Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Разработка профессиональных приложений**

**Лабораторная работа №7**

«Разработка веб-приложения на Django»

Вариант 5

Выполнил

Студент группы ИВТИИбд-22

Гайфуллин Артур

Проверил(а):

Исхаков И.И.

Ульяновск

2025

**Задание по варианту**

Необходимо реализовать веб-сайт в среде Django, удовлетворяющий следующим требованиям:

1. Количество таблиц в БД - не менее 5. Таблицы должны быть спроектированы в соответствии с тремя нормальными формами.

2. Должны быть реализованы 3 типа связей: one-to-one, one-to-many, man-to-many.

3. Таблицы создаются с использованием моделей в Django и миграций.

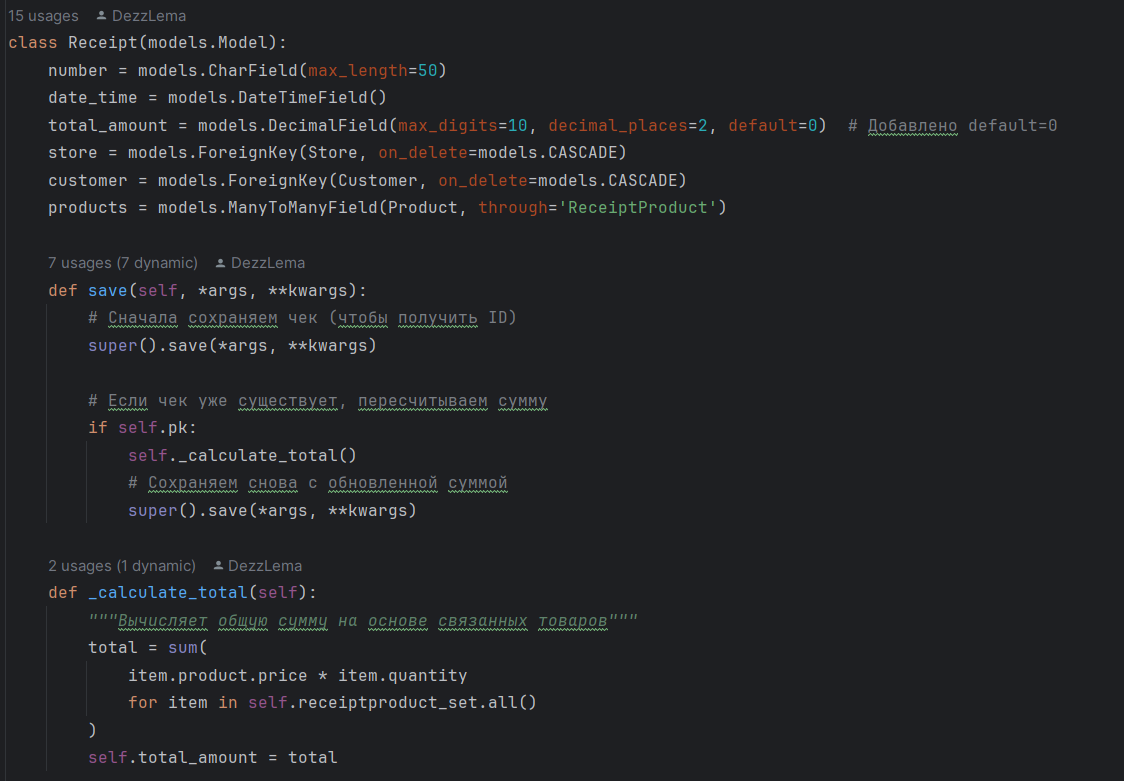
4. Как минимум, для 1 сущности должны быть реализованы CRUD операции (Create, read, update, delete) + представление списка сущностей.

5. Остальные таблицы должны быть доступны для редактирования в административной части сайта.

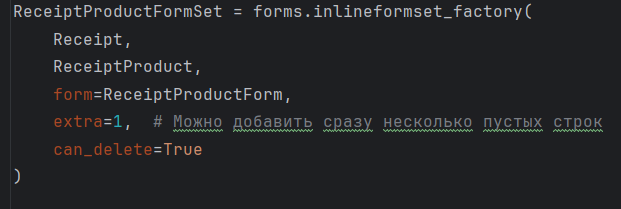
6. Необходимые пять таблиц должны быть разработаны для моделирования предметной области, заданной вариантом. Необходимо придумать эти таблицы, не забывая про здравый смысл.

**Описание реализации**

1. Модели данных



2. Формы и Formset



3. Представления

receipt\_create – обработка формы и formset

receipt\_update – редактирование с пересчетом суммы

receipt\_detail – просмотр чека с товарами

4. Связи

One-to-Many

Чек → Магазин  
Один магазин может иметь много чеков, но каждый чек принадлежит только одному магазину.

class Receipt(models.Model):

store = models.ForeignKey(Store, on\_delete=models.CASCADE)

Чек → Покупатель  
Один покупатель может иметь много чеков, но каждый чек принадлежит только одному покупателю.

customer = models.ForeignKey(Customer, on\_delete=models.CASCADE)

Товар → Категория товара  
Одна категория может содержать много товаров, но каждый товар принадлежит только одной категории.

class Product(models.Model):

category = models.ForeignKey(ProductCategory, on\_delete=models.CASCADE)

Many-to-Many

Чек ↔ Товар  
Один чек может содержать много товаров, и один товар может встречаться во многих чеках.

class Receipt(models.Model):

products = models.ManyToManyField(Product, through='ReceiptProduct')

class ReceiptProduct(models.Model):

receipt = models.ForeignKey(Receipt, on\_delete=models.CASCADE)

product = models.ForeignKey(Product, on\_delete=models.CASCADE)

quantity = models.PositiveIntegerField(default=1)

Здесь ReceiptProduct – промежуточная модель для хранения количества товаров в чеке.

One-to-One

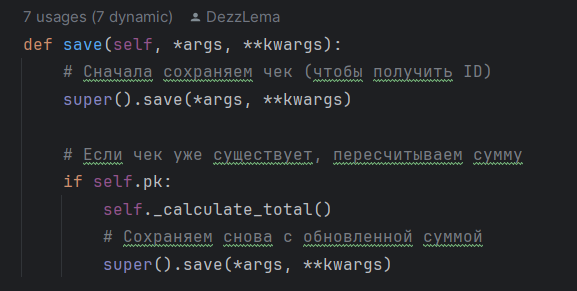
class Customer(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

**Описание возникших затруднений**

1. Сумма чека не обновлялась при создании, только после повторного редактирования.

Решение:  
1. Добавлен пересчет в методе save() модели Receipt:



2. Невозможно добавить несколько одинаковых товаров.

Решение:  
Использован Formset с отдельными строками для каждого товара.

Описание альтернативных способов решения

1. Расчет суммы

Вариант 1: Сигналы Django (post\_save) для пересчета.  
Вариант 2: Переопределение form\_valid во View.

2. Добавление товаров

Готовые решения (Django Admin, django-tables2).

**Исходный код**

Models.py

from django.db import models  
  
class Store(models.Model):  
 name = models.CharField(max\_length=100)  
 address = models.CharField(max\_length=200)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Customer(models.Model):  
 name = models.CharField(max\_length=100)  
 phone = models.CharField(max\_length=20)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class ProductCategory(models.Model):  
 name = models.CharField(max\_length=100)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Product(models.Model):  
 name = models.CharField(max\_length=100)  
 price = models.DecimalField(max\_digits=10, decimal\_places=2)  
 category = models.ForeignKey(ProductCategory, on\_delete=models.CASCADE)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Receipt(models.Model):  
 number = models.CharField(max\_length=50)  
 date\_time = models.DateTimeField()  
 total\_amount = models.DecimalField(max\_digits=10, decimal\_places=2, default=0) # Добавлено default=0  
 store = models.ForeignKey(Store, on\_delete=models.CASCADE)  
 customer = models.ForeignKey(Customer, on\_delete=models.CASCADE)  
 products = models.ManyToManyField(Product, through='ReceiptProduct')  
  
 def save(self, \*args, \*\*kwargs):  
 # Сначала сохраняем чек (чтобы получить ID)  
 super().save(\*args, \*\*kwargs)  
  
 # Если чек уже существует, пересчитываем сумму  
 if self.pk:  
 self.\_calculate\_total()  
 # Сохраняем снова с обновленной суммой  
 super().save(\*args, \*\*kwargs)  
  
 def \_calculate\_total(self):  
 *"""Вычисляет общую сумму на основе связанных товаров"""* total = sum(  
 item.product.price \* item.quantity  
 for item in self.receiptproduct\_set.all()  
 )  
 self.total\_amount = total  
  
 def \_\_str\_\_(self):  
 return f"Чек №{self.number} от {self.date\_time}"  
  
class ReceiptProduct(models.Model):  
 receipt = models.ForeignKey(Receipt, on\_delete=models.CASCADE)  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 quantity = models.PositiveIntegerField(default=1) # Количество товара  
  
 def \_\_str\_\_(self):  
 return f"{self.product.name} (x{self.quantity})"

views.py

from django.shortcuts import render, redirect, get\_object\_or\_404  
from django.views.generic import CreateView, UpdateView  
from django.forms import inlineformset\_factory  
from .models import Receipt, ReceiptProduct, Product  
from .forms import ReceiptForm, ReceiptProductForm  
from .models import Receipt  
from .forms import ReceiptForm, ReceiptProductFormSet  
from django.urls import reverse\_lazy  
  
ReceiptProductFormSet = inlineformset\_factory(  
 Receipt,  
 ReceiptProduct,  
 form=ReceiptProductForm,  
 extra=1,  
 can\_delete=True  
)  
  
  
def receipt\_list(request):  
 receipts = Receipt.objects.all()  
 return render(request, 'receipts/receipt\_list.html', {'receipts': receipts})  
  
  
def receipt\_detail(request, pk):  
 receipt = get\_object\_or\_404(Receipt, pk=pk)  
 return render(request, 'receipts/receipt\_detail.html', {'receipt': receipt})  
  
  
def receipt\_create(request):  
 products = Product.objects.all() # Получаем все товары для выпадающего списка  
  
 if request.method == 'POST':  
 form = ReceiptForm(request.POST)  
 if form.is\_valid():  
 receipt = form.save()  
 return redirect('receipt\_list')  
 else:  
 form = ReceiptForm()  
  
 return render(request, 'receipts/receipt\_form.html', {  
 'form': form,  
 'formset': form.product\_formset,  
 'products': products # Добавляем товары в контекст  
 })  
  
def receipt\_update(request, pk):  
 receipt = get\_object\_or\_404(Receipt, pk=pk)  
 if request.method == 'POST':  
 form = ReceiptForm(request.POST, instance=receipt)  
 if form.is\_valid():  
 form.save()  
 return redirect('receipt\_detail', pk=receipt.pk)  
 else:  
 form = ReceiptForm(instance=receipt)  
  
 return render(request, 'receipts/receipt\_form.html', {  
 'form': form,  
 'formset': form.product\_formset  
 })  
  
  
def receipt\_delete(request, pk):  
 receipt = get\_object\_or\_404(Receipt, pk=pk)  
 if request.method == 'POST':  
 receipt.delete()  
 return redirect('receipt\_list')  
 return render(request, 'receipts/receipt\_confirm\_delete.html', {'receipt': receipt})  
  
  
class ReceiptCreateView(CreateView):  
 model = Receipt  
 form\_class = ReceiptForm  
 template\_name = 'receipts/receipt\_form.html'  
 success\_url = '/receipts/'  
  
  
class ReceiptUpdateView(UpdateView):  
 model = Receipt  
 form\_class = ReceiptForm  
 template\_name = 'receipts/receipt\_form.html'  
 success\_url = reverse\_lazy('receipt\_list')  
  
 def get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 if self.request.POST:  
 context['formset'] = ReceiptProductFormSet(self.request.POST, instance=self.object)  
 else:  
 context['formset'] = ReceiptProductFormSet(instance=self.object)  
 return context  
  
 def form\_valid(self, form):  
 context = self.get\_context\_data()  
 formset = context['formset']  
 if formset.is\_valid():  
 self.object = form.save()  
 formset.instance = self.object  
 formset.save()  
 return super().form\_valid(form)  
 else:  
 return self.render\_to\_response(self.get\_context\_data(form=form))

forms.py

from django import forms  
from .models import Receipt, ReceiptProduct, Product  
  
  
class ReceiptProductForm(forms.ModelForm):  
 # Добавляем поле для поиска товаров  
 product = forms.ModelChoiceField(  
 queryset=Product.objects.all(),  
 widget=forms.Select(attrs={'class': 'product-select'}))  
  
 quantity = forms.IntegerField(min\_value=1, initial=1)  
  
 class Meta:  
 model = ReceiptProduct  
 fields = ['product', 'quantity']  
  
  
# Увеличиваем количество дополнительных форм  
ReceiptProductFormSet = forms.inlineformset\_factory(  
 Receipt,  
 ReceiptProduct,  
 form=ReceiptProductForm,  
 extra=1, # Можно добавить сразу несколько пустых строк  
 can\_delete=True  
)  
  
  
class ReceiptForm(forms.ModelForm):  
 class Meta:  
 model = Receipt  
 fields = ['number', 'date\_time', 'store', 'customer']  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.product\_formset = ReceiptProductFormSet(  
 instance=self.instance,  
 data=self.data if self.is\_bound else None,  
 files=self.files if self.is\_bound else None,  
 prefix='products'  
 )  
  
 def is\_valid(self):  
 return super().is\_valid() and self.product\_formset.is\_valid()  
  
 def save(self, commit=True):  
 # Сначала сохраняем чек  
 receipt = super().save(commit=commit)  
  
 if commit:  
 # Сохраняем товары  
 self.product\_formset.instance = receipt  
 self.product\_formset.save()  
  
 # Принудительно пересчитываем сумму  
 receipt.\_calculate\_total()  
 receipt.save()  
  
 return receipt

admin.py

from django.contrib import admin  
  
# Register your models here.  
from django.contrib import admin  
from .models import Store, Customer, ProductCategory, Product, Receipt, ReceiptProduct  
  
admin.site.register(Store)  
admin.site.register(Customer)  
admin.site.register(ProductCategory)  
admin.site.register(Product)  
admin.site.register(Receipt)  
admin.site.register(ReceiptProduct)

urls.py

from django.contrib import admin  
from django.urls import path, include  
from django.views.generic import RedirectView  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('receipts/', include('receipts.urls')),  
 path('', RedirectView.as\_view(url='receipts/')), # Перенаправление с / на /receipts/  
]

receipt\_list.html

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Список чеков</title>

</head>

<body>

    <h1>Список чеков</h1>

    <ul>

        {% for receipt in receipts %}

            <li>

                <a href="{% url 'receipt\_detail' pk=receipt.pk %}">

                    Чек №{{ receipt.number }} ({{ receipt.date\_time }})

                </a>

                | <a href="{% url 'receipt\_update' pk=receipt.pk %}">Редактировать</a>

                | <a href="{% url 'receipt\_delete' pk=receipt.pk %}">Удалить</a>

            </li>

        {% empty %}

            <li>Чеков пока нет.</li>

        {% endfor %}

    </ul>

    <a href="{% url 'receipt\_create' %}">Добавить новый чек</a>

</body>

</html>

Receipt\_detail.html

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Чек №{{ receipt.number }}</title>

</head>

<body>

    <h1>Чек №{{ receipt.number }}</h1>

    <p><strong>Дата и время:</strong> {{ receipt.date\_time }}</p>

    <p><strong>Сумма:</strong> {{ receipt.total\_amount }}</p>

    <p><strong>Магазин:</strong> {{ receipt.store.name }}</p>

    <p><strong>Покупатель:</strong> {{ receipt.customer.name }}</p>

    <h3>Товары:</h3>

    <ul>

        {% for item in receipt.receiptproduct\_set.all %}

            <li>{{ item.product.name }} ({{ item.quantity }} шт.)</li>

        {% empty %}

            <li>Товары не указаны.</li>

        {% endfor %}

    </ul>

    <a href="{% url 'receipt\_list' %}">Назад к списку</a>

    <a href="{% url 'receipt\_update' pk=receipt.pk %}">Редактировать</a>

</body>

</html>

Receipt\_form.html

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>{% if form.instance.pk %}Редактировать чек{% else %}Создать чек{% endif %}</title>

</head>

<body>

    <h1>{% if form.instance.pk %}Редактировать чек{% else %}Создать чек{% endif %}</h1>

    <form method="post" id="receipt-form">

    {% csrf\_token %}

    {{ form.as\_p }}

    <h3>Товары:</h3>

    {{ formset.management\_form }}

    <div id="products-form-container">

        {% for form in formset %}

            <div class="product-form">

                {{ form.as\_p }}

            </div>

        {% endfor %}

    </div>

    <button type="button" id="add-product" class="btn btn-secondary">Добавить товар</button>

    <button type="submit" class="btn btn-primary">Сохранить чек</button>

</form>

<script>

document.addEventListener('DOMContentLoaded', function() {

    const container = document.getElementById('products-form-container');

    const addButton = document.getElementById('add-product');

    const totalForms = document.getElementById('id\_products-TOTAL\_FORMS');

    let formCount = parseInt(totalForms.value);

    addButton.addEventListener('click', function() {

        const newForm = document.createElement('div');

        newForm.classList.add('product-form');

        newForm.innerHTML = `

            <p>

                <label for="id\_products-${formCount}-product">Товар:</label>

                <select name="products-${formCount}-product" id="id\_products-${formCount}-product" class="product-select">

                    {% for product in products %}

                        <option value="{{ product.id }}">{{ product.name }}</option>

                    {% endfor %}

                </select>

            </p>

            <p>

                <label for="id\_products-${formCount}-quantity">Количество:</label>

                <input type="number" name="products-${formCount}-quantity" value="1" min="1" id="id\_products-${formCount}-quantity">

            </p>

            <p>

                <input type="hidden" name="products-${formCount}-id" id="id\_products-${formCount}-id">

                <input type="checkbox" name="products-${formCount}-DELETE" id="id\_products-${formCount}-DELETE">

                <label for="id\_products-${formCount}-DELETE">Удалить</label>

            </p>

        `;

        container.appendChild(newForm);

        formCount++;

        totalForms.value = formCount;

    });

});

</script>

    <a href="{% url 'receipt\_list' %}">Отмена</a>

</body>

</html>

Receipt\_confirm

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Удалить чек №{{ receipt.number }}?</title>

</head>

<body>

    <h1>Удалить чек №{{ receipt.number }}?</h1>

    <p>Дата: {{ receipt.date\_time }}</p>

    <p>Сумма: {{ receipt.total\_amount }}</p>

    <form method="post">

        {% csrf\_token %}

        <button type="submit">Да, удалить</button>

    </form>

    <a href="{% url 'receipt\_detail' pk=receipt.pk %}">Отмена</a>

</body>

</html>