Facultatea Calculatoare, Informatică și Microelectronică

Universitatea Tehnică a Moldovei

Medii Interactive de Dezvoltare a Produselor Soft

Lucrare de laborator #4

WEB DEVELOPMENT

Autor:

st. gr. TI-141

Buldumac oleg

lector asistent:

Irina Cojanu

lector superior:

Svetlana Cojocaru

LUCRARE DE LABORATOR #4

1. Scopul lucrării

Realizarea unui simplu Web Site personal.

2. Obiectivele lucrării

a) Realizarea unui simplu Web Site personal
b) Familiarizarea cu HTML si CSS
c) Interactiuni Javascript

3. Efectuarea lucrării de laborator

3.1. Task-uri implementate

1. Realizeaza un mini site cu 3 pagini statice
2. Site-ul trebuie sa pastreze toata informatia intr-o baza de date
3. Implimentarea XHR sau JSON responses. Careva din informatie trebuie sa fie dinamic incarcata pe pagina.

3.2. Analiza lucrării de laborator

https://github.com/buldumac/MIDPS.git

Fisierul models.py

```
from django.db import models
from django.contrib import admin


class Category(models.Model):
    title = models.CharField(max_length=150, unique=True)


    class Meta:
        ordering = ('title',)
        verbose_name = 'category'
        verbose_name_plural = 'categories'
```

```python
    def __str__(self):
        return self.title



class Post(models.Model):
    title = models.CharField(max_length = 150)
    stock = models.IntegerField()
    category = models.ForeignKey(Category, related_name="products", default=1)

    class Meta:
        ordering = ('-title',)

    def __str__(self):
        return self.title

class History(models.Model):
    def __str__(self):
        return self.title
```

Fișierul views.py

```python
# -*- coding: utf-8 -*-
from django.template.loader import get_template
from django.views.generic import TemplateView
from django.views.generic.edit import DeleteView
from django.template import RequestContext
from django.http import HttpResponse
from mydjapp.models import Category, Post
from django.shortcuts import render_to_response, get_object_or_404, redirect
from django.core.context_processors import csrf
from forms import AddCategoryForm, AddLotProductsForm
from django.contrib.auth.decorators import permission_required, login_required
from django.core.urlresolvers import reverse_lazy
from django.db.models import F
from utils import generic_search


@login_required(login_url='/login/')
def index(request):
        context = {
                'cats' : Category.objects.all(),
        'articlesallmain' : Post.objects.all().order_by('title'),
        'allgoodieslen' : len(Post.objects.all())
        }
        return render_to_response('base.html', context,
context_instance=RequestContext(request))



@login_required(login_url='/login/')
def category(request, category_id):
    latest_news_category_list =
Post.objects.filter(category=category_id).order_by('title')
    context = {
        'latest_news_category_list': latest_news_category_list,
        'cats' : Category.objects.all(),
        'category': get_object_or_404(Category, id=category_id)}
    return render_to_response('category.html', context,
context_instance=RequestContext(request))


@login_required(login_url='/login/')
@permission_required('mydjapp.add_category')
def cat_new(request):
    if request.method == 'GET':
        form = AddCategoryForm()
    else:
        form = AddCategoryForm(request.POST)
        if form.is_valid():
            title = form.cleaned_data['title']
            cat = Category.objects.create(title=title)
            return render_to_response('base.html', {'cats' : Category.objects.all()},
context_instance=RequestContext(request))

    return render_to_response('new_category.html', {'form' : form, 'cats' :
Category.objects.all()}, context_instance=RequestContext(request))

@login_required(login_url='/login/')
@permission_required('mydj.add_post')
def lot_news(request):
    if request.method == 'GET':
        formm = AddLotProductsForm()
    else:
        formm = AddLotProductsForm(request.POST)

        if formm.is_valid():
            alltxt = formm.cleaned_data
            tt = alltxt.get('txt')
            mm = tt.strip().split('\n')
            for i in mm:
                f, s, t = i.split('|')
                f, s, t = f.strip(), s.strip(), t.strip()
                postt = Post.objects.create(title=f, category=Category(id=s), stock=t)
            return render_to_response('base.html', {'cats' : Category.objects.all()},
context_instance=RequestContext(request))

    return render_to_response('lot_news.html', {'formm' : formm, 'cats' :
Category.objects.all()}, context_instance=RequestContext(request))

@login_required(login_url='/login/')
@permission_required("mydjapp.delete_category")
def cat_del(request, category_id):
        catt = Category.objects.get(id=category_id)

        context = {

                'caat' : catt,

        'cats' : Category.objects.all()

        }

        return render_to_response('confirm.html', context,
context_instance=RequestContext(request))


@login_required(login_url='/login/')

@permission_required("mydjapp.delete_category")

def cat_del_confirm(request, category_id):

    if request.method == 'GET':

        Category.objects.get(id=category_id).delete()

        return redirect('index')

    return render_to_response('confirm.html', {'cats' : Category.objects.all()},
context_instance=RequestContext(request))


@login_required(login_url='/login/')

def post_sell(request, post_id):

        posttt = Post.objects.get(id=post_id)

        context = {

                'poost' : posttt,

        'cats' : Category.objects.all()

        }

        return render_to_response('confirmPost.html', context,
context_instance=RequestContext(request))


@login_required(login_url='/login/')

def post_sell_confirm(request, post_id):

    if request.method == 'GET':

        stcount = Post.objects.filter(id=post_id).update(stock=F('stock') - 1)

        return redirect('index')

    return render_to_response('confirmPost.html', {'cats' : Category.objects.all()},
context_instance=RequestContext(request))
```

```
def error404(request):
    return render_to_response('404.html', status=404)


QUERY="q"


MODEL_MAP = { Post: ["title",], }


@login_required(login_url='/login/')
def search(request):


   objects = []


   for model,fields in MODEL_MAP.iteritems():
       objects += generic_search(request,model,fields,QUERY)


   return render_to_response("search_results.html",
                             {"objects":objects,
                              "cats" : Category.objects.all(),
                              "search_string" : request.GET.get(QUERY,""),},
context_instance=RequestContext(request))


@login_required(login_url='/login/')
@permission_required("mydjapp.delete_post")
def post_delete(request, post_id):
        posttt = Post.objects.get(id=post_id)
        context = {
                'poostd' : posttt,
        'cats' : Category.objects.all()
        }
        return render_to_response('confirmPostDel.html', context,
context_instance=RequestContext(request))
```

```
@login_required(login_url='/login/')
@permission_required("mydjapp.delete_post")
def post_delete_confirm(request, post_id):
    if request.method == 'GET':
        stcount = Post.objects.filter(id=post_id).delete()
        return redirect('index')
    return render_to_response('confirmPost.html', {'cats' : Category.objects.all()},
context_instance=RequestContext(request))
```

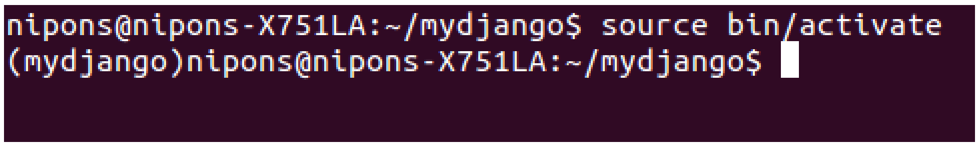
Fișierul urls.py (Controlam URL-urile)

```
# -*- coding: utf-8 -*-
"""mydj URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.9/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import *
from django.contrib import admin
from mydjapp.views import index, category, cat_new, cat_del, cat_del_confirm,
lot_news, post_sell, post_sell_confirm, search, post_delete, post_delete_confirm


urlpatterns = [
    url(r'^category/(?P<category_id>\d+)/$', category, name='category'),
    url(r'^admin/', admin.site.urls),
    url(r'^$', index, name='index'),
    url(r'^search/$', search, name='search'),
    url(r'^category/new/$', cat_new, name='cat_new'),
    url(r'^lot_products/$', lot_news, name='lot_news'),
    url(r'^category/delete/(?P<category_id>\d+)/$', cat_del, name="cat_del"),
    url(r'^category/delete/(?P<category_id>\d+)/confirmation/$', cat_del_confirm,
name="cat_del_confirm"),
    url(r'^post/sell/(?P<post_id>\d+)/$', post_sell, name="post_sell"),
    url(r'^post/sell/(?P<post_id>\d+)/confirmation/$', post_sell_confirm,
name="post_sell_confirm"),
    url(r'^post/delete/(?P<post_id>\d+)/$', post_delete, name="post_delete"),
    url(r'^post/delete/(?P<post_id>\d+)/confirmation/$', post_delete_confirm,
name="post_delete_confirm"),
    url(r'^login/', "django.contrib.auth.views.login", {"template_name":"login.html"},
name="login"),
    url(r'^logout/', "django.contrib.auth.views.logout",{"template_name":"logout.html"},
name="logout"),
]
```

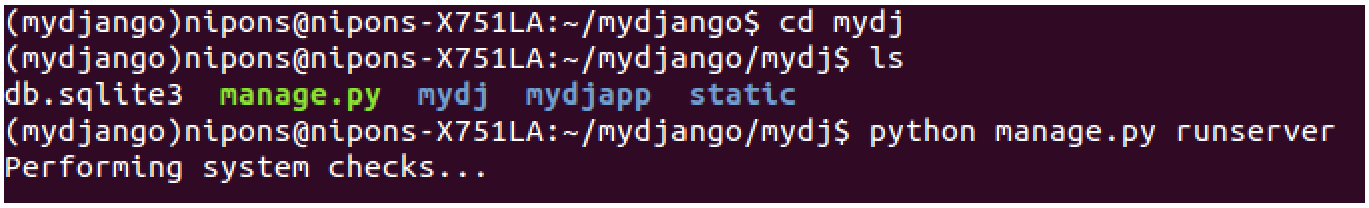
Mai întâi de toate activam **virtualenv** cu ajutorul urmatoarei comenzi:



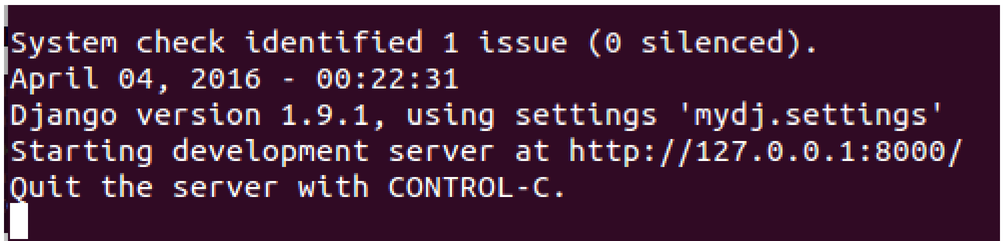
```
nipons@nipons-X751LA:~/mydjango$ source bin/activate
(mydjango)nipons@nipons-X751LA:~/mydjango$
```

După activare ne deplasam în folder-ul unde se afla fișierul **manage.py** care conduce cu Django.

Cu ajutorul comenzei **python manage.py runserver** pornim web-siteul pe localhost pe portul 8000:



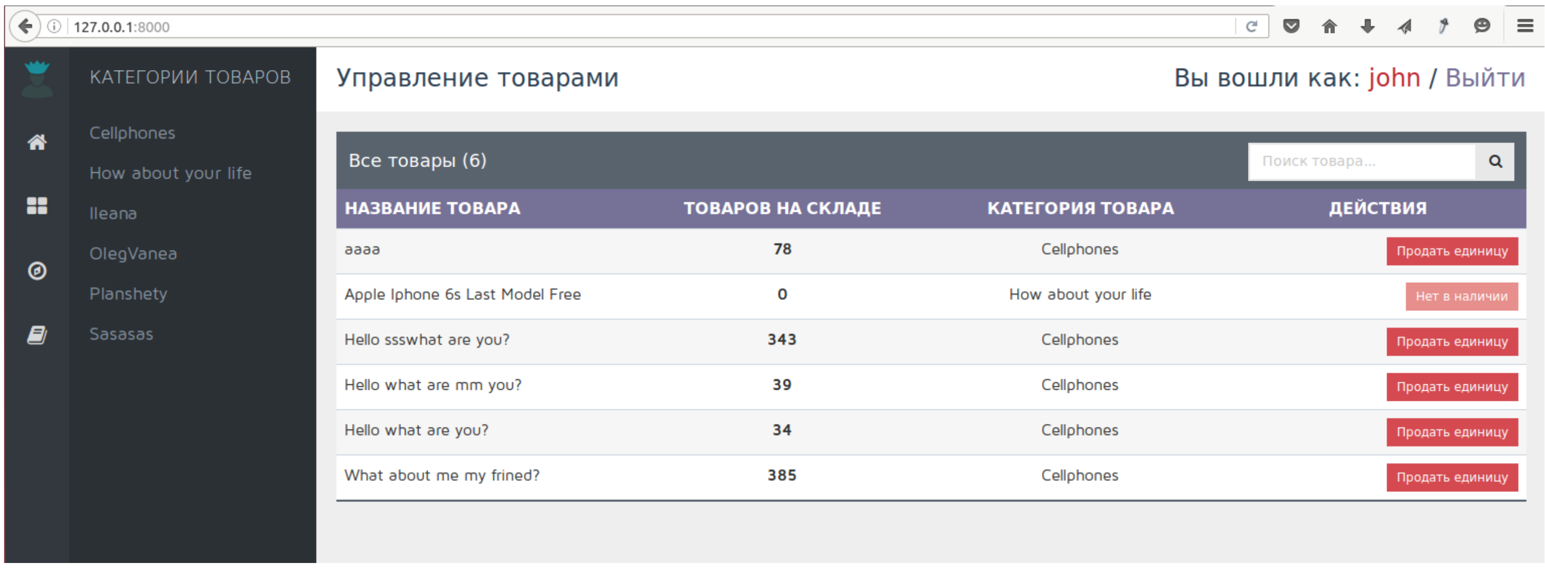
```
(mydjango)nipons@nipons-X751LA:~/mydjango$ cd mydj
(mydjango)nipons@nipons-X751LA:~/mydjango/mydj$ ls
db.sqlite3  manage.py  mydj  mydjapp  static
(mydjango)nipons@nipons-X751LA:~/mydjango/mydj$ python manage.py runserver
Performing system checks...
```



```
System check identified 1 issue (0 silenced).
April 04, 2016 - 00:22:31
Django version 1.9.1, using settings 'mydj.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

După cum se vede, serveul este pornit cu success pe 127.0.0.1:8000.

Site-ul s-a deschis cu success. Orice redactare corecta a fisierilor models.py, views.py … s.a.m.d. nu vor duce la careva deteoriari cu web-siteul. În caz când se întâlnesc greșeli de cod sau de sintaxa în consola putem vedea ce anume s-a întâmplat. În acest caz web-siteul nu va funcționa.



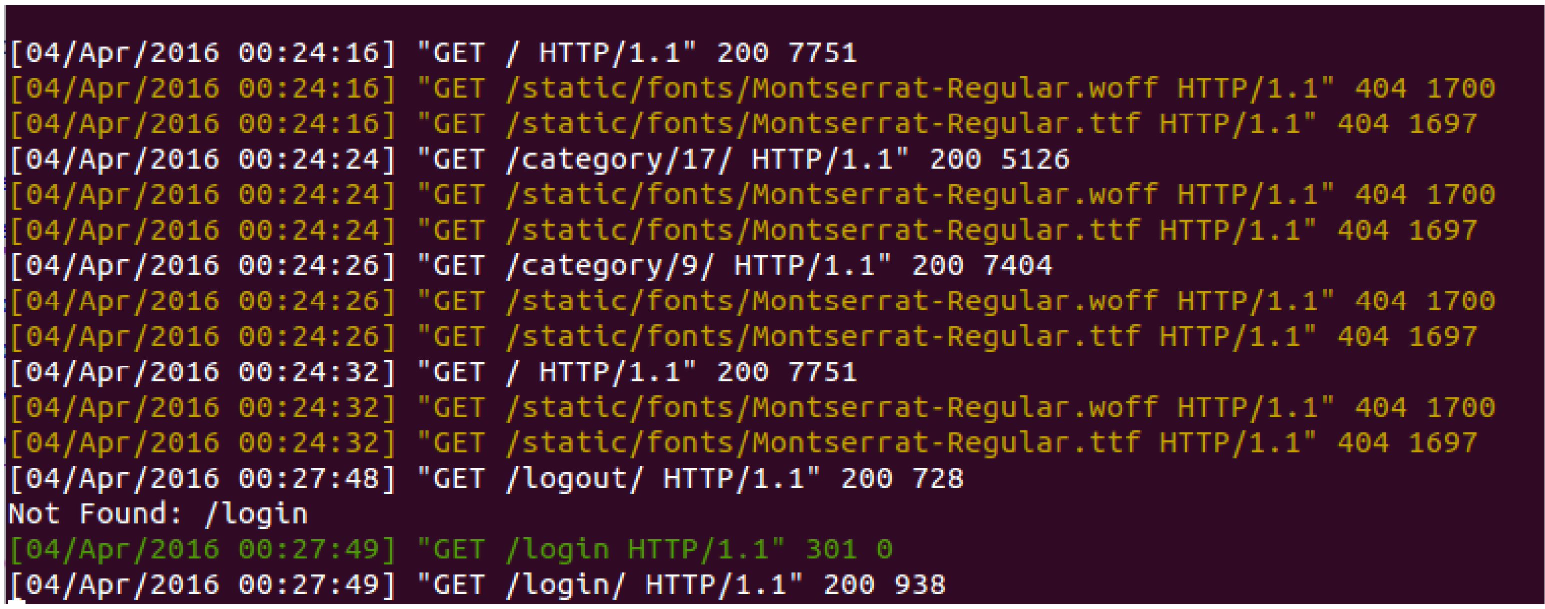
Управление товарами

Вы вошли как: john / Выйти

Все товары (6)

НАЗВАНИЕ ТОВАРА	ТОВАРОВ НА СКЛАДЕ	КАТЕГОРИЯ ТОВАРА	ДЕЙСТВИЯ
aaaa	78	Cellphones	Продать единицу
Apple Iphone 6s Last Model Free	0	How about your life	Нет в наличии
Hello ssswhat are you?	343	Cellphones	Продать единицу
Hello what are mm you?	39	Cellphones	Продать единицу
Hello what are you?	34	Cellphones	Продать единицу
What about me my frined?	385	Cellphones	Продать единицу

În consola putem observa toate requesturile la server unde respectiv serverul răspunde:



```
[04/Apr/2016 00:24:16] "GET / HTTP/1.1" 200 7751
[04/Apr/2016 00:24:16] "GET /static/fonts/Montserrat-Regular.woff HTTP/1.1" 404 1700
[04/Apr/2016 00:24:16] "GET /static/fonts/Montserrat-Regular.ttf HTTP/1.1" 404 1697
[04/Apr/2016 00:24:24] "GET /category/17/ HTTP/1.1" 200 5126
[04/Apr/2016 00:24:24] "GET /static/fonts/Montserrat-Regular.woff HTTP/1.1" 404 1700
[04/Apr/2016 00:24:24] "GET /static/fonts/Montserrat-Regular.ttf HTTP/1.1" 404 1697
[04/Apr/2016 00:24:26] "GET /category/9/ HTTP/1.1" 200 7404
[04/Apr/2016 00:24:26] "GET /static/fonts/Montserrat-Regular.woff HTTP/1.1" 404 1700
[04/Apr/2016 00:24:26] "GET /static/fonts/Montserrat-Regular.ttf HTTP/1.1" 404 1697
[04/Apr/2016 00:24:32] "GET / HTTP/1.1" 200 7751
[04/Apr/2016 00:24:32] "GET /static/fonts/Montserrat-Regular.woff HTTP/1.1" 404 1700
[04/Apr/2016 00:24:32] "GET /static/fonts/Montserrat-Regular.ttf HTTP/1.1" 404 1697
[04/Apr/2016 00:27:48] "GET /logout/ HTTP/1.1" 200 728
Not Found: /login
[04/Apr/2016 00:27:49] "GET /login HTTP/1.1" 301 0
[04/Apr/2016 00:27:49] "GET /login/ HTTP/1.1" 200 938
```

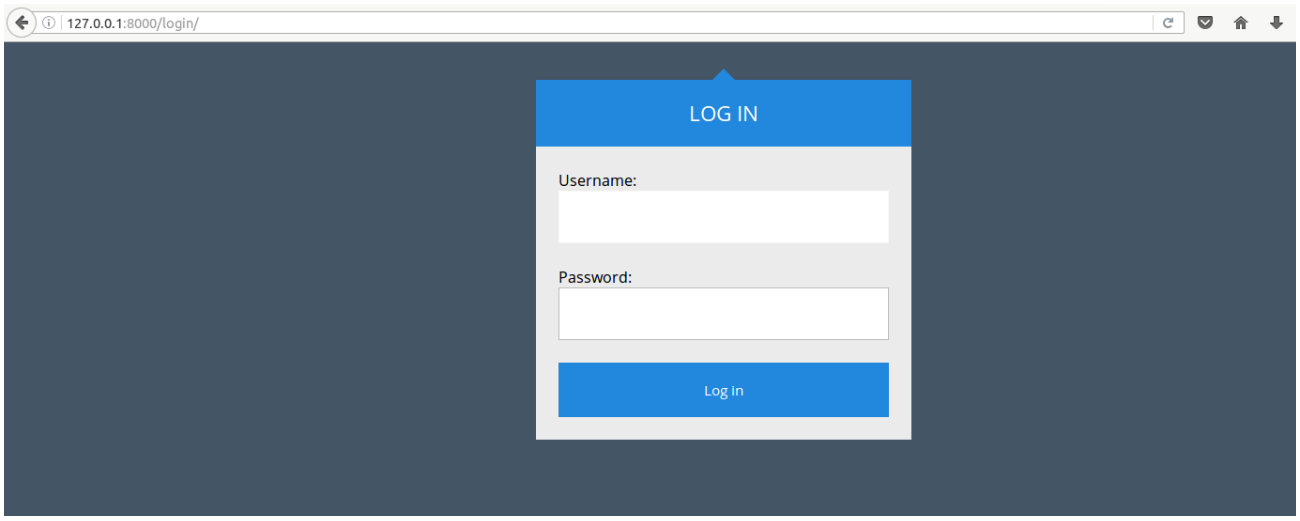
200 → Success

404 → Nu a fost găsit, în cazul nostru un FONT nu exista

Se folosește doar requesturi de tip GET

7751, 5126, 728 → Numărul de bytes, adică lungimea paginii HTML

Am elaborat sistema de logare Administrator / Vânzător:



Unde vinzatorul are mult mai puține drepturi fata de administrator.

Concluzie:

În aceasta lucrare de laborator am creat un web-site dinamic, unde se interactioneaza cu baza MySQL. Am folosit tehnologia Django care este bazata pe Python. În linux foarte ușor de deschis server pentru Django cu ajutorul Virtualenv-ului care instaleaza django doar într-o anumita mapa locala ci nu global în toată sistema de operare.

Web-siteul are și sistema de logare cu drepturi. Unde administratorul poate sterge, adauga, modifica datele pe când vinzatorul poate doar sa reducă din stock.

Pentru infatisarea web-siteului am folosit HTML cu CSS, și anume Bootstrap, care este un salvator de timp in web development.

Am ales limbajul Python din aspectul ca codul este foarte simplu, înțelegător și Django este ceva nou pentru mine.

Am folosit un template global main.html care a fost extins practic de toate fisierele html create.

Bibliografie

1. http://www.tutorialspoint.com/python/
2. http://www.learnpython.org/
3. https://www.codecademy.com/learn/python
4. https://www.codementor.io/python/tutorial
5. https://ru.wikipedia.org/wiki/Django