**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: | Проверил: |
| --- | --- |
| студент группы ИУ5-34Б | преподаватель каф. ИУ5 |
| Евсеев Дмитрий Михайлович | 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;

}

.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' %}">

<img src="https://steamcommunity-a.akamaihd.net/public/images/signinthroughsteam/sits\_large\_border.png" alt="Sign in through Steam">

</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><img src="{{ profile.avatar\_url }}" alt="Avatar" width="64" height="64"></p>

<button type="submit">Get Games</button>

</form>

{% else %}

<p>No profile found.</p>

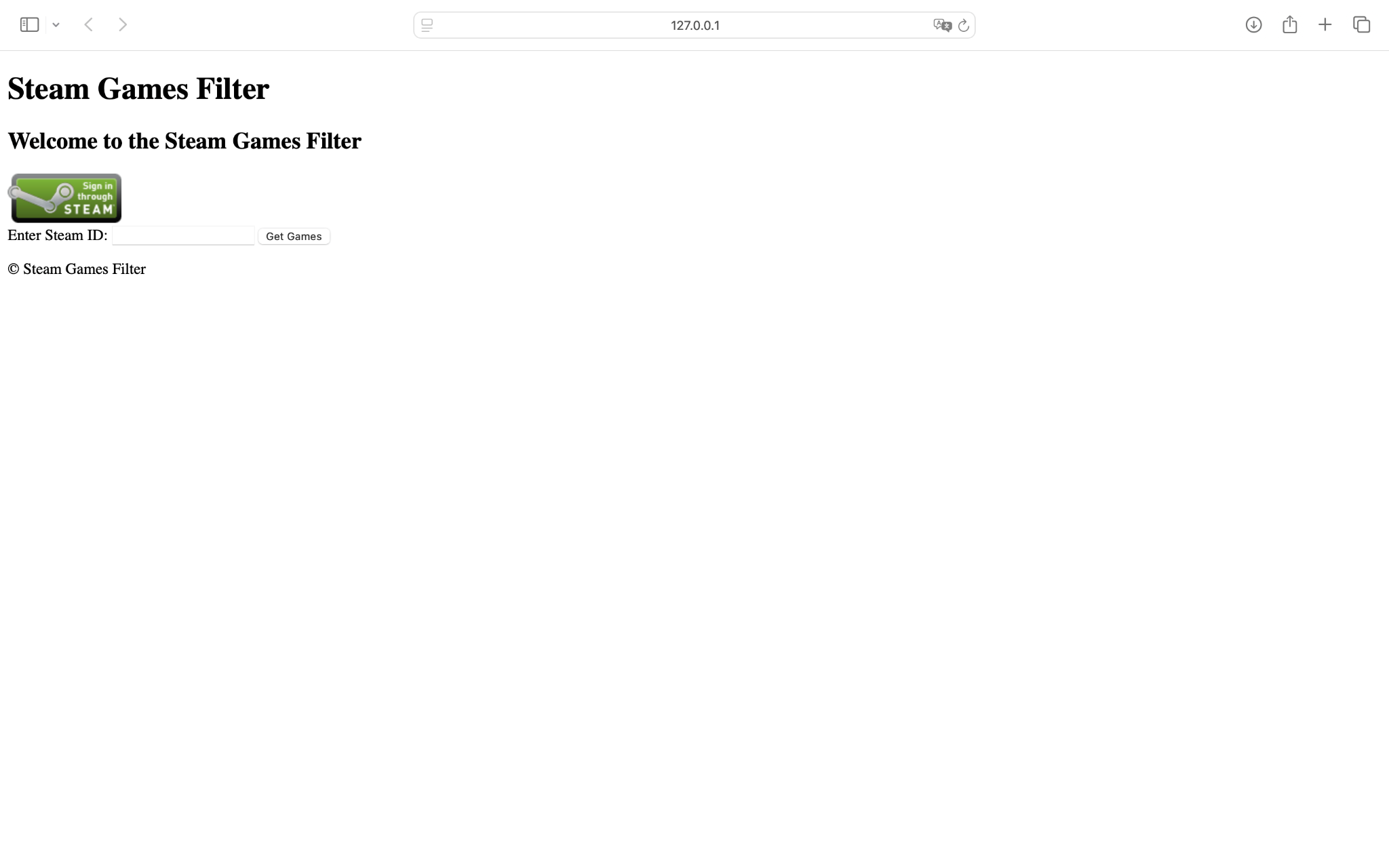
{% endif %}

{% endblock %}

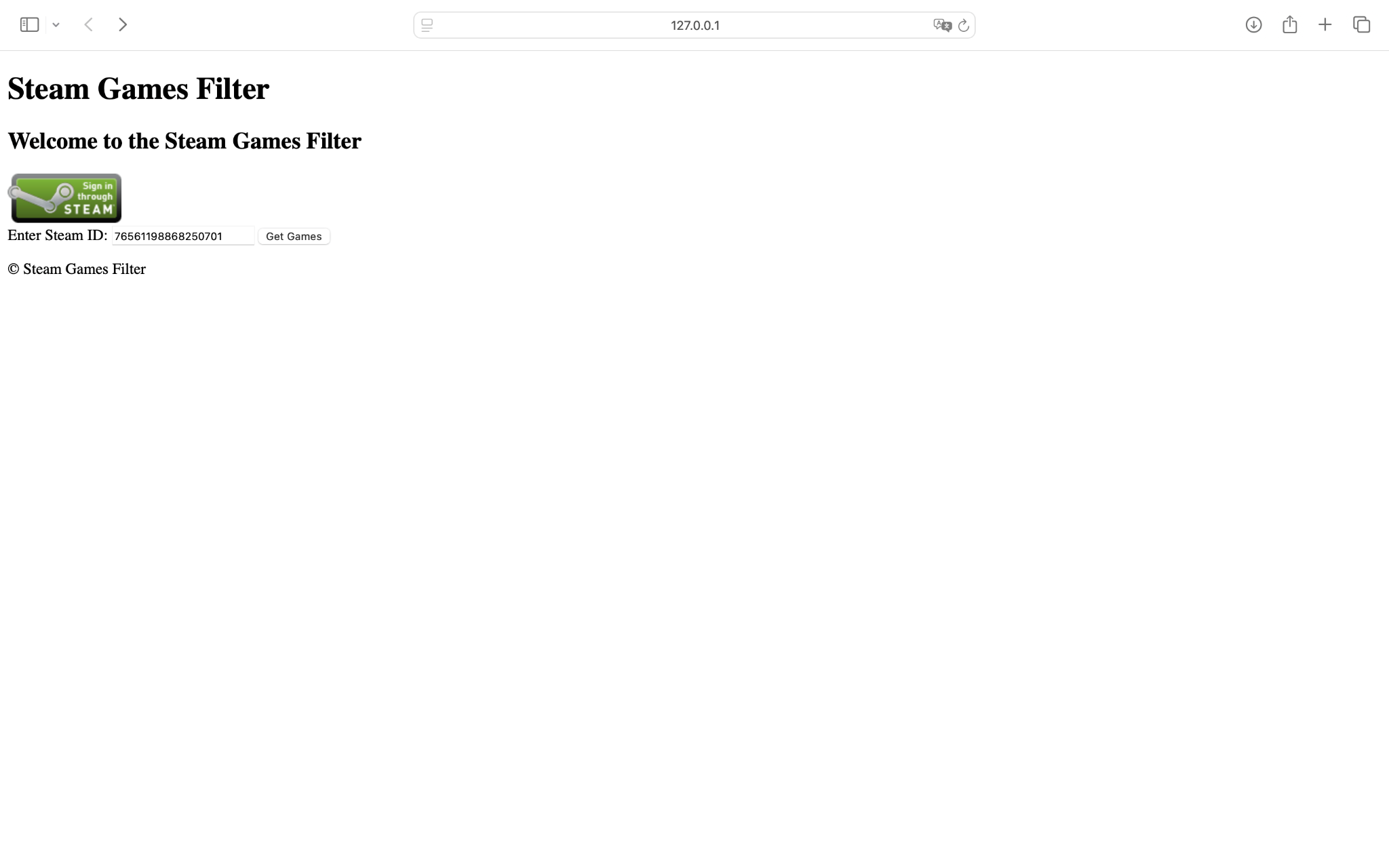
Остальные файлы стандартны для фреймворка Django

**Демонстрация работы программы:**

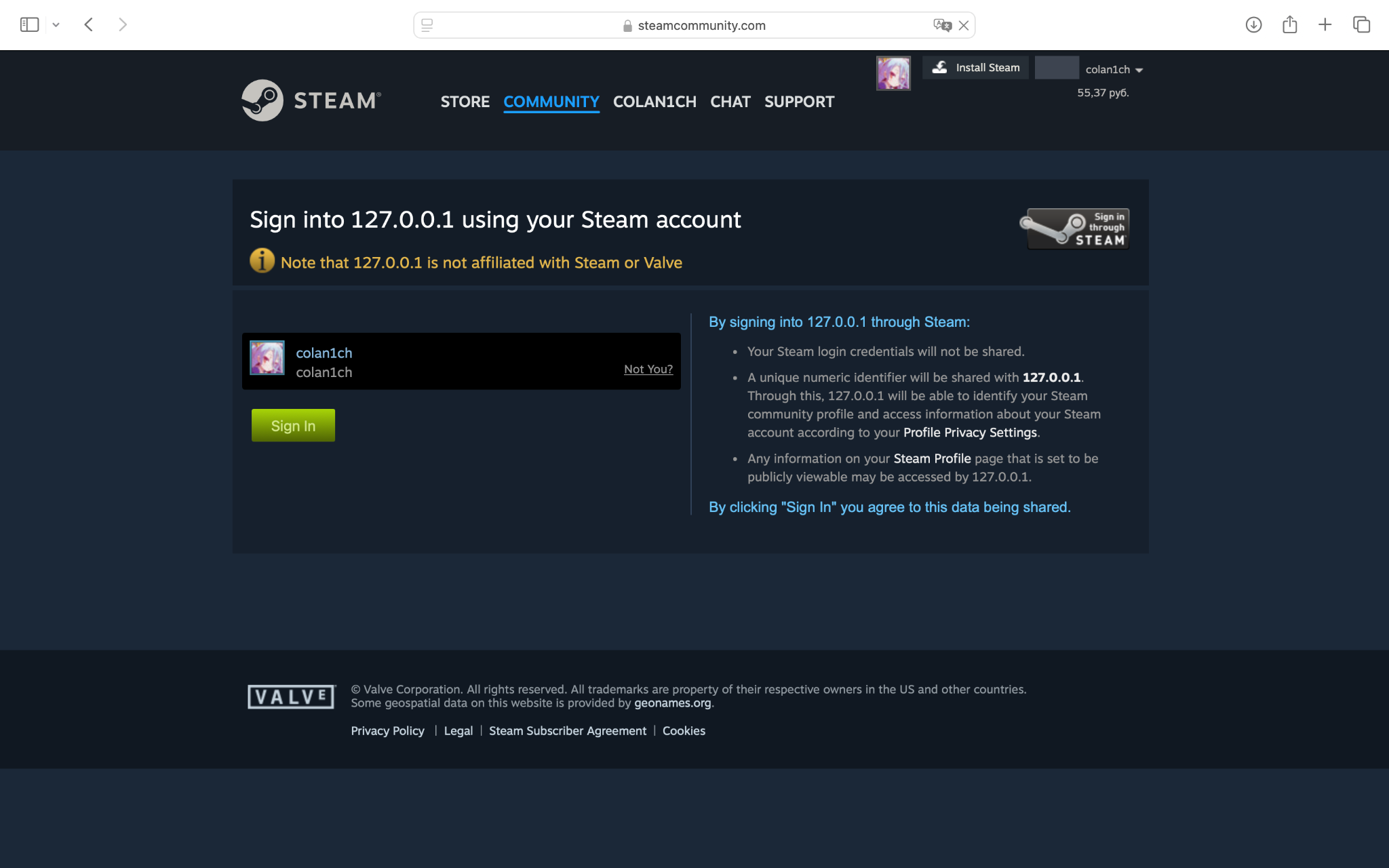
Главная страница



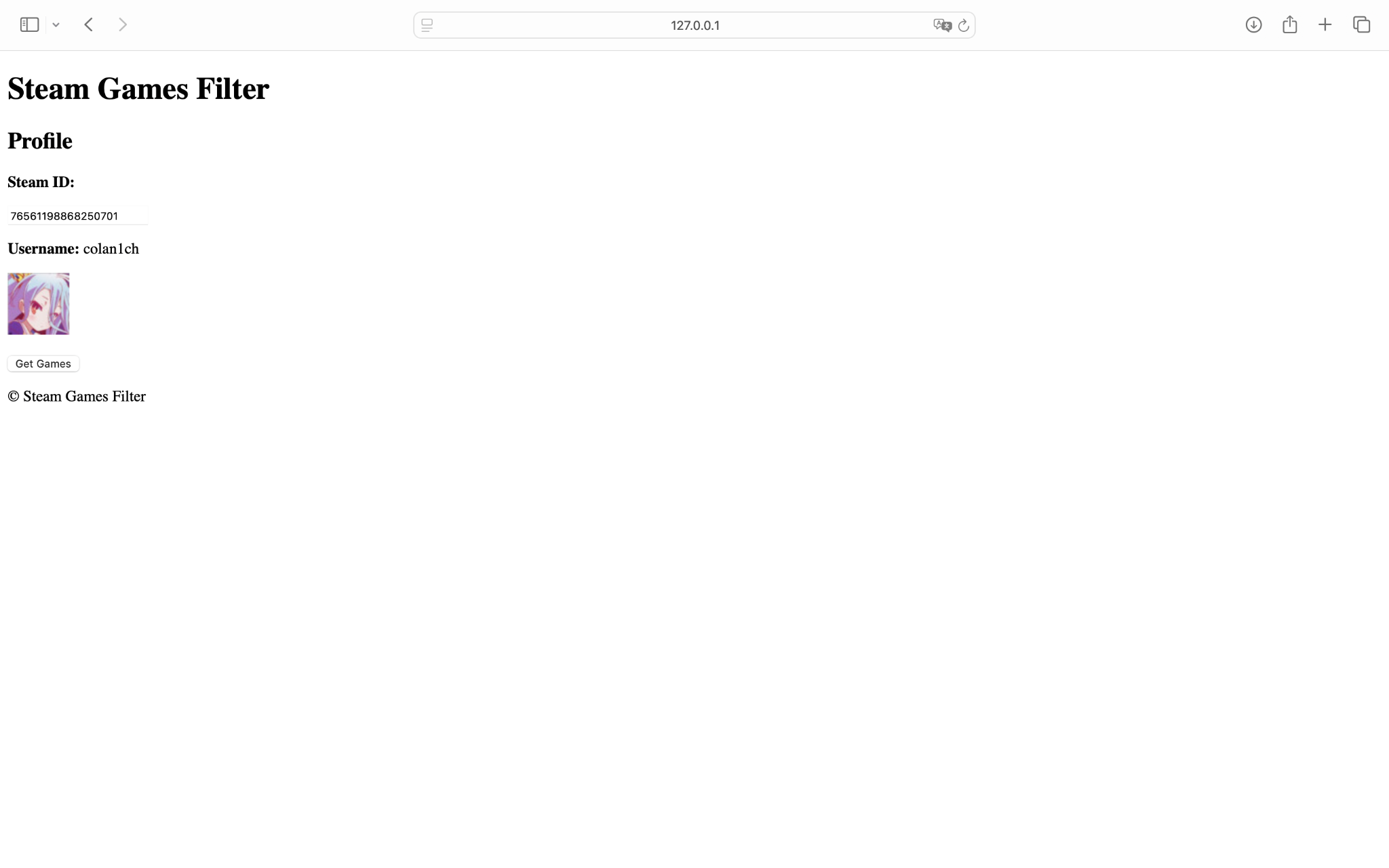
Поиск по Steam ID



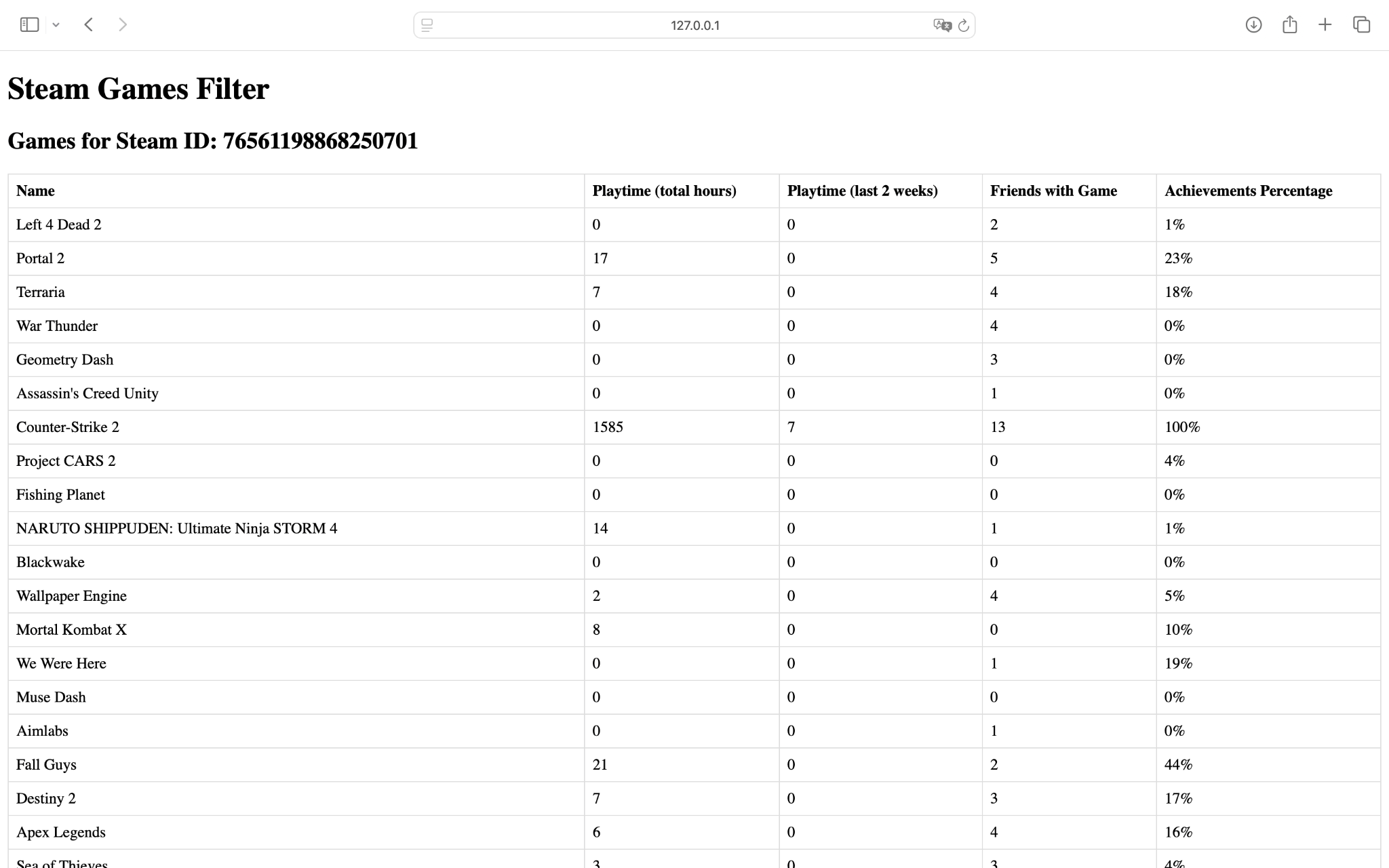
Авторизация через Steam



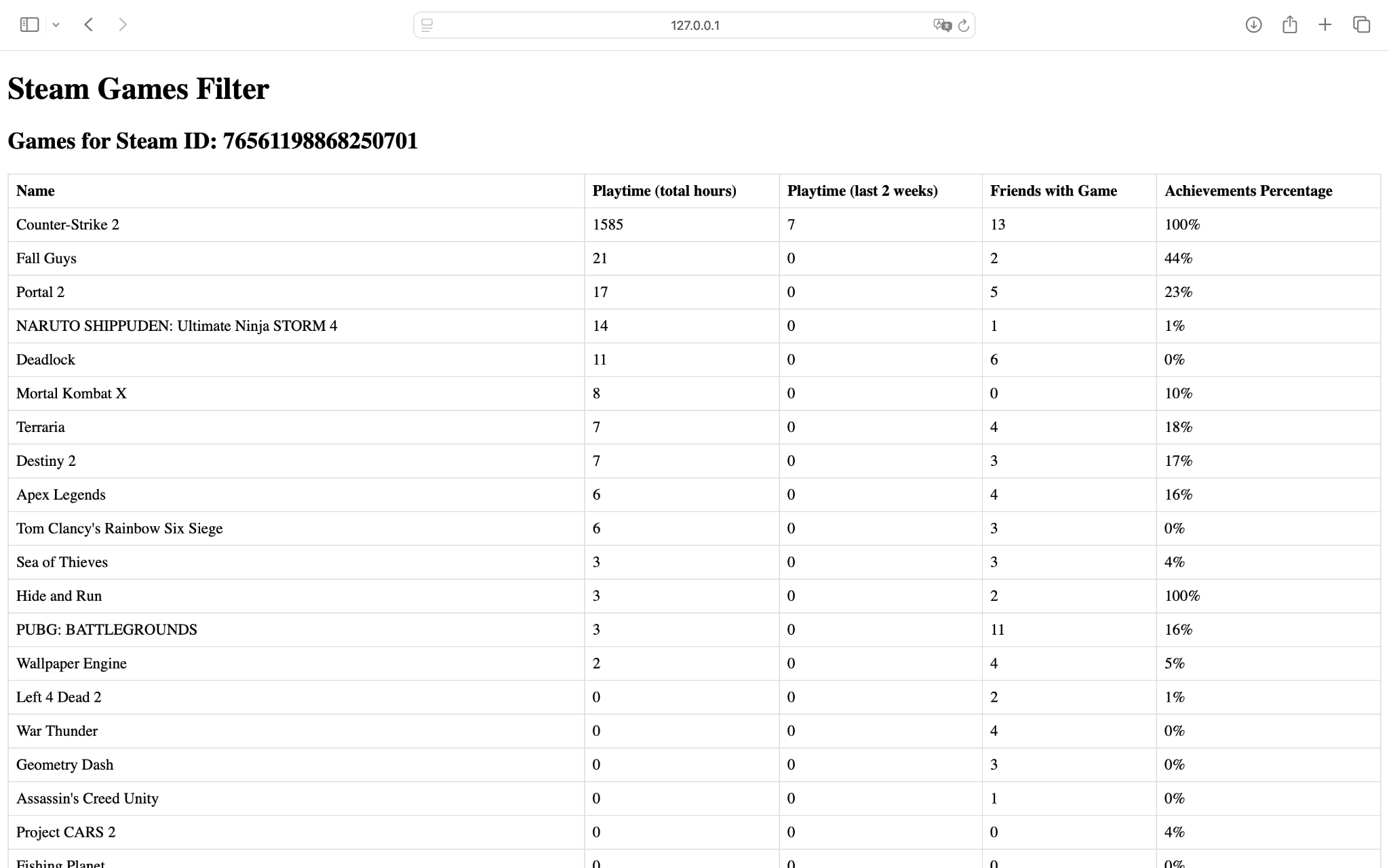
Страница профиля



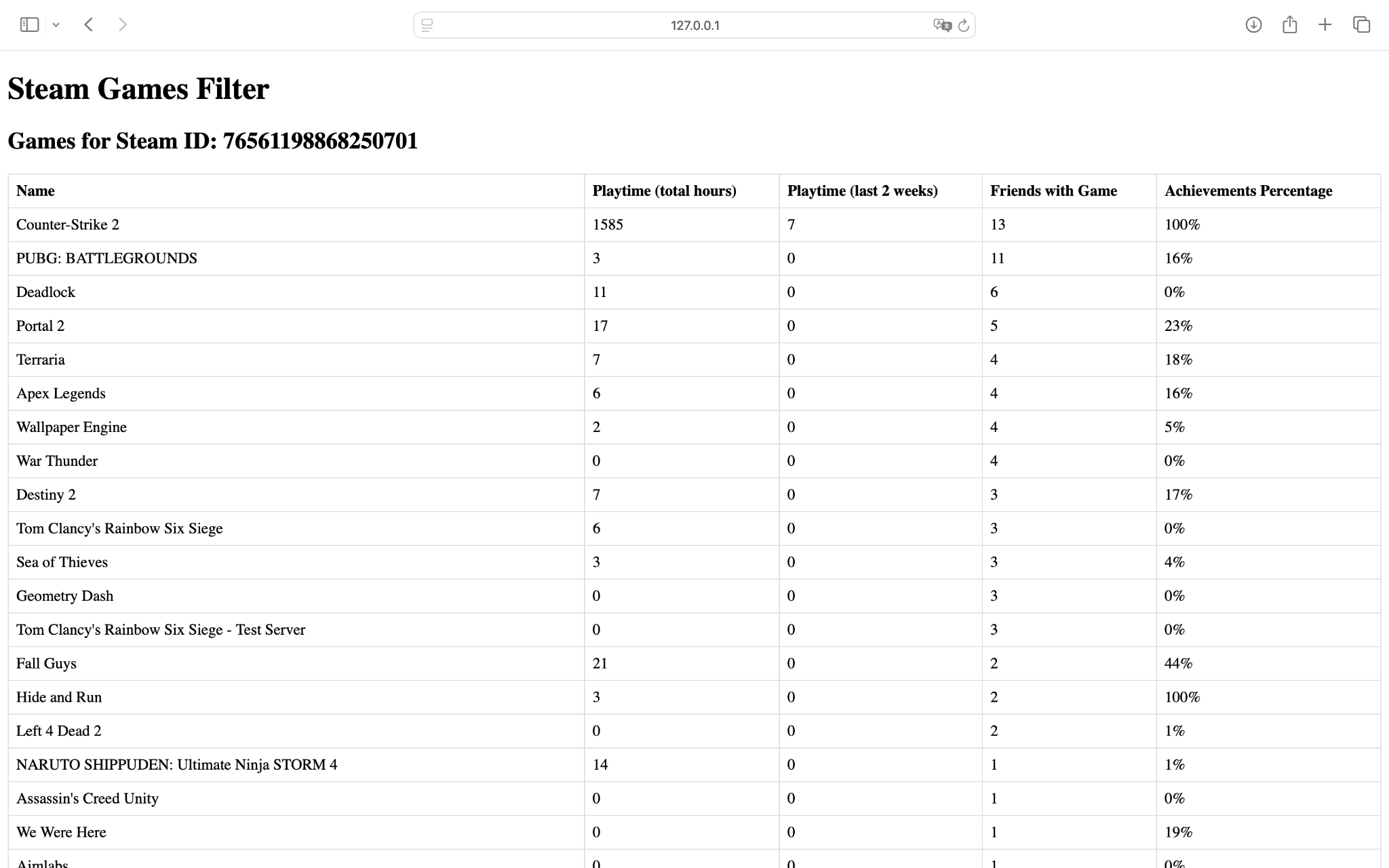
Страница игр



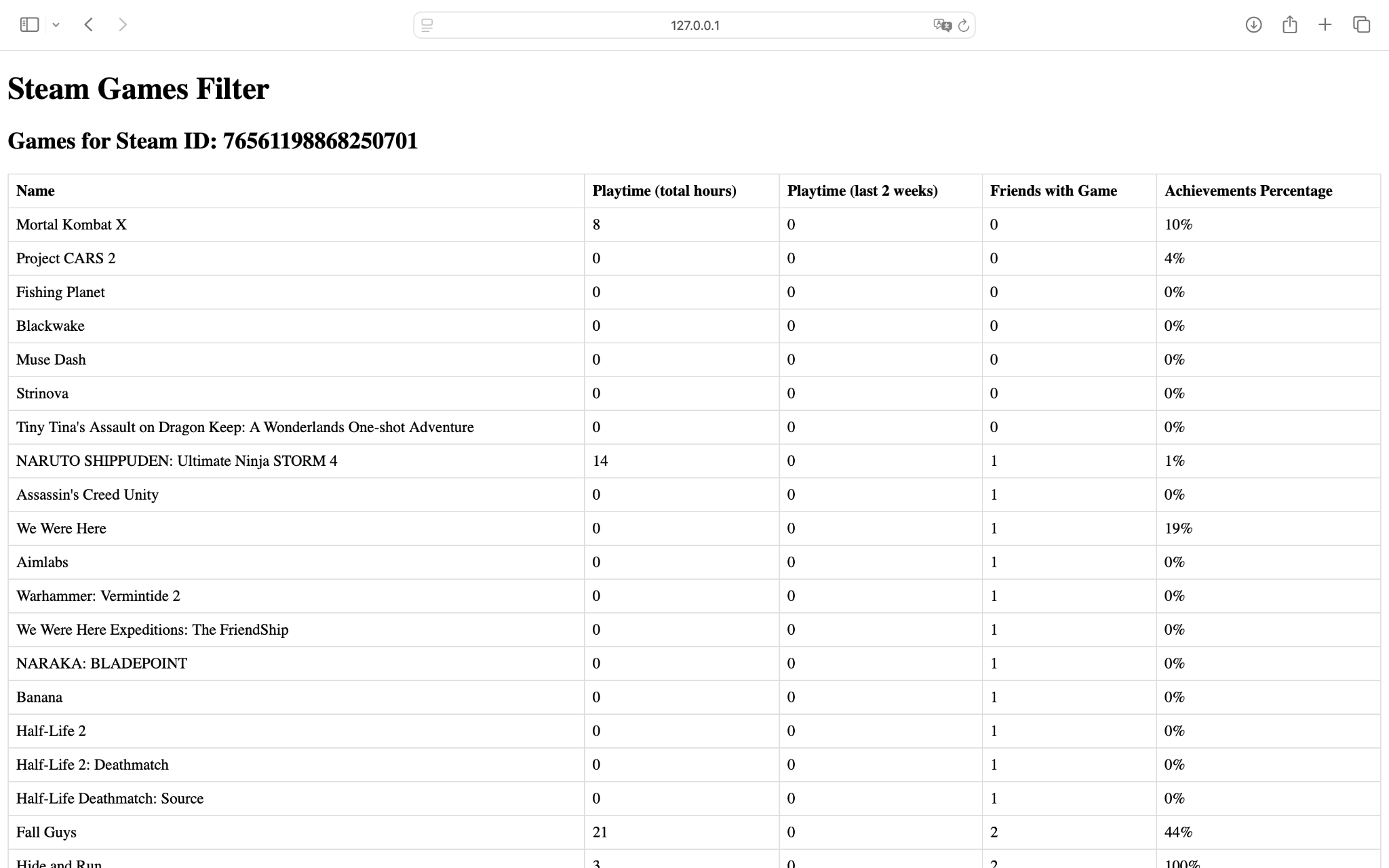
Сортировка по общему времени, проведенному в игре(по убыванию)



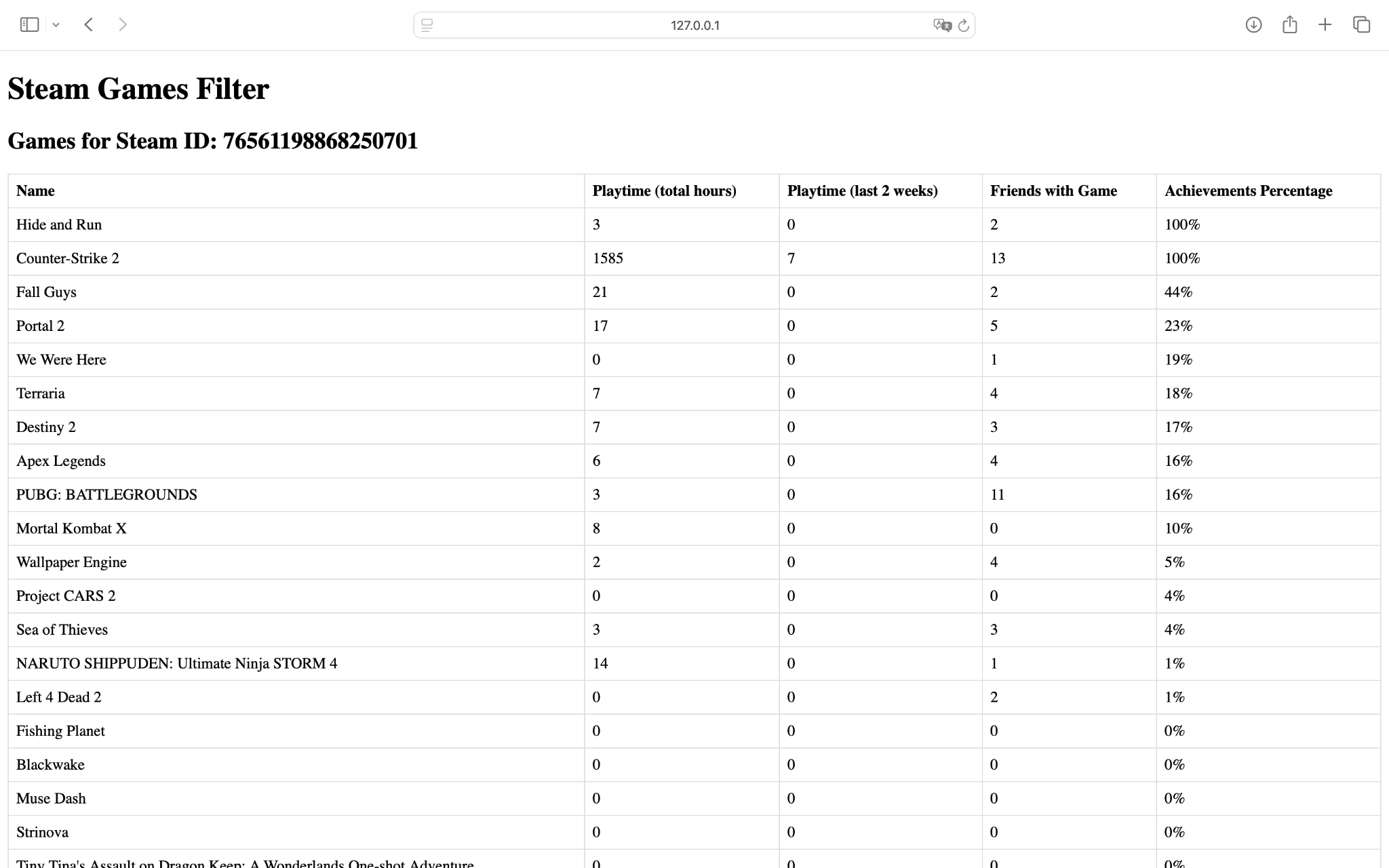
Сортировка по времени в игре за последние 2 недели (по убыванию)



Сортировка по количеству друзей с данной игрой (по возрастанию)



Сортировка по проценту выполненных достижений (по убыванию)



Сортировка по названию (по возрастанию)

