

NodeJS

Express


עבודה על פרויקט מסכם

Users Endpoints

Cards Endpoints

שימוש בMiddleware של isAdmin

```
import { isAdmin } from "../middleware/is-admin";
```



```
router.get("/", isAdmin, async (req, res, next) => {  
  try {  
    const allUsers = await User.find();  
  
    res.json(allUsers);  
  } catch (e) {  
    next(e);  
  }  
});
```

ארכיטקטורה:

ראוטר - ממשק משתמש:
סטטוס, JSON

View

UserService
AuthService

Controller

User/Card

Model

משתמש admin



נרצה למנוע ממשתמש להרשם כadmin

id שייך למשתמש או admin

```
import { RequestHandler, Request } from "express";
import { auth } from "../service/auth-service";
import { User } from "../database/model/user";
import { extractToken } from "../is-admin";
import { BizCardsError } from "../error/biz-cards-error";

const isAdminOrUser: RequestHandler = async (req, res, next) => {
  const { id } = req.params;
  const token = extractToken(req);
  const { email } = auth.verifyJWT(token);

  const user = await User.findOne({ email });

  if (!user) throw new BizCardsError("User does not exist", 401);

  if (id == user.id) return next();

  if (user.isAdmin) return next();

  res.status(401).json({ message: "Only admin/The id must belong to the user" });
};

export { isAdminOrUser };
```

הצגת משתמש לפי ID:

```
router.get("/:id", isAdminOrUser, async (req, res, next) => {  
  try {  
    const { id } = req.params;  
  
    const user = await User.findById(id);  
  
    const { password, ...rest } = user!;  
    return res.json({ user: rest });  
  } catch (e) {  
    next(e);  
  }  
});
```

תיקון של auth-service

```
import jwt from "jsonwebtoken";
import bcrypt from "bcrypt";
import { IJWTPayload } from "../@types/user";
const authService = {
  hashPassword: (plainTextPassword: string, rounds = 12) => {
    return bcrypt.hash(plainTextPassword, rounds);
  },

  validatePassword: (plainTextPassword: string, hash: string) => {
    return bcrypt.compare(plainTextPassword, hash);
  },

  generateJWT: (payload: IJWTPayload) => {
    const secret = process.env.JWT_SECRET!;
    return jwt.sign(payload, secret);
  },

  verifyJWT: (token: string) => {
    const secret = process.env.JWT_SECRET!;

    const payload = jwt.verify(token, secret);

    return payload as IJWTPayload & {iat: number};
  },
};
```

בדיקות:

get Wayne by id with JWT of Willis:

GET http://localhost:8080/api/v1/users/65784e0a927be706a0d28789

Authorization: bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IldpbGxpc0BiYXRjYXZlLmNvbSIsIm1hdCI6MTcwMjgwMDcwNn0.pZDlu6NmQdZhGH1ZDKK5dJAaRY-5S5Dvh3HY6Z4G1Eg

GET Willis by id with jwt of Willis

GET http://localhost:8080/api/v1/users/65784e4a927be706a0d2878f

Authorization: bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IldpbGxpc0BiYXRjYXZlLmNvbSIsIm1hdCI6MTcwMjgwMDcwNn0.pZDlu6NmQdZhGH1ZDKK5dJAaRY-5S5Dvh3HY6Z4G1Eg

GET Wayne with JWT of admin:

GET http://localhost:8080/api/v1/users/65784e0a927be706a0d28789

Authorization: bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFkbWluQGdtYWlsLmNvbSIsIm1hdCI6MTcwMjgwMDg3Nn0.1UBbvM-gXol-pr21_FTWYARyVBYz wfMD9QLs8V8tA4o

הצגת משתמש לפי ID

```
router.get("/:id", isAdminOrUser, async (req, res, next) => {  
  try {  
    const { id } = req.params;  
  
    const user = await User.findOne({"_id": id});  
  
    const { password, ...rest } = user!._doc;  
    return res.json({ user: rest });  
  } catch (e) {  
    next(e);  
  }  
});
```

Middleware - הIDו שיידך למשתמש

```
import { RequestHandler, Request } from "express";
import { auth } from "../service/auth-service";
import { User } from "../database/model/user";
import { extractToken } from "../is-admin";
import { BizCardsError } from "../error/biz-cards-error";

const isUser: RequestHandler = async (req, res, next) => {
  try {
    const { id } = req.params;
    const token = extractToken(req);
    const { email } = auth.verifyJWT(token);
    const user = await User.findOne({ email });

    if (!user) throw new BizCardsError("User does not exist", 401);

    if (id == user.id) return next();

    res
      .status(401)
      .json({ message: "The id must belong to the user" });
  } catch (e) {
    next(e);
  }
};

export { isUser };
```

Middleware - ID שייך למשתמש

המתודה מחזירה לנו Document<IUser>
אובייקט עם מתודות לשמירה/עדכון/מחיקה

User.findById

המתודה lean()
משאירה רק את האובייקט עם המידע בלי מתודות לשמירה/עדכון וכו'.

```
router.get("/:id", isAdminOrUser, async (req, res, next) => {  
  try {  
    const { id } = req.params;  
  
    const user = await User.findById(id).lean() as IUser;  
  
    const { password, ...rest } = user;  
    return res.json({ user: rest });  
  } catch (e) {  
    next(e);  
  }  
});
```

הוספת המשתמש לrequest אחריו חילוץ של הJWT

נוסיף לאובייקט request של Express את התכונה user?

```
import { Request } from "express";  
// add user to express request  
  
import { IUser } from "../user";  
  
declare global {  
  namespace Express {  
    interface Request {  
      user?: IUser;  
    }  
  }  
}
```

עדכון של request

add user prop
↑
req

isAdmin

isAdminOrUser

isUser

req.user

/users

/users/:id

נוסיד id למשתמש:

```
type IUser = {  
  name: IName;  
  address: IAddress;  
  image?: IImage;  
  email: string;  
  phone: string;  
  password: string;  
  isBusiness: boolean;  
  isAdmin?: boolean;  
  createdAt?: Date;  
  _id?: string;  
};
```

isUser

```
const isUser: RequestHandler = async (req, res, next) => {  
  try {  
    const { id } = req.params;  
    const token = extractToken(req);  
    const { email } = auth.verifyJWT(token);  
    const user = (await User.findOne({ email }).lean()) as IUser;  
  
    req.user = user;  
  
    if (!user) throw new BizCardsError("User does not exist", 401);  
  
    if (id == user?._id) return next();  
  
    res.status(401).json({ message: "The id must belong to the user" });  
  } catch (e) {  
    next(e);  
  }  
};
```

routes/users.ts

נותן ערך לreq.user



```
router.put("/:id", isUser, async (req, res, next) => {  
  //let the middleware pass the entire user  
  //hash the password  
  
  console.log(req.user)  
  res.json({ message: "OK" });  
});
```


בדיקה של עדכון פרטי משתמש:

PUT Bruce Wayne with JWT of Wayne:

PUT http://localhost:8080/api/v1/users/65784e0a927be706a0d28789

Authorization: bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bWFpbiCI6IldheW5lQGJhdGNhdmUuY29tIiwiaWF0IjoxNzAyODAzMzE4fQ.t2i7Kuyd4_HbuLqDUT0AWXy62v0zfXd8m5HIGP2nN2E

Content-Type: application/json

```
{
  "name": {
    "first": "Bruce!",
    "last": "Wayne!"
  },
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "country": "Israel",
    "houseNumber": 20,
    "zip": "12345"
  },
  "image": {
    "alt": "user-profile",
    "url": "https://picsum.photos/200/300"
  },
  "phone": "050-8123091",
  "email": "Wayne@batcave.com",
  "isBusiness": true,
  "password": "123456aA!"
}
```

מיני תרגיל:

7.	/users/:id	DELETE	The registered user or admin	Delete user		Delete user
----	------------	--------	------------------------------	-------------	--	-------------

אשמח לשאלות - או פתרון בצ'אט
מוזמנים לשתף מסך בשאלות

מיני תרגיל:

```
router.delete("/:id", isAdminOrUser, async (req, res, next) => {
  try {
    const { id } = req.params;

    const deleteUser = await User.findOneAndDelete({ _id: id });
    return res.json(deleteUser);
  } catch (e) {
    next(e);
  }
});
```

DELETE Bruce Wayne with JWT of Wayne:

```
DELETE http://localhost:8080/api/v1/users/65784e0a927be706a0d28789
```

Authorization: bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IldheW5lQGJhdGNhdmUuY29tIiwiaWF0IjoxNzAyODA3MzE4fQ.t2i7Kuyd4_HbuLqDUT0AWXy62v0zFXd8m5HIGP2nN2E

עבודה עם הספרייה morgan

התקנה:

```
pnpm add morgan
```

```
pnpm add -D @types/morgan
```

עבודה עם הספרייה morgan

שימוש בספרייה:

```
import configDotEnv from "./config";
import express, { json } from "express";
import { logger } from "./middleware/logger";
import { notFound } from "./middleware/not-found";
import { peopleRouter } from "./routes/people";
import { usersRouter } from "./routes/users";
import { connect } from "./database/connection";
import { errorHandler } from "./middleware/error-handler";
import morgan from "morgan";
```

```
configDotEnv();
connect();
```

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

```
//localhost:8080/foo.html
```

```
app.use(express.static("public"));
```

```
// middleware chain:
```

```
app.use(json());
```

```
app.use(morgan("dev"));
```

```
app.use("/api/v1/users", usersRouter); //next(err)
```

```
app.use("/api/v1/people", peopleRouter);
```

```
app.use(errorHandler);
```

```
app.use(notFound);
```

```
app.listen(8080);
```

נשתמש בספרייה במקום logger שהגדרנו קודם

עבודה עם הספרייה morgan

שימוש בספרייה:

```
GET /api/v1/users 200 31.420 ms - 1146
```

```
app.use(  
  morgan(":method :url :status :res[content-length] - :response-time ms")  
);
```

יצירת Logger משלנו:

```
import chalk from "chalk";

class Logger {
  static error(message: string) {
    console.error(chalk.red(message));
  }
  static info(message: string) {
    console.info(chalk.yellow(message));
  }

  static debug(message: string) {
    console.debug(chalk.blue(message));
  }

  static log(message: string) {
    console.log(message);
  }
}

export { Logger };
```

```
Logger.debug("Database Connected");
```

יצירת Custom Logger

מאפשרת לנו להחליט אילו הודעות להשאיר בProduction

```
import chalk from "chalk";

class Logger {
  static error(message: string) {
    console.error(chalk.red(message));
  }

  static info(message: string) {
    if (process.env.NODE_ENV === "prod") return;
    console.info(chalk.yellow(message));
  }

  static debug(message: string) {
    console.debug(chalk.blue(message));
  }

  static verbose(message: string) {
    if (process.env.NODE_ENV === "prod") return;
    console.log(chalk.magenta(message));
  }

  static log(message: string) {
    if (process.env.NODE_ENV === "prod") return;
    console.log(message);
  }
}

export { Logger };
```

Prod
יופיעו רק חלק מההודעות:

debug/error

var args:
מתודה יכולה לקבל יותר מפרמטר אחד - מערך:

```
//Variable Number of Arguments in TypeScript  
//var args js/ts  
function a(...message: string[]){  
  
}
```

var args: מתודה יכולה לקבל יותר מפרמטר אחד - מערך:

```
import chalk from "chalk";

class Logger {
  static error(...messages: any[]) {
    console.error(chalk.red(messages));
  }

  static info(...messages: any[]) {
    if (process.env.NODE_ENV === "prod") return;
    console.error(chalk.yellow(messages));
  }

  static debug(...messages: any[]) {
    console.debug(chalk.blue(messages));
  }

  static verbose(...messages: any[]) {
    if (process.env.NODE_ENV === "prod") return;
    console.log(chalk.magenta(messages));
  }

  static log(...messages: any[]) {
    if (process.env.NODE_ENV === "prod") return;
    console.log(messages);
  }
}

export { Logger };
```

var args: מתודה יכולה לקבל יותר מפרמטר אחד - מערך:

```
import { ErrorHandler } from "express";
import { BizCardsError } from "../error/biz-cards-error";
import { Logger } from "../logs/logger";

const errorHandler: ErrorHandler = (err, req, res, next) => {
  Logger.error(err);
  //userService Error
  if (err instanceof BizCardsError) {
    return res.status(err.status).json({ message: err.message });
  }

  //mongoose error...
  if (err.code && err.code === 11000 && err.keyPattern && err.keyValue) {
    return res.status(400).json({
      message: "Duplicate Key - Must be Unique",
      property: err.keyValue,
      index: err.keyPattern,
    });
  }

  if (err instanceof SyntaxError) {
    return res.status(400).json({ message: "Invalid Json" });
  }

  //catch all
  return res.status(500).json({ message: "Internal Server Error", err });
};

export { errorHandler };
```

שיעורי בית:

נקו את כל הודעות ה-console.log בפרוייקט

במקום console.log השתמשו בLogger

(צבעים + אפשרות להסתרה של הלוג לפי המצב dev/prod)

הספרייה cors

התקנה:

```
pnpm add cors
```

```
pnpm add -D @types/cors
```

צד לקוח חוסם בקשות לשרתים

```
pnpm create vite lec8
```

```
react
```

```
typescript
```

```
cd lec8
```

```
npm start
```

נתיב לבדיקה של CORS

```
import { RequestHandler, Router } from "express";

const router = Router();

// GET /api/v1/people
router.get("/", (req, res) => {
  res.json({ message: "OK" });
});

export { router as peopleRouter };
```

```
###
GET http://localhost:8080/api/v1/people
```

קוד בראקט:

```
import { useEffect } from "react";
import "./App.css";

function App() {
  // use effect -> fetch/axios -> update the state
  useEffect(() => {
    const demo = async () => {
      const res = await fetch("http://localhost:8080/api/v1/people");
      const json = await res.json();

      console.log(json);
    };

    demo();
  }, []);

  return (
    <>
    <h1>Hello, World</h1>
    </>
  );
}

export default App;
```

צד לקוח חוסם את הבקשה בגלל מדיניות CORS

צד שרת:

יאפשר את הכתובת של צד הלקוח:

```
import configDotEnv from "../config";
import express, { json } from "express";
import { notFound } from "../middleware/not-found";
import { peopleRouter } from "../routes/people";
import { usersRouter } from "../routes/users";
import { connect } from "../database/connection";
import { errorHandler } from "../middleware/error-handler";
import morgan from "morgan";
import cors from "cors";
```

```
configDotEnv();
connect();
```

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

```
app.use(
  cors({
    origin: "http://localhost:5173",
  })
);
```

```
//localhost:8080/foo.html
```

```
app.use(express.static("public"));
```

```
// middleware chain:
```

```
app.use(json());
```

```
app.use(morgan("dev"));
```

```
app.use("/api/v1/users", usersRouter); //next(err)
```

```
app.use("/api/v1/people", peopleRouter);
```

```
app.use(errorHandler);
```

```
app.use(notFound);
```

```
app.listen(8080);
```

```
app.use(cors());
```

מילוי ערכים ראשוניים בדטא-בייס:

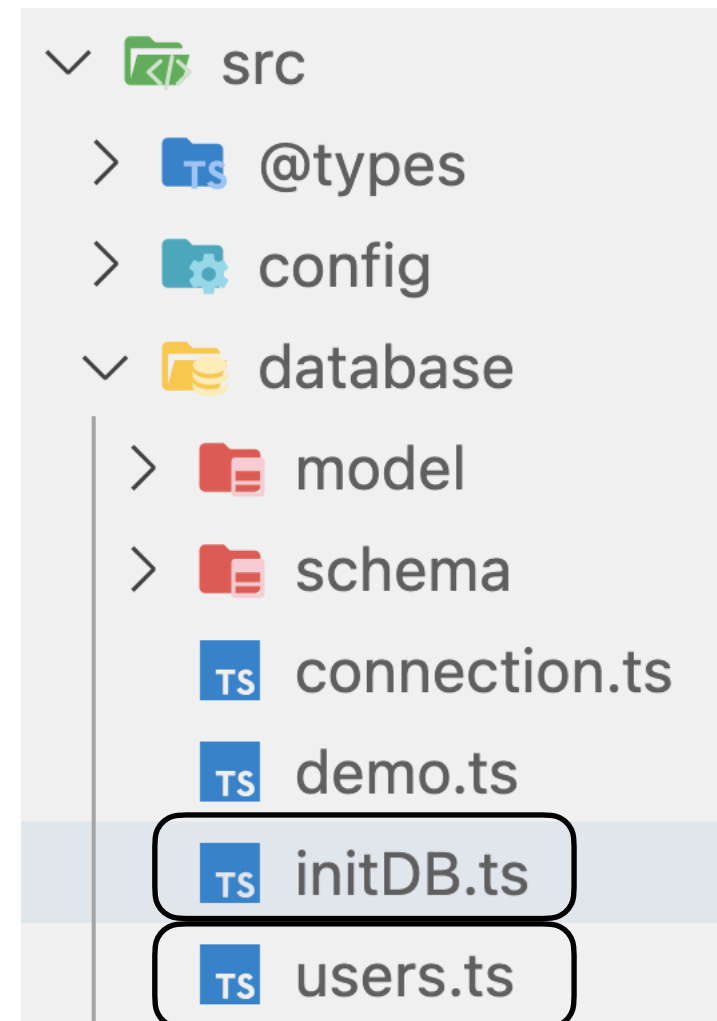
```
import { Logger } from "../logs/logger";
import { User } from "../model/user";
import { users } from "../users";
const initDB = async () => {
  //how many users:

  //add 3 users
  const usersCount = await User.countDocuments();
  if (usersCount !== 0) return;

  for (let user of users) {
    const saved = await new User(user).save();
    Logger.verbose("Added user: ", saved);
  }

  //TODO: add 3 cards
};

export { initDB };
```



```
TS users.ts ×
src > database > TS users.ts > ...
1 export const users = [
2 > { ...
25 } ,
26 > { ...
49 }
50 ]
```

שיעורי בית:

צרו type של Typescript לכרטיסיות:

```
export type ICardInput = {  
  title: string;  
  subtitle: string;  
  description: string;  
  phone: string;  
  email: string;  
  web: string;  
  address: IAddress;  
  image: Image;  
};  
  
export type ICard = ICardInput & {  
  //JWT=> userid  
  //bizNumber => Unique random bizNumber  
  bizNumber?: number;  
  userId?: string;  
  
  _id?: string;  
  likes: string[];  
  createdAt: Date;  
};
```

צרו סכמה של Mongoose לכרטיסיות

צרו מודל של Mongoose לכרטיסיות

שיעורי בית:

צרו סכמה של Joi ליצירת כרטיסיה חדשה:

צרו ראوتر לכרטיסיות:

ממשו את GET /cards

ממשו את POST /card

מי שסיים - מוזמנים להמשיך את הפרוייקט