ТЕКСТ БЕЗ ИЗМЕНЕНИЙ КАК ЕСТЬ

Сайт позволит зарегистрированным пользователям публиковать объявления о продаже чего-либо. Объявления будут разноситься по рубрикам, причём структура рубрик будет иметь два уровня иерархии: на первом уровне расположатся рубрики общего плана («Недвижимость», «Транспорт» и пр.), а на втором – более конкретные («Жильё», «Гаражи», «Дачи», «Легковой», «Грузовой»).

Для вывода списка объявлений мы применим пагинацию, т.к. объявлений может оказаться очень много, и страница, содержащая все объявления, будет слишком большой. Также мы предусмотрим возможность поиска объявлений по введённому посетителем слову.

Под любым объявлением (на странице сведений об объявлении) может быть составлено произвольное количество комментариев. Оставлять комментарии будет позволено любому пользователю, в том числе и гостю.

В составе объявления пользователь может поместить основную графическую иллюстрацию, которая будет выводится и в списке объявлений, и в составе сведений об объявлении, а также произвольное количество дополнительных иллюстраций, которые можно будет увидеть лишь на странице сведений об объявлении. И основная и дополнительные иллюстрации не являются обязательными к размещению.

Процедура регистрации нового пользователя на сайте будет разбита на два этапа. На первом этапе посетитель вводит свои данные на странице регистрации, после чего на указанный им адрес электронной почты приходит письмо с гиперссылкой, ведущей на страницу активации. На втором этапе посетитель переходит по гиперссылке, полученной в письме, попадает на страницу активации и становится полноценным пользователем.

Сайт доски объявлений включит в себя следующие страницы:

* Главная – показывающая десять последних опубликованных объявлений без разбиения их на рубрики;
* Страница списка объявлений – показывающая (с использованием пагинации) объявления из определённой рубрики. Также она будет содержать форму для поиска объявлений по введённому слову;
* Страница сведений о выбранном объявлении – выведет ещё все оставленные для него комментарии и форму для добавления нового комментария;
* Страницы регистрации и активации нового пользователя.
* Страницы входа и выхода.
* Страница профиля зарегистрированного пользователя – выведет список объявлений, оставленных текущим пользователем;
* Страницы добавления, правки, удаления объявлений;
* Страницы изменения пароля, правки и удаления пользовательского профиля;
* Страницы сведений о сайте, о правах его разработчика, пользовательского соглашения и пр.

Таков в общих чертах план сайта. Остальное будет уточняться далее……

Создание проекта

**django-admin startproject имяпроекта**

Переходим в папку проекта и подаём команду на создание приложения

**manage.py startapp ИмяПриложения**

Установка Bootstrap:

**pip install django-bootstrap4**

Запуск проекта

**manage.py runserver**

Создание миграций

**manage.py makemigration**

Выполнение миграции

**manage.py migrate**

После выполнения миграций, создадим Суперпользователя, подав команду:

**manage.py createsuperuser**

Введём выбранные нами имя, адрес электронной почты и пароль создаваемого пользователя.

Напоследок откроем модуль admin.py пакета приложения, в котором объявляются классы-редакторы и регистрируются модели в административном сайте. Зарегистрируем нашу модель пользователя, добавив в этот модуль код:

**from .models import AdvUser**

**admin.site.register(AdvUser)**

AdvUser – Имя модели.

**--------------------------------------------------------------------------------**

**main.views.py. index** - выводит Главную страницу (Контроллер)

**main/templates/main/index.html** - Шаблон главной страницы

**Вывод вспомогательных страниц**

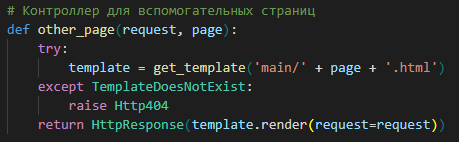
Первый способ, самый очевидный, заключается в том, что для каждой страницы пишется отдельный контроллер и отдельный маршрут. Этот способ весьма трудоёмок, поскольку придётся писать несколько контроллеров с практически одинаковым кодом, и подходит лишь для тех случаев, когда страницы должны выводить какие-либо данные, извлекаемые из базы или формируемые программно.

Второй способ – вывод всех страниц с применением одного контроллера и, соответственно, одного маршрута. Какой либо идентификатор страницы, предназначенной к выводу на экран, передаётся контроллеру с URL-параметром. Трудоёмкость работы в таком случае существенно снижается, поскольку нужно написать один контроллер.

**path(‘<str:page>/’, other\_page, name=’other’),**

Имя шаблона выводимой страницы передаём через URL-параметр **page**. А контроллер, выводящий вспомогательные страницы, назовём **other\_page** и реализуем в виде функции. Вообще, контроллеры-функции – идеальный инструмент для написания чего-либо нестандартного, специфического.

**main.views.other\_page -** Вывод вспомогательных страниц



Имя выводим страницы получаем из параметра **page**, добавляем к нему путь и расширение, получив тем самым полный путь к нужному шаблону, и пытаемся загрузить его вызовом функции **get\_template().** Если загрузка прошла успешно, то формируем на основе этого шаблона страницу.

Если же шаблон загрузить не удалось, то функция **get\_template()** возбудит исключение **TemplateDoesNotExist**. Мы перехватываем это исключение и возбуждаем другое исключение – **Http404**, которое приведёт к отправке страницы с сообщением об ошибке 404 (запрошенная страница не существует).

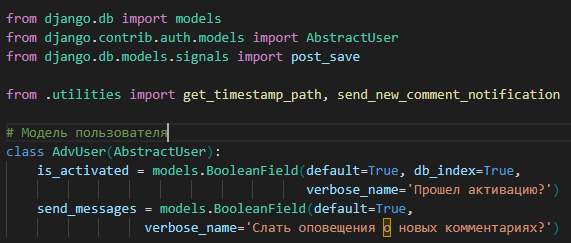
**main/templates/main/about.html** - Шаблон страницы со сведениями о сайте и правах его разработчиков. У этого шаблона нет контроллера. Ссылка на эту страницу в базовом шаблоне.

-------------------------------------------------------------------------

РАБОТА С ПОЛЬЗОВАТЕЛЯМИ РАЗГРАНИЧЕНИЕ ДОСТУПА

Теперь займёмся инструментами для работы с пользователями и разграничения доступа. Мы создадим страницы для входа и выхода, регистрации, активации, страницы пользовательского профиля, для смены данных о пользователе, смены его пароля и удаления профиля.

Стандартная модель пользователя User, предлагаемая стандартным же приложением **django.contrib.auth**, не подходит, поскольку нам нужно хранить дополнительные данные о пользователе. Поэтому создадим свою собственную модель, сделав её производной от абстрактной модели **AbstractUser**, объявленной в модуле django.contrib.auth.



После создания модели пользователя **main.models.AdvUser**

Создаём миграци (отладочный веб-сервер при этом должен быть остановлен)

**manage.py makemigration**

Выполняем миграции

**manage.py migrate**

После выполнения миграций, создадим Суперпользователя, подав команду:

**manage.py createsuperuser**

Введём выбранные нами имя, адрес электронной почты и пароль создаваемого пользователя.

Напоследок откроем модуль admin.py пакета приложения, в котором объявляются классы-редакторы и регистрируются модели в административном сайте. Зарегистрируем нашу модель пользователя, добавив в этот модуль код:

**from .models import AdvUser**

**admin.site.register(AdvUser)**

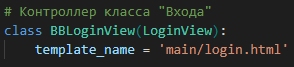
**AdvUser** – Имя модели.

**--------------------------------------------------------------------------**

Для реализации входа создадим подкласс контроллера-класса LoginView, назовём его **BBLoginView**

**main.views.BBLoginView**





В классе мы указали лишь путь к файлу шаблона, занеся его в атрибут **template\_name.** Остальные параметры сохранят значения по умолчанию.

Запишем новый маршрут, указывающий на контроллер BBLoginView, в списке уровня приложения

****

****

В маршруте мы указали шаблонный путь accounts/login/. По умолчанию именно по нему Django выполняет перенаправление при попытке гостя получить доступ к закрытой от него странице.

**НА ЗАМЕТКУ**

Мы могли бы записать маршрут и таким образом:

**from django.contrib.auth.views import LoginView**

**urlpatterns = [**

**path(‘accounts/login/’,**

***LoginView.as\_view(template\_name=’main/login.html’), name=’login’),***

]

Указав в маршруте непосредственно класс **LoginView** и записав все необходимые параметры контроллера в вызове метода **as\_view().** Но в таком случае код контроллеров окажется записан в двух модулях: **view.py** и **urls.py**, и в дальнейшем при сопровождении сайта, в поисках нужного фрагмента кода нам придётся просматривать оба этих модуля, что неудобно.

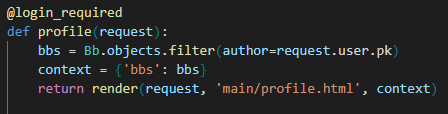
Поэтому давайте держать ***код контроллеров*** в модуле **views.py**, а ***код списка маршрутов*** – в модуле **urls.py**. Так нам будет проще.

**-----------------------------------------------------------------------------------------------**

**Пользовательский профиль.**

main.views.profile





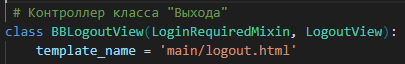
Поскольку страница пользовательского профиля должна быть доступна только зарегистрированным пользователям, выполнившим вход на сайт, мы пометили контроллер-функцию **profile()** декоратором **login\_required().**

**----------------------------------------------------------------------------------------------------**

**Страница выхода**







Страница выхода должна быть доступна только зарегистрированным пользователям, выполнившим вход. Поэтому мы добавили в число суперклассов контроллера-класса **BBLogoutView** примесь **LoginRequiredMixin**.

**Веб-страницы правки личных данных пользователя**

На этой странице пользователь сможет исправить свои имя (логин), адрес электронной почты, реальные имя, фамилию и признак, хочет ли он получать по электронной почте оповещения о появлении новых комментариев к его объявлениям. Адрес электронной почты будет обязательным к заполнению.