Работа с базами данных (авторизация пользователей)

1. Изучить книгу М.Гринберга Мега-учебник Flask ч.4, ч.5
https://habr.com/ru/post/346344/
https://habr.com/ru/post/346346/

2. Для работы с базами данных мы будем использовать пакет Flask-SQLAlchemy, «который является Object Relational Mapper или ORM. ORM позволяют приложениям управлять базой данных с использованием объектов высокого уровня, таких как классы, объекты и методы, а не таблицы и SQL. Задача ORM — перевести операции высокого уровня в команды базы данных.». Эту библиотеку необходимо установить на своем компьютере.
Также необходимо установить пакет Flask-Login. «Это расширение управляет состоянием входа пользователя в систему, так что, например, пользователи могут войти в приложение, а затем перейти на разные страницы, пока приложение «помнит», что пользователь вошел в систему. Оно также предоставляет функциональность «запомнить меня», которая позволяет пользователям оставаться в системе даже после закрытия окна браузера.»

3. Для работы с пользователем создается объект LoginManager:

```
from flask_login import LoginManager
app = Flask(__name__)
# ...
login = LoginManager(app)
```

Для загрузки данных о пользователе в приложение:

```
@login.user_loader
def  load_user(id):
    return User.query.get(int(id))
```

Для выхода из приложения пользователя создать функцию:

```
from flask_login import logout_user
# ...
@app.route('/logout')
def logout():
    logout_user()
    return redirect('index')
```

4. Для каждой таблицы БД создается модель. Например, для таблицы пользователи модель будет:

```
from flask_login import UserMixin
class User(UserMixin, db.Model):
        id = db.Column(db.Integer, primary_key=True)
        username = db.Column(db.String(64), index=True, unique=True)
        email = db.Column(db.String(120), index=True, unique=True)
        password_hash = db.Column(db.String(128))
```

5. Хеширование паролей (в класс пользователь добавляются две функции: генерации хеша и наоборот)

```
from werkzeug.security import generate_password_hash, check_password_hash
 # ...
class User(UserMixin, db.Model):
# ...
    def set_password(self, password):
        self.password_hash = generate_password_hash(password)
    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
```

6. Внесем изменения в обработчик форм авторизации и регистрации, отправив соответствующие запросы

```
from flask_login import current_user, login_user

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password')
            return redirect('/login')
     login_user(user, remember=form.remember_me.data)
     return redirect('/index')
    return render_template('login.html', title='Sign In', form=form)
```

Строка user = User.query.filter_by(username=form.username.data).first() - это запрос к таблице БД на выборку пользователя, у которого значение в поле username совпадает со значением, введенным в соответствующее поле формы.

7. По аналогии можно завершить регистрацию, добавив в обработчик формы регистрации, созданной на предыдущем уроке создание объекта класса User и добавление его в БД:

```
u = User()
user.username = form.username.data
user.email = form.email.data
user.set_password(form.password.data)
db.session.add(user)
db.session.commit()
```

8. Через формы, созданные на предыдущем уроке необходимо провести регистрацию/авторизацию пользователя.