САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Web-программирование

Отчет

Лабораторная работа №2

Выполнил:
Золотов Павел

Группа:
К33401

Проверил:
Говоров А. И.

Санкт-Петербург

2021 г.

Задача

Необходимо реализовать следующий функционал:

- Регистрация новых пользователей.
- Просмотр домашних заданий по всем дисциплинам (сроки выполнения, описание задания).
- Сдача домашних заданий в текстовом виде.
- Администратор (учитель) должен иметь возможность поставить оценку за задание средствами Django-admin.
- В клиентской части должна формироваться таблица, отображающая оценки всех учеников класса.

Ход работы

Модели:

1) Учебная группа (GroupModel)
   Содержит название группы
2) Пользователь (CustomUser)
   Содержит связь один ко многим с GroupModel
3) Преподаватель (TeacherModel)
   Содержит ФИО преподавателя
4) Предмет (SubjectModel)
   Содержит название дисциплины и связь один ко многим с TeacherModel
5) Домашнее задание (HometaskModel)
   Содержит связь один ко многим с SubjectModel, дату публикации и срок сдачи, описание и штраф (положительное число)
6) Сдача задания (SubmissionModel)
   Содержит связь один ко многим с HometaskModel, связь один ко многим с CustomUser, текст сдачи и оценку (положительное число)

Содержимое файла models.py:

```
class GroupModel(models.Model):
    title = models.CharField(max_length=20)

    def __str__(self):
        return self.title


class CustomUser(AbstractUser):
    group = models.ForeignKey(GroupModel, on_delete=models.CASCADE,
blank=True, null=True)


class TeacherModel(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name


class SubjectModel(models.Model):
    title = models.CharField(max_length=100)
    teacher = models.ForeignKey(TeacherModel, on_delete=models.CASCADE)

    def __str__(self):
        return self.title


class HometaskModel(models.Model):
    subject = models.ForeignKey(SubjectModel, on_delete=models.CASCADE)
    published = models.DateTimeField()
    deadline = models.DateTimeField()
    description = models.TextField()
    penalty = models.PositiveIntegerField(default=0)

    def __str__(self):
        return "{}: {}".format(self.subject, self.description)


class SubmissionModel(models.Model):
    hometask = models.ForeignKey(HometaskModel, on_delete=models.CASCADE)
    user = models.ForeignKey(get_user_model(), on_delete=models.CASCADE)
    text = models.TextField()
    grade = models.PositiveIntegerField(blank=True, null=True)

    class Meta:
        unique_together = ('hometask', 'user', "grade")
```

Контроллеры:

1) Список всех заданий

```
class HometaskList(ListView):
    model = HometaskModel
    template_name = 'hometask_list_view.html'
```

2) Форма сдачи домашнего задания

```
@login_required
def submission(request):
    context = {}
    form = SubmissionForm(request.POST or None)
    if form.is_valid():
        data = form.save(commit=False)
        data.user = request.user
        data.user = request.user
        try:
            data.save()
            form = SubmissionForm()  # clean the form
        except IntegrityError:
            messages.error(request, "You can't submit task again")
    context['form'] = form
    return render(request, "hometask_submission.html", context)
```

3) Список всех групп

```
class GroupList(ListView):
    model = GroupModel
    template_name = 'group_list_view.html'
```

4) Таблица оценок

```
class GradesTableView(SingleTableView):
    model = SubmissionModel
    table_class = GradesTable
    queryset = SubmissionModel.objects.all()
    template_name = 'grades_table_view.html'

    def get_queryset(self):
        group = self.kwargs['group']
        if group:
            try:
                group = int(group)
                queryset = self.queryset.filter(user__group=group)
            except ValueError:
                queryset = self.model.objects.none()
            return queryset
        return self.queryset

    def get_context_data(self, *args, **kwargs):
        context = super().get_context_data(**kwargs)
        group = self.kwargs['group']
        context["group"] = GroupModel.objects.get(id=group)
        return context
```

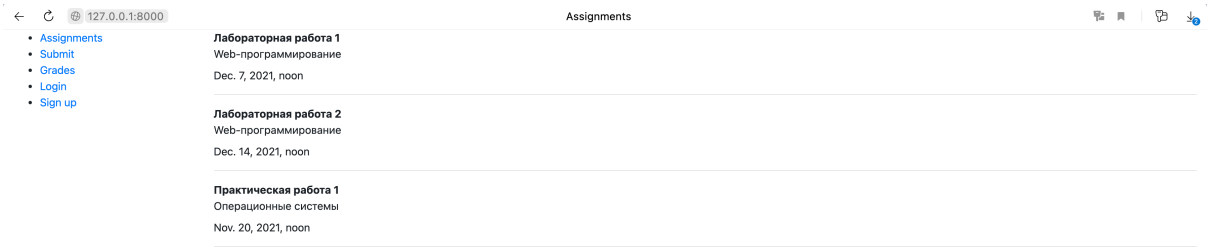
Роутеры

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
    path("register/", reg_views.register, name="register"),
    path('', HometaskList.as_view(), name="home"),
    path('submit/', submission, name="submit"),
    path('grades/', GroupList.as_view(), name="groups"),
    path('grades/<int:group>', GradesTableView.as_view(), name="grades"),
]
```

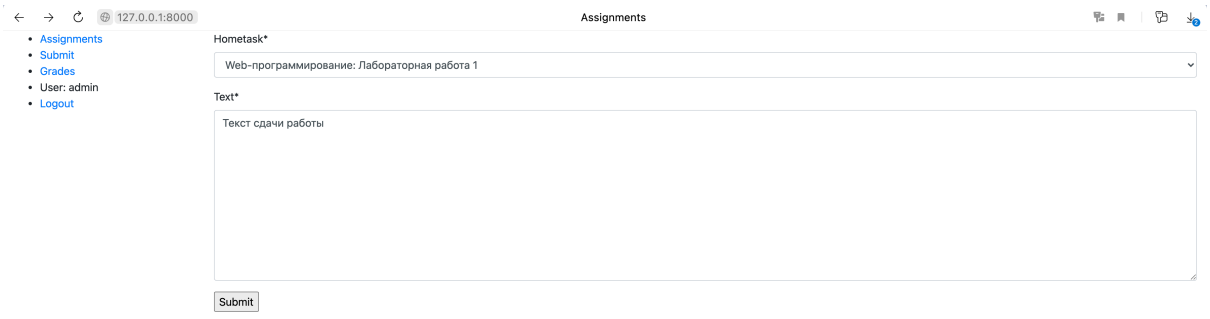
Для обработки ссылок на авторизацию используется django.contrib.auth.urls

Скриншоты работы программы

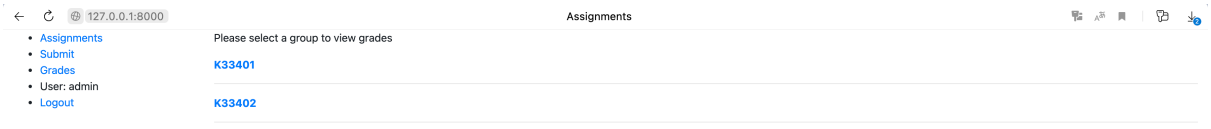
Главная страница со списком заданий:



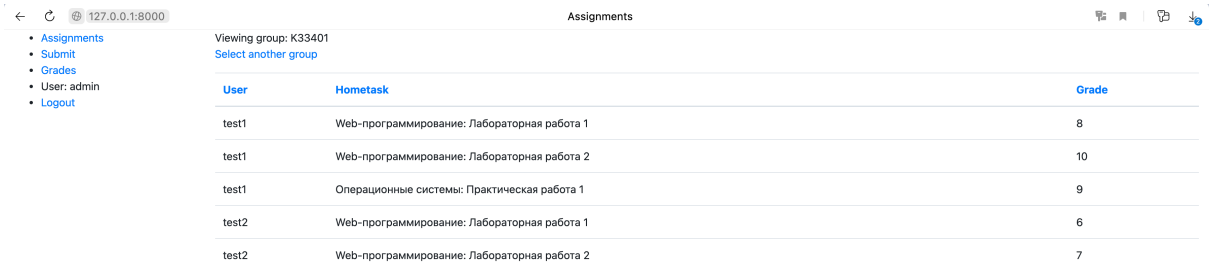
Страница сдачи работы:



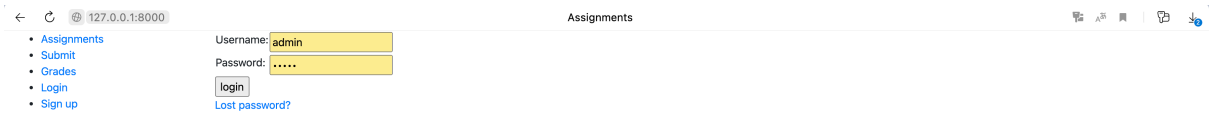
Выбор группы:



Просмотр оценок группы:

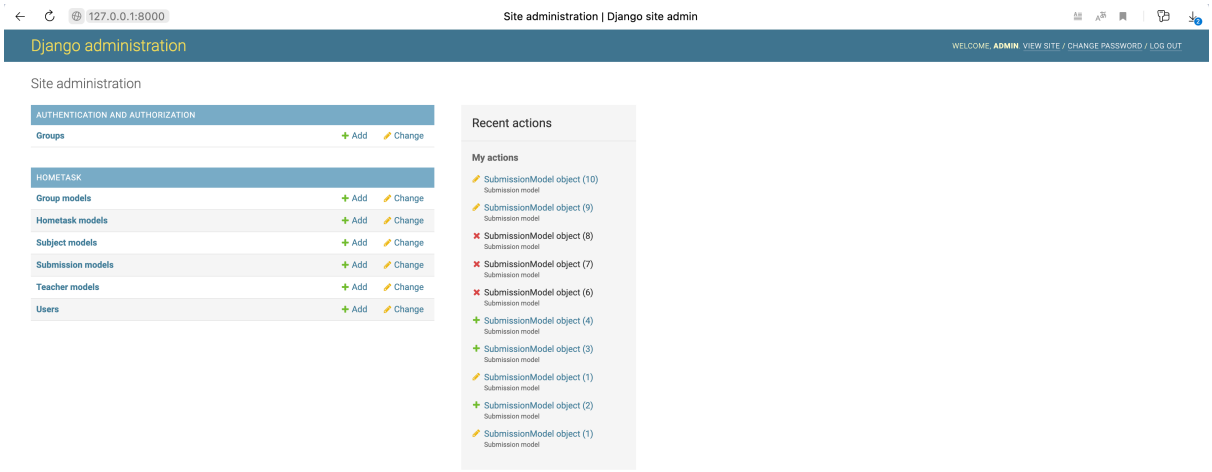


Вход:

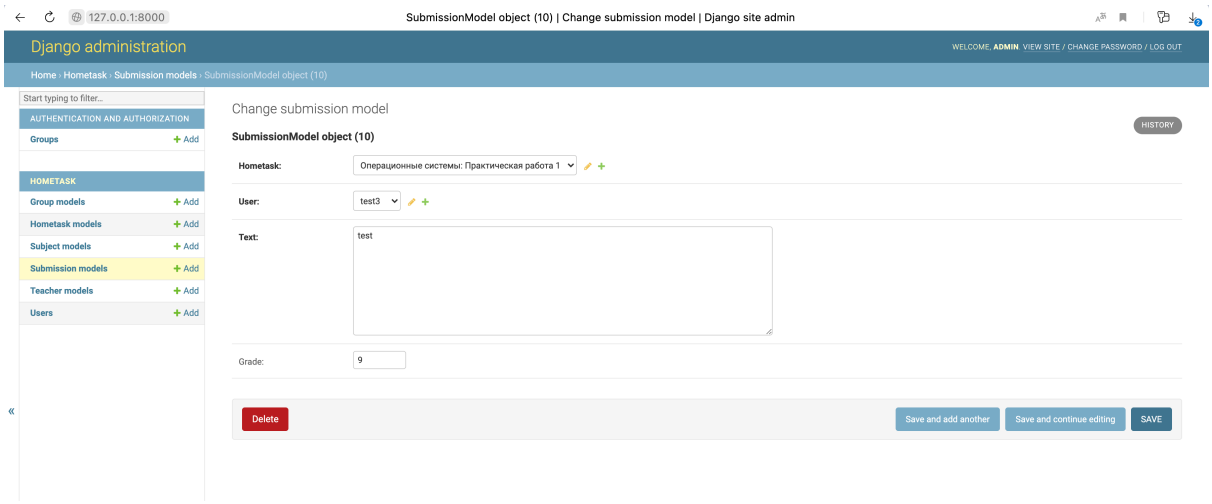


Регистрация:

Адмим-панель:



Выставление оценки за сданную работу:



Вывод

В результате выполнения работы было создано web приложение на базе Django, где были реализованы все основные сценарии взаимодействия с пользователем – вход, регистрация, просмотр заданий, сдача заданий, просмотр оценок и выставление оценок через админ панель.