**СОЗДАНИЕ САЙТА С ОБЪЯВЛЕНИЯМИ НА ЯЗЫКЕ PYTHON**

Автор работы:

учащийся 11-2 класса

Кондратьев Дмитрий

Научный руководитель:

учитель информатики

Мельникова Дарья Юрьевна

САРАТОВ, 2021

**СОДЕРЖАНИЕ**

Введение…………………………………………………………………………..3

Общая информация…… ………………………………………………………...3

Структура сайта……………………………………………………..................…4

Структура базы данных………………………………………………………….4

Код программы…………………………………………………………………...5

Файловая структура……………………………………………………………...5

Заключение……………………………………………………………………….6

Список использованных источников…………………………………………...7

Приложение………………………………………………………………………8

**ВВЕДЕНИЕ**

Интернет-сайты стали неотъемлемой частью человеческой жизни. Сайты с объявлениями повсеместно используются для продажи, покупки товаров или услуг, для поиска работодателя и многих других целей. В своей работе я создал свой упрощенный сайт с объявлениями, изучив структуру и механизмы, обеспечивающие его работу.

**ОБЩАЯ ИНФОРМАЦИЯ**

Сайт представляет собой площадку для размещения объявлений. На сайте доступна регистрация/авторизация пользователей, функции размещения/удаления/изменения и добавление объявлений в избранное, а также просмотр профиля пользователей. Размещение объявлений доступно лишь авторизованным пользователям. Редактировать или удалить можно лишь свое объявление.

Сайт состоит из нескольких страниц. Из главной страницы с объявлениями авторизовавшемуся пользователю можно перейти в свой профиль или на страницу с избранными объявлениями, а также создать свое собственное объявление. С помощью кнопок под объявлением на главном экране его можно добавить в избранное и изменить/удалить, если вы его создатель. В профиле содержится информация, указанная при регистрации. В избранных объявлениях показываются объявления, добавленные в избранное с помощью специальной кнопки. Также есть окна регистрации/авторизации и изменения объявления. Не авторизовавшийся пользователь может лишь просматривать объявления, информацию об их авторах, а также зарегистрироваться или войти в свой аккаунт.

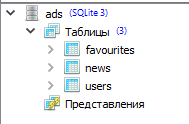
**СТРУКТУРА САЙТА**

Структура сайта включает в себя три основные части:

* Видимая часть, интерфейс сайта
* База данных, хранящая объявления и данные о зарегистрированных пользователях
* Код программы, обеспечивающий работу сайта, взаимодействие с базой данных

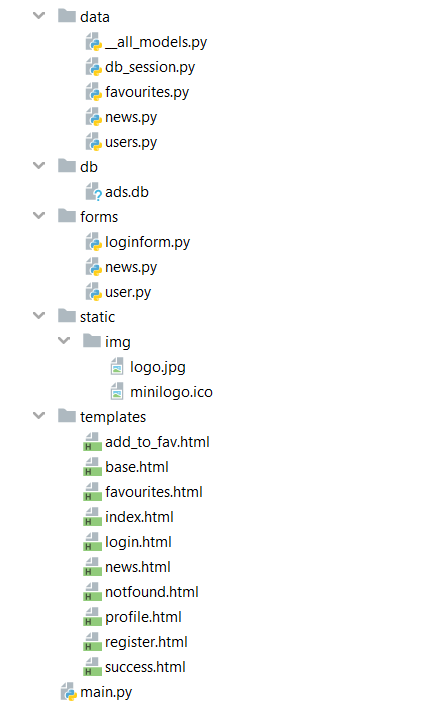
**СТРУКТУРА БАЗЫ ДАННЫХ**

База данных состоит из трех таблиц:

* Таблица пользователей (users). В ней содержится информация о каждом пользователе (уникальный id, имя, электронная почта, дата создания аккаунта, дополнительная информация), а также закодированный пароль
* Таблица объявлений (news). В ней содержится информация о каждом объявлении (уникальный id публикации, текст объявления, дата его публикации) и о его создателе (его уникальный id, показывающий принадлежность объявления)
* Таблица избранных объявлений (favourites). В ней ставятся в соответствие id пользователя и id новости, что позволяет хранить информацию о наличии новостей в избранном у каждого пользователя.

**КОД ПРОГРАММЫ**

Проект реализован на языке Python с использованием библиотек sqlalchemy, flask, позволяющими работать с веб-страницами и базами данных. Для работы с базой данных были созданы классы пользователя, объявления, избранного объявления. Основные возможности сайта обеспечиваются большим количеством различных функций. Например, функции добавления объявления в избранное, создание новости, регистрации и многие другие.

 **ФАЙЛОВАЯ СТРУКТУРА**

Основной код программы содержится в файле main.py. Отдельно в папке data расположены файлы с классами пользователя, объявления и избранного объявления, необходимые для работы с соответствующими таблицами базы данных, а также служебные файлы, обеспечивающие подключение программы к базе данных. В папке db содержится единственный файл базы данных. В папке forms содержатся классы необходимых для создания объявления и регистрации/авторизации пользователя форм. В папке static содержится папка img с маленьким (для отображения во вкладке браузера) и большим, расположенным в левом верхнем углу сайта, логотипами. В папке templates содержатся html файлы с разметкой всех страниц сайта.

**ЗАКЛЮЧЕНИЕ**

В ходе работы был создан сайт объявлений на языке программирования Python. Были изучены принципы работы базы данных, структура страниц веб-сайта, механизмы работы с пользователями. Безусловно, сайт не является готовым продуктом и требует множества доработок, например, добавление функционала (возможность прикрепления к объявлению фото, внутренний мессенджер для взаимодействия пользователей и т. д.). Но, так или иначе, опыт и понимание логики работы программы, полученные в процессе создания сайта, поспособствуют разработке более сложных проектов в будущем.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Для создания проекта использовались следующие источники:

1. лекции и материалы курсов закрытого ресурса Яндекс.Лицей:

https://yandexlyceum.ru/

1. документация библиотеки flask:

https://flask.palletsprojects.com/en/2.0.x/

1. документация библиотеки sqlalchemy:

https://docs.sqlalchemy.org/en/14/

1. документация библиотеки flask WTF:

https://flask-wtf.readthedocs.io/en/1.0.x/

**ПРИЛОЖЕНИЕ**

1. Код основного файла main.py

**from** flask **import** Flask, render\_template, redirect, request, make\_response  
**from** flask **import** session  
**from** data.users **import** User  
**from** data.news **import** News  
**from** data.favourites **import** Favourites  
**from** forms.user **import** RegisterForm  
**from** forms.loginform **import** LoginForm  
**from** forms.news **import** NewsForm  
**from** flask\_login **import** LoginManager, login\_user, login\_required, logout\_user, current\_user  
**from** data **import** db\_session  
**import** datetime  
  
*# ------------------- Инициализация -------------------*app = Flask(\_\_name\_\_)  
app.config[**'SECRET\_KEY'**] = **'yandexlyceum\_secret\_key'**login\_manager = LoginManager()  
login\_manager.init\_app(app)  
  
  
*# ------------------- Главная страница -------------------*@app.route(**"/"**)  
**def** index():  
 db\_sess = db\_session.create\_session()  
 news = db\_sess.query(News).filter()  
 **if** current\_user.is\_authenticated:  
 favs = [int(i.post\_id) **for** i **in** db\_sess.query(Favourites).filter(Favourites.user\_id == current\_user.id)]  
 **else**:  
 favs = []  
 **return** render\_template(**"index.html"**, news=news, title=**'Главная страница'**, favs=favs)  
  
  
*# ------------------- Страница регистрации -------------------*@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,  
 email=form.email.data,  
 about=form.about.data  
 )  
 user.set\_password(form.password.data)  
 db\_sess.add(user)  
 db\_sess.commit()  
 **return** redirect(**'/login'**)  
 **return** render\_template(**'register.html'**, title=**'Регистрация'**, form=form)  
  
  
*# --------- Функция загрузки объекта пользователя из базы данных по id ---------*@login\_manager.user\_loader  
**def** load\_user(user\_id):  
 db\_sess = db\_session.create\_session()  
 **return** db\_sess.query(User).get(user\_id)  
  
  
*# ------------------- Функция авторизации пользователя -------------------*@app.route(**'/login'**, methods=[**'GET'**, **'POST'**])  
**def** login():  
 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)  
  
  
*# ------------------- Функция выхода из аккаунта -------------------*@app.route(**'/logout'**)  
@login\_required  
**def** logout():  
 logout\_user()  
 **return** redirect(**"/"**)  
  
  
*# ------------------- Функция создания нового объявления -------------------*@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  
 current\_user.news.append(news)  
 db\_sess.merge(current\_user)  
 db\_sess.commit()  
 **return** redirect(**'/'**)  
 **return** render\_template(**'news.html'**, title=**'Новое объявление'**,  
 form=form)  
  
  
*# ------------------- Функция редактирования объявления -------------------*@app.route(**'/news/<int:id>'**, methods=[**'GET'**, **'POST'**])  
@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  
 ).first()  
 **if** news:  
 form.title.data = news.title  
 form.content.data = news.content  
 **else**:  
 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  
 db\_sess.commit()  
 **return** redirect(**'/'**)  
 **else**:  
 abort(404)  
 **return** render\_template(**'news.html'**,  
 title=**'Редактирование объявления'**,  
 form=form  
 )  
  
  
*# ------------------- Функция удаления объявления -------------------*@app.route(**'/news\_delete/<int:id>'**, methods=[**'GET'**, **'POST'**])  
@login\_required  
**def** news\_delete(id):  
 db\_sess = db\_session.create\_session()  
 news = db\_sess.query(News).filter(News.id == id,  
 News.user == current\_user  
 ).first()  
 **if** news:  
 db\_sess.delete(news)  
 db\_sess.commit()  
 **else**:  
 abort(404)  
 **return** redirect(**'/'**)  
  
  
*# -------- Функция загрузки страницы избранных объявлений пользователя ---------*@app.route(**'/favourites'**)  
**def** favourite\_posts():  
 db\_sess = db\_session.create\_session()  
 **if** current\_user.is\_authenticated:  
 favourites = db\_sess.query(Favourites).filter(  
 (Favourites.user\_id == current\_user.get\_user\_id()))  
 s = list(set([i.post\_id **for** i **in** favourites]))  
 res = []  
 **for** i **in** s:  
 res.append(db\_sess.query(News).filter(News.id == i).first())  
 **else**:  
 **return** redirect(**'/'**)  
 **return** render\_template(**"favourites.html"**, news=res, title=**'Избранное'**)  
  
  
*# ------------------- Функция добавления объявления в избранное ----------------*@app.route(**'/add\_favourite\_post/<int:id>'**, methods=[**'GET'**, **'POST'**])  
@login\_required  
**def** fav\_add(id):  
 f = Favourites()  
 f.user\_id = current\_user.get\_user\_id()  
 f.post\_id = id  
 db\_sess = db\_session.create\_session()  
 db\_sess.add(f)  
 db\_sess.commit()  
 **return** redirect(**'/'**)  
  
  
*# ------------------- Функция удаления объявления из избранного ----------------*@app.route(**'/delete\_favourite\_post/<int:id>'**, methods=[**'GET'**, **'POST'**])  
@login\_required  
**def** delete\_favourite\_post(id):  
 db\_sess = db\_session.create\_session()  
 news = db\_sess.query(Favourites).filter(Favourites.post\_id == id,  
 Favourites.user\_id == current\_user.id)  
 **if** news:  
 **for** i **in** news:  
 db\_sess.delete(i)  
 db\_sess.commit()  
 **else**:  
 abort(404)  
 **return** redirect(**'/favourites'**)  
  
  
*# ------------------- Функция загрузки страницы профиля пользователя -----------*@app.route(**'/profile/<int:id>'**, methods=[**'GET'**, **'POST'**])  
**def** profile(id):  
 db\_sess = db\_session.create\_session()  
 user = db\_sess.query(User).filter(User.id == id).first()  
 **if** user:  
 info = user.get\_info()  
 info[-1] = str(info[-1]).split(**'.'**)[0]  
 name, email, inf, date = info  
 **else**:  
 **return** redirect(**'/not\_found'**)  
 **return** render\_template(**"profile.html"**, name=name, email=email, info=inf, date=date,  
 title=**'Профиль'**)  
  
  
*# ------------------- Функция страницы Not found -------------------*@app.route(**'/not\_found'**)  
**def** not\_found():  
 **return** render\_template(**"notfound.html"**)  
  
  
*# ------------------- Функция проверки наличия поста в избранном ---------------***def** if\_post\_in\_favourites(user\_id, post\_id):  
 db\_sess = db\_session.create\_session()  
 news = db\_sess.query(Favourites).filter(Favourites.post\_id == post\_id,  
 Favourites.user\_id == user\_id)  
 **if** news:  
 **return True  
 else**:  
 **return False***# ------------------- Запуск кода, функция main -------------------***def** main():  
 db\_session.global\_init(**"db/ads.db"**)  
 app.run()  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

1. Форма для создания объявления news.py

**from** flask\_wtf **import** FlaskForm  
**from** wtforms **import** StringField, TextAreaField  
**from** wtforms **import** BooleanField, SubmitField  
**from** wtforms.validators **import** DataRequired  
  
  
**class** NewsForm(FlaskForm):  
 title = StringField(**'Заголовок'**, validators=[DataRequired()])  
 content = TextAreaField(**"Содержание"**)  
 submit = SubmitField(**'Применить'**)

1. Форма для регистрации user.py

**from** flask\_wtf **import** FlaskForm  
**from** wtforms **import** PasswordField, StringField, TextAreaField, SubmitField  
**from** wtforms.fields.html5 **import** EmailField  
**from** wtforms.validators **import** DataRequired  
  
  
**class** RegisterForm(FlaskForm):  
 email = EmailField(**'Почта'**, validators=[DataRequired()])  
 password = PasswordField(**'Пароль'**, validators=[DataRequired()])  
 password\_again = PasswordField(**'Повторите пароль'**, validators=[DataRequired()])  
 name = StringField(**'Имя пользователя'**, validators=[DataRequired()])  
 about = TextAreaField(**"Немного о себе"**)  
 submit = SubmitField(**'Войти'**)

1. Форма для авторизации login\_form.py

**from** flask\_wtf **import** FlaskForm  
**from** wtforms **import** PasswordField, StringField, TextAreaField, SubmitField, BooleanField  
**from** wtforms.fields.html5 **import** EmailField  
**from** wtforms.validators **import** DataRequired  
  
**class** LoginForm(FlaskForm):  
 email = EmailField(**'Почта'**, validators=[DataRequired()])  
 password = PasswordField(**'Пароль'**, validators=[DataRequired()])  
 remember\_me = BooleanField(**'Запомнить меня'**)  
 submit = SubmitField(**'Войти'**)

1. Класс пользователя user.py

**import** datetime  
**import** sqlalchemy  
**from** .db\_session **import** SqlAlchemyBase  
**from** sqlalchemy **import** orm  
**from** werkzeug.security **import** generate\_password\_hash, check\_password\_hash  
**from** flask\_login **import** UserMixin  
  
**class** User(SqlAlchemyBase, UserMixin):  
 \_\_tablename\_\_ = **'users'** id = sqlalchemy.Column(sqlalchemy.Integer,  
 primary\_key=**True**, autoincrement=**True**)  
 name = sqlalchemy.Column(sqlalchemy.String, nullable=**True**)  
 about = sqlalchemy.Column(sqlalchemy.String, nullable=**True**)  
 email = sqlalchemy.Column(sqlalchemy.String,  
 index=**True**, unique=**True**, nullable=**True**)  
 hashed\_password = sqlalchemy.Column(sqlalchemy.String, nullable=**True**)  
 created\_date = sqlalchemy.Column(sqlalchemy.DateTime,  
 default=datetime.datetime.now)  
 news = orm.relation(**"News"**, back\_populates=**'user'**)  
  
 **def** \_\_repr\_\_(self):  
 **return f"<User> {**self.name**} {**self.email**} {**self.about**} {**self.created\_date**}"  
  
 def** get\_info(self):  
 **return** [self.name, self.email, self.about, self.created\_date]  
  
 **def** get\_user\_id(self):  
 **return** self.id  
  
 **def** set\_password(self, password):  
 self.hashed\_password = generate\_password\_hash(password)  
  
 **def** check\_password(self, password):  
 **return** check\_password\_hash(self.hashed\_password, password)

1. Класс объявления news.py

**import** datetime  
**import** sqlalchemy  
**from** sqlalchemy **import** orm  
  
**from** .db\_session **import** SqlAlchemyBase  
  
categories = orm.relation(**"Category"**,  
 secondary=**"association"**,  
 backref=**"news"**)  
  
  
**class** News(SqlAlchemyBase):  
 \_\_tablename\_\_ = **'news'** id = sqlalchemy.Column(sqlalchemy.Integer,  
 primary\_key=**True**, autoincrement=**True**)  
 title = sqlalchemy.Column(sqlalchemy.String, nullable=**True**)  
 content = sqlalchemy.Column(sqlalchemy.String, nullable=**True**)  
 created\_date = sqlalchemy.Column(sqlalchemy.DateTime,  
 default=datetime.datetime.now())  
  
 user\_id = sqlalchemy.Column(sqlalchemy.Integer,  
 sqlalchemy.ForeignKey(**"users.id"**))  
 user = orm.relation(**'User'**)  
  
 **def** get\_news\_id(self):  
 **return** self.id  
  
 **def** \_\_repr\_\_(self):  
 **return f'<News> {**self.user**} {**self.id**} {**self.title**} {**self.content**}'**

1. Класс избранного объявления favourites.py

**import** sqlalchemy  
**from** sqlalchemy **import** orm  
  
**from** .db\_session **import** SqlAlchemyBase  
  
  
**class** Favourites(SqlAlchemyBase):  
 \_\_tablename\_\_ = **'favourites'** id = sqlalchemy.Column(sqlalchemy.Integer,  
 primary\_key=**True**, autoincrement=**True**)  
  
 user\_id = sqlalchemy.Column(sqlalchemy.Integer,  
 sqlalchemy.ForeignKey(**"users.id"**))  
 post\_id = sqlalchemy.Column(sqlalchemy.Integer,  
 sqlalchemy.ForeignKey(**"news.id"**))  
  
 **def** \_\_repr\_\_(self):  
 **return** str(self.post\_id)

1. Подключение к базе данных (db\_session.py)

**import** sqlalchemy **as** sa  
**import** sqlalchemy.orm **as** orm  
**from** sqlalchemy.orm **import** Session  
**import** sqlalchemy.ext.declarative **as** dec  
*# noinspection PyUnresolvedReferences  
# import \_\_all\_models*SqlAlchemyBase = dec.declarative\_base()  
  
\_\_factory = **None  
  
def** global\_init(db\_file):  
 **global** \_\_factory  
  
 **if** \_\_factory:  
 **return  
  
 if not** db\_file **or not** db\_file.strip():  
 **raise** Exception(**"Необходимо указать файл базы данных."**)  
  
 conn\_str = **f'sqlite:///{**db\_file.strip()**}?check\_same\_thread=False'** print(**f"Подключение к базе данных по адресу {**conn\_str**}"**)  
  
 engine = sa.create\_engine(conn\_str, echo=**False**)  
 \_\_factory = orm.sessionmaker(bind=engine)  
  
 **from** . **import** \_\_all\_models  
  
 SqlAlchemyBase.metadata.create\_all(engine)  
  
**def** create\_session() -> Session:  
 **global** \_\_factory  
 **return** \_\_factory()