Отчет по РК № 2 по курсу
"Разработка Интернет-Приложений"

Выполнил:
Студент группы
ИУ5-55Б
Ширшов А.С.

Москва, МГТУ – 2021

Задание:

Рубежный контроль представляет собой разработку веб-приложения с использованием фреймворка Django. Веб-приложение должно выполнять следующие функции:

1. Создайте проект Python Django с использованием стандартных средств Django.
2. Создайте модель Django ORM, содержащую две сущности, связанные отношением один-ко-многим в соответствии с Вашим вариантом из условий рубежного контроля №1.
3. С использованием стандартного механизма Django сгенерируйте по модели макет веб-приложения, позволяющий добавлять, редактировать и удалять данные.
4. Создайте представление и шаблон, формирующий отчет, который содержит соединение данных из двух таблиц.

№ варианта	Класс 1	Класс 2
19	Деталь	Производитель

Текст программы:

models.py

```python
from django.db import models

# Create your models here.

class Manufacturer(models.Model):
    name = models.CharField(max_length=256, verbose_name="Manufacturer name")

    def __str__ (self):
        return self.name

class Part (models.Model):
    name = models.CharField(max_length=256, verbose_name="Part name")
    cost = models.PositiveIntegerField(verbose_name="Part cost")
    manufacturer = models.ForeignKey(
        Manufacturer,
        on_delete=models.SET_DEFAULT,
        null=True,
        default=None,
        related_name="parts"
    )

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

views.py

```python
def delete_manufacturer(request, manufacturer_id):
    manufacturer = get_object_or_404(Manufacturer, pk=manufacturer_id)
    manufacturer.delete()
    return redirect('read_manufacturer')

def report(request):
    expensive_parts = Part.objects.filter(cost__gt=1000)
    avg_prices = []
    for manufacturer in Manufacturer.objects.all():
        avg_prices.append({"manufacturer": manufacturer,  "price": Part.objects.filter(
            manufacturer=manufacturer.pk).aggregate(Avg('cost'))['cost__avg']})
    return render(request, 'report.html', {"expensive_parts": expensive_parts, "avg_prices": avg_prices})
```

```python
def delete_part(request, part_id):
    part = get_object_or_404(Part, pk=part_id)
    part.delete()
    return redirect ('read_part')

def read_manufacturer(request):
    manufacturers = Manufacturer.objects.all()
    print(Manufacturer.objects.all())
    return render(request, 'manufacturer/manufacturer_list.html', {"manufacturers": manufacturers})


def create_manufacturer(request):
    if request.method == 'GET':
        return render(request, 'manufacturer/create_manufacturer.html')
    else:
        new_manufacturer = Manufacturer(name=request.POST['name'])
        new_manufacturer.save()
        return redirect('read_manufacturer')


def update_manufacturer(request, manufacturer_id):
    if request.method == 'GET':
        manufacturer = get_object_or_404(Manufacturer, pk=manufacturer_id)
        return render(request, 'manufacturer/update_manufacturer.html', {"manufacturer": manufacturer})
    else:
        manufacturer = get_object_or_404(Manufacturer, pk=manufacturer_id)
        for key in request.POST:
            if key in manufacturer.__dict__:
                setattr(manufacturer, key, request.POST[key])
        manufacturer.save()
        return redirect('read_manufacturer')
```

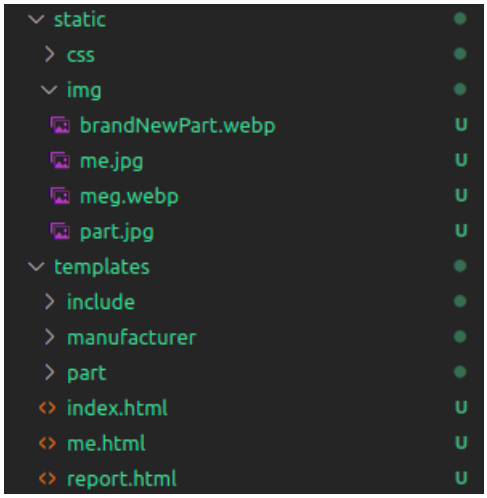
```python
def index(request):
    return render(request, 'index.html')

def me(request):
    return render(request, 'me.html')

def read_part(request):
    parts = Part.objects.all()
    return render(request, 'part/part_list.html',{"parts":parts})

def create_part(request):
    if request.method == 'GET':
        manufacturers = Manufacturer.objects.all()
        return render(request, 'part/create_part.html',{"manufacturers":manufacturers})
    else:
        data = {}
        for key in request.POST:
            if key in Part.__dict__:
                data[key] = request.POST[key]
        data['manufacturer'] = get_object_or_404(Manufacturer, pk=request.POST['manufacturer'])
        new_part = Part(**data)
        new_part.save()
        return redirect ('read_part')

def update_part(request, part_id):
    if request.method == 'GET':
        manufacturers = Manufacturer.objects.all()
        part = get_object_or_404(Part, pk=part_id)
        return render(request, 'part/update_part.html', {"part": part, "manufacturers": manufacturers})
    else:
        part = get_object_or_404(Part, pk=part_id)
        for key in request.POST:
            if key in part.__dict__ and key != 'manufacturer':
                setattr(part, key, request.POST[key])
        if 'manufacturer' in request.POST:
            setattr(part, 'manufacturer', get_object_or_404(
                Manufacturer, pk=request.POST['manufacturer']))
        part.save()
        return redirect('read_part')
```

urls.py

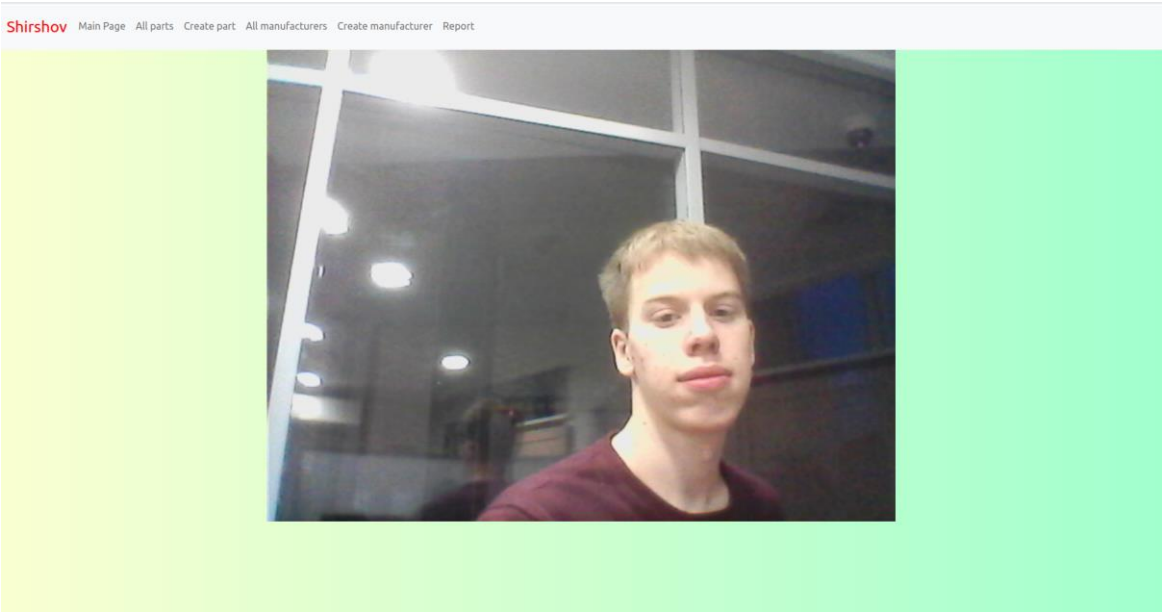
```python
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index, name="main"),
    path('me/', views.me, name="me"),
    path('part/', include([
        path('', views.read_part, name='read_part'),
        path('create/', views.create_part, name="create_part"),
        path('update/<int:part_id>/', views.update_part, name="update_part"),
        path('delete/<int:part_id>/', views.delete_part, name="delete_part"),
    ])),
    path('manufacturer/', include([
        path('', views.read_manufacturer, name='read_manufacturer'),
        path('create/', views.create_manufacturer, name="create_manufacturer"),
        path('update/<int:manufacturer_id>/',
             views.update_manufacturer, name="update_manufacturer"),
        path('delete/<int:manufacturer_id>/',
             views.delete_manufacturer, name="delete_manufacturer"),
    ])),
    path('report/', views.report, name="report"),
]
```

Для более быстрого и красивого дизайна используем Bootstrap

Статика и используемые шаблоны:



Примёр программы:



All Manufacturers

Meg

Обычный поставщик необходимых деталей

Delete Update

Jake

Обычный поставщик необходимых деталей

Delete Update

New Manufacturer

Manufacturer Name

Create

All Parts

Brand New Part

Meg

Очень полезная деталь

1200

Delete Update

Heavy Toolbox

Jake

Очень полезная деталь

800

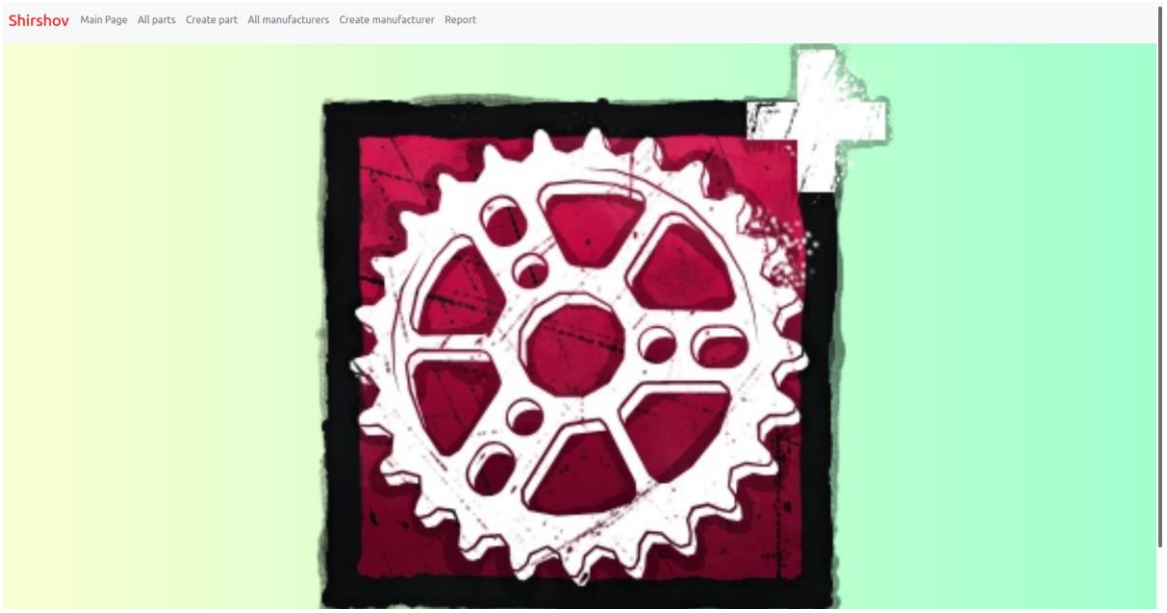
Delete Update

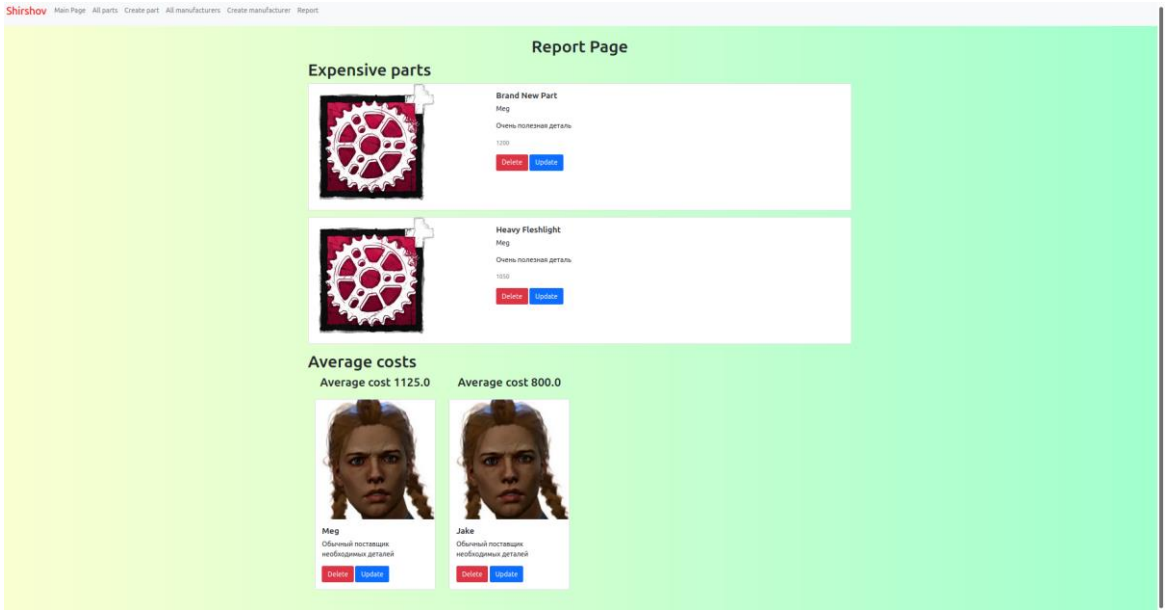
Heavy Fleshlight

Meg

Очень полезная деталь

1050


Shirshov
Main Page
All parts
Create part
All manufacturers
Create manufacturer
Report


Shirshov
Main Page
All parts
Create part
All manufacturers
Create manufacturer
Report
Report Page
Expensive parts
Brand New Part
Meg
Очень полезная деталь
1200
Delete
Update
Heavy Fleshlight
Meg
Очень полезная деталь
1650
Delete
Update
Average costs
Average cost 1125.0
Average cost 800.0
Meg
Обычный поставщик необходимых деталей
Delete
Update
Jake
Обычный поставщик необходимых деталей
Delete
Update

New part

Part Name

Part Cost

Select Manufacturer

Meg

Create