Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



**Отчет**

**Лабораторная работа № 5**

**По курсу «Разработка интернет приложений»**

**«Обработка данных с использованием Django ORM»**

**ИСПОЛНИТЕЛЬ:**

Эдилов Адам

Группа ИУ5-52

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2019 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2019 г.

Москва 2019

1. **Задание лабораторной работы**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:
2. Скрипт с подключением к БД и несколькими запросами.
3. 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)
4. Модели вашей предметной области
5. View для отображения списка ваших сущностей
6. **Модели  
   class** AuthorManager(models.Manager):  
    **def** best(self):  
    **return** self.order\_by(**'-rating'**)[:3]  
     
     
   **class** Author(models.Model):  
    user = models.OneToOneField(settings.AUTH\_USER\_MODEL,  
    on\_delete=models.CASCADE)  
    rating = models.IntegerField(default=0)  
    avatar = models.ImageField(upload\_to=**'images/avatars/'**,  
    default=**'images/avatars/default\_avatar.svg'**)  
     
    **def** \_\_str\_\_(self):  
    **return** self.user.username  
     
    objects = AuthorManager()  
     
     
   **class** QuestionManager(models.Manager):  
    **def** recent(self, page\_number, limit):  
    **return** paginator.paginate(  
    self.order\_by(**'-creation\_time'**).annotate(num\_of\_answers=Count(**'answer'**)), limit, page\_number)  
     
    **def** hot(self, page\_number, limit):  
    **return** paginator.paginate(  
    self.order\_by(**'-rating'**).annotate(num\_of\_answers=Count(**'answer'**)), limit, page\_number)  
     
    **def** by\_id(self, question\_id):  
    **return** self.annotate(num\_of\_answers=Count(**'answer'**)).get(pk=question\_id)  
     
    **def** questions\_by\_tag(self, tag\_name, page\_number, limit):  
    t = Tag.objects.get(name=tag\_name)  
    **return** paginator.paginate(self.filter(tag=t), limit, page\_number)  
     
     
   **class** Question(models.Model):  
    author = models.ForeignKey(**'Author'**, on\_delete=models.CASCADE)  
    title = models.CharField(max\_length=128)  
    content = models.TextField(null=**False**)  
    rating = models.IntegerField(default=0)  
    creation\_time = models.DateTimeField(default=timezone.now)  
     
    **def** \_\_str\_\_(self):  
    **return** self.title  
     
    objects = QuestionManager()  
     
     
   **class** Answer(models.Model):  
    author = models.ForeignKey(**'Author'**, on\_delete=models.CASCADE)  
    question = models.ForeignKey(**'Question'**, on\_delete=models.CASCADE)  
    rating = models.IntegerField(default=0)  
    content = models.TextField()  
    creation\_time = models.DateTimeField(default=timezone.now)  
     
    **def** \_\_str\_\_(self):  
    **return "Answer to question: {}"**.format(self.question.title)  
     
     
   **class** TagManager(models.Manager):  
    **def** popular(self):  
    **return** self.annotate(question\_count=Count(**'question'**)).order\_by(**'-question\_count'**)[:3]  
     
     
   **class** Tag(models.Model):  
    question = models.ManyToManyField(Question)  
    name = models.CharField(max\_length=32, unique=**True**)  
     
    **def** \_\_str\_\_(self):  
    **return** self.name  
     
    objects = TagManager()  
     
     
   **class** QuestionVote(models.Model):  
    value = models.SmallIntegerField(default=0)  
    question = models.ForeignKey(Question, on\_delete=models.CASCADE)  
    author = models.ForeignKey(Author, on\_delete=models.CASCADE)  
     
    **def** \_\_str\_\_(self):  
    **return "{0} votes to question: {1}"**.format(self.author,  
    self.question.author)  
     
     
   **class** AnswerVote(models.Model):  
    value = models.SmallIntegerField(default=0)  
    answer = models.ForeignKey(Answer, on\_delete=models.CASCADE)  
    author = models.ForeignKey(Author, on\_delete=models.CASCADE)  
     
    **def** \_\_str\_\_(self):  
    **return "{0} votes to answer: {1}"**.format(self.author,  
    self.answer)
7. **View для отображения моделей  
   class** QuestionView(View):  
    **def** get(self, request, question\_id, \*\*kwargs):  
    q = Question.objects.by\_id(question\_id)  
    page\_number = request.GET.get(**'page'**, 1)  
    limit = request.GET.get(**'limit'**, 5)  
    form = kwargs.get(**'form'**, AnswerForm(request.user, question\_id))  
    context = {  
    **'question'**: q,  
    **'answers'**: paginate(q.answer\_set.all(), limit, page\_number),  
    **'form'**: form  
    }  
    **return** render(request, **'question.html'**, context)