Blogs

...

INTRODUCCIÓN:

A menudo los estudiantes quieren mostrar imágenes o pdfs y compartir ese tipo de cosas pero en la comodidad de no recibir ningún comentario acerca de lo subido simplemente suben sus archivos para compartir lo que han aprendido como Pdfs o fotos.El problema radica en donde subirlo sin ningun comentario sea constructivo u ofensivo,por eso hice una interfaz de blogs en el cual subes tus archivos sean .png , .jpg o pdf y puedes buscar el de otros usarios tambien.

PARTES IMPORTANTES DEL CÓDIGO:

En esta parte del código estoy creando la base de datos y mi esquema va a contener la tabla que en este caso los declaró como clase ya que almaceno todos los datos y de ahi los voy a utilizar.

```
25  class Base(db.Model):
26      __abstract__ = True
27
28      id = db.Column(db.Integer, primary_key=True)
29      date_created = db.Column(db.DateTime, default=db.func.current_timestamp())
30      date_modified = db.Column(db.DateTime, default=db.func.current_timestamp(),onupdate=db.func.current_timestamp())
31
32  class Users(Base):
33      username = db.Column(db.String(50), unique=True, nullable=False)
34      email = db.Column(db.String(50), unique=True, nullable=False)
35      password = db.Column(db.String(80), nullable=False)
36      about_me = db.Column(db.String(280), default="")
37      images = db.relationship("Image", backref="owner", lazy="dynamic")
38
39  class Image(Base):
40      filename = db.Column(db.String(80), nullable=False)
41      user_id = db.Column(db.Integer, db.ForeignKey("users.id"), nullable=False)
42      owner_username = db.Column(db.String(50), nullable=False)
43
```

en la función signup si recibes un post jala los datos del form que te enviaron y si no están uso te puedes registrar exitosamente.

```python
@app.route("/signup", methods=["GET", "POST"])
def signup():
    if not g.user:
        if request.method == "POST":
            username = Users.query.filter_by(username=request.form["username"].lower()).first()
            email = Users.query.filter_by(email=request.form["email"].lower()).first()

            if username or email:
                flash("Ese nombre o email ya esta en uso", "alert-warning")

                return redirect(request.url)
            hashed_pw = generate_password_hash(request.form["password"], method="sha256")
            new_user = Users(username=request.form["username"].lower(),email=request.form["email"].lower(), password=hashed
            db.session.add(new_user)
            db.session.commit()

            flash("Te has registrado exitosamente", "alert-success")

            return redirect(url_for("login"))

        return render_template("signup.html")
    flash("Ya estas registrado", "alert-primary")
```

Esta parte del código recibe un POST de parte de signup.html con un form con los datos username,email y password. Entonces si recibe este va a preguntar si los datos son incorrectos y te mandara un mensaje emergente en caso sea cierto,sino te registrara los datos y te cifrara la contraseña con el método sha256 en caso de que este logueado te mandar a home.

```
87  @app.route("/login", methods=["GET", "POST"])
88  def login():
89      if not g.user:
90          if request.method == "POST":
91              user = Users.query.filter_by(username=request.form["username"].lower()).first
92              if user and check_password_hash(user.password, request.form["password"]):
93                  session["username"] = user.username
94                  flash("Ya se ha autentificado", "alert-success")
95                  return redirect("home")
96              flash("Tus credenciales son invalidas", "alert-warning")
97
98          return render_template("login.html")
99      flash("Necesitas loggearte.", "alert-primary")
100
101     return redirect(url_for("home"))
```

En la primera imagen hacemos uso de otro tipo de request que verifica antes de hayan enviado una solicitud y verificas si la cookie de username esta.

```python
@app.before_request
def before_request():
    if "username" in session:
        g.user = session["username"]
    else:
        g.user = None
```

La siguiente función es por si buscas una ruta que no encuentra en la app y para controlar el error 404.

```python
@app.errorhandler(404)
def page_not_found(error):

    return render_template("404_page_not_found.html"), 404
```

En esta función de igual manera verifica si esta logueado y luego verifica si no hay archivo o si, y lo agrega a la base de datos.

```python
@app.route("/home", methods=["GET", "POST"])
def home():
    if g.user:
        if request.method == "POST":
            if "file" not in request.files:
                flash("ninguna parte del archivo", "alert-danger")
                return redirect(request.url)
            file = request.files["file"]
            if file.filename == "":
                flash("No hay archivo", "alert-warning")
                return redirect(request.url)
            if file and allowed_file(file.filename):
                filename = secure_filename(file.filename)
                user = Users.query.filter_by(username=g.user).first()
                file_to_db = Image(filename=filename, owner=user, \
                                   owner_username=user.username)
                db.session.add(file_to_db)
                db.session.commit()
                file.save(os.path.join(app.config["UPLOAD_FOLDER"], filename))
                return redirect(url_for("get_files"))
        return render_template("home.html")
    flash("Debes registrarte", "alert-warning")

    return redirect(url_for("login"))
```

En estas dos funciones paso la imagen del perfil y en la otra cambio la descripción.

```python
@app.route("/profile/<username>")
def profile(username):
    user = Users.query.filter_by(username=username).first()

    if user:
        files = user.images.order_by(Image.date_modified).limit(10).all()
        picture = get_profile_picture(user.email)

        return render_template("profile.html", user=user, files=files, picture=pictu

    abort(404)

@app.route("/profile/edit", methods=["GET", "POST"])
def edit_profile():
    user = Users.query.filter_by(username=g.user).first()

    if request.method == "POST":
        user.about_me = request.form["about"]
        db.session.commit()

        flash("Los cambios han sido guardados exitosamente", "alert-success")
        return redirect(url_for("profile", username=g.user))

    return render_template("edit_profile.html", user=user)
```

en estas dos funciones es donde saco los archivos de el atributo date_modified y en la segunda función es para devolver todos tus archivos.

```python
@app.route("/files")
def get_all_files():
    files = Image.query.order_by(Image.date_modified).limit(200).all()

    return render_template("files.html", files=files)


@app.route("/myfiles")
def get_files():
    if g.user:
        user = Users.query.filter_by(username=g.user).first()

        if user.images.all():
            files = user.images.order_by(Image.date_modified).all()
            return render_template("my_files.html", files=files)

        files = []
        return render_template("my_files.html", files=files)

    flash("Debes registrarte", "alert-warning")

    return redirect(url_for("login"))
```