

**Московский государственный технический  
университет им. Н.Э. Баумана**

**Факультет «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»**

**Курс «Парадигмы и конструкции языков программирования»  
Отчет по домашнему заданию**

Выполнил:  
студент группы ИУ5-34Б  
Чернев Николай Андреевич

Проверил:  
преподаватель каф. ИУ5  
Нардид Анатолий  
Николаевич

Москва, 2024 г

### **Постановка задачи:**

Создать web-приложение на языке Python с использованием фреймворка Django. Приложение по заданному Steam ID (либо с помощью авторизации через Steam) получает с помощью Steam API информацию о списке игр данного пользователя, времени, проведенном в конкретной игре за все время и за последние 2 недели, количестве друзей с этой игрой, проценте выполненных достижений и выводит эту информацию на страницу в виде таблицы, которую можно сортировать по столбцам по возрастанию и убыванию

## Текст программы

games/models.py

```
from django.db import models

class Game(models.Model):
    name = models.CharField(max_length=255)
    app_id = models.IntegerField(unique=True)
    playtime_forever = models.IntegerField(default=0)

    def __str__(self):
        return self.name

class Profile(models.Model):
    steam_id = models.CharField(max_length=50, unique=True)
    avatar_url = models.URLField(blank=True, null=True)
    nickname = models.CharField(max_length=50, blank=True, null=True)

    def __str__(self):
        return f"{self.nickname} ({self.steam_id})"
```

games/views.py

```
import requests
from django.shortcuts import render, redirect, get_object_or_404
from django.http import HttpResponse
from openid.consumer.consumer import Consumer
from openid.fetchers import HTTPFetchingError
from openid.store.memstore import MemoryStore
from .models import Profile

STEAM_API_KEY = "Enter your key here"
STEAM_OPENID_URL = "https://steamcommunity.com/openid"

def index(request):
    return render(request, 'games/index.html')

def profile_page(request, steam_id):
    profile = get_object_or_404(Profile, steam_id=steam_id)
    return render(request, 'games/profile.html', {'profile': profile})

def update_profile(steam_id, profile):
    url = f"http://api.steampowered.com/ISteamUser/GetPlayerSummaries/v0002/"
    params = {
        'key': STEAM_API_KEY,
        'steamids': steam_id,
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        data = response.json().get('response', {}).get('players', [])[0]
        updated_profile = profile
        updated_profile.steam_id = steam_id
        updated_profile.avatar_url = data.get('avatarmedium')
        updated_profile.nickname = data.get('personaname')
        updated_profile.save()

openid_store = MemoryStore()

def steam_login(request):
    consumer = Consumer({}, openid_store)
    auth_request = consumer.begin(STEAM_OPENID_URL)
```

```

    if not auth_request:
        return redirect('/error/')
    callback_url = request.build_absolute_uri('/login/callback/')
    realm = request.build_absolute_uri('/')
    redirect_url = auth_request.redirectURL(realm=realm,
return_to=callback_url)
    return redirect(redirect_url)

def steam_callback(request):
    consumer = Consumer(request.session, openid_store)
    try:
        response = consumer.complete(request.GET, request.build_absolute_uri())
        if response.status == "success":
            steam_id = response.identity_url.split("/")[-1]
            if steam_id:
                profile, created =
Profile.objects.get_or_create(steam_id=steam_id)
                update_profile(steam_id, profile)
                profile.save()
                return redirect('profile', steam_id=steam_id)
            else:
                return HttpResponse("Не удалось получить данные пользователя.",
status=400)
        else:
            return HttpResponse("Ошибка в OpenID-ответе", status=400)

    except HTTPFetchingError as e:
        return HttpResponse(f"Ошибка при соединении с Steam: {str(e)}",
status=500)

def get_friend_list(steam_id):
    url = f"https://api.steampowered.com/ISteamUser/GetFriendList/v0001/"
    params = {
        'key': STEAM_API_KEY,
        'steamid': steam_id,
        'relationship': 'friend',
    }
    response = requests.get(url, params=params)
    if response.status_code != 200:
        print(f"Failed to fetch friend list for Steam ID {steam_id}. Status
code: {response.status_code}")
        return []
    data = response.json()
    friends = [friend['steamid'] for friend in data.get('friendslist',
{}).get('friends', [])]
    return friends

def get_friends_games(steam_id):
    friends = get_friend_list(steam_id)
    friends_games = {}
    for friend_id in friends:
        url =
f"https://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/"
        params = {
            'key': STEAM_API_KEY,
            'steamid': friend_id,
            'include_appinfo': True,
            'format': 'json',
        }
        response = requests.get(url, params=params)
        if response.status_code == 200:
            data = response.json()
            for game in data.get('response', {}).get('games', []):
                if game['appid'] not in friends_games:
                    friends_games[game['appid']] = 1

```

```

        else:
            friends_games[game['appid']] += 1
    return friends_games

def count_friends_with_game(game_appid, friends_games):
    if game_appid not in friends_games:
        return 0
    return friends_games[game_appid]

def get_achievements_percentage(steam_id, app_id):
    url =
    f"http://api.steampowered.com/ISteamUserStats/GetPlayerAchievements/v0001/"
    params = {
        'appid': app_id,
        'key': STEAM_API_KEY,
        'steamid': steam_id,
    }
    response = requests.get(url, params=params)
    if response.status_code != 200:
        print(f"No player stats available for Steam ID {steam_id} and App ID {app_id}.")
        return 0
    data = response.json()
    if 'playerstats' in data:
        achievements = data['playerstats'].get('achievements', [])
        total_achievements = len(achievements)
        completed_achievements = sum(1 for achievement in achievements if
            achievement['achieved'] == 1)
        if total_achievements > 0:
            percentage = int((completed_achievements / total_achievements) *
100)
        else:
            percentage = 0
        return percentage
    else:
        print(f"No player stats available for Steam ID {steam_id} and App ID {app_id}.")
        return 0

def get_games(request):
    steam_id = request.GET.get('steam_id')
    if not steam_id:
        return render(request, 'games/index.html', {'error': 'Steam ID is
required'})
    url = f"https://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/"
    params = {
        'key': STEAM_API_KEY,
        'steamid': steam_id,
        'include_appinfo': True,
        'format': 'json',
    }
    response = requests.get(url, params=params)
    if response.status_code != 200:
        return render(request, 'games/index.html', {'error': 'Failed to fetch
data from Steam API'})
    data = response.json()
    games = data.get('response', {}).get('games', [])
    friends_games = get_friends_games(steam_id)
    if not games:
        return render(request, 'games/index.html', {'error': 'No games found or
the account is private.'})
    game_list = [
        {
            'name': game.get('name', 'Unknown'),
            'playtime_hours': game.get('playtime_forever', 0) // 60,

```

```

        'playtime_2weeks': game.get('playtime_2weeks', 0) // 60,
        'app_id': game.get('appid'),
        'friends_with_game': count_friends_with_game(game.get('appid'),
friends_games),
        'achievements_percentage': get_achievements_percentage(steam_id,
game.get('appid')),
    }
    for game in games
]
return render(request, 'games/game_list.html', {'games': game_list,
'steam_id': steam_id})

```

### steam\_games/settings.py

```

---
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "games"
]
---

```

### steam\_games/urls.py

```

from django.contrib import admin
from django.urls import path
from games import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index, name='index'),
    path('get-games/', views.get_games, name='get_games'),
    path('login/', views.steam_login, name='steam_login'),
    path('login/callback/', views.steam_callback, name='steam_callback'),
    path('profile/<str:steam_id>', views.profile_page, name='profile')
]

```

### requirements.txt

```

Django==4.2.16
python3-openid==3.2.0
requests==2.32.3

```

### games/templates/base.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        .profile-info {
            margin-top: 20px;
        }

        .profile-info p {
            font-size: 18px;
            margin-bottom: 10px;
        }

        .games-button {
            margin-top: 20px;
        }

        .games-button button {
            padding: 10px 20px;
            font-size: 16px;
            cursor: pointer;
        }
    </style>

```

```

        .games-button button:hover {
            background-color: #007BFF;
            color: white;
        }
    </style>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Steam Games{% endblock %}</title>
    <style>
        table {
            width: 100%;
            border-collapse: collapse;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: left;
        }
        th {
            cursor: pointer;
        }
    </style>
</head>
<body>
    <header>
        <h1>Steam Games Filter</h1>
    </header>

    <main>
        {% block content %}
        <!-- Page-specific content will go here -->
        {% endblock %}
    </main>

    <footer>
        <p>&copy; Steam Games Filter</p>
    </footer>
</body>
</html>

```

## games/templates/games/game\_list.html

```

{% extends "base.html" %}

{% block title %}Game List{% endblock %}

{% block content %}
<h2>Games for Steam ID: {{ steam_id }}</h2>

<table id="gamesTable">
    <thead>
        <tr>
            <th onclick="sortTable('name')">Name</th>
            <th onclick="sortTable('playtime')">Playtime (total hours)</th>
            <th onclick="sortTable('playtime_2weeks')">Playtime (last 2
weeks)</th>
            <th onclick="sortTable('friends_with_game')">Friends with Game</th>
            <th onclick="sortTable('achievements_percentage')">Achievements
Percentage</th>
        </tr>
    </thead>
    <tbody>
        {% for game in games %}
        <tr>
            <td data-name="{{ game.name }}">{{ game.name }}</td>
            <td data-playtime="{{ game.playtime_hours }}">{{
game.playtime_hours }}</td>
            <td data-playtime_2weeks="{{ game.playtime_2weeks }}">{{

```

```

game.playtime_2weeks }}</td>
      <td data-friends_with_game="{{ game.friends_with_game }}">{{
game.friends_with_game }}</td>
      <td data-achievements_percentage="{{ game.achievements_percentage
}}">{{ game.achievements_percentage }}%</td>
    </tr>
    {% endfor %}
  </tbody>
</table>

<script>
  let sortOrder = {
    name: true, // true - ascending, false - descending
    playtime: true,
    playtime_2weeks: true,
    friends_with_game: true,
    achievements_percentage: true,
  };

  function sortTable(column) {
    const table =
document.getElementById('gamesTable').querySelector('tbody');
    const rows = Array.from(table.rows);
    const isAscending = sortOrder[column];

    rows.sort((rowA, rowB) => {
      const getData = (row, key) =>
row.querySelector(`[data-${key}]`).dataset[key];
      let valA = getData(rowA, column);
      let valB = getData(rowB, column);

      if (['playtime', 'playtime_2weeks', 'friends_with_game',
'achievements_percentage'].includes(column)) {
        valA = parseFloat(valA) || 0;
        valB = parseFloat(valB) || 0;
      } else {
        valA = valA.toLowerCase();
        valB = valB.toLowerCase();
      }

      return isAscending ? (valA > valB ? 1 : valA < valB ? -1 : 0) :
(valA < valB ? 1 : valA > valB ? -1 : 0);
    });

    rows.forEach(row => table.appendChild(row));
    sortOrder[column] = !isAscending;
  }

  function sortGames() {
    const filter = document.getElementById('filter').value;
    sortTable(filter); // Сортировка по выбранному фильтру
  }
</script>
{% endblock %}

```

## games/templates/games/index.html

```

{% extends "base.html" %}

{% block title %}Home{% endblock %}

{% block content %}
<h2>Welcome to the Steam Games Filter</h2>
<a href="{% url 'steam_login' %}">
  
</a>
<form method="get" action="/get-games/">

```



```

        <label for="steam_id">Enter Steam ID:</label>
        <input type="text" id="steam_id" name="steam_id" required>
        <button type="submit">Get Games</button>
    </form>
    {% if error %}
        <p style="color: red;">{{ error }}</p>
    {% endif %}
    {% endblock %}

games/templates/games/profile.html

{% extends 'base.html' %}

{% block title %}Profile{% endblock %}

{% block content %}
<h2>Profile</h2>

{% if profile %}
    <form method="get" action="/get-games/">
        <p><strong>Steam ID:</strong></p>
        <input type="text" id="steam_id" name="steam_id" required value="{{
profile.steam_id }}" readonly>
        <p><strong>Username:</strong> {{ profile.nickname }}</p>
        <p></p>
        <button type="submit">Get Games</button>
    </form>
{% else %}
    <p>No profile found.</p>
{% endif %}

{% endblock %}

```

Остальные файлы стандартны для фреймворка Django


# Демонстрация работы программы:

## Главная страница

127.0.0.1

### Steam Games Filter

Welcome to the Steam Games Filter



Enter Steam ID:

Get Games


© Steam Games Filter

## Поиск по Steam ID

127.0.0.1

### Steam Games Filter

Welcome to the Steam Games Filter

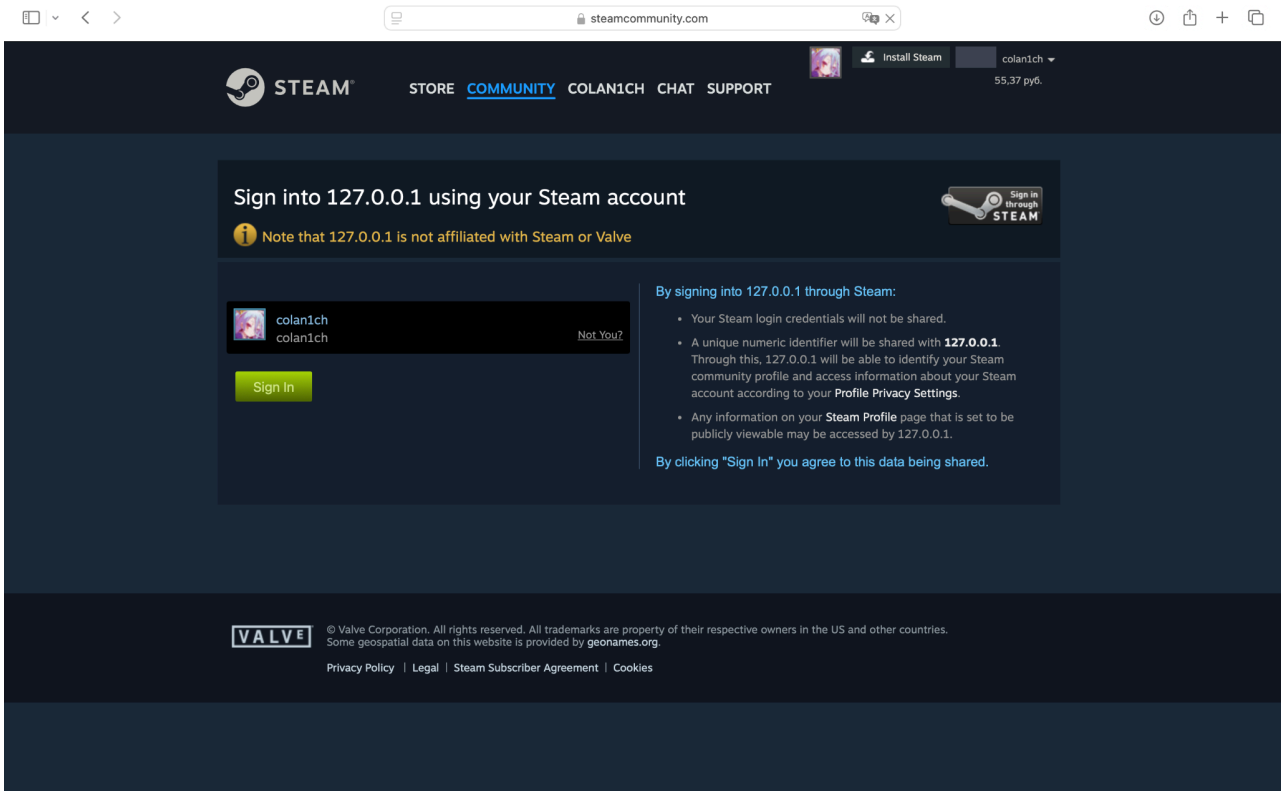


Enter Steam ID: 76561198868250701

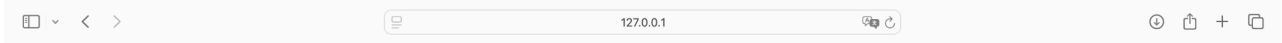
Get Games

© Steam Games Filter

# Авторизация через Steam



# Страница профиля



## Steam Games Filter

### Profile

Steam ID:

76561198868250701

Username: colan1ch



Get Games

© Steam Games Filter

Страница игр

Steam Games Filter

Games for Steam ID: 76561198868250701

Name	Playtime (total hours)	Playtime (last 2 weeks)	Friends with Game	Achievements Percentage
Left 4 Dead 2	0	0	2	1%
Portal 2	17	0	5	23%
Terraria	7	0	4	18%
War Thunder	0	0	4	0%
Geometry Dash	0	0	3	0%
Assassin's Creed Unity	0	0	1	0%
Counter-Strike 2	1585	7	13	100%
Project CARS 2	0	0	0	4%
Fishing Planet	0	0	0	0%
NARUTO SHIPPUDEN: Ultimate Ninja STORM 4	14	0	1	1%
Blackwake	0	0	0	0%
Wallpaper Engine	2	0	4	5%
Mortal Kombat X	8	0	0	10%
We Were Here	0	0	1	19%
Muse Dash	0	0	0	0%
Aimlabs	0	0	1	0%
Fall Guys	21	0	2	44%
Destiny 2	7	0	3	17%
Apex Legends	6	0	4	16%
Sea of Thieves	3	0	3	4%

Сортировка по общему времени, проведенному в игре(по убыванию)

Steam Games Filter

Games for Steam ID: 76561198868250701

Name	Playtime (total hours)	Playtime (last 2 weeks)	Friends with Game	Achievements Percentage
Counter-Strike 2	1585	7	13	100%
Fall Guys	21	0	2	44%
Portal 2	17	0	5	23%
NARUTO SHIPPUDEN: Ultimate Ninja STORM 4	14	0	1	1%
Deadlock	11	0	6	0%
Mortal Kombat X	8	0	0	10%
Terraria	7	0	4	18%
Destiny 2	7	0	3	17%
Apex Legends	6	0	4	16%
Tom Clancy's Rainbow Six Siege	6	0	3	0%
Sea of Thieves	3	0	3	4%
Hide and Run	3	0	2	100%
PUBG: BATTLEGROUNDS	3	0	11	16%
Wallpaper Engine	2	0	4	5%
Left 4 Dead 2	0	0	2	1%
War Thunder	0	0	4	0%
Geometry Dash	0	0	3	0%
Assassin's Creed Unity	0	0	1	0%
Project CARS 2	0	0	0	4%
Fishing Planet	0	0	0	0%

Сортировка по времени в игре за последние 2 недели (по убыванию)

Steam Games Filter

Games for Steam ID: 76561198868250701

Name	Playtime (total hours)	Playtime (last 2 weeks)	Friends with Game	Achievements Percentage
Counter-Strike 2	1585	7	13	100%
PUBG: BATTLEGROUNDS	3	0	11	16%
Deadlock	11	0	6	0%
Portal 2	17	0	5	23%
Terraria	7	0	4	18%
Apex Legends	6	0	4	16%
Wallpaper Engine	2	0	4	5%
War Thunder	0	0	4	0%
Destiny 2	7	0	3	17%
Tom Clancy's Rainbow Six Siege	6	0	3	0%
Sea of Thieves	3	0	3	4%
Geometry Dash	0	0	3	0%
Tom Clancy's Rainbow Six Siege - Test Server	0	0	3	0%
Fall Guys	21	0	2	44%
Hide and Run	3	0	2	100%
Left 4 Dead 2	0	0	2	1%
NARUTO SHIPPUDEN: Ultimate Ninja STORM 4	14	0	1	1%
Assassin's Creed Unity	0	0	1	0%
We Were Here	0	0	1	19%
Aimlabs	0	0	1	0%

Сортировка по количеству друзей с данной игрой (по возрастанию)

Steam Games Filter

Games for Steam ID: 76561198868250701

Name	Playtime (total hours)	Playtime (last 2 weeks)	Friends with Game	Achievements Percentage
Mortal Kombat X	8	0	0	10%
Project CARS 2	0	0	0	4%
Fishing Planet	0	0	0	0%
Blackwake	0	0	0	0%
Muse Dash	0	0	0	0%
Strinova	0	0	0	0%
Tiny Tina's Assault on Dragon Keep: A Wonderlands One-shot Adventure	0	0	0	0%
NARUTO SHIPPUDEN: Ultimate Ninja STORM 4	14	0	1	1%
Assassin's Creed Unity	0	0	1	0%
We Were Here	0	0	1	19%
Aimlabs	0	0	1	0%
Warhammer: Vermintide 2	0	0	1	0%
We Were Here Expeditions: The FriendShip	0	0	1	0%
NARAKA: BLADEPOINT	0	0	1	0%
Banana	0	0	1	0%
Half-Life 2	0	0	1	0%
Half-Life 2: Deathmatch	0	0	1	0%
Half-Life Deathmatch: Source	0	0	1	0%
Fall Guys	21	0	2	44%
Hide and Run	3	0	2	100%

# Сортировка по проценту выполненных достижений (по убыванию)

## Steam Games Filter

Games for Steam ID: 76561198868250701

Name	Playtime (total hours)	Playtime (last 2 weeks)	Friends with Game	Achievements Percentage
Hide and Run	3	0	2	100%
Counter-Strike 2	1585	7	13	100%
Fall Guys	21	0	2	44%
Portal 2	17	0	5	23%
We Were Here	0	0	1	19%
Terraria	7	0	4	18%
Destiny 2	7	0	3	17%
Apex Legends	6	0	4	16%
PUBG: BATTLEGROUNDS	3	0	11	16%
Mortal Kombat X	8	0	0	10%
Wallpaper Engine	2	0	4	5%
Project CARS 2	0	0	0	4%
Sea of Thieves	3	0	3	4%
NARUTO SHIPPUDEN: Ultimate Ninja STORM 4	14	0	1	1%
Left 4 Dead 2	0	0	2	1%
Fishing Planet	0	0	0	0%
Blackwake	0	0	0	0%
Muse Dash	0	0	0	0%
Strinova	0	0	0	0%
Tim's Time's Assault on Dragon Keep: A Wonderland's One-shot Adventure	0	0	0	0%

# Сортировка по названию (по возрастанию)

## Steam Games Filter

Games for Steam ID: 76561198868250701

Name	Playtime (total hours)	Playtime (last 2 weeks)	Friends with Game	Achievements Percentage
Aimlabs	0	0	1	0%
Apex Legends	6	0	4	16%
Assassin's Creed Unity	0	0	1	0%
Banana	0	0	1	0%
Blackwake	0	0	0	0%
Counter-Strike 2	1585	7	13	100%
Deadlock	11	0	6	0%
Destiny 2	7	0	3	17%
Fall Guys	21	0	2	44%
Fishing Planet	0	0	0	0%
Geometry Dash	0	0	3	0%
Half-Life 2	0	0	1	0%
Half-Life 2: Deathmatch	0	0	1	0%
Half-Life Deathmatch: Source	0	0	1	0%
Hide and Run	3	0	2	100%
Left 4 Dead 2	0	0	2	1%
Mortal Kombat X	8	0	0	10%
Muse Dash	0	0	0	0%
NARAKA: BLADEPOINT	0	0	1	0%
NARUTO SHIPPUDEN: Ultimate Ninja STORM 4	14	0	1	1%