Содержание

[1. Описание предметной области 3](#_Toc501729847)

[2. Задание 3](#_Toc501729848)

[3. Описание структуры проекта 3](#_Toc501729849)

[Приложение 1. Исходный код 4](#_Toc501729850)

[Admin.py 4](#_Toc501729851)

[Forms.py 4](#_Toc501729852)

[Models.py 5](#_Toc501729853)

[Urls.py 6](#_Toc501729854)

[Views.py 7](#_Toc501729855)

[Приложение 2. Примеры шаблонов страниц приложения 11](#_Toc501729856)

1. Описание предметной области

Django — бесплатный и свободный фреймворк для веб-приложений, написанный на Python. Фреймворк — это набор компонентов, которые помогают разрабатывать веб-сайты быстро и просто.

База данных - набор сведений, хранящихся некоторым упорядоченным способом. Можно сравнить базу данных со шкафом, в котором хранятся документы. Иными словами, база данных - это хранилище данных. Сами по себе базы данных не представляли бы интереса, если бы не было систем управления базами данных (СУБД).

Система управления базами данных – это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД это система, позволяющая создавать базы данных и манипулировать сведениями из них. А осуществляет этот доступ к данным СУБД посредством специального языка - SQL. SQL (structured query language) - язык структурированных запросов, основной задачей которого является предоставление простого способа считывания и записи информации в базу данных.

Логическая модель данных является начальным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных не обязательно должна быть выражена средствами именно реляционной модели данных.

Физическая модель данных описывает реализацию объектов логической модели на уровне объектов конкретной базы данных.

1. Задание

Разработать веб-приложение с использованием фреймворка Django, реализующее предметную область "Компьютерное оборудование", позволяющее выполнять ввод, вывод, изменение, удаление данных в таблицах

1. Описание структуры проекта

admin.py: для того чтобы в /admin Django объекты моделей можно было редактировать, их нужно сначала зарегистрировать в admin.py.

forms.py: описывает форму объекта класса, как она работает и показывается пользователю.

model.py: Модели отображают информацию о данных, с которыми вы работаете. Они содержат поля и поведение ваших данных. Обычно одна модель представляет одну таблицу в базе данных. Каждая модель это класс унаследованный от django.db.models.Model. Атрибут модели представляет поле в базе данных. Django предоставляет автоматически созданное API для доступка к данных.

urls.py: конфигурирует URL для вашего проекта Django. Это “содержание” всех Django -сайтов. При запросе к странице проекта, Django определяет корневой модуль URLconf.

После этого загружает модуль конфигурации URL и ищет переменную urlpatterns. Django перебирает каждый URL-шаблон по порядку, и останавливается при первом совпадении с запрошенным URL-ом. Если одно из регулярных выражений соответствует URL-у, Django импортирует и вызывает соответствующее представление, которое является просто функцией Python.

view.py: это функции Python, которые принимают Web-запрос и возвращает Web-ответ. Представление содержит всю необходимую логику для создания ответа.

Приложение 1. Исходный код

Admin.py

from django.contrib import admin

from database.models import \*

# Register your models here.

admin.site.register(Firm)

admin.site.register(Seller)

admin.site.register(Equipment\_type)

admin.site.register(Equipment)

admin.site.register(Provider)

Forms.py

from django.forms import ModelForm

from database.models import \*

class Provider\_Form(ModelForm):

class Meta:

fields = [ "provider\_name", "provider\_country", "provider\_number", "provider\_address"]

labels = {

"provider\_name": "Наименование поставщика",

"provider\_country": "Страна поставщика",

"provider\_number": "Номер поставщика",

"provider\_address": "Адрес поставщика"

}

class Firm\_Form(ModelForm):

class Meta:

model = Firm

fields = [ "firm\_name", "firm\_address", "firm\_country", "firm\_number", "firm\_site"]

labels = {

"firm\_name": "Наименование фирмы",

"firm\_country": "Страна фирмы",

"firm\_number": "Номер фирмы",

"firm\_address": "Адрес фирмы",

"firm\_site": "Сайт фирмы"

}

class Seller\_Form(ModelForm):

class Meta:

model = Seller

fields = ["seller\_name", "seller\_exp", "seller\_address"]

labels = {

"seller\_name": "ФИО продавца",

"seller\_exp": "Опыт работы",

"seller\_address": "Адрес продавца",

}

class Type\_Form(ModelForm):

class Meta:

model = Equipment\_type

fields = ["et\_name"]

labels = {

"et\_name": "Наименование типа",

}

class Equipment\_Form(ModelForm):

class Meta:

model = Equipment

fields = ["equipment\_name", "equipment\_price", "equipment\_size", "equipment\_date", "equipment\_country", "id\_equipment\_type", "id\_seller", "id\_provider", "id\_firm"]

labels = {

"equipment\_name": "Оборудование",

"equipment\_price": "Цена",

"equipment\_size": "Количество",

"equipment\_date": "Дата поступления",

"equipment\_country": "Страна",

"id\_equipment\_type": "Тип оборудования",

"id\_seller": "Продавец",

"id\_provider": "Поставщик",

"id\_firm": "Фирма производитель"

}

Models.py

from django.db import models

# Create your models here.

class Provider(models.Model):

provider\_id = models.AutoField(primary\_key=True)

provider\_name = models.CharField(max\_length=50)

provider\_address = models.CharField(max\_length=50)

provider\_country = models.CharField(max\_length=30)

provider\_number = models.CharField(max\_length=50)

class Meta:

verbose\_name = "Поставщик"

verbose\_name\_plural = "Поставщики"

def \_\_str\_\_(self):

return self.provider\_name

class Firm(models.Model):

firm\_id = models.AutoField(primary\_key=True)

firm\_name = models.CharField(max\_length=50)

firm\_address = models.CharField(max\_length=50)

firm\_country = models.CharField(max\_length=30)

firm\_number = models.CharField(max\_length=50)

firm\_site = models.CharField(max\_length=50)

class Meta:

verbose\_name = "Фирма"

verbose\_name\_plural = "Фирмы"

def \_\_str\_\_(self):

return self.firm\_name

class Seller(models.Model):

seller\_id = models.AutoField(primary\_key=True)

seller\_name = models.CharField(max\_length=50)

seller\_address = models.CharField(max\_length=50)

seller\_exp = models.IntegerField()

provider\_number = models.CharField(max\_length=50)

class Meta:

verbose\_name = "Продавец"

verbose\_name\_plural = "Продавцы"

def \_\_str\_\_(self):

return self.seller\_name

class Equipment\_type(models.Model):

et\_id = models.AutoField(primary\_key=True)

et\_name = models.CharField(max\_length=50)

def \_\_str\_\_(self):

return self.et\_name

class Meta:

verbose\_name = "Тип оборудования"

verbose\_name\_plural = "Типы оборудований"

class Equpment\_type\_seller(models.Model):

et\_id\_f = models.ForeignKey(Equipment\_type, models.ProtectedError)

sel\_id = models.ForeignKey(Seller, models.ProtectedError)

class Equipment(models.Model):

equipment\_id = models.AutoField(primary\_key=models.ProtectedError)

equipment\_name = models.CharField(max\_length=50)

equipment\_price = models.IntegerField()

equipment\_size = models.IntegerField(default=0)

equipment\_date = models.DateField()

equipment\_country = models.CharField(max\_length=50)

id\_firm = models.ForeignKey(Firm, on\_delete=models.CASCADE)

id\_provider = models.ForeignKey(Provider, on\_delete=models.CASCADE)

id\_equipment\_type = models.ForeignKey(Equipment\_type, on\_delete=models.CASCADE)

id\_seller = models.ForeignKey(Seller, on\_delete=models.CASCADE)

class Meta:

verbose\_name = "Оборудование"

verbose\_name\_plural = "Оборудование"

def \_\_str\_\_(self):

return self.equipment\_name

Urls.py

from django.conf.urls import url, include

from database import views

urlpatterns = [

url(r"^main\_menu/", views.main\_menu, name="main\_menu"),

# URL providers

url(r"^provider/$", views.show\_provider, name="provider"),

url(r"^edit\_provider/(?P<id\_provider>\d+)$", views.edit\_provider, name="edit\_provider"),

url(r"^delete\_provider/(?P<id\_provider>\d+)$", views.delete\_provider, name="delete\_provider"),

url(r"^add\_provider/$", views.add\_provider, name="add\_provider"),

# URL firms

url(r"^firm/$", views.show\_firm, name="firm"),

url(r"^edit\_firm/(?P<id\_firm>\d+)$", views.edit\_firm, name="edit\_firm"),

url(r"^delete\_firm/(?P<id\_firm>\d+)$", views.delete\_firm, name="delete\_firm"),

url(r"^add\_firm/$", views.add\_firm, name="add\_firm"),

# URL sellers

url(r"^seller/$", views.show\_seller, name="seller"),

url(r"^edit\_seller/(?P<id\_seller>\d+)$", views.edit\_seller, name="edit\_seller"),

url(r"^delete\_seller/(?P<id\_seller>\d+)$", views.delete\_seller, name="delete\_seller"),

url(r"^add\_seller/$", views.add\_seller, name="add\_seller"),

# URL equipment type

url(r"^type/$", views.show\_type, name="type"),

url(r"^edit\_type/(?P<id\_type>\d+)$", views.edit\_type, name="edit\_type"),

url(r"^delete\_type/(?P<id\_type>\d+)$", views.delete\_type, name="delete\_type"),

url(r"^add\_type/$", views.add\_type, name="add\_type"),

# URL equipment

url(r"^equipment/$", views.show\_equipment, name="equipment"),

url(r"^edit\_equipment/(?P<id\_equipment>\d+)$", views.edit\_equipment, name="edit\_equipment"),

url(r"^delete\_equipment/(?P<id\_equipment>\d+)$", views.delete\_equipment, name="delete\_equipment"),

url(r"^add\_equipment/$", views.add\_equipment, name="add\_equipment"),

]

Views.py

from django.shortcuts import render, redirect

from database.forms import \*

from database.models import \*

def main\_menu(request):

return render(request, "main.html", locals())

#PROVIDERS VIEWS

def show\_provider(request):

provider\_list = Provider.objects.all()

return render(request, "provider/show\_provider.html", locals())

def edit\_provider(request, id\_provider):

provider = Provider.objects.get(provider\_id=id\_provider)

form = Provider\_Form(instance=provider)

if request.method == "POST":

form = Provider\_Form(request.POST, instance=provider)

if form.is\_valid():

form.save()

return redirect('provider')

else:

return render(request, "provider/edit\_provider.html", locals())

return render(request, "provider/edit\_provider.html", locals())

def delete\_provider(request, id\_provider):

provider = Provider.objects.get(provider\_id=id\_provider)

if request.method == "POST":

if 'yes' in request.POST.keys() and request.POST['yes']:

provider.delete()

return redirect('provider')

else:

return redirect("provider")

return render(request, 'provider/delete\_provider.html', locals())

def add\_provider(request):

form = Provider\_Form()

if request.method == "POST":

form = Provider\_Form(request.POST)

if form.is\_valid():

form.save()

return redirect('provider')

else:

return render(request, "provider/add\_provider.html", locals())

return render(request, "provider/add\_provider.html", locals())

#FIRMS VIEWS

def show\_firm(request):

firm\_list = Firm.objects.all()

return render(request, "firm/show\_firm.html", locals())

def edit\_firm(request, id\_firm):

firm = Firm.objects.get(firm\_id=id\_firm)

form = Firm\_Form(instance=firm)

if request.method == "POST":

form = Firm\_Form(request.POST, instance=firm)

if form.is\_valid():

form.save()

return redirect('firm')

else:

return render(request, "firm/edit\_firm.html", locals())

return render(request, "firm/edit\_firm.html", locals())

def delete\_firm(request, id\_firm):

firm = Firm.objects.get(firm\_id=id\_firm)

if request.method == "POST":

if 'yes' in request.POST.keys() and request.POST['yes']:

firm.delete()

return redirect("firm")

else:

return redirect("firm")

return render(request, 'firm/delete\_firm.html', locals())

def add\_firm(request):

form = Firm\_Form()

if request.method == "POST":

form = Firm\_Form(request.POST)

if form.is\_valid():

form.save()

return redirect('firm')

else:

return render(request, "firm/add\_firm.html", locals())

return render(request, "firm/add\_firm.html", locals())

#SELLER VIEWS

def show\_seller(request):

seller\_list = Seller.objects.all()

return render(request, "seller/show\_seller.html", locals())

def edit\_seller(request, id\_seller):

seller = Seller.objects.get(seller\_id=id\_seller)

form = Seller\_Form(instance=seller)

if request.method == "POST":

form = Seller\_Form(request.POST, instance=seller)

if form.is\_valid():

form.save()

return redirect('seller')

else:

return render(request, "seller/edit\_seller.html", locals())

return render(request, "seller/edit\_seller.html", locals())

def delete\_seller(request, id\_seller):

seller = Seller.objects.get(seller\_id=id\_seller)

if request.method == "POST":

if 'yes' in request.POST.keys() and request.POST['yes']:

seller.delete()

return redirect("seller")

else:

return redirect("seller")

return render(request, 'seller/delete\_seller.html', locals())

def add\_seller(request):

form = Seller\_Form()

if request.method == "POST":

form = Seller\_Form(request.POST)

if form.is\_valid():

form.save()

return redirect('seller')

else:

return render(request, "seller/add\_seller.html", locals())

return render(request, "seller/add\_seller.html", locals())

#TYPE VIEWS

def show\_type(request):

type\_list = Equipment\_type.objects.all()

return render(request, "type/show\_type.html", locals())

def edit\_type(request, id\_type):

etype = Equipment\_type.objects.get(et\_id=id\_type)

form = Type\_Form(instance=etype)

if request.method == "POST":

form = Type\_Form(request.POST, instance=etype)

if form.is\_valid():

form.save()

return redirect('type')

else:

return render(request, "type/edit\_type.html", locals())

return render(request, "type/edit\_type.html", locals())

def delete\_type(request, id\_type):

etype = Equipment\_type.objects.get(et\_id=id\_type)

if request.method == "POST":

if 'yes' in request.POST.keys() and request.POST['yes']:

etype.delete()

return redirect("type")

else:

return redirect("type")

return render(request, 'type/delete\_type.html', locals())

def add\_type(request):

form = Type\_Form()

if request.method == "POST":

form = Type\_Form(request.POST)

if form.is\_valid():

form.save()

return redirect('type')

else:

return render(request, "type/add\_type.html", locals())

return render(request, "type/add\_type.html", locals())

#EQUIPMENT VIEWS

def show\_equipment(request):

equipment\_list = Equipment.objects.all()

return render(request, "equipment/show\_equipment.html", locals())

def edit\_equipment(request, id\_equipment):

equipment = Equipment.objects.get(equipment\_id=id\_equipment)

form = Equipment\_Form(instance=equipment)

if request.method == "POST":

form = Equipment\_Form(request.POST, instance=equipment)

if form.is\_valid():

form.save()

return redirect('equipment')

else:

return render(request, "equipment/edit\_equipment.html", locals())

return render(request, "equipment/edit\_equipment.html", locals())

def delete\_equipment(request, id\_equipment):

equipment = Equipment.objects.get(equipment\_id=id\_equipment)

if request.method == "POST":

if 'yes' in request.POST.keys() and request.POST['yes']:

equipment.delete()

return redirect("equipment")

else:

return redirect("equipment")

return render(request, 'equipment/delete\_equipment.html', locals())

def add\_equipment(request):

form = Equipment\_Form()

if request.method == "POST":

form = Equipment\_Form(request.POST)

if form.is\_valid():

form.save()

return redirect('equipment')

else:

return render(request, "equipment/add\_equipment.html", locals())

return render(request, "equipment/add\_equipment.html", locals())

Приложение 2. Примеры шаблонов страниц приложения











