

Проверил:
Гапанюк Ю.Е.

"__" _____ 2019 г.

**Отчет по лабораторной работе № 5 по курсу
“Разработка интернет-приложений”
«Обработка данных с использованием Django ORM»**

8
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы **ИУ5-54**

(подпись)

Меркулова Н. А.

"__" _____ 2019 г.

1. Задание и порядок выполнения

В этой лабораторной работе вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также вам нужно будет дополнить свои классы предметной области, связав их с созданной базой. После этого вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей и ClassBasedViews.

Для сдачи вы должны иметь:

- Скрипт с подключением к БД и несколькими запросами.
- Набор классов вашей предметной области с привязкой к СУБД (класс должен уметь хотя бы получать нужные записи из БД и преобразовывать их в объекты этого класса)
- Модели вашей предметной области
- View для отображения списка ваших сущностей

2. Описание проекта

В качестве темы проекта был выбран форум с вопросами (техническое задание взято отсюда https://github.com/ziontab/tp-tasks/blob/master/files/markdown/technical_details.md).

3. Исходный код

3.1. Файл models.py

Основные сущности:

- Пользователь – электронная почта, никнейм, пароль, аватарка, дата регистрации, рейтинг.
- Вопрос – заголовок, содержание, автор, дата создания, теги, рейтинг.
- Ответ – содержание, автор, дата написания, флаг правильного ответа, рейтинг.
- Тег – слово тега.

Также для связей сущностей были добавлены дополнительные таблицы.

```
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    avatar = models.ImageField()
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.user.username

class Question(models.Model):
    title = models.CharField(max_length=32)
    text = models.TextField()
    author = models.ForeignKey(Profile, on_delete=models.CASCADE)
    creation_time = models.DateTimeField(auto_now_add=True)
    votes = models.IntegerField(default=0)

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

```

class Answer(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    text = models.TextField()
    author = models.ForeignKey(Profile, on_delete=models.CASCADE)
    creation_time = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.text

class Tag(models.Model):
    name = models.CharField(max_length=16)
    questions = models.ManyToManyField(Question)

    def __str__(self):
        return self.name

class QuestionVote(models.Model):
    value = models.IntegerField(default=0)
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    author = models.ForeignKey(Profile, on_delete=models.CASCADE)

class AnswerVote(models.Model):
    value = models.IntegerField(default=0)
    answer = models.ForeignKey(Answer, on_delete=models.CASCADE)
    author = models.ForeignKey(Profile, on_delete=models.CASCADE)

```

3.2. Файл model_manager.py

Менеджер для работы с моделями.

```

def new_questions(request):
    return paginator.paginate(Question.objects.all().order_by("-creation_time"), 3, request)

def hot_questions(request):
    return paginator.paginate(Question.objects.all().order_by("-votes"), 3, request)

def popular_tags(request):
    tags = Tag.objects.annotate(questions_count=Count("questions")).order_by("-questions_count")[:3]
    return tags

def best_members(request):
    members = Profile.objects.order_by("-votes")[:3]
    return members

def question_by_id(question_id, request):
    question = Question.objects.get(id=question_id)
    return question

def questions_by_tag(tag, request):
    tag = Tag.objects.get(name=tag)
    return tag.questions.all()

def answers_by_questions(question, request):
    answers = Answer.objects.all() #Answer.objects.get(question=question).all()
    return paginator.paginate(answers.order_by("-answervote"), 3, request)

```

3.3. Файл views.py

View для отображения списка сущностей

```
def login(request):
    redirect = request.GET.get('continue', '/')
    if request.user.is_authenticated:
        return HttpResponseRedirect(redirect)
    if request.method == "POST":
        form = LoginForm(request.POST)
        if form.is_valid():
            auth.login(request, form.cleaned_data['user'])
            return HttpResponseRedirect(redirect)
    else:
        form = LoginForm()

    return render(request, 'login.html', {
        'form': form,
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request)
    })

@login_required
def logout(request):
    redirect = request.GET.get('continue', '/')
    auth.logout(request)

    return HttpResponseRedirect(redirect)

def signup(request):
    context = {
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    if request.user.is_authenticated:
        return HttpResponseRedirect('/')
    if request.method == "POST":
        form = SignupForm(request.POST, request.FILES)
        if form.is_valid():
            user = form.save()
            auth.login(request, user)
            return HttpResponseRedirect('/')
    else:
        form = SignupForm()

    return render(request, 'signup.html', {'form': form, 'popular_tags': model_manager.popular_tags(request),
                                           'best_members': model_manager.best_members(request)})
```

```

def tag(request, tag):
    title = "Tag: %s" % tag

    context = {
        'title': title,
        'questions_to_show': model_manager.questions_by_tag(tag, request),
        'questions_switcher': {
            'title': "",
            'href': "",
        },
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    return render(request, 'index.html', context)

def question(request, question_id):
    question = model_manager.question_by_id(question_id, request)

    context = {
        'question': question,
        'answers': model_manager.answers_by_questions(question, request),
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    return render(request, 'question.html', context)

```

```

def ask(request):
    context = {
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    return render(request, 'ask.html', context)

@login_required
def settings(request):
    context = {
        'title': "New Questions",
        'questions_to_show': model_manager.new_questions(request),
        'questions_switcher': {
            'title': "Hot Questions",
            'href': "/hot",
        },
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    return render(request, 'settings.html', context)

```

```

class Command(BaseCommand):
    USERS_COUNT = 3
    TAGS_COUNT = 3
    QUESTIONS_COUNT = 20
    ANSWERS_COUNT = 3
    QUESTION_VOTES_COUNT = 50
    MAX_TAGS_COUNT_FOR_ONE_QUESTIONS = 3

    fake = Faker()

    def ProfilesGenerator(self):
        for _ in range(0, self.USERS_COUNT):
            u = User.objects.create(username=self.fake.name())
            u.save()
            p = Profile.objects.create(user=u)
            p.save()

    def QuestionsGenerator(self):
        profiles = Profile.objects.all()
        for _ in range(self.QUESTIONS_COUNT):
            q = Question()
            q.author = choice(profiles)
            q.title = self.fake.text(16)
            q.text = self.fake.text(128)
            q.save()

    def TagsGenerator(self):
        questions = Question.objects.all()
        for _ in range(self.TAGS_COUNT):
            t = Tag.objects.create(name=self.fake.color_name())
            for _ in range(self.QUESTIONS_COUNT):
                t.questions.add(choice(questions))
            t.save()

    def AnswersGenerator(self):
        profiles = Profile.objects.all()
        questions = Question.objects.all()
        for _ in range(self.ANSWERS_COUNT):
            a = Answer()
            a.author = choice(profiles)
            a.question = choice(questions)
            a.text = self.fake.text(128)
            a.save()

```

```

def QuestionVotesGenerator(self):
    profiles = Profile.objects.all()
    questions = Question.objects.all()
    for question in questions:
        for _ in range(self.USERS_COUNT):
            v = QuestionVote()
            v.author = choice(profiles)
            v.question = question
            v.value = choice([-1, 1])
            v.save()

def Generator(self):
    self.ProfilesGenerator()
    self.QuestionsGenerator()
    self.TagsGenerator()
    self.AnswersGenerator()
    self.QuestionVotesGenerator()

def handle(self, *args, **options):
    self.Generator()

```

3.4. Файл do_faker.py

Для заполнения базы «фейковыми» данными.

```
def index(request):
    context = {
        'title': "New Questions",
        'questions_to_show': model_manager.new_questions(request),
        'questions_switcher': {
            'title': "Hot Questions",
            'href': "/hot",
        },
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    return render(request, 'index.html', context)

def hot(request):
    context = {
        'title': "Hot Questions",
        'questions_to_show': model_manager.hot_questions(request),
        'questions_switcher': {
            'title': "New Questions",
            'href': "/",
        },
        'popular_tags': model_manager.popular_tags(request),
        'best_members': model_manager.best_members(request),
    }

    return render(request, 'index.html', context)
```

4. Скриншоты

Ask NMERK

ask me a question


Ask

Login

Sign Up

New Questions

Hot Questions




Until political.

by Denise Banks

Watch report site push food room pretty. Establish prove car seat common stuff within hot. Form blood soon so.

Tags: MediumAquaMarine

0 upvotes




North trip the.

by Todd Dalton

Friend hair its true police. Mr computer simple article move still table talk.

Tags: AntiqueWhite

0 upvotes



Mr what require.

by Todd Dalton

Yard impact person true prevent business country. Allow woman relate evidence. Store art billion successful.

Tags: AntiqueWhite

0 upvotes

Popular Tags

AntiqueWhite

GhostWhite

MediumAquaMarine

Best Members

Todd Dalton

Steven Aguirre

Kelly Munoz

Page 1 of 18

Next

Last

Django administration

Site administration

ASK

Answers

[+ Add](#)[Change](#)

Profiles

[+ Add](#)[Change](#)

Question votes

[+ Add](#)[Change](#)

Questions

[+ Add](#)[Change](#)

Tags

[+ Add](#)[Change](#)

AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#)[Change](#)

Users

[+ Add](#)[Change](#)