

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«Национальный исследовательский университет ИТМО»
Факультет инфокоммуникационных технологий

Лабораторная работа №2

«Реализация простого сайта средствами Django»

по дисциплине

«Web-программирование»

Выполнил:

студент III курса ФИКТ

группы K33402

Ф.И.О. Кондрашов Егор Юрьевич

Проверил:

Говоров А. И.

Санкт-Петербург

2021

Цель работы: овладеть практическими навыками и умениями реализации web-сервисов средствами Django 3.2.

Выполнение работы:

В ходе выполнения курса “Harvard's CS50's Web Programming with Python and JavaScript”, я написал простой веб-сервис на Django, который позволяет участвовать в аукционах на продажу вещей (аналог ebay).

Сервис имеет следующий функционал:

1. Регистрация и авторизация пользователей.
2. Просмотр списка аукционов, категорий, аукционов в категории, конкретного аукциона.
3. Добавление аукциона в watch list, просмотр своего watch list'a.
4. Добавление комментариев к аукциону.
5. Предложение цены на аукцион.
6. Создание своего аукциона, закрытие своего аукциона с автоматическим определением победителя.

Исходный код файлов:

Содержание auctions/views.py (описание конкретных Views представлено в doc-strings):

```
from django.contrib.auth import login
from django.contrib.auth.mixins import LoginRequiredMixin
from django.db import IntegrityError
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.urls import reverse
from django.views.generic import ListView, CreateView, DetailView
from django.contrib.auth.decorators import login_required

from .models import Listing, ListingComment, ListingCategory, Bid,
WatchList, User
from .forms import ListingCommentForm

class ListingList(ListView):
```

```

    """Список всех вещей, выставленных на аукцион"""
    model = Listing
    template_name = "auctions/index.html"

    def get_queryset(self):
        listings = Listing.objects.filter(active=True)
        return listings

class CategoryList(ListView):
    """Список всех категорий аукционов"""
    model = ListingCategory
    template_name = "auctions/categories.html"

class WatchListList(LoginRequiredMixin, ListView):
    """Список всех аукционов, добавленных в watch-list текущего
пользователя"""
    model = Listing
    template_name = 'auctions/watchlist.html'

    def get_queryset(self):
        w = self.request.user.watchlist
        return w.items.all()

class CreateListing(LoginRequiredMixin, CreateView):
    """View для выставления вещи на аукцион"""
    model = Listing
    fields = ['title', 'description', 'starting_bid', 'image_url',
              'category']

    def form_valid(self, form):
        form.instance.created_by = self.request.user
        return super().form_valid(form)

class ListingDetail(DetailView):
    """View для просмотра информации по конкретному аукциону"""
    model = Listing

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

```

```

        listing = Listing.objects.get(pk=self.kwargs['pk'])

        if self.request.user == listing.created_by:
            is_author = True
        else:
            is_author = False

        if self.request.user == listing.winner:
            is_winner = True
        else:
            is_winner = False

        comments = ListingComment.objects.filter(original_listing=listing).all()

        context['is_winner'] = is_winner
        context['is_author'] = is_author
        context['form'] = ListingCommentForm()
        context['comments'] = comments
        return context

def register(request):
    """View для регистрации нового пользователя"""
    if request.method == "POST":
        username = request.POST["username"]
        email = request.POST["email"]

        # Ensure password matches confirmation
        password = request.POST["password"]
        confirmation = request.POST["confirmation"]
        if password != confirmation:
            return render(request, "auctions/register.html", {
                "message": "Passwords must match."
            })

        # Attempt to create new user
        try:
            user = User.objects.create_user(username, email,
password)
            user.save()
            watchlist = WatchList(owner=user)
            watchlist.save()

```

```

        except IntegrityError:
            return render(request, "auctions/register.html", {
                "message": "Username already taken."
            })
        login(request, user)
        return HttpResponseRedirect(reverse("index"))
    else:
        return render(request, "auctions/register.html")

@login_required
def add_to_watchlist(request, pk):
    """View для добавления аукциона в Watch list пользователя"""
    watchlist = request.user.watchlist
    watchlist.items.add(Listing.objects.get(pk=pk))
    return HttpResponseRedirect(reverse('listing_detail',
kwargs={'pk': pk}))

@login_required
def remove_from_watchlist(request, pk):
    """View для удаления аукциона из Watch list пользователя"""
    watchlist = request.user.watchlist
    watchlist.items.remove(Listing.objects.get(pk=pk))
    return HttpResponseRedirect(reverse('listing_detail',
kwargs={'pk': pk}))

@login_required
def new_bid(request, pk):
    """View для создания нового предложения цены на аукционе"""
    if request.method == 'POST':
        listing = Listing.objects.get(pk=pk)
        error = ''
        try:
            value = float(request.POST['bid_value'])

            if value <= listing.starting_bid:
                error = 'Your bid should be higher than the
starting value!'

            if listing.current_price():
                if value <= listing.current_price():

```

```

        error = 'Your bid should be higher than other
bids for this listing!'
    except ValueError:
        error = 'You have to type in a valid number'

    if error:
        request.session['new_bid_error'] = error
    else:
        bid = Bid.objects.create(bidder=request.user,
                                listing=listing,
                                value=value)

        bid.save()
        return HttpResponseRedirect(reverse('listing_detail',
kwargs={'pk': pk}))
    else:
        return HttpResponseRedirect(reverse('listing_detail',
kwargs={'pk': pk}))

@login_required
def close_listing(request, pk):
    """View для закрытия аукциона по вещи"""
    listing = Listing.objects.get(pk=pk)
    listing.active = False
    listing.winner = listing.highest_bidder()
    listing.save()
    return HttpResponseRedirect(reverse('listing_detail',
kwargs={'pk': pk}))

@login_required
def make_comment(request, pk):
    """View для создания комментария к аукциону"""
    listing = Listing.objects.get(pk=pk)
    form = ListingCommentForm(request.POST)
    if form.is_valid():
        comment = form.save(commit=False)
        comment.original_listing = listing
        comment.author = request.user
        comment.save()
        return HttpResponseRedirect(reverse('listing_detail',
kwargs={'pk': pk}))

```

```

class ViewCategory(ListView):
    """View для получения списка аукционов по конкретной
категории"""
    model = Listing
    template_name = "auctions/view_category.html"

    def get_queryset(self):
        pk = self.kwargs['pk']
        category = ListingCategory.objects.get(pk=pk)
        listings = Listing.objects.filter(active=True).filter(category=category)
        return listings

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        pk = self.kwargs['pk']
        category = ListingCategory.objects.get(pk=pk)
        context['category'] = category.name
        return context

```

Содержание файла forms.py:

```

from django import forms
from .models import ListingComment

class ListingCommentForm(forms.ModelForm):
    class Meta:
        model = ListingComment
        fields = ['text']

```

Содержание файла models.py (описание моделей представлено в doc strings, описание полей - в параметре verbose_name):

```

from django import VERSION
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.urls import reverse

class User(AbstractUser):
    """Кастомная модель юзера, если возникнет необходимость

```

```

        добавлять дополнительные поля"""
    pass

class ListingCategory(models.Model):
    """Модель категории аукционов"""
    name = models.CharField(
        max_length=128,
        verbose_name="название"
    )

    def __str__(self) -> str:
        return f"{self.name}"

    class Meta:
        verbose_name = "категория аукциона"
        verbose_name_plural = "категории аукционов"

class Listing(models.Model):
    """Модель вещи, выставленной на аукцион"""

    title = models.CharField(
        max_length=128,
        verbose_name="название"
    )

    description = models.TextField(
        verbose_name="описание"
    )

    starting_bid = models.DecimalField(
        decimal_places=2,
        max_digits=15,
        verbose_name="начальная стоимость"
    )

    image_url = models.URLField(
        blank=True,
        verbose_name="ссылка на изображение"
    )

    category = models.ForeignKey(

```



```

        ListingCategory,
        on_delete=models.PROTECT,
        related_name='listings',
        null=True,
        blank=True,
        verbose_name="категория"
    )

    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name="время создания"
    )

    created_by = models.ForeignKey(
        User,
        on_delete=models.PROTECT,
        related_name='listings',
        verbose_name="создатель"
    )

    active = models.BooleanField(
        default=True,
        verbose_name="активен?"
    )

    winner = models.ForeignKey(
        User,
        on_delete=models.PROTECT,
        related_name='winner',
        null=True,
        blank=True,
        verbose_name="победитель аукциона"
    )

    def __str__(self) -> str:
        return self.title

    def get_absolute_url(self):
        return reverse('listing_detail', kwargs={'pk': self.pk})

    def highest_bidder(self):
        highest_bid =
Bid.objects.filter(listing=self).order_by('-value')

```

```

        if highest_bid:
            return highest_bid[0].bidder
        else:
            return None

    def current_price(self):
        highest_bid =
Bid.objects.filter(listing=self).order_by('-value')
        if highest_bid:
            return highest_bid[0].value
        else:
            return None

class Meta:
    verbose_name = "вещь, выставленная на аукцион"
    verbose_name_plural = "вещи, выставленные на аукцион"

class ListingComment(models.Model):
    """Модель комментария к аукциону"""

    original_listing = models.ForeignKey(
        Listing,
        on_delete=models.PROTECT,
        related_name='comments',
        verbose_name="вещь на аукционе"
    )

    text = models.TextField(
        verbose_name="текст"
    )

    author = models.ForeignKey(
        User,
        on_delete=models.PROTECT,
        related_name='comments',
        verbose_name="автор"
    )

    rating = models.IntegerField(default=1)
    posted = models.DateTimeField(auto_now_add=True)

    def __str__(self) -> str:

```

```

        s = f'On {self.original_listing} by {self.author} with
{self.rating}'

        return s

class Meta:
    verbose_name = "комментарий к аукциону"
    verbose_name_plural = "комментарии к аукциону"

class Bid(models.Model):
    """Модель предложения цены на аукционе"""

    bidder = models.ForeignKey(
        User,
        on_delete=models.PROTECT,
        related_name='bids',
        verbose_name="участник аукциона"
    )

    listing = models.ForeignKey(
        Listing,
        on_delete=models.PROTECT,
        related_name='bids',
        verbose_name="вещь на аукционе"
    )

    value = models.DecimalField(
        decimal_places=2,
        max_digits=15,
        verbose_name="предлагаемая цена"
    )

    created = models.DateTimeField(
        auto_now_add=True,
        verbose_name="время создания"
    )

    def __str__(self) -> str:
        return f'{self.bidder} {self.value}'

class Meta:
    verbose_name = "предложение цены"
    verbose_name_plural = "предложения цены"

```

```

class WatchList(models.Model):
    """Модель списка аукционов, за которыми следит пользователь"""

    owner = models.OneToOneField(
        User,
        on_delete=models.CASCADE,
        verbose_name="пользователь"
    )

    items = models.ManyToManyField(
        Listing,
        verbose_name="вещи на аукционе"
    )

    def __str__(self) -> str:
        return f"{self.owner}'s watchlists"

    class Meta:
        verbose_name = "watch list пользователя"
        verbose_name_plural = "watch list'ы пользователей"

```

Содержание файла urls.py:

```

from django.urls import path
from django.contrib.auth import views as auth_views

from . import views

urlpatterns = [
    path("", views.ListingList.as_view(), name="index"),
    path("login/",
auth_views.LoginView.as_view(template_name='auctions/login.html'),
name="login"),
    path("logout", auth_views.LogoutView.as_view(), name="logout"),
    path("register", views.register, name="register"),
    path('listing/create', views.CreateListing.as_view(),
name='create_listing'),
    path('listing/<int:pk>/', views.ListingDetail.as_view(),
name='listing_detail'),
    path('add_to_watchlist/<int:pk>',
views.add_to_watchlist, name='add_to_watchlist'),

```

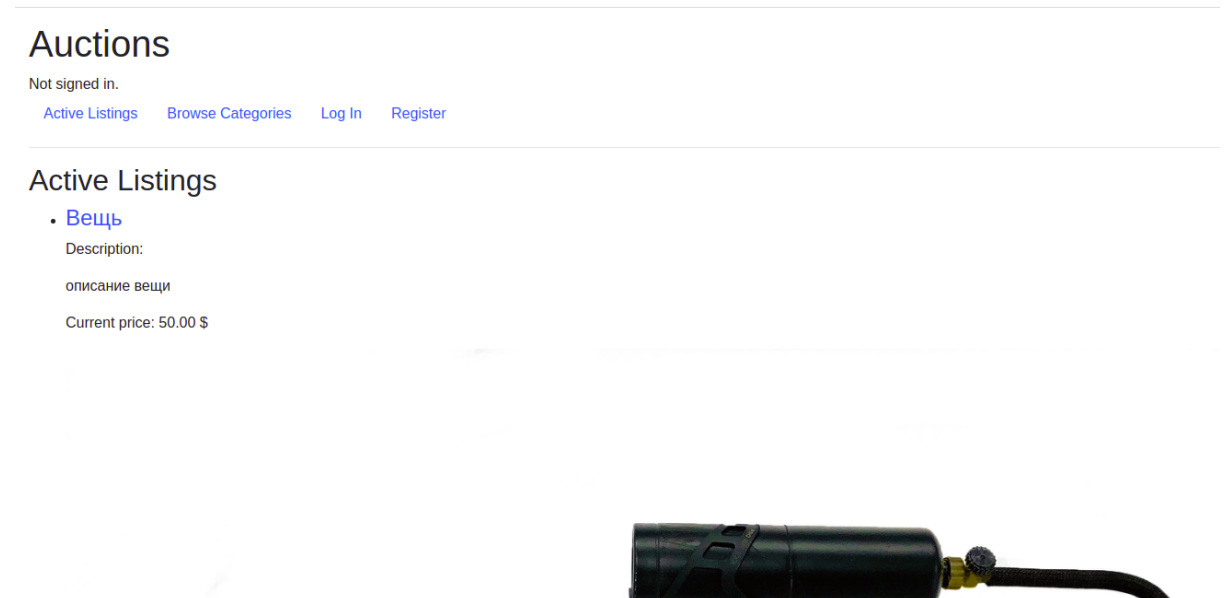
```

        path('remove_from_watchlist/<int:pk>',
              views.remove_from_watchlist,
name='remove_from_watchlist'),
        path('watchlist', views.WatchListList.as_view(),
name='watchlist'),
        path('new_bid/<int:pk>', views.new_bid, name='new_bid'),
        path('close_listing/<int:pk>', views.close_listing,
name='close_listing'),
        path('make_comment/<int:pk>', views.make_comment,
name='make_comment'),
        path('categories', views.CategoryList.as_view(),
name='categories'),
        path('categories/<int:pk>', views.ViewCategory.as_view(),
name='view_category'),
    ]

```

Скриншоты веб-сервиса:

Главная страница со списком аукционов:



Список категорий:

Auctions

Not signed in.

[Active Listings](#)

[Browse Categories](#)

[Log In](#)

[Register](#)

Categories

[Food](#)

[Collectibles](#)

Аукционы в категории:

Auctions

Not signed in.

[Active Listings](#)

[Browse Categories](#)

[Log In](#)

[Register](#)

Active Listings in the category Food

- [Spaghetto](#)

Description:

a pack of spaghet

Current price: 2.00 \$



Вход:

Auctions

Not signed in.

[Active Listings](#) [Browse Categories](#) [Log In](#) [Register](#)

Login

Login

Don't have an account? [Register here.](#)

Регистрация:

Auctions

Not signed in.

[Active Listings](#) [Browse Categories](#) [Log In](#) [Register](#)

Register

Register

Already have an account? [Log In here.](#)

Watch list:

Auctions

Signed in as **e.kondr01**.

[Active Listings](#) [Browse Categories](#) [Create listing](#) [My Watchlist](#) [Log Out](#)

My Watchlist

- [Вещь](#)

Description:

описание вещи

Current price: 50.00 \$



Страница отдельного аукциона:

Auctions

Signed in as **e.kondr01**.

[Active Listings](#) [Browse Categories](#) [Create listing](#) [My Watchlist](#) [Log Out](#)

This is your listing. You can [close this listing](#)

Вещь

Description: описание вещи

Starting bid: 50.00\$

No bids yet. Be the first one to make it!

My bid: \$



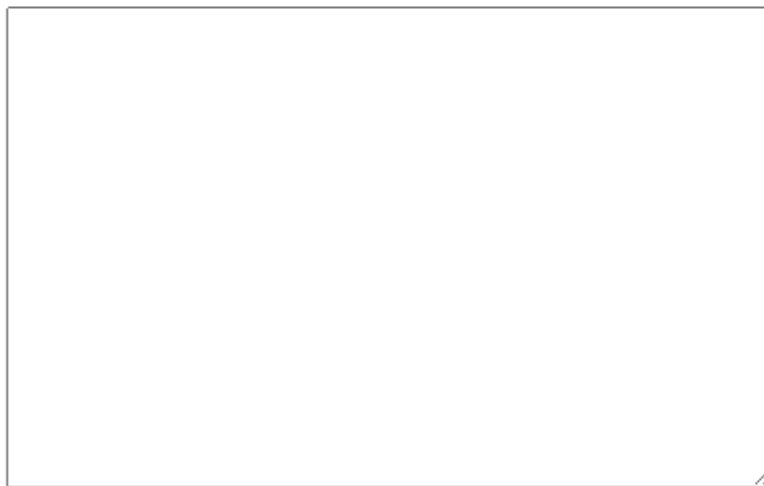
Categories:

Collectibles

Created by: e.kondr01

Created at: Oct. 31, 2021, 10:26 a.m.

Текст:



Submit

[Remove from my watchlist](#)

Создание аукциона:

Auctions

Signed in as **e.kondr01**.

[Active Listings](#)

[Browse Categories](#)

[Create listing](#)

[My Watchlist](#)

[Log Out](#)

Название:

Описание:

Начальная стоимость:

Ссылка на изображение:

Категория:



Save

Вывод: с помощью Django 3.2 был реализован простой веб-сервис, который позволяет пользователям принимать участие в аукционах (аналог ebay).