|  |
| --- |
| **디자인** |
| **동아리 디테일 페이지**  **▪ 동아리 이름과 동아리 소개**  **▪ 동아리 포스트 목록(옆으로 드래그)**  **▪ 동아리 Q&A 목록**  **동아리 목록 페이지**  **▪ navbar – 사이드바메뉴,프로필**  **▪ 검색기능**  **▪ 동아리 목록 – 동아리카테고리, 조회수** |
| **동아리 게시물 목록 페이지**  **▪ navbar – 사이드바메뉴,프로필**  **▪ 인기 동아리 게시물 목록**  **▪ 전체 동아리 게시물 목록 – 조회수, 댓글 수 표시**  **게시물 디테일 페이지**  **▪ 게시물 제목**  **▪ 게시물 내용**  **▪ 게시물 댓글 대댓글** |
| Model 정의 |
| from django.db import models  from django.contrib.auth.models import User  from django.db.models import signals  from django.dispatch import receiver  from allauth.socialaccount.models import SocialAccount  class AccountEmailaddress(models.Model):  email = models.CharField(unique=True, max\_length=254)  verified = models.IntegerField()  primary = models.IntegerField()  user = models.ForeignKey('AuthUser', models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'account\_emailaddress'  class AccountEmailconfirmation(models.Model):  created = models.DateTimeField()  sent = models.DateTimeField(blank=True, null=True)  key = models.CharField(unique=True, max\_length=64)  email\_address = models.ForeignKey(AccountEmailaddress, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'account\_emailconfirmation'  class AuthGroup(models.Model):  name = models.CharField(unique=True, max\_length=150)  class Meta:  managed = False  db\_table = 'auth\_group'  class AuthGroupPermissions(models.Model):  group = models.ForeignKey(AuthGroup, models.DO\_NOTHING)  permission = models.ForeignKey('AuthPermission', models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'auth\_group\_permissions'  unique\_together = (('group', 'permission'),)  class AuthPermission(models.Model):  name = models.CharField(max\_length=255)  content\_type = models.ForeignKey('DjangoContentType', models.DO\_NOTHING)  codename = models.CharField(max\_length=100)  class Meta:  managed = False  db\_table = 'auth\_permission'  unique\_together = (('content\_type', 'codename'),)  class AuthUser(models.Model):  password = models.CharField(max\_length=128)  last\_login = models.DateTimeField(blank=True, null=True)  is\_superuser = models.IntegerField()  username = models.CharField(unique=True, max\_length=150)  first\_name = models.CharField(max\_length=150)  last\_name = models.CharField(max\_length=150)  email = models.CharField(max\_length=254)  is\_staff = models.IntegerField()  is\_active = models.IntegerField()  date\_joined = models.DateTimeField()  class Meta:  managed = False  db\_table = 'auth\_user'  class AuthUserGroups(models.Model):  user = models.ForeignKey(AuthUser, models.DO\_NOTHING)  group = models.ForeignKey(AuthGroup, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'auth\_user\_groups'  unique\_together = (('user', 'group'),)  class AuthUserUserPermissions(models.Model):  user = models.ForeignKey(AuthUser, models.DO\_NOTHING)  permission = models.ForeignKey(AuthPermission, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'auth\_user\_user\_permissions'  unique\_together = (('user', 'permission'),)  class ClubTypes(models.Model):  club\_type\_id = models.AutoField(primary\_key=True)  club\_type\_name = models.CharField(max\_length=200, null=True)  club\_type\_desc = models.CharField(max\_length=200, blank=True, null=True)  created\_at = models.DateTimeField()  updated\_at = models.DateTimeField()  class Meta:  managed = False  db\_table = 'club\_types'  class Clubs(models.Model):  club\_id = models.AutoField(primary\_key=True)  club\_name = models.CharField(max\_length=200, blank=True, null=True)  club\_desc = models.CharField(max\_length=200, blank=True, null=True)  user = models.ForeignKey(User, models.DO\_NOTHING, blank=True, null=True)  club\_type = models.ForeignKey(ClubTypes, models.DO\_NOTHING, null=True)  club\_img\_url = models.ImageField(upload\_to="%Y/%m/%d", null=True)  club\_logo\_url = models.ImageField(upload\_to="%Y/%m/%d", null=True)  established = models.DateTimeField()  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  def \_\_str\_\_(self):  return self.club\_name  class Meta:  managed = False  db\_table = 'clubs'  class ClubsMember(models.Model):  club\_member\_id = models.AutoField(primary\_key=True)  club = models.ForeignKey(Clubs, models.DO\_NOTHING, blank=True, null=True)  club\_member\_name = models.CharField(max\_length=200, blank=True, null=True)  club\_member\_img\_url = models.ImageField(upload\_to="%Y/%m/%d", null=True)  club\_member\_position = models.CharField(max\_length=45, blank=True, null=True)  class Meta:  managed = False  db\_table = 'clubs\_member'  class ClubsQna(models.Model):  question\_id = models.AutoField(primary\_key=True)  question\_title = models.CharField(max\_length=150, blank=True, null=True)  question\_content = models.CharField(max\_length=3000, blank=True, null=True)  user = models.ForeignKey(User, models.DO\_NOTHING, blank=True, null=True)  club = models.ForeignKey(Clubs, models.DO\_NOTHING, blank=True, null=True)  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  is\_deleted = models.IntegerField(default=0)  class Meta:  managed = False  db\_table = 'clubs\_qna'  class DjangoAdminLog(models.Model):  action\_time = models.DateTimeField()  object\_id = models.TextField(blank=True, null=True)  object\_repr = models.CharField(max\_length=200)  action\_flag = models.PositiveSmallIntegerField()  change\_message = models.TextField()  content\_type = models.ForeignKey('DjangoContentType', models.DO\_NOTHING, blank=True, null=True)  user = models.ForeignKey(AuthUser, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'django\_admin\_log'  class DjangoContentType(models.Model):  app\_label = models.CharField(max\_length=100)  model = models.CharField(max\_length=100)  class Meta:  managed = False  db\_table = 'django\_content\_type'  unique\_together = (('app\_label', 'model'),)  class DjangoMigrations(models.Model):  app = models.CharField(max\_length=255)  name = models.CharField(max\_length=255)  applied = models.DateTimeField()  class Meta:  managed = False  db\_table = 'django\_migrations'  class DjangoSession(models.Model):  session\_key = models.CharField(primary\_key=True, max\_length=40)  session\_data = models.TextField()  expire\_date = models.DateTimeField()  class Meta:  managed = False  db\_table = 'django\_session'  class DjangoSite(models.Model):  domain = models.CharField(unique=True, max\_length=100)  name = models.CharField(max\_length=50)  class Meta:  managed = False  db\_table = 'django\_site'  class Posts(models.Model):  post\_id = models.AutoField(primary\_key=True)  post\_title = models.CharField(max\_length=150)  post\_content = models.CharField(max\_length=3000)  post\_introduce = models.CharField(max\_length=200, blank=True, null=True)  post\_img\_url = models.ImageField(upload\_to="%Y/%m/%d", null=True)  user = models.ForeignKey(User, models.DO\_NOTHING, blank=True, null=True)  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  is\_deleted = models.IntegerField(default=0) # This field type is a guess.  club = models.ForeignKey(Clubs, models.DO\_NOTHING, blank=True, null=True, related\_name='club\_posts')  def \_\_str\_\_(self):  return self.post\_title  class Meta:  managed = False  db\_table = 'posts'  class PostsReplies(models.Model):  post\_reply\_id = models.AutoField(primary\_key=True)  user = models.ForeignKey(User, models.DO\_NOTHING, blank=True, null=True)  post = models.ForeignKey(Posts, models.DO\_NOTHING, blank=True, null=True)  parent\_reply = models.ForeignKey('self', models.DO\_NOTHING, blank=True, null=True, related\_name='reply')  post\_reply\_content = models.CharField(max\_length=500, blank=True, null=True)  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  is\_deleted = models.IntegerField(default=0)  class Meta:  managed = False  db\_table = 'posts\_replies'  class PostsViews(models.Model):  post = models.ForeignKey(Posts, models.DO\_NOTHING, null=True, related\_name = 'view')  views\_id = models.AutoField(primary\_key=True)  user = models.ForeignKey(User, models.DO\_NOTHING, blank=True, null=True)  user\_ip = models.CharField(max\_length=16)  checked\_at = models.DateTimeField(auto\_now\_add=True)  def \_\_str\_\_(self):  return str(self.checked\_at)  class Meta:  managed = False  db\_table = 'posts\_views'  class QnaReplies(models.Model):  qna\_reply\_id = models.AutoField(primary\_key=True)  user = models.ForeignKey(User, on\_delete=models.CASCADE, blank=True, null=True)  question = models.ForeignKey(ClubsQna, models.DO\_NOTHING, blank=True, null=True)  parent\_reply = models.ForeignKey('self', models.DO\_NOTHING, blank=True, null=True, related\_name='reply')  qna\_reply\_content = models.CharField(max\_length=500, blank=True, null=True)  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  is\_deleted = models.IntegerField(default=0)  is\_secret = models.IntegerField(default=0)  class Meta:  managed = False  db\_table = 'qna\_replies'  class RelInterestClub(models.Model):  interest\_club\_id = models.AutoField(primary\_key=True)  club = models.ForeignKey(Clubs, on\_delete=models.CASCADE, null=True, related\_name = 'like\_user')  user = models.ForeignKey(User, on\_delete=models.CASCADE , null=True, related\_name='interest\_club')  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  class Meta:  managed = False  db\_table = 'rel\_interest\_club'  class SocialaccountSocialaccount(models.Model):  provider = models.CharField(max\_length=30)  uid = models.CharField(max\_length=191)  last\_login = models.DateTimeField()  date\_joined = models.DateTimeField(auto\_now\_add=True)  extra\_data = models.TextField()  user = models.ForeignKey(User, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'socialaccount\_socialaccount'  unique\_together = (('provider', 'uid'),)  class SocialaccountSocialapp(models.Model):  provider = models.CharField(max\_length=30)  name = models.CharField(max\_length=40)  client\_id = models.CharField(max\_length=191)  secret = models.CharField(max\_length=191)  key = models.CharField(max\_length=191)  class Meta:  managed = False  db\_table = 'socialaccount\_socialapp'  class SocialaccountSocialappSites(models.Model):  socialapp = models.ForeignKey(SocialaccountSocialapp, models.DO\_NOTHING)  site = models.ForeignKey(DjangoSite, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'socialaccount\_socialapp\_sites'  unique\_together = (('socialapp', 'site'),)  class SocialaccountSocialtoken(models.Model):  token = models.TextField()  token\_secret = models.TextField()  expires\_at = models.DateTimeField(blank=True, null=True)  account = models.ForeignKey(SocialaccountSocialaccount, models.DO\_NOTHING)  app = models.ForeignKey(SocialaccountSocialapp, models.DO\_NOTHING)  class Meta:  managed = False  db\_table = 'socialaccount\_socialtoken'  unique\_together = (('app', 'account'),)  class UsersAdditionalInfo(models.Model):  user\_info = models.OneToOneField(User, models.DO\_NOTHING, primary\_key=True)  profile = models.ImageField(upload\_to="%Y/%m/%d", null=True)  name = models.CharField(max\_length = 45)  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)  class Meta:  managed = False  db\_table = 'users\_additional\_info'  class PostsLike(models.Model):  posts\_like\_id = models.AutoField(primary\_key=True)  user = models.ForeignKey(User, on\_delete=models.CASCADE)  posts = models.ForeignKey(Posts, on\_delete=models.CASCADE, related\_name='like')  created\_at = models.DateTimeField(auto\_now\_add=True)  updated\_at = models.DateTimeField(auto\_now=True)    class Meta:  managed = False  db\_table = 'posts\_like'  @receiver(signals.post\_save, sender=SocialAccount)  def create\_addtional\_user\_info(sender, instance, created, \*\*kwargs):  if created:  user = instance.user  if instance.provider == "google":  UsersAdditionalInfo.objects.create(user\_info= user, profile= instance.get\_avatar\_url(), name = user.last\_name + user.first\_name)  else:  UsersAdditionalInfo.objects.create(user\_info=instance.user, profile= instance.get\_avatar\_url(), name = user.username) |