Министерство науки и высшего образования РФ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Лицей Академии Яндекса

Mirox Social Network

Выполнил:

Сизов Иван Александрович

Омск, 2023

РЕФЕРАТ

Предмет исследования – веб-приложение.

Цель исследования – разработка веб-приложения в виде социальной сети для закрепления навыков в изученных ранее темах

В процессе выполнения проекта был получен опыт работы с библиотекой Flask а так же sqlalchemy.

В результате выполнения проекта была создана социальная сеть с возможностью публикации записей у себя на странице и возможностью взаимодействия с ними.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.2r9v0h46yy99)

[1 Описание проблемы и формулирование гипотезы для ее решения 4](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.ifzanbkzs7df)

[2 Проектирование приложения 4](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.pd7u93hyun9o)

[3 Реализация приложения 5](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.67z7petns5sz)

[ЗАКЛЮЧЕНИЕ 8](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.2et92p0)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.tyjcwt)

[ПРИЛОЖЕНИЕ А](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.1t3h5sf) Листинг [10](https://docs.google.com/document/d/1fSqMT2F6PBVnwDeYn31MG4SPsBQf6I_0oJtb799ewhA/edit#heading=h.1t3h5sf)

ВВЕДЕНИЕ

В наше время люди всех возрастов так или иначе проводят своё время в социальных сетях. Социальные сети включают в себя различного типа приложения, например просмотр коротких роликов (TikTok), мессенджер с обширными возможностями общения (Telegram) и многие другие. В своём проекте я решил реализовать свою социальную сеть на базе библиотеки Flask.

Цель исследования – разработка приложения в виде социальной сети для закрепления ранее изученных тем и возможностью её последующего развития.

Для достижения поставленной цели требуется решить следующие задачи:

1. исследовать проблему предметной области ;
2. спроектировать интерфейс приложения и систему хранения данных ;
3. реализовать спроектированное приложение.

**1 Описание проблемы и формулирование гипотезы для ее решения**

Проект был создан в области разработки веб-приложений, а именно социальной сети. В проблемы проекта входит реализация таких функций как:

* Система друзей и подписчиков
* Возможность ставить/убирать лайки
* Написание комментариев под записями
* Добавление фотографий в запись

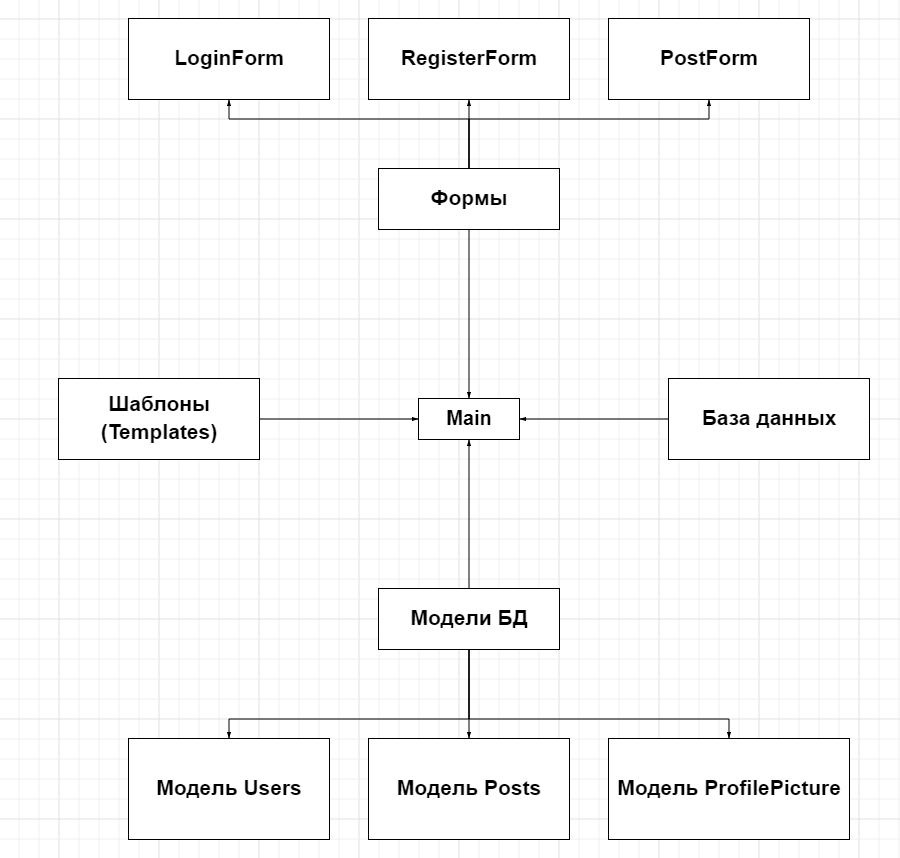
Исследуется самостоятельная реализация этих проблем в социальной

сети, так как имеется желание разобраться как эти системы работают.

**2 Проектирование приложения**

В основу приложения легла библиотека для языка Python под названием Flask. Были реализованы отдельные страницы для просмотра постов пользователей, профиля определённого пользователя, логина и регистрации. Была использована библиотека sqlalchemy для создания системы аккаунтов с их хранением в базе данных. Для создания страниц использовались HTML-шаблоны.

*Схематичное строение проекта:*

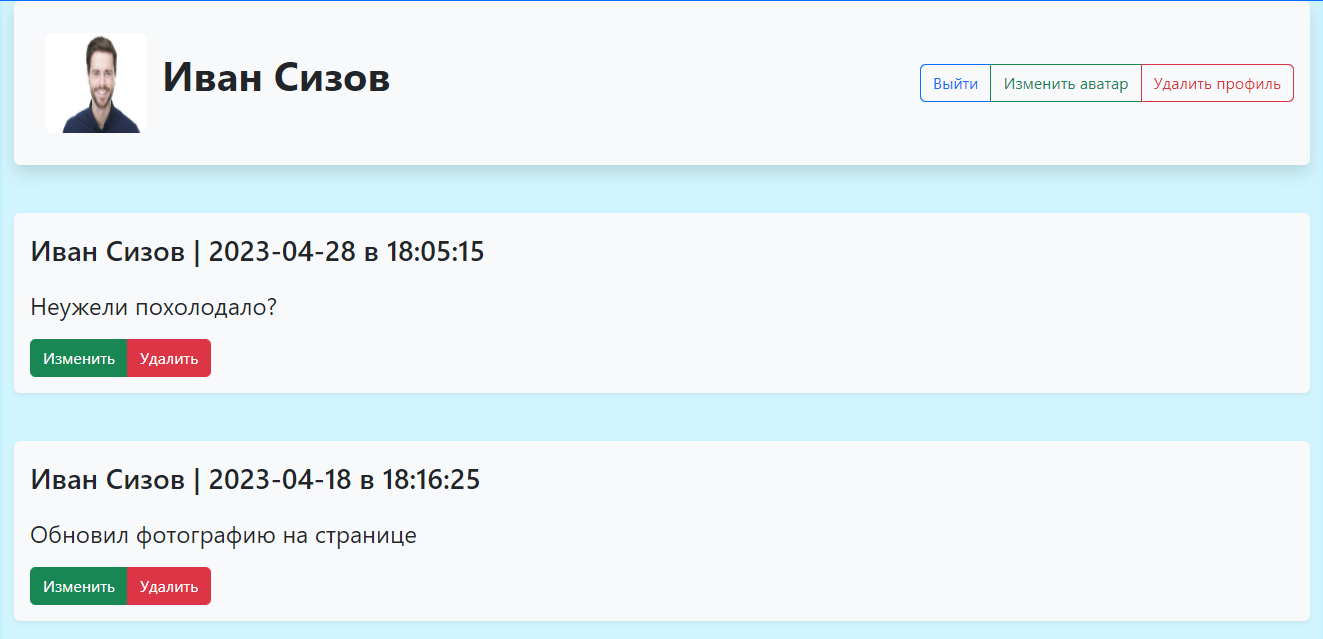


**3 Реализация приложения**

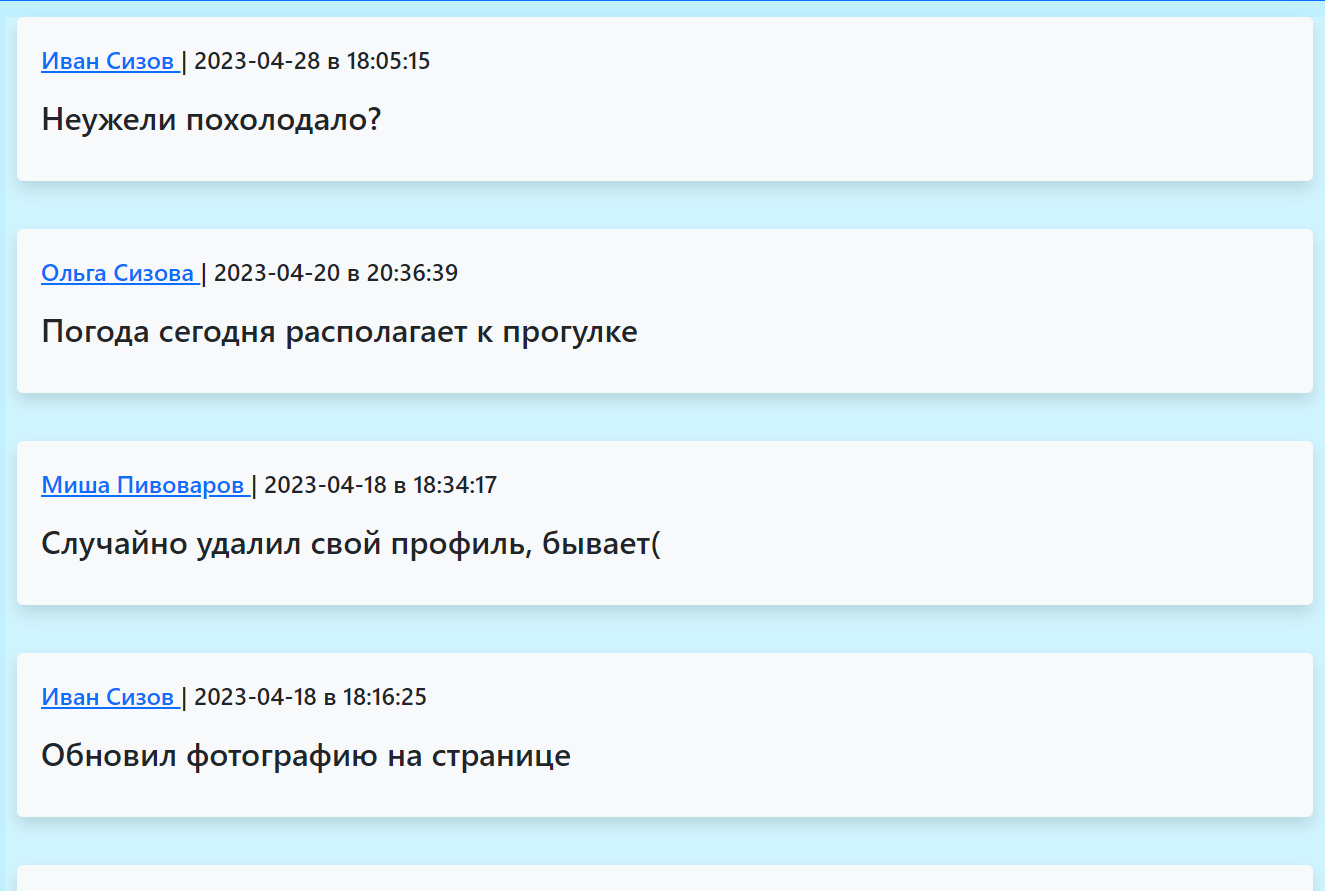
В приложении были реализованы следующие возможности:

* Создание профиля и вход в него
* Удаление профиля
* Смена изображения профиля
* Создание записей от лица пользователя
* Редактирование и удаление определённой записи
* Возможность просмотра профиля другого пользователя

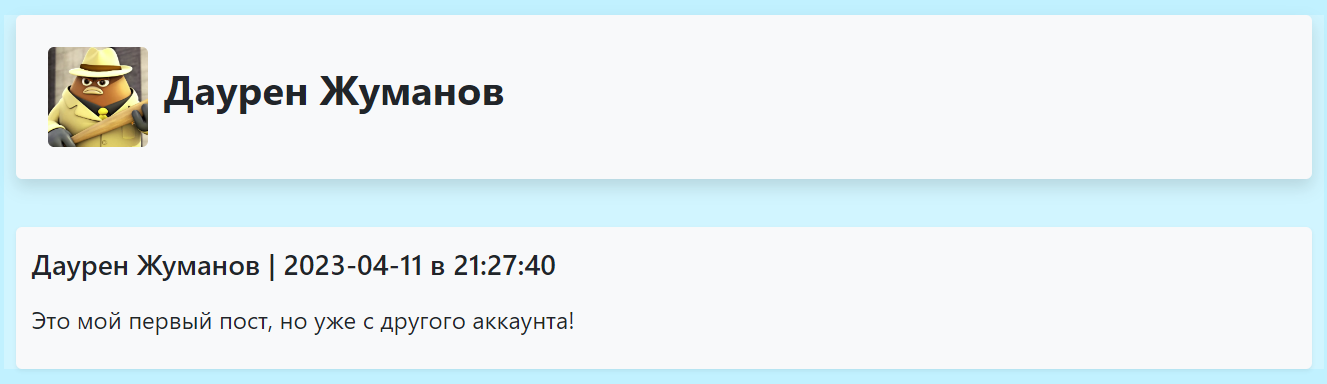
***Внешний вид сайта:***



*Профиль пользователя который вошёл в свой аккаунт*



*Новостная лента*



*Профиль стороннего пользователя*

ЗАКЛЮЧЕНИЕ

В проекте не были реализованы вышеописанные системы, так как мне не удалось разобраться в их строении и интегрировать их в социальную сеть. В будущем обязательно будет рассмотрена возможность реализации этих возможностей в своём проекте.

Мне удалось реализовать социальную сеть начального уровня на базе

языка программирования Python. Была построена система функциональности проекта с использованием различных форм, HTML-шаблонов, базы данных и библиотеки Flask. В базе данных также присутствуют orm-модели. Был реализован стандарт REST-API. Я считаю, что с этой задачей я справился достойно.

Социальная сеть имеет хорошие перспективы развития. В обозримом будущем может быть добавлена возможность публиковать записи с фотографиями. Имеет смысл добавить систему лайков и комментариев, возможность добавлять кого-либо в друзья и многое другое что уже есть в других социальных сетях. MiroxSN создана в первую очередь для опыта работы с социальными сетями и понимания их структуры, хотя в будущем возможен рост до весомого проекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Yandex Lyceum (Уроки и документация к ним)

2 <https://flask.palletsprojects.com/en/latest/> (Документация Flask)

3 <https://docs.sqlalchemy.org/en/20/> (Документация sqlalchemy)

ПРИЛОЖЕНИЕ А

(обязательное)

**Листинг**

import datetime  
import io  
import os  
  
from PIL import Image  
from flask import Flask, render\_template, request, send\_file, url\_for, abort  
from flask\_login import LoginManager, login\_user, login\_required, logout\_user, current\_user  
from flask\_restful import Api  
from werkzeug.utils import secure\_filename, redirect  
  
from data import db\_session, posts\_resources, users\_resourses  
from data.users import User  
from data.posts import Posts  
from data.profile\_pictures import ProfilePicture  
from forms.RegisterForm import RegisterForm  
from forms.LoginForm import LoginForm  
from forms.PostForm import PostForm  
  
app = Flask(\_\_name\_\_)  
api = Api(app)  
app.config['SECRET\_KEY'] = 'mirox\_secretkey'  
  
login\_manager = LoginManager()  
login\_manager.init\_app(app)  
  
  
@login\_manager.user\_loader  
def load\_user(user\_id):  
 db\_sess = db\_session.create\_session()  
 return db\_sess.query(User).get(user\_id)  
  
  
@app.route('/')  
def index():  
 clear\_img()  
 db\_sess = db\_session.create\_session()  
 post = db\_sess.query(Posts).order\_by(Posts.id.desc()).all()  
 return render\_template('feed.html', title="MiroxSN", posts=post)  
  
  
@app.route('/register', methods=['GET', 'POST'])  
def reqister():  
 form = RegisterForm()  
 if form.validate\_on\_submit():  
 if form.password.data != form.password\_again.data:  
 return render\_template('register.html', title='Регистрация',  
 form=form,  
 message="Пароли не совпадают")  
 db\_sess = db\_session.create\_session()  
 if db\_sess.query(User).filter(User.email == form.email.data).first():  
 return render\_template('register.html', title='Регистрация',  
 form=form,  
 message="Такой пользователь уже есть")  
 user = User(  
 name=form.name.data,  
 surname=form.surname.data,  
 email=form.email.data,  
 date\_of\_birth=form.date\_of\_birth.data,  
 created\_date=datetime.datetime.now()  
 )  
 user.set\_password(form.password.data)  
 db\_sess.add(user)  
 db\_sess.commit()  
 return redirect('/')  
 return render\_template('register.html', title='Регистрация', form=form)  
  
  
@app.route('/post', methods=['GET', 'POST'])  
@login\_required  
def add\_post():  
 form = PostForm()  
 if form.validate\_on\_submit():  
 db\_sess = db\_session.create\_session()  
 post = Posts(content=form.content.data,  
 user\_id=current\_user.id)  
 db\_sess.add(post)  
 db\_sess.commit()  
 return redirect('/')  
 return render\_template('post.html', title='Добавить пост', form=form)  
  
  
@app.route('/post/<int:id>', methods=['GET', 'POST'])  
@login\_required  
def edit\_post(id):  
 form = PostForm()  
 if request.method == "GET":  
 db\_sess = db\_session.create\_session()  
 posts = db\_sess.query(Posts).filter(Posts.id == id,  
 Posts.user\_id == current\_user.id  
 ).first()  
 if posts:  
 form.content.data = posts.content  
 else:  
 abort(404)  
 if form.validate\_on\_submit():  
 db\_sess = db\_session.create\_session()  
 posts = db\_sess.query(Posts).filter(Posts.id == id,  
 Posts.user\_id == current\_user.id  
 ).first()  
 if posts:  
 posts.content = form.content.data  
 db\_sess.commit()  
 return redirect('/')  
 else:  
 abort(404)  
 return render\_template('post\_edit.html',  
 title='Редактирование поста',  
 form=form  
 )  
  
  
@app.route('/post\_delete/<int:id>', methods=['GET', 'POST'])  
@login\_required  
def post\_delete(id):  
 db\_sess = db\_session.create\_session()  
 posts = db\_sess.query(Posts).filter(Posts.id == id,  
 Posts.user\_id == current\_user.id  
 ).first()  
 if posts:  
 db\_sess.delete(posts)  
 db\_sess.commit()  
 else:  
 abort(404)  
 return redirect('/')  
  
  
@app.route('/account\_delete/<int:id>', methods=['GET', 'POST'])  
def account\_delete(id):  
 db\_sess = db\_session.create\_session()  
 user = db\_sess.query(User).filter(User.id == id, User.id == current\_user.id).first()  
 posts = db\_sess.query(Posts).filter(  
 Posts.user\_id == current\_user.id  
 ).all()  
 if posts:  
 for post in posts:  
 db\_sess.delete(post)  
 db\_sess.commit()  
 if user:  
 db\_sess.delete(user)  
 db\_sess.commit()  
 else:  
 abort(404)  
 return logout()  
  
  
@app.route('/upload', methods=['POST'])  
@login\_required  
def upload():  
 pic = request.files['pic']  
  
 if not pic:  
 return "No pfp uploaded", 400  
  
 filename = secure\_filename(pic.filename)  
 mimetype = pic.mimetype  
 img = ProfilePicture(img=pic.read(), mimetype=mimetype, name=filename, user\_id=current\_user.id)  
 db\_sess = db\_session.create\_session()  
 if db\_sess.query(ProfilePicture).filter(ProfilePicture.user\_id == current\_user.id).first():  
 img\_to\_del = db\_sess.query(ProfilePicture).filter(ProfilePicture.user\_id == current\_user.id).first()  
 db\_sess.delete(img\_to\_del)  
 db\_sess.add(img)  
 post = Posts(content="Обновил фотографию на странице",  
 user\_id=current\_user.id)  
 db\_sess.add(post)  
 db\_sess.commit()  
  
 return redirect(f'/user/{current\_user.id}')  
  
  
@app.route('/login', methods=['GET', 'POST'])  
def login():  
 clear\_img()  
 form = LoginForm()  
 if form.validate\_on\_submit():  
 db\_sess = db\_session.create\_session()  
 user = db\_sess.query(User).filter(User.email == form.email.data).first()  
 if user and user.check\_password(form.password.data):  
 login\_user(user, remember=form.remember\_me.data)  
 return redirect("/")  
 return render\_template('login.html',  
 message="Неправильный логин или пароль",  
 form=form)  
 return render\_template('login.html', title='Авторизация', form=form)  
  
  
@app.route('/upload\_pfp')  
@login\_required  
def test():  
 return render\_template("upload\_pfp.html")  
  
  
@app.route('/logout')  
@login\_required  
def logout():  
 logout\_user()  
 clear\_img()  
 return redirect("/")  
  
  
@app.route('/user/<int:id>')  
def profile(id):  
 clear\_img()  
 db\_sess = db\_session.create\_session()  
 img = db\_sess.query(ProfilePicture).filter(ProfilePicture.user\_id == id).first()  
 db\_sess = db\_session.create\_session()  
 post = db\_sess.query(Posts).order\_by(Posts.id.desc()).filter(Posts.user\_id == id).all()  
 user = db\_sess.query(User).filter(User.id == id).first()  
 if img:  
 image = Image.open(io.BytesIO(img.img))  
 image.save("static/img/temp\_img.png")  
 else:  
 image = Image.open("static/img/empty\_img.jpg")  
 image.save("static/img/temp\_img.png")  
 return render\_template("profile.html", user\_id=id, user=user, posts=post,  
 img=url\_for("static", filename="img/temp\_img.png"),  
 alt=url\_for("static", filename="img/empty\_img.jpg"))  
  
  
def serve\_pil\_image(pil\_img):  
 img\_io = io.StringIO()  
 pil\_img.save(img\_io, 'JPEG', quality=70)  
 img\_io.seek(0)  
 return send\_file(img\_io, mimetype='image/jpeg')  
  
  
def clear\_img():  
 if "temp\_img.png" in os.listdir("static/img"):  
 os.remove("static/img/temp\_img.png")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 db\_session.global\_init("db/mirox\_db.db")  
 api.add\_resource(posts\_resources.PostListResource, '/api/post')  
 api.add\_resource(posts\_resources.PostResource, '/api/post/<int:post\_id>')  
 api.add\_resource(users\_resourses.UserListResource, '/api/user')  
 api.add\_resource(users\_resourses.UserResource, '/api/user/<int:user\_id>')  
 app.run()