**Express – Create a New Server**

במדריך הבא נלמד לבנות שרת בNodeJs ונעזר בExpress framework - .

1. נפתח את Terminal ונפתח פרוייקט חדש ע"י npm –y init:
2. נתקין את התלויות הבאות:
   * Npm I –save-dev nodemon body-parser @types/body-parser ts-node
   * Npm I –save express @types/express
3. כדי שנוכל להפעיל את השרת בnpm start, בpackage.json - בתוך scripts הוסיפו את השורה הבאה:

    "start": "nodemon --exec ts-node src/app.ts"

1. אנו נשתמש בקובץ json כdata ולשם כך נצטרך להגדיר כמה דברים שיאפשרו לנו לקרוא אותו. צרו קובץ חדש מחוץ לתקייה src בשם tsconfig.json והוסיפו את ההגדרות הבאות:

{

  "compilerOptions": {

    "resolveJsonModule": true,

    "esModuleInterop": true

  }

}

1. לאחר שהתקנו את התלויות נפתח תיקיה בשם src ובתוכה קובץ חדש בשם app.ts
2. בתוך הקובץ נעשה import לexpress ולtypes שאנו צריכים:

import express, { Request, Response, Router } from "express";

1. לאחר מכן ניצור instance לexpress כך:

const app = express();

1. נייבא את body-parser ונכניס לתוך פונקציית app.use כדי שנוכל לעבד מידע מקובץ Json

(ללא body-parser לא נוכל לקרוא מקבצי (Json.

1. כעת נייבא את קובץ placesRoutes שבנינו:

import placesRoutes from "./routes/places\_routes";

1. עלינו להשתמש בפונקציה app.use כדי להשתמש בקובץ routes שבנינו וייבאנו.

כאן בעצם אנחנו אומרים לexpress לקבוע את הנתיב(route) בצד שמאל את נתיב הבסיס לapi ממנו יצאו כל הענפים הנוספים. בנוסף, מימין לנתיב נעביר את הפונקציה המנהלת את הroutes.

app.use\* מאפשר לנו לקבוע את סדר ההפעלה של כל פעולה שנרצה לעשות ע"י מיקום השורה בצורה סינכרונית בקוד.

זה נראה כך:

app.use("/api/places", placesRoutes);

1. כעת, עלינו לנהל שגיאות לגבי הנתיבים בלבד אם יגיעו. נעשה זאת ע"י שימוש בפונקציה שפתחנו מבעוד מועד בשם HttpError שנועדה להציג את קוד השגיאה והודעה כאינדיקציה למה שהשתבש וגרם לשגיאה.

app.use((req:Request, res:Response, next:NextFunction) => {

const error = HttpError('Could not find this route.', 404);

throw error;

});

1. הפונקציה הבאה גם נועדה לטפל בשגיאות אבל בקשת רחבה יותר של שגיאות מכל מיני סוגים.

אסביר את הפונקציה :

הבדיקה שאנו רואים בודקת האם הclient קיבל את הheaders של response.

זה חשוב כי במידה וכבר קיבל אותם, לא נוכל לעשות כלום עם הerror ולכן נעביר את הerror בתוך פונקציית next לפונקצית הmiddleware הבאה. שבמקרה זה אין לנו.

בחלק שאחרי הבדיקה ניקח את הקוד מerror ונגדיר אותו כסטטוס לשגיאה. במידה ואין, נציג 500 – שגיאה פנימית בשרת. ובנוסף נציג הודעה.

הפונקציה נראית כך:

app.use((error:any, req, res, next) => {

  if (res.headerSent) {

    return next(error);

  }

  res.status(error.code || 500)

  res.json({message: error.message || 'An unknown error occurred!'});

});

1. כעת, נגדיר פורט להאזנה כך:

app.listen(5000);

**זהו פתחנו שרת כעת ניתן לבדוק את כתובת הבסיס שהגדרנו בroute הבסיסי ולראות האם בurl שהגדרנו מוצג מידע.**

לבסוף app.ts צריך להראות כך:

iimport express, { NextFunction, Request, Response, Router } from "express";

import bodyParser from "body-parser";

import placesRoutes from "./routes/places\_routes";

import HttpError from "./models/Http\_Error";

const app = express();

app.use(bodyParser.json());

app.use("/api/places", placesRoutes);

app.use((req:Request, res:Response, next:NextFunction) => {

  const error = HttpError('Could not find this route.', 404);

  throw error;

});

app.use((error:any, req, res, next) => {

  if (res.headerSent) {

    return next(error);

  }

  res.status(error.code || 500)

  res.json({message: error.message || 'An unknown error occurred!'});

});

app.listen(5000);