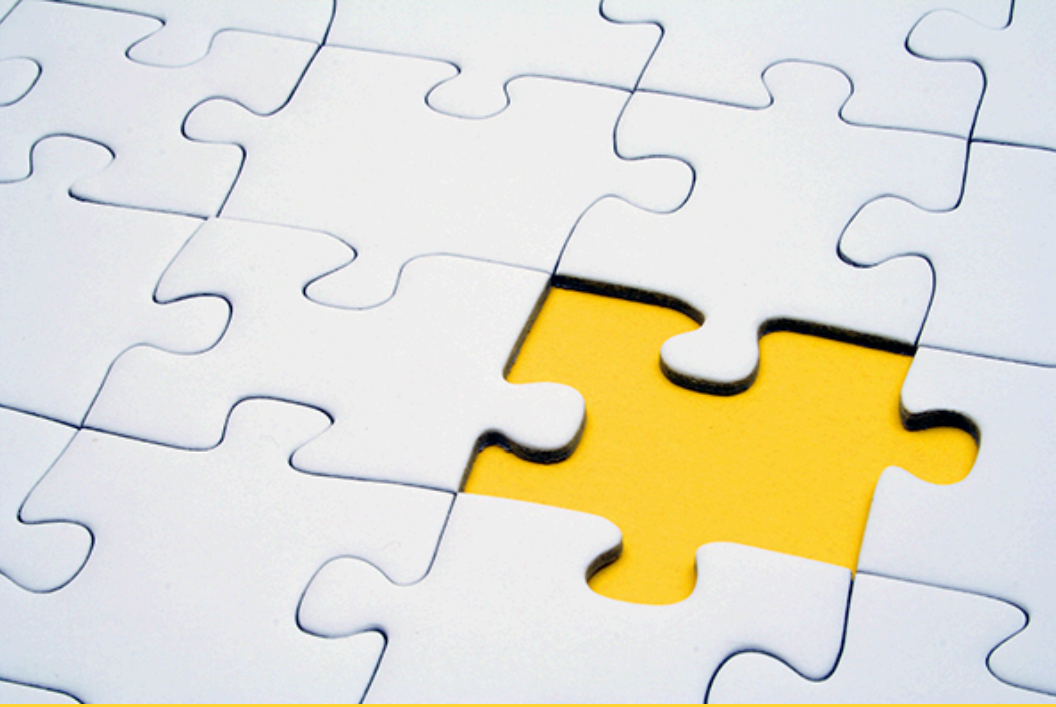


# Node js

Express and MongoDB book *By* keynan perez

# Lession 1

## Advanced JS

### ANONUMOUS FUNCTION

```
let f1 = function(x)
{
    return x*2;
}

let result = f1(4)
console.log(result);

f2 = x => x*2

console.log( f2(7))

f3 = (x,y) =>
{
    //Some code..
    return x+y;
}
```

### HIGH ORDER FUNCTION FOREACH

```
let arr = [5,2,3,7];

arr.forEach(x => console.log(x*2));
```

### HIGH ORDER FUNCTION MAP

```
let arr = [5,2,3,7];

let arr2 = arr.map(x => x*2)

console.log(arr)
console.log(arr2)
```

**********

```
let arr = [{name : "Avi" , age : 20},
           {name : "Dana" , age : 30},
           {name : "Gil" , age : 40}];


let arr2 = arr.map(x => x.age)
console.log(arr2)
```

high_order_function_filter

```
let arr = [5,7,1,5,9,3,12,65,27]

let arr2 = arr.filter(x => x > 10)

console.log(arr)
console.log(arr2)
```

## HIGH ORDER FUNCTION REDUCE

```
let arr = [5,2,3,6]

let total = arr.reduce((x,y) => x+y)

console.log(total);
```

# FUNCTIONPOINTER

```
function sayHello()
{
    console.log("Hello");
}

function exec_func(f)
{
    f();
}

exec_func(sayHello);
```

# CALLBACK

```
setTimeout( () => console.log("OK") ,1000)
console.log("End");
console.log("End2");
console.log("End3");
console.log("End4");
```

# PROMISE

```
const prom = new Promise(resolve =>
    {
        setTimeout( () =>
        {
            resolve("OK")
        },1000)
    })
prom.then( data => console.log(data) );

//some code...
console.log(“End")
```

# PROMISEWITHFUNCTION

```
/*
Rule 1 - An async function us a function with ANY async function
Rule 2 = any async function can returns ONLY Promise
*/


function getAsyncData(num)
{
   return new Promise(resolve =>
    {
        setTimeout( () => resolve(num*2),1000)
    })
}

getAsyncData(5).then(data => console.log(data));
```

```
//Code...
console.log("End")
```

# PROMISE

```
/*
Rule 1 - An async function us a function with ANY async function
Rule 2 = any async function can returns ONLY Promise
*/


function getAsyncData(num)
{
   return new Promise(resolve =>
    {
        setTimeout( () => resolve(num*2),1000)
    })
}

//Option 1
function getAsyncData2()
{
    getAsyncData(6).then(data => console.log(data))
}

//Option 2
async function getAsyncData3()
{
    let data = await getAsyncData(5)
    console.log(data)
}

getAsyncData3()
console.log("End")
```

**********

```
/*
Rule 1 - An async function us a function with ANY async function
Rule 2 = any async function can returns ONLY Promise
*/


function getAsyncData(num)
{
   return new Promise(resolve =>
    {
        setTimeout( () => resolve(num*2),1000)
    })
}

//Option 1
function getAsyncData2()
{
    return new Promise(resolve =>
        {
```

```
            getAsyncData(6).then(data => resolve(data));
        })
}

//Option 2
async function getAsyncData3()
{
    let data = await getAsyncData(5)
    console.log("Result : " + data);
    return data;
}

getAsyncData3().then(data => console.log(data))
console.log("End")
```

# Node Js

## NodeJS Project

How to start:
Oepn new folder
Open new terminal
Npm init - - yes
Npm init -y יוצר קובץ פאקאג׳
Node main.js מפעיל את השרת
npm i express cors jsonfile axios mongoose

```
"scripts": {
    "start": "nodemon main"
  },
```

# main.js

```
let my_utils =  require('./utils');
let my_utils2 =  require('./utils2');

console.log(my_utils.getGreet("Avi"))

console.log(my_utils.add(5,6));

let total = my_utils2.add(4,5)



my_utils.getAsyncData().then(data => console.log(data));
/*
async function doSomething()
{
    let data = await my_utils.getAsyncData();
    console.log(data)
}
*/
```

# utils.js

```
exports.getGreet = function (name) {
  return 'Hello ' + name;
};

exports.add = function (num1, num2) {
  return num1 + num2;
};

exports.getAsyncData = function () {
  return new Promise((resolve) => {
    setTimeout(() => resolve('OK'), 1000);
  });
};
```

# utils2.js

```
const getGreet = function(name)
{
    return "Hello " + name;
};

const add = function(num1,num2)
{
    return num1+num2;
};

module.exports = {getGreet, add};
```

# Read JSON Files

## *utils.js*

```
const jFile = require('jsonfile');

//Option 1
jFile.readFile('./person.json', function (err, data) {
  if (err) {
    console.log(err);
  } else {
    console.log(data);
  }
});

//Option 2 - get Promise with "then"
jFile
  .readFile('./person.json')
  .then((data) => console.log(data))
  .catch((err) => console.log(err));

//Option 3 - get Promise with  async-await
async function readFromFile() {
  let data = await jFile.readFile('./person.json');
}
```

## *person.json*

```
{
    "name" : "Ron",
    "age" : 20,
    "address" :
    {
        "street" : "Herzel",
        "city" : "Tel Aviv"
    }
}
```

# Read Json FILE

## *main.js*

```
const utils = require('./utils');

//Option 1
utils.getPersonsFromFile().then(data => console.log(data));

/*
//Option 2
async function getData()
{
    let data = await utils.getPersonsFromFile();
    console.log(data)
}

getData()
*/
```

## *utils.js*

```
const jFile = require('jsonfile');

const file = 'persons.json';

const getPersonsFromFile = async () => {
  // ALWAYS DO - read async way !!!
  let data = await jFile.readFile(file);
  return data.persons;

  // DONT DO !!!
  /*
    let data = jFile.readFileSync("persons.json")
    return data.persons
    */
};

module.exports = { getPersonsFromFile };
```

# Lession 2

## write json file

### *Main.js*

```
const utils = require('./utils.js');

const obj = { name: 'Avi', age: 40 }; מבנה האובייקט שנרצה לכתוב

utils.setPerson(obj).then((data) => console.log(data)); נשתמש בת׳ן
מכיוון שאנחנו מחוץ לפונקציה
במקום באסינק אווייט
```

### *Utils.js*

```
const jf = require('jsonfile'); חבילה זו עושה סטרינגיפי אוטומטית

const file = './data/persons.json'; בתוך דאטה ניגש גייסון

const setPerson = async (obj) => { הפונקציה מקבלת אובייקט פרסון
  const data = await jf.readFile(file); נקרא את הקובץ ונשמור בדאטה
  data.persons.push(obj); נדחוף לדאטה את האובייקט
  await jf.writeFile(file, data); נכתוב מחדש את כל דאטה
  return 'Added Successfully';
};

module.exports = { setPerson };
```

### *Data folder -> Persons.json*

```
{
  "persons": [
    { "name": "Ron", "age": 20 },
    { "name": "Gil", "age": 30 },
    { "name": "Avi", "age": 40 }
  ]
}
```

# rest api

**נלמד כיצד באמצעות רסט נוכל לשלוח בקשות גט - פוט - דליט וכו..**

## *Main.js*

```
const axios = require('axios'); נביא את אקסיוס

const url = 'https://jsonplaceholder.typicode.com/users';

const getAll = async () => {
  const resp = await axios.get(url);אקסיוס נקודה גט קורא את הכל
  console.log(resp.data);
  console.log(resp.status);
};

const getById = async (id) => {
  // // Option 1
  // const resp = await axios.get(url + '/' + id); נשרשר לו סלאש
ואז את האיידי
  // console.log(resp.data);

  // Option 2 - 'Destructuring'
  const { data: user } = await axios.get(url + '/' + id);
  console.log(user); דיסטרקשרינג מכניס ישר את הדאטה מיוזר למשתנה
יוזר
};

const addUser = async (obj) => { נרצה להוסיף משתמש
  const { data } = await axios.post(url, obj);
  console.log(data); הדאטה שנקבל הוא ריספונס על השליחה
};

const updateUser = async (id, obj) => {  עדכון משתמש
  // const { data } = await axios.put(url + '/' + id, obj);
  const { data } = await axios.put(`${url}/${id}`, obj); //
פוט דורס ומחליף ישן בחדש Template Literals
  // const { data } = await axios.patch(`${url}/${id}`, obj); //
Template Literals בק טיקס מקבל ביטוי גאווה מבצע את החישוב עליו
ומחזיר אותו בתור סטרינג
פאטץ׳ לעומת פוט לא דורס הכל אלא מעדכן את מה שאנחנו רוצים !!
  console.log(data);
};
```

```
const deleteUser = async (id) => {
  const { data } = await axios.delete(`${url}/${id}`);
  console.log(data);
};

// getAll();
// getById(2);
// addUser({ name: 'Avi', age: 30 });
// updateUser(5, { name: 'Dana', age: 40 });
deleteUser(1);
```

# rest api module

## *Main.js*

```
const utils = require('./utils');

// Option 1
utils
  .getAllUsers()
  .then((resp) => console.log('1:', resp.data))
  .catch((err) => console.log('2:', err.response.status));
// // Option 2 - 'Destructuring'
// utils.getAllUsers().then(({ data }) => console.log(data));

utils.getUsersEmails().then((emails) =>
console.log(emails)).catch((err) => console.log(err));
```

נקודה ת׳ן
מקבלת פונקציה אנונימית שמקבלת את אימייל ואז מציגה את האימייל בלוג

## *Utils.js*

```
const axios = require('axios');

const url = 'https://jsonplaceholder.typicode.com/userss';

function getAllUsers() {
  return axios.get(url);
}
```

```
const getUsersEmails = async () => { החזרת המיילים בלבד
  const { data: users } = await getAllUsers(); קודם נמשוך את כל
המשתמשים
  const emails = users.map((user) => user.email); לאחר מכן מאפ
נעבור על כולם ונמשוך רק את המיילים
  return emails;
};

module.exports = { getAllUsers, getUsersEmails };
```

# ex4 rest api

Create a program that receives a username as a parameter and gets the full name

for that username from: https://jsonplaceholder.typicode.com/users.

If that name starts with "E", save his tasks' titles from: https://jsonplaceholder.typicode.com/todos to a JSON file.

*main.js*

```
const utils = require('./utils');

const username = 'Antonette';

utils.getUserFullName(username).then((fullName) =>
console.log(fullName));
```

*utils.js*

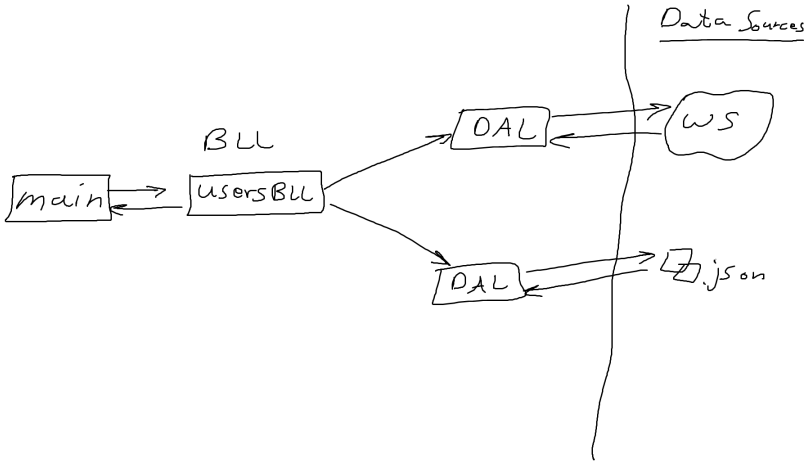
```
const axios = require('axios');
const jf = require('jsonfile');

const usersUrl = 'https://jsonplaceholder.typicode.com/users';
const todosUrl = 'https://jsonplaceholder.typicode.com/todos';

const getUserFullName = async (username) => {
בקשת גט בתלות בשם ונקבל בחזרה את כל המשתמש
על ידי שימוש בפילטרינג שנמצא במדריך של אייפיאיי
הפילטור תמיד מחזיר לי מערך!
  const { data: users } = await axios.get(`${usersUrl}?username=$
{username}`);
  const user = users[0];נמשוך את האיבר הראשון והיחיד במערך



  if (user.name.startsWith('E')) {
    const { data: todos } = await axios.get(`${todosUrl}?userId=$
{user.id}`);
    // const { data: todos } = await axios.get(todosUrl + '?
userId=' + user.id);
    const titles = todos.map((todo) => todo.title);
    await jf.writeFile('./user_titles.json', titles);
  }
סיגון כל הטודו ומשיכת הכותרת בלבד וכתיבת המערך לקובץ כותרות

  return user.name;
};

module.exports = { getUserFullName };
```

# micro services

## BLL- אחראי על כל הלוגיקה, מצד אחד מקושר למיין ומצד שני מקושר לdal ממנו הוא יקבל את המידע הגולמי

# DAL- data access layer

## מתקשר עם הbll ועם המקור מידע ככה שאם מקור המידע משתנה משנים את הדאל ולא את כל הbll




# Main.js

```
const utils = require('./BLL/userBLL');

const id = 2;

utils.getUserData(id).then((data) => console.log(data));
```

**Data folder -> Users.json**

```
[
  { "id": 1, "name": "Avi" },
  { "id": 2, "name": "Dana" }
]
```

# DAL folder -> usersFile.js

## כל תפקידו לקרוא מידע מהקובץ

```
const jf = require('jsonfile');

const file = './data/users.json'; ספציפית בקריאה וכתיבה מקבצי
ג'ייסון הנתיב מתחיל מהתיקייה הראשית!
                הנתיב הוא ביחס למיקום בו אנחנו נמצאים עכשיו  Require
const getUsers = () => {
  return jf.readFile(file);
};

module.exports = { getUsers };
```

## *DAL folder -> usersWS.js*

```
const axios = require('axios');

const url = 'https://jsonplaceholder.typicode.com/users';

const getUser = (id) => {
  return axios.get(`${url}/${id}`);
};

module.exports = { getUser };
```

## *BLL folder -> userBLL.js*

```
const usersFile = require('../DAL/usersFile');
const usersWS = require('../DAL/usersWS');

const getUserData = async (id) => {
  // Data from File
  const users = await usersFile.getUsers();
  const user = users.find((user) => user.id === id);

  // Data from WS
  const { data } = await usersWS.getUser(id);

  return {
    name: user.name,
    email: data.email,
  };
};

module.exports = { getUserData };
```

# Lession 3

## Ex - Micro Services Architecture

Create a “service” that receives a username as a parameter and returns the following data about this user:

- - His name and email from the following REST API: https://jsonplaceholder.typicode.com/users
- - A list of the titles of his first 10 tasks from the following REST API: https://jsonplaceholder.typicode.com/todos
- - A list of his phones from the following JSON file:

```
{
  "users": [
    { "username": "Bret", "phones": ["055-555", "052-222"] },
    { "username": "Antonette", "phones": ["053-333", "054-444"] }
  ]
}
```

- Use BLL & DAL as necessary, and a main file for calling the BLL.

## *main.js*

```
const usersBLL = require('./BLL/usersBLL');

const username = 'Antonette';

usersBLL
  .getUserDataByUsername(username)
  .then((userData) => console.log(userData))
  .catch((error) => console.log(error));
```

# BLL FOLDER

## *usersBLL.js*

```
const usersWS = require('../DAL/usersWS');
const todosWS = require('../DAL/todosWS');
const usersFile = require('../DAL/usersFile');

const getUserDataByUsername = async (username) => {
  const userData = {};

  // Step 1 - user's name & email
  const { data: users } = await
usersWS.getUserByUsername(username);
  const user = users[0];
  userData.name = user.name;
  userData.email = user.email;

  // Step 2 - user's titles of his 10 first todos
  const { data: todos } = await todosWS.getUserTodos(user.id);
  const titles = todos.slice(0, 10).map((todo) => todo.title);
  userData.titles = titles;

  // Step 3 - user's phones
  const data = await usersFile.getAllUsers();
  const { phones } = data.users.find((user) => user.username ===
username);
  userData.phones = phones;

  return userData;
};

module.exports = { getUserDataByUsername };
```

## DAL FOLDER

### *todosWS.js*

```
const axios = require('axios');

const todosUrl = 'https://jsonplaceholder.typicode.com/todos';

const getUserTodos = (userId) => {
  return axios.get(`${todosUrl}?userId=${userId}`);
};

module.exports = { getUserTodos };
```

### *usersFile.js*

```
const jf = require('jsonfile');

const file = './data/users.json';

const getAllUsers = () => {
  return jf.readFile(file);
};

module.exports = { getAllUsers };
```

### *usersWS.js*

```
const axios = require('axios'); הדאל הזה אחראי להביא מהשירות את
המשתמש על ידי שליחת שם המשתמש

const usersUrl = 'https://jsonplaceholder.typicode.com/users';

const getUserByUsername = (username) => {
  return axios.get(`${usersUrl}?username=${username}`);
};

module.exports = { getUserByUsername };
```

## DATA FOLDER

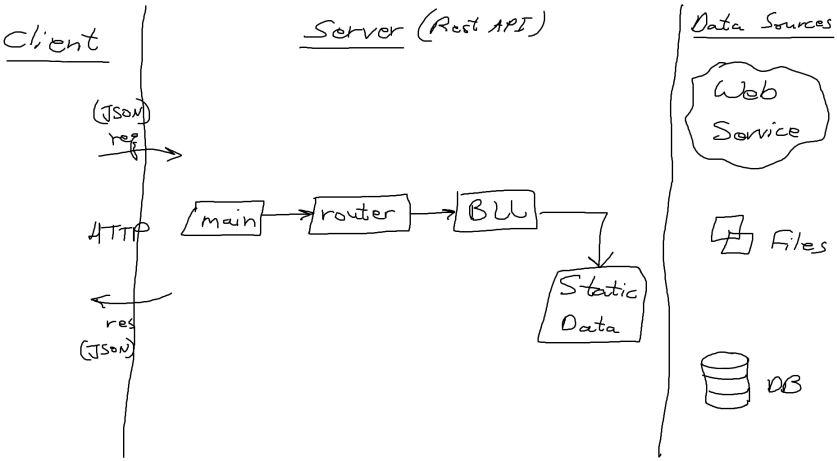
### *users.json*

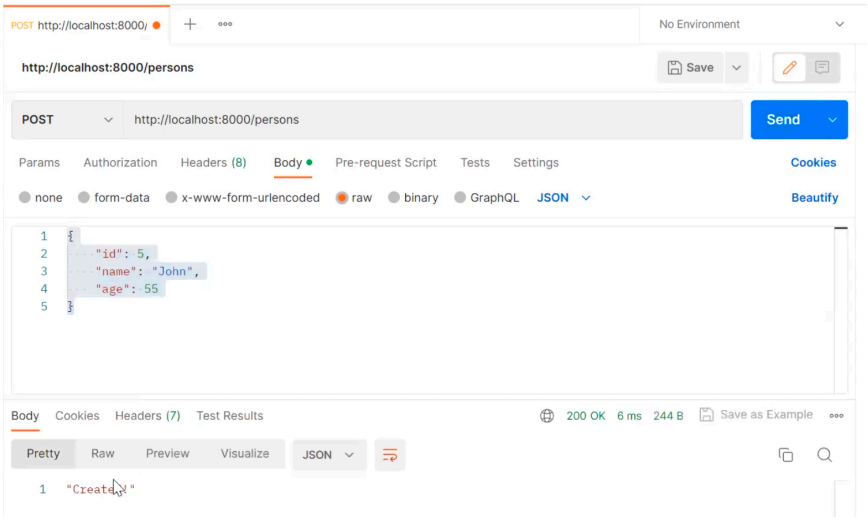
```
{
  "users": [
    { "username": "Bret", "phones": ["055-555", "052-222"] },
    { "username": "Antonette", "phones": ["053-333", "054-444"] }
  ]
}
```

# RESTful web service

בניית שרת rest api משלנו על ידי שימוש בספריית express
שנותנת לנו כלים לבניית ארכיטקטורה זו.
מתפלת בבקשות http וכו

## CLIENT FOLDER



# website.html

```html
<html>
  <body>
    <button onclick="getData()">Get Data</button>

    <script>
      const url = 'http://localhost:8000/persons';

      async function getData() {
        const resp = await fetch(url);
        const data = await resp.json();
        console.log(data);
      }
    </script>
  </body>
</html>
```

## SERVER FOLDER

# Main.js

```
const express = require('express');פריימוורק ליצירת שרת רסט
const cors = require('cors')

const personsRouter = require('./routers/personsRouter');

const app = express();ניצור אינסטנס של אקספרס
const port = 8000;נגדיר משתנה פורט להיות 8000 ואז נאזין לבקשות
מהפורט זה

/* Middlewares */

// Cross-Origin Resource Sharing (CORS) is a mechanism that gives
permission for one origin (domain) to access another origin
app.use(cors())

// Parse incoming request bodies in a middleware before the
handlers, available under the 'req.body' property
app.use(express.json());
כל בקשה שמגיעה לשרת ויש לה באדי תעבור פארסינג לפני שתגיע לראוטר

// routers
app.use('/persons', personsRouter);
הראוטר הזה אומר שאם מגיעה בקשה לשרת ומייד אחרי הסלש יש פרסונס השרת
יפנה את הבקשה לפרסונס ראוטר
*הסלש מציין את האנטרי פוינט
```

```
app.listen( האינסטנס של אקספרס מאזין לפורט שהוגדר וגם מקבל פונקציה שמחזירה הודעה שהשרת בהאזנה
  port,
  () => console.log(`app is listening at http://localhost:${port}`)
// The server's Entry Point
);
```

# *BLL folder -> personsBLL.js*

```
/* Data Storage */
const persons = [
  { id: 1, name: 'Avi', age: 30 },
  { id: 2, name: 'Ron', age: 40 },
  { id: 3, name: 'Dana', age: 50 },
  { id: 4, name: 'Gili', age: 60 },
];
```

```
/* CRUD - Create, Read, Update, Delete */

// GET - Get All - Read
const getAllPersons = () => {
  return persons;
};

// GET - Get By ID - Read
const getPersonById = (id) => {
  const person = persons.find((per) => per.id === +id); הפלוס הופך סטרינג לאינטיגר
  // find() returns 'undefined' if no element is found
  return person ? person : 'Wrong ID‘;
אם לא קיים אדם כזה פיינד יחזיר אנדיפיינד ולכן התנאי יחזיר שהאיידי לא נכון
};

// POST - Create
const addPerson = (obj) => {
  persons.push(obj);
  return 'Created!';
};

// PUT - Update
const updatePerson = (id, obj) => {
  const index = persons.findIndex((per) => per.id === +id);
פיינד אינדקס נועדה להחזיר אינדקס בלבד
  if (index !== -1) { מינוס אחד אומר שהוא לא מצא
    persons[index] = obj;
    return 'Updated!';
  }
  return 'Wrong ID';
};

// DELETE - Delete
```

```
const deletePerson = (id) => {
  const index = persons.findIndex((per) => per.id === +id);
  if (index !== -1) {
    persons.splice(index, 1);
    return 'Deleted!';
  }
  return 'Wrong ID';
};

module.exports = {
  getAllPersons,
  getPersonById,
  addPerson,
  updatePerson,
  deletePerson,
};
```

# *Routers folder -> personsRouter.js*

**הראוטר מזהה את סוג הבקשה , משתמש בפונקציה מהbll ואז מחזיר תשובה על יד רספונס**

```
const express = require('express');
const personsBLL = require('../BLL/personsBLL');

const router = express.Router();נייצר אינסטנט של ראוטר

// 'http://localhost:8000/persons' is the Entry Point

// Get All Persons
router.route('/').get((req, res) => { במידה והבקשה היא גט
  const persons = personsBLL.getAllPersons();
  res.json(persons);נחזיר את המידע על ידי רספונס
});

// Get Person By ID
// the 'id' property is available under the object 'req.params'
router.route('/:id').get((req, res) => { נגדיר שאם אחרי הסלש יש גם
איידי נפנה לראוטר הזה
  const { id } = req.params; נמשוך את האיידי מהריקווסט
  const person = personsBLL.getPersonById(id);
  res.json(person);
});

// Add a Person
router.route('/').post((req, res) => {
  const obj = req.body;בתוך הריקווסט יש באדי שמכיל את הפוסט
אבל הוא מגיע בתור ג׳ייסון מהבקשה של הקליינט ולכן נעשה עליו פארסינג
עוד בקובץ המיין על ידי אקספרס נקודה ג׳ייסון
  const result = personsBLL.addPerson(obj);
  res.json(result);
});
```

```
// Update a Person
router.route('/:id').put((req, res) => {
  const { id } = req.params;
  const obj = req.body;
  const result = personsBLL.updatePerson(id, obj);
  res.json(result);
});

// Delete a Person
router.route('/:id').delete((req, res) => {
  const { id } = req.params;
  const result = personsBLL.deletePerson(id);
  res.json(result);
});

module.exports = router;
```

# Lession 4

## WS with DB

npm i express cors jsonfile axios mongoose

### *main.js*

```
const express = require('express');
const cors = require('cors')
const connectDB = require('./configs/db')

const personsRouter = require('./routers/personsRouter');

const app = express();
const port = 8000;

connectDB()

app.use(cors())
app.use(express.json());

// routers
app.use('/persons', personsRouter);

app.listen(
  port,
  () => console.log(`app is listening at http://localhost:${port}`)
);
```

**BLL FOLDER**

### *personsBLL.js*

```
const Person = require('../models/personModel');

// GET - Get All - Read
const getAllPersons = () => {
  return Person.find({});
```

```
};

// GET - Get By ID - Read
const getPersonById = (id) => {
  return Person.findById({ _id: id });
};

// Post - Create a new person - Create
const addPerson = async (obj) => {
  const per = new Person(obj);
  await per.save();
  return 'Created!';
};

// PUT - Update a person - Update
const updatePerson = async (id, obj) => {
  await Person.findByIdAndUpdate(id, obj);
  return 'Updated!';
};

// DELETE - Delete a person - Delete
const deletePerson = async (id) => {
  await Person.findByIdAndDelete(id);
  return 'Deleted!';
};

module.exports = {
  getAllPersons,
  getPersonById,
  addPerson,
  updatePerson,
  deletePerson,
};
```

## CONFIGS FOLDER

### *db.js*

```
const mongoose = require('mongoose');

const connectDB = () => {
  mongoose
    .connect('mongodb://localhost:27017/personsDB')
    .then(() => console.log('Connected to personsDB!'))
    .catch((error) => console.log(error));
};

module.exports = connectDB;
```

## MODELS FOLDER

### *personModel.js*

```
const mongoose = require('mongoose');

// 'Schema' maps to a MongoDB collection and defines the shape of
the documents within that collection
// 'Schema' is the blueprint of the documents
const personSchema = new mongoose.Schema(
  {
    name: String,
    age: Number,
    // phones: [String],
    // car: {
    //   model: String,
    //   color: String,
    //   year: Number
    // },
    // cars: [{ model: String, color: String, year: Number }]
  },
  { versionKey: false }
);

// A 'model' is a class with which we construct documents in a
collection
const Person = mongoose.model('person', personSchema, 'persons');
// The first argument is the singular name of the collection that
will be created for the model (Mongoose will create the database
collection for the above model 'person').
// The second argument is the schema to use in creating the model.
// The third argument is the name of the collection.

module.exports = Person;
```

## ROUTERS FOLDER

### *personsRouter.js*

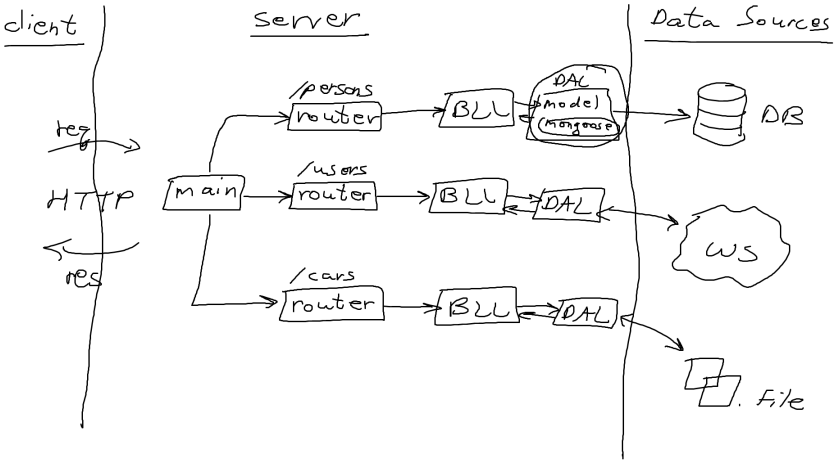
```
const express = require('express');
const personsBLL = require('../BLL/personsBLL');

const router = express.Router();

// 'http://localhost:8000/persons' is the Entry Point

// Get All Persons
router.route('/').get(async (req, res) => {
  try {
    const persons = await personsBLL.getAllPersons();
    res.json(persons); // 200 - OK
  } catch (error) {
    res.json('There was an error!');
  }
});

// Get person by ID
router.route('/:id').get(async (req, res) => {
  const { id } = req.params;
```

```
  const person = await personsBLL.getPersonById(id);
  res.json(person);
});

// Add a new person
router.route('/').post(async (req, res) => {
  const obj = req.body;
  const result = await personsBLL.addPerson(obj);
  res.status(201).json(result);
});

// Update a person
router.route('/:id').put(async (req, res) => {
  try {
    const { id } = req.params;
    const obj = req.body;
    const result = await personsBLL.updatePerson(id, obj);
    res.json(result);
  } catch (error) {
    res.status(500).json('There was an error!');
  }
});

// Delete a person
router.route('/:id').delete(async (req, res) => {
  const { id } = req.params;
  const result = await personsBLL.deletePerson(id);
  res.json(result);
});

module.exports = router;
```

# ws with multiple data sources

מונגוס - מחליף את הדאל, מכיל את הפונקציות לכתיבה קריאה עדכון בדאטה בייס וגם הופך את האובייקט שאנו שולחים לדאטה בייס לרשימה מסוג ג׳ייסון שנכנסת לתוך קולקשיין ולהפך הופכת את הג׳ייסון שנקבל חזרה לאובייקט
משמש גם להגדרת התקשורת הראשונית לדאטה בייס

לא צריך לבנות דקיית דאל אלא להשתמש בפונקציות מהמונגוס בbll




**אך נוסיף תיקיית מודל שבתוך המודל נגדיר מה המבנה של האובייקטים שנעביר לדאטה בייס**

# *Main.js*

```
const express = require('express');
const cors = require('cors')
const connectDB = require('./configs/db') משתמש בחבילת מונגוס

const personsRouter = require('./routers/personsRouter');
const usersRouter = require('./routers/usersRouter');
const carsRouter = require('./routers/carsRouter');

const app = express();
const port = 8000;

connectDB()

app.use(cors())
app.use(express.json());

// routers
app.use('/persons', personsRouter); // DB
app.use('/users', usersRouter); // WS
app.use('/cars', carsRouter); // File
```

```
app.listen(
  port,
  () => console.log(`app is listening at http://localhost:${port}`)
);
```

## BLL FOLDER

### *carsBLL.js*

```
const carsFile = require('../DAL/carsFile');

const getAllCars = async () => {
  const { cars } = await carsFile.getCars();
  return cars;
};

const getCar = async (id) => {
  const cars = await getAllCars();
  return cars.find((car) => car.id === +id);
};

const addCar = async (obj) => {
  const cars = await getAllCars();
  cars.push(obj);
  const data = { cars };
  return carsFile.setCars(data);
};

const updateCar = async (id, obj) => {
  const cars = await getAllCars();
  const index = cars.findIndex((car) => car.id === +id);
  if (index !== -1) {
    cars[index] = obj;
    const data = { cars };
    return carsFile.setCars(data);
  }
};

const deleteCar = async (id, obj) => {
  const cars = await getAllCars();
  const index = cars.findIndex((car) => car.id === +id);
  if (index !== -1) {
    cars.splice(index, 1);
    const data = { cars };
    return carsFile.setCars(data);
  }
};

module.exports = { getAllCars, getCar, addCar, updateCar, deleteCar
};
```

# *personsBLL.js*

קובץ זה לא יעבוד מול דאל אלא מול דאטה בייס באמצעות מונגוס

כל הפונקציות של מונגוס הן אסינכרוניות

```
const Person = require('../models/personModel'); נייבא את המודל של פרסון כדי לעבוד עם פונקציות של מונגוס

const getAllPersons = () => {
  return Person.find({}); פונקציית פיינד מחזירה את כל הרשימות בקולקשיין שביקשנו
}; מי שיממש פונקציה זו יצטרך אסינק אווייט

const getPersonById = (id) => {
  return Person.findById({ _id: id });
};

const addPerson = async (obj) => {
  const per = new Person(obj);
  await per.save();
  return 'Created!';
};

const updatePerson = async (id, obj) => {
  await Person.findByIdAndUpdate(id, obj);
  return 'Updated!';
};

const deletePerson = async (id) => {
  await Person.findByIdAndDelete(id);
  return 'Deleted!';
};

module.exports = {
  getAllPersons,
  getPersonById,
  addPerson,
  updatePerson,
  deletePerson,
};
```

# *usersBLL.js*

```
const usersWS = require('../DAL/usersWS');

const getAllUsers = async () => {
  let { data: users } = await usersWS.getAllUsers();

  users = users.map((user) => {
    return {
      id: user.id,
      username: user.username,
      city: user.address.city,
```

```
    };
  });

  return users;
};

module.exports = { getAllUsers };
```

## CONFIGS FOLDER

### *db.js*

```
const mongoose = require('mongoose'); החבילה שמתקשרת עם הדאטה ביס

const connectDB = () => {
  mongoose
    .connect('mongodb://localhost:27017/personsDB')מתחבר לדאטה ביס
ואם לא קיים יוצר אותו
    .then(() => console.log('Connected to personsDB!')) במידה
והצליח תדפיס הצלחה אם לא תדפיס ארור
    .catch((error) => console.log(error));
};

module.exports = connectDB;
```

## DAL FOLDER

### *carsFile.js*

```
const jf = require('jsonfile');

const file = './data/cars.json';

// read from a JSON file
const getCars = () => {
  return jf.readFile(file);
};

// write to a JSON file
const setCars = async (obj) => {
  await jf.writeFile(file, obj);
  return 'Done!';
};

module.exports = {
  getCars,
  setCars,
};
```

## *usersWS.js*

```
const axios = require('axios');

const url = 'https://jsonplaceholder.typicode.com/users';

const getAllUsers = () => {
  return axios.get(url);
};

module.exports = { getAllUsers };
```

# DATA FOLDER

## *cars.json*

```
{
  "cars": [
    { "id": 1, "model": "Mazda", "color": "Green" },
    { "id": 2, "model": "Subaru", "color": "White" },
    { "id": 3, "model": "Ferrari", "color": "Red" }
  ]
}
```

# MODELS FOLDER

## *personModel.js*

**במודל אנחנו קובעים מהו הסכמה של האובייקט שאיתו נעבוד ולאיזה קולקשיין הוא ישתייך**

```
const mongoose = require('mongoose');

const personSchema = new mongoose.Schema(הגדרת המבנה
  {
    name: String,
    age: Number
  },
  { versionKey: false }
);

const Person = mongoose.model('person', personSchema, 'persons');
הגדרה מול איזה קולקשיין הוא יעבוד
המשתנה הראשון יהיה השם של פריט יחיד בקולקשיין
לדוגמא: פרסון כי הקולקשיין הוא פרסונס
המשתנה השני הוא הסכמה שהגדרנו למעלה
המשתנה השלישי הוא שם הקולקשיין

module.exports = Person;
```

## ROUTERS FOLDER

### *carsRouter.js*

```
const express = require('express');
const carsBLL = require('../BLL/carsBLL');

const router = express.Router();

router.route('/').get(async (req, res) => {
  try {
    const cars = await carsBLL.getAllCars();
    res.json(cars); // 200 - OK
  } catch (error) {
    res.json('There was an error!');
  }
});

router.route('/:id').get(async (req, res) => {
  const { id } = req.params;
  const car = await carsBLL.getCar(id);
  res.json(car);
});

router.route('/').post(async (req, res) => {
  const obj = req.body;
  const result = await carsBLL.addCar(obj);
  res.status(201).json(result);
});

router.route('/:id').put(async (req, res) => {
  try {
    const { id } = req.params;
    const obj = req.body;
    const result = await carsBLL.updateCar(id, obj);
    res.json(result);
  } catch (error) {
    res.status(500).json('There was an error!');
  }
});

router.route('/:id').delete(async (req, res) => {
  const { id } = req.params;
  const result = await carsBLL.deleteCar(id);
  res.json(result);
});

module.exports = router;
```

### *personsRouter.js*

```
const express = require('express');
const personsBLL = require('../BLL/personsBLL');

const router = express.Router();

router.route('/').get(async (req, res) => {
```

```
  try {
    const persons = await personsBLL.getAllPersons();
    res.json(persons); // 200 - OK
  } catch (error) {
    res.json('There was an error!');
  }
});

router.route('/:id').get(async (req, res) => {
  const { id } = req.params;
  const person = await personsBLL.getPersonById(id);
  res.json(person);
});

router.route('/').post(async (req, res) => {
  const obj = req.body;
  const result = await personsBLL.addPerson(obj);
  res.status(201).json(result);
});

router.route('/:id').put(async (req, res) => {
  try {
    const { id } = req.params;
    const obj = req.body;
    const result = await personsBLL.updatePerson(id, obj);
    res.json(result);
  } catch (error) {
    res.status(500).json('There was an error!');
  }
});

router.route('/:id').delete(async (req, res) => {
  const { id } = req.params;
  const result = await personsBLL.deletePerson(id);
  res.json(result);
});

module.exports = router;
```

## *usersRouter.js*

```
const express = require('express');
const usersBLL = require('../BLL/usersBLL');

const router = express.Router();

router.route('/').get(async (req, res) => {
  const users = await usersBLL.getAllUsers();
  res.json(users);
});

module.exports = router;
```

# Lession 5

## query string filtering

שילוב של בקשה מהשרת עם תנאי מסויים
התנאי יגיע אחרי הסימן שאלה.
את השינויים נבצע בקובץ הראוטר על ידי שימוש ב- **req.query**

אחר כך

## BLL FOLDER

### *personsBLL.js*

```
const Person = require('../models/personModel');

// GET - Get All - Read
const getAllPersons = (filters) => { נקבל את הפילטר מהראוטר ונציב
את התנאי בפונקציית הפיינד
  return Person.find(filters);
};

// GET - Get By ID - Read
const getPersonById = (id) => {
  return Person.findById({ _id: id });
};

// Post - Create a new person - Create
const addPerson = async (obj) => {
  const per = new Person(obj);
  await per.save();
  return 'Created!';
};

// PUT - Update a person - Update
const updatePerson = async (id, obj) => {
  await Person.findByIdAndUpdate(id, obj);
  return 'Updated!';
};

// DELETE - Delete a person - Delete
const deletePerson = async (id) => {
  await Person.findByIdAndDelete(id);
  return 'Deleted!';
};

module.exports = {
```

```
  getAllPersons,
  getPersonById,
  addPerson,
  updatePerson,
  deletePerson,
};
```

## MODELS FOLDER

### *personModel.js*

```
const mongoose = require('mongoose');

const personSchema = new mongoose.Schema(
  {
    name: String,
    age: Number,
  },
  { versionKey: false }
);

const Person = mongoose.model('person', personSchema, 'persons');

module.exports = Person;
```

## ROUTERS FOLDER

### *personsRouter.js*

```
const express = require('express');
const personsBLL = require('../BLL/personsBLL');

const router = express.Router();

// 'http://localhost:8000/persons' is the Entry Point

// Get All Persons
router.route('/').get(async (req, res) => {
  try {
    const filters = req.query
    console.log(filters);
    const persons = await personsBLL.getAllPersons(filters);
    res.json(persons); // 200 - OK
  } catch (error) {
    res.json('There was an error!');
  }
});

// Get person by ID
router.route('/:id').get(async (req, res) => {
```

```
  const { id } = req.params;
  const person = await personsBLL.getPersonById(id);
  res.json(person);
});

// Add a new person
router.route('/').post(async (req, res) => {
  const obj = req.body;
  const result = await personsBLL.addPerson(obj);
  res.status(201).json(result);
});

// Update a person
router.route('/:id').put(async (req, res) => {
  try {
    const { id } = req.params;
    const obj = req.body;
    const result = await personsBLL.updatePerson(id, obj);
    res.json(result);
  } catch (error) {
    res.status(500).json('There was an error!');
  }
});

// Delete a person
router.route('/:id').delete(async (req, res) => {
  const { id } = req.params;
  const result = await personsBLL.deletePerson(id);
  res.json(result);
});

module.exports = router;
```

# autentication with JWT

## CLIENT FOLDER

### *login.html*

```
<html>
  <body>
    Username: <input type="text" id="username" /> <br />
    Password: <input type="password" id="password" /> <br />
    <button onclick="login()">Login</button>

    <script>
      const url = 'http://localhost:8000/auth/login';
```

```
      const login = async () => {
        const loginData = {
          username: document.getElementById('username').value,
          password: document.getElementById('password').value,
        };

        const resp = await fetch(url, {
          method: 'post',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(loginData),
        });
        const data = await resp.json();

        sessionStorage['accessToken'] = data.accessToken;
        location.href = './products.html';
      };
    </script>
  </body>
</html>
```

-

## *products.html*

```
<html>
  <body onload="getProducts()">
    <h1>Products Page</h1>

    <script>
      const url = 'http://localhost:8000/products';

      const getProducts = async () => {
        const accessToken = sessionStorage['accessToken'];

        const resp = await fetch(url, {
          method: 'get',
          headers: { 'x-access-token': accessToken },
        });
        const data = await resp.json();
        console.log(data);
      };
    </script>
  </body>
</html>
```

# SERVER FOLDER

## *main.js*

```
const express = require('express');
```

```
const cors = require('cors');

const authRouter = require('./routers/authRouter');
const productsRouter = require('./routers/productsRouter');

const app = express();
const port = 8000;

app.use(cors());
app.use(express.json());

// routers
app.use('/auth', authRouter);
app.use('/products', productsRouter);

app.listen(port, () =>
  console.log(`app is listening at http://localhost:${port}`)
);
```

## SERVER FOLDER -> ROUTERS FOLDER

### *authRouter.js*

```
const express = require('express');
const jwt = require('jsonwebtoken');

const router = express.Router();

// Entry Point: 'http://localhost:8000/auth

router.route('/login').post((req, res) => {
  const { username, password } = req.body;

  // if 'username' and 'password' are exist in DB
  if (true) {
    const userId = 'someId'; // find user's ID
    const ACCESS_SECRET_TOKEN = 'someKey';

    const accessToken = jwt.sign(
      { id: userId },
      ACCESS_SECRET_TOKEN
      // { expiresIn: 7200 } // expires after 7200s (2 hours)
    );
    res.json({ accessToken });
  }

  res.status(401); // Unauthorized
});

module.exports = router;
```

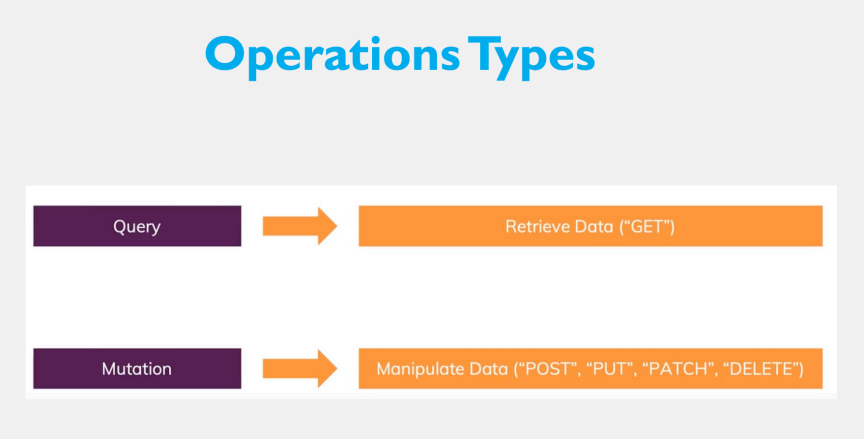
# *productsRouter.js*

```
const express = require('express');
const jwt = require('jsonwebtoken');

const router = express.Router();

// Entry Point: 'http://localhost:8000/products

router.route('/').get((req, res) => {
  const token = req.headers['x-access-token'];

  if (!token) {
    res.status(401).json('No Token Provided!');
  }

  const ACCESS_SECRET_TOKEN = 'someKeyy';

  jwt.verify(token, ACCESS_SECRET_TOKEN, (err, data) => {
    if (err) {
      res.status(500).json('Failed to authenticate token');
    }
    console.log(data);
    // Get data from DB and send to the user
    const products = [{ name: 'car' }, { name: 'phone' }];
    res.json(products);
  });
});

module.exports = router;
```

# Lession 6

## GraphQL with static data

גרף קיו אל הוא תצורה של שרת כמו רסט אייפיאיי רק מבית פייסבוק
בשיטה זו יש פחות מגבלות והקליינט לדוגמא יכול לבקש מהשרת בדיוק איזה שדות הוא מעוניין
לקבל לעומת רסט שבוא נקבל את כל המידע כולל מאטה דאטה מיותר




npm i express-graphql graphql

## CLIENT FOLDER

### *website.html*

```html
<html>
  <body>
    <button onclick="getPerson()">Get Person</button>

    <script>
      const url = 'http://localhost:4000/graphql';

      const query = `
        query ($perId: Int) {
          getPerson(id: $perId) {
            age
            name
          }
        }
      `;

      async function getPerson() {
        const obj = {
          query,
          variables: {
            perId: 3,
          },
        };

        const resp = await fetch(url, {
          method: 'post',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(obj),
        });

        const data = await resp.json();

        // console.log(data);
        console.log(data.data.getPerson);
      }
    </script>
  </body>
</html>
```

# SERVER FOLDER

## *main.js*

```
const express = require('express');
const cors = require('cors');

// mount a GraphQL API server on the '/graphql' HTTP endpoint:
האנדפויינט שלי בעצם הופך להיות הכתובת הרגילה סלש גראף קיו אל
const { graphqlHTTP } = require('express-graphql');
const { buildSchema } = require('graphql');

const personsBLL = require('./BLL/personsBLL');

// construct a schema, using GraphQL schema language
const schema = buildSchema(`
  input PersonInput { נגדיר את האינפוט בנפרד כי לפעמים לא נרצה לתת
איידי בתור פרמטר
    id: Int
    name: String
    age: Int
  }

  type Person {פה נגדיר איך נראה אובייקט מסוג פרסון
    id: Int
    name: String
    age: Int
  }
query נגדיר את הפונקציות בסוג הבקשות
  type Query { קווארי זה קריאת מידע
    allPersons: [Person] מצד שמאל מה הפונקציה מקבלת ומימין מה מחזיר
    getPerson(id: Int): Person מחזיר אובייקט פרסון id מקבל
    getPersonsOlderThan(age: Int): [Person] מחזיר מערך פרסון
  }
נגדיר את הפונקציות בסוג הבקשות Mutation

  type Mutation {מוטייטשן זה כתיבה,עדכון,מחיקת מידע
    addPerson(per: PersonInput): String
    updatePerson(per: PersonInput): String
    deletePerson(id: Int): String
  }
`);

// The 'root' provides a resolver function for each API endpoint
עבור כל אנד פויינט נגדיר לו לאיזה פונקציה הוא הולך בbll
const root = {
  allPersons: personsBLL.getAllPersons,
  getPerson: personsBLL.getPersonById,
  getPersonsOlderThan: personsBLL.getPersonsByAge,
  addPerson: personsBLL.addPerson,
  updatePerson: personsBLL.updatePerson,
  deletePerson: personsBLL.deletePerson,
};

const app = express();
const port = 4000;
```

```
/* Middlewares */
app.use(cors());

app.use( מחליף את ראוטר
  '/graphql',
  graphqlHTTP({
    schema, // schema: schema
    rootValue: root,
    graphiql: true, // GraphQL interface in the browser נותן מין
אינטרפייס לבדיקת בקשות כמו פוסטמן
  })
);

app.listen(port, () =>
  console.log(`app is listening at http://localhost:${port}`)
);
```

## *clientQueries.graphql*

```
# allPersons
query {
  allPersons {
    name
    age
  }
}


# getPerson
query ($perId: Int) {
  getPerson(id: $perId) {
    age
    name
  }
}

QUERY VARIABLES:
{
  "perId": 3
}


# getPersonsOlderThan
query ($age: Int) {
  getPersonsOlderThan(age: $age) {
    age
    name
  }
}

QUERY VARIABLES:
{
  "age": 25
}


# addPerson
```

```
mutation ($per: PersonInput) {
  addPerson(per: $per)
}

QUERY VARIABLES:
{
  "per": {
    "id": 4,
    "name": "Lian",
    "age": 17
  }
}


# updatePerson
mutation ($per: PersonInput) {
  updatePerson(per: $per)
}

QUERY VARIABLES:
{
  "per": {
    "id": 3,
    "name": "Ronny"
  }
}


# deletePerson
mutation ($id: Int) {
  deletePerson(id: $id)
}

QUERY VARIABLES:
{
  "id": 3
}
```

## SERVER FOLDER -> BLL FOLDER

### *personsBLL.js*

```
/* Static Data */
const persons = [
  { id: 1, name: 'Avi', age: 20 },
  { id: 2, name: 'Dana', age: 30 },
  { id: 3, name: 'Ron', age: 40 },
];

const getAllPersons = () => {
  return persons;
};

const getPersonById = (args) => {
  const { id } = args;
  return persons.find((per) => per.id === id);
```

```
};

const getPersonsByAge = (args) => {
  const { age } = args;
  return persons.filter((per) => per.age > age);
};

const addPerson = (args) => {
  const obj = args.per;
  persons.push(obj);
  return 'Created!';
};

const updatePerson = (args) => {
  const obj = args.per;
  const index = persons.findIndex((per) => per.id === obj.id);
  if (index !== -1) {
    // persons[index] = obj;
    persons[index] = { ...persons[index], ...obj };
    return 'Updated!';
  }
  return 'Wrong ID';
};

const deletePerson = (args) => {
  const { id } = args;
  const index = persons.findIndex((per) => per.id === id);
  if (index !== -1) {
    persons.splice(index, 1);
    return 'Deleted!';
  }
  return 'Wrong ID';
};

module.exports = {
  getAllPersons,
  getPersonById,
  getPersonsByAge,
  addPerson,
  updatePerson,
  deletePerson,
};
```