Team9

정글 마니또

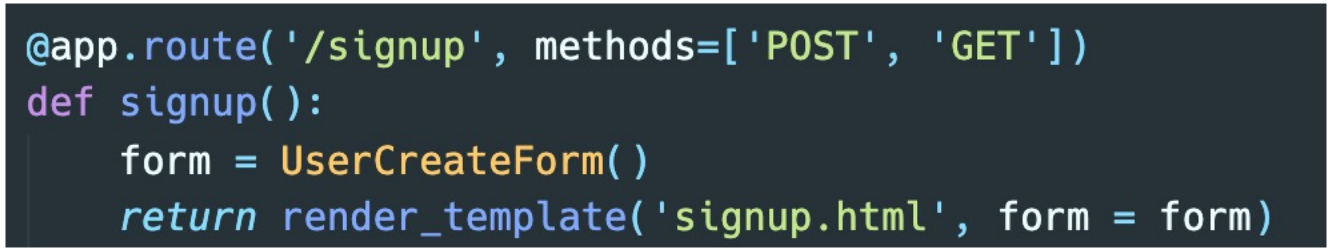
김승환, 이후명, 조헌일

목차

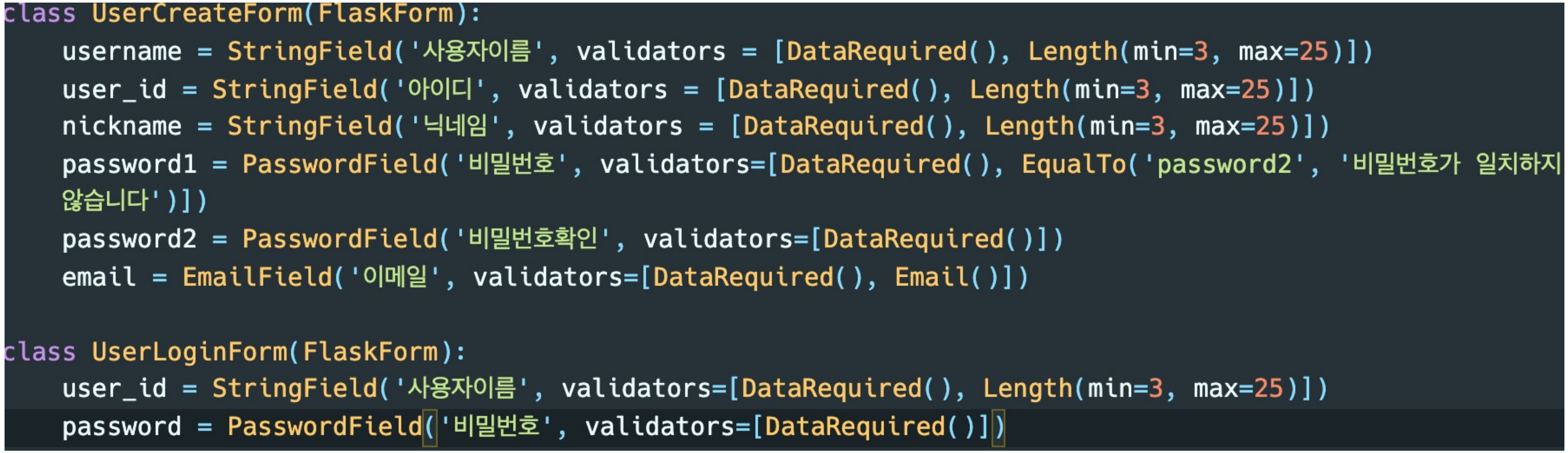
회원가입

- Flask form
- Flask_wtf
- 비밀번호 암호화

Flask의 form

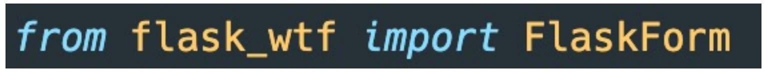


```python
@app.route('/signup', methods=['POST', 'GET'])
def signup():
    form = UserCreateForm()
    return render_template('signup.html', form = form)
```

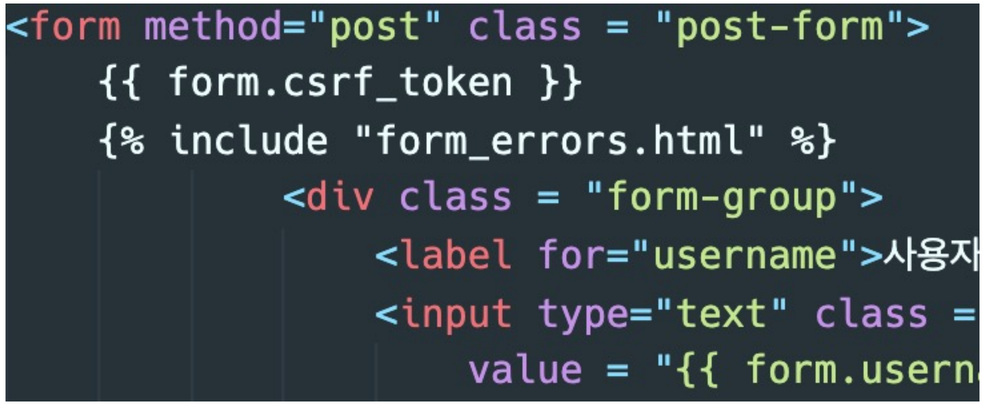


```python
class UserCreateForm(FlaskForm):
    username = StringField('사용자이름', validators = [DataRequired(), Length(min=3, max=25)])
    user_id = StringField('아이디', validators = [DataRequired(), Length(min=3, max=25)])
    nickname = StringField('닉네임', validators = [DataRequired(), Length(min=3, max=25)])
    password1 = PasswordField('비밀번호', validators=[DataRequired(), EqualTo('password2', '비밀번호가 일치하지
    않습니다')])
    password2 = PasswordField('비밀번호확인', validators=[DataRequired()])
    email = EmailField('이메일', validators=[DataRequired(), Email()])

class UserLoginForm(FlaskForm):
    user_id = StringField('사용자이름', validators=[DataRequired(), Length(min=3, max=25)])
    password = PasswordField('비밀번호', validators=[DataRequired()])
```

Flask_wtf

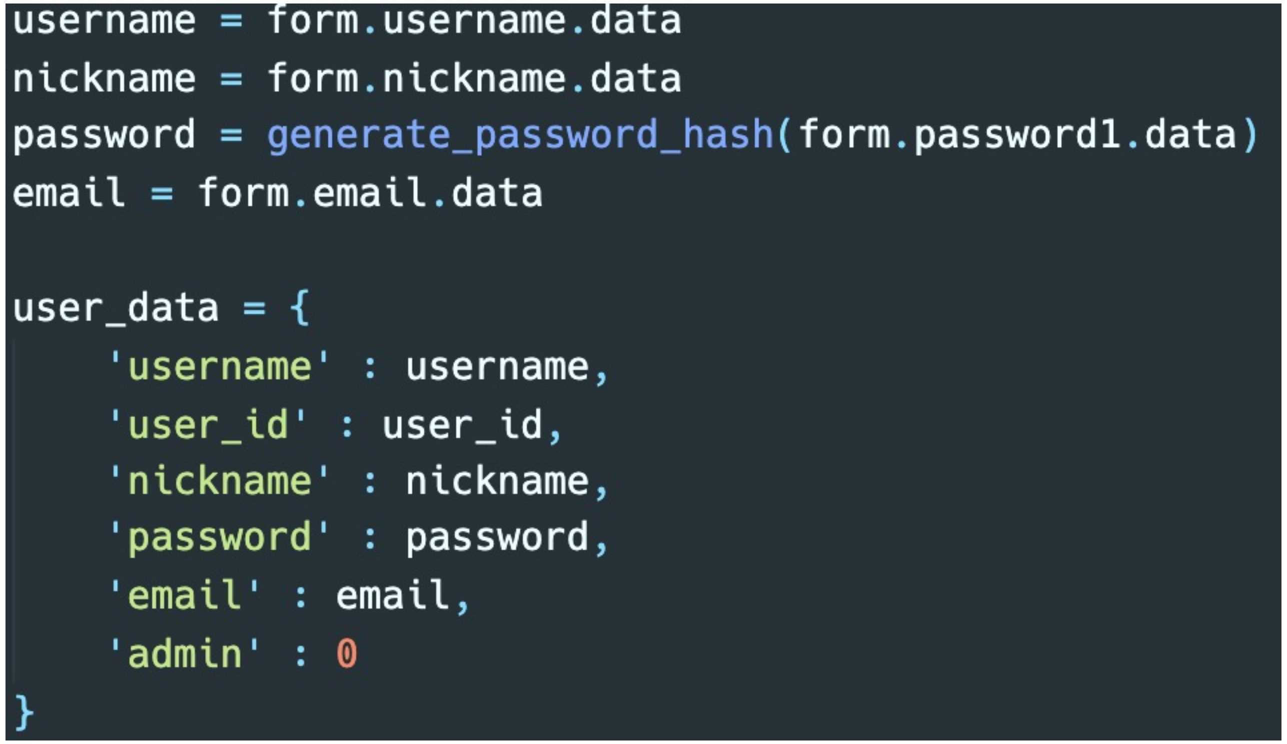


```
from flask_wtf import FlaskForm
```

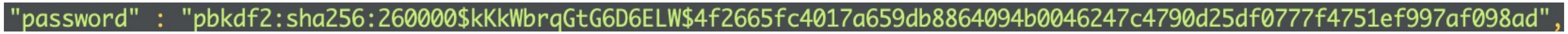


```
<form method="post" class = "post-form">
    {{ form.csrf_token }}
    {% include "form_errors.html" %}
            <div class = "form-group">
                <label for="username">사용지
                <input type="text" class =
                    value = "{{ form.userni
```

비밀번호 암호화



```python
username = form.username.data
nickname = form.nickname.data
password = generate_password_hash(form.password1.data)
email = form.email.data

user_data = {
    'username' : username,
    'user_id' : user_id,
    'nickname' : nickname,
    'password' : password,
    'email' : email,
    'admin' : 0
}
```

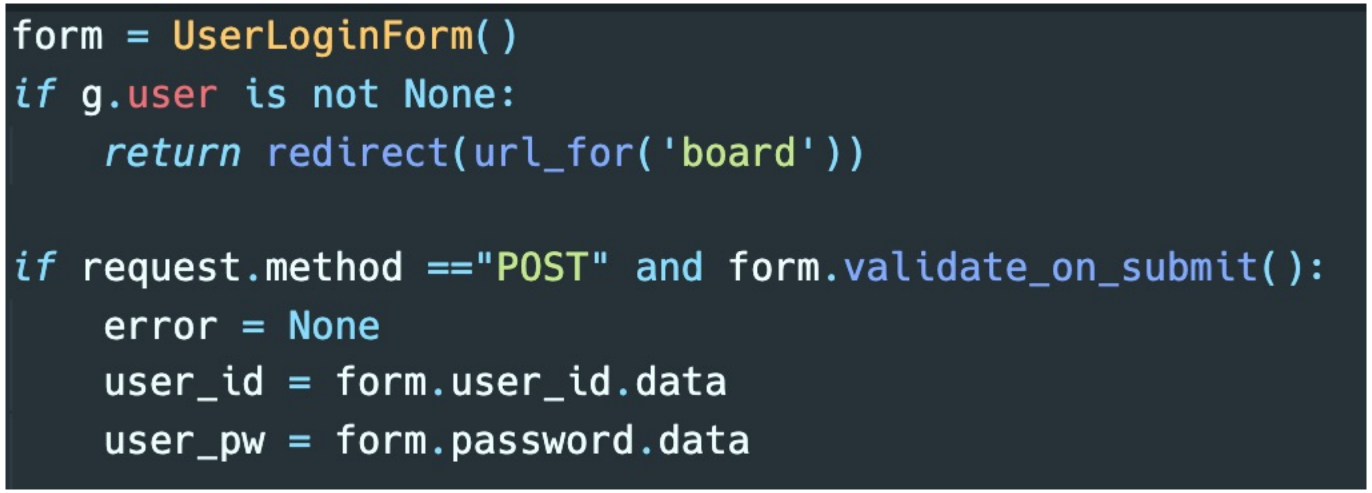


```
"password" : "pbkdf2:sha256:260000$kKkWbrqGtG6D6ELW$4f2665fc4017a659db8864094b0046247c4790d25df0777f4751ef997af098ad",
```

로그인

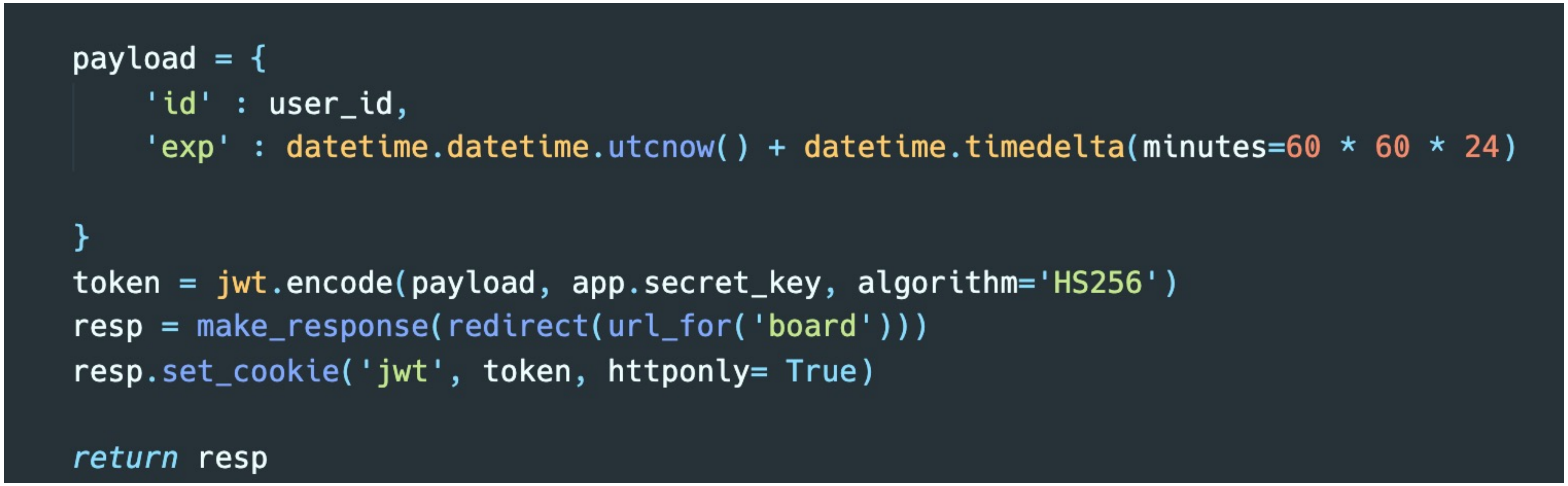
- FlaskForm
- JWT

Flask의 form



```
form = UserLoginForm()
if g.user is not None:
    return redirect(url_for('board'))

if request.method =="POST" and form.validate_on_submit():
    error = None
    user_id = form.user_id.data
    user_pw = form.password.data
```

JWT

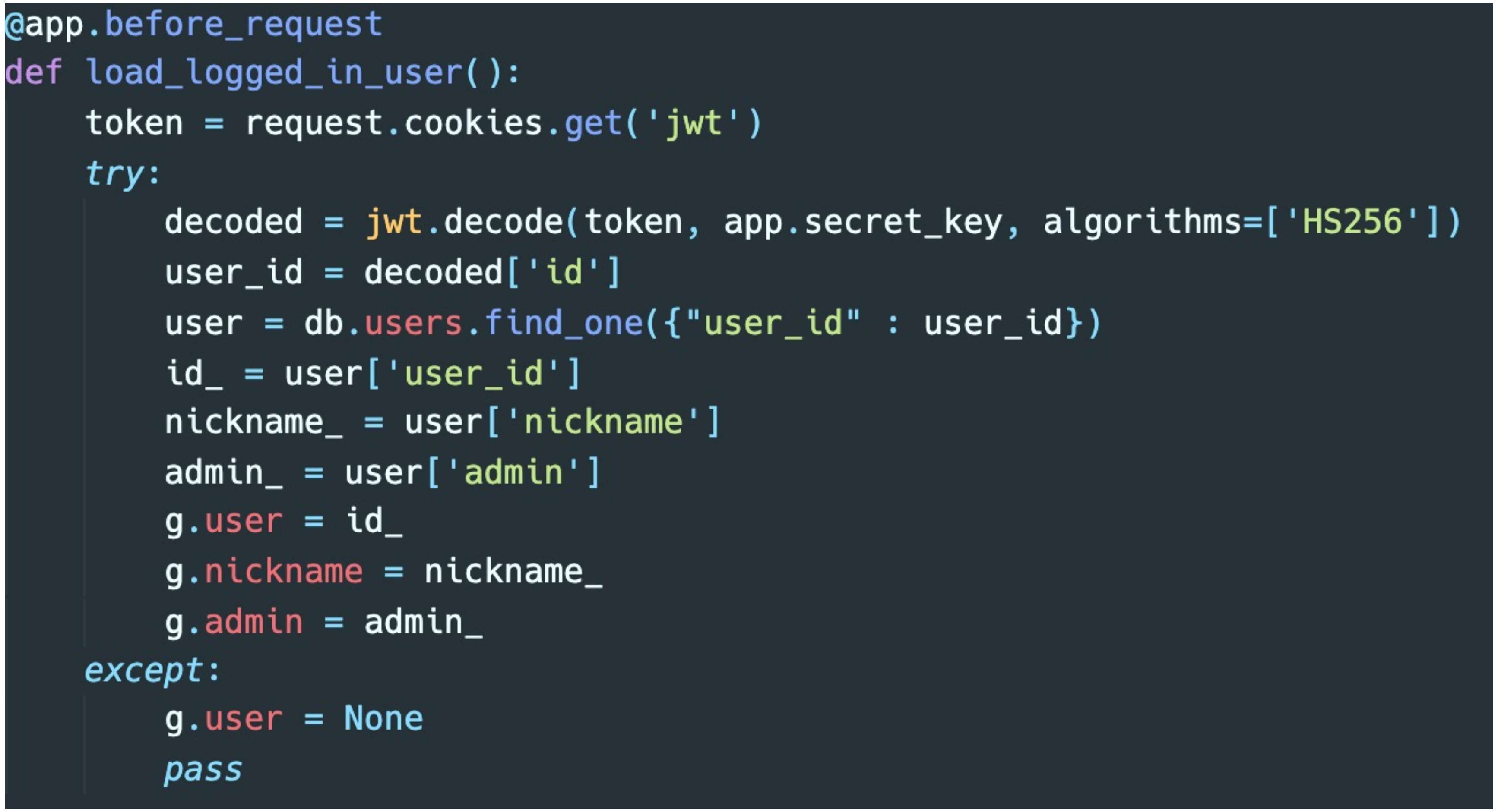


```python
payload = {
    'id' : user_id,
    'exp' : datetime.datetime.utcnow() + datetime.timedelta(minutes=60 * 60 * 24)

}
token = jwt.encode(payload, app.secret_key, algorithm='HS256')
resp = make_response(redirect(url_for('board')))
resp.set_cookie('jwt', token, httponly= True)

return resp
```

페이지 구현

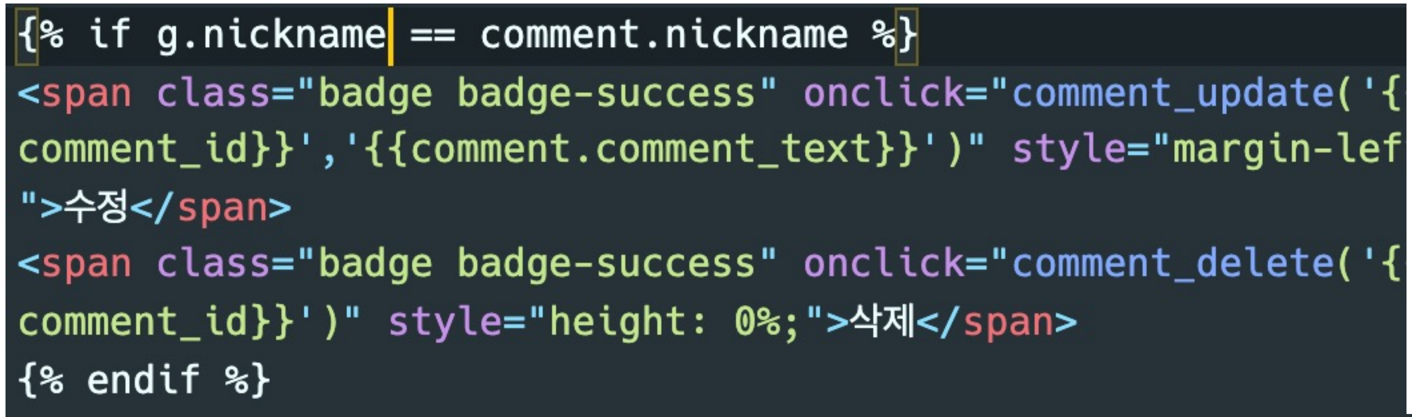
- Flask g
- jinja2
- Embeded data

Flask의 g

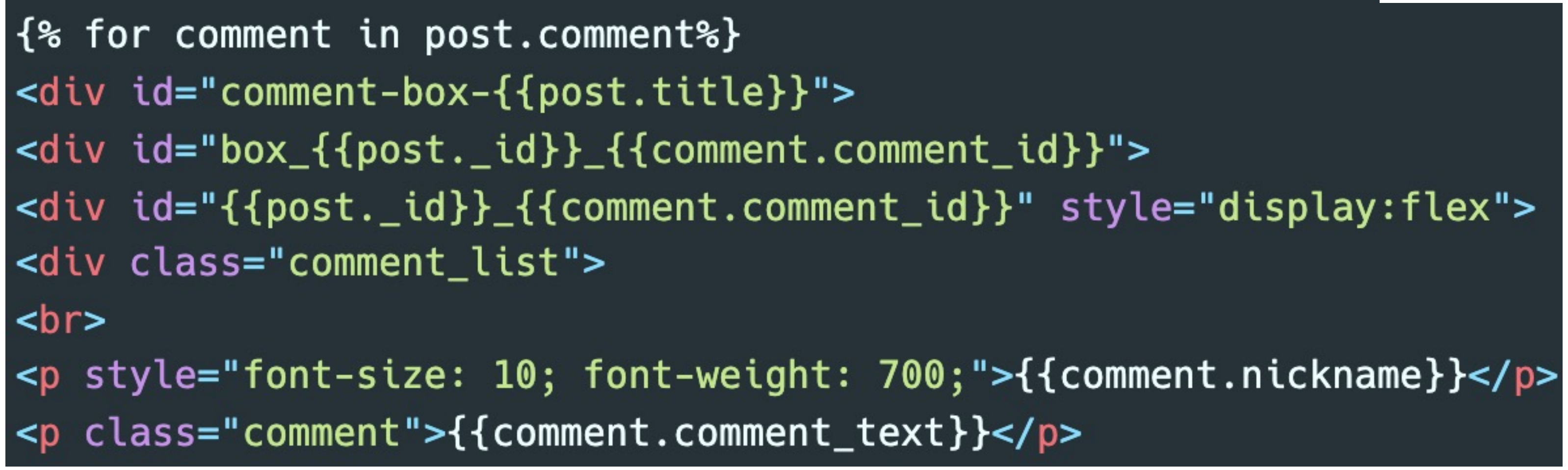


```python
@app.before_request
def load_logged_in_user():
    token = request.cookies.get('jwt')
    try:
        decoded = jwt.decode(token, app.secret_key, algorithms=['HS256'])
        user_id = decoded['id']
        user = db.users.find_one({"user_id" : user_id})
        id_ = user['user_id']
        nickname_ = user['nickname']
        admin_ = user['admin']
        g.user = id_
        g.nickname = nickname_
        g.admin = admin_
    except:
        g.user = None
        pass
```

Jinja2

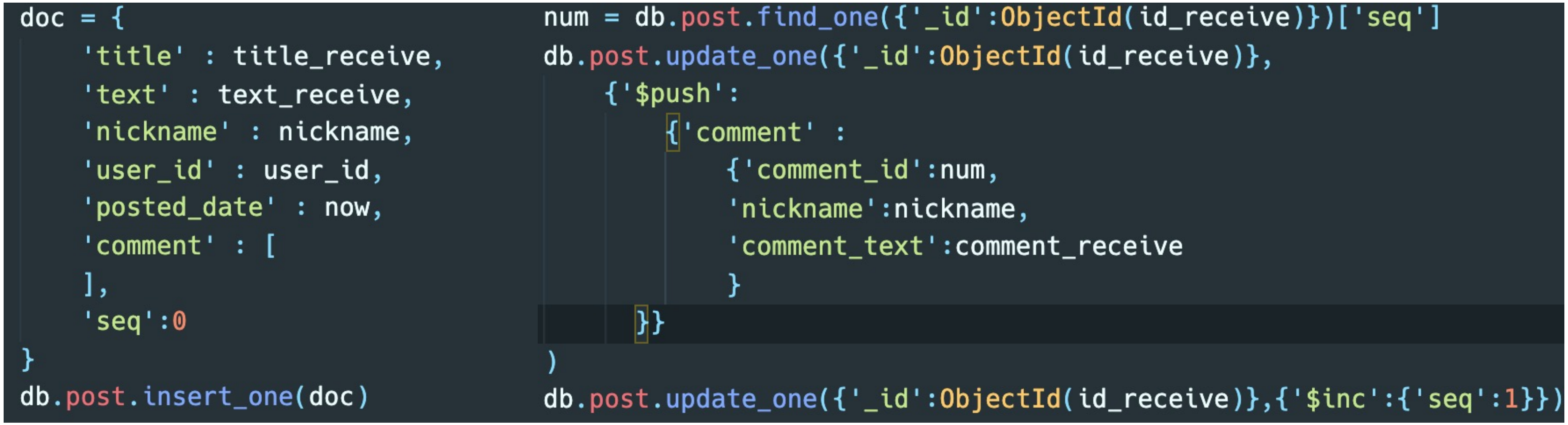


```
{% if g.nickname == comment.nickname %}
<span class="badge badge-success" onclick="comment_update('{
comment_id}}','{{comment.comment_text}}')" style="margin-lef
">수정</span>
<span class="badge badge-success" onclick="comment_delete('{
comment_id}}')" style="height: 0%;">삭제</span>
{% endif %}
```



```
{% for comment in post.comment%}
<div id="comment-box-{{post.title}}">
<div id="box_{{post._id}}_{{comment.comment_id}}">
<div id="{{post._id}}_{{comment.comment_id}}" style="display:flex">
<div class="comment_list">
<br>
<p style="font-size: 10; font-weight: 700;">{{comment.nickname}}</p>
<p class="comment">{{comment.comment_text}}</p>
```

Embeded data



```
doc = {
    'title' : title_receive,
    'text' : text_receive,
    'nickname' : nickname,
    'user_id' : user_id,
    'posted_date' : now,
    'comment' : [
    ],
    'seq':0
}
db.post.insert_one(doc)
```

```
num = db.post.find_one({'_id':ObjectId(id_receive)})['seq']
db.post.update_one({'_id':ObjectId(id_receive)},
    {'$push':
        {'comment' :
            {'comment_id':num,
            'nickname':nickname,
            'comment_text':comment_receive
            }
    }}
)
db.post.update_one({'_id':ObjectId(id_receive)},{'$inc':{'seq':1}})
```

마치며...

- Git을 활용하지 못했다.
- 기획 단계에서 변수명, url 명을 통일하지 않아서 후에 많은 문제가 발생했다.
- DB에 데이터를 좀 더 좋은 방식으로 저장할 수 있는 방법이 있을 것 같다.
- 세부 기능을 더 구현할 수 있었을 것 같다(ex.관리자가 모든 글 삭제 가능).

감사합니다.