Алматинский Университет Менеджмента (AlmaU)

**Маштаков Кирилл Юрьевич**

**The system of social networks**

5B070300 Информационные Системы

Курсовой проект для предмета «Основы информационных систем»

Научные консультанты:

……………..

……………..

Республика Казахстан

Алматы, 2019

**Введение**

**Обоснование актуальности темы The system of social networks**

В современном мире почти каждый человек использует социальные сети. Кто-то занимается работой, кто-то общается, а кто-то просто прожигает свое время. Но в основном они используют их для коммуникации , обменом информации, просмотром видео, отдыха и т.д. Там люди ищут новых друзей, собираются в сообщества и обсуждают им интересные темы.

Актуальность заключается в том что , в наше время социальные сети играют огрумную роль в жизни современного человека. Люди перестали пользоваться смс сообщениями и стали меньше звонить друг другу. В основном этими вещами теперь занимаются в социальных сетях. Также основная аудитория социальных сетей молодежь, потому что они идут в ногу с новыми технологиями.

**Практическая значимость**

Эта система социальной сети создается для обмена сообщениями, публикации постов с текстом и картинками, для оценок публикации.

**Оценка современного состояния решаемой проблемы**

Это система создана, чтобы решить проблему деанонимизации и свободы слова в интернете. Потому что государства начали нарушать наши права на анонимность в социальных сетях и свободу слова.

**Цели**

Цель данной курсовой работы, создание готовой социальной сети для обмена информацией , публикацией новостей и полной свободе слова и анонимности внутри социальной сети.

**Задачи**

- Создать функцию регистрации и аутентификации в социальной сети.

- Создание функции выставления оценок публикациям.

- Создание списка друзей.

- Создать функцию управления друзьями(добавление, отклонить, принять)

- Создать страницу пользователя .

- Создание удобного и понятного интерфейса.

- Загрузить готовый проект на хостинг Google Cloud.

**Теоретическая и методологическая основа и практическая база**

Методологической основой послужили обучающие видео на youtube и просмотр и анализ кода на GitHub, Stack Overflow, и т.д.. В качестве теоретической базы были использована документация Python и фреймворка Django. Практической базой послужил web-фреймворк Django, djangorestframework, библиотека работы с изображениями Pillow. Сервером и хостингом послужил Google Cloud Platform..

**Объект исследования** – создание удобной и качественной социальной сети.

**Метод исследования** – анализ, применения навыков полученных на кафедре ШИМ и их развитие .

**Новизна**

В этой системе будет магазин кастомизации. Валютой магазина послужит лайки(которые пользователь будет получать, когда ему будут ставить лайк).В этом магазине будет такая кастомизация как: рамка в профиле для аватара, обои на задний фон, различные эффекты и т.д.

**Описание структуры**

**Введение…………………………………………………………………………..2**

Обоснование актуальности темы………………………………………….2

Практическая значимость………………………………………………….2

Оценка современного состояния решаемой проблемы………………….2

Задачи………………………………………..……………………………...2

Теоретическая и методологическая основа и практическая база……….2

Объект исследования ……………………………………………………...3

Метод исследования ……………………………………………………….3

Новизна………………………………………..…………………………….3

Описание структуры……………………………………………………….4

**UML диаграммы…………………………………………………………………5**

Data-logical model…………………………………………………………..5

Physical model………………………………………………………………6

Conceptual model……………………………………………………………7

Graph model…………………………………………………………………8

**Описание страниц сайта………………………………………………………..9**

Header……………………………………………………………………….9

Сайдбар…………………………………………………………………….10

Моя страница……………………………………………………………...11

Новости…………………………………………………………………….13

Друзья………………………………………..…………………………….18

Пользователи……………………………………………………………...19

Footer……………………………………………………………………….21

**Заключение………………………………………..…………………………….22**

**Список использованных источников……………………………………….23**

Приложение А-Г………………………………………..……………………..

**UML диаграммы:**



Рисунок 1.1 - Data-logical model

Эта модель описывает все используемые модели и дает список атрибутов.

Рисунок 1.2 - Physical model

Это модель похожа на Data-logical model, но отличается тем, что здесь идет более конкретное описание. Здесь указывается тип данных у атрибутов.



Рисунок 1.3 - Conceptual model

Концептуальная модель дает представление о системе. Здесь можно увидеть все взаимосвязи и что от кого зависит.

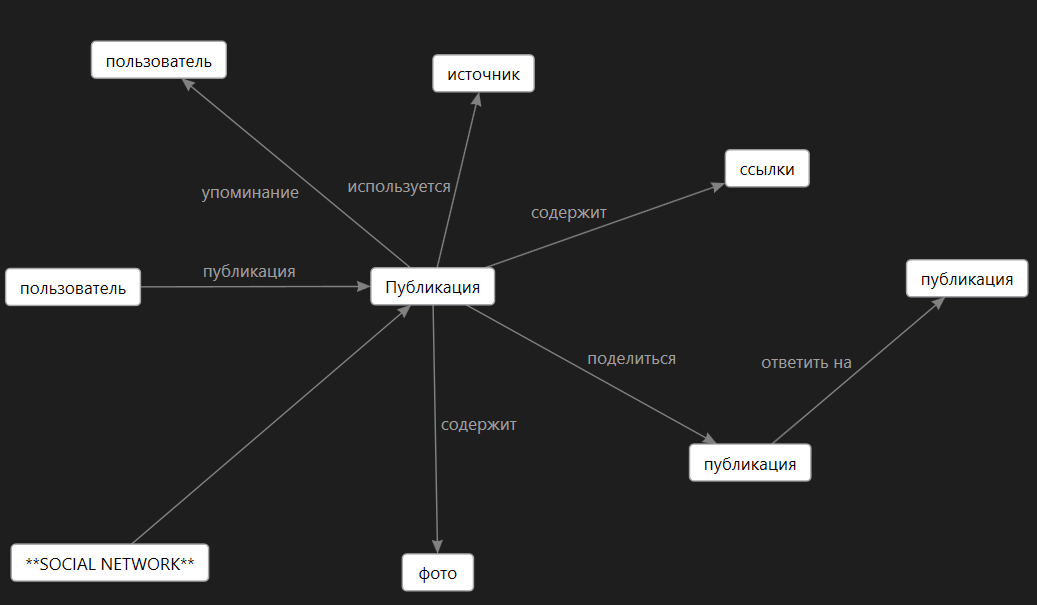


Рисунок 1.4 - Graph model

Графическая модель позволяет увидеть схему публикации. И понять как она работает. Тут описывается что содержится, как происходит отправка и репост публикаций.

**Описание страниц сайта:**

**Header:**



Рисунок 2.1 – Header

Header сайта состоит из логотип и трех кнопок «войти», «регистрация», и «выйти». После того как пользователь заходит в систему кнопки «войти» «регистрация» пропадают , а вместо них появляется кнопка «выйти» и надпись что пользователь зашел в систему под своим логином. Это можно посмотреть на рисунке 2.2.



Рисунок 2.2 – Header

**Сайдбар:**

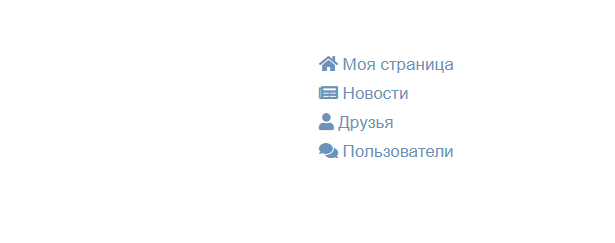


Рисунок 2.3 – Сайдбар

Слева на сайте расположен сайдбар . В нем находятся четыре ссылки: Моя страница, Новости, Друзья и Пользователи.

**Моя страница:**

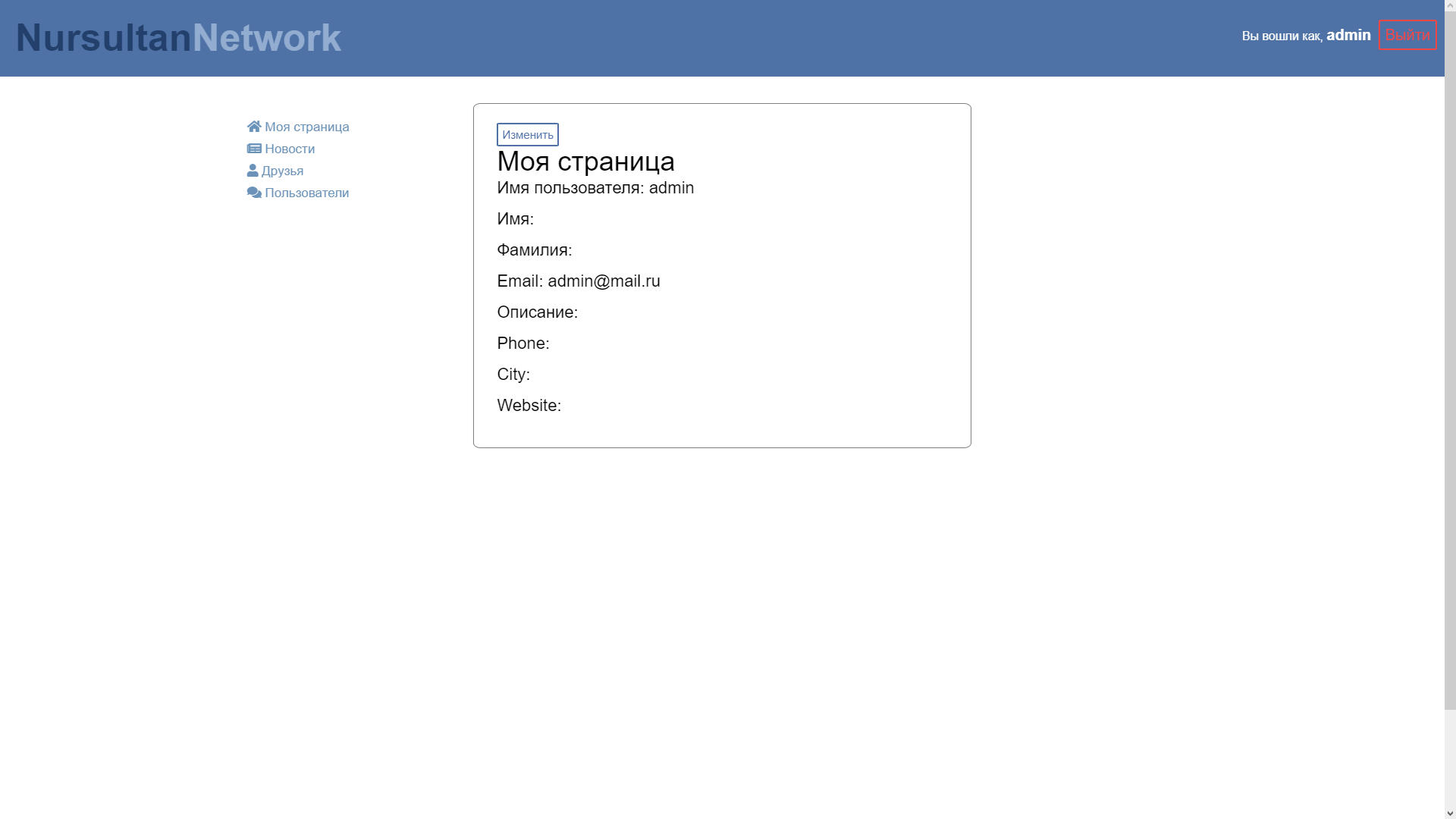


Рисунок 2.4 – Моя страница

«Моя страница» состоит из блока информации о пользователе и содержит внутри кнопку «Изменить». Это кнопка позволяет открыть меню изменения профиля. Это можно посмотреть на Рисунке 2.4.

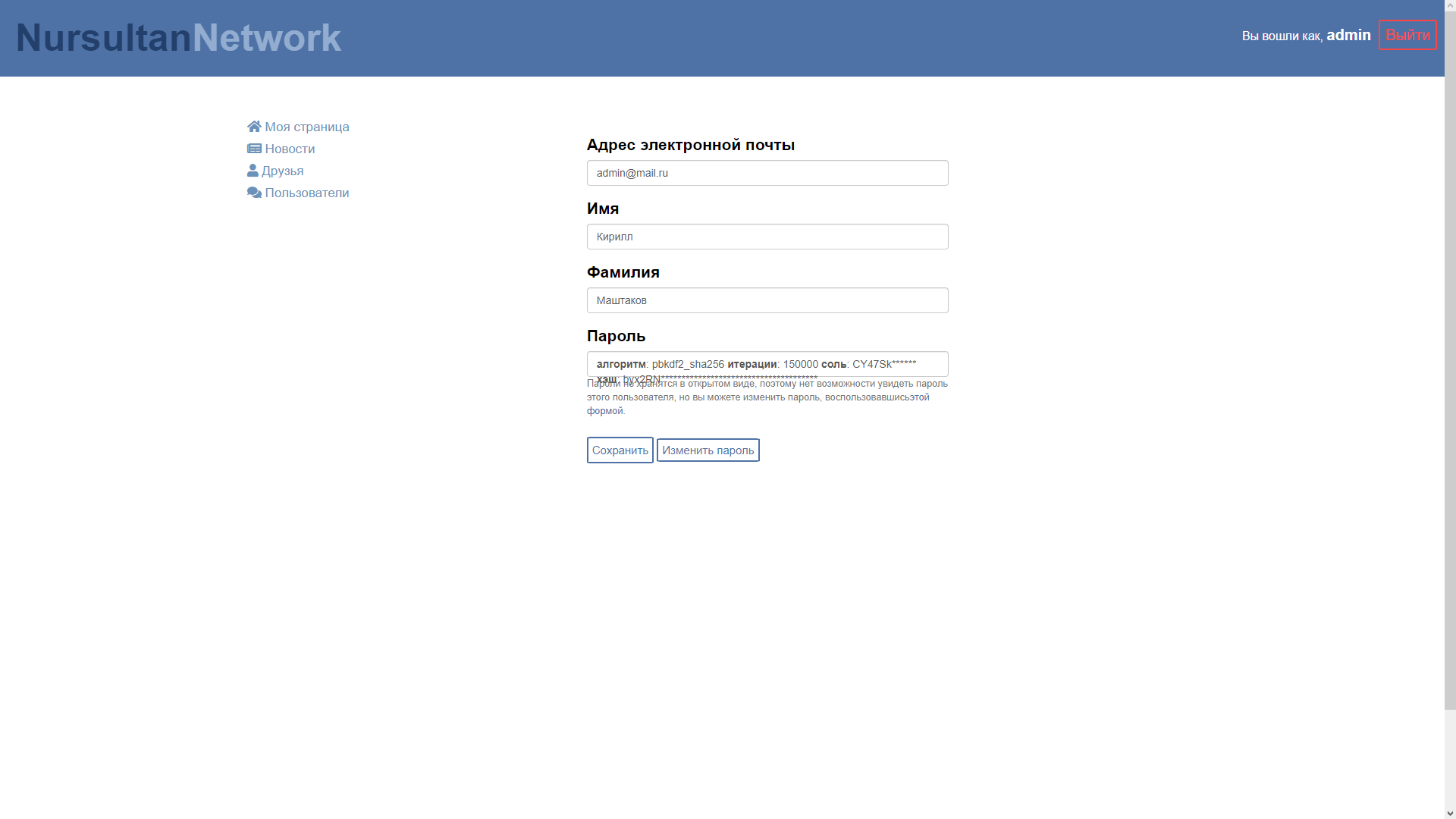


Рисунок 2.5 – Моя страница, изменение профиля

На странице изменения профиля есть поля для изменения: Имени, Фамилии, и адреса электронной почты. Также эта форма содержит еще одну кнопку «Изменить пароль». После нажатия кнопки она переносит на страницу изменения пароля. Для изменения нужно будет ввести свой старый пароль, и два раза ввести новый пароль.

**Новости:**

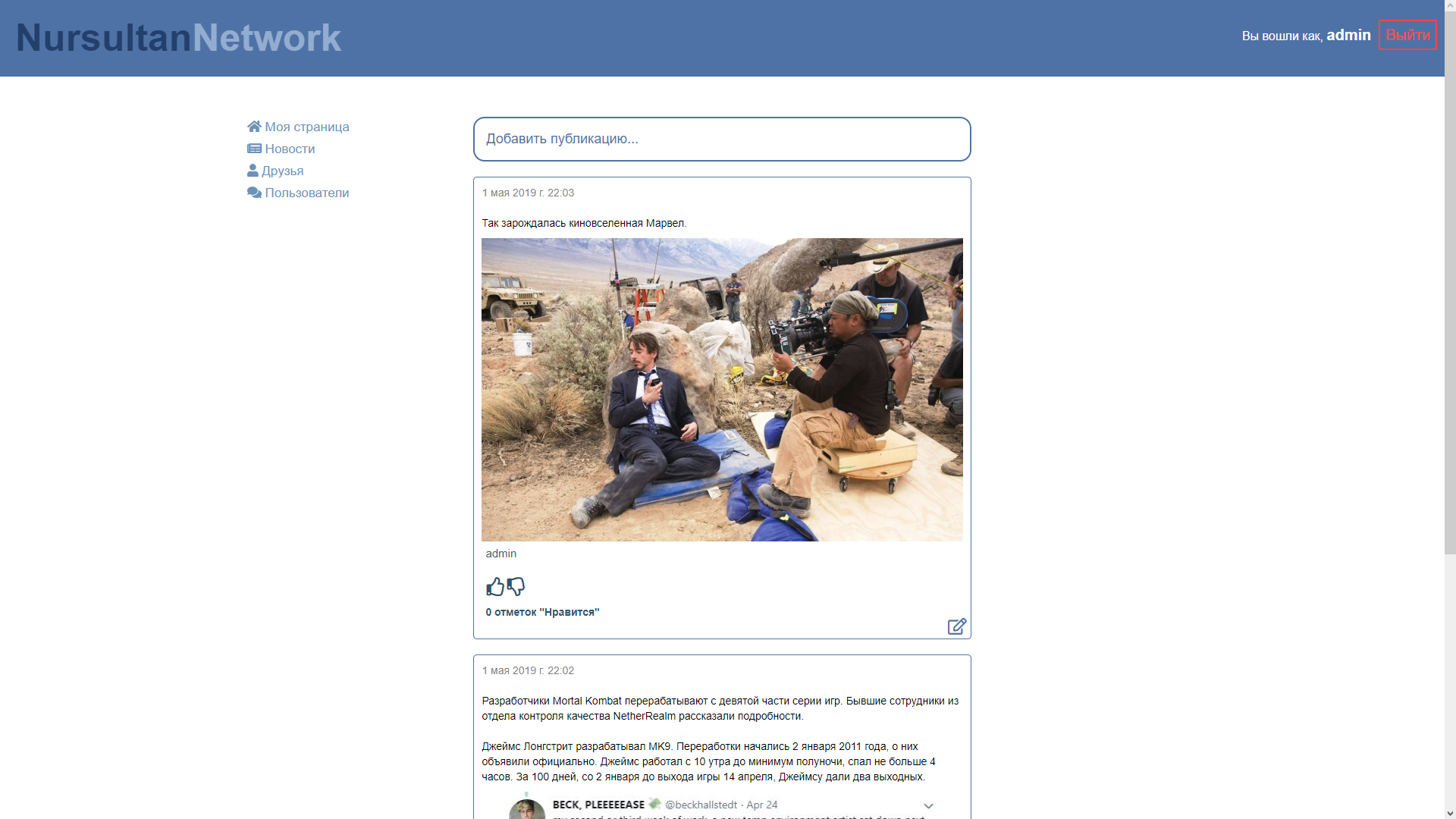


Рисунок 2.5 – Новости

Страница «Новости» состоит из кнопки «Добавить публикацию», которая позволяет создавать новые публикации. На рисунке 2.6 показано меню создания новой публикации. Чтобы создавать публикации надо быть авторизованным пользователем.

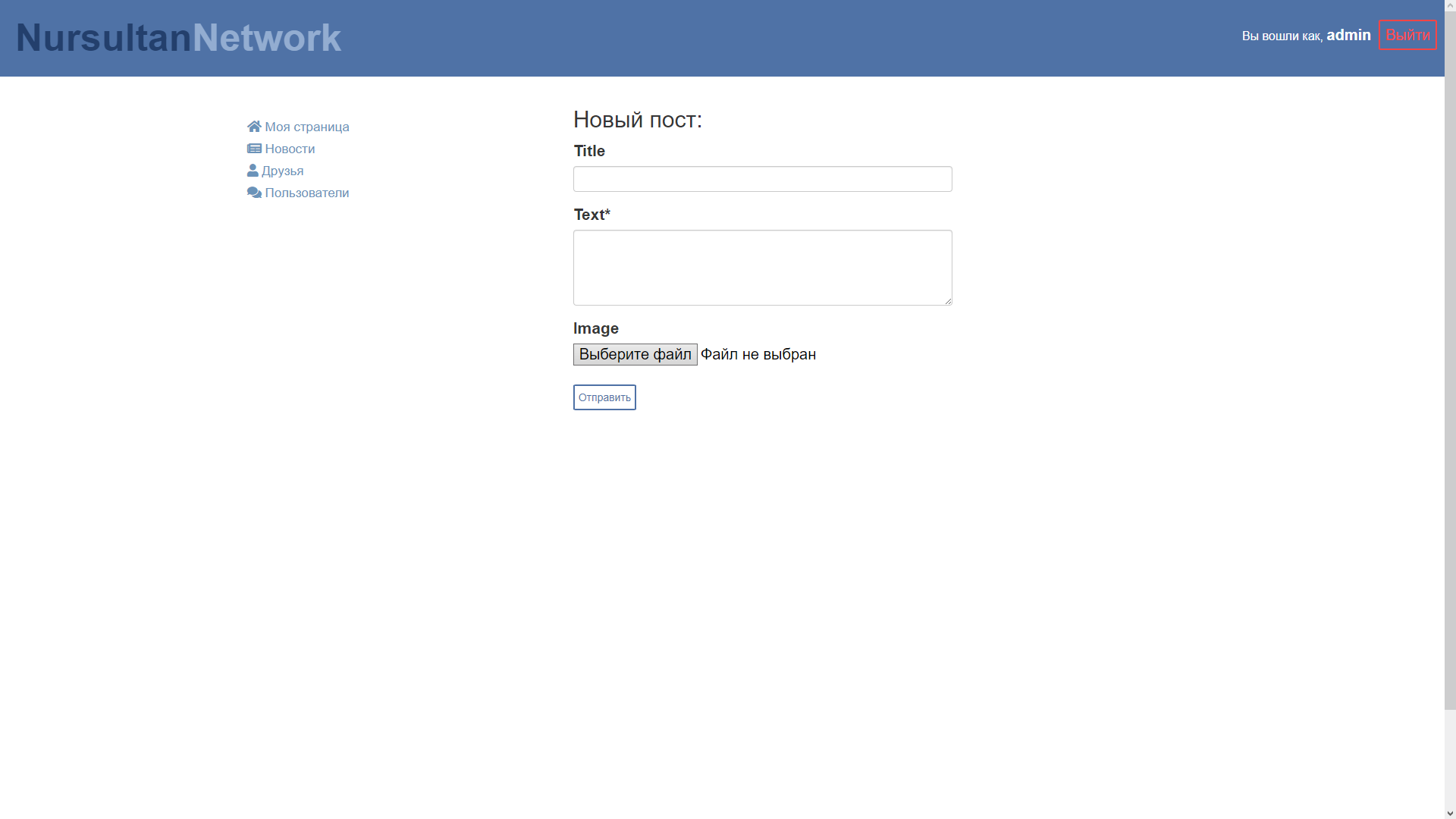


Рисунок 2.6 – Новости, создание новой публикации

На этой странице есть два поля ввода текста и возможность прикрепить картинку к публикации. Обязательным полем для создания новой публикации служит поле «Text».

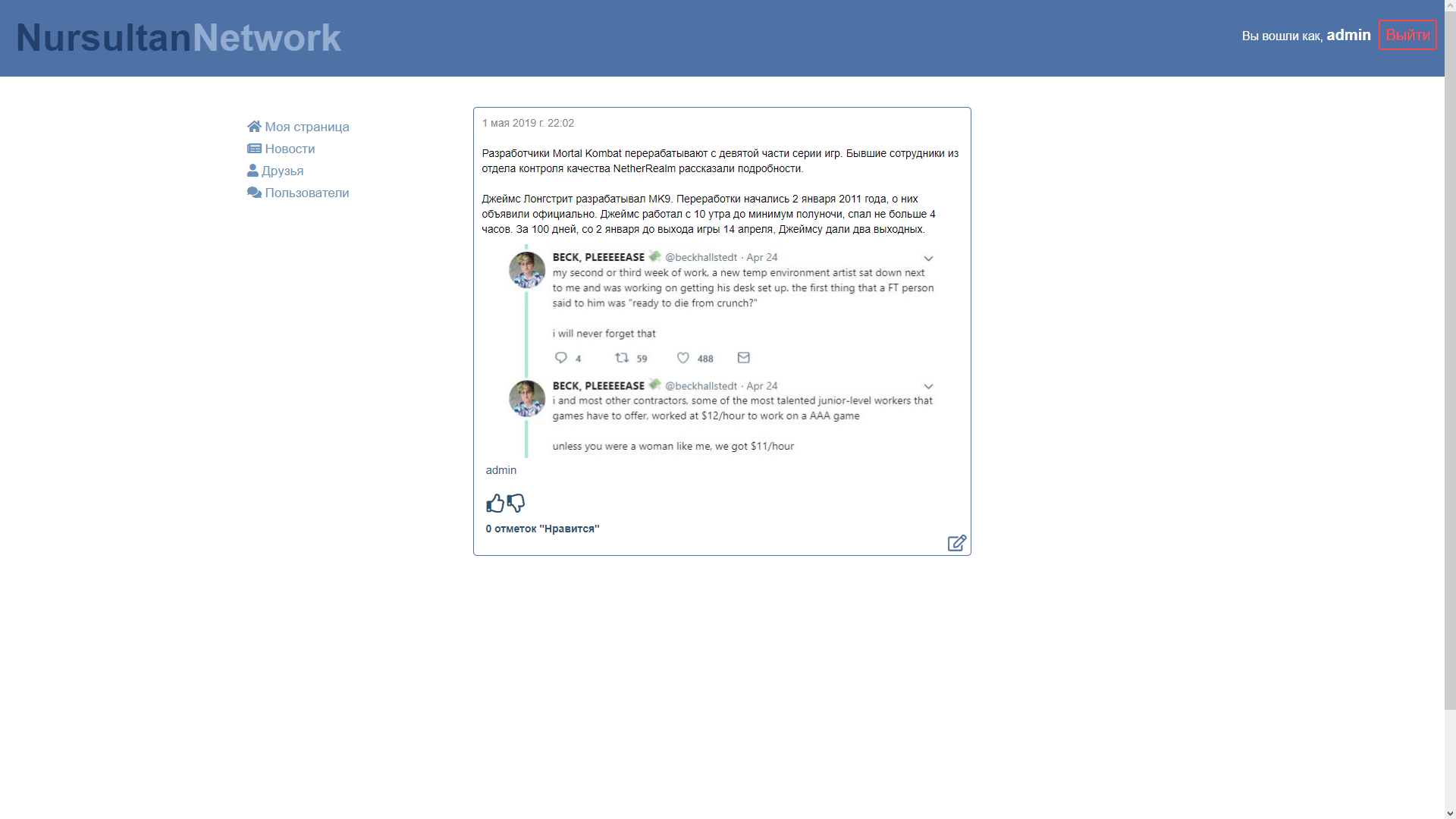


Рисунок 2.7 – Новости, публикация

Данный блок состоит из:

- даты создания

- заглавия(не обязательно)

- текста

- картинки(не обязательно)

- имени автора

- кнопки редактировать эту публикацию

- двух кнопок оценки данной публикации. Палец вверх и палец вниз.

- количества отметок «Нравится»

На Рисунке 2.8 можно увидеть как выглядит меню изменения публикации

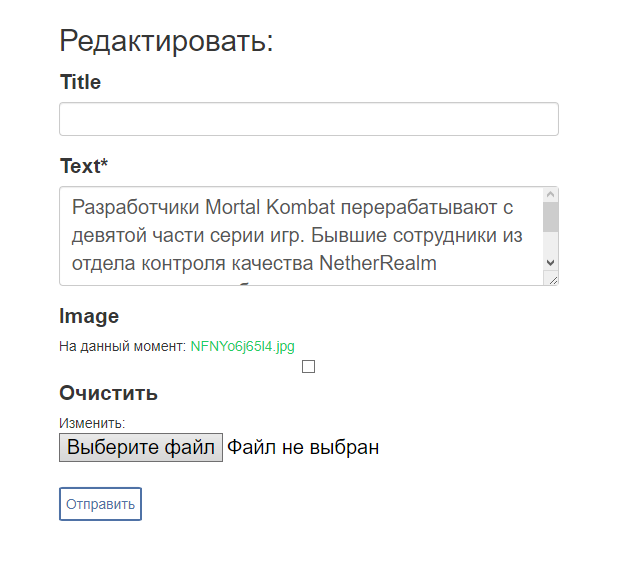


Рисунок 2.8 – Новости, редактирование публикации

Эта страница выглядит как страница создания новой публикации как на Рисунке 2.6. Здесь можно редактировать текст, добавить или убрать заглавие и загрузить новую картинку.



Рисунок 2.9 – Новости, лайк-дизлайк и изменить публикацию

На этом рисунке изображены составляющие каждой публикации. Кнопку «изменить» может использовать только автор данной публикации. Чтобы появилась возможность ставить оценки публикациям нужно авторизоваться в систему.

**Друзья:**

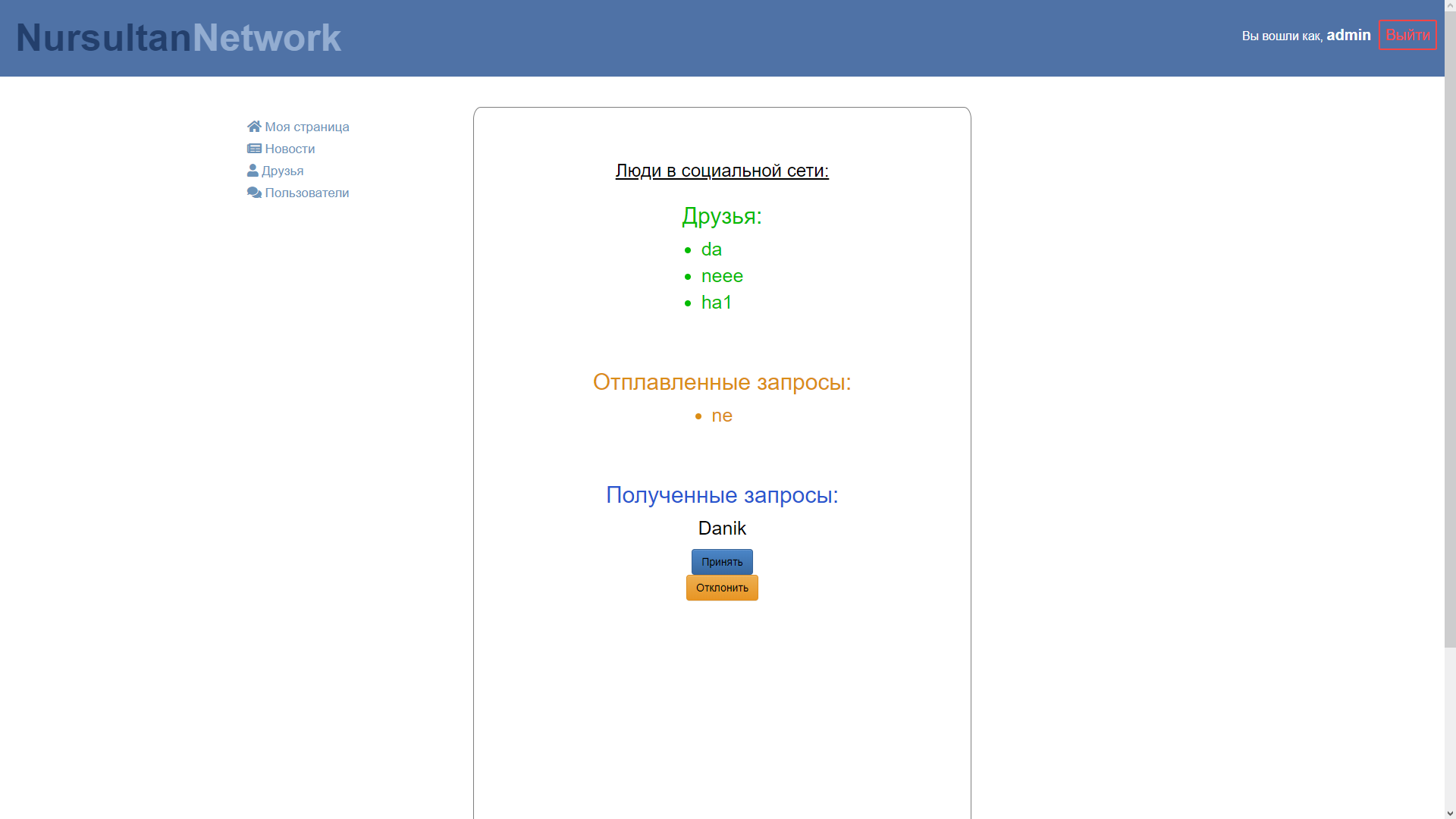


Рисунок 2.10 – Друзья

Страница «Друзья» состоит из трех блоков.

Первый блок это друзья.

Второй блок это отправленные запросы от пользователя.

Третий блок это полученные запросы от других пользователей.

В третьем блоке после того как вас добавляют в друзья появляется, имя пользователя и две кнопки с вариантами принять в друзья или отклонить. После нажатия кнопки принять данный пользователь попадает в первый блок «друзья». При нажатии кнопки «отклонить» данный запрос удаляется и может быть послан снова.

**Пользователи:**

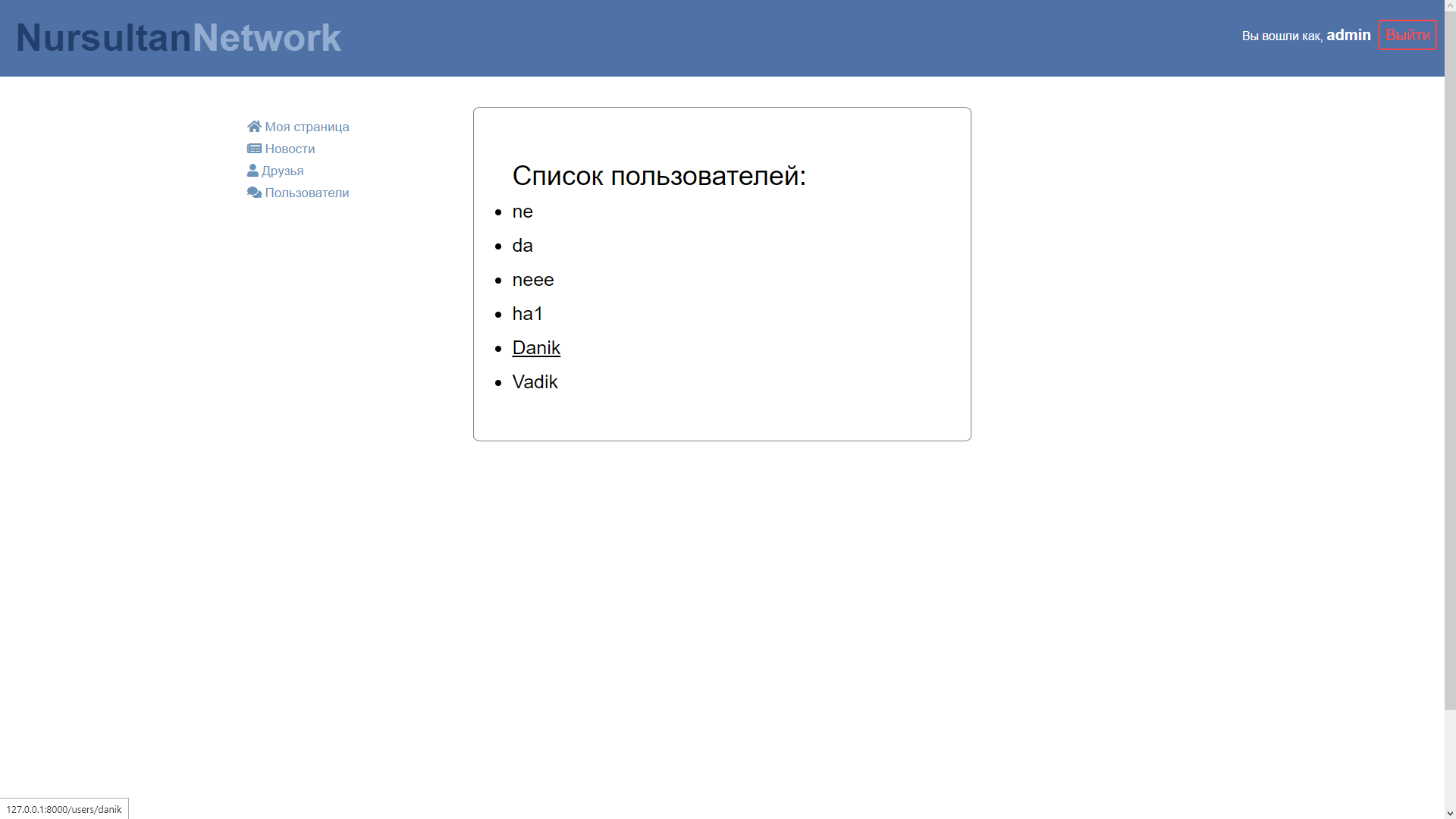


Рисунок 2.11 – Пользователи,

На странице «Пользователи» выведен список из всех пользователей зарегистрированных в сети. На имена пользователей можно нажать, чтобы добавить их в друзья. На рисунке 2.12 показано как происходить добавление в друзья.

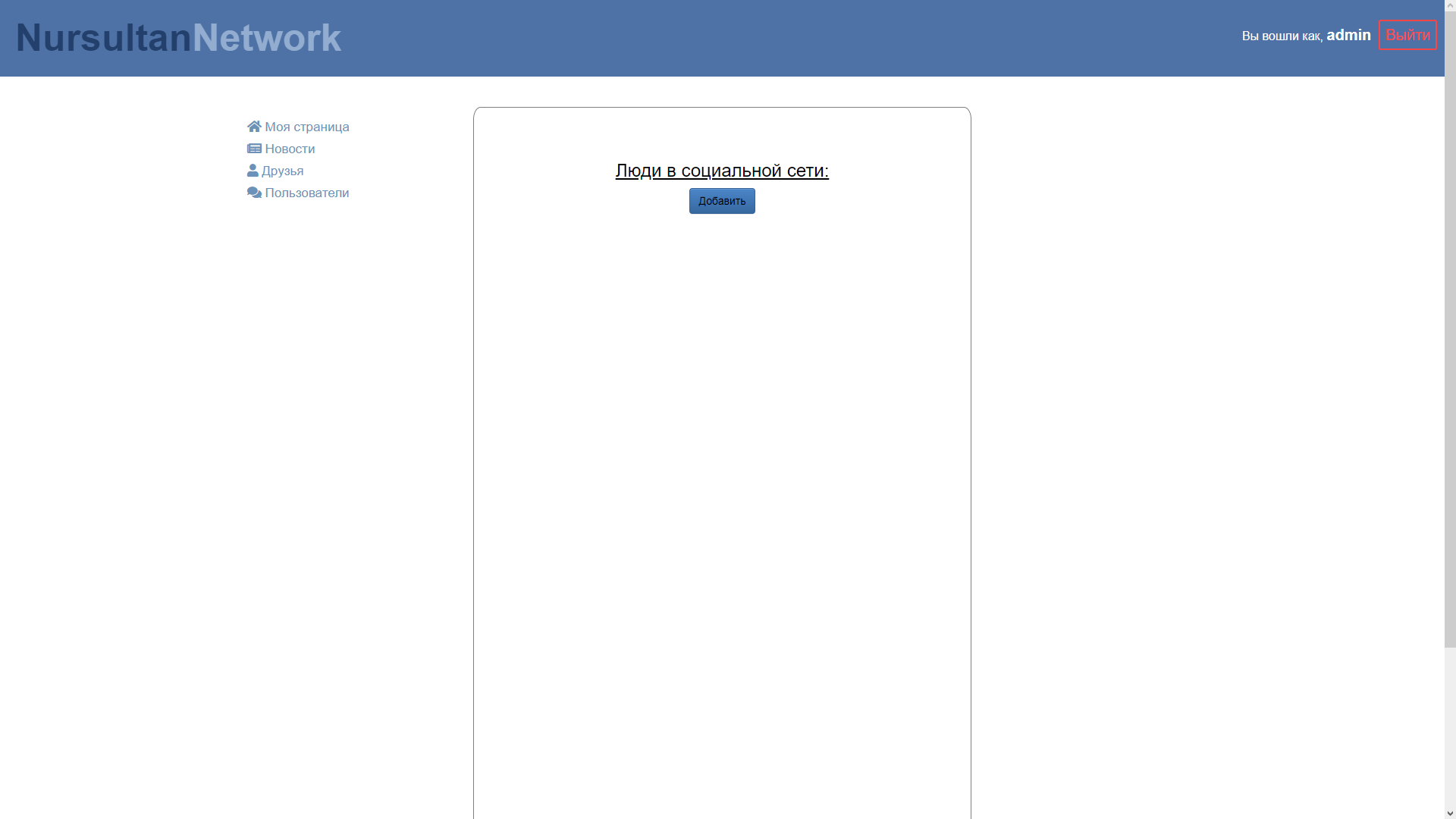


Рисунок 2.12 – Пользователи, добавление в друзья

На этом рисунке видно что после нажатия на имя пользователя, его переносит на страницу с добавлением. При нажатии кнопки «добавить», пользователю отправляется запрос на добавление в друзья на странице «Друзья». Также после нажатия кнопки «добавить» она пропадает и на ее месте появляется кнопка «Отменить запрос», которая в свою очередь удаляет запрос на добавление в друзья.

**Footer:**



В самом низу странице можно увидеть футер. Он состоит из дополнительной информации и контактов автора сайта.

**Заключение**

**Краткие выводы по результатам курсового проекта**

Все поставленные задачи в курсовом проекты были выполнены. Однако возникли проблемы с добавлением функции Списка друзей и функции Оценок публикаций. Данная система социальной сети работает эффективно и имеет дружелюбный пользовательский интерфейс. В дальнейшем эта система будет дорабатываться и оптимизироваться. В планах создать функцию обмена сообщениями, функцию восстановления пароля через почту.

**Список использованных источников:**

1. [Электронный ресурс] <https://tutorial.djangogirls.org/ru/> 10.02.2019
2. [Электронный ресурс] <https://books.agiliq.com/projects/django-orm-cookbook/en/latest/slugfield.html> 28.04.2019
3. [Электронный ресурс] <https://apirobot.me/posts/how-to-implement-liking-in-django> 24.04.19
4. [Электронный ресурс] <https://www.youtube.com/watch?v=Rr1-UTFCuH4&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=19&t=102s> 04.03.19
5. [Электронный ресурс] <https://www.youtube.com/watch?v=q4jPR-M0TAQ&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=18&t=2391s> 04.03.19
6. [Электронный ресурс] <https://www.youtube.com/watch?v=3aVqWaLjqS4&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=17&t=0s> 04.03.19
7. [Электронный ресурс] <https://www.youtube.com/watch?v=Z4IwTFgXtrA&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=16&t=2s> 21.02.19
8. [Электронный ресурс] <https://www.youtube.com/watch?v=UI6lqHOVHic&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=15&t=0s> 25.03.19
9. [Электронный ресурс] <https://www.youtube.com/watch?v=IXJ46DitsIg> 17.02.19
10. [Электронный ресурс] <https://github.com/maxg203/Django-Tutorials> 04.03.19
11. [Электронный ресурс] <https://github.com/codingforentrepreneurs/try-django-19> 04.03.19
12. [Электронный ресурс] <https://github.com/justdjango/friends> 10.04.19
13. [Электронный ресурс] <https://www.youtube.com/watch?v=_DqmVMlJzqA&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=13&t=0s> 28.02.19
14. [Электронный ресурс] <https://www.youtube.com/watch?v=QxGKTvx-Vvg&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=12&t=0s> 08.04.19
15. [Электронный ресурс] <https://www.youtube.com/watch?v=QS9z4g0JAOY&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=11&t=911s> 10.04.19
16. [Электронный ресурс] <https://www.youtube.com/watch?v=6WnNoPq8UOY&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=9&t=2186s> 16.04.19
17. [Электронный ресурс] <https://www.youtube.com/watch?v=q2dF3z7VuME&list=PLJMh_xuJIifmMsc3KsRsne-oosjNRelsm&index=2&t=1123s> 28.04.19
18. [Электронный ресурс] <https://www.django-rest-framework.org> 24.03.19
19. [Электронный ресурс] <https://www.youtube.com/playlist?list=PL4cUxeGkcC9ib4HsrXEYpQnTOTZE1x0uc> 08.04.19
20. [Электронный ресурс] <https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-16-04> 15.04.19

**Приложение-А**

**Исходный код для создания публикаций, для лайков и просмотр профиля**

Файл models.py

**from** django.conf **import** settings  
**from** django.db **import** models  
**from** django.utils **import** timezone  
**from** django.contrib.auth.models **import** User  
  
**from** django.contrib.contenttypes.fields **import** GenericForeignKey  
**from** django.contrib.contenttypes.models **import** ContentType  
**from** django.contrib.contenttypes.fields **import** GenericRelation  
  
**class** Like(models.Model):  
 user = models.ForeignKey(settings.AUTH\_USER\_MODEL,  
 related\_name=**'likes'**,  
 on\_delete=models.CASCADE)  
 content\_type = models.ForeignKey(ContentType, on\_delete=models.CASCADE)  
 object\_id = models.PositiveIntegerField()  
 content\_object = GenericForeignKey(**'content\_type'**, **'object\_id'**)  
  
**class** Post(models.Model):  
 author = models.ForeignKey(User, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=200)  
 image = models.ImageField(null=**True**, blank=**True**, width\_field=**"width\_field"**, height\_field=**"height\_field"**)  
 height\_field=models.IntegerField(default=330)  
 width\_field = models.IntegerField(default=330)  
 text = models.TextField()  
 created\_date = models.DateTimeField(default=timezone.now)  
 published\_date = models.DateTimeField(blank=**True**, null=**True**)  
  
 likes = GenericRelation(Like)  
  
 **def** publish(self):  
 self.published\_date = timezone.now()  
 self.save()  
  
 **def** \_\_str\_\_(self):  
 **return** self.title  
  
 @property  
 **def** total\_likes(self):  
 **return** self.likes.count()

Файл views.py

**from** .models **import** Post, Like  
**from** django.utils **import** timezone  
**from** django.shortcuts **import** render, get\_object\_or\_404  
**from** .forms **import** PostForm, UserRegisterForm, EditProfileForm  
**from** django.contrib.auth.forms **import** UserChangeForm, PasswordChangeForm  
**from** django.contrib.auth **import** update\_session\_auth\_hash  
**from** django.shortcuts **import** redirect  
**from** django.contrib.auth.models **import** User  
**from** django.contrib **import** messages  
  
**from** django.contrib.auth **import** get\_user\_model  
**from** django.contrib.contenttypes.models **import** ContentType  
**from** rest\_framework **import** viewsets  
**from** rest\_framework.permissions **import** IsAuthenticatedOrReadOnly  
**from** .serializers **import** PostSerializer  
**from** .mixins **import** LikedMixin  
  
  
*# Создание постов, изменение, просмотр***def** com\_list(request):  
 posts = Post.objects.filter(published\_date\_\_lte=timezone.now()).order\_by(**'-published\_date'**)  
 **return** render(request, **'mysocial/com\_list.html'**, {**'posts'**: posts})  
  
**def** com\_detail(request, pk):  
 post = get\_object\_or\_404(Post, pk=pk)  
 **return** render(request, **'mysocial/com\_detail.html'**, {**'post'**: post})  
  
**def** com\_new(request):  
 **if** request.method == **"POST"**:  
 form = PostForm(request.POST, request.FILES **or None**)  
 **if** form.is\_valid():  
 post = form.save(commit=**False**)  
 post.author = request.user  
 post.published\_date = timezone.now()  
 post.save()  
 *# messages.success(request, f'Вы добавили новую публикацию')* **return** redirect(**'com\_detail'**, pk=post.pk)  
 **else**:  
 form = PostForm()  
 **return** render(request, **'mysocial/com\_edit.html'**, {**'form'**: form})  
  
  
**def** com\_edit(request, pk):  
 post = get\_object\_or\_404(Post, pk=pk)  
 **if** request.method == **"POST"**:  
 form = PostForm(request.POST **or None**, request.FILES **or None**, instance=post)  
 **if** form.is\_valid():  
 post = form.save(commit=**False**)  
 post.author = request.user  
 post.published\_date = timezone.now()  
 post.save()  
 *# messages.success(request, f'Вы изменили публикацию')* **return** redirect(**'com\_detail'**, pk=post.pk)  
 **else**:  
 form = PostForm(instance=post)  
 **return** render(request, **'mysocial/com\_edit.html'**, {**'form'**: form})  
  
*#Форма регистрации***def** register(request):  
 **if** request.method == **'POST'**:  
 form = UserRegisterForm(request.POST)  
 **if** form.is\_valid():  
 form.save()  
 username = form.cleaned\_data.get(**'username'**)  
 *# messages.success(request, f'Вы зарегестрировались как {username} и можете войти в свой аккаунт!')* **return** redirect(**'login'**)  
 **else**:  
 form = UserRegisterForm()  
 **return** render(request, **'mysocial/register.html'**, {**'form'**: form})  
  
  
*# Profile view***def** view\_profile(request, pk=**None**):  
 **if** pk:  
 user = User.objects.get(pk=pk)  
 **else**:  
 user = request.user  
 args = {**'user'**: user}  
 **return** render(request, **'mysocial/profile.html'**, args)  
  
**def** edit\_profile(request):  
 **if** request.method == **'POST'**:  
 form = EditProfileForm(request.POST, instance=request.user)  
  
 **if** form.is\_valid():  
 form.save()  
 **return** redirect(view\_profile)  
 **else**:  
 form = EditProfileForm(instance=request.user)  
 args = {**'form'**: form}  
 **return** render(request, **'mysocial/edit\_profile.html'**, args)  
  
**def** change\_password(request):  
 **if** request.method == **'POST'**:  
 form = PasswordChangeForm(data=request.POST, user=request.user)  
  
 **if** form.is\_valid():  
 form.save()  
 update\_session\_auth\_hash(request, form.user)  
 **return** redirect(**'view\_profile'**)  
 **else**:  
 **return** redirect(**'change\_password'**)  
 **else**:  
 form = PasswordChangeForm(user=request.user)  
  
 args = {**'form'**: form}  
 **return** render(request, **'mysocial/change\_password.html'**, args)  
  
*#Лайки*User = get\_user\_model()  
**def** add\_like(obj, user):  
 *"""Лайкает `obj`.  
 """* obj\_type = ContentType.objects.get\_for\_model(obj)  
 like, is\_created = Like.objects.get\_or\_create(  
 content\_type=obj\_type, object\_id=obj.id, user=user)  
 **return** like  
**def** remove\_like(obj, user):  
 *"""Удаляет лайк с `obj`.  
 """* obj\_type = ContentType.objects.get\_for\_model(obj)  
 Like.objects.filter(  
 content\_type=obj\_type, object\_id=obj.id, user=user  
 ).delete()  
**def** is\_fan(obj, user) -> bool:  
 *"""Проверяет, лайкнул ли `user` `obj`.  
 """* **if not** user.is\_authenticated:  
 **return False** obj\_type = ContentType.objects.get\_for\_model(obj)  
 likes = Like.objects.filter(  
 content\_type=obj\_type, object\_id=obj.id, user=user)  
 **return** likes.exists()  
**def** get\_fans(obj):  
 *"""Получает всех пользователей, которые лайкнули `obj`.  
 """* obj\_type = ContentType.objects.get\_for\_model(obj)  
 **return** User.objects.filter(  
 likes\_\_content\_type=obj\_type, likes\_\_object\_id=obj.id)  
  
**class** PostViewSet(LikedMixin,viewsets.ModelViewSet):  
 queryset = Post.objects.all()  
 serializer\_class = PostSerializer  
 permission\_classes = (IsAuthenticatedOrReadOnly,)

Файл urls.py

**from** django.urls **import** path  
**from** . **import** views  
**from** django.contrib.auth **import** views **as** auth\_views  
  
urlpatterns = [  
 path(**''**, views.com\_list, name=**'com\_list'**),  
 path(**'feed/'**, views.com\_list, name=**'com\_list'**),  
 path(**'feed/post/<int:pk>/'**, views.com\_detail, name=**'com\_detail'**),  
 path(**'feed/post/new/'**, views.com\_new, name=**'com\_new'**),  
 path(**'feed/post/<int:pk>/edit/'**, views.com\_edit, name=**'com\_edit'**),  
 *# path('friend\_list/', views.list\_friends, name='friend\_list'),* path(**'profile/'**, views.view\_profile, name=**'view\_profile'**),  
 path(**'profile/<int:pk>/'**, views.view\_profile, name=**'view\_profile\_with\_pk'**),  
 path(**'profile/edit/'**, views.edit\_profile, name=**'edit\_profile'**),  
 path(**'change-password/'**, views.change\_password, name=**'change\_password'**),  
  
  
]  
  
**from** rest\_framework.routers **import** DefaultRouter  
**from** .views **import** PostViewSet  
  
router = DefaultRouter()  
router.register(**r'posts'**, PostViewSet)  
  
urlpatterns+=router.urls

serializers.py

**from** rest\_framework **import** serializers  
**from** .models **import** Post  
  
**from** . **import** views **as** likes\_services  
**from** django.contrib.auth **import** get\_user\_model  
  
**class** PostSerializer(serializers.ModelSerializer):  
  
 is\_fan = serializers.SerializerMethodField()  
  
 **class** Meta:  
 model = Post  
 fields = (  
 **'author'**,  
 **'id'**,  
 **'title'**,  
 **'text'**,  
 **'created\_date'**,  
 **'published\_date'**,  
 **'total\_likes'**,  
 **'is\_fan'**,  
 )  
  
 **def** get\_is\_fan(self, obj) -> bool:  
 *"""Проверяет, лайкнул ли `request.user` твит (`obj`).  
 """* user = self.context.get(**'request'**).user  
 **return** likes\_services.is\_fan(obj, user)  
  
User = get\_user\_model()  
**class** FanSerializer(serializers.ModelSerializer):  
 full\_name = serializers.SerializerMethodField()  
 **class** Meta:  
 model = User  
 fields = (  
 **'username'**,  
 **'full\_name'**,  
 )  
 **def** get\_full\_name(self, obj):  
 **return** obj.get\_full\_name()

mixins.py

**from** rest\_framework.decorators **import** detail\_route  
**from** rest\_framework.response **import** Response  
**from** django.shortcuts **import** redirect  
**from** . **import** views  
**from** .serializers **import** FanSerializer  
**class** LikedMixin:  
 @detail\_route(methods=[**'POST'**])  
 **def** like(self, request, pk=**None**):  
 *"""Лайкает `obj`.  
 """* obj = self.get\_object()  
 views.add\_like(obj, request.user)  
 **return** redirect(**'com\_list'**)  
 @detail\_route(methods=[**'POST'**])  
 **def** unlike(self, request, pk=**None**):  
 *"""Удаляет лайк с `obj`.  
 """* obj = self.get\_object()  
 views.remove\_like(obj, request.user)  
 **return** redirect(**'com\_list'**)  
 @detail\_route(methods=[**'GET'**])  
 **def** fans(self, request, pk=**None**):  
 *"""Получает всех пользователей, которые лайкнули `obj`.  
 """* obj = self.get\_object()  
 fans = views.get\_fans(obj)  
 serializer = FanSerializer(fans, many=**True**)  
 **return** Response(serializer.data)

forms.py

**from** django **import** forms  
**from** .models **import** Post  
**from** django.contrib.auth.models **import** User  
**from** django.contrib.auth.forms **import** UserCreationForm, UserChangeForm  
  
  
**class** PostForm(forms.ModelForm):  
 title = forms.CharField(max\_length=200,required=**False**)  
  
 **class** Meta:  
 model = Post  
 fields = (**'title'**, **'text'**,**'image'**)  
  
**class** UserRegisterForm(UserCreationForm):  
 email = forms.EmailField()  
 first\_name = forms.CharField(max\_length=30, required=**True**)  
 last\_name = forms.CharField(max\_length=30, required=**True**)  
  
 **class** Meta:  
 model = User  
 fields = [**'first\_name'**, **'last\_name'**, **'username'**, **'email'**, **'password1'**, **'password2'**]  
  
  
*#Change profile***class** EditProfileForm(UserChangeForm):  
  
 **class** Meta:  
 model = User  
 fields = [**'email'**,**'first\_name'**,**'last\_name'**,**'password'**]

**Приложение-Б**

**Исходный код для списка друзей**

models.py

**from** django.conf **import** settings  
**from** django.db **import** models  
**from** django.db.models.signals **import** post\_save  
**from** django.contrib.auth.models **import** User  
**from** django.utils.text **import** slugify  
  
**class** Profile(models.Model):  
 user = models.OneToOneField(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)  
 slug = models.SlugField(unique=**True**)  
 friends = models.ManyToManyField(**"Profile"**, blank=**True**)  
  
 **def** \_\_str\_\_(self):  
 **return** str(self.user.username)  
  
 **def** save(self, \*args, \*\*kwargs):  
 self.slug = slugify(self.user)  
 super(Profile, self).save(\*args, \*\*kwargs)  
  
 **def** get\_absolute\_url(self):  
 **return "/users/{}"**.format(self.slug)  
  
  
**def** post\_save\_user\_model\_receiver(sender, instance, created, \*args, \*\*kwargs):  
 **if** created:  
 **try**:  
 Profile.objects.create(user=instance)  
 **except**:  
 **pass**post\_save.connect(post\_save\_user\_model\_receiver, sender=settings.AUTH\_USER\_MODEL)  
  
  
**class** FriendRequest(models.Model):  
 to\_user = models.ForeignKey(settings.AUTH\_USER\_MODEL, related\_name=**'to\_user'**, on\_delete=models.CASCADE)  
 from\_user = models.ForeignKey(settings.AUTH\_USER\_MODEL, related\_name=**'from\_user'**, on\_delete=models.CASCADE)  
 timestamp = models.DateTimeField(auto\_now\_add=**True**) *# set when created* **def** \_\_str\_\_(self):  
 **return "From {}, to {}"**.format(self.from\_user.username, self.to\_user.username)

views.py

**from** django.conf **import** settings  
**from** django.contrib.auth **import** get\_user\_model  
**from** django.http **import** HttpResponseRedirect  
**from** django.shortcuts **import** render, get\_object\_or\_404  
  
**from** .models **import** Profile, FriendRequest  
  
User = get\_user\_model()  
  
**def** users\_list(request):  
 users = Profile.objects.exclude(user=request.user)  
 context = {  
 **'users'**: users  
 }  
 **return** render(request, **"friends/user\_list.html"**, context)  
  
**def** send\_friend\_request(request, id):  
 **if** request.user.is\_authenticated:  
 user = get\_object\_or\_404(User, id=id)  
 frequest, created = FriendRequest.objects.get\_or\_create(  
 from\_user=request.user,  
 to\_user=user)  
 **return** HttpResponseRedirect(**'/users'**)  
  
**def** cancel\_friend\_request(request, id):  
 **if** request.user.is\_authenticated:  
 user = get\_object\_or\_404(User, id=id)  
 frequest = FriendRequest.objects.filter(  
 from\_user=request.user,  
 to\_user=user).first()  
 frequest.delete()  
 **return** HttpResponseRedirect(**'/users'**)  
  
**def** accept\_friend\_request(request, id):  
 from\_user = get\_object\_or\_404(User, id=id)  
 frequest = FriendRequest.objects.filter(from\_user=from\_user, to\_user=request.user).first()  
 user1 = frequest.to\_user  
 user2 = from\_user  
 user1.profile.friends.add(user2.profile)  
 user2.profile.friends.add(user1.profile)  
 frequest.delete()  
 **return** HttpResponseRedirect(**'/users/{}'**.format(request.user.profile.slug))  
  
**def** delete\_friend\_request(request, id):  
 from\_user = get\_object\_or\_404(User, id=id)  
 frequest = FriendRequest.objects.filter(from\_user=from\_user, to\_user=request.user).first()  
 frequest.delete()  
 **return** HttpResponseRedirect(**'/users/{}'**.format(request.user.profile.slug))  
  
**def** profile\_view(request, slug):  
 p = Profile.objects.filter(slug=slug).first()  
 u = p.user  
 sent\_friend\_requests = FriendRequest.objects.filter(from\_user=p.user)  
 rec\_friend\_requests = FriendRequest.objects.filter(to\_user=p.user)  
  
 friends = p.friends.all()  
  
 *# is this user our friend* button\_status = **'none'  
 if** p **not in** request.user.profile.friends.all():  
 button\_status = **'not\_friend'** *# if we have sent him a friend request* **if** len(FriendRequest.objects.filter(  
 from\_user=request.user).filter(to\_user=p.user)) == 1:  
 button\_status = **'friend\_request\_sent'** context = {  
 **'u'**: u,  
 **'button\_status'**: button\_status,  
 **'friends\_list'**: friends,  
 **'sent\_friend\_requests'**: sent\_friend\_requests,  
 **'rec\_friend\_requests'**: rec\_friend\_requests  
 }  
  
 **return** render(request, **"friends/friend\_list.html"**, context)

urls.py

**from** django.conf.urls **import** url  
**from** django.urls **import** path  
**from** . **import** views  
  
urlpatterns = [  
 path(**''**, views.users\_list, name=**'list'**),  
 path(**'<slug>/'**, views.profile\_view),  
 path(**'friend-request/send/<id>/'**, views.send\_friend\_request),  
 path(**'friend-request/cancel/<id>/'**, views.cancel\_friend\_request),  
 path(**'friend-request/accept/<id>/'**, views.accept\_friend\_request),  
 path(**'friend-request/delete/<id>/'**, views.delete\_friend\_request),  
]

signals.py

**from** django.db.models.signals **import** post\_save  
**from** django.contrib.auth.models **import** User  
**from** django.dispatch **import** receiver  
**from** social.friends.models **import** Profile  
  
  
@receiver(post\_save, sender=User)  
**def** create\_profile(sender, instance, created, \*\*kwargs):  
 **if** created:  
 Profile.objects.create(user=instance)  
  
  
@receiver(post\_save, sender=User)  
**def** save\_profile(sender, instance, \*\*kwargs):  
 instance.profile.save()

**Приложение-В**

**Исходный код основной части**

Главный urls.py

**from** django.conf **import** settings  
**from** django.conf **import** settings  
**from** django.conf.urls **import** url  
**from** django.conf.urls.static **import** static  
**from** django.contrib **import** admin  
**from** django.urls **import** path, include  
**from** django.contrib.auth **import** views **as** auth\_views  
**from** mysocial **import** views **as** user\_views  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
 path(**''**, include(**'mysocial.urls'**)),  
 path(**'register/'**, user\_views.register, name=**'register'**),  
 path(**'login/'**, auth\_views.LoginView.as\_view(template\_name=**'mysocial/login.html'**), name=**'login'**),  
 path(**'logout/'**, auth\_views.LogoutView.as\_view(template\_name=**'mysocial/logout.html'**), name=**'logout'**),  
 path(**'users/'**, include(**'friends.urls'**)),  
 path(**'accounts/'**, include(**'allauth.urls'**)),  
]  
  
  
apipatterns = [  
 path(**''**, include(**'mysocial.urls'**)),  
]  
  
urlpatterns+= [  
 path(**'api/v1/'**, include(apipatterns)),  
]  
  
  
**if** settings.DEBUG:  
 urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)  
 urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**Приложение-Г**

**Необходимые библиотеки и модули для работы проекта**

certifi==2019.3.9  
chardet==3.0.4  
colorama==0.4.1  
defusedxml==0.5.0  
Django==2.2  
django-allauth==0.39.1  
django-crispy-forms==1.7.2  
djangorestframework==3.9.2  
httpie==1.0.2  
idna==2.8  
oauthlib==3.0.1  
Pillow==6.0.0  
Pygments==2.3.1  
python3-openid==3.1.0  
pytz==2018.9  
PyYAML==5.1  
requests==2.21.0  
requests-oauthlib==1.2.0  
requirements==0.1  
sqlparse==0.3.0  
urllib3==1.24.1