Blog

Введение

Идея проекта заключается в создании социальной сети, в которой можно обмениваться информацией путём публикации новостей и комментирования их.

Описание реализации

В проекте использовались библиотеки flask, flask-login, flask-wtf, sqlalchemy, wtforms, datetime и оз (для Heroku)

```
from flask import Flask, render template, redirect, request, abort
from flask login import LoginManager
from flask login import login user, logout user, login required, current user
from forms.users form import RegisterForm, LoginForm, RefactorForm
from forms.news form import NewsForm
from forms.comment form import CommentForm
from data.news class import News
from data.users class import User
from data.comment class import Comment
from data import db session
import os
from datetime import datetime
```

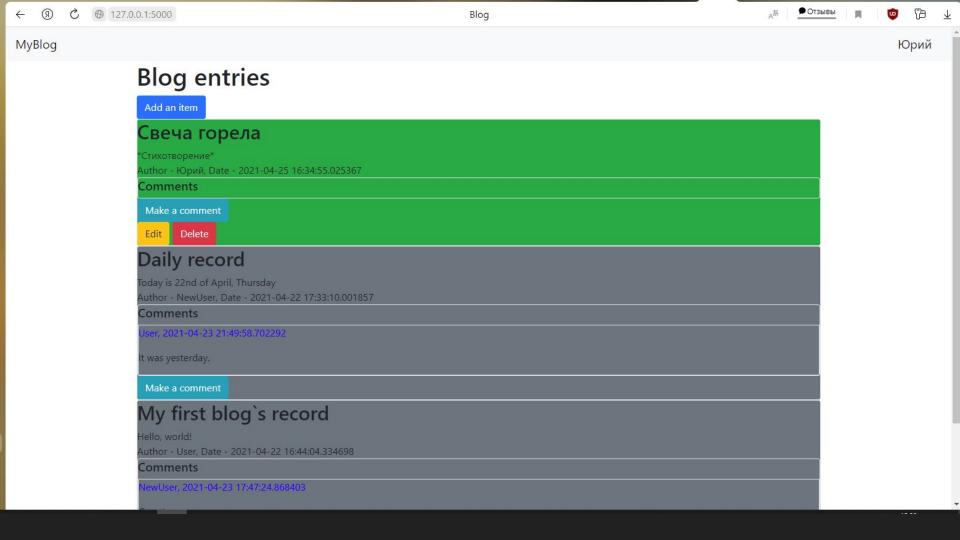
```
from wtforms import PlaskForm

from wtforms import PasswordField, StringField

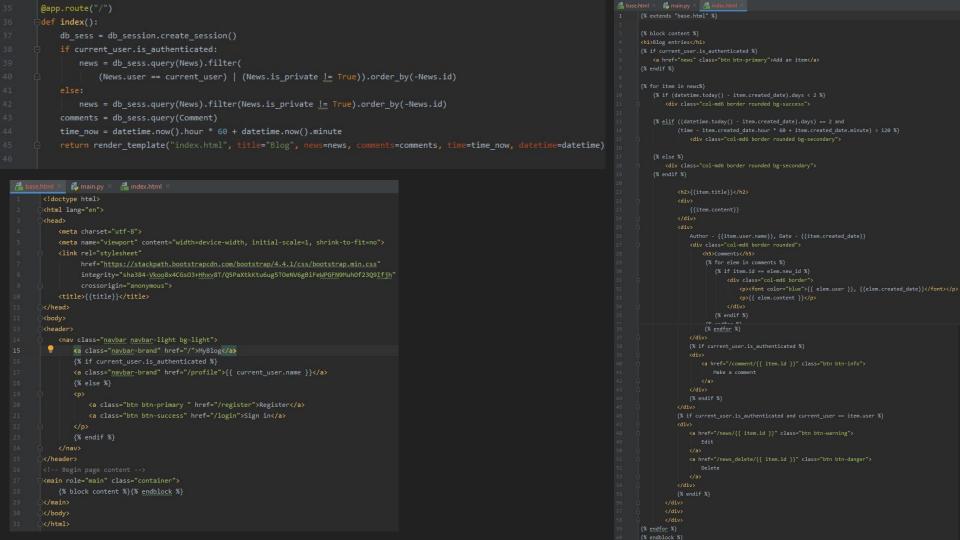
from wtforms.fields.html5 import EmailField

from wtforms.validators import DataRequired

5
```



За главную страницу отвечают метод файла main.py init() и файлы base.html и index.html. В них определяется то, что зашёл ли пользователь в свой аккаунт, кнопки register и sign up, если нет и кнопка для перехода на профиль, если да, а также кнопки для создания новости или комментария и редактирования или удаления своих прошлых новостей (если пользователь зашел в аккаунт) и кнопка MyBlog для перехода на главную страницу. Также эти файлы отвечают за то, чтобы показывать все комментарии и новости, кроме приватных (если пользователь не автор публикации). Если новости меньше 2 дней и 2 часов, то она окрашена в зелёный цвет, другие же публикации в серый.



За написание, редактирование и удаление новостей отвечают методы главного файла add_news, news_delete и edit_news и файлы news_form.py, news_class.py и news.html. В методы редактирования и удаления передаётся id новости, чтобы определить нужную новость

```
@app.route('/news', methods=['GET', 'POST'])

@login_required

def add_news():
    form = NewsForm()

if form.validate_on_submit():
    db_sess = db_session.create_session()

news = News()
    news.title = form.title.data

news.content = form.content.data

news.is_private = form.is_private.data

current_user.news.append(news)

db_sess.merge(current_user)

db_sess.commit()

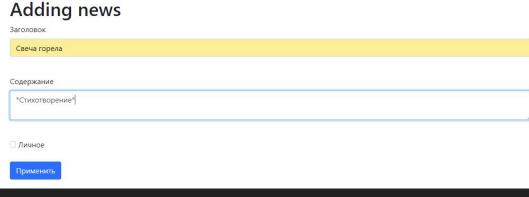
return redirect('/')

return render_template('news.html', title='Adding news',

form=form)
```

```
@login required
def edit news(id):
    form = NewsForm()
    if request.method == "GET":
        db sess = db session.create session()
        news = db sess.query(News).filter(News.id == id,
                                          News.user == current user
        if news:
            form.title.data = news.title
            form.content.data = news.content
            form.is private.data = news.is private
            abort(404)
    if form.validate on submit():
        db_sess = db_session.create_session()
        news = db sess.query(News).filter(News.id == id,
                                          News.user == current user
                                          ).first()
        if news:
            news.title = form.title.data
            news.content = form.content.data
            news.is private = form.is private.data
            db sess.commit()
    return render template('news.html',
                           form=form
```

```
👸 main.py 👋 👸 news_form.py 💉 👸 news_class.py 🗡 🗂 news.html 🔾
                                                                                 👸 main.py 👋 👸 news_form.py 🗡 👸 news_class.py 🗡 📇 news.html 👋
      from flask wtf import FlaskForm
                                                                                         {% extends "base.html" %}
      fem wtforms import StringField, TextAreaField
      from wtforms import BooleanField, SubmitField
      from wtforms.validators import DataRequired
                                                                                         {% block content %}
                                                                                         <h1>Adding news</h1>
                                                                                         <form action="" method="post">
      class NewsForm(FlaskForm):
                                                                                             {{ form.hidden tag() }}
          title = StringField('Заголовок', validators=[DataRequired()])
          content = TextAreaField("Содержание")
                                                                                                 {{ form.title.label }}<br>
          is private = BooleanField("Личное")
                                                                                                 {{ form.title(class="form-control") }} <br>
          submit = SubmitField('Применить')
                                                                                                 {% for error in form.title.errors %}
                                                                                                     {{ error }}
👸 main.py × 👸 news_form.py × 👸 news_class.py × 🛗 news.html
      import datetime
                                                                                                 {% endfor %}
      import sqlalchemy
      from sqlalchemy import orm
                                                                                                 {{ form.content.label }}<br>
                                                                                                 {{ form.content(class="form-control") }}<br>
                                                                                                 {% for error in form.content.errors %}
      class News(SqlAlchemyBase):
          tablename = 'news'
                                                                                                     {{ error }}
          id = sqlalchemy.Column(sqlalchemy.Integer,
                                                                                                 {% endfor %}
          title = sqlalchemy.Column(sqlalchemy.String, nullable=True)
          created date = sqlalchemy.Column(sqlalchemy.DateTime,
                                                                                             {{p>{{ form.is private() }} {{ form.is private.label }}
                                     default=datetime.datetime.now)
                                                                                             {{ form.submit(type="submit", class="btn btn-primary") }}
          is private = sqlalchemy.Column(sqlalchemy.Boolean, default=True)
                                                                                             {{message}}
                                                                                        </form>
                                                                                         {% endblock %}
          user = orm.relation('User')
```



Добавление/редактирование новости (одинаково выглядят)

Blog entries
Add an item
Новость
Текст Author - Юрий, Date - 2021-04-26 17:10:37.884964
Comments
Make a comment Edit Delete
Свеча горела
"Стихотворение" Author - Юрий, Date - 2021-04-26 17:10:25.720983
Comments
Make a comment Edit Delete

Blog entries			
Add an item			
Новость			
Текст Author - Юрий, Date - 2021-04-26 17:10:37.8	84964		
Comments			
Make a comment			
Edit Delete			
Daily record			

Удаление новости "Свеча горела"

За написание комментариев отвечают метод главного файла add_comment и файлы comment_class.py, comment_form.py и comment.html. При вызове метода добавления комментариев передаётся іd новости, чтобы поставить комментарий под нужной новостью

```
@app.route('/comment/<int:id>', methods=['GET', 'POST'])
@login required
def add comment(id):
    form = CommentForm()
    if form.validate on submit():
        db sess = db session.create session()
        comment = Comment()
        comment.content = form.content.data
        comment.user = current user.name
        comment.new id = id
        db sess.add(comment)
        db sess.commit()
        return redirect('/')
    return render template('comment.html', title='Adding comments',
                           form=form)
```

```
🐞 comment_class.py × 🐞 comment_form.py ×
                                                                            main.py
            tomment_class.py × tomment_form.py × tomment.html >
main.py X
                                                                                  {% extends "base.html" %}
       import datetime
      import sqlalchemy
                                                                                   {% block content %}
                                                                                  <form action="" method="post">
                                                                                      {{ form.hidden tag() }}
      from .db session import SqlAlchemyBase
                                                                                          {{ form.content.label }}<br>
                                                                                          {{ form.content(class="form-control") }}<br>
      class Comment(SqlAlchemyBase):
                                                                                          {% for error in form.content.errors %}
           tablename = 'comments'
                                                                                              {{ error }}
          id = sqlalchemy.Column(sqlalchemy.Integer,
                                                                                          {% endfor %}
          content = sqlalchemy.Column(sqlalchemy.String, nullable=True)
          created date = sqlalchemy.Column(sqlalchemy.DateTime,
                                                                                      {{ form.submit(type="submit", class="btn btn-primary") }}
                                         default=datetime.datetime.now)
                                                                                      {{message}}
          user = sqlalchemy.Column(sqlalchemy.String)
                                                                                  </form>
          new id = sqlalchemy.Column(sqlalchemy.Integer)
                                                                                   {% endblock %}
  Комментарий
```



Отправить

За регистрацию аккаунта, вход в него и редактирование аккаунта отвечают функции главного файла register (регистрация), login (вход), profile (страница профиля), edition (редактирование) и logout (выход) и файлы users_form.py, users_class.py, register.html, login.html, profile.html и edition.html

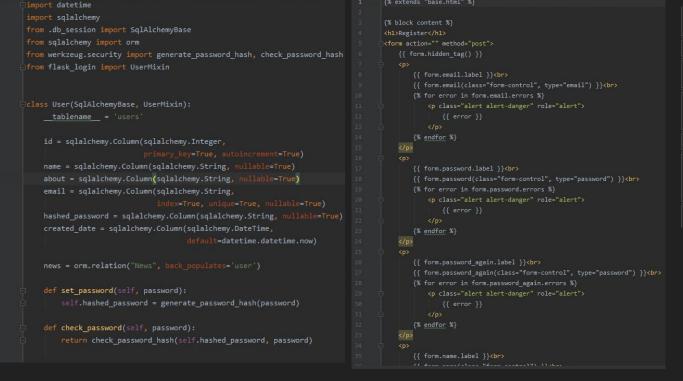
```
@app.route('/register', methods=['GET', 'POST'])
                                                                                    @app.route('/login', methods=['GET', 'POST'])
|def register():
                                                                                    |def login():
   form = RegisterForm()
                                                                                        form = LoginForm()
   if form.validate on submit():
                                                                                        if form.validate on submit():
       if form.password.data != form.password again.data:
                                                                                             db sess = db session.create session()
           return render template('register.html', title='Register',
                                                                                             user = db_sess.query(User).filter(User.email == form.email.data).first()
                                 form=form.
                                message="Пароли не совпадают")
                                                                                             if user and user.check password(form.password.data):
       db sess = db session.create session()
                                                                                                 login user(user, remember=form.remember me.data)
       if db sess.query(User).filter(User.email == form.email.data).first():
                                                                                                 return redirect("/")
           return render template('register.html', title='Регистрация',
                                                                                             return render template('login.html',
                                 form=form,
                                                                                                                      message="Неправильный логин или пароль",
                                message="Такой пользователь уже есть")
                                                                                                                      form=form)
           name=form.name.data.
                                                                                        return render template('login.html', title='Authorization', form=form)
           email=form.email.data.
           about=form.about.data
                                                                                    @app.route('/profile')
       user.set password(form.password.data)
                                                                                    @login required
       db sess.add(user)
       db sess.commit()
                                                                                    def profile():
       return redirect('/login')
                                                                                        return render template('profile.html', title="Profile")
```

```
fom flask wtf import FlaskForm
def edition():
                                                                                                 from wtforms import PasswordField, StringField, TextAreaField, SubmitField, BooleanField
    form = RefactorForm()
                                                                                                 from wtforms.fields.html5 import EmailField
    if request.method == "GET":
                                                                                                 from wtforms.validators import DataRequired
        db sess = db session.create session()
        user = db sess.query(User).filter(User.id == current user.id).first()
        if user:
                                                                                                class RegisterForm(FlaskForm):
            form.name.data = user.name
                                                                                                     email = EmailField('Nouta', validators=[DataRequired()])
            form.about.data = user.about
                                                                                                     password = PasswordField('Пароль', validators=[DataRequired()])
                                                                                                     password_again = PasswordField('Повторите пароль', validators=[DataRequired()])
            abort(404)
                                                                                                     name = StringField('Имя пользователя', validators=[DataRequired()])
                                                                                                     about = TextAreaField("Hemmoro o ce6e")
    if form.validate on submit():
                                                                                                     submit = SubmitField('Войти')
        db sess = db session.create session()
        user = db sess.query(User).filter(User.id == current user.id).first()
        user.name = form.name.data
                                                                                                class LoginForm(FlaskForm):
        user.about = form.about.data
                                                                                                     email = EmailField('Nouta', validators=[DataRequired()])
        db sess.commit()
                                                                                                     password = PasswordField('Пароль', validators=[DataRequired()])
        return redirect('/profile')
                                                                                                     remember me = BooleanField('Запомнить меня')
                                                                                                     submit = SubmitField('Войти')
    return render template('edition.html', title='Refactoring', form=form)
                                                                                                class RefactorForm(FlaskForm):
@app.route('/logout')
                                                                                                     name = StringField('Имя пользователя', validators=[DataRequired()])
@login required
                                                                                                     about = TextAreaField("HemHoro o ce6e")
def logout():
                                                                                                    confirm = SubmitField('Confirm')
    logout user()
    return redirect("/")
```

👸 users_form.py 👋 👸 users_class.py 🗶 📇 register.html 🗡 🗂 login.html 🗡 🗂 profile.html

Mapp.route('/edition', methods=['GET', 'POST'])

@login required



{% extends "base.html" %}

alogin.html

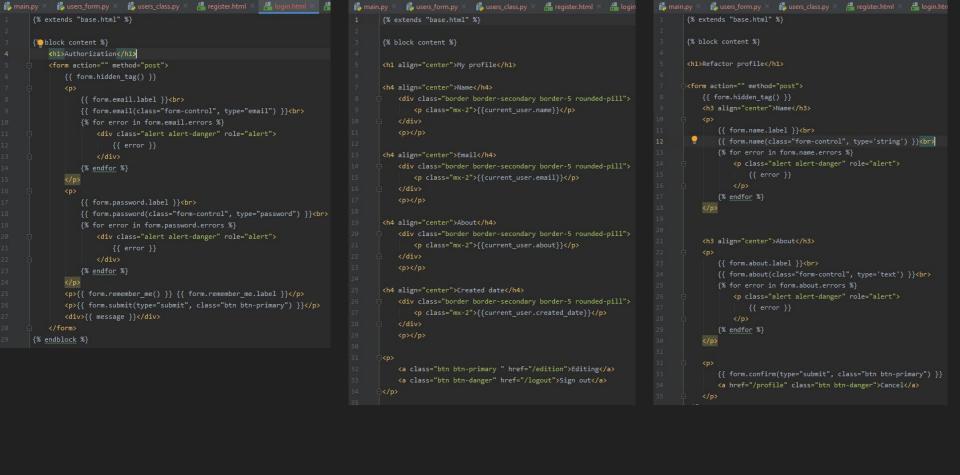
👸 users_class.py 💉 🏭 register.html 🗵

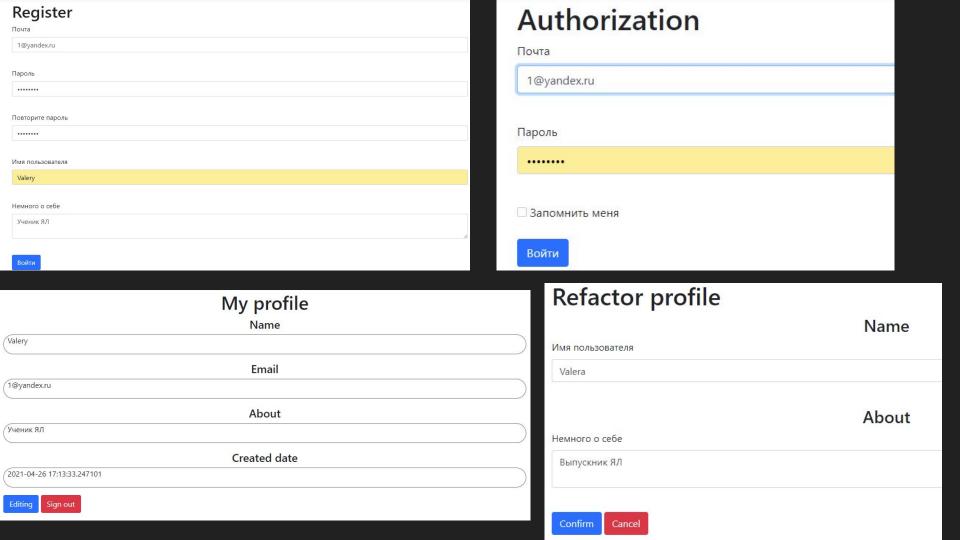
📥 main.py 🔻

👸 users_form.py 🦠

```
{% for error in form.name.errors %}
        {{ error }}
     {% endfor %}
     {{ form.about.label }}<br>
     {{ form.about(class="form-control") }} <br>
        {% endfor %}
  {{ form.submit(type="submit", class="btn btn-primary") }}
{% endblock %}
```

{{ form.name(class="form-control") }}





Вывод

В данном проекте мы сделали простую социальную сеть.

В дальнейшем можно реализовать такие вещи, как просмотр профилей других пользователей, добавление аватара в профиле, улучшение комментариев (функции редактирования и удаления), а также личные сообщения между пользователями.

Спасибо за внимание!