

해커톤 대비: Django + Git 복습

Session 20

NEXT X LIKELION 강동혁

목차

1. Git 복습

2. Django 복습

3. 오늘의 실습: Django + Git

4. 해커톤 안내

1. Git 복습

혼자서 캐리



정확한 역할 분배를 통한
협업



1. Git 복습

프론트엔드와 백엔드로 나눠서 역할 분배

프론트엔드

- Django Templates의 html, css, javascript를 다룬다.
- html에 어떤 변수들이 어디에 들어갈지를 구상한다.
- css로 스타일을 입힌다.
- Javascript로 동작처리 + 데이터 전송을 처리한다.

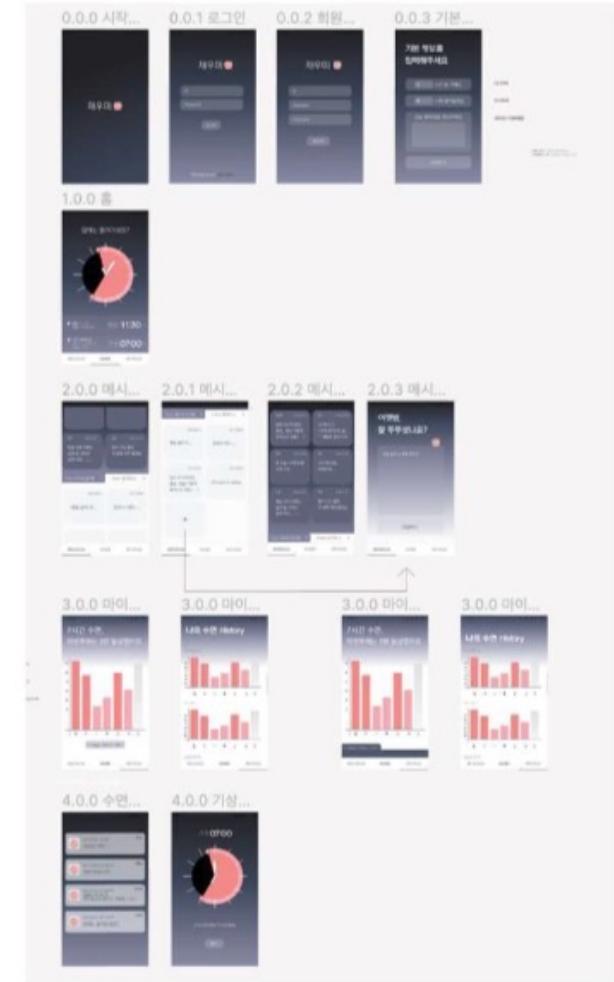
백엔드

- 서비스를 구성하기에 적합한 model을 짠다.
- models.py, urls.py, views.py를 다룬다.
- urls.py에서 페이지별 경로와 특정 동작을 수행하는 경로를 제작한다.
- views.py에서 각 함수마다의 로직을 짠다.

1. Git 복습

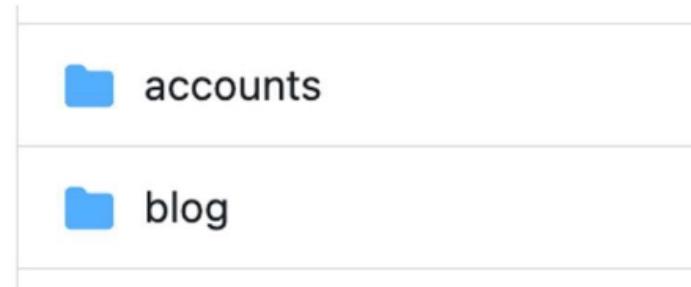
페이지 단위별로 역할 분배 : 프론트엔드

- A = 시작 페이지, B = 홈, C = 메세지 페이지, D = 마이페이지
- 각 페이지마다 필요한 기능과 요구사항들을 정리하기
- 각 페이지마다 필요한 백엔드와 프론트엔드 구현하기



1. Git 복습

독립 개발 단위로 역할 분배 : 백엔드



- 향후 Django에서 배울 app 분리를 이용
- A는 account를, B는 blog를 맡아서 모두다
개발하는 방법

1. Git 복습

기능 단위별로 역할 분배 : 백엔드

재우미 기능 명세서

상태	기능분류	type	담당자	Au 기능명	기능설명	중요도
Done	-2.페이지 나누기	프론트		-2.0.Dreact페이지 나누기	react-router-dom 5v를 활용하여 페이지를 나눈다.	필수
Done	-3.인포너트					
Done	0.시작	프론트		0.0.0 시작 화면 구성	처음시작 하면	필수
Done	0.시작	프론트	백엔드	0.0.1 로그인	1.로그인 하면 firebase를 통한 로그인 인증 3.로그인 후 헤더만으로 이동	필수
Done	0.시작	프론트	백엔드	0.0.2 회원가입	회원가입 정보 입력 → 기본정보 입력 으로 이동	필수
Done	0.시작	프론트	백엔드	0.0.3 기본 정보 입력	기본 정보 입력, db 저장	필수
Done	1.로그인	프론트	백엔드	1.0.0로그인	1.유저 데이터 가져오기 시간별 예제 지 데이터 가져오기 2.유저 데이터 총 회원 시간 데이터, 기 상 시간 데이터 보여주기 3.데이터 데이터 시간별로 보여주기	필수
Done	1.로그인	프론트		1.0.0로그인	시간 별로 주는 유 저 시간이 갈수록 보여주기	필수
Done	1.로그인	프론트		1.0.0로그인	백엔드 흡수면 수정	
Done	2.메세지 페이	프론트	백엔드	2.0.0 메세지 헤	프론트 메시지 화면 구성 메시지 db 요청 api 전송하기 지급하지 작성한 나의 메세지, user에 세지 보여주기	중
Done	2.메세지 페이	프론트	백엔드	2.0.1 메세지 개인	개인 메세지 db 불러오기 개인 메세지 보여주기	중
Done	2.메세지 페이	프론트	백엔드	2.0.2 메세지 모두	유저 메세지 db 불러오기 유저 메세지 보여주기	중
Done	2.메세지 페이	프론트	백엔드	2.0.2 메세지 작성하기	메시지 작성 db 저장	중

- **요구사항을 정확하게 정리하여 기능 단위별로 역할을 분배한다.**
- **장점: 요구사항별 역할분배 가능**
- **단점: 요구사항을 정리하고 분배하기 번거로움**

1. Git 복습

[경로 탐색] **pwd**: print (current) working directory

[경로 이동] **cd**: change directory

[파일 확인] **ls**: list segments

ls -a: list segments of all

ls -l: list directory with long listing

ls -al: list directory of all with long listing

[파일 생성] **touch**: create a file

mkdir: make a directory

[파일 제거] **rm**: remove a file

rm -rf: remove forcibly including directory and the files in it

⟨git 초기 설정⟩

[전역 사용자명/이메일 구성하기] **git config --global user.name \${user.name}**
git config --global user.email \${user.email}
git config --list

[새로운 저장소 초기화] **git init**

[저장소 복제] **git clone \${깃헙 주소}**

[Local & Remote 저장소 연결] **git remote add origin \${본인 레포 주소}**
git remote -v

⟨git 파일 스테이징 & 푸시⟩

[git 스테이징 영역에 새로운 파일 추가] **git add \${file_name}**
git status

[git 스테이징 영역에 존재하는 파일 커밋하기] **git commit -m "\${commit message}"**
git log

[Github 레포에 코드 올리기] **git push origin \${branch name}**

1. Git 복습

git clone

- Remote repository의 모든 데이터를 복사하여 Local repository에 새로운 git 저장소를 생성
- Remote repository의 모든 커밋 히스토리, 브랜치, 태그 등이 포함 됨.
- 자동으로 Local에 Remote repository가 origin이라는 이름으로 등록됨

git remote add origin

- Local repository에 Remote repository를 연결함
- origin 은 remote repository의 별칭(별명), origin을 통해서 local repository와 remote repository의 통신이 가능해짐

👉 git clone은 복제, git remote add origin 연결 이구나

복제하면 연결까지 미리 다 되는구나!

1. Git 복습

Reset

- `git reset -${option} ${커밋id}`
- C4의 상황에서 C2의 커밋으로 Head를 변경함

주의사항: Local Repository에서만 사용 권고
다른 코드를 삭제하거나 다른 코드와 충돌할 수 있음
즉, Push하기 이전에만 사용하는게 좋음



1. Git 복습

Reset 명령어 정리

reset --soft: **git reset --soft \${commit ID}**

reset --mixed: **git reset --mixed \${commit ID}**

= git reset \${commit ID}

reset --hard: **git reset --hard \${commit ID}**

reset HEAD~숫자: **git reset HEAD~10**

reset HEAD^: **git reset HEAD^**

reset HEAD: **git reset HEAD**

--soft 또는 -mixed 또는 --hard

git reset \${원하는 옵션} \${commit Id 또는 HEAD 형식}

soft → commit 취소하기

Commit한 파일들을 staging area로 돌린다.

mixed → (default값) commit 취소 + add 취소

Commit한 파일들을 working directory로 돌린다.

hard → commit 취소 + add 취소 + 변경된 파일 삭제

Commit한 파일들을 working directory에서 삭제한다.

Head~\${취소할 커밋 수} → 현재로부터 숫자만큼 commit 취소

HEAD^ → 가장 최근 커밋이 취소된다.

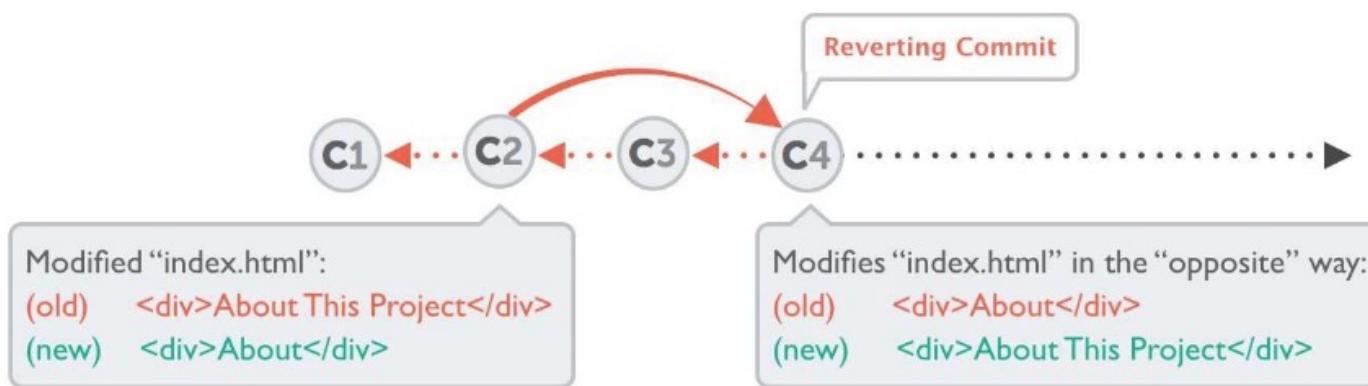
HEAD → 변경된 파일들을 unstaged 상태로 되돌린다.

1. Git 복습

Revert

- `git revert ${커밋id}`
- C2에 해당하는 변경사항을 되돌리고, C4라는 새로운 commit

을 남김



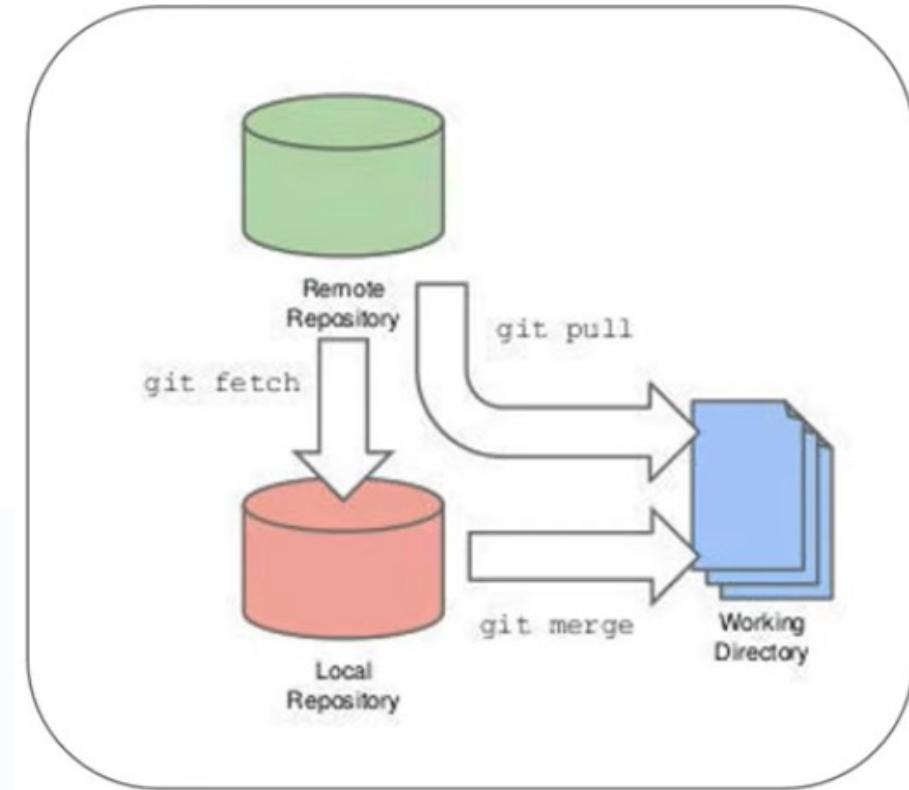
1. Git 복습

Fetch : 변동 사항만 가져오기

```
git fetch [원격저장소 이름]  
# 원격저장소에 변동사항만을 가져 옵니다
```

```
git merge FETCH_HEAD  
# FETCH_HEAD에 업데이트된 원격저장소의 최신 커밋이, 현재 브랜치에 병합 됩니다.
```

```
git fetch; git merge FETCH_HEAD  
# git fetch + merge FETCH_HEAD 명령어를 한 번에 사용
```



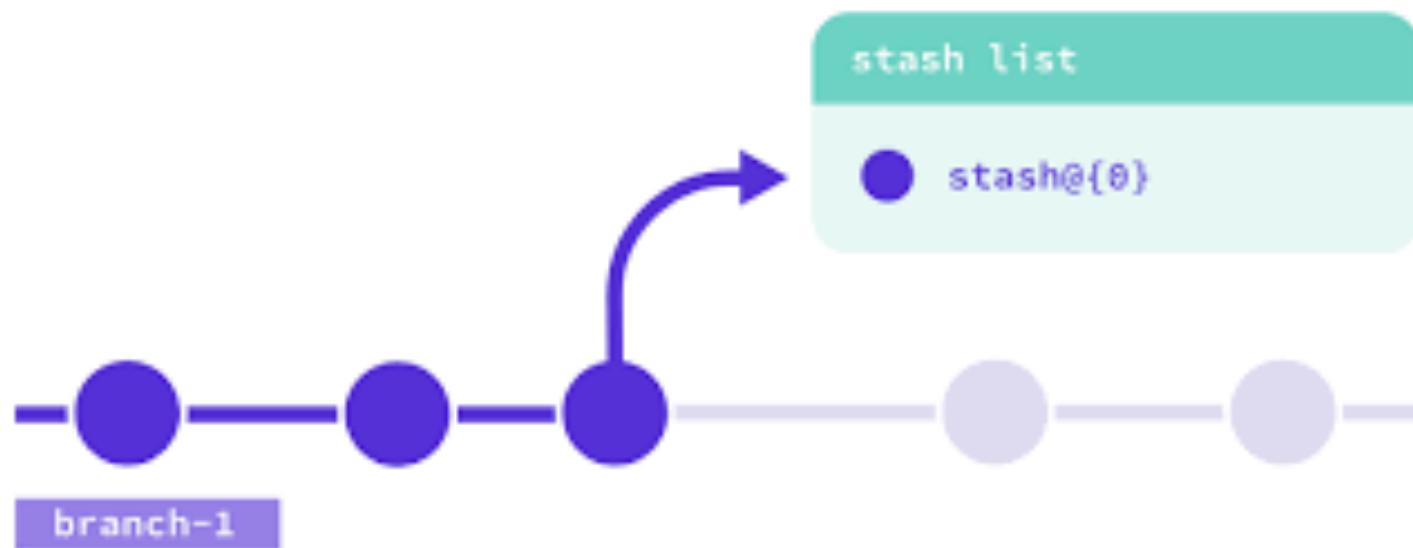
1. Git 복습

Stash : 임시저장

```
(Git-test2) kangdonghyuk@gangdonghyeog-ui-noteubug Git-test2 % git pull origin main
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 3), reused 6 (delta 3), pack-reused 0
Unpacking objects: 100% (6/6), 1.16 KiB | 119.00 KiB/s, done.
From https://github.com/cucumber5252/Git-test2
 * branch           main      -> FETCH_HEAD
   d6fe413..7716730  main      -> origin/main
Updating d6fe413..7716730
error: Your local changes to the following files would be overwritten by merge:
      testProject/db.sqlite3
Please commit your changes or stash them before you merge.
Aborting
```

1. Git 복습

Stash : 임시저장

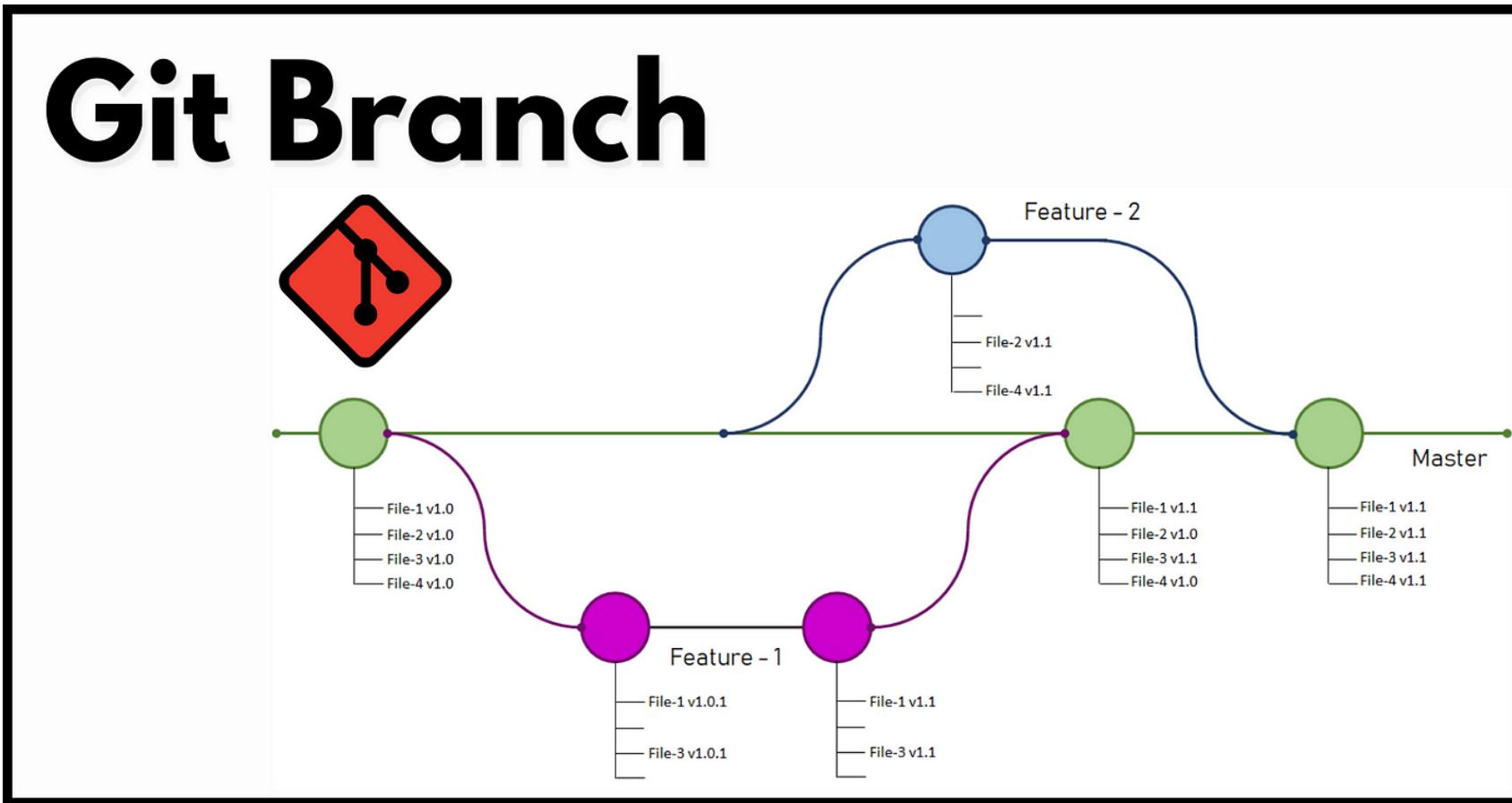


1. Git 복습

Stash : 임시저장

명령어	설명	비고
git stash	현 작업들 치워두기	끝에 save 생략
git stash apply	치워둔 마지막 항목(번호 없을 시) 적용	끝에 번호로 항목 지정 가능
git stash drop	치워둔 마지막 항목(번호 없을 시) 삭제	끝에 번호로 항목 지정 가능
git stash pop	치워둔 마지막 항목(번호 없을 시) 적용 및 삭제	apply + drop
💡 git stash branch (브랜치명)	새 브랜치를 생성하여 pop	충돌사항이 있는 상황 등에 유용
git stash clear	치워둔 모든 항목들 비우기	

1. Git 복습



1. Git 복습

Branch 명령어 정리

Branch 생성: `git branch ${브랜치 이름}`

Branch 이동: `git checkout ${브랜치 이름}`

Branch 생성과 이동: `git checkout -b ${브랜치 이름}`

Branch 삭제: `git branch -d ${브랜치 이름}`

현재 Branch가 무엇인지 확인: `git branch('Q'로 나가기)`

모든 Branch가 확인: `git branch -r('Q'로 나가기)`

Branch 합치기: `git merge ${브랜치 이름}`

최초 repository 생성 시 브랜치 이름은
→ 'master' or 'main'

대부분의 상황에서 새로운 브랜치를 생성할 때는
→ `git checkout -b ${브랜치 이름}`를 사용
기존 것을 똑같이 복사한 채로 이동하기 때문이야.
(`git branch ${브랜치 이름}`은 생성만 해줌)

`git checkout`을 통해 브랜치 이동하여 독립적인
작업 가능

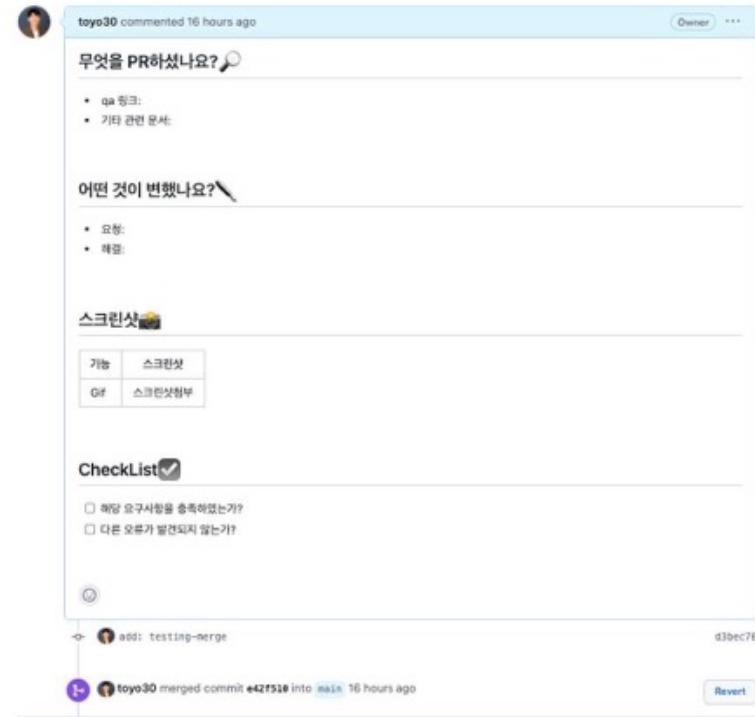
⚠️ 이동하기 전 working directory에 있는 내역
은 사라지고, 바뀐 브랜치의 최신 커밋이
working directory에 반영됨.

1. Git 복습

Git Pull Request

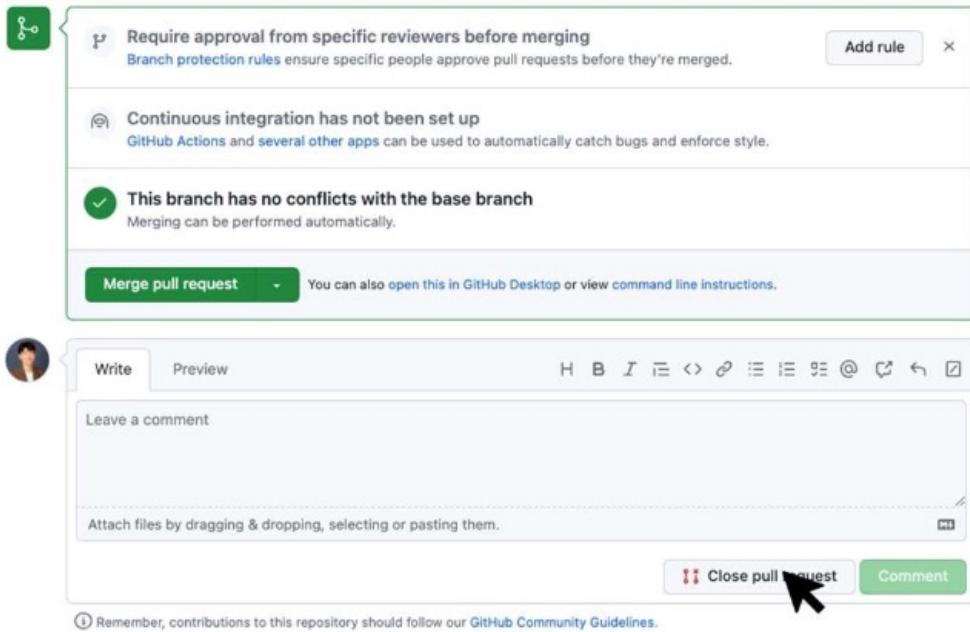
- 코드 변경 사항을 다른 사람에게 알리고 그들의 검토와 승인을 받는 방법
- 이 기능은 주로 GitHub 같은 원격 저장소에서 활용되며, 협업을 하는 개발 환경에서 사용됨
- Pull을 요청(Request)하다

git pull은 다른 저장소 혹은 다른 branch의 코드를 가져오는 git fetch와 코드를 합치는 git merge의 조합이다.



1. Git 복습

Git Merge



코드가 마음에 안 들면, close pull request로
Pull-request를 반영하지 않을 수 있다

1. Git 복습

추천 진행 방식

1. 일단 설계 및 분업

페이지 몇개 만들지, 모델 어떻게 할지, 앱 분리 할지말지, html/url/views의 함수 이름 뭘로 할지 등등

2. Git 환경 설정

한명이 repository 생성 후 설계에 맞춰 Django 프로젝트 기본 틀 생성 후 main에 push

3. 각자 Branch에서 담당 작업하기

Branch명은 각자 이름으로 하는 걸 추천, 파일 단위로 분업화 하는 걸 추천

4. Pull-Request => main에 Merge

한 파일을 여러명이 동시 수정할 경우 Confliction 생길 위험이 크니 조심하기

1. Git 복습

Git을 사용하는 최악의 방법



아 왜 push가 안되지 push -f ㅋㅋㅋ

커밋 메시지 쓰기 귀찮아
git commit -m "커밋1"
git commit -m "커밋2"

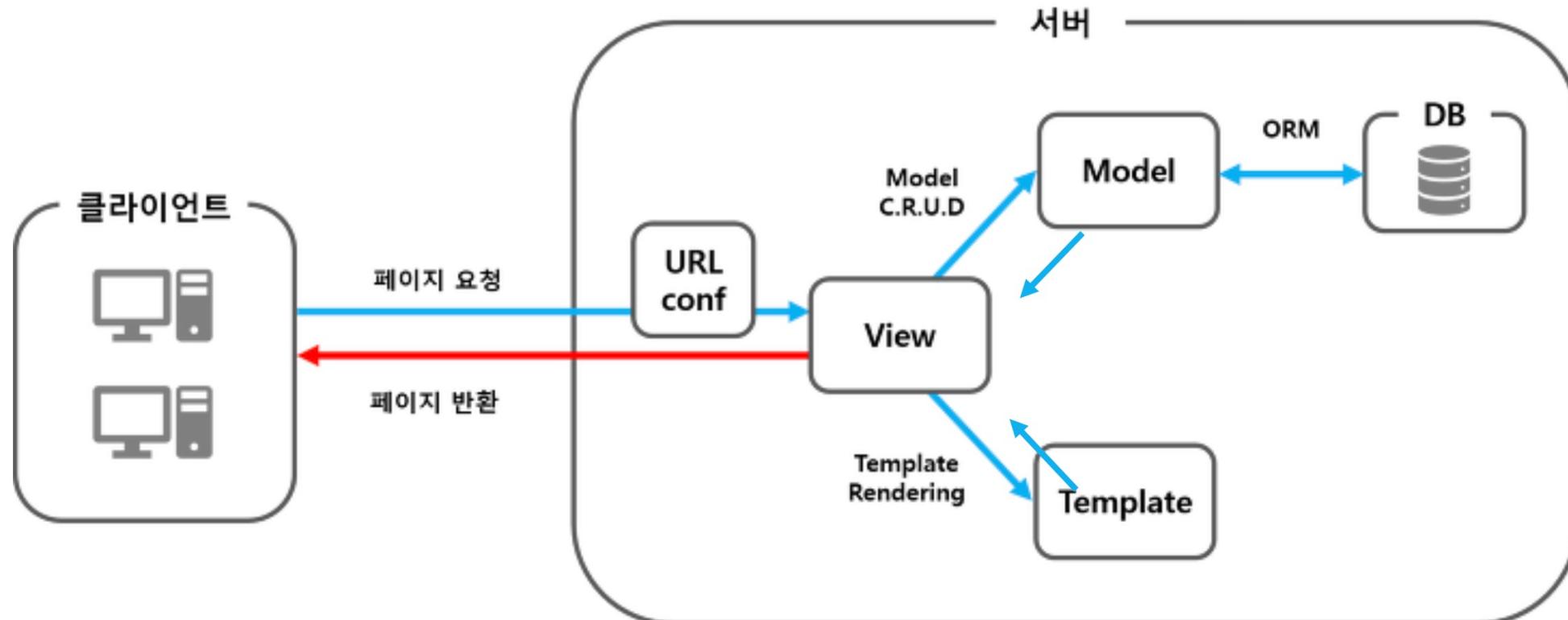


(수정한 것 깜빡하고) git pull origin ~

Pull-request->merge 귀찮아
그냥 main에 바로 merge



2. Django 복습



2. Django 복습

- **MTV** : 장고의 설계 패턴으로, 소프트웨어의 비즈니스 로직과 화면을 구분하는 데 중점을 둔 MVC(Model-View-Controller) 패턴의 개념을 용어만 달리하여 그대로 받아옴

Model : 데이터베이스 설계

- 데이터베이스에 저장되는 '데이터' 영역

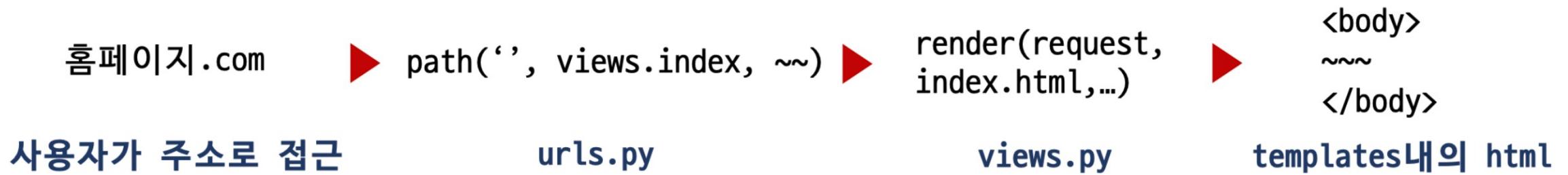
Template : 화면 UI 설계

- 사용자에게 보여지는 HTML '화면' 영역

View : 프로그램 로직 설계

- 요청에 따라 Model에서 필요한 데이터 가져오기 → 처리 → 처리 결과를 Template에 전달

2. Django 복습



2. Django 복습

pipenv shell

가상환경 생성 & 시작

pipenv install django

가상환경 내에 Django 패키지 설치

django-admin startproject {project}

장고 프로젝트 생성 & 이동

cd {project}

python manage.py startapp {app}

프로젝트 폴더 내에 장고 앱 생성

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'app',  
]
```

```
TIME_ZONE = 'Asia/Seoul'
```

python manage.py runserver

로컬 서버에서 장고 서버 실행

python manage.py makemigrations

migrations 만들기

python manage.py migrate

DB에 반영

python manage.py createsuperuser

관리자 계정 생성

2. Django 복습

CRUD

create

MTV

project/project/urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('new/', views.new, name="new"),
]
```

2. Django 복습

CRUD

create

MTV

project/app/templates/new.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="" method="post">
        {% csrf_token %}
        <div>
            <label for="title">제목</label>
            <input type="text" name="title" id="title" placeholder="제목을 입력해주세요.">
        </div>
        <div>
            <label for="content">내용</label>
            <textarea name="content" id="content" cols="30" rows="10" placeholder="내용을 입력해주세요."></textarea>
        </div>
        <button type="submit">작성하기</button>
    </form>
</body>
</html>
```

2. Django 복습

CRUD

create

MTV

```
project/app/templates/new.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="" method="post">
        {% csrf_token %}
        <div>
            <label for="title">제목</label>
            <input type="text" name="title" id="title" placeholder="제목을 입력해주세요.">
        </div>
        <div>
            <label for="content">내용</label>
            <textarea name="content" id="content" cols="30" rows="10" placeholder="내용을 입력해주세요."></textarea>
        </div>
        <button type="submit">작성하기</button>
    </form>
</body>
</html>
```

2. Django 복습

CRUD

create

MTV

project/app/views.py

```
def new(request):
    if request.method == 'POST':
        Post.objects.create(
            title = request.POST['title'],
            content = request.POST['content']
        )

    return render(request, 'new.html')
```

2. Django 복습

CRUD

read

MTV

project/project/urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", views.home, name="home"),
    path('new/', views.new, name="new"),
    path('detail/<int:post_pk>', views.detail, name="detail"),
]
```

2. Django 복습

CRUD

read

project/app/templates/detail.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div>
    <h2>책 제목</h2>
    <p>{{ post.title }}</p>
  </div>
  <div>
    <h2>책 내용</h2>
    <p>{{ post.content }}</p>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
</body>
</html>
```

MTV

2. Django 복습

CRUD

read

project/app/templates/home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>독후감 블로그</h1>
    <div>
        <ul>
            {% for post in posts %}
                <li>
                    <a href="{% url 'detail' post.pk %}">{{ post.title }}</a>
                </li>
            {% endfor %}
        </ul>
    </div>
    <a href="{% url 'new' %}">글 작성하기</a>
</body>
</html>
```

MTV

2. Django복습

CRUD

read

MTV

project/app/views.py

```
from django.shortcuts import render, redirect

def new(request):
    if request.method == 'POST':
        new_post = Post.objects.create(
            title = request.POST['title'],
            content = request.POST['content']
        )
        return redirect('detail', new_post.pk)

    return render(request, 'new.html')

def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    return render(request, 'detail.html', {'post': post})
```

2. Django 복습

CRUD

update

MTV

project/project/urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", views.home, name="home"),
    path('new/', views.new, name="new"),
    path('detail/<int:post_pk>/', views.detail, name="detail"),
    path('update/<int:post_pk>/', views.update, name="update"),
]
```

2. Django 복습

CRUD

update

project/app/templates/update.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="" method="post">
        {% csrf_token %}
        <div>
            <label for="title">제목</label>
            <input type="text" id="title" name="title" value="{{ post.title }}">
        </div>
        <div>
            <label for="content">내용</label>
            <textarea name="content" id="content" cols="30" rows="10" >{{ post.content }}</textarea>
        </div>
        <button type="submit">수정하기</button>
    </form>
</body>
</html>
```

MTV

2. Django 복습

CRUD

update

MTV

project/app/views.py

```
def update(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    if request.method == 'POST':
        Post.objects.filter(pk=post_pk).update(
            title = request.POST['title'],
            content = request.POST['content']
        )
        return redirect('detail', post_pk)

    return render(request, 'update.html', {'post': post})
```

주의! update()는 QuerySet 타입에서만 사용이 가능합니다.

filter()는 여러개의 객체를 포함하는 QuerySet을 반환해줍니다. (all()도 QuerySet을 반환함)
그러나, get()은 객체(여기선 Post) 하나만 반환해줍니다.
따라서 update()는 all()이나 filter() 후에 사용해야합니다.

2. Django 복습

CRUD

delete

MTV

project/project/urls.py

```
from django.contrib import admin
from django.urls import path
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('new/', views.new, name="new"),
    path('detail/<int:post_pk>/', views.detail, name="detail"),
    path('update/<int:post_pk>/', views.update, name="update"),
    path('delete/<int:post_pk>/', views.delete, name="delete"),
]
```

2. Django 복습

CRUD

delete

MTV

project/app/templates/detail.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div>
    <h2>책 제목</h2>
    <p>{{ post.title }}</p>
  </div>
  <div>
    <h2>책 내용</h2>
    <p>{{ post.content }}</p>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
  <a href="{% url 'update' post.pk %}">수정하기 </a>
  <a href="{% url 'delete' post.pk %}">삭제하기 </a>
</body>
</html>
```

2. Django 복습

CRUD

delete

MTV

project/app/views.py

```
def delete(request, post_pk):
    post = Post.objects.get(pk=post_pk)
    post.delete()

    return redirect('home')
```

detail 페이지에서 ‘삭제하기’를 눌러

pk를 통해 받아온 모델이 삭제되는지 확인해보세요!

2. Django 복습

RDBMS (Relational DBMS)

- Foreign Key

[고려대학교 학생]

학번	이름	학과	수강 강의번호
2021130840	손예원	컴퓨터학과	4
328956274	최유빈	미디어학부	1
1234567890	김태균	체육교육과	2
2037561298	나성희	경영학과	3
3042783569	박소정	경영학과	3

참조

[고려대학교 강의]

강의번호	강의명	담당교수
1	미디어경제	김정현
2	보건교육론	이승희
3	회계원리	김진배
4	운영체제	유현창

Foreign Key :

참조 대상 테이블의 tuple(=행)을 식별할 수 있는 key
[학생]과 [강의] 테이블은 각각 존재하며, 두 테이블 간에는 참조 관계가 있다

NEXT X LIKELION

NEXT X LIKELION

2. Django 복습

Foreign Key

```
class Post(models.Model):
    title = models.CharField(max_length=50)
    content = models.TextField()

    def __str__(self):
        return self.title

class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name='comments')
    content = models.TextField()

    def __str__(self):
        return self.content
```

2. Django 복습

on_delete?

- **CASCADE** : 참조 대상인 Post가 삭제되면, 해당 Post를 참조하고 있던 Comment들도 따라서 삭제!
- **SET_NULL** : 참조 대상인 Post가 삭제되면, 해당 Post를 참조하고 있던 Comment의 ForeignKey를 NULL값으로! (**null=True)
- **SET_DEFAULT** : 참조 대상인 Post가 삭제되면, 해당 Post를 참조하고 있던 Comment의 ForeignKey를 '기본값'으로! (**기본값 설정이 필요합니다)
- **PROTECT** : 참조 대상인 Post를 삭제하려 하면, Protected Error를 띄워 삭제를 막습니다!
- ※ 기타 : DO NOTHING, RESTRICT....

2. Django 복습

Admin에서 Model 보여주기

```
from django.contrib import admin
from .models import Post

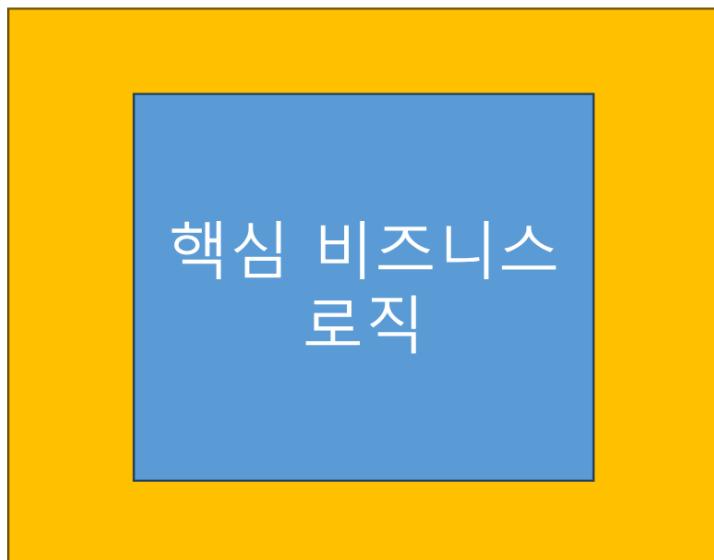
# Register your models here.
admin.site.register(Post)      cucumber5252,
```

2. Django 복습

Django.contrib.auth

직관적 구조 : decorator란?

Decorator란 함수에 덮어 씌어져 추가적인 기능을 제공하는 **함수**
Wrapper function



@login_required

1. 유저가 로그인 되어 있는지를 검증하고,
- 2-1. 안 되어있다면 login_url로 보내고
- 2-2. 되어있다면 request.user에 유저 객체를 첨부해서
3. 다음 핵심 비즈니스 로직을 실행한다.

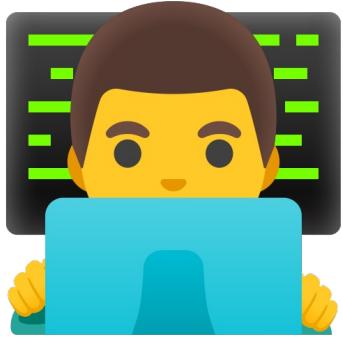
감싸고 있다는 것은, 핵심 비즈니스 로직이 끝난 다음에도 무언가를
수행할 수 있다는 것

3. 오늘의 실습: Django + Git

조건

- Model은 Post 하나이고, title과 content라는 객체를 가져야 함
- Templates는 각각 list.html과 detail.html이며, 두개의 url은 '/list'와 '/detail/<int>'
- list.html에서는 모든 Post의 제목이 있어야 하며 클릭 시 각 detail 페이지로 넘어가야함
- detail.html에서는 선택한 Post의 제목과 내용이 보여야만 함
- 각 templates에서 사용되는 view 함수의 이름은 template 명과 동일함
- 테스트를 위해 admin 페이지를 통해 다섯개의 post를 넣기

3. 오늘의 실습: Django + Git



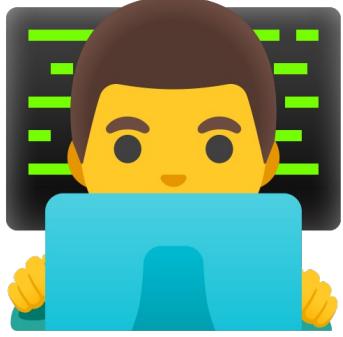
A

- Repo + 초반 프로젝트 세팅
- list 관련 로직 구현

- list 관련 프론트 구현



C



B

- detail 관련 로직 구현
- confliction 해결

- detail 관련 프론트 구현



D

3. 오늘의 실습: Django + Git

- 1) A가 Repository를 만들기
- 2) A, B, C, D는 빈 Repository를 Clone하기
- 3) A가 로컬에서 프로젝트와 앱을 만들고, 기본 파일 구조 만들기
- 4) 이후 A가 admin.py와 models.py를 수정한 이후 admin 페이지를 이용하여 테스트용 Post를 다섯 개 추가하기
- 5) A는 변경이 완료된 이후 main에 push하고 B, C, D는 main을 pull
- 6) 각자 이름으로 branch를 만든 후 각자 branch에 수정 사항들을 push
- 7) 작업이 끝나면 각 branch를 main으로 merge
- 8) 백 관련 로직에서 발생한 confliction을 D가 Github 상에서 해결하기

4. 해커톤 안내

주제&팀원: 6/6 세션 중 공개

장소: 서울시 중구 청계천로 100 시그니처타워, 10층 동관 코드잇

일정

개발- 6월 8일 오전 10시 ~ 6월 9일 오전 10시 (Github push 기준)

발표 및 시연- 6월 8일 오전 11시 (예정)

4. 해커톤 안내

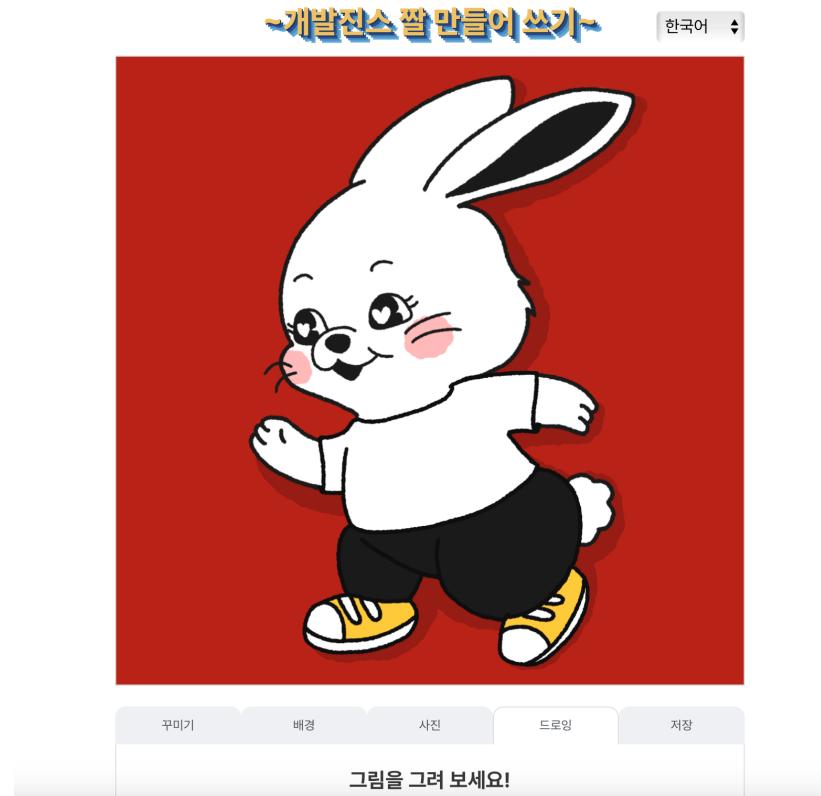
!! 핵심 포인트 !!

- 1) 모든 수상팀은 학회원 투표를 통해 결정
- 2) 발표의 형식은 자유이며 서비스를 매력적으로 어필할 수 있는 모든 수단 활용 가능
- 3) 시연은 필수이나, 배포는 필수는 아님
- 4) 데모 데이가 아님 => 비즈니스적으로 리즈너블해도 되고, 하지 않아도 됨

4. 예시 서비스

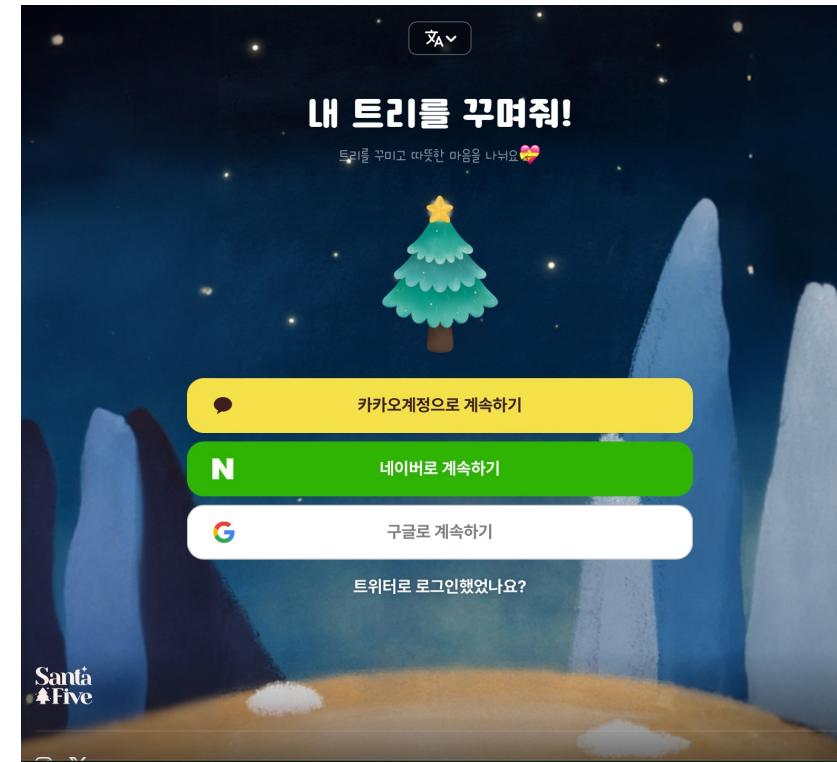
-개발진스 짤 만들기

<https://devjeans.dev-hee.com/>



-내 트리를 꾸며줘

<https://colormytree.me/auth>



4. 작년 수상작

-연세치킨

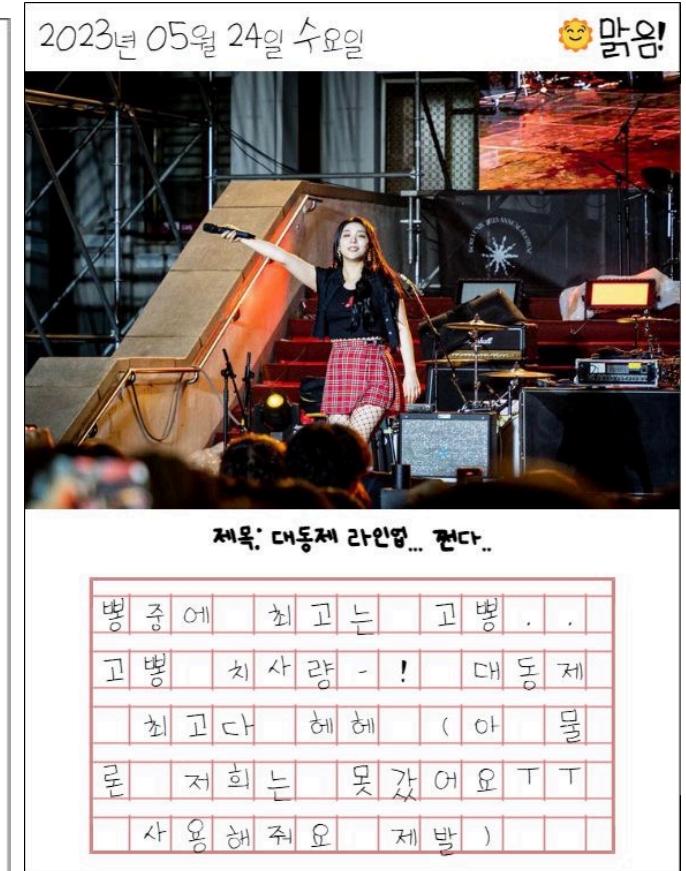
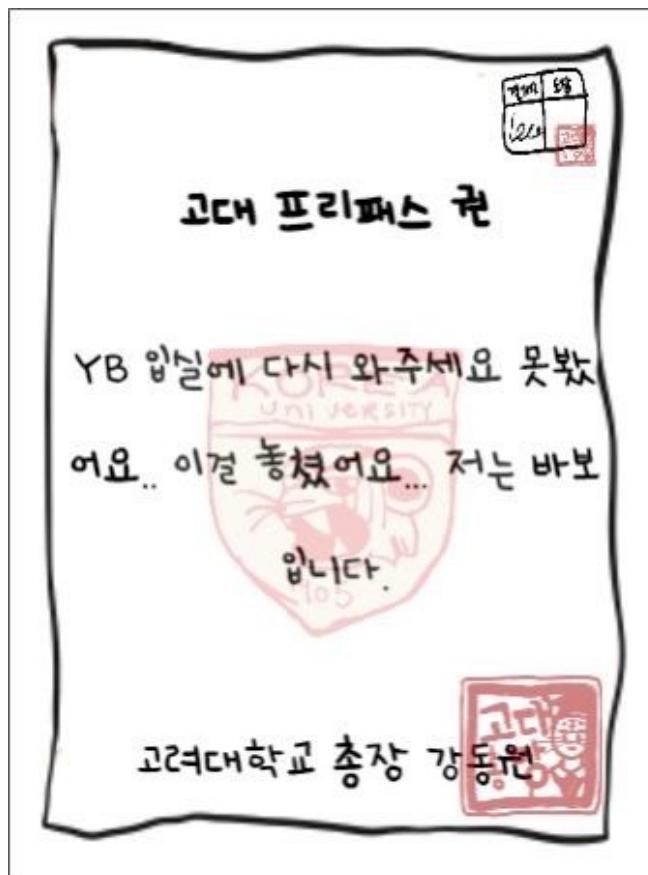


연세 치킨 튀기기

연세 잡기

연세 닭장

-상장 & 그림일기 제작



4. 주의사항

- 1) 개발적 완성도에 매몰되시면 안됩니다 ! 제가 그랬습니다..
- 2) 단순 아이디어 회의 이외에, 코드는 절대 미리 작성해오시면 안됩니다 !
- 3) 운영진 질의응답은 최소한으로 진행됩니다 !
- 4) 배포는 필수는 아니지만, 추천 드립니다 !
- 5) 모든 평가는 9일 10시 깃허브 push 기록을 기준으로 진행됩니다 !
- 6) 프레임워크는 Django를 기본으로 하되, React+DRF 조합도 가능합니다 !

책임질 수 있다면

과제

없습니다!

밀린 과제가 있으시다면 늦더라도 꼭 제출해주시고,

밤샘 해커톤을 위한 **체력 안배**에 힘써주세요!!