

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский политехнический университет»

Кафедра «Инфокогнитивные технологии»
Образовательная программа «Веб-технологии»

Отчет по курсовому проекту
по дисциплине «Основы серверной веб-
разработки»

Тема: «Социальная сеть»

Выполнил:

Студент группы 221-321

Ямалтдинов Р.И.

подпись, дата

Принял:

Старший преподаватель

Даньшина М.В.

подпись, дата

Москва 2023

Содержание

| | |
|---|----|
| Введение..... | 3 |
| §1. Проект базы данных..... | 4 |
| 1.1. Используемая технология..... | 4 |
| 1.2. Конкретная схема..... | 4 |
| 1.3. Определённые модели..... | 5 |
| 1.4. Созданные записи..... | 7 |
| §2. Django tutorial..... | 8 |
| 2.1. Инициализация проекта..... | 8 |
| 2.2. Работа с базой данных..... | 9 |
| 2.3. Использование template для представления | 9 |
| 2.4. Работа с формами..... | 10 |
| 2.5. Тесты..... | 11 |
| 2.6. Статика..... | 12 |
| 2.7. Улучшение admin панели..... | 13 |
| 2.8. Установка сторонних пакетов..... | 14 |
| §3. Глубокая настройка админ-панели Django..... | 15 |
| §4. Реализованный API..... | 16 |
| 4.1. Посты..... | 16 |
| 4.2. Комментарии к постам..... | 16 |
| 4.3. Лайки к постам..... | 16 |
| 4.4. Пользователи..... | 16 |
| 4.5. Сообщения..... | 17 |
| Заключение..... | 18 |
| Список использованных источников..... | 19 |
| Приложение 1 Записи таблиц базы данных..... | 20 |

Введение

Курсовой проект выполнен для объединения знаний, полученных в ходе обучения различным дисциплинам в предыдущие семестры. Предполагаемый результат разработки по данной теме является выполненный при помощи Django фреймворка (v. 4.2.2) гибкий API, предоставляющий удобную работу с базой данных, для дальнейшего его использования при разработке отдельной клиентской части. Также проделанная работа поспособствует получению опыта в разработке приложений на Django, проектировании баз данных и в целом работе с ними. Для контроля версий Django проекта будет использоваться git.

§1. Проект базы данных

1.1. Используемая технология

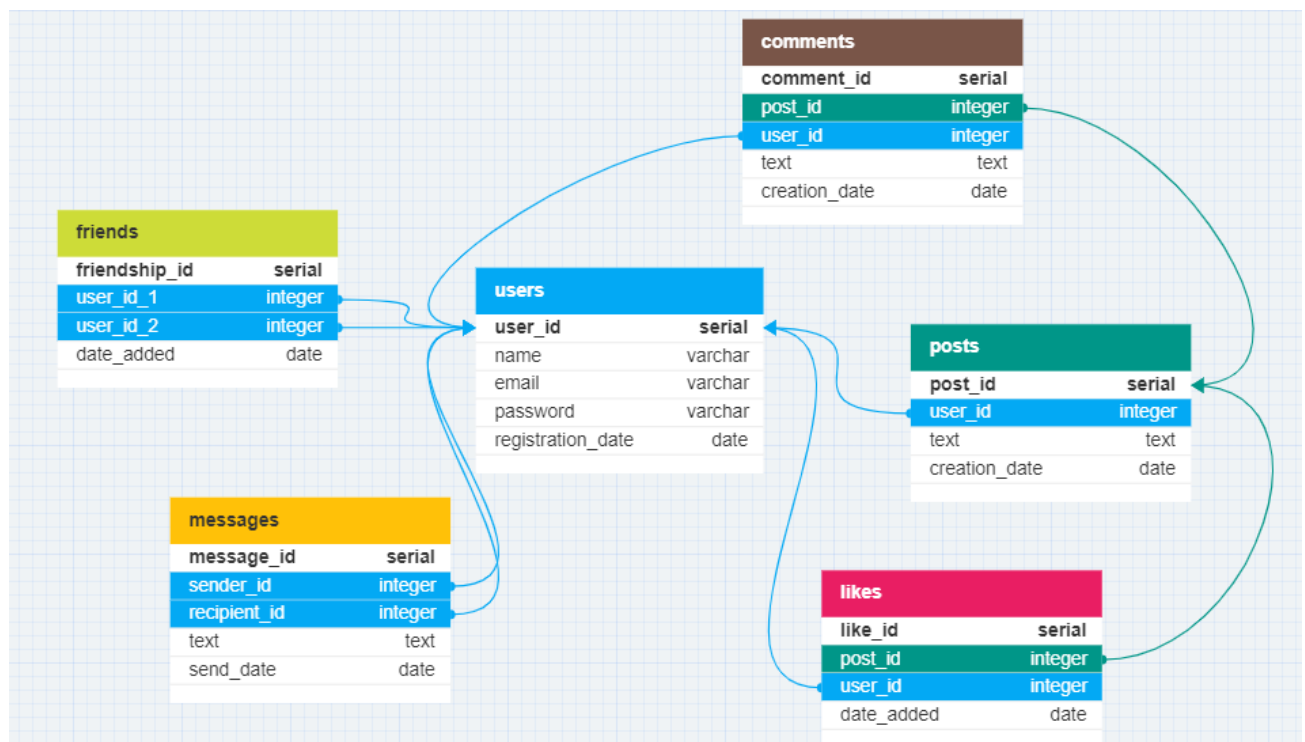
В качестве используемой в Django проекте СУБД является PostgreSQL. PostgreSQL является мощной и надежной СУБД с богатым функционалом, открытым исходным кодом, высокой производительностью и расширяемостью, что делает его привлекательным решением для широкого круга проектов и приложений.

1.2. Конкретная схема

Для начала потребовалось изучить информационную область, связанную с темой проекта. Исходя из изучения были определены требуемые таблицы для базы данных:

- 1) Users — пользователи, которые будут пользоваться социальной сетью;
- 2) Friends — модель для реализации функционала заключения дружбы между пользователями;
- 3) Messages — сообщения между пользователями будут описаны здесь;
- 4) Posts — посты, которые пользователи могут публиковать;
- 5) Comments — комментарии к постам от пользователей;
- 6) Likes — лайки пользователей на постах.

Чтобы иметь удобное представление схемы базы данных для разработки был использован веб-сервис:



1.3. Определённые модели

В Django проекте модели описаны следующим образом:

```
# api/models.py

class Post(models.Model):
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    text = models.TextField()
    creation_date = models.DateField()

    def __str__(self):
        return str(self.user) + ": " + str(self.text)

class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    text = models.TextField()
    creation_date = models.DateField()

    def __str__(self):
        return str(self.user) + ' at ' + str(self.post) + ': ' + str(self.text)

class Like(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    date_added = models.DateField()

    def __str__(self):
        return str(self.user) + ' at ' + str(self.post) + ' '

# message/models.py

class Message(models.Model):
    sender = models.ForeignKey(
        CustomUser, on_delete=models.CASCADE, related_name="sent_messages"
    )
    recipient = models.ForeignKey(
        CustomUser, on_delete=models.CASCADE, related_name="received_messages"
    )
    text = models.TextField()
    send_date = models.DateTimeField()

    @admin.display(
        boolean=True,
        ordering="send_date",
```

```

        description="Sent recently?",
    )
    def was_sent_recently(self):
        now = timezone.now()
        return now - datetime.timedelta(days=1) <= self.send_date <= now

    def __str__(self):
        return (
            str(self.send_date)
            + " "
            + str(self.sender)
            + " -> "
            + str(self.recipient)
            + ": "
            + str(self.text)
        )

# users/models.py

class CustomUser(AbstractUser):
    USERNAME_FIELD = "email"
    REQUIRED_FIELDS = ["firstname"]

    firstname = models.CharField(max_length=255, default="")
    lastname = models.CharField(max_length=255, default="")
    email = models.CharField(unique=True)
    password = models.CharField(max_length=255)
    registration_date = models.DateField(default="2000-01-01")

    is_staff = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    is_superuser = models.BooleanField(default=False)

    objects = CustomUserManager()

    def get_username(self):
        return self.email

    def get_id(self):
        return self.id

    def __str__(self):
        return str(self.firstname) + " " + str(self.lastname)

class Friend(models.Model):
    user_1 = models.ForeignKey(
        CustomUser, on_delete=models.CASCADE, related_name="user_1_friends"
    )

```

```

user_2 = models.ForeignKey(
    CustomUser, on_delete=models.CASCADE, related_name="user_2_friends"
)
date_added = models.DateField()

def __str__(self):
    return str(self.user_1) + " " + str(self.user_2)

```

1.4. Созданные записи

В каждой таблице имеется не менее 10 записей. В пример приведена таблица Posts:

Select post to change

🔍 Search

◀ All dates April 2023 May 2023 June 2023

Action: Go 0 of 10 selected

| <input type="checkbox"/> | ID | USER | CREATION DATE | TEXT |
|--------------------------|----|---------------------|----------------|--|
| <input type="checkbox"/> | 10 | Wolfgang Mozart | April 28, 2023 | Послушайте мою новую композицию на сайте https://classic-music.com/mozart/rap-hit |
| <input type="checkbox"/> | 9 | Daria Whiskas | June 1, 2023 | С днём защиты детей! |
| <input type="checkbox"/> | 8 | Joe Biden | May 11, 2023 | Я упал с лестницы |
| <input type="checkbox"/> | 7 | Joe Biden | April 1, 2023 | Я упал с лестницы |
| <input type="checkbox"/> | 6 | Clone Anon | May 25, 2023 | Вы слышали про субмарину? |
| <input type="checkbox"/> | 5 | Bob Bobson | June 20, 2023 | Очень длинная и интересная история про что-то бла-бла-бла |
| <input type="checkbox"/> | 4 | Rustem YamaItldinov | June 20, 2023 | Сегодня я впервые вышел на улицу |
| <input type="checkbox"/> | 3 | Rosalina Rorakiki | June 20, 2023 | Мой милый кот! |
| <input type="checkbox"/> | 2 | Bob Bobson | June 20, 2023 | I have a cat, she's cute |
| <input type="checkbox"/> | 1 | Rustem YamaItldinov | June 20, 2023 | New post |

10 posts

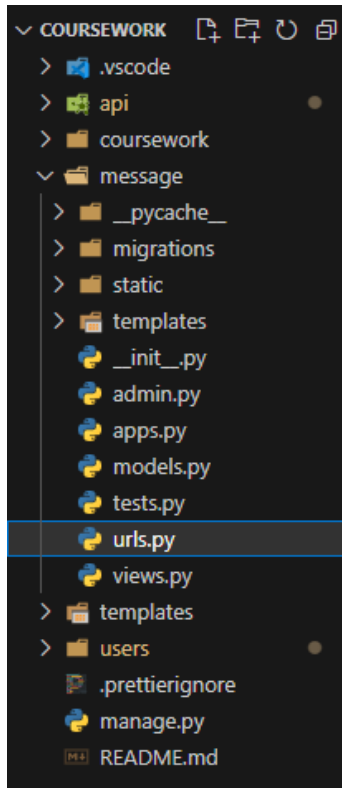
Записи остальных таблиц приведены в приложении 1

§2. Django tutorial

Использованные в Django tutorialе методы, приёмы и т.п. были для удобства интерпретированы в message app.

2.1. Инициализация проекта

Был инициализирован Django проект с приложениями api, message, users:



Создано index представление с приветственным сообщением:

```
message > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  app_name = "message"
5  urlpatterns = [
6      path("", views.index, name="index"),
7      path("<int:user_pk>/", views.chat, name="chat"),
8      path("<int:user_pk>/send", views.send, name="send"),
9  ]
10

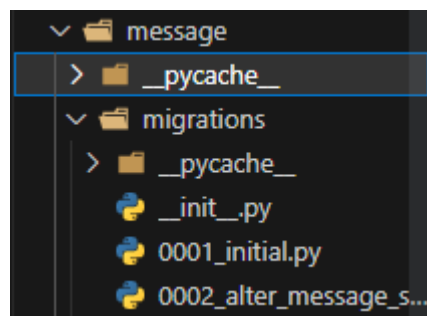
12
13  # Create your views here.
14  def index(request):
15      return HttpResponse(
16          "Hello, you are at messages index. I"
17      )
18
```


2.2. Работа с базой данных

Подключена база данных postgresql со стороннего хостинга:

```
coursework > settings.py > ...
90 # Database
91 # https://docs.djangoproject.com/en/4.2/ref/settings/#databases
92
93 DATABASES = {
94     "default": {
95         "ENGINE": "django.db.backends.postgresql",
96         "NAME": "railway",
97         "USER": "postgres",
98         "PASSWORD": "-",
99         "HOST": "-",
100         "PORT": "-",
101     }
102 }
```

Проведены миграции в приложениях для переноса моделей из django проекта в базу данных:



2.3. Использование template для представления

Реализовано отображение чата с выводом всех сообщений между клиентом и конкретизированным при помощи user_pk пользователем с использованием template:

```
# message/views.py
def chat(request, user_pk):
    if not request.user.is_authenticated:
        login_url = reverse(settings.LOGIN_URL)
        return redirect(login_url)
    user_1 = request.user

    try:
        user_2 = CustomUser.objects.get(pk=user_pk)
    except CustomUser.DoesNotExist:
        return HttpResponse("Error! User not found.")
    messages = Message.objects.filter(
        Q(sender=user_1, recipient=user_2) | Q(sender=user_2, recipient=user_1)
    ).order_by("send_date")

    context = {
        "messages": messages,
```

```

        "user_1": user_1,
        "user_2": user_2,
    }

    return render(request, "message/chat.html", context)

```

message > templates > message > **5** chat.html > ...

```

6  <a href="{% url 'message:index' %}" class="chat__index">Message index</a>
7  <div class="chat__container">
8      {% if messages %}
9      <ul class="message__list">
10         {% for message in messages %}
11         <li class="message__item">
12             <div class="message__flex">
13                 <p class="message__text">
14                     <span class="message__sender">{{message.sender}} </span>
15                     ->
16                     <span class="message__recipient"> {{message.recipient}}</span>
17                     : {{message.text}}
18                 </p>
19                 <span class="message__date"> {{message.send_date}}</span>
20             </div>
21         </li>
22         {% endfor %}
23     </ul>
24     {% else %}
25     <p class="chat__empty">
26         There are no messages with {{user_1}} and {{user_2}} yet.
27     </p>
28     {% endif %}

```

2.4. Работа с формами

Написана форма и используемый с ней view для отправки сообщения:

```

<form
    class="chat__form"
    action="{% url 'message:send' user_2.id %}"
    method="post"
>
    {% csrf_token %}
    <input
        class="chat__form-text"
        type="text"
        name="text"
        placeholder="type your message"
    />
    <input class="chat__form-submit" type="submit" value="Send" />
</form>

```

```

45 def send(request, user_pk):
46     if not request.user.is_authenticated:
47         login_url = reverse(settings.LOGIN_URL)
48         return redirect(login_url)
49
50     user_1 = request.user
51     user_2 = get_object_or_404(CustomUser, pk=user_pk)
52
53     text = request.POST["text"]
54
55     message = Message(
56         sender=user_1, recipient=user_2, text=text, send_date=timezone.now()
57     )
58     message.save()
59
60     return HttpResponseRedirect(reverse("message:chat", args=(user_pk,)))

```

2.5. Тесты

Созданы тесты для Message модели, а конкретно для её метода was_sent_recently (тест с сообщением из будущего, тест со старым сообщением и тест с недавним сообщением):

```

message > tests.py > MessageModelTests
11 class MessageModelTests(TestCase):
12     def test_was_sent_recently_with_future_message(self):
13         """
14         was_sent_recently() returns False for messages whose send_date
15         is in the future.
16         """
17         time = timezone.now() + datetime.timedelta(days=30)
18         future_message = Message(send_date=time)
19         self.assertIs(future_message.was_sent_recently(), False)
20
21     def test_was_sent_recently_with_old_message(self):
22         """
23         was_sent_recently() returns False for messages whose send_date
24         is older than 1 day.
25         """
26         time = timezone.now() - datetime.timedelta(days=1, seconds=1)
27         old_message = Message(send_date=time)
28         self.assertIs(old_message.was_sent_recently(), False)
29
30     def test_was_sent_recently_with_recent_message(self):
31         """
32         was_sent_recently() returns True for messages whose send_date
33         is within the last day.
34         """
35         time = timezone.now() - datetime.timedelta(hours=23, minutes=59, seconds=59)
36         recent_message = Message(send_date=time)
37         self.assertIs(recent_message.was_sent_recently(), True)
38

```

Также написан тест для chat view на проверку status_code, response и context:

```

40 class ChatTests(TestCase):
41     def test_no_messages(self):
42         """
43         If no messages exist, an appropriate message is displayed.
44         """
45         user_pk = 1
46         response = self.client.get(reverse("message:chat", args=(user_pk,)))
47         self.assertEqual(response.status_code, 200)
48         self.assertContains(response, "There are no messages with and yet.")
49         self.assertQuerySetEqual(response.context["messages"], [])
50

```

2.6. Статика

Для улучшения внешнего вида template chat.html был подключен static и написаны CSS-стили:

```

message > templates > message > chat.html > ...
1  {% load static %}
2
3  <link rel="stylesheet" href="{% static 'message/style.css' %}" />

```

```

message > static > message > style.css > .chat
8  .chat {
9      padding: 2rem 4rem 0;
10     font-size: 20px;
11 }
12 .chat__index {
13     background-color: #fff;
14     padding: 1rem;
15     text-decoration: none;
16     border-radius: 15px;
17     color: black;
18 }
19 .chat__index:hover {
20     text-decoration: underline;
21 }

```

Также было добавлено фоновое изображение:

```

message > static > message > style.css > .chat__index:hover
1  body {
2      margin: 0;
3      padding: 0;
4      background: #cbefff url("images/blue-cats.jpg") no-repeat;
5      height: 100vh;
6  }
7

```




2.7. Улучшение admin панели

Чтобы получить удобную работу в админ панели Django проекта, были улучшены admin формы у моделей при помощи inline related objects, fieldsets и т.п.:

```
api > admin.py > PostAdmin
7  class CommentInline(admin.StackedInline):
8      model = Comment
9      extra = 1
10
11
12  class LikeInline(admin.TabularInline):
13      model = Like
14      extra = 1
15
16
17  class PostAdmin(admin.ModelAdmin):
18      fieldsets = [
19          ("User Info", {"fields": ["user"]}),
20          ("Date Info", {"fields": ["creation_date"]}),
21          (None, {"fields": ["text"]}),
22      ]
23      date_hierarchy = "creation_date"
24      inlines = [CommentInline, LikeInline]
25      list_display = ["id", "user", "creation_date", "text"]
26      list_filter = ["user", "creation_date", "text"]
27      search_fields = ["text"]
```

Также был немного кастомизирован template admin base_site.html

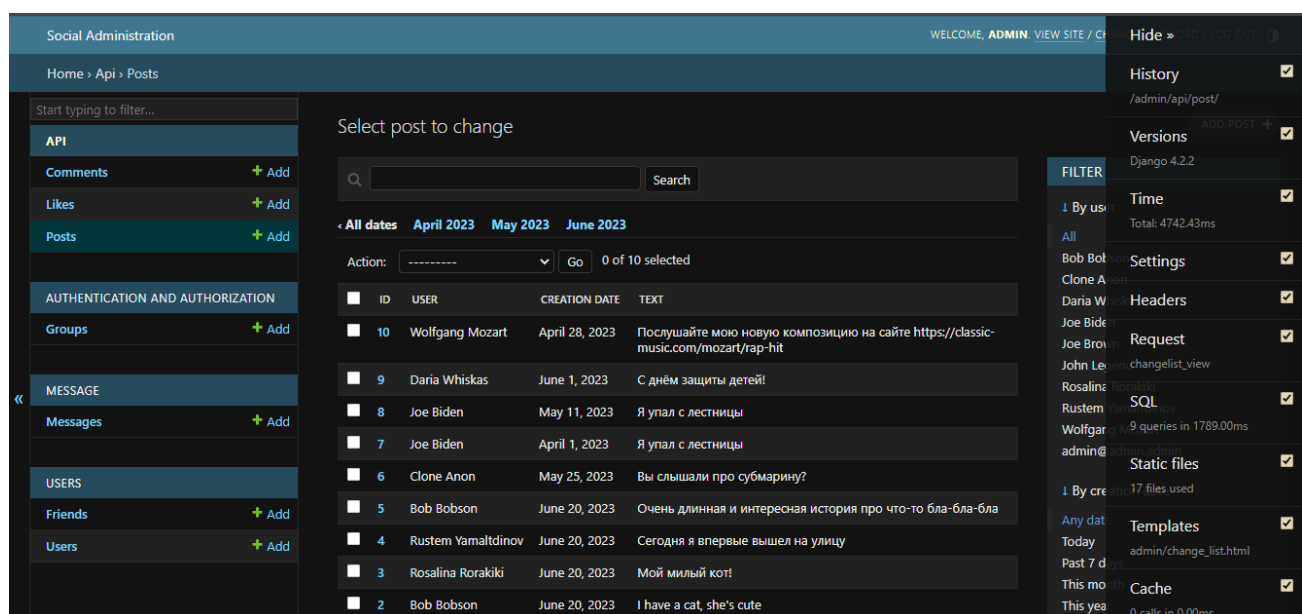
```

templates > admin >  base_site.html > ...
1  {% extends "admin/base.html" %}
2
3  {% block title %}{% if subtitle %}{{ subtitle }}{% endif %}{{ title %}}
4
5  {% block branding %}
6  <div id="site-name">
7   <a href="{% url 'admin:index' %}">Social Administration</a>
8  </div>
9  {% if user.is_anonymous %}
10  {% include "admin/color_theme_toggle.html" %}
11 {% endif %}
12 {% endblock %}
13
14 {% block nav-global %}{% endblock %}

```

2.8. Установка сторонних пакетов

Установлен и настроен Django Debug Toolbar:



The screenshot displays the Django Social Administration web application. The top navigation bar includes the site name 'Social Administration' and a user profile 'WELCOME, ADMIN'. The main content area shows a list of posts with columns for ID, USER, CREATION DATE, and TEXT. The left sidebar contains navigation links for API, Comments, Likes, Posts, AUTHENTICATION AND AUTHORIZATION, Groups, MESSAGE, Messages, USERS, Friends, and Users. The right sidebar features the Django Debug Toolbar (DDT) with various tools: History, Versions, Time, Settings, Headers, Request, SQL, Static files, Templates, and Cache. The DDT panel shows the current request details, including the URL, method, status, and response time.

§3. Глубокая настройка админ-панели Django

Для хорошего отображения записей в таблицах у каждой модели добавлен `list_display` с нужными полями; там, где это требуется, реализованы `list_filter`, `search_fields`, `date_hierarchy`; а для удобного изменения записей настроены `fieldsets`

```
message > admin.py > ...
6 class MessageAdmin(admin.ModelAdmin):
7     fieldsets = [
8         ("Users Info", {"fields": ["sender", "recipient"]}),
9         ("Date Info", {"fields": ["send_date"]}),
10        (None, {"fields": ["text"]}),
11    ]
12
13    date_hierarchy = "send_date"
14
15    list_display = [
16        "id",
17        "sender",
18        "recipient",
19        "send_date",
20        "text",
21        "was_sent_recently",
22    ]
23
24    list_filter = [
25        "sender",
26        "recipient",
27        "send_date",
28        "text",
29    ]
30    search_fields = [
31        "text",
32    ]
```

```
api > admin.py > PostAdmin
30 class CommentAdmin(admin.ModelAdmin):
31     list_display = ["id", "user", "post", "creation_date", "text"]
32     date_hierarchy = "creation_date"
33     list_filter = ["user", "post", "creation_date", "text"]
34     search_fields = ["text"]
35
36
37 class LikeAdmin(admin.ModelAdmin):
38     list_display = ["id", "user", "post", "date_added"]
39     date_hierarchy = "date_added"
40     list_filter = ["user", "post", "date_added"]
```

§4. Реализованный API

4.1. Посты

- 1) `api/posts/` – GET-запрос на данный URL выведет список постов всех пользователей. Имеется пагинация с query параметрами `limit` и `page`;
- 2) `api/posts/create/` – POST-запрос на данный URL позволяет создать новый пост с переданным `text`. Требуется авторизация;
- 3) `api/posts/<int:post_pk>/` – GET-запрос на данный URL возвращает пост с определённым `id = post_pk`;
- 4) `api/posts/<int:post_pk>/delete` – POST-запрос на данный URL позволяет пользователю удалить свой пост с определённым `id = post_pk`. Требуется авторизация.

4.2. Комментарии к постам

Каждый API комментарий связан с определённым постом и требует конкретизации `api/posts/<int:post_pk>/`

- 1) `api/posts/<int:post_pk>/comments` – GET-запрос на данный URL возвращает список всех комментариев к посту с определённым `id = post_pk`. Поддерживает пагинацию;
- 2) `api/posts/<int:post_pk>/comments/create` – POST-запрос на данный URL позволяет создать новый комментарий к определённому посту с переданным `text`. Требуется авторизация;
- 3) `api/posts/<int:post_pk>/comment/<int:comment_pk>/` – GET-запрос на данный URL возвращает конкретный комментарий к конкретному посту;
- 4) `api/posts/<int:post_pk>/comment/<int:comment_pk>/delete/` – POST-запрос на данный URL позволяет пользователю удалить конкретный свой комментарий к конкретному посту. Требуется авторизация.

4.3. Лайки к постам

Каждый API лайк связан с определённым постом и требует конкретизации `api/posts/<int:post_pk>/`

- 1) `api/posts/<int:post_pk>/likes/` – GET-запрос на данный URL возвращает количество лайков к конкретному посту; POST-запрос требует авторизации и позволяет поставить лайк на конкретный пост: DELETE-запрос требует авторизации и позволяет пользователю убрать свой лайк с конкретного поста.

4.4. Пользователи

- 1) `users/` – GET-запрос на данный URL возвращает список зарегистрированных пользователей. Поддерживает пагинацию;

- 2) users/<int:user_pk>/ – GET-запрос на данный URL возвращает пользователя с конкретным id;
- 3) users/<int:user_pk>/friend/ – POST-запрос на данный URL позволяет создать пользователю новую дружбу с другим конкретным пользователем. Требуется авторизации;
- 4) users/<int:user_pk>/unfriend/ – POST-запрос на данный URL позволяет удалить пользователю существующую дружбу с другим конкретным пользователем. Требуется авторизации;
- 5) users/register/ – POST-запрос на данный URL позволяет зарегистрировать нового пользователя с полями email, firstname, lastname и password;
- 6) users/login/ – POST-запрос на данный URL позволяет авторизоваться пользователю с введенными email и password;
- 7) users/logout/ – POST-запрос на данный URL позволяет пользователю выйти из аккаунта.

4.5. Сообщения

- 1) messages/<int:user_pk>/ – GET-запрос на данный URL возвращает html-страницу с окном вывода всех сообщений этого пользователя с указанным конкретным пользователем и с формой отправки нового сообщения. Требуется авторизации;
- 2) messages/<int:user_pk>/send – POST-запрос на данный URL с полем text позволяет создать новое сообщение для указанного пользователя. Требуется авторизации.

Заключение

В итоге получился backend-сервис с гибким API и хорошей админкой, который можно использовать при разработке frontend-приложения. Весь django проект можно найти в github репозитории <https://github.com/rustem-yam/coursework>. Визуализированная схема базы данных находится по адресу <https://dbdesigner.page.link/KgxviFZrBARQYucf8>

Список использованных источников

1. <https://docs.djangoproject.com/en/4.2/> – официальная документация Django
2. <https://stackoverflow.com/> – полезный форум ответов на вопросы
3. <https://erd.dbdesigner.net/> – веб-сервис для визуализации схемы бд
4. <https://railway.app/> – бесплатный сервис с возможностью создания бд
5. <https://habr.com/ru/articles/348560/> – статья про отношения в бд

Select post to change

< All dates
April 2023
May 2023
June 2023

Action: 0 of 10 selected

| <input type="checkbox"/> | ID | USER | CREATION DATE | TEXT |
|--------------------------|----|--------------------|----------------|--|
| <input type="checkbox"/> | 10 | Wolfgang Mozart | April 28, 2023 | Послушайте мою новую композицию на сайте https://classic-music.com/mozart/rap-hit |
| <input type="checkbox"/> | 9 | Daria Whiskas | June 1, 2023 | С днём защиты детей! |
| <input type="checkbox"/> | 8 | Joe Biden | May 11, 2023 | Я упал с лестницы |
| <input type="checkbox"/> | 7 | Joe Biden | April 1, 2023 | Я упал с лестницы |
| <input type="checkbox"/> | 6 | Clone Anon | May 25, 2023 | Вы слышали про субмарину? |
| <input type="checkbox"/> | 5 | Bob Bobson | June 20, 2023 | Очень длинная и интересная история про что-то бла-бла-бла |
| <input type="checkbox"/> | 4 | Rustem Yamaltdinov | June 20, 2023 | Сегодня я впервые вышел на улицу |
| <input type="checkbox"/> | 3 | Rosalina Rorakiki | June 20, 2023 | Мой милый кот! |
| <input type="checkbox"/> | 2 | Bob Bobson | June 20, 2023 | I have a cat, she's cute |
| <input type="checkbox"/> | 1 | Rustem Yamaltdinov | June 20, 2023 | New post |

10 posts

Select comment to change



< All dates May 2023 June 2023

Action: 0 of 15 selected

| <input type="checkbox"/> | ID | USER | POST |
|--------------------------|----|--------------------|---|
| <input type="checkbox"/> | 17 | John Legend | Wolfgang Mozart: Послушайте мою новую композицию на сайте https://classic-mu |
| <input type="checkbox"/> | 16 | Rosalina Rorakiki | Joe Biden: Я упал с лестницы |
| <input type="checkbox"/> | 15 | Daria Whiskas | Joe Biden: Я упал с лестницы |
| <input type="checkbox"/> | 14 | Daria Whiskas | Joe Biden: Я упал с лестницы |
| <input type="checkbox"/> | 13 | Bob Bobson | Joe Biden: Я упал с лестницы |
| <input type="checkbox"/> | 12 | John Legend | Clone Anon: Вы слышали про субмарину? |
| <input type="checkbox"/> | 11 | Rustem Yamaltdinov | Bob Bobson: Очень длинная и интересная история про что-то бла-бла-бла |
| <input type="checkbox"/> | 10 | Rustem Yamaltdinov | Rustem Yamaltdinov: New post |
| <input type="checkbox"/> | 9 | Bob Bobson | Rustem Yamaltdinov: Сегодня я впервые вышел на улицу |
| <input type="checkbox"/> | 8 | Bob Bobson | Rustem Yamaltdinov: Сегодня я впервые вышел на улицу |
| <input type="checkbox"/> | 7 | Rustem Yamaltdinov | Rosalina Rorakiki: Мой милый кот! |

Select like to change

< All dates May 2023 June 2023

Action: 0 of 14 selected

| <input type="checkbox"/> | ID | USER | POST | DATE |
|--------------------------|----|--------------------|---|-----------|
| <input type="checkbox"/> | 19 | Joe Biden | Joe Biden: Я упал с лестницы | June 2023 |
| <input type="checkbox"/> | 18 | Clone Anon | Joe Biden: Я упал с лестницы | May 2023 |
| <input type="checkbox"/> | 17 | Rustem Yamaltdinov | Joe Biden: Я упал с лестницы | June 2023 |
| <input type="checkbox"/> | 16 | Rustem Yamaltdinov | Clone Anon: Вы слышали про субмарину? | June 2023 |
| <input type="checkbox"/> | 15 | Wolfgang Mozart | Clone Anon: Вы слышали про субмарину? | June 2023 |
| <input type="checkbox"/> | 14 | Daria Whiskas | Clone Anon: Вы слышали про субмарину? | June 2023 |
| <input type="checkbox"/> | 13 | admin@admin.admin | Rosalina Rorakiki: Мой милый кот! | June 2023 |
| <input type="checkbox"/> | 11 | Bob Bobson | Rosalina Rorakiki: Мой милый кот! | June 2023 |
| <input type="checkbox"/> | 10 | Rustem Yamaltdinov | Bob Bobson: Очень длинная и интересная история про что-то бла-бла-бла | June 2023 |
| <input type="checkbox"/> | 8 | Bob Bobson | Rustem Yamaltdinov: New post | June 2023 |
| <input type="checkbox"/> | 7 | Bob Bobson | Bob Bobson: Очень длинная и интересная история про что-то бла-бла-бла | June 2023 |
| <input type="checkbox"/> | 6 | Rustem Yamaltdinov | Rustem Yamaltdinov: Сегодня я впервые вышел на улицу | June 2023 |
| <input type="checkbox"/> | 4 | Rosalina Rorakiki | Rustem Yamaltdinov: New post | June 2023 |
| <input type="checkbox"/> | 2 | Rustem Yamaltdinov | Rustem Yamaltdinov: New post | June 2023 |

Select message to change



< 2023 June 28

Action: 0 of 10 selected

| <input type="checkbox"/> | ID | SENDER | RECIPIENT | SEND DATE | TEXT | SENT RECENTLY? |
|--------------------------|----|--------------------|--------------------|---------------------------|-----------------------------------|----------------|
| <input type="checkbox"/> | 10 | admin@admin.admin | Rosalina Rorakiki | June 28, 2023, 5:13 p.m. | Ку-ку | ✓ |
| <input type="checkbox"/> | 9 | Rosalina Rorakiki | admin@admin.admin | June 28, 2023, 5:11 p.m. | Hello | ✓ |
| <input type="checkbox"/> | 8 | Bob Bobson | Rustem Yamaltdinov | June 28, 2023, 5:10 p.m. | Wow! Cool | ✓ |
| <input type="checkbox"/> | 7 | Rustem Yamaltdinov | Bob Bobson | June 28, 2023, 4:34 p.m. | Абсолютно не тестовый текст | ✓ |
| <input type="checkbox"/> | 6 | Rustem Yamaltdinov | Bob Bobson | June 28, 2023, 12:53 p.m. | I'm testing new form | ✓ |
| <input type="checkbox"/> | 5 | Rustem Yamaltdinov | Bob Bobson | June 28, 2023, 12:53 p.m. | Oops, double sent | ✓ |
| <input type="checkbox"/> | 4 | Rustem Yamaltdinov | Bob Bobson | June 28, 2023, 12:52 p.m. | What's up? | ✓ |
| <input type="checkbox"/> | 3 | Rustem Yamaltdinov | Bob Bobson | June 28, 2023, 12:52 p.m. | What's up? | ✓ |
| <input type="checkbox"/> | 2 | Rustem Yamaltdinov | Bob Bobson | June 28, 2023, 10:25 a.m. | Hi! | ✓ |
| <input type="checkbox"/> | 1 | Bob Bobson | Rustem Yamaltdinov | June 28, 2023, midnight | Hello | ✗ |

10 messages

Select user to change

Search

< All dates May 2023 June 2023

Action:

 Go 0 of 10 selected

| <input type="checkbox"/> | ID | EMAIL | FIRSTNAME | 1 ▲ | LASTNAME | 2 ▲ | PASSWORD |
|--------------------------|----|----------------------|-------------------|-----|-------------|-----|-------------------------------------|
| <input type="checkbox"/> | 9 | osueo@soug.sougb | Bob | | Bobson | | pbkdf2_sha256\$600000\$2PQ5rWfkHP |
| <input type="checkbox"/> | 16 | ogaeion@mail.ru | Clone | | Anon | | pbkdf2_sha256\$600000\$Ge1ZE2xWuh |
| <input type="checkbox"/> | 15 | qgipno@yandex.ru | Daria | | Whiskas | | pbkdf2_sha256\$600000\$Glm1hj8IPOU |
| <input type="checkbox"/> | 17 | yoursecret@gmail.com | Joe | | Biden | | pbkdf2_sha256\$600000\$x7mwvb3V5p |
| <input type="checkbox"/> | 13 | noab@nksd.wroi | Joe | | Brown | | pbkdf2_sha256\$600000\$uqEucHklkRO |
| <input type="checkbox"/> | 18 | example@mail.ru | John | | Legend | | pbkdf2_sha256\$600000\$Kb7JGSu9Ftj- |
| <input type="checkbox"/> | 10 | rosa@mail.ru | Rosalina | | Rorakiki | | pbkdf2_sha256\$600000\$NG5v3jhDBA: |
| <input type="checkbox"/> | 6 | test@test.test | Rustem | | Yamaltdinov | | pbkdf2_sha256\$600000\$oALIF5WjgaG |
| <input type="checkbox"/> | 14 | opwof@gmail.com | Wolfgang | | Mozart | | pbkdf2_sha256\$600000\$6NHFKjnRhwl |
| <input type="checkbox"/> | 4 | admin | admin@admin.admin | | | | pbkdf2_sha256\$600000\$hCLBSmt2eSC |

10 users

Select friend to change

[ADD FRIEND +](#)

< All dates March 2023 April 2023 June 2023

Action:

 Go 0 of 10 selected

| <input type="checkbox"/> | USER 1 | USER 2 | DATE ADDED |
|--------------------------|-------------------|--------------------|----------------|
| <input type="checkbox"/> | admin@admin.admin | Joe Brown | June 28, 2023 |
| <input type="checkbox"/> | admin@admin.admin | Rosalina Rorakiki | March 8, 2023 |
| <input type="checkbox"/> | Bob Bobson | Daria Whiskas | March 31, 2023 |
| <input type="checkbox"/> | Clone Anon | Joe Brown | June 4, 2023 |
| <input type="checkbox"/> | Joe Biden | Wolfgang Mozart | April 6, 2023 |
| <input type="checkbox"/> | Clone Anon | Rustem Yamaltdinov | June 11, 2023 |
| <input type="checkbox"/> | Daria Whiskas | Rustem Yamaltdinov | June 28, 2023 |
| <input type="checkbox"/> | Bob Bobson | Rustem Yamaltdinov | June 18, 2023 |
| <input type="checkbox"/> | Bob Bobson | Joe Biden | June 28, 2023 |
| <input type="checkbox"/> | admin@admin.admin | Bob Bobson | June 29, 2023 |

10 friends