NodeJS

Node JS API With MongoDB

TomerBu

כרטיס עסק לפי _id

מחזיר כרטיס עסק רק אם כרטיס העסק שייך למשתמש ששלח JWT

req.params.id

```
router.get('/:id', auth, async (req, res) => {
    const card = await Card.findOne({ _id: req.params.id, user_id: req.user._id });
    if (!card) return res.status(404).json({message: 'The card with the given ID was not found.'});
    res.send(card);
});
```

לכל אלמנט שנוסף לדטה-בייס יש _id

קיבלנו את הid_ כשיצרנו את הכרטיס בעמוד הקודם.

```
###get card by id
GET http://localhost:3002/api/cards/627a4f689938329ff1e4ff77
x-auth-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjdhMzAxMmMyNmE3ZGM4NjViNzV
jZTAiLCJiaXoiOnRydWUsImlhdCI6MTY1MjE3NzI0NX0.3zeVTbNbPSIsHSFJy-
X4kEd6oAzvsALxaxe3DW2FhtQ
```

עדכון כרטיס עסק לפי _id

```
router.put('/:id', auth, async (req, res) => {
    const { error } = validateCard(req.body);
    if (error) return res.status(400).json({ message: error.details[0].message });

    //findOneAndUpdate returns a document whereas updateOne does not
    let card = await Card.findOneAndUpdate({ _id: req.params.id, user_id: req.user._id }, req.body);
    if (!card) return res.status(404).json({ message: 'Card not found.' });

    card = await Card.findOne({ _id: req.params.id, user_id: req.user._id });
    res.send(card);
});
```

```
PUT http://localhost:3002/api/cards/627a4f689938329ff1e4ff77 HTTP/1.1
x-auth-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjdhMzAxMmMyNmE3ZGM4NjViNzVjZTAiLCJiaXoiOnRydWUsImlhdC
I6MTY1MjE3NzI0NX0.3zeVTbNbPSIsHSFJy-X4kEd6oAzvsALxaxe3DW2FhtQ
Content-Type: application/json

{
    "bizName":"Biz Name Number 1!!!",
    "bizDescription":"Demo Description, Yeah!",
    "bizAddress":"Biz Address, Iseral Yay!",
    "bizPhone":"03-6135565"
}
```

מחיקת כרטיס עסק לפי _id

```
router.delete('/:id', auth, async (req, res) => {
    const card = await Card.findOneAndRemove({ _id: req.params.id, user_id: req.user._id });
    if (!card) return res.status(404).json({ message: 'Card not found.' });
    res.send(card);
});
```

```
### remove card by id:
DELETE http://localhost:3002/api/cards/627a54172231a9bed6aa57f6 HTTP/1.1
x-auth-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjdhMzAxMmMyNmE3ZGM4NjV
iNzVjZTAiLCJiaXoiOnRydWUsImlhdCI6MTY1MjE3NzI0NX0.3zeVTbNbPSIsHSFJy-
X4kEd6oAzvsALxaxe3DW2FhtQ
```

שמירת כרטיסי עסק עבור משתמש:

models/users.js

```
const Joi = require('joi');
const mongoose = require('mongoose');
const jwt = require('jsonwebtoken');

const userSchema = new mongoose.Schema({
    ...
    biz: {...},
    cards: Array
});

userSchema.methods.generateToken = function () {
    const token = jwt.sign(
        { _id: this._id, biz: this.biz },
        process.env.JWT_SECRET || 'SECRET'
    );
    return token;
};


const userJoishcema = Joi.object({
    name:...
});

function validateUser(user) {
    return userJoishcema.validate(user, { abortEarly: false });
}

const userCardSchema = Joi.object({
    cards: Joi.array().min(1).required()
});

function validateCards(data) {
    return userCardSchema.validate(data);
}

const User = mongoose.model('User', userSchema);

module.exports = {
    User,
    validateUser,
    validateCards
};
```

הוספנו תכונה למשתמש:
מערך של כרטיסי ביקור (Array)
המערך יכיל מחרוזות - מספרי עסק.
השדה הוא לא חובה.

אם נקבל ערך - נרצה לוודא שיש לפחות איבר אחד במערך.

שמירת כרטיסי עסק עבור משתמש:

```
###add cards to a user:
PATCH  http://localhost:3002/api/users/cards
x-auth-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjdhMzAxMmMyNmE3ZGM4NjViNzVjZTAiLCJiaXoi
OnRydWUsImlhdCI6MTY1MjE3NzI0NX0.3zeVTbNbPSIsHSFJy-X4kEd6oAzvsALxaxe3DW2FhtQ
Content-Type: application/json

{
    "cards":["624721"]
}
```

כך נראה משתמש אחרי patch מוצלח

```
  _id: ObjectId('627a3012c26a7dc865b75ce0')
  name: "Tomerbu"
  email: "Tomerbu@gmail.com"
  password: "$2b$12$MzZHAA4kr98cpwxp78teouh/Wu/mNdNs4r7LC.GlsDic1qC8XiVJq"
  biz: true
  createdAt: 2022-05-10T09:27:46.682+00:00
  __v: 1
  cards: Array
    0: "624721"
```

שמירת כרטיסי עסק עבור משתמש:

routes/users.js

```
const usersRouter = require('express').Router();
const { User, validateUser, validateCards } = require('../models/users');
const { Card } = require('../models/cards');
const bcrypt = require('bcrypt');
const _ = require('lodash');
const auth = require('../middleware/auth');

const getCards = async (bizNumbersArray) => {
    const cards = await Card.find({ "bizNumber": { $in: bizNumbersArray } });
    return cards;
};

usersRouter.patch('/cards', auth, async (req, res) => {
    const { error } = validateCards(req.body);
    if (error) res.status(400).json(error.details[0].message);

    const cards = await getCards(req.body.cards);
    if (cards.length != req.body.cards.length)
        res.status(400).json({ message: "Card numbers don't match" });

    let user = await User.findById(req.user._id);
    user.cards = req.body.cards;
    user = await user.save();
    res.json(user);
});
```

1 פעולת עזר:
מוצא כרטיסים שמוכלים בתוך **מערך של מחרוזות**
מערך של מספרי עסק.

2 עדכון משתמש:
מוסיף למשתמש קיים מערך של מחרוזות

בדיקה שיש לפחות איבר אחד

מוצא את כל הכרטיסים לפי מספרי העסק שהתקבלו בגוף הבקשה:

אם מספר הכרטיסים לא תואם את מה שקיבלנו מהשאילתה - הוזנו מספרי כרטיסים שגויים בבקשה.

אם הכל תקין - נמצא את המשתמש ונוסיף לו את התכונה cards

כך נראה משתמש אחרי patch מוצלח

```
_id: ObjectId('627a3012c26a7dc865b75ce0')
name: "Tomerbu"
email: "Tomerbu@gmail.com"
password: "$2b$12$MzZHAA4kr98cpwxp78teouh/Wu/mNdNs4r7LC.GlsDic1qC8XiVJq"
biz: true
createdAt: 2022-05-10T09:27:46.682+00:00
__v: 1
cards: Array
  0: "624721"
```

צפיה במידע של כרטיסי עסק עבור משתמש:

הפעם לשם גיוון משתמשים בדוגמא בquery string:

```
###get cards saved for user:
GET  http://localhost:3002/api/users/cards?numbers=624721
x-auth-token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjdhMzAxMmMyNmE3ZGM4NjViNzVjZTAiLCJiaXoiOnR
ydWUsImlhdCI6MTY1MjE3NzI0NX0.3zeVTbNbPSIsHSFJy-X4kEd6oAzvsALxaxe3DW2FhtQ
Content-Type: application/json
```

routes/users.js

```
const usersRouter = require('express').Router();
const { User, validateUser, validateCards } = require('../models/users');
const { Card } = require('../models/cards');
const bcrypt = require('bcrypt');
const _ = require('lodash');
const auth = require('../middleware/auth');

const getCards = async (cardsArray) => {
    const cards = await Card.find({ "bizNumber": { $in: cardsArray } });
    return cards;
};

usersRouter.get('/cards', auth, async (req, res) => {
    if (!req.query.numbers) res.status(400).send('Missing numbers data');

    let data = {};
    data.cards = req.query.numbers.split(",");
    console.log(data);
    const cards = await getCards(data.cards);
    res.send(cards);
});
usersRouter.patch('/cards', auth, async (req, res) => {
    ...
});
```