1. **Создадим** единообразное оформление страниц приложения с помощью общего HTML-шаблона страниц **riddles/templates/base.html**, в который в отмеченном месте будем подставлять уникальное содержание страницы. Кроме того, этот общий шаблон страниц будет содержать общую шапку, в которой будут размещаться:

− ссылки на выход и смену пароля для авторизированных пользователей   
({% **if user**.**is\_authenticated** %});

− ссылки на страницы регистрации и авторизации для не вошедших пользователей:

<!doctype **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="utf-8"**>  
 <**meta name="viewport"**

**content="width=device-width, initial-scale=1, shrink-to-fit=no"**>  
 <**link rel="stylesheet"**

**href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"**>  
 <**title**>Riddles</**title**>  
</**head**>  
<**body**>  
<**div class="container"**>  
 <**div class="row justify-content-center"**>  
 <**div class="col-8"**>  
 <**h1 class="mt-2"**>Riddles</**h1**>  
 {% **if user**.**is\_authenticated** %}  
 <**p class="text-right"**>  
 Hello, {{ **user**.**username** }}!  
 <**a href="/riddles/logout"**>Log out</**a**> |  
 <**a href="/riddles/password-change"**>

Change password

</**a**>  
 </**p**>  
 {% **else** %}  
 <**p class="text-right"**>  
 <**a href="/riddles/login"**>Log in</**a**> |  
 <**a href="/riddles/register"**>Register</**a**>  
 </**p**>  
 {% **endif** %}  
 <**hr class="mt-0 mb-4"**>  
 <**div class="card"**>  
 <**div class="card-body"**>  
 {% **block main** %}  
 {% **endblock** %}  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>  
</**body**>  
</**html**>

1. **Заменим весь текст имеющихся шаблонов страниц** так, чтобы задействовать общий шаблон страниц страниц, применим CSS-классы Bootstrap alert-success и alert-danger:

**index.html**:

{% **extends 'base.html'** %}  
{% **block main** %}  
 <**h4 class="card-title"**>Available Riddles</**h4**>  
 {% **if message** %}  
 **<p class="alert alert-success">** <**strong**>{{ **message** }}</**strong**>  
 </**p**>  
 {% **endif** %}  
 {% **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** %}  
{% **endblock** %}

**answer.html**:

{% **extends 'base.html'** %}  
{% **block main** %}  
<**h4 class="card-title"**>{{ **riddle**.**riddle\_text** }}</**h4**>  
{% **if error\_message** %}  
 **<p class="alert alert-danger">** <**strong**>{{ **error\_message** }}</**strong**>  
 </**p**>  
{% **endif** %}  
<**form action="answer/" method="post"**>  
 {% **csrf\_token** %}  
 {% **for option in riddle**.**option\_set**.**all** %}  
 <**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**>  
{% **endblock** %}

1. В Django есть готовые формы для регистрации и авторизации, но для их оформления потребуется **установить через Settings дополнительный модуль django-crispy-forms.**
2. Подключим и настроим приложение **django-crispy-forms в 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'**]  
...

**CRISPY\_TEMPLATE\_PACK = 'bootstrap4'**

1. **Создадим папку** **riddles/templates/reg** для шаблонов страниц, связанных с регистрацией и авторизацией пользователей.
2. В первую очередь для обеспечения возможности входа пользователей в веб-приложение нужно дать им возможность регистрации. Для удобства разработки (отсутствие выделенных ошибок в текущем файле при переходе к следующему, подсказка при вводе названий классов, созданных в предыдущих файлах) вначале **создадим HTML-шаблон riddles/templates/reg/register.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"**>  
 Register</**button**>  
 <**a href=".." class="btn btn-danger btn-block"**>  
 Back to home</**a**>  
</**form**>  
{% **endblock** %}

1. Теперь **добавим в riddles/views.py новый класс**, который будет отвечать за заполнение шаблона страницы регистрации:

*# Базовый класс для обработки страниц с формами.***from** django.views.generic.edit **import** FormView  
*# Спасибо django за готовую форму регистрации.***from** django.contrib.auth.forms **import** UserCreationForm

*...*

*# базовый URL приложения, главной страницы -  
# часто нужен при указании путей переадресации*app\_url = **"/riddles/"***# наше представление для регистрации***class** RegisterFormView(FormView):  
 *# будем строить на основе  
 # встроенной в django формы регистрации* form\_class = UserCreationForm  
 *# Ссылка, на которую будет перенаправляться пользователь  
 # в случае успешной регистрации.  
 # В данном случае указана ссылка на  
 # страницу входа для зарегистрированных пользователей.* success\_url = app\_url + **"login/"** *# Шаблон, который будет использоваться  
 # при отображении представления.* template\_name = **"reg/register.html"  
 def** form\_valid(self, form):  
 *# Создаём пользователя,  
 # если данные в форму были введены корректно.* form.save()  
 *# Вызываем метод базового класса* **return** super(RegisterFormView, self).form\_valid(form)

1. И, наконец, привяжем созданный класс к соответствующему URL – **допишем строчку в riddles/urls.py**:

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

1. **Запустим проект, перейдем по ссылке на страницу регистрации, зарегистрируем пользователя, с помощью DB Browser посмотрим, появился ли пользователь в таблице auth\_user.**
2. В том же порядке, что и для страницы регистрации, создадим компоненты для страницы авторизации. Начнем с **создания HTML-шаблона** **riddles/templates/reg/login.html**:

{% **extends 'base.html'** %}  
{% **load crispy\_forms\_tags** %}  
{% **block main** %}  
<**h4 class="card-title"**>Log in to your account</**h4**>  
<**form method="post"**>  
 {% **csrf\_token** %}  
 <**input type="hidden" name="next" value="**{{ **next** }}**"**>  
 {{ **form**|**crispy** }}  
 <**button type="submit" class="btn btn-success btn-block"**>  
 Log in  
 </**button**>  
 <**a href=".." class="btn btn-danger btn-block"**>  
 Back to home  
 </**a**>  
</**form**>  
{% **endblock** %}

1. Теперь **добавим в riddles/views.py новый класс**, который будет отвечать за заполнение шаблона страницы авторизации:

*# Спасибо django за готовую форму аутентификации.***from** django.contrib.auth.forms **import** AuthenticationForm  
*# Функция для установки сессионного ключа.  
# По нему django будет определять,   
# выполнил ли вход пользователь.***from** django.contrib.auth **import** login

...

*# наше представление для входа***class** LoginFormView(FormView):  
 *# будем строить на основе  
 # встроенной в django формы входа* form\_class = AuthenticationForm  
 *# Аналогично регистрации,  
 # только используем шаблон аутентификации.* template\_name = **"reg/login.html"** *# В случае успеха перенаправим на главную.* success\_url = app\_url  
 **def** form\_valid(self, form):  
 *# Получаем объект пользователя  
 # на основе введённых в форму данных.* self.user = form.get\_user()  
 *# Выполняем аутентификацию пользователя.* login(self.request, self.user)  
 **return** super(LoginFormView, self).form\_valid(form)

1. И, наконец, привяжем созданный класс к соответствующему URL – **допишем строчку в riddles/urls.py**:

urlpatterns = [  
 url(**r'^$'**, views.index, name=**'index'**),  
 url(**r'^([0-9]+)/$'**, views.detail, name=**'detail'**),  
 url(**r'^([0-9]+)/answer/$'**, views.answer, name=**'answer'**),url(**r'^register/$'**, views.RegisterFormView.as\_view()),  
 **url(r'^login/$', views.LoginFormView.as\_view()),**]

1. **Запустим проект, перейдем на страницу авторизации, введем указанные при регистрации логин и пароль, посмотрим на появившееся вместо ссылок для регистрации и авторизации сообщение «Hello, <username>!» на главной странице.**
2. Для выхода из веб-приложения не нужно делать HTML-шаблон – можно просто переадресовать на главную страницу, поэтому сразу **добавим новый класс в riddles/views.py:**

*# Для Log out с перенаправлением на главную***from** django.http **import** HttpResponseRedirect  
**from** django.views.generic.base **import** View  
**from** django.contrib.auth **import** logout  
...

*# для выхода - миниатюрное представление без шаблона -  
# после выхода перенаправим на главную***class** LogoutView(View):  
 **def** get(self, request):  
 *# Выполняем выход для пользователя,  
 # запросившего данное представление.* logout(request)  
 *# После чего перенаправляем пользователя на  
 # главную страницу.* **return** HttpResponseRedirect(app\_url)

1. Привяжем созданный класс к соответствующему URL – **допишем строчку в riddles/urls.py**:

urlpatterns = [  
 url(**r'^$'**, views.index, name=**'index'**),  
 url(**r'^([0-9]+)/$'**, views.detail, name=**'detail'**),  
 url(**r'^([0-9]+)/answer/$'**, views.answer, name=**'answer'**),url(**r'^register/$'**, views.RegisterFormView.as\_view()),  
 url(**r'^login/$'**, views.LoginFormView.as\_view()),  
 **url(r'^logout/$', views.LogoutView.as\_view()),**]

1. **Запустим проект, нажмем на ссылку для выхода, посмотрим на исчезновение сообщения «Hello, <username>!» на главной странице.**
2. Закрепим создание страниц веб-приложения на примере страницы изменения пароля. Создадим HTML-шаблон **riddles/templates/reg/** **password\_change\_form.html:**

{% **extends 'base.html'** %}  
{% **load crispy\_forms\_tags** %}  
{% **block main** %}  
 <**h4 class="card-title"**>Change your password</**h4**>  
 <**form method="post"**>  
 {% **csrf\_token** %}  
 {{ **form**|**crispy** }}  
 <**button type="submit"   
 class="btn btn-success btn-block"**>  
 Change password  
 </**button**>  
 <**a href=".."   
 class="btn btn-danger btn-block"**>  
 Back to home  
 </**a**>  
 </**form**>  
{% **endblock** %}

1. **Добавим новый класс в riddles/views.py:**

*# Для смены пароля - форма***from** django.contrib.auth.forms **import** PasswordChangeForm  
...  
*# наше представление для смены пароля***class** PasswordChangeView(FormView):  
 *# будем строить на основе  
 # встроенной в django формы смены пароля* form\_class = PasswordChangeForm  
 template\_name = **'reg/password\_change\_form.html'** *# после смены пароля нужно снова входить* success\_url = app\_url + **'login/'  
 def** get\_form\_kwargs(self):  
 kwargs = super(PasswordChangeView, self).get\_form\_kwargs()  
 kwargs[**'user'**] = self.request.user  
 **if** self.request.method == **'POST'**:  
 kwargs[**'data'**] = self.request.POST  
 **return** kwargs  
 **def** form\_valid(self, form):  
 form.save()  
 **return** super(PasswordChangeView, self).form\_valid(form)

1. **Допишем строчку в riddles/urls.py**:

urlpatterns = [  
 url(**r'^$'**, views.index, name=**'index'**),  
 url(**r'^([0-9]+)/$'**, views.detail, name=**'detail'**),  
 url(**r'^([0-9]+)/answer/$'**, views.answer, name=**'answer'**),*\* url(**r'^register/$'**, views.RegisterFormView.as\_view()),  
 url(**r'^login/$'**, views.LoginFormView.as\_view()),  
 url(**r'^logout/$'**, views.LogoutView.as\_view()),  
 **url(r'^password-change/', views.PasswordChangeView.as\_view()**),  
]

1. **Запускаем, тестируем: регистрируемся, входим, меняем пароль, входим с новым паролем, выходим.**