text-right"**>  
 Hello, {{ **user**.**username** }}!  
 <**a href="/riddles/logout"**>Log out</**a**> |  
 <**a href="/riddles/password-change"**>Change password</**a**>  
 {% **if user**.**is\_staff** %}  
 | <**a href="/riddles/admin"**>Admin panel</**a**>  
 {% **endif** %}  
</**p**>

...

1. **Заходим в приложение под пользователем-администратором, проверяем наличие ссылки, переходим по ней, чтобы убедится, что откроется страница добавления загадки, выходим из учетной записи администратора и заходим под обычным пользователем, проверяем, что ссылка на администраторскую панель не отображается.**
2. О новых добавленных загадках нужно уведомлять пользователей. Для этого нужно предоставить им возможность подписываться на рассылку писем, указав свой E-mail. **Создадим новый шаблон страницы подписки на рассылку riddles/templates/subscribe.html:**

{% **extends 'base.html'** %}  
{% **load crispy\_forms\_tags** %}  
{% **block main** %}  
 <**form action="" method="post"**>  
 {% **csrf\_token** %}  
 {{ **form**|**crispy** }}  
 <**button type="submit"  
 class="btn btn-success btn-block"**>  
 Subscribe</**button**>  
 <**a href=".." class="btn btn-danger btn-block"**>  
 Back to home  
 </**a**>  
 </**form**>  
{% **endblock** %}

1. Мы уже использовали готовые классы с логикой для форм регистрации, авторизации и смены пароля. Для этих классов логики мы доделывали свои классы-представления для взаимодействия форм на страницах веб-приложения с этой логикой. Такие классы позволяют отображать форму (одно действие) и обрабатывать введенные в нее данные (второе действие) на одном URL благодаря использованию разных *методов* – дополнительной характеристики запросов, отправляемых от браузера к веб-серверу:

– при обращении к URL по ссылке или из строки браузера используется метод GET – при таком обращении отображается форма;

– при отправке формы нажатием на кнопку выполняется запрос к тому же URL (в шаблоне выше за это отвечает <form action=""), но с методом POST (в шаблоне выше за это отвечает method="post">) и с передачей введенных в форму данных для их обработки.

Для формирования формы подписки и обработки введенных в нее данных **добавим 2 своих класса, один – отвечающий за логику обработки формы и второй – за взаимодействие со страницами веб-приложения, в riddles/views.py:**

**from** django **import** forms  
**from** django.utils.translation **import** gettext, gettext\_lazy **as** \_  
*# ...  
  
  
# класс, описывающий логику формы:  
# список заполняемых полей и их сохранение***class** SubscribeForm(forms.Form):  
 *# поле для ввода e-mail* email = forms.EmailField(  
 label=\_(**"E-mail"**),  
 required=**True**,  
 )  
  
 *# конструктор для запоминания пользователя,  
 # которому задается e-mail* **def** \_\_init\_\_(self, user, \*args, \*\*kwargs):  
 self.user = user  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
  
 *# сохранение e-mail* **def** save(self, commit=**True**):  
 self.user.email = self.cleaned\_data[**"email"**]  
 **if** commit:  
 self.user.save()  
 **return** self.user  
  
  
*# класс, описывающий взаимодействие логики  
# со страницами веб-приложения***class** SubscribeView(FormView):  
 *# используем класс с логикой* form\_class = SubscribeForm  
 *# используем собственный шаблон* template\_name = **'subscribe.html'** *# после подписки возвращаем на главную станицу* success\_url = app\_url  
  
 *# передача пользователя для конструктора класса с логикой* **def** get\_form\_kwargs(self):  
 kwargs = super().get\_form\_kwargs()  
 kwargs[**'user'**] = self.request.user  
 **return** kwargs  
  
 *# вызов логики сохранения введенных данных* **def** form\_valid(self, form):  
 form.save()  
 **return** HttpResponseRedirect(self.success\_url)

1. **Привяжем класс-представление к URL в riddles/urls.py:**

urlpatterns = [  
 *# ...*

re\_path(**r'^post\_riddle/$'**,   
 views.post\_riddle,   
 name=**’post\_riddle’**),  
 re\_path(**r'^subscribe/$'**, views.SubscribeView.as\_view()),  
]

1. Чтобы пользователям было удобно попадать на страницу подписки **допишем ссылку на эту страницу, которая будет появляться у неподписавшихся пользователей после ссылок Log out и Change password в шаблоне riddles/templates/base.html:**

<**p class="text-right"**>  
 Hello, {{ **user**.**username** }}!  
 <**a href="/riddles/logout"**>Log out</**a**> |  
 <**a href="/riddles/password-change"**>Change password</**a**>  
 {% **if user**.**is\_staff** %}  
 | <**a href="/riddles/admin"**>Admin panel</**a**>  
 {% **endif** %}  
 {% **if user**.**email** == **''** %}  
 | <**a href="/riddles/subscribe"**>Subscribe</**a**>

{% **endif** %}  
</**p**>

1. **Заходим в приложение под простым пользователем, проверяем наличие ссылки, переходим по ней, указываем e-mail, отправляем форму, убеждаемся, что ссылка «Subscribe» исчезла, проверяем через DB Browser for SQLite наличие у пользователя в таблице auth\_user указанного e-mail.**
2. Чтобы подписанные пользователи могли отписаться не нужна форма для ввода чего-либо. Поэтому сразу начнем не с создания шаблона, а с **добавления функции в riddles/views.py:**

*# функция для удаления подписки (форма не нужна,  
# поэтому без классов, просто функция)***def** unsubscribe(request):  
 request.user.email = **''** request.user.save()  
 **return** HttpResponseRedirect(app\_url)

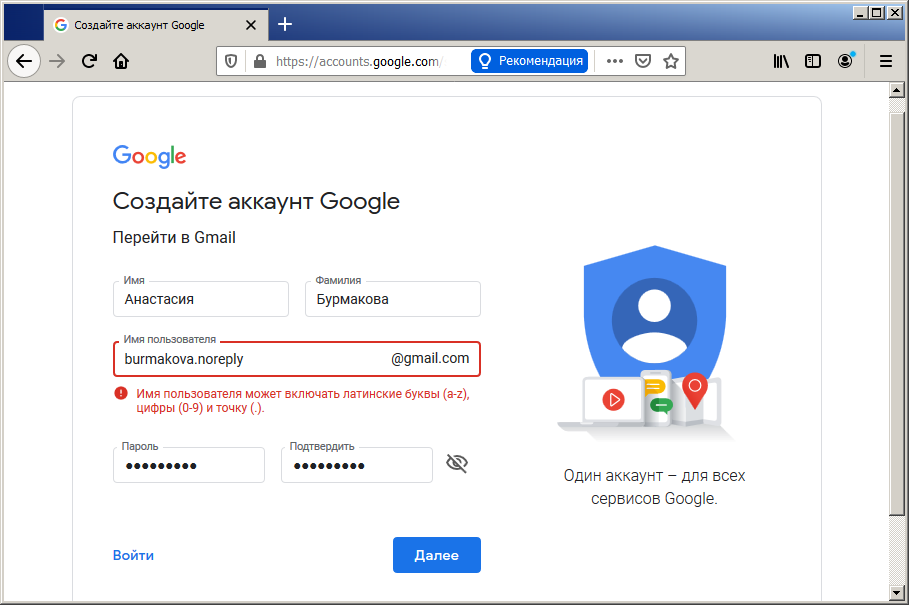
1. **Привяжем функцию к URL в riddles/urls.py:**

urlpatterns = [  
 *# ...*  
 re\_path(**r'^subscribe/$'**, views.SubscribeView.as\_view()),  
 re\_path(**r'^unsubscribe/$'**, views.unsubscribe, name=**'unsubscribe'**),  
]

1. **Добавим подписавшимся ссылку для отписки вместо ссылки для подписки в шаблон** **riddles/templates/base.html:**

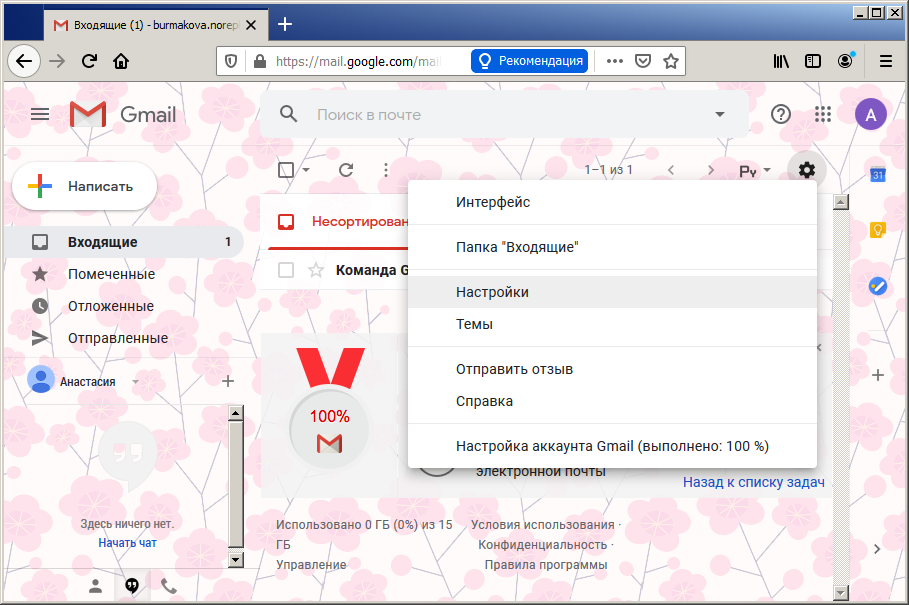
<**p class="text-right"**>  
 Hello, {{ **user**.**username** }}!  
 <**a href="/riddles/logout"**>Log out</**a**> |  
 <**a href="/riddles/password-change"**>Change password</**a**>  
 {% **if user**.**is\_staff** %}  
 | <**a href="/riddles/admin"**>Admin panel</**a**>  
 {% **endif** %}  
 {% **if user**.**email** == **''** %}  
 | <**a href="/riddles/subscribe"**>Subscribe</**a**>  
 {% **else** %}  
 | <**a href="/riddles/unsubscribe"**>Unsubscribe</**a**>  
 {% **endif** %}  
</**p**>

1. **Заходим в приложение под тем же пользователем, нажимаем ссылку «Unsubscribe», проверяем через DB Browser for SQLite отсутствие в таблице auth\_user у пользователя e-mail.**
2. **Для рассылки писем пользователям зарегистрируем отдельный почтовый ящик на gmail.com:**

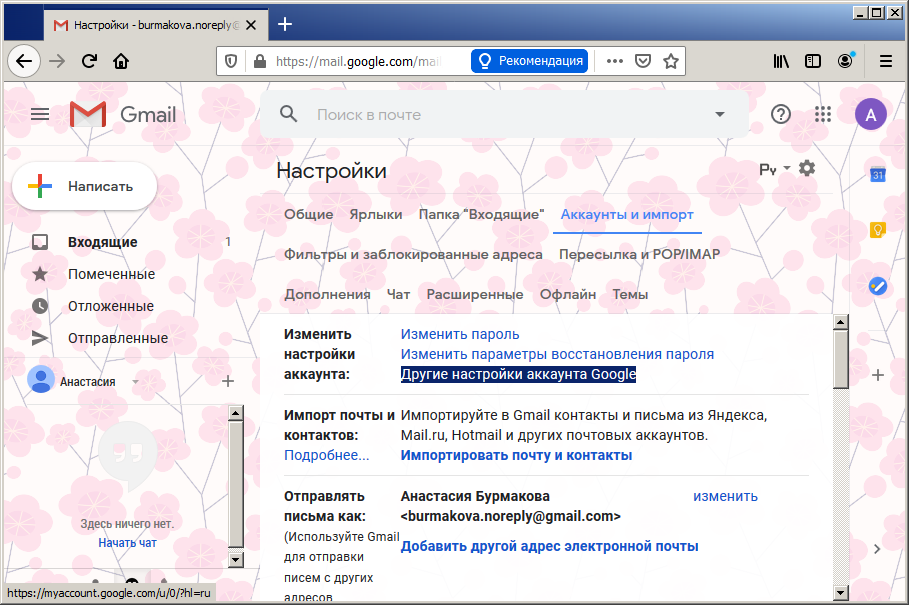
****

1. **Разрешим доступ к ящику небезопасных приложений:**

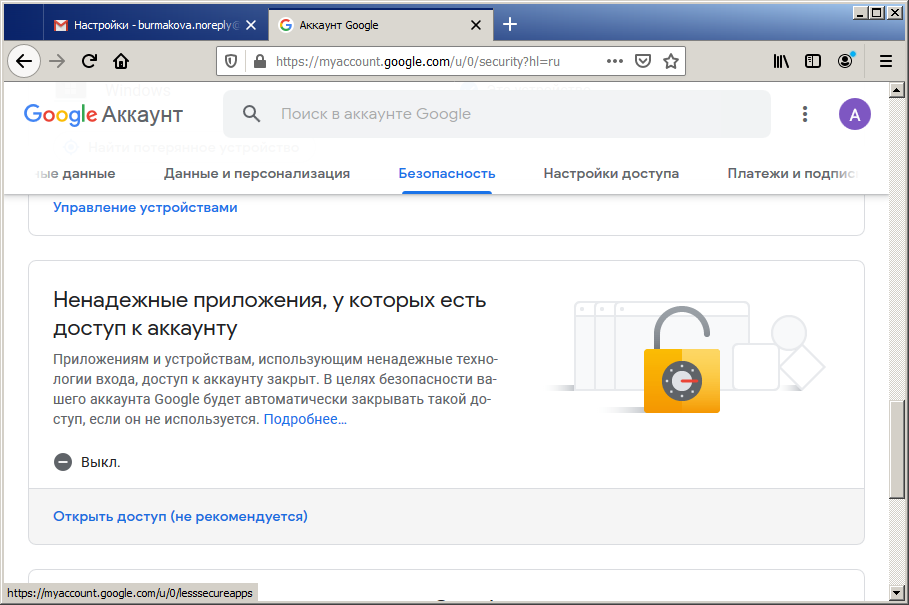
**«Шестеренка» – «Настройки»**

****

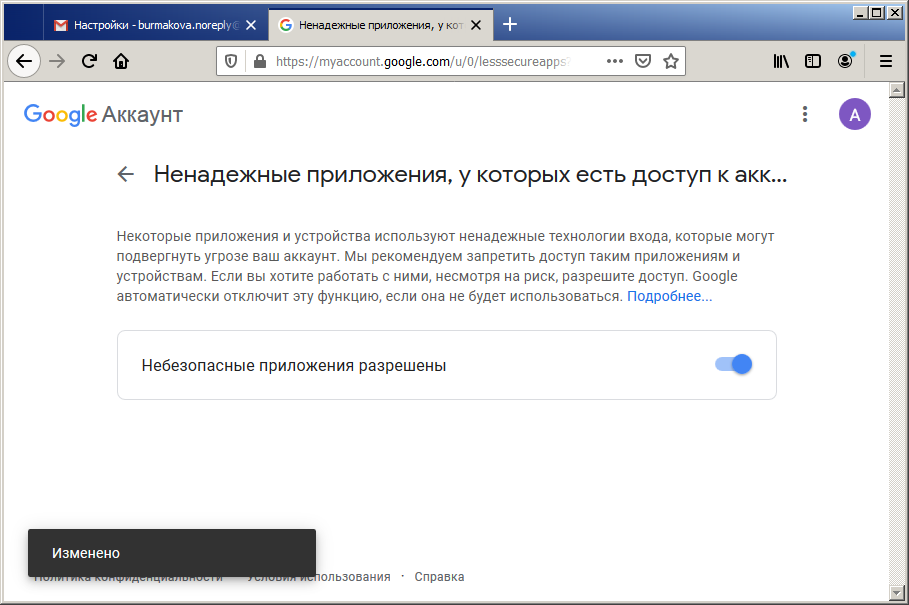
**«Аккаунты и импорт» – «Другие настройки аккаунта Google»**

****

**«Безопасность» – «Ненадежные приложения, у которых есть доступ к аккаунту» – «Открыть доступ (не рекомендуется)»**

****

**Переведем переключатель во включенное положение.**

****

1. **Настроим параметры для отправления писем в конце файла settings.py:**

*# настройка отправки почты -  
# параметры подключения к почтовому SMTP-серверу*EMAIL\_HOST = **'smtp.gmail.com'**EMAIL\_HOST\_USER = **'Ваш только что зарегистрированный ящик'**EMAIL\_HOST\_PASSWORD = **'Ваш пароль'**EMAIL\_PORT = 465  
EMAIL\_USE\_SSL = **True**

1. **Добавим рассылку писем всем подписавшимся при добавлении загадки в конце функции post\_riddle (перед return HttpResponseRedirect(app\_url+str(rid.id))) в riddles/views.py:**

*# цикл по всем пользователям***for** i **in** User.objects.all():  
 *# проверка, что текущий пользователь подписан - указал e-mail* **if** i.email != **''**:  
 send\_mail(  
 *# тема письма* **'New riddle'**,  
 *# текст письма* **'A new riddle was added on riddles portal:\n'** +  
 **'http://localhost:8000/riddles/'** + str(rid.id) + **'.'**,  
 *# отправитель* **'Ваш только что зарегистрированный ящик'**,  
 *# список получателей из одного получателя* [i.email],  
 *# отключаем замалчивание ошибок,  
 # чтобы из видеть и исправлять* **False** )

**return** HttpResponseRedirect(app\_url+str(rid.id))

1. **Заходим в приложение под простым пользователем, подписываемся на рассылку, выходим. Заходим в приложение под администратором, добавляем загадку, проверяем наличие письма со ссылкой на загадку в почтовом ящике подписавшегося.**