Модуль 2. Разработка функционала

# Шаг 1. Файл module.py

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Customer(models.Model):

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

name = models.CharField(max\_length=200, null=True)

email = models.CharField(max\_length=200)

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

return self.name

# Step 2. More new classes в models.py

class Product(models.Model):

name = models.CharField(max\_length=200)

price = models.FloatField()

digital = models.BooleanField(default=False,null=True, blank=True)

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

return self.name

class Order(models.Model):

customer = models.ForeignKey(Customer, on\_delete=models.SET\_NULL, null=True, blank=True)

date\_ordered = models.DateTimeField(auto\_now\_add=True)

complete = models.BooleanField(default=False)

transaction\_id = models.CharField(max\_length=100, null=True)

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

return str(self.id)

class OrderItem(models.Model):

product = models.ForeignKey(Product, on\_delete=models.SET\_NULL, null=True)

order = models.ForeignKey(Order, on\_delete=models.SET\_NULL, null=True)

quantity = models.IntegerField(default=0, null=True, blank=True)

date\_added = models.DateTimeField(auto\_now\_add=True)

# Step 3. Adding ShippingAddress в models.py

class ShippingAddress(models.Model):

customer = models.ForeignKey(Customer, on\_delete=models.SET\_NULL, null=True)

order = models.ForeignKey(Order, on\_delete=models.SET\_NULL, null=True)

address = models.CharField(max\_length=200, null=False)

city = models.CharField(max\_length=200, null=False)

state = models.CharField(max\_length=200, null=False)

zipcode = models.CharField(max\_length=200, null=False)

date\_added = models.DateTimeField(auto\_now\_add=True)

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

return self.address

# Step 4. Adding new models to admin.py

from django.contrib import admin

from .models import \*

admin.site.register(Customer)

admin.site.register(Product)

admin.site.register(Order)

admin.site.register(OrderItem)

admin.site.register(ShippingAddress)

# Step 5. Add new func in views.py

from .models import \*

def store(request):

    products = Product.objects.all()

    context = {'products':products}

    return render(request, 'store/store.html', context)

# Шаг 6. Расписали на питоне функции для рендера

<https://github.com/divanov11/ecom_steps/blob/master/m2-p3-s2-shipping-att.py>

# Шаг 7. Добавили обращение к urlpatterns в usrl.py

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

# Step 8. Add MEDIA\_ROOT in setting.py

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'static/images')

# Step 9. Полностью поменяли store.html сделав так, чтобы список товаров грузился из бд

{% extends 'store/main.html' %}

{% load static %}

{% block content %}

    <div class="row">

        {% for product in products %}

        <div class="col-lg-4">

            <img class="thumbnail" src="{% static 'images/placeholder.png' %}">

            <div class="box-element product">

                <h6><strong>{{product.name}}</strong></h6>

                <hr>

                <button  class="btn btn-outline-secondary add-btn">Add to Cart</button>

                <a class="btn btn-outline-success" href="#">View</a>

                <h4 style="display: inline-block; float: right"><strong>${{product.price}}</strong></h4>

            </div>

        </div>

        {% endfor %}

    </div>

{% endblock content %}

# Шаг 10. Добавили характеристику property в класс Product в models.py

@property

    def imageURL(self):

        try:

            url = self.image.url

        except:

            url = ''

        return url

# Шаг 11. Добавили характеристику property в класс OrderItem в models.py

@property

    def get\_cart\_total(self):

        orderitems = self.orderitem\_set.all()

        total = sum([item.get\_total for item in orderitems])

        return total

    @property

    def get\_cart\_items(self):

        orderitems = self.orderitem\_set.all() # выбор всех итемов из set

        total = sum([item.quantity for item in orderitems])

        return total

# Шаг 12. Добавили характеристику property в класс Order в models.py

@property

    def get\_total(self):

        total = self.product.price \* self.quantity

        return total

# Step 13. Add new func updateItem in views.py

def updateItem(request):

    data = json.loads(request.body)

    productId = data['productId']

    action = data['action']

    print('Action:', action)

    print('Product:', productId)

    customer = request.user.customer

    product = Product.objects.get(id=productId)

    order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

    orderItem, created = OrderItem.objects.get\_or\_create(order=order, product=product)

    if action == 'add':

        orderItem.quantity = (orderItem.quantity + 1)

    elif action == 'remove':

        orderItem.quantity = (orderItem.quantity - 1)

    orderItem.save()

    if orderItem.quantity <= 0:

        orderItem.delete()

    return JsonResponse('Item was added', safe=False)

# Step 14. Add new js file cart.js

var updateBtns = document.getElementsByClassName('update-cart')

for (i = 0; i < updateBtns.length; i++) {

    updateBtns[i].addEventListener('click', function(){

        var productId = this.dataset.product

        var action = this.dataset.action

        console.log('productId:', productId, 'Action:', action)

        console.log('USER:', user)

        if (user == 'AnonymousUser'){

            console.log('User is not authenticated')

        }else{

           console.log('User is authenticated, sending data...')

              }

    })

}

# Step 15. Add new func updateUserOrder in cart.js

var updateBtns = document.getElementsByClassName('update-cart')

for (i = 0; i < updateBtns.length; i++) {

    updateBtns[i].addEventListener('click', function(){

        var productId = this.dataset.product

        var action = this.dataset.action

        console.log('productId:', productId, 'Action:', action)

        console.log('USER:', user)

        if (user == 'AnonymousUser'){

            console.log('User is not authenticated')

        }else{

            updateUserOrder(productId, action)

        }

    })

}

function updateUserOrder(productId, action){

    console.log('User is authenticated, sending data...')

        var url = '/update\_item/'

        fetch(url, {

            method:'POST',

            headers:{

                'Content-Type':'application/json',

            },

            body:JSON.stringify({'productId':productId, 'action':action})

        })

        .then((response) => {

           return response.json();

        })

        .then((data) => {

            console.log('Data:', data)

        });

}

# Step 16. Add new handling to cart.js in main.html

<script type="text/javascript" src="{% static 'js/cart.js' %}"></script>

</body>

</html>

# Step 17. Add new script in main.html

<script type="text/javascript">

        var user = '{{request.user}}'

        function getToken(name) {

            var cookieValue = null;

            if (document.cookie && document.cookie !== '') {

                var cookies = document.cookie.split(';');

                for (var i = 0; i < cookies.length; i++) {

                    var cookie = cookies[i].trim();

                    // Does this cookie string begin with the name we want?

                    if (cookie.substring(0, name.length + 1) === (name + '=')) {

                        cookieValue = decodeURIComponent(cookie.substring(name.length + 1));

                        break;

                    }

                }

            }

            return cookieValue;

        }

        var csrftoken = getToken('csrftoken')

    </script>

# Step 18. Add property get\_cart\_items in models.py

@property

    def get\_cart\_items(self):

        orderitems = self.orderitem\_set.all() # выбор всех итемов из set

        total = sum([item.quantity for item in orderitems])

        return total

# Step 19. Add property get\_cart\_total in models.py

@property

    def get\_cart\_total(self):

        orderitems = self.orderitem\_set.all()

        total = sum([item.get\_total for item in orderitems])

        return total

# 20. Change checkout in views.py

def checkout(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        carItems = order.get\_cart\_items

    else:

        #Create empty cart for now for non-logged in user

        items = []

        order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':false}

        carItems = order['get\_cart\_items']

    context = {'items':items, 'order':order, 'carItems':carItems}

    return render(request, 'store/checkout.html', context)

# 21. Change def store in views.py

def store(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        carItems = order.get\_cart\_items

    else:

        items = []

        order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':False}

        carItems = order['get\_cart\_items']

    products = Product.objects.all()

    context = {'products':products}

    return render(request, 'store/store.html', context)

# 22. Change def cart in views.py

def cart(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

    else:

        #Create empty cart for now for non-logged in user

        items = []

        order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':False}

        carItems = order['get\_cart\_items']

    context = {'items':items, 'order':order, 'carItems':carItems}

    return render(request, 'store/cart.html', context)

# 23. Change checkout, working with db, in checkout.html

<div class="col-lg-6">

            <div class="box-element">

                <a  class="btn btn-outline-dark" href="{% url 'cart' %}">&#x2190; Back to Cart</a>

                <hr>

                <h3>Order Summary</h3>

                <hr>

                {% for item in items %}

                <div class="cart-row">

                    <div style="flex:2"><img class="row-image" src="{{item.product.imageURL}}"></div>

                    <div style="flex:2"><p>{{item.product.name}}</p></div>

                    <div style="flex:1"><p>${{item.product.price|floatformat:2}}</p></div>

                    <div style="flex:1"><p>x{{item.quantity}}</p></div>

                </div>

                {% endfor %}

                <h5>Items:   {{order.get\_cart\_items}}</h5>

                <h5>Total:   ${{order.get\_cart\_total|floatformat:2}}</h5>

            </div>

        </div>

# 24. Add javascript in checkout.html

    <script type="text/javascript">

        var shipping = '{{order.shipping}}'

        if (shipping == 'False'){

            document.getElementById('shipping-info').innerHTML = ''

        }

    </script>

# 25. Change script in checkout.html

    <script type="text/javascript">

        var shipping = '{{order.shipping}}'

        if (shipping == 'False'){

            document.getElementById('shipping-info').innerHTML = ''

        }

        var form = document.getElementById('form')

        form.addEventListener('submit', function(e){

            e.preventDefault()

            console.log('Form Submitted ...')

            document.getElementById('form-button').classList.add("hidden")

            document.getElementById('payment-info').classList.remove("hidden")

        })

    </script>

# 26. Добавили обработку сообщений от нажатия кнопки make payment в script in checkout.html

document.getElementById('make-payment').addEventListener('click', function(e){

            submitFormData()

        })

        function submitFormData(){

            console.log('Payment button clicked')

# 27. Добавили в скрипт if shipping == 'False' && user != 'AnonymousUser'

if( shipping == 'False' && user != 'AnonymousUser'){

            //Hide entie form if user is logged in and shipping is false

            document.getElementById('form-wrapper').classList.add("hidden");

            // Show payment if logged in user wants to buy an item does not require shipping

            document.getElementById('payment-info').classList.remove("hidden");

        }

# 28. Add function submitFormData in script in checkout.html

document.getElementById('make-payment').addEventListener('click', function(e){

            submitFormData();

        })

        function submitFormData(){

            console.log('Payment button clicked');

var shipping = '{{order.shipping}}'

            var total = '{{order.get\_cart\_total|floatformat:2}}'

            var userFromData = {

                'name': null,

                'email':null,

                'total':total,

            };

            var shippingInfo = {

                'address':null,

                'city':null,

                'state':null,

                'zipcode':null,

            };

            if(shipping != 'False'){

                shippingInfo.address = form.address.value;

                shippingInfo.city = form.city.value;

                shippingInfo.state = form.state.value;

                shippingInfo.zipcode = form.zipcode.value;

            }

            if( user == 'AnonymousUser'){

                userFromData.name = form.name.value;

                userFromData.email = form.email.value;

            }

            console.log('Shipping Info:', shippingInfo);

            console.log('User Info:', userFromData);

        }

# 29. Added url process\_order in script in checkout.html

var url = "process\_order";

            fetch(url, {

                    method:'POST',

                    headers:{

                        'Content-Type':'application/json',

                        'X-CSRFToken':csrftoken,

                    },

                    body:JSON.stringify({'form':userFromData, 'shipping':shippingInfo})

            })

            .then((response) => response.json())

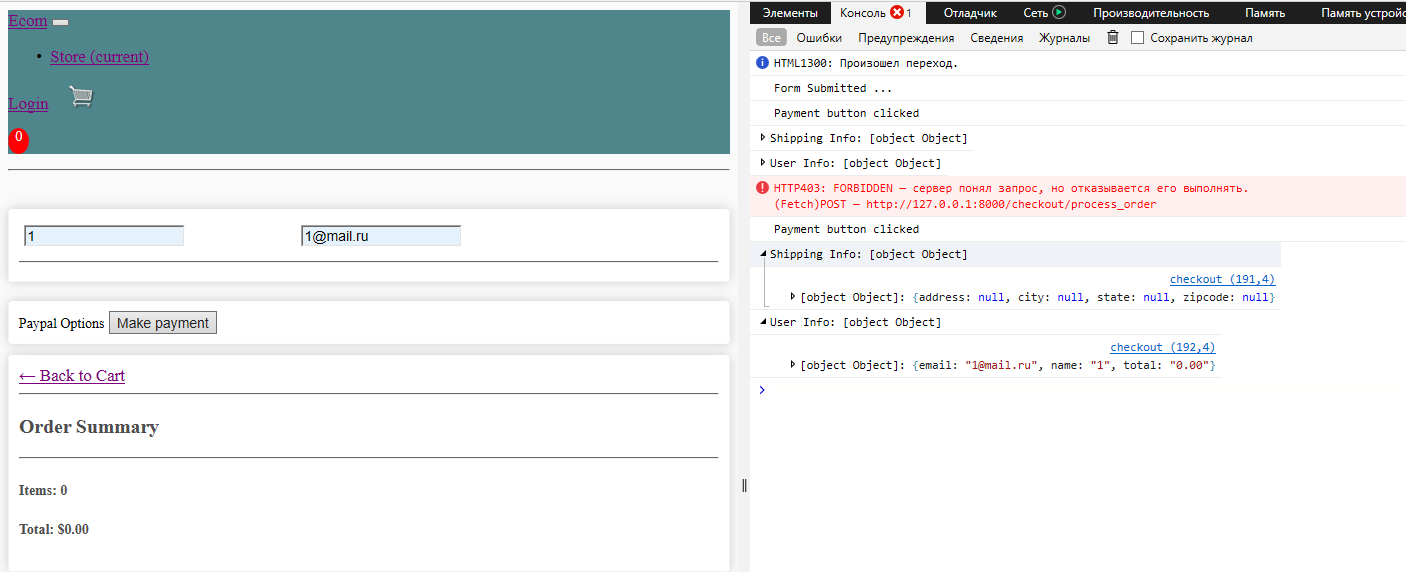
            .then((data) => {

                console.log('Success:', data);

                alert('Transaction completed');

                window.location.href = "{% url 'store' %}"

            })



# 30. Added new function processOrder in views.py

import datetime

def processOrder(request):

    transaction\_id = datetime.datetime.now().timestamp()

    data = json.loads(request.body)

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        total = float(data['form']['total'])

        order.transaction\_id = transaction\_id

    else:

        print('User is not logged in')

    return JsonResponse('Payment submitted...', safe=False)

# 31. Added new fitchers in proccessOrder in views.py

if total == order.get\_cart\_total:

            order.complete = True

        order.save()

        if order.shipping == True:

            ShippingAddress.objects.create(

                customer=customer,

                order=order,

                address=data['shipping']['address'],

                city=data['shipping']['city'],

                state=data['shipping']['state'],

                zipcode=data['shipping']['zipcode'],

            )

# 32. Added new func getCookie in javascript in main.html

function getCookie(name){

            // Split cookie string and get all individual name=value paris in an array

                var cookieArr = document.cookie.split(";");

                // Loop throught the array elements

                for(var i = 0; i < cookieArr.length; i++){

                    var cookiePair = cookieArr[i].split("=");

                    /\*

                        Removing whitespace at the beggining of the cookie name and compare it with the given string

                    \*/

                    if(name == cookiePair[0].trim()){

                        return decodeURIComponent(cookiePair[1]);

                    }

                }

                // Return null if not found

                return null;

        }

        var cart = JSON.parse(getCookie('cart'));

        if(cart == undefined){

            cart = {};

            console.log('Cart Created!', cart);

            document.cookie ='cart=' + JSON.stringify(cart) + ";domain=;path=/";

        }

        console.log('Cart:', cart)

# 33. Added new function addCookieItem in cart.js

function addCookieItem(productId, action){

    console.log('User is not authenticated');

    if(action == 'add'){

        if(cart[productId] == undefined){

            cart[productId]= {'quantity':1}

        }

        else{

            cart[productId]['quantity'] += 1

        }

    }

    if(action == 'remove'){

        cart[productId]['quantity'] -= 1

        if(cart[productId]['quantity'] <= 0 ){

            console.log('Item should be deleted')

            delete cart[productId]

        }

    }

    console.log('CART:', cart)

    document.cookie = 'cart=' + JSON.stringify(cart) + ";domain=;path/"

    location.reload()

}

# 34. Change def cart in views.py

if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        #Create empty cart for now for non-logged in user

        try:

            cart = json.loads(request.COOKIES['cart'])

        except:

            cart = {}

            print('CART:', cart)

        items = []

        order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':False}

        cartItems = order['get\_cart\_items'] # NEW

        for i in cart: # NEW

            cartItems += cart[i]['quantity'] # NEW

context = {'items':items, 'order':order, 'cartItems':cartItems}

    return render(request, 'store/cart.html', context)

# 35. Change def checkout in views.py

def checkout(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        #Create empty cart for now for non-logged in user

        items = []

        order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':False}

        cartItems = order['get\_cart\_items']

context = {'items':items, 'order':order, 'cartItems':cartItems}

    return render(request, 'store/checkout.html', context)

# 36. Change def store in views.py

def store(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        items = []

        order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':False}

        cartItems = order['get\_cart\_items']

    products = Product.objects.all()

    context = {'products':products, 'cartItems':cartItems}

    return render(request, 'store/store.html', context)

m4-p2-s2-loop.py

# 37. Add new var in def cart in views.py

product = Product.objects.get(id=i)

            total = (product.price \* cart[i]['quantity'])

            order['get\_cart\_total'] += total

            order['get\_cart\_items'] += cart[i]['quantity']

# 38. Добавляем обращение к итемам в функции cart in views.py

            item = {

                'id':product.id,

                'product':{'id':product.id,'name':product.name, 'price':product.price,

                'imageURL':product.imageURL}, 'quantity':cart[i]['quantity'],

                'digital':product.digital,'get\_total':total,

                }

            items.append(item)

# 39. Добавили условие и try except в функции cart in views.py

for i in cart:

            #We use try block to prevent items in cart that may have been removed from causing error

            try:

                cartItems += cart[i]['quantity']

                product = Product.objects.get(id=i)

                total = (product.price \* cart[i]['quantity'])

                order['get\_cart\_total'] += total

                order['get\_cart\_items'] += cart[i]['quantity']

                item = {

                    'id':product.id,

                    'product':{'id':product.id,'name':product.name, 'price':product.price,

                    'imageURL':product.imageURL}, 'quantity':cart[i]['quantity'],

                    'digital':product.digital,'get\_total':total,

                    }

                items.append(item)

                if product.digital == False:

                    order['shipping'] = True

            except:

                pass

# 40. Продолжаем обработку cookie. Create utils.py with function cookieCart, в которую записали метод обработки из def cart в views.py

def cookieCart(request):

    #Create empty cart for now for non-logged in user

    try:

        cart = json.loads(request.COOKIES['cart'])

    except:

        cart = {}

        print('CART:', cart)

    items = []

    order = {'get\_cart\_total':0, 'get\_cart\_items':0, 'shipping':False}

    cartItems = order['get\_cart\_items']

    for i in cart:

        #We use try block to prevent items in cart that may have been removed from causing error

        try:

            cartItems += cart[i]['quantity']

            product = Product.objects.get(id=i)

            total = (product.price \* cart[i]['quantity'])

            order['get\_cart\_total'] += total

            order['get\_cart\_items'] += cart[i]['quantity']

            item = {

                'id':product.id,

                'product':{'id':product.id,'name':product.name, 'price':product.price,

                'imageURL':product.imageURL}, 'quantity':cart[i]['quantity'],

                'digital':product.digital,'get\_total':total,

                }

            items.append(item)

            if product.digital == False:

                order['shipping'] = True

        except:

            pass

    return {'cartItems':cartItems ,'order':order, 'items':items}

# 41. Отредактировали весь файл views.py так, чтобы все функции store, cart и checkout работали через function cookieCart, помещая туда request, и возвращая в переменную cartData. Все это при условии, что пользователь не авторизован

def store(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        cookieData = cookieCart(request)

        cartItems = cookieData['cartItems']

        order = cookieData['order']

        items = cookieData['items']

…

def cart(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        cookieData = cookieCart(request)

        cartItems = cookieData['cartItems']

        order = cookieData['order']

        items = cookieData['items']

def checkout(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        #Create empty cart for now for non-logged in user

        cookieData = cookieCart(request)

        cartItems = cookieData['cartItems']

        order = cookieData['order']

        items = cookieData['items']

# 42. Создали новую функцию cartData в utils.py

def cartData(request):

    return {}

# 43. Записали в функцию cartData обработку из функций store, cart и checkout

def cartData(request):

    if request.user.is\_authenticated:

        customer = request.user.customer

        order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

        items = order.orderitem\_set.all()

        cartItems = order.get\_cart\_items

    else:

        cookieData = cookieCart(request)

        cartItems = cookieData['cartItems']

        order = cookieData['order']

        items = cookieData['items']

    return {'cartItems':cartItems ,'order':order, 'items':items}

# 44. Изменили функции store, cart и checkout в views.py так, чтобы все работало через функции cartData and cookieCart

def store(request):

    data = cartData(request)

    cartItems = data['cartItems']

    order = data['order']

    items = data['items']

    products = Product.objects.all()

    context = {'products':products, 'cartItems':cartItems}

    return render(request, 'store/store.html', context)

def cart(request):

    data = cartData(request)

    cartItems = data['cartItems']

    order = data['order']

    items = data['items']

    context = {'items':items, 'order':order, 'cartItems':cartItems}

    return render(request, 'store/cart.html', context)

def checkout(request):

    data = cartData(request)

    cartItems = data['cartItems']

    order = data['order']

    items = data['items']

    context = {'items':items, 'order':order, 'cartItems':cartItems}

    return render(request, 'store/checkout.html', context)

45. Добавили словарь cart в js в checkout.hmtl

cart = {}

                document.cookie ='cart=' + JSON.stringify(cart) + ";domain=;path=/"

# 46. Add new function def guestOrder to utils.py

def guestOrder(request, data):

    name = data['form']['name']

    email = data['form']['email']

    cookieData = cookieCart(request)

    items = cookieData['items']

    customer, created = Customer.objects.get\_or\_create(

            email=email,

            )

    customer.name = name

    customer.save()

    order = Order.objects.create(

        customer=customer,

        complete=False,

        )

    for item in items:

        product = Product.objects.get(id=item['id'])

        orderItem = OrderItem.objects.create(

            product=product,

            order=order,

            quantity=item['quantity'],

        )

    return customer, order

47. Добавили кнопку Paypal in checkout.html

# 48. Изменили store.html так, чтобы по кнопке срабатывала функция add и data-product="{{product.id}}" data-action="add"

{% extends 'store/main.html' %}

{% load static %}

{% block content %}

    <div class="row">

        {% for product in products %}

        <div class="col-lg-4">

            <img class="thumbnail" src="{{product.imageURL}}">

            <div class="box-element product">

                <h6><strong>{{product.name}}</strong></h6>

                <hr>

                <button data-product="{{product.id}}" data-action="add" class="btn btn-outline-secondary add-btn update-cart">Add to Cart</button>

                <a class="btn btn-outline-success" href="#">View</a>

                <h4 style="display: inline-block; float: right"><strong>${{product.price}}</strong></h4>

            </div>

        </div>

        {% endfor %}

    </div>

{% endblock content %}