|  |
| --- |
| FIIT STU |
| DBS zadanie 2 |
| Dokumentácia Databázové Systémy |

|  |
| --- |
| Norbert Matuška3-9-2024 |

# Contents

[FIIT STU 0](#_Toc161004425)

[DBS zadanie 2 0](#_Toc161004426)

[Dokumentácia Databázové Systémy 0](#_Toc161004427)

[Norbert Matuška 0](#_Toc161004428)

[3-9-2024 0](#_Toc161004429)

[Contents 1](#_Toc161004430)

[GET /v2/posts/:post id/users 2](#_Toc161004431)

[Example postID 1819157 3](#_Toc161004432)

[GET /v2/users/:user id/friends 4](#_Toc161004433)

[Example user ID 1076348 5](#_Toc161004434)

[GET /v2/tags/:tagname/stats 6](#_Toc161004435)

[Example tag name ´linux´ 7](#_Toc161004436)

[GET /v2/posts/?duration=:duration in minutes&limit=:limit 8](#_Toc161004437)

[Example duration 5 limit 2 9](#_Toc161004438)

[GET /v2/posts?limit=:limit&query=:query 10](#_Toc161004439)

[Example query ´linux´ a limit 1 11](#_Toc161004440)

# GET /v2/posts/:post id/users

cur.execute("""  
SELECT  
 u.id,  
 u.reputation,  
 u.creationdate,  
 u.displayname,  
 u.lastaccessdate,  
 u.websiteurl,  
 u.location,  
 u.aboutme,  
 u.views,  
 u.upvotes,  
 u.downvotes,  
 u.profileimageurl,  
 u.age,  
 u.accountid  
FROM  
 users u  
INNER JOIN comments c ON u.id = c.userid  
WHERE  
 c.postid = %s  
ORDER BY  
 c.creationdate DESC  
""", (post\_id,))  
  
users\_data = cur.fetchall()  
cur.close()  
conn.close()  
return {"items": users\_data}

Najprv SELECTnem všetky dôležité údaje, ktoré sa od nás vyžadujú, spravím INNER JOIN usera a comments kde sa users ID zhodujú v oboch a nakoniec iba porovnávam vo WHERE či sa post ID zhodujé s požadovaným post ID.

## Example postID 1819157

A screen shot of a computer program

Description automatically generated

# GET /v2/users/:user id/friends

cur.execute("""  
SELECT  
 u.id,  
 u.reputation,  
 u.creationdate,  
 u.displayname,  
 u.lastaccessdate,  
 u.websiteurl,  
 u.location,  
 u.aboutme,  
 u.views,  
 u.upvotes,  
 u.downvotes,  
 u.profileimageurl,  
 u.age,  
 u.accountid  
FROM  
 users u  
WHERE  
 EXISTS (  
 SELECT 1  
 FROM comments c  
 INNER JOIN posts p ON c.postid = p.id  
 WHERE (p.owneruserid = %s OR c.userid = %s)  
 AND u.id = c.userid  
 )  
ORDER BY  
 u.creationdate;  
""", (user\_id, user\_id))  
  
friends\_data = cur.fetchall()  
cur.close()  
conn.close()  
return {"items": friends\_data}

V podmienke pri tejto query kontrolujeme pomocou subquery, či existujú nejaké záznamy v tabuľke comments kde user je buď vlastníkom príspevku alebo k nemu pridal komentár. Ak existuje, zahrnieme ho do výsledku.

## Example user ID 1076348

A screen shot of a computer

Description automatically generated

# GET /v2/tags/:tagname/stats

cur.execute("""  
WITH PostDay AS (  
 SELECT  
 EXTRACT(DOW FROM p.creationdate) AS day\_of\_week,  
 COUNT(DISTINCT p.id) AS day\_count\_tag  
 FROM posts p  
 JOIN post\_tags pt ON p.id = pt.post\_id  
 JOIN tags t ON pt.tag\_id = t.id  
 WHERE t.tagname = %s  
 GROUP BY day\_of\_week  
),  
TotalPosts AS (  
 SELECT  
 EXTRACT(DOW FROM p.creationdate) AS day\_of\_week,  
 COUNT(DISTINCT p.id) AS day\_count  
 FROM posts p  
 JOIN post\_tags pt ON p.id = pt.post\_id  
 JOIN tags t ON pt.tag\_id = t.id  
 GROUP BY day\_of\_week  
)  
SELECT  
 CASE  
 WHEN pd.day\_of\_week = 0 THEN 'sunday'  
 WHEN pd.day\_of\_week = 1 THEN 'monday'  
 WHEN pd.day\_of\_week = 2 THEN 'tuesday'  
 WHEN pd.day\_of\_week = 3 THEN 'wednesday'  
 WHEN pd.day\_of\_week = 4 THEN 'thursday'  
 WHEN pd.day\_of\_week = 5 THEN 'friday'  
 WHEN pd.day\_of\_week = 6 THEN 'saturday'  
 END AS day,  
 ROUND((pd.day\_count\_tag::NUMERIC/tp.day\_count::NUMERIC) \* 100, 2) AS day\_percentage  
FROM PostDay pd  
JOIN TotalPosts tp ON tp.day\_of\_week = pd.day\_of\_week  
ORDER BY pd.day\_of\_week;  
 """, (tagname,))  
  
 stats\_data = cur.fetchall()  
 cur.close()  
 conn.close()  
  
 stats\_result = {"result": {day['day']: day['day\_percentage'] for day in stats\_data}}  
 return stats\_result

V prvom CTE (Common Table Expression) PostDay extractujem deň v týždni (DOW) a počítam počet postov podľa daného tagu. Potom v CTE TotalPosts už počítam všetky posty. Nakoniec iba predelím pre každý deň tagnute posty/všetky posty.

## Example tag name ´linux´

A screenshot of a computer

Description automatically generated

# GET /v2/posts/?duration=:duration in minutes&limit=:limit

cur.execute("""  
SELECT  
 id,  
 creationdate,  
 viewcount,  
 lasteditdate,  
 lastactivitydate,  
 title,  
 closeddate,  
 ROUND(EXTRACT(EPOCH FROM (closeddate - creationdate)) / 60, 2) AS duration  
FROM  
 posts  
WHERE  
 closeddate IS NOT NULL  
 AND EXTRACT(EPOCH FROM (closeddate - creationdate)) / 60 <= %s  
ORDER BY  
 creationdate DESC  
LIMIT  
 %s;  
 """, (duration, limit))  
  
 posts\_data = cur.fetchall()  
 cur.close()  
 conn.close()  
 return {"items": posts\_data}

Pomocou EPOCH konvertujem čas na sekundy, predelím ho 60 aby z neho boli minúty. V podmienke iba pozerám či bol post zatvorený a či trval menej ako požadovaný čas.

## Example duration 5 limit 2

A screenshot of a computer program

Description automatically generated

# GET /v2/posts?limit=:limit&query=:query

cur.execute("""  
SELECT  
 p.id,  
 p.creationdate,  
 p.viewcount,  
 p.lasteditdate,  
 p.lastactivitydate,  
 p.title,  
 p.body,  
 p.answercount,  
 p.closeddate,  
 ARRAY\_AGG(t.tagname) AS tags  
FROM  
 posts p  
JOIN  
 post\_tags pt ON p.id = pt.post\_id  
JOIN  
 tags t ON pt.tag\_id = t.id  
WHERE  
 (LOWER(UNACCENT(p.title)) LIKE %s OR  
 LOWER(UNACCENT(p.body)) LIKE %s)  
GROUP BY  
 p.id, p.creationdate, p.viewcount, p.lasteditdate, p.lastactivitydate,  
 p.title, p.body, p.answercount, p.closeddate  
ORDER BY  
 p.creationdate DESC  
LIMIT  
 %s;

Podľa zadaného stringu hľadám, či sa objavuje niekde v nadpise alebo v body postu. LOWER a UNACCENT používam aby hľadanie bolo case-insensitive a accent-insensitive. ARRAY\_AGG používam preto, aby posty s viacerými tagmi neboli zobrazené viackrát.

## Example query ´linux´ a limit 1

A screen shot of a computer program

Description automatically generated