פרויקט סיום NodeJS - Nodebook

עליכם לבנות שרת של רשת חברתית בשם Nodebook שמכילה משתמשים שמחפשים להכיר חברים ובני זוג.

הדאטה עבור התרגיל:

בשרת קיימת הישות **משתמש** – **User** שמכילה את השדות הבאים:

- id – integer מזהה רץ אוטומטית
- first_name - string
- last_name - string
- phone_number - string
- location: {x, y} – נקודה גאוגרפית
- gender – "male" / "female"
- relationship_status – "single" / "in a relationship"
- interested_in – "male" / "female"
- hobbies – Array of hobbies. Example: ["Programming", "Dancing"]
- friends – Array of user ids

חלק ראשון:

1. GET /api/user/:id

- כאשר המשתמש קיים – מחזיר קוד OK ומחזיר ב-body ערך json של פרטי המשתמש (ללא רשימת החברים וכולל id – תקף לכל פעולות הGET בתרגיל)
- כאשר המשתמש לא קיים – מחזיר קוד NOT FOUND

<u>רמז</u>: ניתן להשתמש ב- listagg

2. POST /api/user

ביצירת משתמש יש לשלוח ב body את כל השדות (ללא id ,friends), מותר לשלוח מערך hobbies ריק.

- כאשר אין משתמש דומה (אותו מספר טלפון) – יוצר את המשתמש במערכת, מחזיר קוד CREATED ואת ה-id שנוצר
- כאשר יש משתמש דומה יש להחזיר קוד CONFLICT
- אם נשלחה בקשה עם שדות שונים מהתקינים או לא מלאים יש להחזיר קוד BAD REQUEST

3. DELETE /api/user/:id

- מוחק את המשתמש מהמערכת. מחיקת משתמש צריכה למחוק גם את כל התחביבים שלו וכל קשרי החברות שלו. מחזיר קוד OK

4. DELETE /api/users

- מוחק את כל המשתמשים מהמערכת וכל הקשרים שלהם. מחזיר קוד OK

חלק שני:

1. PATCH /api/user/:id

מקבל body עם פרטי משתמש לעדכון, ניתן לעדכן את כל השדות (פרט ל-friends, id), כולל hobbies (מחליף את המערך). שימו לב שניתן לקבל פרטים חלקיים (רק first_name לדוגמה).

- כאשר המשתמש קיים והפרטים תקינים מעדכן את המשתמש במערכת ומחזיר OK
- במידה והמשתמש לא קיים מחזיר קוד NOT FOUND
- במידה ומפתחות הפרטים לא תקינים מחזיר BAD REQUEST

2. GET /api/users

מחזיר ב-body מערך של משתמשים במערכת, יכול לקבל query params על מנת לסנן את התוצאה. על התוצאה להכיל את כל הערכים שהתקבלו בפרמטרים (עבור תחביבים – כל התחביבים). הפרמטרים יכולים להיות כל אחד מהמפתחות התקינים (פרט ל-friends) ונקבל משתמשים שיש להם את הערכים האלה.

- מחזיר את המשתמשים וקוד OK כאשר הפרמטרים תקינים (יכול להיות גם מערך ריק)
- כאשר הפרמטרים לא תקינים, מחזיר קוד BAD REQUEST
- דגש – ניתן לכתוב בקשה כזאת:

  GET /api/users?hobbies=Programming&hobbies=Dancing

  וכך לקבל מספר תחביבים (express יכניס לנו את זה למערך)

  על מנת לחפש לפי מיקום נכתוב כך:

  GET /api/users?locationX=30&locationY=20

  במידה וקיבלנו אחד מהם (X או Y) נדרוש לקבל גם את השני (אחרת נחזיר BAD REQUEST)

3. GET /api/user/:id/friends

מחזיר ב-body מערך של המשתמשים שהם חברים של המשתמש עם ה-id שהתקבל

- במידה והמשתמש קיים מחזיר את החברים שלו עם קוד OK (יכול להיות מערך ריק)
- במידה והמשתמש לא קיים מחזיר NOT FOUND

4. POST /api/user/:id/friends

מקבל ב-body מערך שמכיל מזהים של משתמשים (כ-number). דוגמה – [1, 2, 3]

שומר כל שני המשתמשים כחברים במערכת

- כאשר כל המזהים הם של משתמשים קיימים, שומר אותם במערכת ומחזיר קוד CREATED
- כאשר **לפחות אחד** מהמזהים לא קיים, לא משנה את מצב המערכת ומחזיר קוד NOT FOUND
- כאשר מנסים להוסיף משתמש כחבר של עצמו (ב-body יש את id שנשלח) מחזיר קוד BAD REQUEST
- כאשר לפחות אחד מהמזהים ב-body כבר נשמר כחבר של השתמש (כבר חברים), לא משנה את מצב המערכת ומחזיר קוד CONFLICT

5. DELETE /api/user/:id/friends

מקבל ב-body מערך שמכיל מזהים של משתמשים (כ-number). דוגמה – [1, 2, 3]

מוחק כל זוג (id והערכים מהמערך) כחברים במערכת

- אם ה-id שנמצא ב-path קיים – מוחק את החברים שלו שהתקבלו בבקשה ומחזיר קוד OK
- אם ה-id לא קיים – מחזיר קוד NOT FOUND

חלק שלישי:

1. GET /api/user/:id/suggestions

מחזיר מערך של משתמשים שהם **הצעות חברות** של המשתמש הנוכחי. כלומר הם חברים של חברים שלו (והם לא כבר חברים כרגע).
לדוגמה, אוראל חבר של ספיר, ספיר חברה של דני
לכן ניתן להציע את דני לאוראל ב suggestions

- כאשר המשתמש קיים, מחזיר את התוצאה (יכול להיות מערך ריק) עם קוד OK
- כאשר המשתמש לא קיים, מחזיר קוד NOT FOUND

2. GET /api/user/:id/matches

מחזיר מערך של משתמשים שהם הצעות לפרטנר רומנטי
לפי הדרישות:

1. על המשתמשים להיות עם relationship_status = single
2. על ה-gender להתאים למין שנמצא ב interested_in של המשתמש השני (בשני הכיוונים)
3. התוצאות ימוינו לפי הפרמטרים:
   i. מספר התחביבים המשותפים – common_hobbies
   ii. המרחק הגיאוגרפי – distance

   יש למיין את התוצאה בסדר יורד לפי הציון הבא:

$$common\_hobbies - sqrt(distance)$$

- כאשר המשתמש קיים מחזיר את מערך המשתמשים עם קוד OK
- כאשר המשתמש לא קיים מחזיר קוד NOT FOUND

דגשים כלליים:

- יש להשתמש ב Oracle SQL כ-DB לצורך התרגיל.
- יש לחלק לטבלאות לפי הצורך והרלציות (One-to-many....).
- יש לתת constraints כגון primary / foreign keys לפי הצורך.
- אין לעשות מספר שאילתות לא ידוע בלולאה בקוד (או map, forEach...)
- מותר להשתמש ב-executeBulk של oracledb client
- יש לחפש באינטרנט מה מספר הקוד המתאים לתשובות השרת
- יש לכתוב End to end tests **לכל הנתיבים** ולהשתדל לכסות כמה שיותר מקרים
- מומלץ לכתוב Integration, Unit tests במקומות המתאימים
- יש לשים את פקודות יצירת הטבלאות של SQL בתיקיית SQL בתרגיל (כפי שנמצא בתיקיית ה-template)
- יש להשתמש בארכיטקטורה נכונה של NodeJS
- יש להשתמש ב Promise / async await ולא ב- callbacks
- על השרת לרוץ על פורט 5000 (כך זה ייבדק)
- ספריות מומלצות לשימוש:
  - Express, oracledb, lodash, http-status
  - לטסטים – mocha, chai, supertest