Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

Отчёт по домашней работе

по курсу

«Разработка интернет-приложений»

Вебсервис на базе технологий: Python, Django, JS, MySQL

Исполнитель: студентка группы **РТ5-51**

Галичий Д.А.

Преподаватель: **Гапанюк Ю.Е.**

Москва, 2017

Цель работы: разработать вебсервис на базе технологий: Python, Django, JS, MySQL.

Предметная область: субъект: пользователь, объект: фильм, отношение: отзыв.

Содержание файла «settings.py»:

```
"""
Django settings for lab6 project.

Generated by 'django-admin startproject' using Django 1.11.7.

For more information on this file, see
https://docs.djangoproject.com/en/1.11/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.11/ref/settings/
"""

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 = 'y2a(#+h0c6dui573k5s@9i@=d0f&13#*erc*i3y(@14&=h(*$m'

# 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',
    'lab6.apps.db_app.apps.DbAppConfig',
    #'lab6.apps.db_app',
]

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 = 'lab6.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'lab6/apps/db_app/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 = 'lab6.wsgi.application'


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

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'films',
        'USER': 'dbuser',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': 3306,
        'OPTIONS': {'charset': 'utf8'},
        'TEST_CHARSET': 'utf8',
    }
}


# 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 = 'ru-ru'

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/'

MEDIA_ROOT = 'C:\\Users\\Дарья\\PycharmProjects\\HW\\static\\new_imgs'
MEDIA_URL = '/static/new_imgs/'
```

Содержание файла «lab6\urls.py»:

```
"""lab6 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.11/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 url, include
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^hw_app/', include('lab6.apps.db_app.urls')),
]
```

Содержание файла «db_app\urls.py»:

```
from django.conf.urls import url
from lab6.apps.db_app import views


urlpatterns = [
    url(r'^main/', views.main, name='main_url'),
    url(r'^registration2/', views.registration2, name='reg2_url'),
    url(r'^login/', views.login, name='login_url'),
    url(r'^account/', views.account, name='account_url'),
    url(r'^logout/', views.logout, name='logout_url'),
    url(r'^film/get/(?P<pk>\d+)/$', views.FilmDetailView.as_view(),
name='film_url'),
    url(r'^page/(\d+)/$', views.main, name='page_url'),
]
```

Содержание файла «views.py»:

```
from django.shortcuts import render, HttpResponseRedirect, render_to_response
from django.views.generic import ListView, DetailView
from lab6.apps.db_app.models import *
from django.contrib.auth.models import User
from lab6.apps.db_app.registration import *
from django.contrib import auth
from django.contrib.auth import authenticate
from django.views.generic.edit import CreateView
from django.forms.models import ModelForm
from django.contrib.auth.decorators import login_required
from django.urls import reverse
from django.core.paginator import Paginator

# Create your views here.


def main(request, page_number=1):
    films_all = Films.objects.all()
    current_page = Paginator(films_all, 2)
    return render_to_response('main_page.html', {'films':
current_page.page(page_number)})


class FilmDetailView(DetailView):
    model = Films
    context_object_name = 'film'
    template_name = 'film.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['comments'] = Comments.objects.all()
        return context

    def get_object(self):
        object = super(FilmDetailView, self).get_object()
        return object


def film(request, film_id=1):
    return render(request, 'film.html', {'film':
Films.objects.get(film_id=film_id), 'comments':
Comments.objects.filter(comment_film_id=film_id)})


class CommentForm(ModelForm):
    class Meta:
        model = Comments
        exclude = ['comment_id']


class CommentCreate(CreateView):
    form_class = CommentForm
    template_name = 'film.html'

    def post(self, request):
        new_comm = Comments(comment_user=request.user,
comment_film=request.form.film,
                            comment_text=request.form.comment_text,
                              comment_photo=request.form.comment_photo)
        new_comm.save()
        return super().post(request)

    def get(self, request):
        if not self.request.user.is_authenticated:
            return HttpResponseRedirect('/hw_app/main/')
        return super().get(request)


def registration2(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'),

last_name=request.POST.get('surname'),

first_name=request.POST.get('firstname'))
            return HttpResponseRedirect('/hw_app/login/')
        else:
            form = RegistrationForm()
    return render(request, 'registration2.html', {'form': form})


def login(request):
    error = ""
    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('/hw_app/main/')
        else:
            error = "Неверный логин или пароль."
    return render(request, 'login.html', locals())


def account(request):
    return render(request, 'account.html')


def logout(request):
    auth.logout(request)
    return render_to_response('logout.html')
```

Содержание файла «models.py»:

```
from django.db import models
from django.contrib.auth.models import User
from django.urls import reverse

# Create your models here.


class Actors(models.Model):
    actor_id = models.AutoField(primary_key=True)
    actor_name = models.CharField(max_length=100)
    def __str__(self):
        return self.actor_name

    class Meta:
        verbose_name_plural = "Actors"
        verbose_name = "Actor"


class Filmmakers(models.Model):
    filmmaker_id = models.AutoField(primary_key=True)
    filmmaker_name = models.CharField(max_length=100)

    def __str__(self):
        return self.filmmaker_name

    class Meta:
        verbose_name_plural = "Filmmakers"
        verbose_name = "Filmmaker"


class Film_writers(models.Model):
    film_writer_id = models.AutoField(primary_key=True)
    film_writer_name = models.CharField(max_length=100)

    def __str__(self):
        return self.film_writer_name

    class Meta:
        verbose_name_plural = "Film_writers"
        verbose_name = "Film_writer"


class Producers(models.Model):
    producer_id = models.AutoField(primary_key=True)
    producer_name = models.CharField(max_length=100)

    def __str__(self):
        return self.producer_name

    class Meta:
        verbose_name_plural = "Producers"
        verbose_name = "Producer"


class Cameramen(models.Model):
    cameraman_id = models.AutoField(primary_key=True)
    cameraman_name = models.CharField(max_length=100)

    def __str__(self):
        return self.cameraman_name

    class Meta:
        verbose_name_plural = "Cameramen"
        verbose_name = "Cameraman"


class Countries(models.Model):
    country_id = models.AutoField(primary_key=True)
    country_name = models.CharField(max_length=100)

    def __str__(self):
        return self.country_name
    class Meta:
        verbose_name_plural = "Countries"
        verbose_name = "Country"


class Films(models.Model):
    film_id = models.AutoField(primary_key=True)
    film_name = models.CharField(max_length=100)
    release_date = models.DateField()
    in_the_lead_role = models.ManyToManyField(Actors)
    filmmaker = models.ForeignKey(Filmmakers, on_delete=models.CASCADE)
    film_writer = models.ForeignKey(Film_writers, on_delete=models.CASCADE)
    producer = models.ForeignKey(Producers, on_delete=models.CASCADE)
    cameraman = models.ForeignKey(Cameramen, on_delete=models.CASCADE)
    country = models.ForeignKey(Countries, on_delete=models.CASCADE)
    box_office_results = models.IntegerField()

    def __str__(self):
        return self.film_name

    def get_absolute_url(self):
        return reverse('film_url', args=[str(self.id)])

    def get_actors(self):
        return "\n".join([i.actor_name for i in self.in_the_lead_role.all()])
    get_actors.short_description = 'In_the_lead_roles'

    class Meta:
        verbose_name_plural = "Films"
        verbose_name = "Film"


class Comments(models.Model):
    comment_id = models.AutoField(primary_key=True)
    comment_film = models.ForeignKey(Films, on_delete=models.CASCADE)
    comment_user = models.ForeignKey(User, on_delete=models.CASCADE)
    comment_text = models.TextField()
    comment_photo = models.ImageField(upload_to='downloaded',
default='def.jpg', null=True, blank=True)

    def __str__(self):
        return self.comment_text

    class Meta:
        verbose_name_plural = "Comments"
        verbose_name = "Comment"
```

Содержание файла «connection.py»:

```
import MySQLdb


class Connection:
    def __init__(self, user, password, db, host='localhost', charset='utf8'):
        #Параметры соединения
        self.user = user
        self.host = host
        self.password = password
        self.db = db
        self._connection = None
        self.charset = charset
    @property
    def connection(self):
        return self._connection

    def __enter__(self):
        self.connect()

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.disconnect()

    def connect(self):
        #Открытие соединения
        if not self._connection:
            self._connection = MySQLdb.connect(
                host = self.host,
                user = self.user,
                passwd = self.password,
                db = self.db
            )

    def disconnect(self):
        #Закрытие соединения
        if self._connection:
            self._connection.close()


class Countries:

    def __init__(self, db_connection, country_name):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.country_name = country_name

    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_countries (country_name) VALUES (%s);",
(self.country_name))
        self.db_connection.commit()
        c.close()


class Actors:

    def __init__(self, db_connection, actor_name):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.actor_name = actor_name

    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_actors (actor_name) VALUES (%s);",
(self.actor_name))
        self.db_connection.commit()
        c.close()


class Filmmakers:

    def __init__(self, db_connection, filmmaker_name):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.filmmaker_name = filmmaker_name
    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_filmmakers (filmmaker_name) VALUES
(%s);", (self.filmmaker_name))
        self.db_connection.commit()
        c.close()


class Film_writers:

    def __init__(self, db_connection, film_writer_name):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.film_writer_name = film_writer_name

    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_film_writers (film_writer_name) VALUES
(%s);", (self.film_writer_name))
        self.db_connection.commit()
        c.close()


class Producers:

    def __init__(self, db_connection, producer_name):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.producer_name = producer_name

    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_producers (producer_name) VALUES
(%s);", (self.producer_name))
        self.db_connection.commit()
        c.close()


class Cameramen:

    def __init__(self, db_connection, cameraman_name):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.cameraman_name = cameraman_name

    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_cameramen (cameraman_name) VALUES
(%s);", (self.cameraman_name))
        self.db_connection.commit()
        c.close()


class Films:

    def __init__(self, db_connection, film_name, release_date, filmmaker_id,
film_writer_id, producer_id, cameraman_id, country_id, box_office_results):
        #Сохранение соединения и данных
        self.db_connection = db_connection.connection
        self.film_name = film_name
        self.release_date = release_date
        self.filmmaker_id = filmmaker_id
        self.film_writer_id = film_writer_id
        self.producer_id = producer_id
        self.cameraman_id = cameraman_id
        self.country_id = country_id
        self.box_office_results = box_office_results

    def save(self):
        #Запись данных из объекта в запись БД
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_films (film_name, release_date,
filmmaker_id, film_writer_id, producer_id, cameraman_id, country_id,
box_office_results) VALUES (%s, %s, %s, %s, %s, %s, %s, %s);",
(self.film_name, self.release_date, self.filmmaker_id, self.film_writer_id,
self.producer_id, self.cameraman_id, self.country_id,
self.box_office_results))
        self.db_connection.commit()
        c.close()


class FilmsActors:
    def __init__(self, db_connection, film_id, actor_id):
        self.db_connection = db_connection.connection
        self.film_id = film_id
        self.actor_id = actor_id

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO db_app_films_in_the_lead_role (films_id,
actors_id) VALUES (%s, %s);", (self.film_id, self.actor_id))
        self.db_connection.commit()
        c.close()


con = Connection(user='dbuser', password='123', db='films')

with con:
    country = Countries(con, 'Россия')
    country.save()
    actor = Actors(con, 'Данила Козловский')
    actor.save()
    filmmaker = Filmmakers(con, 'Андрей Кравчук')
    filmmaker.save()
    film_writer = Film_writers(con, 'Андрей Рубанов')
    film_writer.save()
    producer = Producers(con, 'Константин Эрнст')
    producer.save()
    cameraman = Cameramen(con, 'Игорь Гринякин')
    cameraman.save()
    film = Films(con, 'Викинг', '2016.12.29', '1', '1', '1', '1', '1',
'27018393')
    film.save()
    film_actor = FilmsActors(con, '1', '1')
    film_actor.save()
```

Содержание файла «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:')
    surname = forms.CharField(label='Фамилия:')
    firstname = forms.CharField(label='Имя:')
```

Содержание файла «admin.py»:

```
from django.contrib import admin
from lab6.apps.db_app.models import *

# Register your models here.


class FilmsInline(admin.StackedInline):
    fields = ('comment_film', 'comment_text', 'comment_user',
'comment_photo')
    model = Comments
    extra = 2


class FilmsAdmin(admin.ModelAdmin):
    fields = ('film_name', 'release_date', 'in_the_lead_role', 'filmmaker',
'film_writer', 'producer', 'cameraman',
              'country', 'box_office_results')
    list_filter = ('release_date', 'country', ('in_the_lead_role',
admin.RelatedOnlyFieldListFilter),)
    list_display = ('film_name', 'release_date', 'get_actors', 'filmmaker',
'film_writer', 'producer',
                    'cameraman', 'country', 'box_office_results')
    search_fields = ('film_name',)
    list_per_page = 10
    inlines = [FilmsInline]


class ActorsAdmin(admin.ModelAdmin):
    list_per_page = 10


class CommentsAdmin(admin.ModelAdmin):
    fields = ('comment_film', 'comment_text', 'comment_user',
'comment_photo')
    list_per_page = 10


admin.site.register(Actors, ActorsAdmin)
admin.site.register(Films, FilmsAdmin)
admin.site.register(Comments, CommentsAdmin)

admin.site.site_url = '/main/'
admin.site.site_header = 'Django-администрирование'
admin.site.index_title = 'Администрирование'
```

Содержание файла «main_page.html»:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    {% block title1 %}<title>Main page</title>{% endblock %}
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.2/css/bootstrap.min.css"
    integrity="sha384-
PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb"
    crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
    integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
    crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.j
s"
    integrity="sha384-
vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh"
    crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.2/js/bootstrap.min.js"
    integrity="sha384-
alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ"
    crossorigin="anonymous"></script>
    {% load static %}
    <link rel="stylesheet" href="{% static 'css/style.css' %}"
type="text/css">
</head>
<body>
<div id="container">
    {% block title2 %}
        <div id="header"><h1>___FILMS___</h1></div>
    {% endblock %}
    {% block body1 %}
        <nav class="navbar navbar-expand-lg navbar-light">
            <a class="navbar-brand" href="{% url 'main_url' %}">Main page</a>
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse"
id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="{% url 'reg2_url'
%}">Registration <span class="sr-only">(current)</span></a>
                    </li>
                    <li class="nav-item active">
                        <a class="nav-link" href="{% url 'login_url'
%}">Login <span class="sr-only">(current)</span></a>
                    </li>
                    <li class="nav-item active">
                        <a class="nav-link" href="{% url 'account_url'
%}">Account <span class="sr-only">(current)</span></a>
                    </li>
                </ul>
            </div>
        </nav>
    <br>
    {% endblock %}
    {% block body %}
        <div class="container" id="cont1">
            {% for f in films %}
```

```
                <div class="container" id="cont2">
                    <h2><a href="/hw_app/film/get/{{ f.film_id }}"
style="font-size: 35px; font-family: Palatino Linotype;">{{ f.film_name
}}</a></h2>
                    <h4>Дата выхода: {{ f.release_date }}</h4>
                    <h4>Режиссёр: {{ f.filmmaker }}</h4>
                    <h5>Страна: {{ f.country }}</h5>
                </div>
                {% if  f.film_id == 3 %}
                    <div class="img_center" style="text-align: center;">
                        <img src="{% static 'imgs/StarWars.jpg' %}"
width="260" height="240">
                    </div>
                {% elif f.film_id == 2 %}
                    <div class="img_center" style="text-align: center;">
                        <img src="{% static 'imgs/gold.jpg' %}" width="220"
height="300">
                    </div>
                {% elif f.film_id == 1 %}
                    <div class="img_center" style="text-align: center;">
                        <img src="{% static 'imgs/viking.jpg' %}" width="320"
height="200">
                    </div>
                {% endif %}
                <hr>
            {% endfor %}
        </div>
        <br>
        <div>
            <div class="container" id="cont11">
                <ul class="pagination">
                    {% if films.has_previous %}
                        <li class="arrow"><a href="/hw_app/page/{{
films.previous_page_number }}/">&laquo;</a> </li>
                    {% else %}
                        <li class="arrow unavailable"><a href="">&laquo;</a>
</li>
                    {% endif %}
                    {% for page in films.paginator.page_range %}
                        {% if page == films.number %}
                            <li class="current"><a href="/hw_app/page/{{ page
}}">{{ page }}</a> </li>
                        {% else %}
                            <li><a href="/hw_app/page/{{ page }}/">{{ page
}}</a> </li>
                        {% endif %}
                    {% endfor %}
                    {% if films.has_next %}
                        <li class="arrow"><a href="/hw_app/page/{{
films.next_page_number }}/">&raquo;</a> </li>
                    {% else %}
                        <li class="arrow unavailable"><a
href="">&raquo;</a></li>
                    {% endif %}
                </ul>
            </div>
        </div>
        <br><br>
    {% endblock %}
</div>
</body>
</html>
```

Содержание файла «account.html»:

```
{% extends 'main_page.html' %}
{% block title1 %}<title>Account</title>{% endblock %}
{% block title2 %}
    <div id="header"><h1>Account</h1></div>
{% endblock %}
{% block body %}
    <div class="container">
        {% if request.user.is_authenticated %}
            <h3>You are now logged in as {{ request.user.username }}. </h3>
            <p style="background-color: transparent">Click <a href="{% url
'logout_url' %}">here</a> to logout.</p>
        {% else %}
            <h3>You are not logged in. </h3>
            <p style="background-color: transparent">Click <a href="{% url
'login_url' %}">here</a> to login or <a href="{% url 'reg2_url' %}">here</a>
to register. </p>
        {% endif %}
    </div>
    <br>
{% endblock %}
```

Содержание файла «login.html»:

```
{% extends 'main_page.html' %}
{% block title1 %}<title>Login</title>{% endblock %}
{% block title2 %}
    <div id="header"><h1>Login</h1></div>
{% endblock %}
{% block body %}
    <form method= "POST">
        <div class="auth_block">
            {% csrf_token %}
            <div class="auth_blank">
                <div class="form-group">
                    <label>
                        Логин:
                        <input type= "text" name= "username">
                    </label>
                </div>
            </div>
            <div class="auth_blank">
                <div class="form-group">
                    <label>
                        Пароль:
                        <input type= "password" name= "password">
                    </label>
                </div>
            </div>
            {% if error != "" %}
                <p>{{ error }}</p>
            {% endif %}
            <button type="submit">Войти</button>
            <br><br><br>
        </div>
    </form>
{% endblock %}
```

Содержание файла «logout.html»:

```
{% extends 'main_page.html' %}
{% block title1 %}<title>Logout</title>{% endblock %}
{% block title2 %}
    <div id="header"><h1>Logout</h1></div>
{% endblock %}
{% block body %}
    <div class="container">
        <h3>You are now logged out.</h3>
    </div>
    <br>
    <meta http-equiv="refresh" content="3; url=/hw_app/main/">
{% endblock %}
```

Содержание файла «registration2.html»:

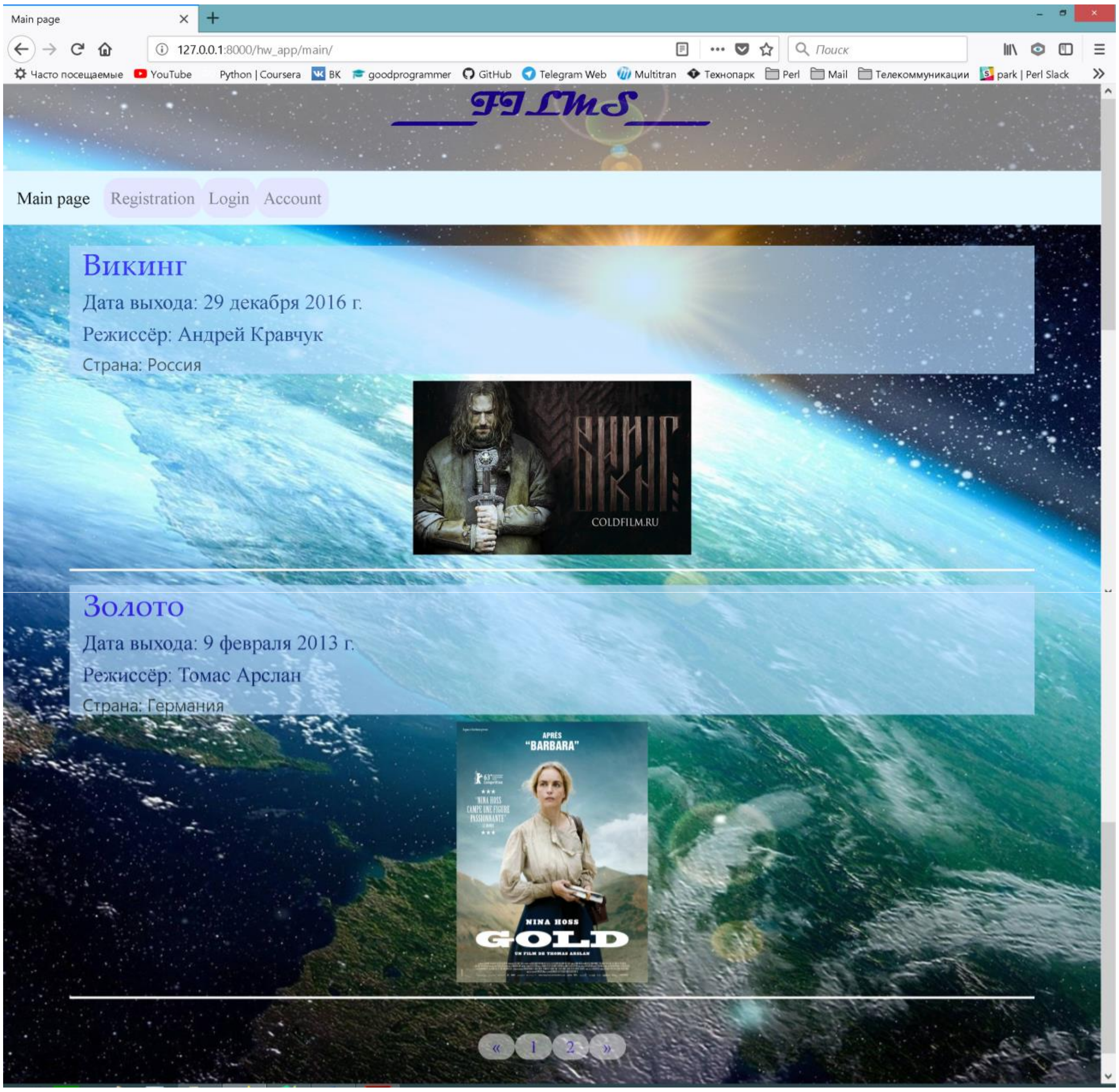
```
{% extends 'main_page.html' %}
{% block title1 %}<title>Registration</title>{% endblock %}
{% block title2 %}
    <div id="header"><h1>Registration</h1></div>
{% endblock %}
{% block body %}
    <form method= "POST">
        <div class="auth_block">
            {% csrf_token %}
            {{ form.as_p }}
            <button type= "submit"> Зарегистрироваться </button>
        </div>
    <br><br>
    </form>
{% endblock %}
```

Содержание файла «style.css»:

```
body {
    background-image: url("../imgs/bckg.jpg");
    background-size: cover;
}

#header {
    background: white;
    width: 100%;
    height: 100px;
    line-height:100px;
    opacity: 0.5;
    color: #3100ff;
    font-family: Magneto;
    font-size: 38px;
    text-align: center;
}

li {
    background: white;
    opacity: 0.5;
    color: white;
    list-style-type: none;
    link: black;
    border-radius: 18px;
    width: 100%;
    margin: 0 auto;
    text-align:  center;
}

a {
    color: #3100ff;
    text-decoration:none;
    font-family: "Times New Roman";
    font-size: 20px;
}

a:hover {
    color: #ad00ff;
    text-decoration:none
}

table {
    background-color: white;
    opacity: 0.9;
    border: 2px solid #4500ff;
    width: 100%;
   height:100px;
    font-family: "Times New Roman";
    font-size: 18px;
    text-align: center;
    border-collapse:separate;
}

th, td {
    border: 1px solid #4500ff;
}

.navbar {
    background-color: #e2f7ff;
}

.nav-item {
    background-color: #e6d5ff;
}

form {
    position: relative;
    top: 10%;
    left: 50%;
    width: 200px;
    height:100px;
    margin: -10px 0 0 -100px;
}

.form-group {
    background-color: #caf7ff;
}

input {
    background-color: #d1fdff;
}

button {
    background-color: #0f1a6f;
    color: white;
}

label {
```

```css
    background-color: #c8e6ff;
    font-size: 16px;
    color: #3100ff;
}

h4 {
    font-family: "Times New Roman";
    color: #0f1a6f;
}

p {
    background-color: #c8e6ff;
}


hr {
    color: white;
    background-color: white;
    height: 2px;
}

#cont2 {
    background-color: rgb(183, 221, 255);
    opacity: 0.7;
}

h2 {
    font-family: "Palatino Linotype";
    font-size: 35px;
    color: black;
}

#cont11 {
    width: 200px;
}
```

Результат работы программы:



Main page

127.0.0.1:8000/hw_app/main/

Часто посещаемые YouTube Python | Coursera ВК goodprogrammer GitHub Telegram Web Multitran Технопарк Perl Mail Телекоммуникации park | Perl Slack

__FILMS__

Main page Registration Login Account

Викинг

Дата выхода: 29 декабря 2016 г.

Режиссёр: Андрей Кравчук

Страна: Россия

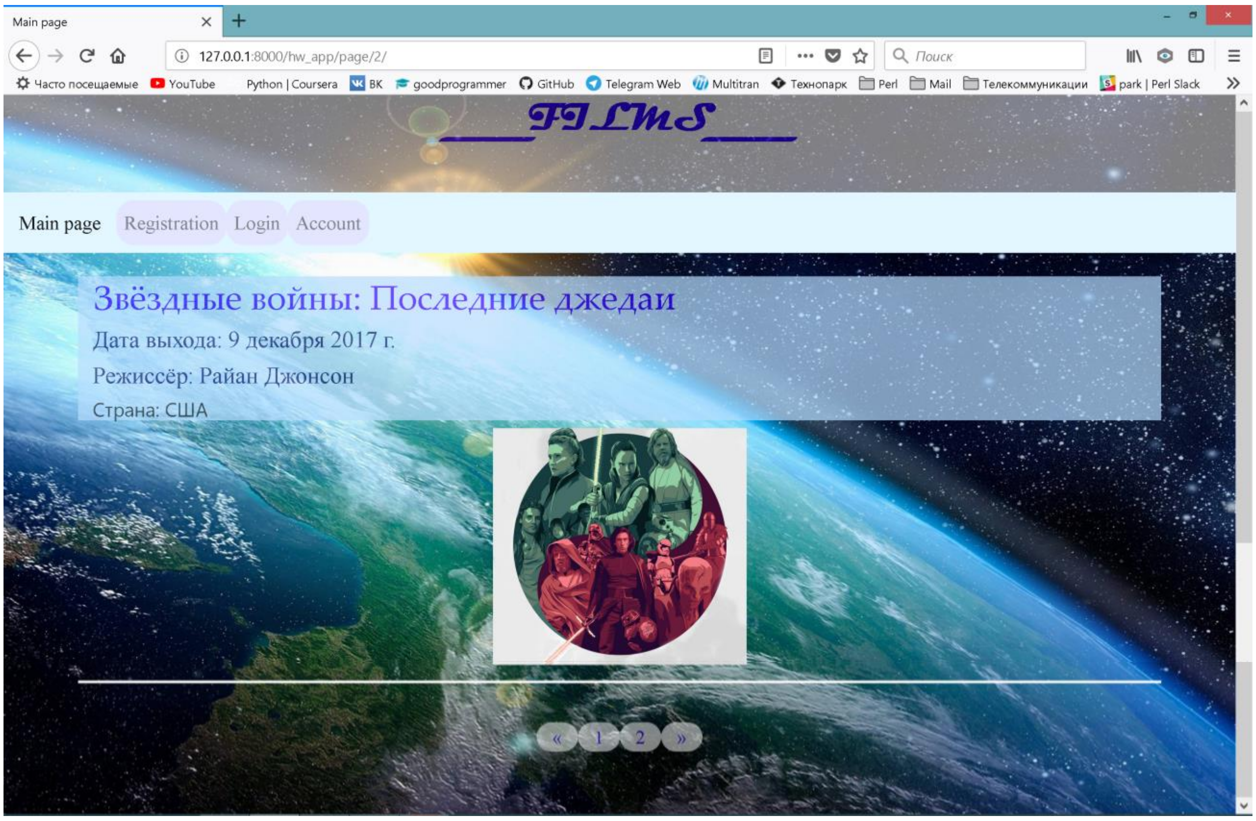
Золото

Дата выхода: 9 февраля 2013 г.

Режиссёр: Томас Арслан

Страна: Германия

« 1 2 »



___FILMS___

Main page Registration Login Account

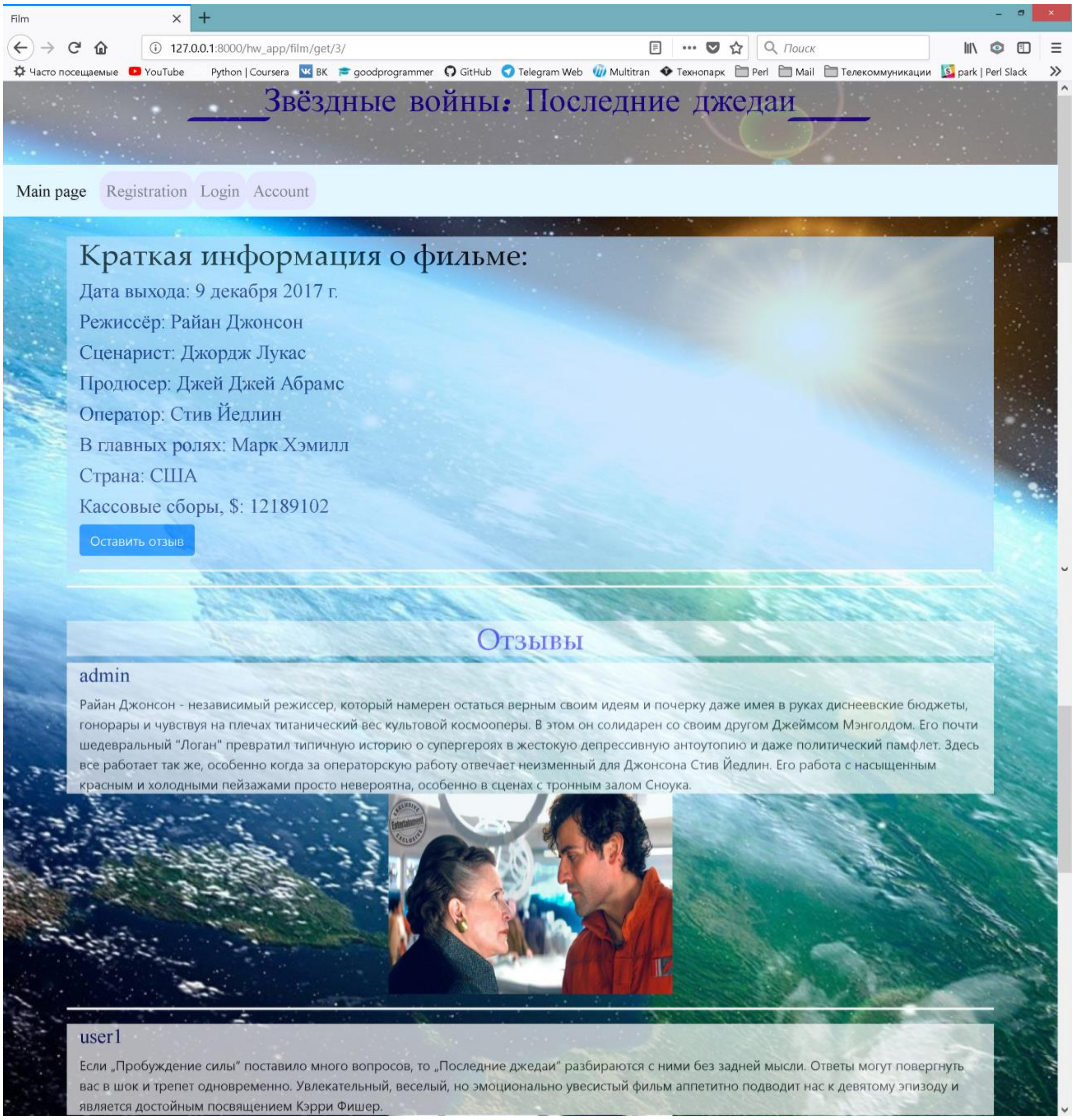
Звёздные войны: Последние джедаи

Дата выхода: 9 декабря 2017 г.

Режиссёр: Райан Джонсон

Страна: США

« 1 2 »



Звёздные войны: Последние джедаи

Main page Registration Login Account

Краткая информация о фильме:

Дата выхода: 9 декабря 2017 г.

Режиссёр: Райан Джонсон

Сценарист: Джордж Лукас

Продюсер: Джей Джей Абрамс

Оператор: Стив Йедлин

В главных ролях: Марк Хэмилл

Страна: США

Кассовые сборы, $: 12189102

Оставить отзыв

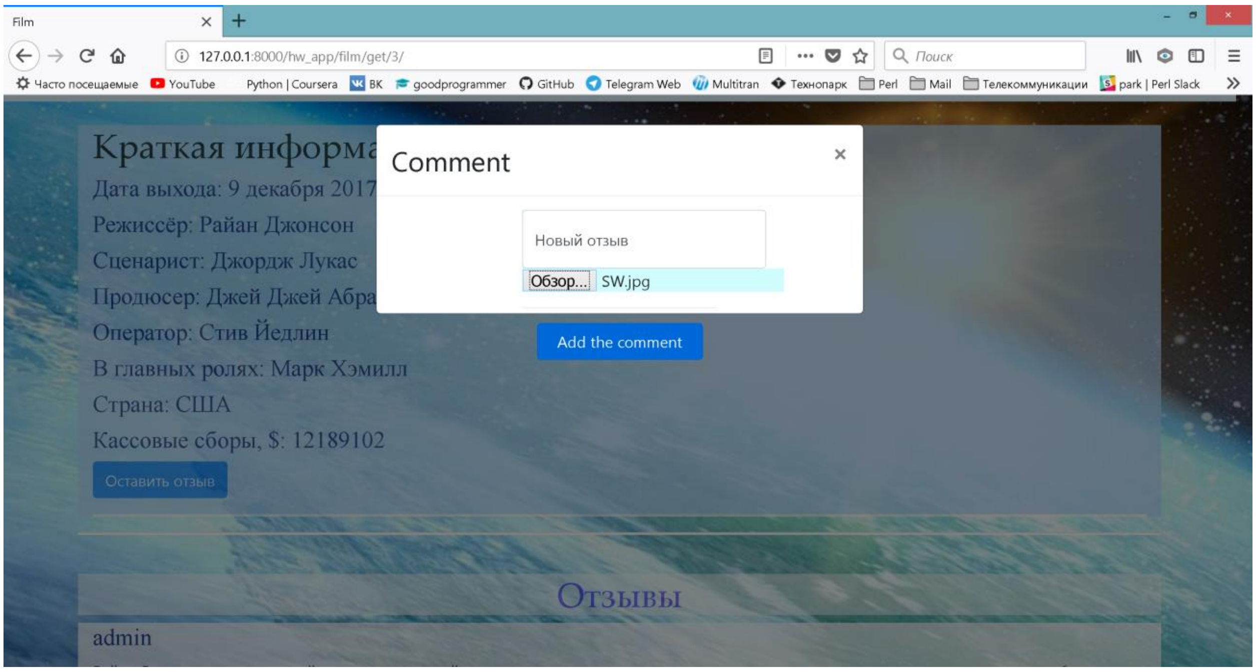
Отзывы

admin

Райан Джонсон - независимый режиссер, который намерен остаться верным своим идеям и почерку даже имея в руках диснеевские бюджеты, гонорары и чувствуя на плечах титанический вес культовой космооперы. В этом он солидарен со своим другом Джеймсом Мэнголдом. Его почти шедевральный "Логан" превратил типичную историю о супергероях в жестокую депрессивную антоутопию и даже политический памфлет. Здесь все работает так же, особенно когда за операторскую работу отвечает неизменный для Джонсона Стив Йедлин. Его работа с насыщенным красным и холодными пейзажами просто невероятна, особенно в сценах с тронным залом Сноука.

user1

Если „Пробуждение силы" поставило много вопросов, то „Последние джедаи" разбираются с ними без задней мысли. Ответы могут повергнуть вас в шок и трепет одновременно. Увлекательный, веселый, но эмоционально увесистый фильм аппетитно подводит нас к девятому эпизоду и является достойным посвящением Кэрри Фишер.



Краткая информа

Дата выхода: 9 декабря 2017

Режиссёр: Райан Джонсон

Сценарист: Джордж Лукас

Продюсер: Джей Джей Абра

Оператор: Стив Йедлин

В главных ролях: Марк Хэмилл

Страна: США

Кассовые сборы, $: 12189102

Оставить отзыв

Comment

×

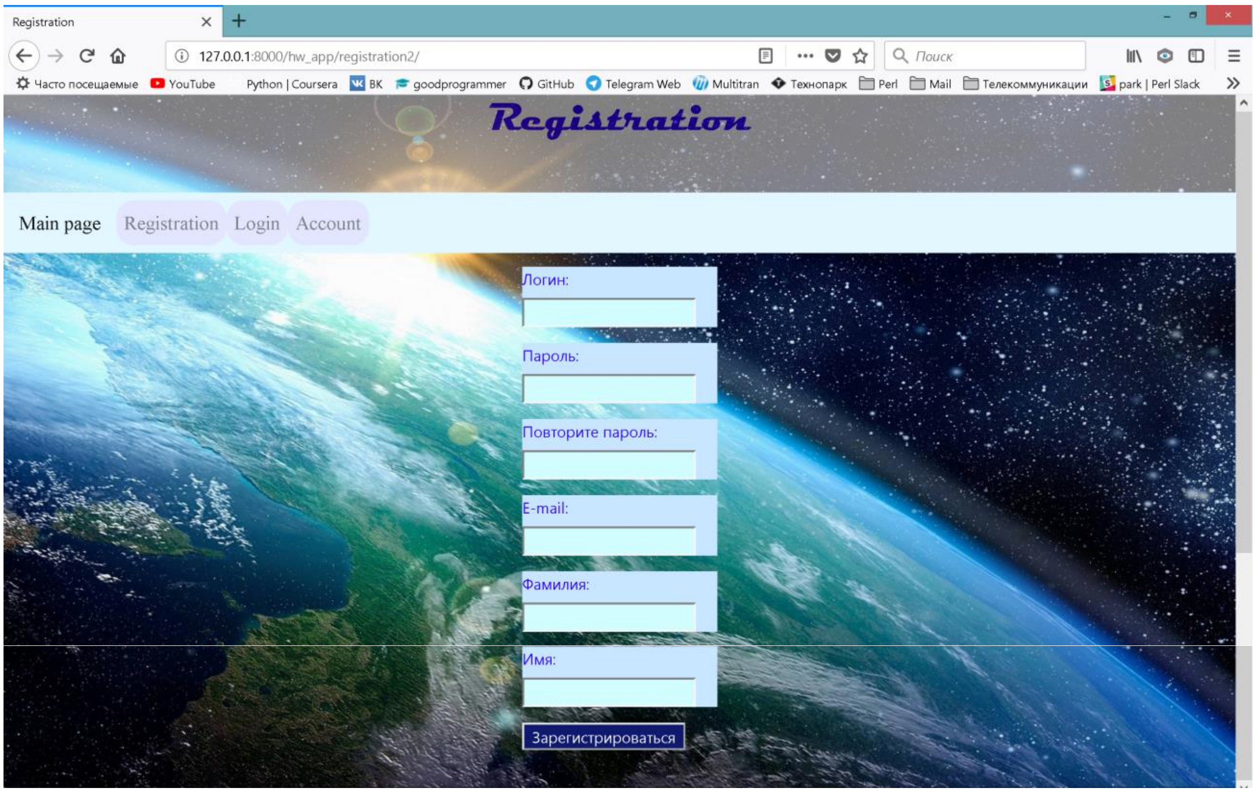
Новый отзыв

Обзор... SW.jpg

Add the comment

Отзывы

admin



Registration

Main page Registration Login Account

Логин:

Пароль:

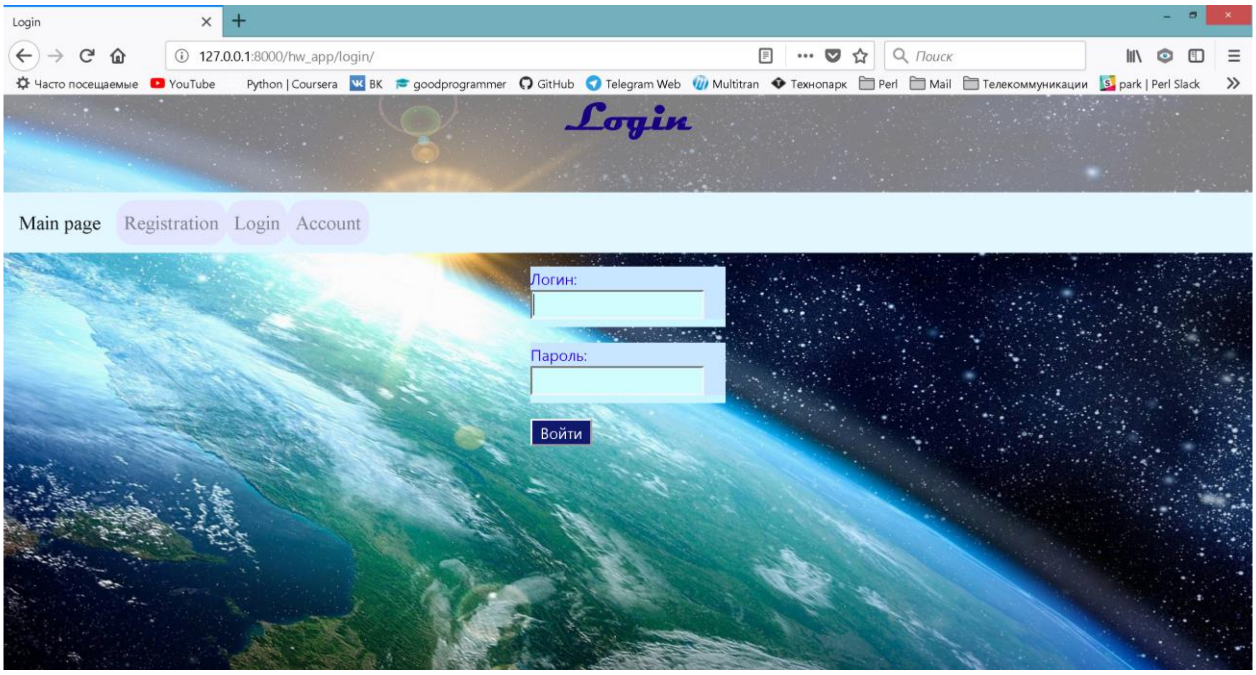
Повторите пароль:

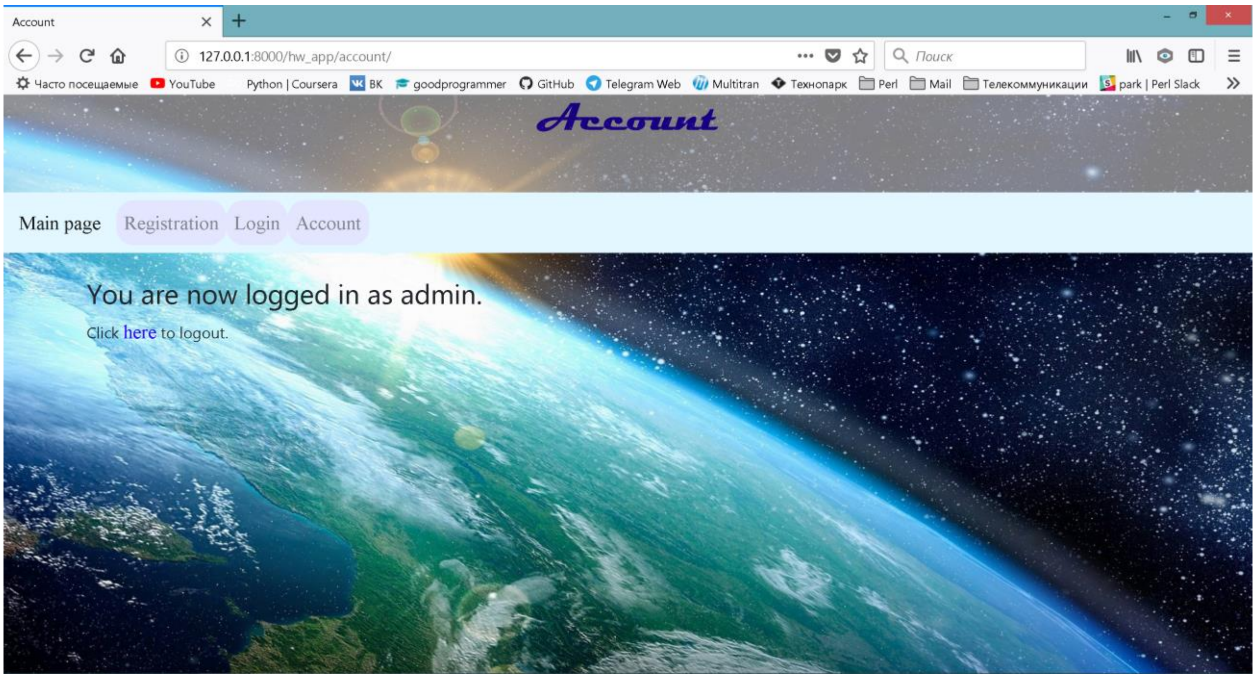
E-mail:

Фамилия:

Имя:

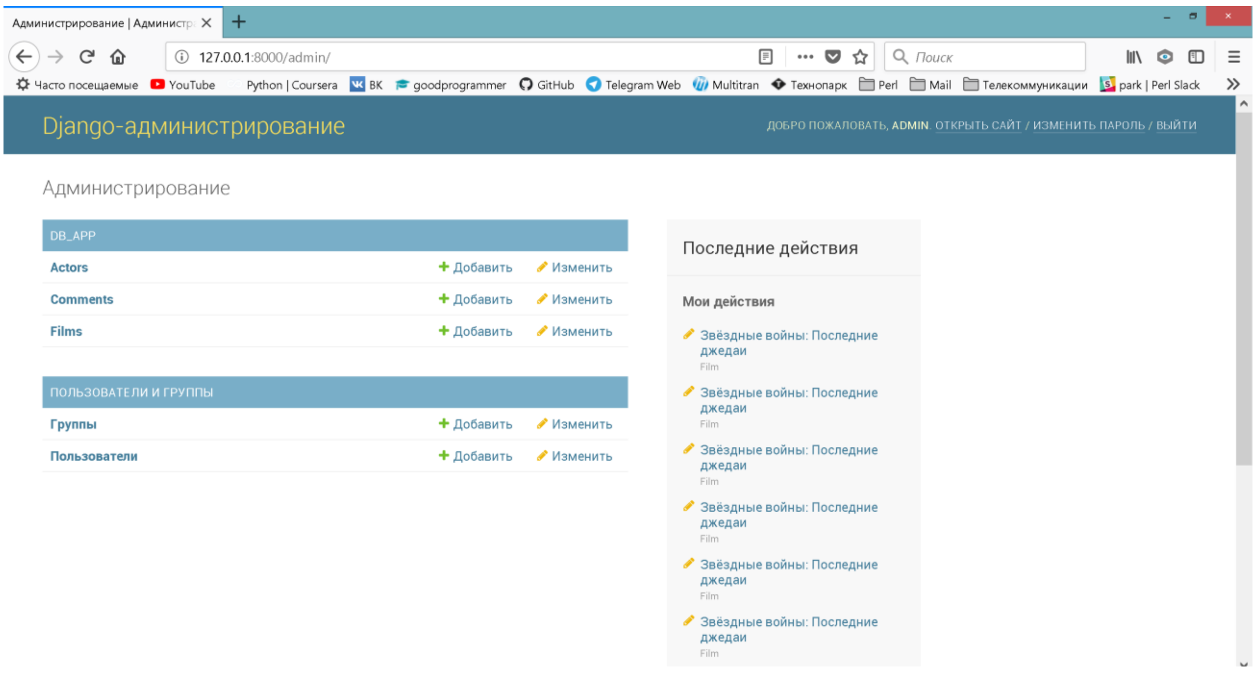
Зарегистрироваться





You are now logged in as admin.

Click here to logout.


Администрирование | Администр

127.0.0.1:8000/admin/

Часто посещаемые · YouTube · Python | Coursera · ВК · goodprogrammer · GitHub · Telegram Web · Multitran · Технопарк · Perl · Mail · Телекоммуникации · park | Perl Slack

Django-администрирование

ДОБРО ПОЖАЛОВАТЬ, ADMIN. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Администрирование

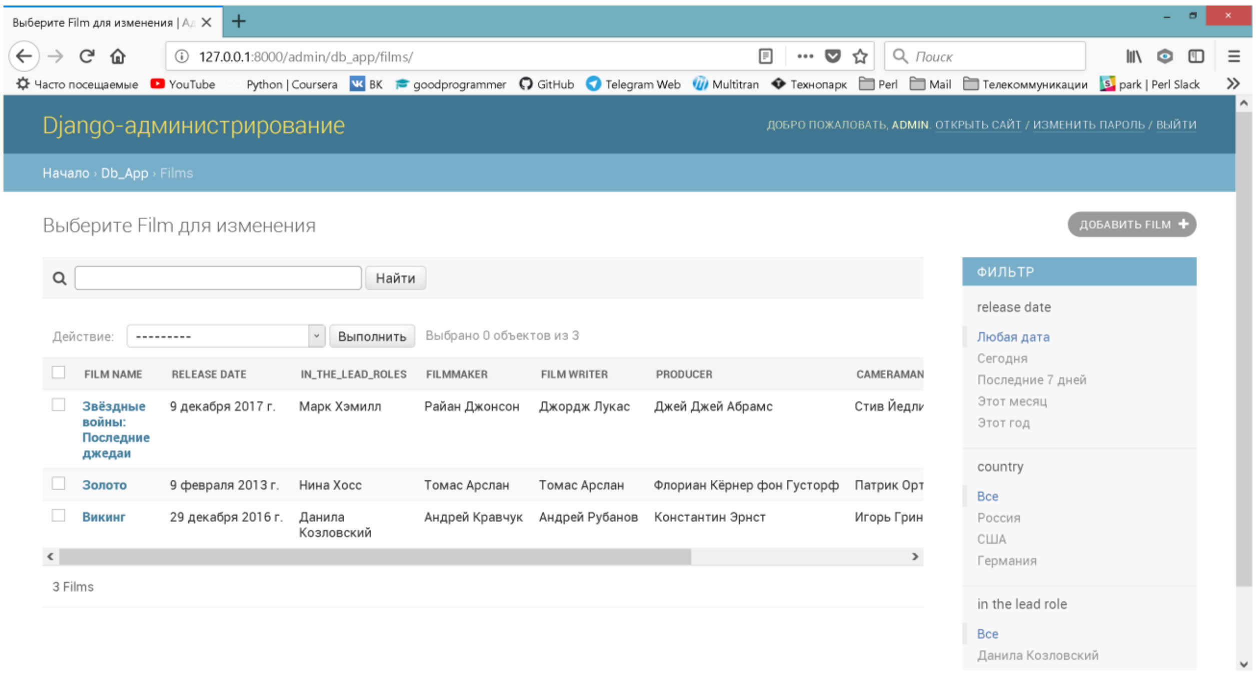
DB_APP		
Actors	+ Добавить	Изменить
Comments	+ Добавить	Изменить
Films	+ Добавить	Изменить

ПОЛЬЗОВАТЕЛИ И ГРУППЫ		
Группы	+ Добавить	Изменить
Пользователи	+ Добавить	Изменить

Последние действия

Мои действия

- Звёздные войны: Последние джедаи
  Film
- Звёздные войны: Последние джедаи
  Film
- Звёздные войны: Последние джедаи
  Film
- Звёздные войны: Последние джедаи
  Film
- Звёздные войны: Последние джедаи
  Film
- Звёздные войны: Последние джедаи
  Film


Выберите Film для изменения | A

127.0.0.1:8000/admin/db_app/films/

Часто посещаемые · YouTube · Python | Coursera · ВК · goodprogrammer · GitHub · Telegram Web · Multitran · Технопарк · Perl · Mail · Телекоммуникации · park | Perl Slack

Django-администрирование

ДОБРО ПОЖАЛОВАТЬ, ADMIN. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Начало › Db_App › Films

Выберите Film для изменения

ДОБАВИТЬ FILM +

Найти

Действие: --------- Выполнить Выбрано 0 объектов из 3

FILM NAME	RELEASE DATE	IN_THE_LEAD_ROLES	FILMMAKER	FILM WRITER	PRODUCER	CAMERAMAN
Звёздные войны: Последние джедаи	9 декабря 2017 г.	Марк Хэмилл	Райан Джонсон	Джордж Лукас	Джей Джей Абрамс	Стив Йедли
Золото	9 февраля 2013 г.	Нина Хосс	Томас Арслан	Томас Арслан	Флориан Кёрнер фон Густорф	Патрик Орт
Викинг	29 декабря 2016 г.	Данила Козловский	Андрей Кравчук	Андрей Рубанов	Константин Эрнст	Игорь Грин

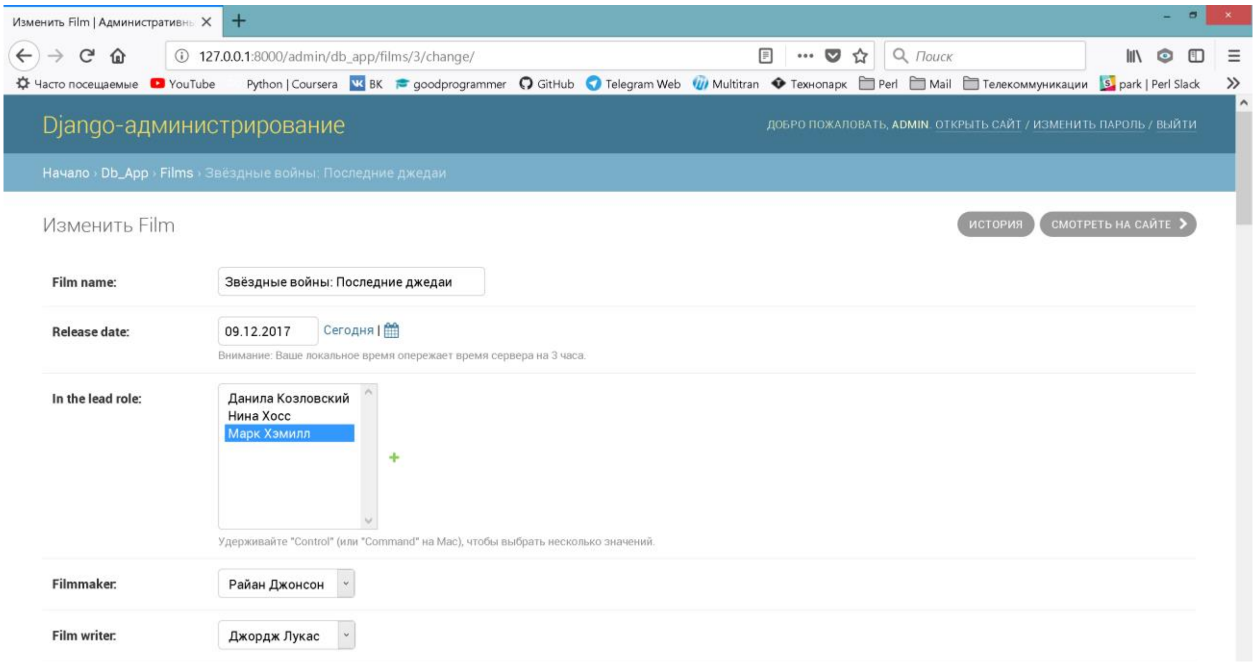
3 Films

ФИЛЬТР

release date
- Любая дата
- Сегодня
- Последние 7 дней
- Этот месяц
- Этот год

country
- Все
- Россия
- США
- Германия

in the lead role
- Все
- Данила Козловский


Изменить Film | Административн
127.0.0.1:8000/admin/db_app/films/3/change/
Поиск
Часто посещаемые
YouTube
Python | Coursera
ВК
goodprogrammer
GitHub
Telegram Web
Multitran
Технопарк
Perl
Mail
Телекоммуникации
park | Perl Slack
Django-администрирование
ДОБРО ПОЖАЛОВАТЬ, ADMIN. ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ
Начало › Db_App › Films › Звёздные войны: Последние джедаи
ИСТОРИЯ
СМОТРЕТЬ НА САЙТЕ

Изменить Film

Film name: Звёздные войны: Последние джедаи

Release date: 09.12.2017 Сегодня |

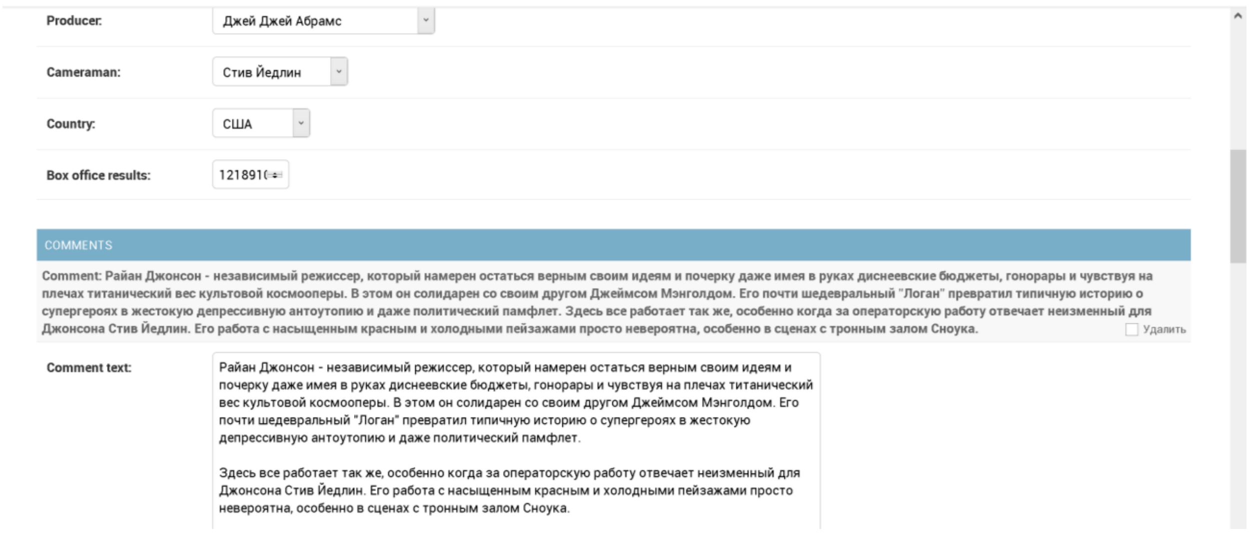
Внимание: Ваше локальное время опережает время сервера на 3 часа.

In the lead role:
Данила Козловский
Нина Хосс
Марк Хэмилл

Удерживайте "Control" (или "Command" на Mac), чтобы выбрать несколько значений.

Filmmaker: Райан Джонсон

Film writer: Джордж Лукас



Producer: Джей Джей Абрамс

Cameraman: Стив Йедлин

Country: США

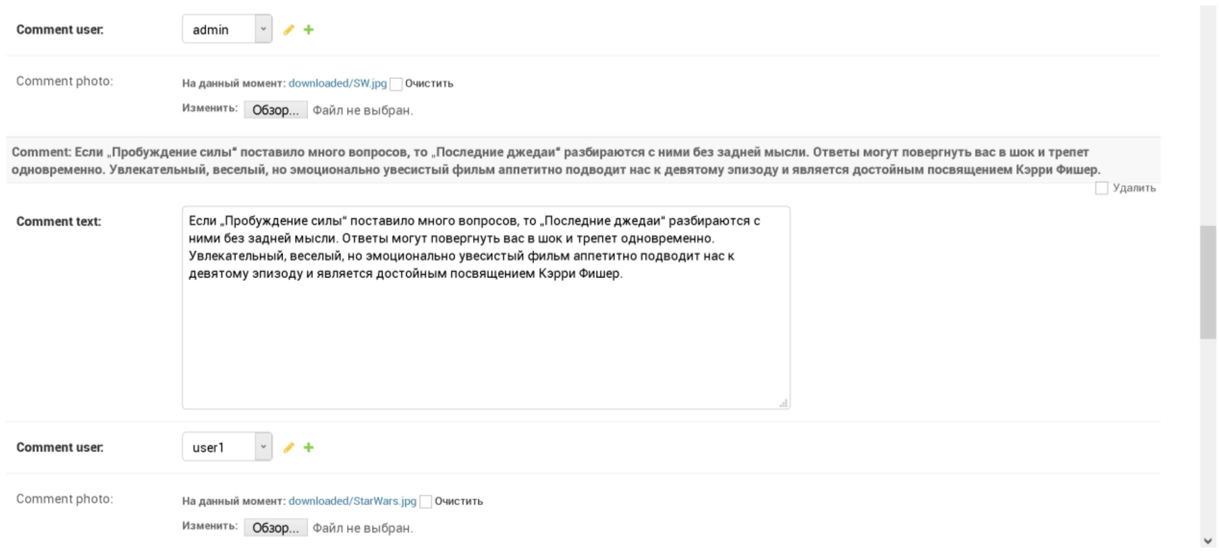
Box office results: 121891(

COMMENTS

Comment: Райан Джонсон - независимый режиссер, который намерен остаться верным своим идеям и почерку даже имея в руках диснеевские бюджеты, гонорары и чувства на плечах титанический вес культовой космооперы. В этом он солидарен со своим другом Джеймсом Мэнголдом. Его почти шедевральный "Логан" превратил типичную историю о супергероях в жестокую депрессивную антоутопию и даже политический памфлет. Здесь все работает так же, особенно когда за операторскую работу отвечает неизменный для Джонсона Стив Йедлин. Его работа с насыщенным красным и холодными пейзажами просто невероятна, особенно в сценах с тронным залом Сноука. Удалить

Comment text: Райан Джонсон - независимый режиссер, который намерен остаться верным своим идеям и почерку даже имея в руках диснеевские бюджеты, гонорары и чувства на плечах титанический вес культовой космооперы. В этом он солидарен со своим другом Джеймсом Мэнголдом. Его почти шедевральный "Логан" превратил типичную историю о супергероях в жестокую депрессивную антоутопию и даже политический памфлет.

Здесь все работает так же, особенно когда за операторскую работу отвечает неизменный для Джонсона Стив Йедлин. Его работа с насыщенным красным и холодными пейзажами просто невероятна, особенно в сценах с тронным залом Сноука.



Comment user: admin

Comment photo: На данный момент: downloaded/SW.jpg Очистить
Изменить: Обзор... Файл не выбран.

Comment: Если „Пробуждение силы" поставило много вопросов, то „Последние джедаи" разбираются с ними без задней мысли. Ответы могут повергнуть вас в шок и трепет одновременно. Увлекательный, веселый, но эмоционально увесистый фильм аппетитно подводит нас к девятому эпизоду и является достойным посвящением Кэрри Фишер. Удалить

Comment text: Если „Пробуждение силы" поставило много вопросов, то „Последние джедаи" разбираются с ними без задней мысли. Ответы могут повергнуть вас в шок и трепет одновременно. Увлекательный, веселый, но эмоционально увесистый фильм аппетитно подводит нас к девятому эпизоду и является достойным посвящением Кэрри Фишер.

Comment user: user1

Comment photo: На данный момент: downloaded/StarWars.jpg Очистить
Изменить: Обзор... Файл не выбран.