Apollo GraphQL

מבוא

רכיב הApollo QraphQL בפרויקט מהווה את הממשק המרכזי בין האתר הראשי Lionfish לבין ה data bases מבוססי הPostgreSQL והMongoDB. רכיב זה מאפשר תקשורת מסודרת, מאובטחת ומבוססת מבנה עם הData bases באמצעות GraphQL ומספק תשתית יעילה לשליחת שאילתות לPostgreSQL ולחיבור לניהול השאילתות בMongoDB.

ייעוד

הייעוד של רכיב Apollo GraphQL הוא להוות שכבת ממשק מאובטחת וגמישה בין האתר Lionfish לבין הdata bases מבוססי הPostgreSQL והMongoDB. הוא מאפשר שליחת שאילתות מובנות בצורה מבוקרת, מבצע המרה של שאילתות GraphQL לשאילתות SQL ומאפשר הרצה של שאילתות מסוגים מובנים מתוך הMongoDB database על הPostgreSQL database.
רכיב זה מבטיח שכל אינטראקציה עם הנתונים נעשית דרך API מוגדר ומבוקר, ללא גישה ישירה לPostgreSQL database מהאתרים.

תפקוד

מול הלקוח:
משתמשים באתר Lionfish משתמשים בממשק הגרפי שמאפשר הרכבת שאילתה. ההרכבה נעשית באמצעות הצבת ערכי פרמטרים לשאילתה שבוחרים באתר מבין השאילתות המוחזקות ב MongoDB database. בבחירתם, המשתמשים למעשה שולחים שאילתות GraphQL אל שרת Apollo. השרת מחזיר את התוצאות בפורמט GraphQL בהתאם לדרישות השאילתות, כשהתשובות מעובדות ומעוצבות לפי המבנה שהוגדר בסכימה.

מול הdatabases:
Apollo GQL מקבל את השאילתות מהלקוח בפורמט GraphQL, ממיר אותן לשאילתות SQL המתבצעות על הPostgreSQL DB ומחזיר את התוצאות ללקוח.
רכיב הApollo GQL אחראי גם על ההתממשקות עם הMongoDB שמחזיק את כלל השאילתות שנבנו וניתן להריץ מאתר Lionfish על הPostgreSQL DB דרך Apollo GQL, ומאפשר את הניהול שלהן.

רעיון המימוש

השימוש ב Apollo GraphQL בפרויקט מושתת על התממשקות עם שני databases: אחד מסוג PostgreSQL ואחד מסוג MongoDB.
המימוש מבוסס על הגדרת מבנה השאילתות והתגובות באמצעות סכמה והגדרת פונקציות resolvers הממשות את הפעולות השונות של הGraphQL API. הפונקציות הללו מקשרות בין שאילתות

GraphQL המתקבלות מהלקוח לבין שאילתות הSQL שמורצות על הPostgreSQL DB. כך, הGraphQL API משמש כגשר בין הלקוח לdatabases מבלי שהלקוח פונה ישירות לdatabases.

השרת מאזין על פורט 4000 ומקבל שאילתות מהלקוח דרך הGraphQL API.
שאילתות SQL מתבצעות באמצעות הפונקציה runQuery, שמבצעת מחרוזת שאילתה ומחזירה את התוצאה.
שאילתות MongoDB מנוהלות באמצעות runMongo,.

הresolvers הם פונקציות שמבצעות את הפעולות הנדרשות על הdatabases כאשר מתקבלות שאילתות GraphQL מהלקוח.

<u>Query Resolvers</u>:

- get_flights: מבצע שאילתת SQL דינאמית לשליפת מזהי טיסות וזמני ההקלטה שלהם על סמך תת-שאילתה שסופקה על ידי המשתמש.
- heat_map: מריץ שאילתת SQL שסופקה כארגומנט ליצירת מפת חום.
- marker_map: משמש ליצירת נתונים לסימונים על מפה.
- flight: שולף טיסה על פי מזהה.
- row: שולף שורות ספציפיות מטבלאות slow_params וfast_params בהתבסס על מזהה טיסה ומספר חבילה (packet).
- param: מחפש ערך פרמטר בטבלאות slow_params וfast_params.
- insert_query: מוסיף מסמך שאילתה חדש ל-MongoDB ושולף מסמך בדיקה בשם 'test1'.
- get_queries: שולף את כל השאילתות השמורות מ-MongoDB.

<u>Flight Resolvers</u>:

- row: שולף שורה מטבלת fast_params בהתבסס על מזהה הטיסה והחבילה של האובייקט האב.
- row_by_time: שולף שורה מ-slow_params או fast_params על ידי התאמה של חותמת זמן בתוך טווח מסוים.
- heatmap_from_rows: יוצר מפת חום על ידי בחירת נקודות נתונים ספציפיות מטבלאות slow_params וfast_params.

<u>Row Resolvers</u>:

- flight: שולף את פרטי הטיסה הקשורים לשורה.
- param: שולף פרמטר ספציפי מ-slow_params או fast_params בהתבסס על מזהה הטיסה והחבילה.
- params: שולף מספר פרמטרים, וממזג תוצאות מslow_params וfast_params.

Param Resolvers:

- row: מחפשת ומחזירה את השורה שמתאימה לנתוני פרמטר מסוים.

קיימות גם פונקציות עזר:

getCols(table):

שולפת את שמות העמודות של טבלה נתונה מבסיס הנתונים של PostgreSQL ומחזירה אותן כSet.

runQuery(s):

מבצעת שאילתת SQL ומחזירה את התוצאה. אם מתרחשת שגיאה, היא נרשמת לקונסולה ומוחזרת הודעת שגיאה.

runMongo(s):
פונקציה זו מיועדת לביצוע שאילתות MongoDB אך אינה בשימוש בגרסה הנוכחית של הקוד.

כמו כן, קיים קובץ עיקרי נוסף: schema.js. קובץ זה מגדיר את סכמת ה-GraphQL המשמשת לשרת הApollo. סכמת ה-GraphQL קובעת את סוגי הנתונים (Types) ואת השאילתות (Queries) הזמינות במערכת, ומאפשרת לבצע שאילתות על בסיס מבני הנתונים המוגדרים בה.