*Федеральное государственное бюджетное образовательное учреждение* *высшего профессионального образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Компьютерные системы и сети

**Отчет**

**по лабораторной работе №4**

**Разработка основного функционала frontend-части для приложения “Интернет-магазин компьютеров”**

Студент гр. ИУ6-13М\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Макаров А.В.

(Подпись, дата) (Фамилия И.О.)

Студент гр. ИУ6-12М\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Разуваев К.А.

(Подпись, дата) (Фамилия И.О.)

Студент гр. ИУ6-12М\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Колесников В.С.

(Подпись, дата) (Фамилия И.О.)

Студент гр. ИУ6-12М\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пирмамедов М.Э.

(Подпись, дата) (Фамилия И.О.)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Еремин О.Ю.

(Подпись, дата) (Фамилия И.О.)

Москва, 2019

**Задание:** разработать основной функционал для представления frontend-части приложения.

Для выполнения поставленной задачи необходимо разработать представления для веб-страницы.

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Исходный код, отвечающий за выполнение пользовательских действий представлен ниже:

**from** django.shortcuts **import** render, HttpResponseRedirect  
**from** django.urls **import** reverse  
**from** django.views.generic **import** View,ListView  
**from** app.forms **import** \*  
**from** .models **import** \*  
**from** app.registration **import** \*  
**from** django.contrib.auth.models **import** User  
**from** django.contrib **import** auth  
**from** django.contrib.auth **import** authenticate  
  
  
**def** main(request):  
 computers = Computer.objects.all()  
 **return** render(request, **'main.html'**, {**"computers"**: computers})  
  
  
**def** order\_non\_auth(request):  
 **return** render(request, **'order\_non\_auth.html'**)  
  
  
**def** registration(request):  
 form = RegistrationForm(request.POST **or None**)  
 **if** request.method == **'POST'**:  
 **if** form.is\_valid():  
 user = User.objects.create\_user(username=request.POST.get(**'username'**),  
 email=request.POST.get(**'email'**),  
 password=request.POST.get(**'password'**),  
 first\_name=request.POST.get(**'firstname'**),  
 last\_name=request.POST.get(**'surname'**))  
 *# ...* **return** HttpResponseRedirect(**'/computers'**)  
 **return** render(request, **'registration.html'**, {**'form'**: form})  
  
  
**def** logout(request):  
 auth.logout(request)  
 **return** HttpResponseRedirect(**'/'**)  
  
  
**def** authorization(request):  
 error = **""** username = **None** password = **None  
 if** request.method == **'POST'**:  
 username = request.POST.get(**'username'**)  
 password = request.POST.get(**'password'**)  
 user = authenticate(username=username, password=password)  
 **if** user:  
 auth.login(request, user)  
 **return** HttpResponseRedirect(**'/'**)  
 **else**:  
 error = **"Пользователь не найден"  
 return** render(request, **'auth.html'**, locals())  
  
  
**def** loginn(request):  
 **return** render(request, **'user.html'**, locals())  
  
  
**def** order\_add(request):  
 form = OrderForm(request.POST **or None**, request.FILES **or None**)  
 computers = Computer.objects.all()  
 **if** form.is\_valid():  
 instance = form.save()  
 Order.objects.get(id\_order=instance.pk).id\_user.add(request.user.id)  
 **return** HttpResponseRedirect(reverse(**'computers\_url'**))  
 **return** render(request, **"order\_add.html"**, {**"computers"**: computers, **"form"**: form})  
  
  
**class** ComputersList(ListView):  
 model = Computer  
 template\_name = **"computers.html"** paginate\_by = 3  
 form\_class = ComputerAddForm  
  
 **def** get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context[**'computers'**] = Computer.objects.all()  
 **return** context  
  
 **def** computer\_add\_modal(self, request):  
 form = self.form\_class(request.POST **or None**, request.FILES **or None**)  
 **if** form.is\_valid():  
 instance = form.save(commit=**False**)  
 instance.save()  
 **return** HttpResponseRedirect(reverse(**'computer\_url'**, args=(instance.pk,)))  
 **return** render(request, self.template\_name, {**'form'**: form})  
  
  
**def** computer(request, computer\_id=0):  
 **if** request.method == **'POST'**:  
 Computer.objects.get(id\_computer=computer\_id)  
 **return** render(request, **"computer.html"**, {**'computer'**: Computer.objects.get(id\_computer=computer\_id)}, locals())  
  
  
**class** OrdersView(ListView):  
 model = Order  
 template\_name = **'orders.html'  
  
 def** get\_context\_data(self, \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 print(Order.objects.all().first().get\_users)  
 **return** context  
  
  
**def** computer\_add(request):  
 form = ComputerAddForm(request.POST **or None**, request.FILES **or None**)  
 context = {**"form"**: form}  
 **if** form.is\_valid():  
 instance = form.save()  
 **return** HttpResponseRedirect(reverse(**'computer\_url'**, args=(instance.pk, )))  
 **return** render(request, **"computer\_add.html"**, context)

Ниже приведено описание методов:

* main отвечает за отображение информации на главной странице приложения;
* registration – отвечает за выполнение функции регистрации нового аккаунта пользователя;
* logout – отвечает за выход пользователя из аккаунта;
* authorization – отвечает за вход пользователя в систему;
* order\_add – отвечает за обработку нового заказа;
* computer – возвращает идентификатор товара;
* computer\_add – срабатывает при добавлении нового компьютера в список товаров.

Для достижения поставленной задачи были разработаны следующие классы:

* ComputersList – отвечает за отображение списка компьютеров на странице;
* OrdersView – отвечает за отображение заказов. Используется только в режиме администратора.

**Вывод:** в данной работе был разработан основной функционал представления frontend-части для приложения.