

NodeJS

שיעור מסכם והכנה לפרוייקטים

סכמה (מונגוס) לכרטיסיות:

```
import { ICard } from "../../@types/card";
import { Schema } from "mongoose";
import { addressSchema } from "../address-schema";
import { imageSchema } from "../image-schema";

const cardSchema = new Schema<ICard>({
  title: { type: String, required: true },
  subtitle: { type: String, required: true },
  description: { type: String, required: true },
  phone: { type: String, required: true },
  email: { type: String, required: true },
  web: { type: String, required: true },
  address: { type: addressSchema, required: true },
  image: { type: imageSchema, required: true },
  userId: { type: String, required: true },
  bizNumber: {
    type: Number,
    required: false,
    //we will fill this field in the card-service
    default: () => Math.round(Math.random() * 1_000_000),
    unique: true,
  },
  createdAt: {
    type: Date,
    required: false,
    default: new Date(),
  },
  likes: [
    {
      type: String,
    },
  ],
});

export { cardSchema };
```

הוסיפו אורך מקסימלי מינימלי ובדיקות לפי הדרישות במסמך אפיון

מודל (מונגוס) לכרטיסיות:

```
import mongoose from "mongoose";  
import { cardSchema } from "../schema/card-schema";  
  
const Card = mongoose.model("Card", cardSchema);  
  
export { Card };
```

ולידציה עם Joi לכרטיסיות:

```
import Joi from "joi";
import { ICard } from "../@types/card";
import { IAddress, IImage } from "../@types/user";

const schema = Joi.object<ICard>({
  title: Joi.string().min(1).max(100).required(),
  subtitle: Joi.string().min(1).max(100).required(),
  description: Joi.string().min(1).max(500).required(),
  phone: Joi.string().min(1).max(50).required(),
  email: Joi.string().email().min(5).max(255).required(),
  web: Joi.string().uri().min(5).max(255).required(),
  address: Joi.object<IAddress>({
    street: Joi.string().min(1).max(100).required(),
    city: Joi.string().min(1).max(100).required(),
    state: Joi.string().min(1).max(100).required(),
    zip: Joi.string().min(1).max(20).required(),
    country: Joi.string().min(1).max(100).required(),
    houseNumber: Joi.string().min(1).max(20).required(),
  }).required(),
  image: Joi.object<IImage>({
    url: Joi.string().uri().min(5).max(255).required(),
    alt: Joi.string().min(1).max(100).required(),
  }),
});

export default schema;
export { schema as joiCardSchema };
```

Middleware של Joi לכרטיסיות:

middleware
validation
index.ts

```
import { joiCardSchema } from "../../joi/card.joi";
import { joiLoginSchema } from "../../joi/login.joi";
import { joiUserSchema } from "../../joi/user.joi";
import { validateSchema } from "../validate-schema";

export const validateRegistration = validateSchema(joiUserSchema);
export const validateLogin = validateSchema(joiLoginSchema);
export const validateCard = validateSchema(joiCardSchema);
```

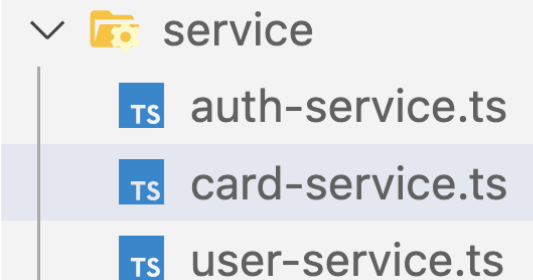
Service עבור כרטיסיות:

```
import { Card } from "../database/model/card";
import { ICardInput } from "../../@types/card.d";
const createCard = async (data: ICardInput, userId: string) => {
  //bizNumber, userId
  const card = new Card(data);

  card.userId = userId;
  //random number that does not exist in the database:
  while (true) {
    const random = Math.floor(Math.random() * 1_000_000);
    const dbRes = await Card.findOne({ bizNumber: random });
    if (!dbRes) {
      card.bizNumber = random;
      break;
    }
  }

  const savedCard = card.save();
  return savedCard;
};

export {createCard}
```



```
service
├── auth-service.ts
├── card-service.ts
└── user-service.ts
```

אם לפי הדרישה bizNumber
הוא מספר שחייב להיות unique
ואנחנו צריכים להגדיר אותו

(לא יעיל - במציאות נעדיף עם UUID)

בדיקה של isBusiness

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

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

    //get user from database
    const user = await User.findOne({ email });

    if (!user) {
      throw new BizCardsError("User does not exist", 401);
    }
    req.user = user;
    const isBusiness = user?.isBusiness;
    if (isBusiness) {
      return next();
    }

    throw new BizCardsError("User Must be a business", 401);
  } catch (e) {
    next(e);
  }
};

export { isBusiness };
```

חייב try/catch
RequestHandler ב

(צריך להוסיף לכל המiddlewares)

אחרת שגיאה שנזרקת תעצור את הפרוייקט.

נעדיף לזרוק שגיאות:

ErrorHandler כי

יכול להוסיף קוד כגון

העלאה של השגיאה/או שליחת אימייל למנהל
בנוסף לסטטוס response ו

הצגת כל הכרטיסיות:

```
router.get("/", async (req, res, next) => {  
  try {  
    //move to service  
    const cards = await Card.find();  
    return res.json(cards);  
  } catch (e) {  
    next(e);  
  }  
});
```

מומלץ להעביר לService
עקרון האחריות הבודדת

```
### Get all cards:  
GET http://localhost:8080/api/v1/cards
```


הכרטיסיות שלי:

```
router.get("/my-cards", validateToken, async (req, res, next) => {  
  try {  
    const userId = req.user?._id!;  
  
    const cards = await Card.find({ userId });  
  
    return res.json(cards);  
  } catch (e) {  
    next(e);  
  }  
});
```

Middleware עבור הכרטיסיות שלי:

```
const validateToken: RequestHandler = async (req, res, next) => {  
  try {  
    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);  
    req.user = user;  
    next();  
  } catch (e) {  
    next(e);  
  }  
};
```

Middleware עבור הכרטיסיות שלי:

```
router.get("/:id", async (req, res, next) => {  
  try {  
    const { id } = req.params;  
  
    const card = await Card.findById(id);  
  
    return res.json(card);  
  } catch (e) {  
    next(e);  
  }  
});
```

GET Card by id:

GET http://localhost:8080/api/v1/cards/658154e8a41b0c0ac526cfc6

הרחבות: ספריה מומלצת לעבודה עם Logs

<https://www.npmjs.com/package/winston>

הספריה דורשת הגדרות ראשוניות.

```
//  
// Logging  
//  
logger.log({  
  level: 'info',  
  message: 'Hello distributed log files!'  
});
```

```
logger.info('Hello again distributed logs');
```

הספריה מאפשרת לשמור לוגים בקובץ (לא רק ל console.log)

חסימת משתמש חסום משתמש שניסה להתחבר שלוש פעמים רצוף
באמצעות אותו המייל אבל עם סיסמה לא נכונה ל – 24 שעות

הגדרה: מה זה רצף?

נשמור את כל הנסיונות השגויים בסכמה לנושא

אם יש 3 נסיונות -> נוסיף שדה חסום: new Date()

ניצור Middleware שיבדוק האם חסום והחסימה בתוקף -> unauthorized

נבדוק האם עברו 24 שעות (לפי התאריך ואם עברו 24 נרוקן את הרשומות של אותו משתמש)

תוכנית להמשך היום:

צהריים עד אחת

יונתן יכנס וידבר על הפרוייקט סיום והבחינה

אפשר מבוא להעלאה של צד שרת לאחסון

אפשר מבוא לNextJs

תקשורת מאובטחת:

ssh-keygen
יוצרת עבורנו 2 קבצים

מפתח ציבורי (משמש לפענוח)

מפתח פרטי (משמש להצפנה)

מפתח ציבורי - מאפשר לפענח את המסרים שמוצפנים עם המפתח הפרטי.

העלאה של הpublic key לשרת - כך יוכל לפענח את המסרים שנשלח לו

Add public SSH key

Copy your public SSH key and paste it in the space below. For instructions on how, follow the steps on the right.

SSH key content

```
XbiAhZE+TLKqrsq1GFcsi05Z8LUymXcmuo0dqXLOunwADFYQ5xl+4xQUvP3Y
7ZrzqGK3K8sdqFT0pHNpjSR2JNsWGVrg5K7hfESDmV6rxDvWR2vYbE//Jugh
OmSMK7uAfj4lWM1gFA08/IPnNYbn3MuEoU4+ZKFYmgVstijcth0tN04+pno+idB
TZW3Sfvyn99EKDasmhc2vEoD1HU7wyPS1LVHuElr/Pxb03teaJBbSbRhBFCKe/
pgFx4M01rRPbfRUomTxjRCcgDJA+/38Y3Uy2O2halkpTBe+IJmU5ksKBMCxHN
Hx3kv23jSqoCPPIxtCSqh0H9rsZOamVst2SZtP7CMArx7QUscIiBs=
tomerbuzaglo@Tomers-iMac-6.local
```

Name
bizkey

Add SSH Key

Enter passphrase (empty for no passphrase):
Enter same passphrase again:

This will generate two files, by default called `id_rsa` and `id_rsa.pub`. Next, add this public key.

Add the public key

Copy and paste the contents of the `.pub` file, typically `id_rsa.pub`, into the **SSH key content** field on the left.

```
cat ~/.ssh/id_rsa.pub
```

Copy

For more detailed guidance, see [How to Add SSH Keys to Droplets](#)

יצירת droplet לוקח זמן



Bizcards Backend

Class project / Educational purposes / Backend for bizcards project

→ Move Resources

Resources

Activity

Settings

DROPLETS (1)

ubuntu-s-1vcpu-1gb-fra1-01

VPS - Virtual Private Server



Bizcards Backend

Class project / Educational purposes / Backend for bizcards project

→ Move Resources

Resources

Activity

Settings

DROPLETS (1)



ubuntu-s-1vcpu-1gb-fra1-01

FRA1 / 1GB / 25GB Disk

46.101.209.94 [Copy](#)

[Add tags](#)



יש לנו מחשב עם כתובת IP

אפשר לתקשר עם המחשב בSSH

46.101.209.94

```
ssh -i dgo root@46.101.209.94
```



שם הקובץ של המפתח הפרטי
שיצרנו עם הפקודה ssh-keygen

בטרמינל שמחובר בSSH לשרת:

התקנה של NodeJS

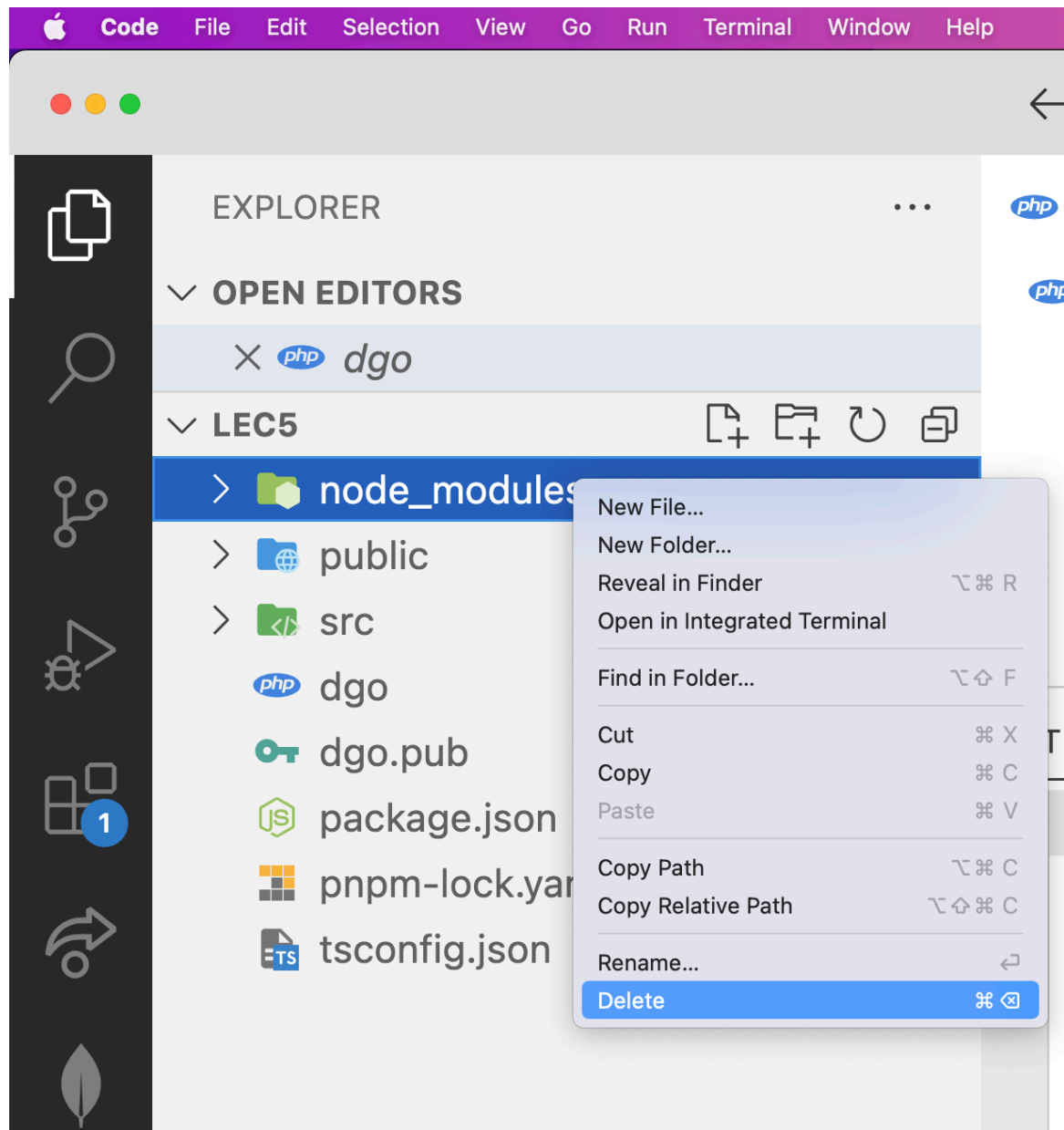
```
apt-get install -y nodejs
```

התקנה של npm

```
apt-get install npm
```

הכנה של הפרויקט להעלאה:

נמחק את node_modules



הכנה של הפרויקט להעלאה:

prod.env ×

src > config > prod.env

```
1 DB_CONNECTION_STRING=mongodb+srv://tomербу:AqnIY4IvmBpfG9u8@cluster0.gz2x7c
2 CLIENT_URL=http://localhost:3000
3 PORT=80
```

הכנה של הפרויקט להעלאה:

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

configDotEnv();
connect();

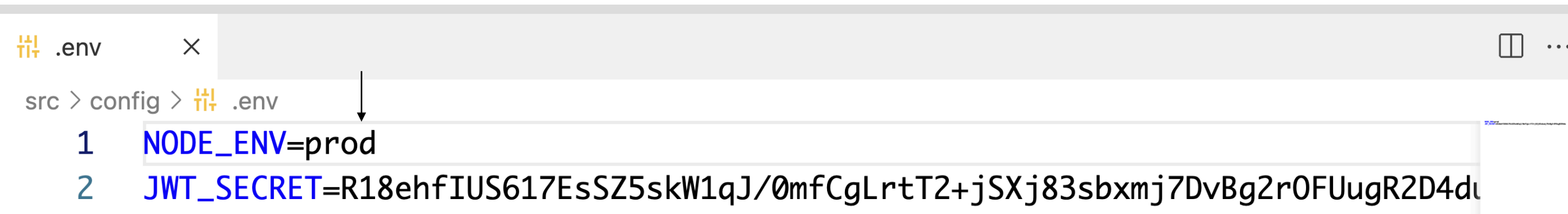
const app = express();
app.use(
  cors({
    // allow my client side
    origin: "http://localhost:5173/",
  })
);
//localhost:8080/foo.html
app.use(express.static("public"));
// middleware chain:
app.use(json());
app.use(morgan("dev"));
app.use("/api/v1/users", usersRouter); //next(err)
app.use("/api/v1/cards", cardsRouter);
app.use(errorHandler);
app.use(notFound);

const PORT = process.env.PORT ?? 8081;

app.listen(process.env.PORT, ()=>{
  // callback when the app is running:
  Logger.info(`App is running: http://localhost:${PORT}`);
});
```

האפליקציה תרוץ על פורט 80

הכנה של הפרויקט להעלאה:



The screenshot shows a code editor with a tab labeled ".env". The breadcrumb navigation indicates the file path is "src > config > .env". The file content consists of two lines: "1 NODE_ENV=prod" and "2 JWT_SECRET=R18ehfIUS617EsSZ5skW1qJ/0mfCgLrtT2+jSXj83sbxmj7DvBg2r0FUugR2D4du". A black arrow points from the ".env" tab to the first line of the file.

```
.env
src > config > .env
1  NODE_ENV=prod
2  JWT_SECRET=R18ehfIUS617EsSZ5skW1qJ/0mfCgLrtT2+jSXj83sbxmj7DvBg2r0FUugR2D4du
```

בדיקה של החיבור לAtlas:

