



Simple Memo

DRF를 이용한 SPA방식의 웹 서비스

서비스 소개

주요 기능

1. 구글 계정으로 회원가입, 로그인
2. Rest api 를 이용한 간단한 메모 CRUD
3. URL 변경 없이 SPA으로 화면 UI 변경

제작 동기

첫 프로젝트인 BTC village 프로젝트를 성공적으로 배포까지 마친 후, 취업 전까지 뭘 더 공부하면 좋을까 고민하다가 DRF를 사용하여 restful api를 설계해보고 싶었다.

restful하게 설계하기 좋은 프로젝트가 뭐가 있을까 하는 고민 중, 아이폰의 메모 앱을 최대한 비슷하게 구현해보기로 결정했다.

개발 환경

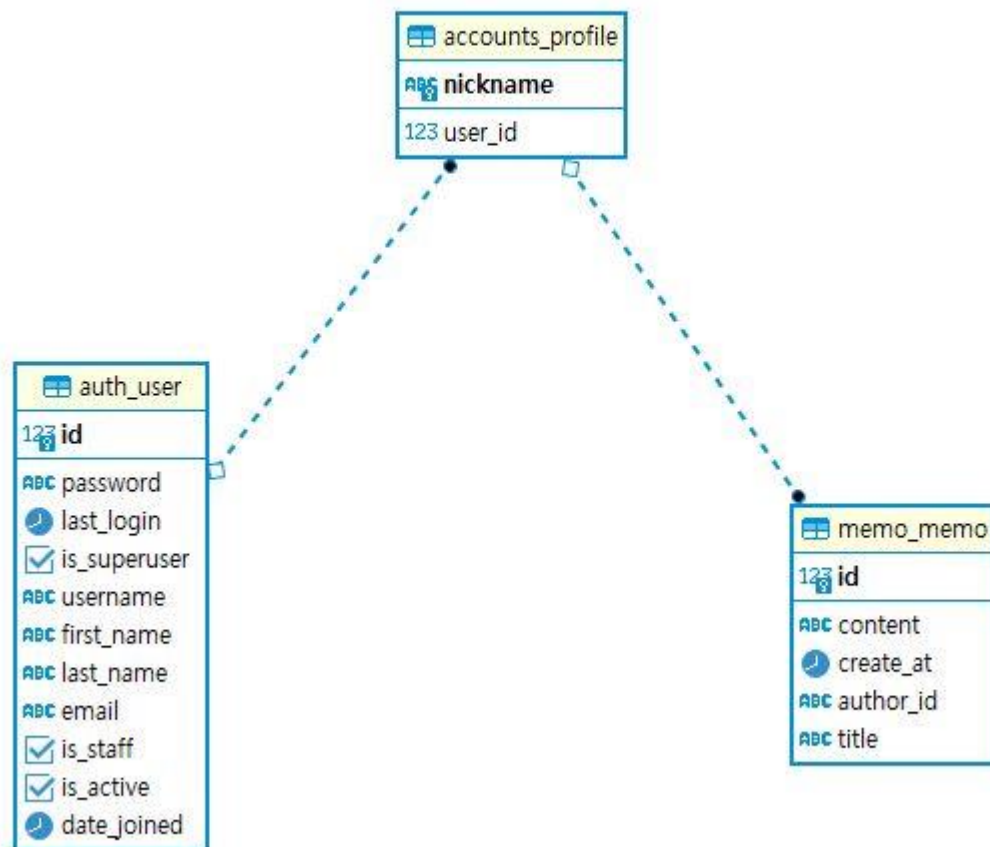
Front-end

- HTML
- Css
- Javascript
- JQuery
- Bootstarp

Back-end

- Python
- Django
- DRF
- PostgreSQL
- Github

데이터 구조 (ERD)



새 글 작성

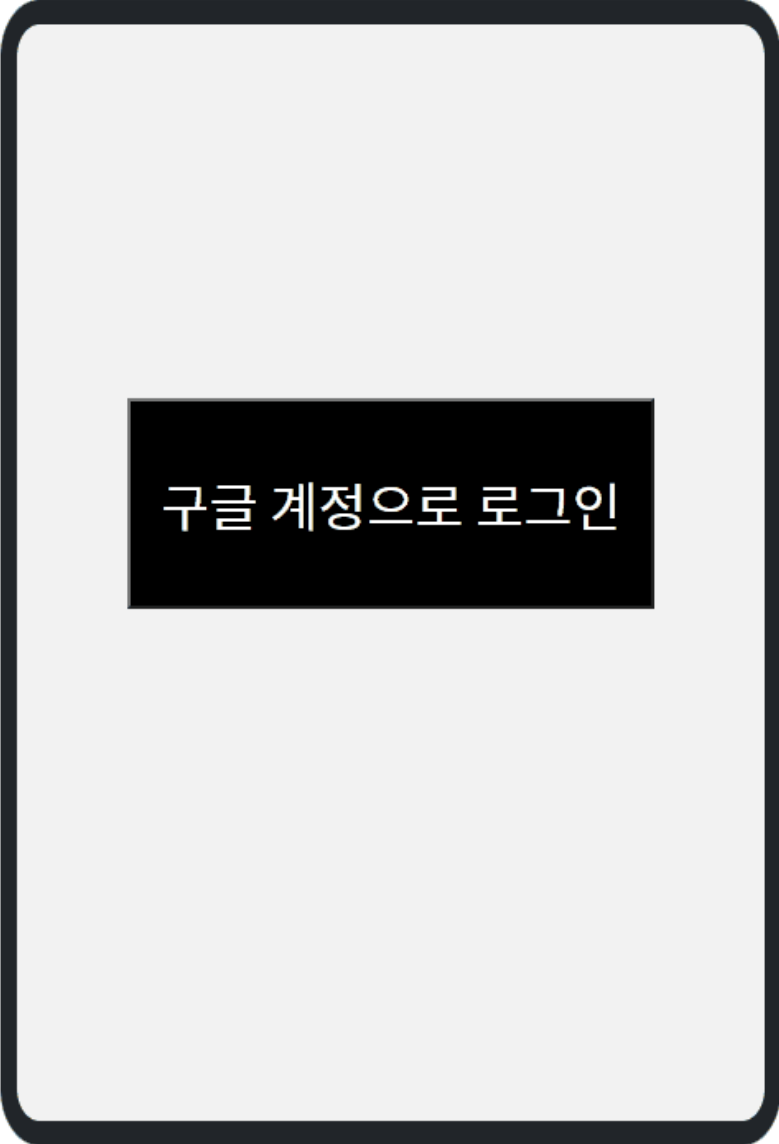
Notes

메모

2022-02-03 취업 준비하기

구글 로그인

PDF에선 해당 gif가 움직이지 않습니다.



구글 계정으로 로그인

구글 로그인

HOW ?

- allauth 라이브러리로 구글 로그인 구현
- 장고 User 모델 커스텀
- allauth 로그인 로직 커스텀

Issue

장고 allauth를 통해 구글 계정으로 로그인 할 경우 유저 모델의 username이 'user'로 저장된다. 그러므로 username으로 user를 식별하기 어려우며, email도 사용하고 싶지 않았다.

Solution

Profile 모델을 별도로 만들어 user를 OneToOneField로 1:1 관계를 설정해준다. 이후 memo를 조회할 때는 닉네임을 이용하여 유저를 식별 하도록 한다.

구글 로그인

allauth 커스텀

allauth의 기본 template의 디자인이 쓸 수 없는 정도로 좋지 못하다.

-> Django에서 렌더링할 때에 template을 찾는 원리를 이용하여 template 커스텀

allauth의 소셜 로그인 과정이 너무 번거롭다.

버튼만 누르면 구글 로그인 page로 넘어가도록 설정하고 싶다.

-> allauth의 구글 로그인 form을 만드는 view를 찾아서 사용

관련 정리 글 : <https://junghogit.github.io/django/django-allauth-custom/>

구글 로그인

닉네임 생성

구글 로그인에 성공 했을 경우 1:1 관계인 Profile 모델을 생성하도록 설계

```
@login_required(login_url='/accounts/login')
def index(request):
    try:
        Profile.objects.get(user_id__= request.user.pk)

        return render(request, 'memo/index.html')
    except Profile.DoesNotExist:
        return redirect('accounts:create_nickname')
```

1. login_required 데코레이터로 로그인하지 않았을 경우 구글 로그인 페이지로 redirect
2. 로그인 된 상태라면 request의 user 정보를 이용하여 Profile 모델을 조회
3. 생성된 프로필이 없을 경우 예외처리 문법으로 닉네임 생성 페이지로 redirect

메모 api

List api View

```
@api_view(['GET', 'POST'])
def memo_list(request, nickname):

    if request.method == 'GET':
        memo = Memo.objects.filter(author=nickname)
        serializer = MemoSerializer(memo, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

    elif request.method == 'POST':
        memo = MemoSerializer(data=request.data)
        if memo.is_valid():
            memo.save()
            return Response(status=status.HTTP_201_CREATED)

    return Response(status=status.HTTP_400_BAD_REQUEST)
```

endpoint

```
urlpatterns = [
    path('', views.index, name='index'),
    path('api/memo/<str:nickname>', views.memo_list, name='memo_list'),
    path('api/memo/<str:nickname>/<int:pk>', views.memo_detail, name='memo_detail'),
]
```

Detail api View

```
@api_view(['GET', 'PUT', 'DELETE'])
def memo_detail(request, pk, nickname):
    try:
        memo = Memo.objects.get(id=pk)
    except Memo.DoesNotExist:
        return Response(status=status.HTTP_400_BAD_REQUEST)

    if request.method == 'GET':
        serializer = MemoSerializer(memo)
        return Response(serializer.data, status=status.HTTP_200_OK)

    elif request.method == 'PUT':
        serializer = MemoSerializer(memo, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(status=status.HTTP_201_CREATED)
        else:
            return Response(status=status.HTTP_400_BAD_REQUEST)

    elif request.method == 'DELETE':
        memo.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

메모 api

Serializer

```
class MemoSerializer(ModelSerializer):  
    class Meta:  
        model = Memo  
        fields = [  
            'title',  
            'content',  
            'author',  
            'create_at',  
            'id',  
        ]
```

HOW ?

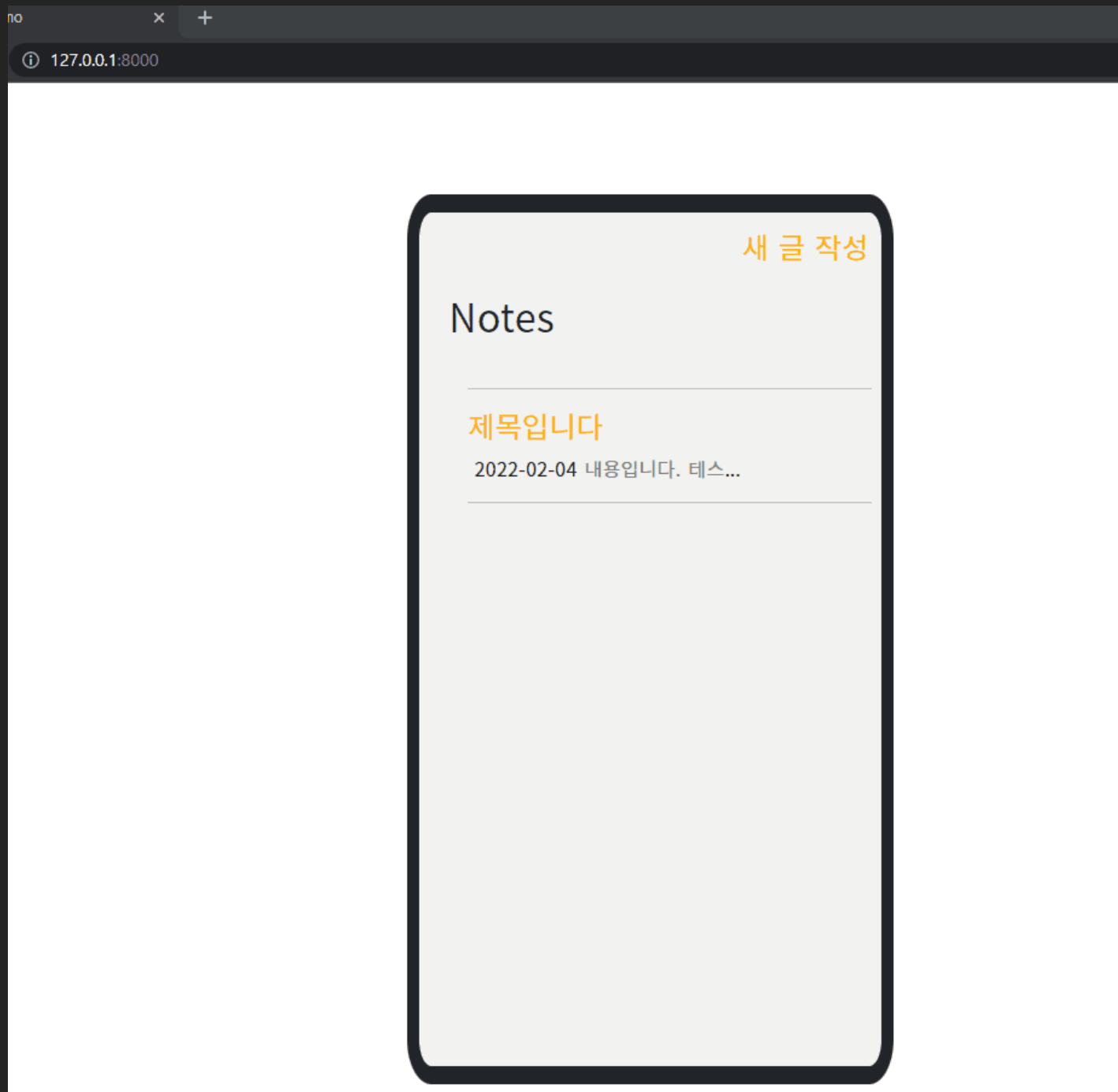
1. 클라이언트에서 ajax 방식으로 API 요청
2. 서버에서 HTTP method로 요청을 분기하여 처리
3. 유효성 검사
4. Json 형식으로 데이터 응답 혹은 데이터 저장, 삭제
5. 클라이언트에서 응답을 받아 비동기 방식으로 HTML 재구성

메모 api

URL 이동 없이 SPA 구현

Javascript 코드는 깃헙에서 확인 가능합니다.

PDF에선 해당 gif가 움직이지 않습니다.



프로젝트 후기

보완점

함수기반이 아닌 클래스 기반으로도 만들어 보고 싶다. (ViewSet, generic View)

CSS에서 계획과 다르게 타협을 본 부분이 많다.

소감

'서비스를 개발한다.'라는 느낌보단 restful api를 설계하고 사용 하는 것을 실습하는 것에 초점을 맞춰서 개발했다.

allauth를 커스텀 하는 과정에서 OOP에 대해서 레벨업을 하고, 소스 코드를 보는 기술이 생겼다.

수박 겉핥기 수준의 restful을 구현했기 때문에 내가 잘 만들고 옳게 이해하고 있는지에 대한 의문이 든다.

Btc village 프로젝트도 restful하게 api를 만들 수 있었을까 ? 하는 호기심이 생겼다.

예상보다 프론트의 비중이 높아서 HTML이나 css, javascript에서 더 큰 공부가 됐다.