Московский государственный университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Домашнее задание по курсу:

«Разработка Интернет Приложений»

Студент группы РТ5-51:

Разуваев К. А.

Преподаватель:

Гапанюк Ю. Е.

«___»______________________________

Москва 2017 г.

Задание и порядок выполнения

Разработать вебсервис на базе технологий: Python, Django, JS, MySQL

№	Субъект	Отношение	Объект
15	Пользователь	Заказ	Компьютер

Исходный код:

settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '74gex83*3yh7=&y9p#ob7#3jliqqz1p82@mm(u^%if@+bip&!k'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app.apps.AppConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Homework.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Homework.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.11/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.11/howto/static-files/

STATIC_URL = '/static/'


STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
    '/app/static/',
]

MEDIA_ROOT = os.path.join(BASE_DIR, 'app/media/')
MEDIA_URL = '/media/'
```

urls.py

```
from django.conf.urls import url, include
from django.contrib import admin
from app.views import *
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', main, name="main"),
    url(r'^registration/', registration, name='registration_url'),
    url(r'^logout/', logout, name='logout_url'),
    url(r'^auth/', authorization, name='auth_url'),
    url(r'^login/', loginn, name="login"),
    url(r'^order/', order_add, name="order"),
    url(r'^orderna/', order_non_auth, name="order_non_auth"),
    url(r'^computers/$', ComputersList.as_view(), name='computers_url'),
    url(r'^computer/(?P<computer_id>\d+)/$', computer, name='computer_url'),
    url(r'^orders/', OrdersView.as_view(), name="orders"),
    url(r'^add/', computer_add,  name="computer_add"),
]


if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

views.py

```
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('main'))
    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)
```

forms.py

```
from django import forms
from app.models import *


class OrderForm(forms.ModelForm):
    class Meta:
        model = Order
        fields = ["id_comp"]


class ComputerAddForm(forms.ModelForm):
    class Meta:
        model = Computer
        fields = ["name_computer", "model_computer", "company_computer",
                  "price_computer", "characteristics_computer",
"photo_computer"]
```

models.py

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.


class Computer(models.Model):
    id_computer = models.AutoField(primary_key=True)
    name_computer = models.CharField(max_length=100,
verbose_name='Компьютер')
    model_computer = models.CharField(max_length=100, verbose_name='Модель')
    company_computer = models.CharField(max_length=100,
verbose_name='Производитель')
    price_computer = models.IntegerField(verbose_name='Цена')
    characteristics_computer =
models.TextField(verbose_name='Характеристики')
    photo_computer = models.ImageField(null=True, blank=True,
verbose_name='Фото')

    def __str__(self):
        return self.name_computer

    class Meta:
        verbose_name_plural = "Компьютеры"
        verbose_name = "Компьютер"


class Order(models.Model):
    id_order = models.AutoField(primary_key=True, verbose_name='ID Заказа')
    id_comp = models.ForeignKey(Computer, on_delete=models.CASCADE,
verbose_name='Название компьютера')
    id_user = models.ManyToManyField(User, verbose_name='ID Пользователя')

    def get_users(self):
        return [u.username for u in self.id_user.all()]
    get_users.short_description = 'Заказы пользователей'

    class Meta:
        verbose_name_plural = "Заказы"
        verbose_name = "Заказ"
```

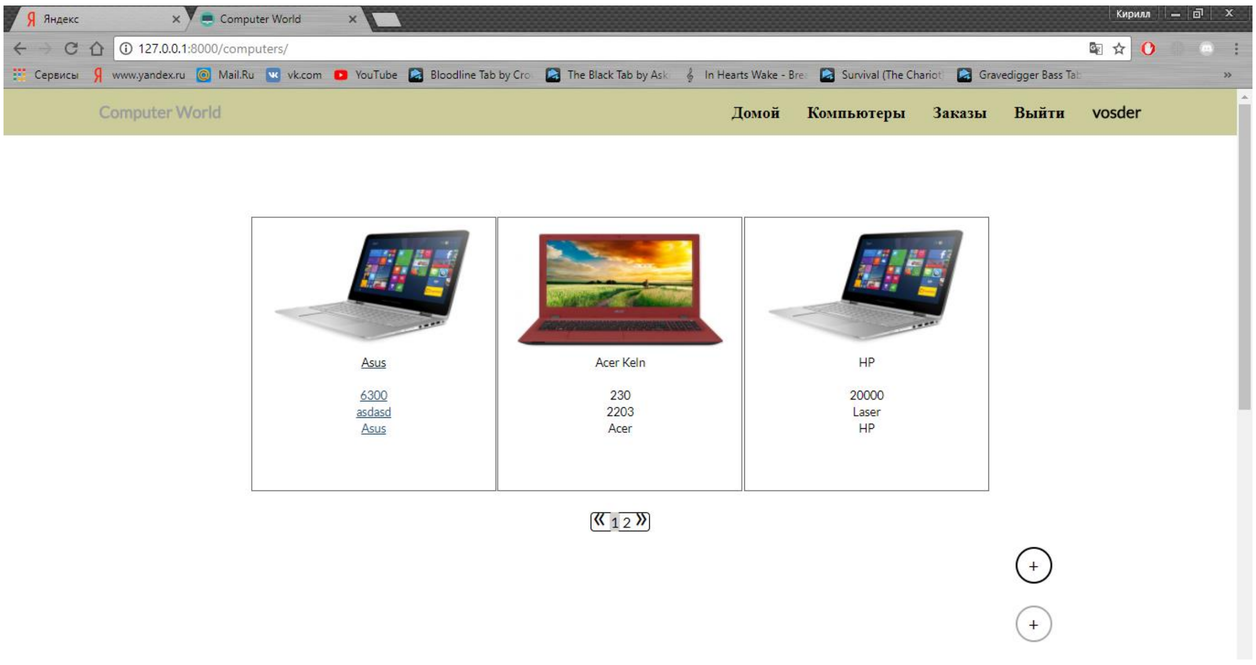
registration.py

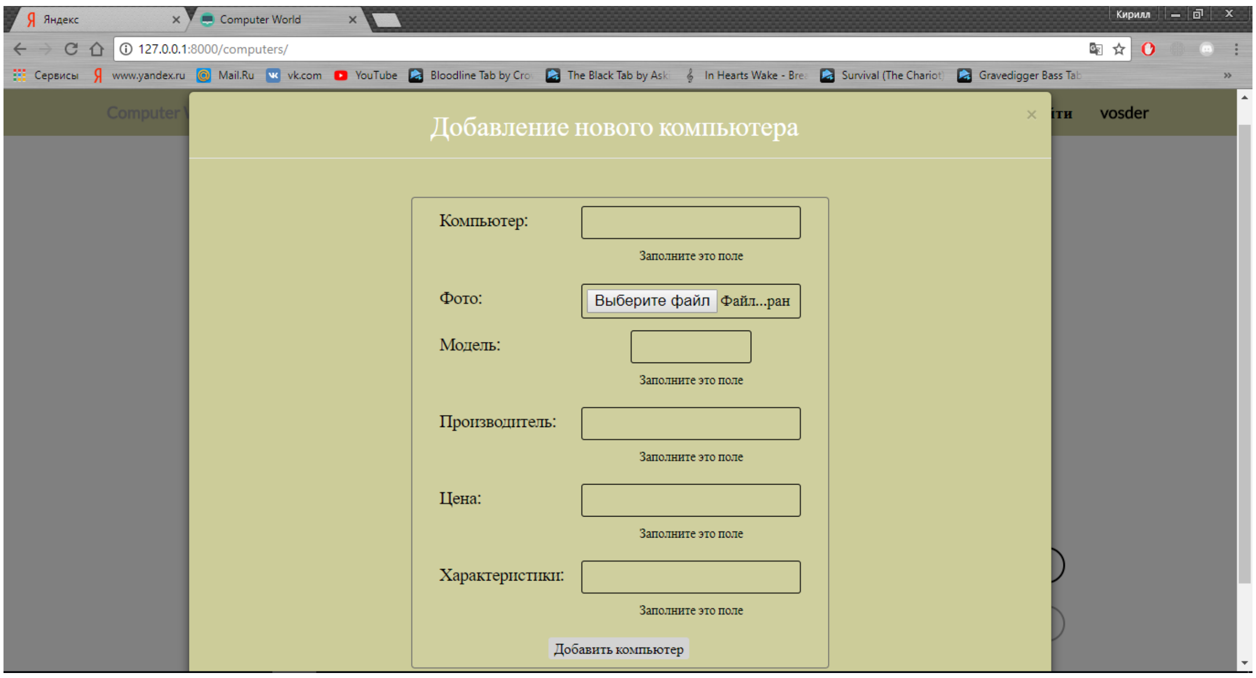
```
from django import forms


class RegistrationForm(forms.Form):
    username = forms.CharField(min_length=5, label='Логин')
    password = forms.CharField(min_length=6, widget=forms.PasswordInput,
label='Пароль')
    password2 = forms.CharField(min_length=6, widget=forms.PasswordInput,
label='Повторите пароль')
    email = forms.EmailField(widget=forms.EmailInput, label='E-mail')
    firstname = forms.CharField(label='Имя')
    surname = forms.CharField(label='Фамилия')
```

admin.py

```
from django.contrib import admin
from app.models import *

class OrderAdmin(admin.ModelAdmin):
    list_display = ['id_order']
    list_filter = ['id_comp']


class ComuterAdmin(admin.ModelAdmin):
    list_filter = ['company_computer']
    search_fields = ['name_computer', 'company_computer', 'model_computer']


admin.site.register(Order, OrderAdmin)
admin.site.register(Computer, ComuterAdmin)
admin.site.site_header = 'Django Администрирование'
admin.site.index_title = 'Администрирование'
```

<u>Результат:</u>





Добавление нового компьютера

Компьютер:

Заполните это поле

Фото: Выберите файл Файл...ран

Модель:

Заполните это поле

Производитель:

Заполните это поле

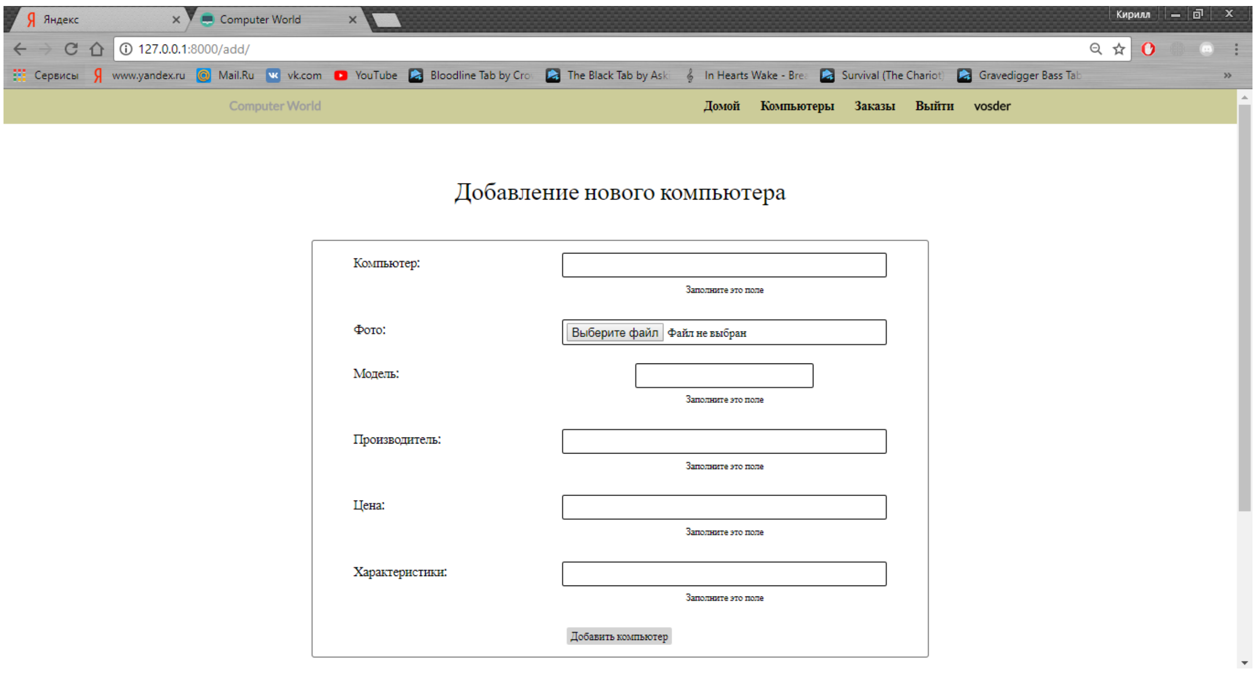
Цена:

Заполните это поле

Характеристики:

Заполните это поле

Добавить компьютер



Добавление нового компьютера

Компьютер:

Заполните это поле

Фото: Выберите файл Файл не выбран

Модель:

Заполните это поле

Производитель:

Заполните это поле

Цена:

Заполните это поле

Характеристики:

Заполните это поле

Добавить компьютер

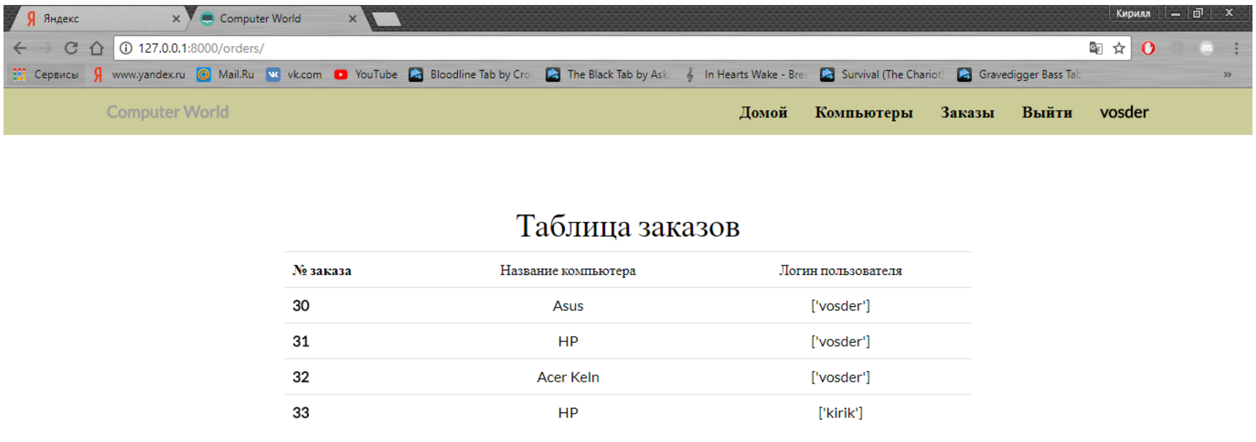
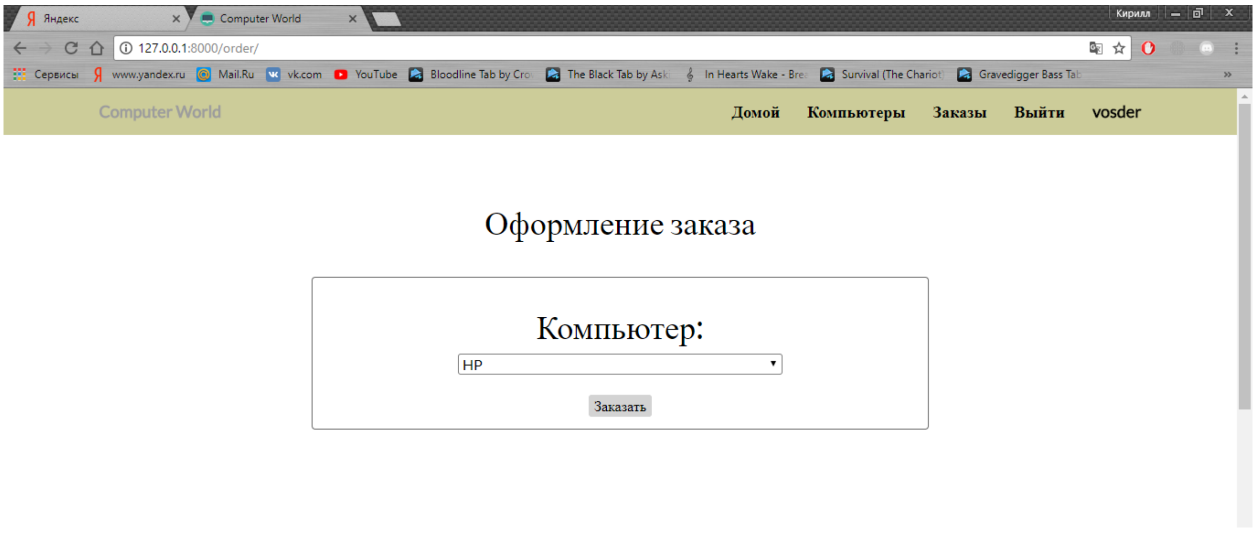


Таблица заказов

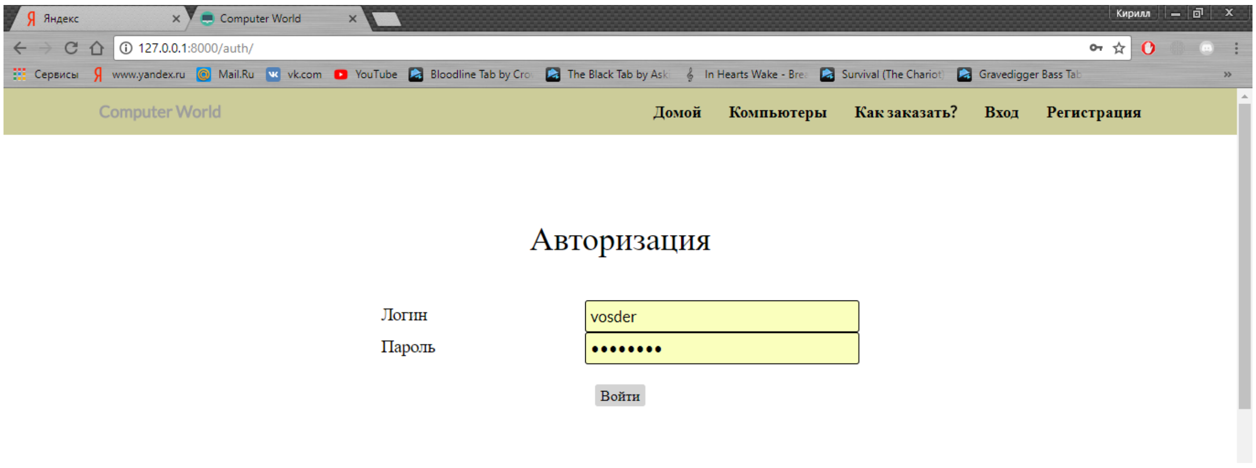
№ заказа	Название компьютера	Логин пользователя
30	Asus	['vosder']
31	HP	['vosder']
32	Acer Keln	['vosder']
33	HP	['kirik']



Оформление заказа

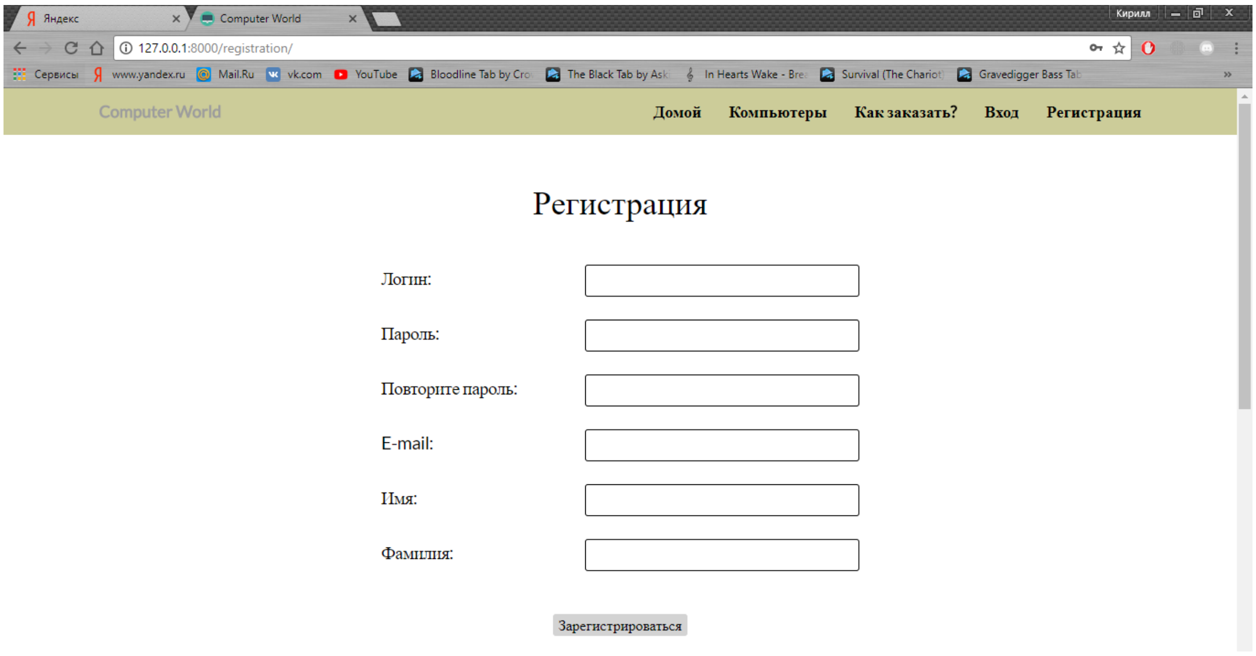
Компьютер:

HP



Авторизация

Логин	vosder
Пароль	••••••••



Computer World

Домой Компьютеры Как заказать? Вход Регистрация

Регистрация

Логин:	
Пароль:	
Повторите пароль:	
E-mail:	
Имя:	
Фамилия:	

Зарегистрироваться