**Задание 7. Сохранение информации в базе данных**

**Задание:**

К проекту ИС для торговой компании (созданному в предыдущем задании) добавить следующее:

Добавить форму для ввода данных в таблицы. Добавить функцию, позволяющую сохранить данные с формы.

В каждую из таблиц добавить по 2 записи.

**Пример:**

**1. Создадим форму на основе моделей**: продукт, поставщик, склад

В папке приложения создадим файл forms.py и опишем в нем формы.

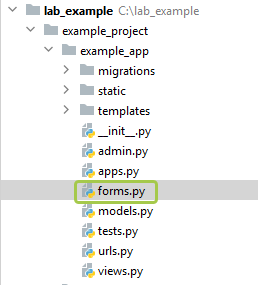
****

Рисунок. Пример – добавление файла для создания форм

Примеры описания форм на основе моделей.

from django import forms  
from .models import \*  
  
  
class ProductForm(forms.ModelForm):  
 class Meta:  
 model = Product  
 fields = [**'name'**, **'description'**]  
 labels = {**"name"**: **"Название продукта"**, **"description"**: **"Описание продукта"**}  
 widgets = {  
 **'name'**: forms.TextInput(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 300px;'**,  
 **'placeholder'**: **'Название продукта'** }),  
 **'description'**: forms.TextInput(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 300px;'**,  
 **'placeholder'**: **'Описание продукта'** })  
 }  
  
class ProviderForm(forms.ModelForm):  
 class Meta:  
 model = Provider  
 fields = [**'name'**]  
 labels = { **"name"**: **"Название компании"**}  
 widgets = {  
 **'name'**: forms.TextInput(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 300px;'**,  
 **'placeholder'**: **'Название компании'** })  
 }  
  
class StoreForm(forms.ModelForm):  
 class Meta:  
 model = Store  
 fields = [**'product'**, **'provider'**, **'date'**, **'price'**, **'quantity'**]  
 labels = {**'product'**: **"Продукт"**, **'provider'**:**"Поставщик"**, **'date'**:**"Дата"**, **'price'**:**"Цена"**, **'quantity'**:**"Количество"**}  
 widgets = {  
 **'product'**: forms.Select(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 350px;'**,  
 **'placeholder'**: **'Название продукта'** }),  
 **'provider'**: forms.Select(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 350px;'**,  
 **'placeholder'**: **'Поставщик'** }),  
 **'date'**: forms.DateInput(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 300px;'**,  
 **'placeholder'**: **'Дата'**,  
 **'type'**: **'date'** }),  
 **'price'**: forms.NumberInput(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 300px;'**,  
 **'placeholder'**: **'Цена'** }),  
 **'quantity'**: forms.NumberInput(attrs={  
 **'class'**: **"type-1"**,  
 **'style'**: **'max-width: 300px;'**,  
 **'placeholder'**: **'Количество'** })  
 }

**2. В файл стилей style.html добавим описание стилей формы**

Пример. Стили для классов элементов формы type-1 и btn

.btn {  
 background: rgb(210, 180, 140);  
 color: #fff;  
 border: 1px solid #eee;  
 border-radius: 20px;  
 box-shadow: 5px 5px 5px #eee;  
 text-shadow: none;  
 width: 160px;  
}

input {  
 display:block;  
 width:100%;  
 margin:10px 0;  
 padding:10px;  
}  
  
.type-1 {  
 border-radius:10px;  
 border: 1px solid #eee;  
 transition: .3s border-color;  
}  
.type-1:hover {  
 border: 1px solid #aaa;  
}

**3. В файле views.py создадим переменные для формы** на страницах Продукты (формы для Продуктов и Поставщиков) и Склад. Далее приведены функции, которые изменились в ходе выполнения данного пункта задания (остальной код пока остался без изменений)

…

from .forms import \*

**…**

def products(request):  
 productForm = ProductForm()  
 providerForm = ProviderForm()  
 data = {**'productForm'**: productForm, **'providerForm'**: providerForm}  
 return render(request, **'products1.html'**, context=data)  
  
  
def store(request):  
 storeForm = StoreForm()  
 return render(request, **'store1.html'**, context={**'storeForm'**: storeForm})

**4. На страницах products.html, store.html добавим разметку с описанием форм.**

Пример. В файл products.html добавим следующее описание:

<h2>Добавление данных</h2>  
<form method**="post"** action**="add\_product"**>  
 {% csrf\_token %}  
 <table class**="form-table"**>  
 {{productForm}}  
 </table>  
 <input class**="btn"** type**="submit"** value**="Добавить продукт"**>  
</form>  
  
<form method**="post"** action**="add\_provider"**>  
 {% csrf\_token %}  
 <table>  
 {{providerForm}}  
 </table>  
 <input class**="btn"** type**="submit"** value**="Добавить поставщика"**>  
</form>

В файл store.html добавим следующее описание:

<h2>Добавление данных</h2>  
<form method**="post"**>  
 {% csrf\_token %}  
 <table>  
 {{storeForm}}  
 </table>  
 <input class**="btn"** type**="submit"** value**="Добавить"**>  
</form>

Пример отображения созданных форм на странице:

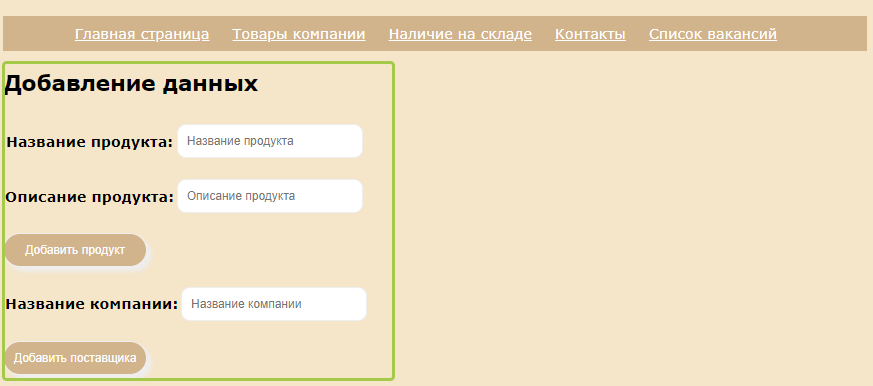


Рисунок. Формы добавления продуктов и поставщиков на странице products (отображение в браузере)

**5. Создадим функции в файле views.py, которые будут выполняться при получении данных с формы**

from django.shortcuts import render  
from .forms import \*  
from django.http import HttpResponseRedirect  
from .models import \*

def add\_product(request):  
 if request.method == **"POST"**:  
 product = ProductForm(request.POST)  
 if product.is\_valid():  
 product.save()  
 return HttpResponseRedirect(**"products"**)  
  
  
def add\_provider(request):  
 if request.method == **"POST"**:  
 provider = ProviderForm(request.POST)  
 if provider.is\_valid():  
 provider.save()  
 return HttpResponseRedirect(**"products"**)

Также поменяем код в функции, связанной со страницей Склад (добавленный код выделен цветом).

def store(request):  
 if request.method == **"POST"**:  
 product = StoreForm(request.POST)  
 if product.is\_valid():  
 product.save()  
 return HttpResponseRedirect(**"store"**)  
 storeForm = StoreForm()  
 storeProducts = Store.objects.all()  
 print(storeProducts)  
 return render(request, **'store1.html'**, context={**'storeForm'**: storeForm, **'store'**: storeProducts})

**6. Добавим в файл urls.py маршруты с данными функциями**

from django.urls import path  
from .views import \*  
  
urlpatterns = [  
 path(**''**, index),  
 path(**'contacts'**, contacts),  
 path(**'vacancy'**, vacancy),  
 path(**'products'**, products),  
 path(**'store'**, store),  
 path(**'add\_product'**, add\_product),  
 path(**'add\_provider'**, add\_provider)  
]

(при добавлении данных с формы они пока не будут отображаться в таблицах, так как требуется добавить вывод данных из таблиц) На этапе данного задания можно увидеть добавленные записи в таблицы продукты и поставщики при заполнении таблицы склад (в поле выбора продукта).