

# 01 יצירת מסד נתונים

לפני הכל יצרנו מסד נתונים על MySQL - והכנסנו לשם נתונים

```
CREATE DATABASE notes_app;
USE notes_app;

CREATE TABLE notes (
  id integer PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  contents TEXT NOT NULL,
  created TIMESTAMP NOT NULL DEFAULT NOW()
);

INSERT INTO notes (title , contents)
VALUES
('My First Note', 'A note about something'),
('My Second Note', 'A note about something else');
```

## 02 הקמת השרת

על מנת לאתחל את הפרוייקט נריץ

```
npm init -y
```

שמאתחל לנו את הפרוייקט ויוצר קובץ package.json

```
npm i express
```

```
npm i nodemon -D
```

נשנה בקובץ הPACKAGE

```
"main": "server.js",
```

ונעדכן את הסקריפט

```
"scripts": {  
  "start": "node server",  
  "dev": "nodemon server"  
},
```

נוסיף קובץ .gitignore למרות שכרגע אנו לא משייכות את הפרוייקט לגיט אך זה נכון גם אם נעדכן בשלב יותר מאוחר

ונכניס אליה בשלב הראשון שתתעלם מהתקייה של node\_modules

ועכשיו אנו מוכנים להוסיף את קובץ server.js

```
const express = require('express')  
const app = express()  
const PORT = process.env.PORT || 3600  
  
app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

npm run dev

```
PS D:\דוגמיל\node\server\ex1> npm run dev  
  
> ex1@1.0.0 dev  
> nodemon server  
  
[nodemon] 2.0.20  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node server.js`  
Server running on port 3500
```

נאפשר קריאה של קבצים סטטיים

```
const path = require('path')  
app.use('/', express.static(path.join(__dirname, 'public')))
```

ונוסיף את התקייה עם הסטייל הבא

```
EXPLORER
  > node_modules
  > public\css
  # style.css
  .gitignore
  package-lock.json
  package.json
  JS server.js

JS server.js > ...
1  const express = require('express')
2  const app = express()
3  const path = require('path')
4  const PORT = process.env.PORT || 3500
5
6  app.use('/', express.static(path.join(__dirname, 'public')))
7
8  app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

תוכן קובץ הסטייל לא חובה

```
@import
url('https://fonts.googleapis.com/css2?family=Share+Tech+Mono&display=swap');

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

html {
  font-family: 'Share Tech Mono', monospace;
  font-size: 2.5rem;
}

body {
  min-height: 100vh;
  background-color: #000;
  color: whitesmoke;
  display: grid;
  place-content: center;
  padding: 1rem;
}
```

נוסף

```
app.use('/', require('./routes/root'))
```

```

JS server.js > ...
1  const express = require('express')
2  const app = express()
3  const path = require('path')
4  const PORT = process.env.PORT || 3500
5
6  app.use('/', express.static(path.join(__dirname, 'public')))
7  app.use('/', require('./routes/root'))
8  app.listen(PORT, () => console.log(`Server running on port ${PORT}`))

```

ועכשיו נוסיף הפנייה לקובץ INDEX

routes/root.js

```

const express = require('express')
const router = express.Router()
const path = require('path')

router.get('^/$|/index(.html)?', (req, res) => {
  res.sendFile(path.join(__dirname, '..', 'views', 'index.html'))
})

module.exports = router

```

ונוסיף גם את קובץ האינדקס

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>geshtop API</title>
  <link rel="stylesheet" href="css/style.css" />
</head>

<body>
  <h1>geshtop API</h1>

```

```
</body>  
  
</html>
```

ובשלב הזה יצרנו דף ראשוני לממשק ה-API שלנו



התייחסות לקובץ 404

נוסיף את הקובץ לתקיית ה-VIEWS

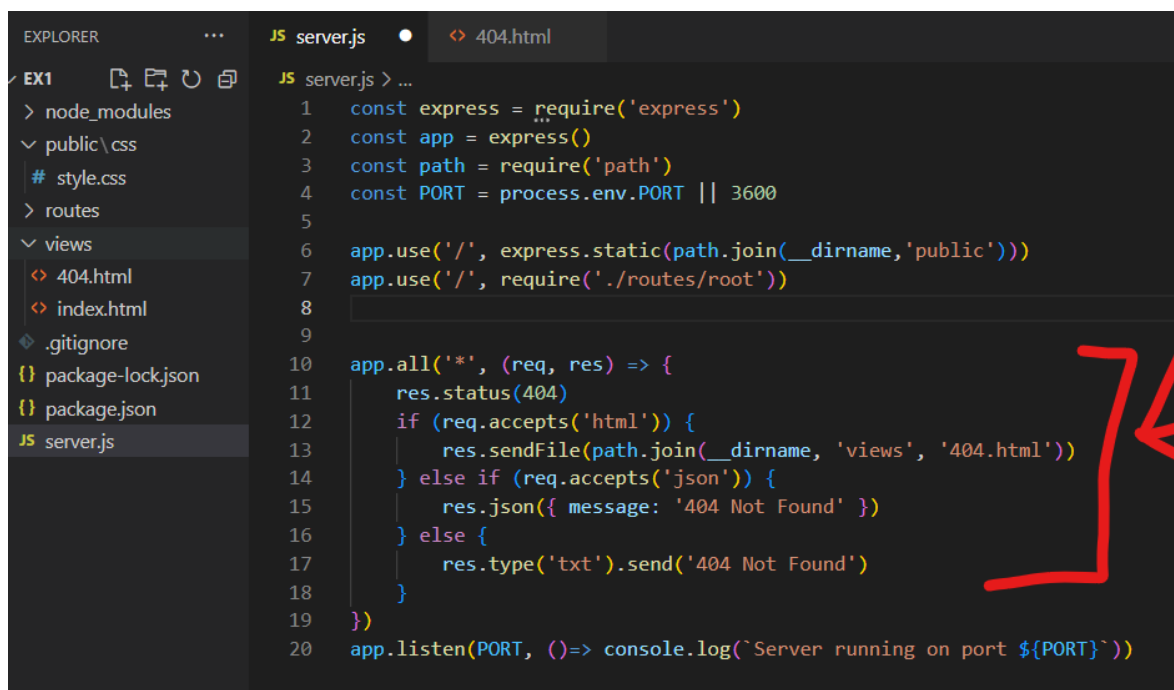
```
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>404 Not Found</title>  
  <link rel="stylesheet" href="css/style.css" />  
</head>  
  
<body>  
  <h1>Sorry!</h1>  
  <p>The resource you have requested does not exist.</p>
```

```
</body>
```

```
</html>
```

ונעדכן את קובץ ה-SERVER אחרי כל הפניות ה-Routes

```
app.all('*', (req, res) => {
  res.status(404)
  if (req.accepts('html')) {
    res.sendFile(path.join(__dirname, 'views', '404.html'))
  } else if (req.accepts('json')) {
    res.json({ message: '404 Not Found' })
  } else {
    res.type('txt').send('404 Not Found')
  }
})
```

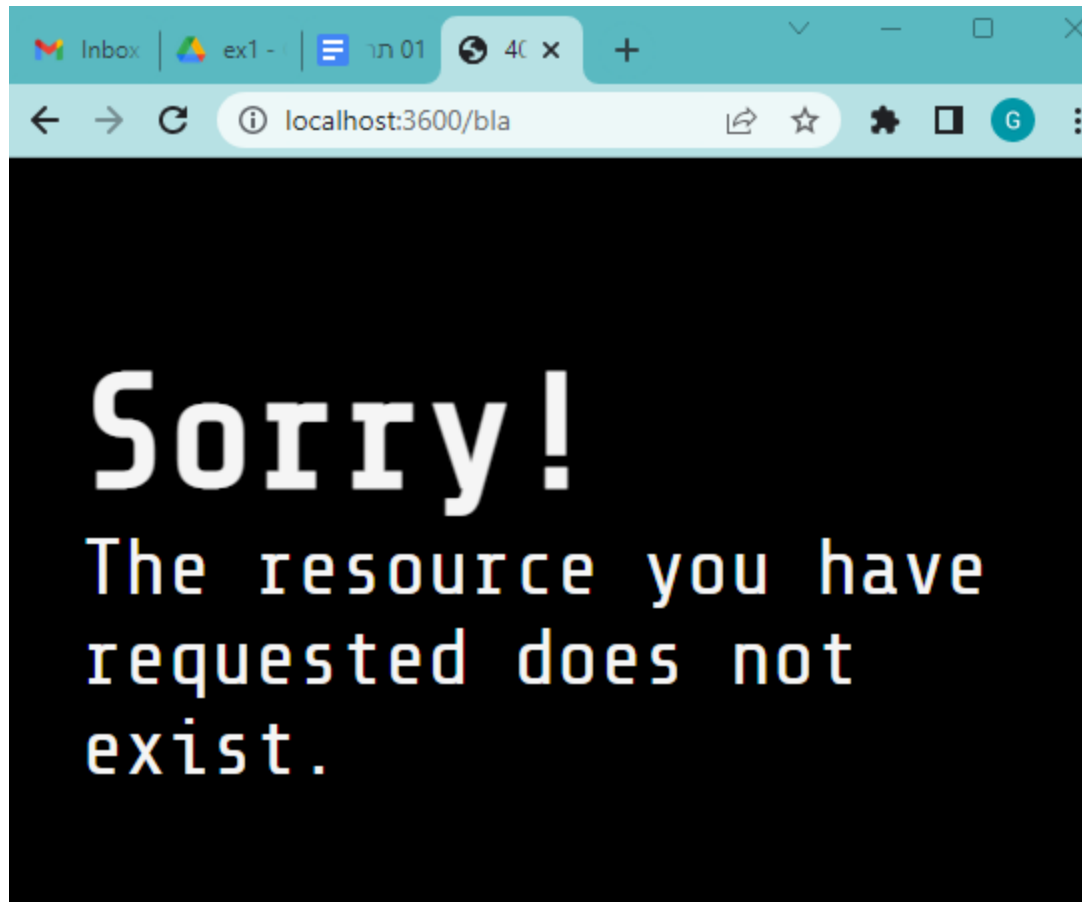


```
EXPLORER
...
JS server.js
404.html

EX1
> node_modules
  public\css
    # style.css
  routes
  views
    404.html
    index.html
  .gitignore
  package-lock.json
  package.json
  JS server.js

JS server.js > ...
1  const express = require('express')
2  const app = express()
3  const path = require('path')
4  const PORT = process.env.PORT || 3600
5
6  app.use('/', express.static(path.join(__dirname, 'public')))
7  app.use('/', require('./routes/root'))
8
9
10 app.all('*', (req, res) => {
11   res.status(404)
12   if (req.accepts('html')) {
13     res.sendFile(path.join(__dirname, 'views', '404.html'))
14   } else if (req.accepts('json')) {
15     res.json({ message: '404 Not Found' })
16   } else {
17     res.type('txt').send('404 Not Found')
18   }
19 })
20 app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

ואז במידה ואין הפנייה קיימת נקבל שגיאת 404



## 03 הגדרות נוספות לפיתוח השרת

נוסיף את האפשרות של שימוש בJSON באמצעות MIDDLEWARE

```
app.use(express.json())
```

```
JS server.js 404.html
JS server.js > ...
1  const express = require('express')
2  const app = express()
3  const path = require('path')
4  const PORT = process.env.PORT || 3600
5  app.use(express.json())
6  app.use('/', express.static(path.join(__dirname, 'public')))
7  app.use('/', require('./routes/root'))
8
9
10 app.all('*', (req, res) => {
11   res.status(404)
12   if (req.accepts('html')) {
13     res.sendFile(path.join(__dirname, 'views', '404.html'))
14   } else if (req.accepts('json')) {
15     res.json({ message: '404 Not Found' })
16   } else {
17     res.type('txt').send('404 Not Found')
18   }
19 })
20 app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

נוסיף אפשרות שהשרת שלנו יוכל לקבל ולקרוא עוגיות

npm i cookie-parser

לאחר ההתקנה נעדכן גם בקובץ ה-SERVER אפשרות שנשתמש בעוגיות בשרת

```
const cookieParser = require('cookie-parser')
...
app.use(cookieParser())
```

וכך הקובץ שלנו נראה כרגע



```

JS server.js > app.all(*) callback
1  const express = require('express')
2  const app = express()
3  const path = require('path')
4  const cookieParser = require('cookie-parser')
5
6  const PORT = process.env.PORT || 3600
7  app.use(express.json())
8  app.use(cookieParser())
9  app.use('/', express.static(path.join(__dirname, 'public')))
10 app.use('/', require('./routes/root'))
11
12
13 app.all('*', (req, res) => {
14   res.status(404)
15   if (req.accepts('html')) {
16     res.sendFile(path.join(__dirname, 'views', '404.html'))
17   } else if (req.accepts('json')) {
18     res.json({ message: '404 Not Found' })
19   } else {
20     res.type('txt').send('404 Not Found')
21   }
22 })
23 app.listen(PORT, () => console.log(`Server running on port ${PORT}`))

```

בגלל שאנו עובדות עם API ולרוב מערכת הקליינט היא אפליקציה נפרדת מה API ולעיתים על דומיין נפרד נצטרך לפתור את בעיית האבטחה של CORS

להבנת הבעיה נפתח את הקונסול של גוגל כרום ונכתוב כך:

`fetch("http://localhost:3600")`

## ומתקבלת השגיאה של CORS

```

> fetch('http://localhost:3600')
< Promise {<pending>}

Access to fetch at 'http://localhost:3600/' from origin 'https://www.google.com' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

GET http://localhost:3600/ net::ERR_FAILED 200

```

ולכן אנו צריכות לאפשר באופן גלובלי בתחילה ולאחר מכן לאבטח דומיינים ספציפיים

`npm i cors`

נעדכן את קובץ ה-SERVER

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

app.use(cors())
```

```
const express = require('express')
const app = express()
const path = require('path')
const cookieParser = require('cookie-parser')
const cors = require('cors')
const PORT = process.env.PORT || 3600

app.use(cors())
app.use(express.json())
app.use(cookieParser())
app.use('/', express.static(path.join(__dirname, 'public')))
app.use('/', require('./routes/root'))

app.all('*', (req, res) => {
  res.status(404)
  if (req.accepts('html')) {
    res.sendFile(path.join(__dirname, 'views', '404.html'))
  } else if (req.accepts('json')) {
    res.json({ message: '404 Not Found' })
  } else {
    res.type('txt').send('404 Not Found')
  }
})
app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

ועכשיו זה תקין

```
> fetch('http://localhost:3600')
< ▶ Promise {<pending>}
>
```

לא נשאיר את זה כך אלא נאפשר להגיע למערכת רק מדומיינים מאושרים  
נוסיף תחת תקיית CONFIG שניצור קובץ בשם **allowedOrigins.js**  
ונעדכן את התוכן שלו

```
const allowedOrigins = [  
  'http://localhost:3000',  
]  
  
module.exports = allowedOrigins
```

נוסיף לתקיית הCONFIG את **corsOptions.js**

```
const allowedOrigins = require('./allowedOrigins')  
  
const corsOptions = {  
  origin: (origin, callback) => {  
    if (allowedOrigins.indexOf(origin) !== -1 || !origin) {  
      callback(null, true)  
    } else {  
      callback(new Error('Not allowed by CORS'))  
    }  
  },  
  credentials: true,  
  optionsSuccessStatus: 200  
}  
  
module.exports = corsOptions
```

נעדכן שימוש בCORS שהגדרנו בקובץ הSERVER

```
const corsOptions = require('./config/corsOptions')  
app.use(cors(corsOptions))
```

וכך נראה כרגע הקובץ שלנו

```
const express = require('express')
const app = express()
const path = require('path')
const cookieParser = require('cookie-parser')
const cors = require('cors')
const corsOptions = require('./config/corsOptions')
const PORT = process.env.PORT || 3600

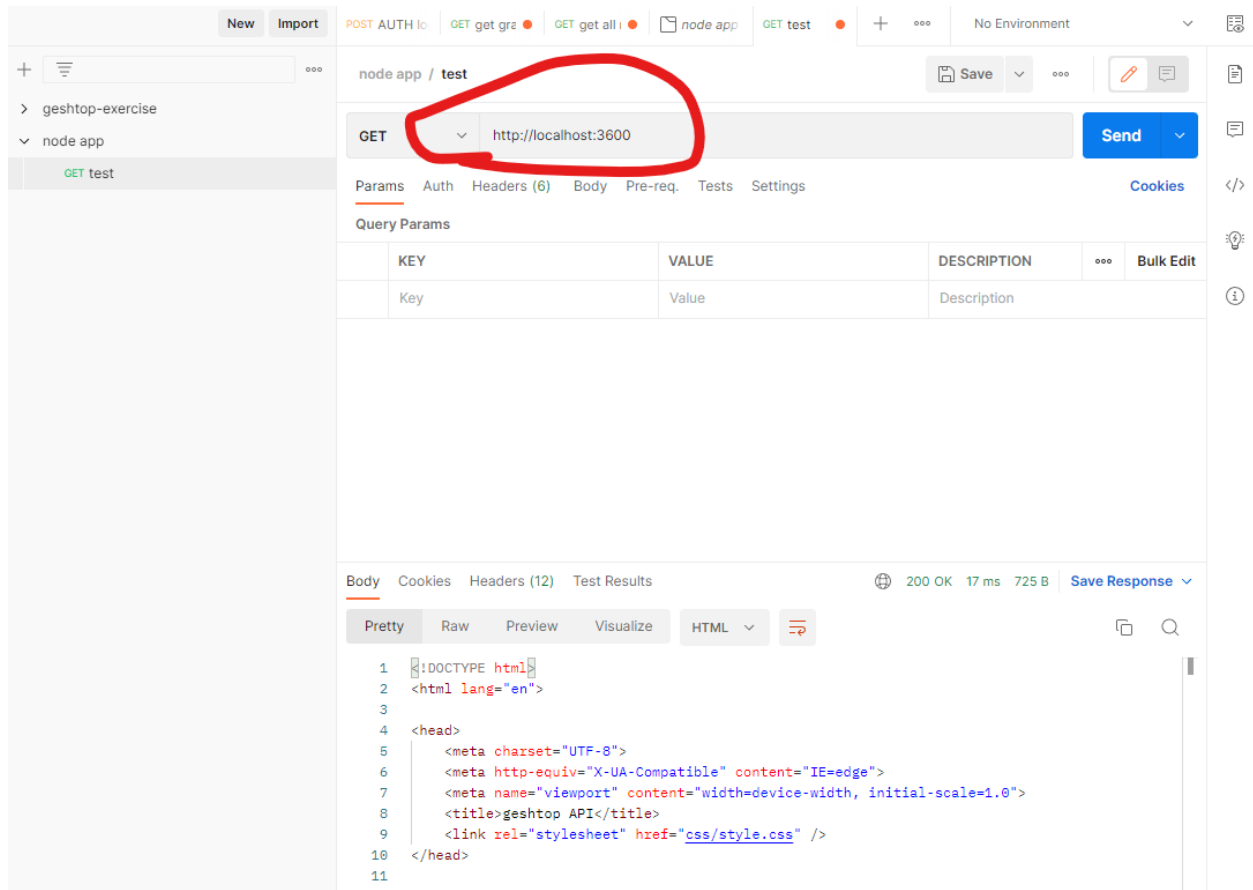
app.use(cors(corsOptions))
app.use(express.json())
app.use(cookieParser())
app.use('/', express.static(path.join(__dirname, 'public')))
app.use('/', require('./routes/root'))

app.all('*', (req, res) => {
  res.status(404)
  if (req.accepts('html')) {
    res.sendFile(path.join(__dirname, 'views', '404.html'))
  } else if (req.accepts('json')) {
    res.json({ message: '404 Not Found' })
  } else {
    res.type('txt').send('404 Not Found')
  }
})

app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

עכשיו שוב בגוגל אנו חסומים

ובPOSTMAN ניתן כרגע לגשת לשרת שלנו




## 04 קינפוג מסד נתונים

נוסיף את dotenv על ידי הרצת הפקודה על מנת להוסיף משתנים לשרת שלנו

`npm i dotenv`

קובץ ה-PACKAGE מכיל כרגע את ה-dotenv עם שאר החבילות שהתקנו בשלבים קודמים

```

{
  "name": "ex1",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
   Debug
  "scripts": {
    "start": "node server",
    "dev": "nodemon server"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2"
  },
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}


```

נוסף

```
require('dotenv').config()
```

בשורה הראשונה של הקובץ על מנת שנוכל לעבוד עם המשתנים בכל הקבצים שנצטרך במערכת

כך נראה הקובץ בינתיים



```
require('dotenv').config()
const express = require('express')
const app = express()
const path = require('path')
const cookieParser = require('cookie-parser')
const cors = require('cors')
const corsOptions = require('./config/corsOptions')
const PORT = process.env.PORT || 3600

app.use(cors(corsOptions))
app.use(express.json())
app.use(cookieParser())
app.use('/', express.static(path.join(__dirname, 'public')))
app.use('/', require('./routes/root'))

app.all('*', (req, res) => {
  res.status(404)
  if (req.accepts('html')) {
    res.sendFile(path.join(__dirname, 'views', '404.html'))
  } else if (req.accepts('json')) {
    res.json({ message: '404 Not Found' })
  } else {
    res.type('txt').send('404 Not Found')
  }
})
app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
```

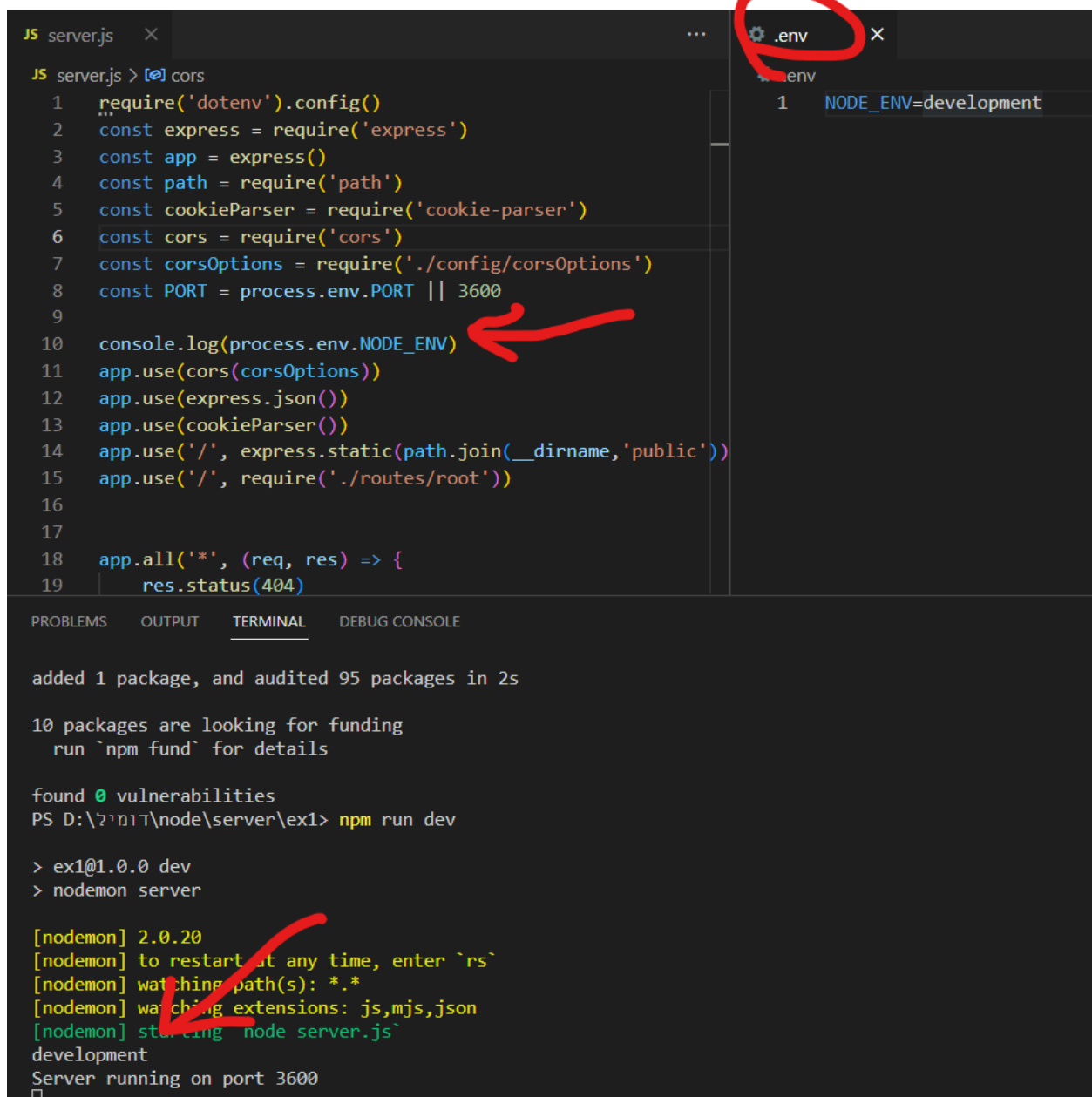
ונוסיף את קובץ ה `env..` בתקיית השורש  
נתחיל עם הגדרה דיפולטית רק לראות איך אנו "מתקשרים" עם הקובץ

```
NODE_ENV=development
```

נוסיף לרגע את הקוד כתיבה לקונסול

```
console.log(process.env.NODE_ENV)
```

ורואים איך נכתב בקונסול



```
JS server.js > [?] cors
1  require('dotenv').config()
2  const express = require('express')
3  const app = express()
4  const path = require('path')
5  const cookieParser = require('cookie-parser')
6  const cors = require('cors')
7  const corsOptions = require('./config/corsOptions')
8  const PORT = process.env.PORT || 3600
9
10 console.log(process.env.NODE_ENV)
11 app.use(cors(corsOptions))
12 app.use(express.json())
13 app.use(cookieParser())
14 app.use('/', express.static(path.join(__dirname, 'public')))
15 app.use('/', require('./routes/root'))
16
17
18 app.all('*', (req, res) => {
19   res.status(404)
```

```
.env
1  NODE_ENV=development
```

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE

```
added 1 package, and audited 95 packages in 2s

10 packages are looking for funding
  run `npm fund` for details

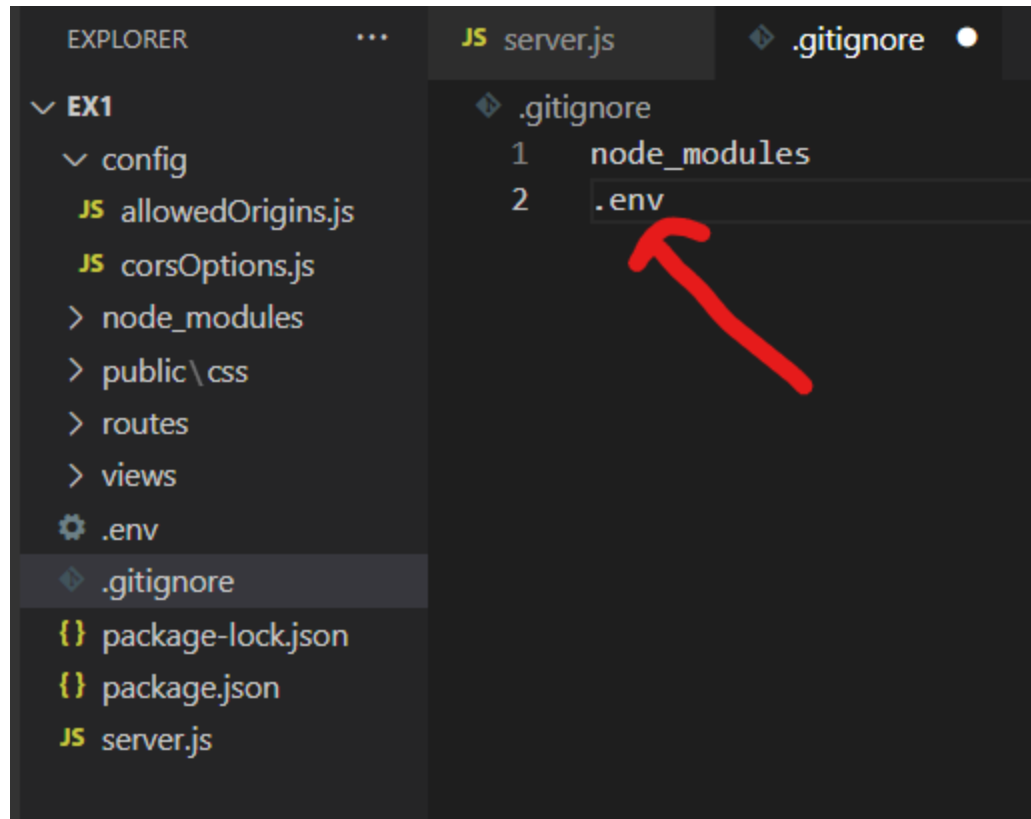
found 0 vulnerabilities
PS D:\דומייל\node\server\ex1> npm run dev

> ex1@1.0.0 dev
> nodemon server

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting node server.js
development
Server running on port 3600
```

וכמובן של הכמובן נחסום את הקובץ מלעלות לגיט על ידי שנעדן את קובץ ה.GITIGNORE





## 05 בחירת ORM sequelize לעבודה עם מסד

לפרטים

<https://sequelize.org/>

נתקין

```
npm install sequelize  
npm install mysql2  
//npm install -D sequelize-cli
```

```

6   "scripts": {
7     "start": "node server",
8     "dev": "nodemon server"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "cookie-parser": "^1.4.6",
15    "cors": "^2.8.5",
16    "dotenv": "^16.0.3",
17    "express": "^4.18.2",
18    "mysql2": "^2.3.3",
19    "sequelize": "^6.28.0"
20  },
21  "devDependencies": {
22    "nodemon": "^2.0.20",
23    "sequelize-cli": "^6.5.2"
24  }
25 }

```



תחת תקיית CONFIG נוסף את הקובץ dbConfig.js  
הקוד הרלוונטי

```

module.exports = {
  HOST: process.env.DATABASE_HOST,
  USER: process.env.DATABASE_USER,
  PASSWORD: process.env.DATABASE_PASSWORD,
  DB: process.env.DATABASE_DB,
  dialect: 'mysql',
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
  }
}

```

```

    idle: 10000
  }
}

```

ונוסיף בהתאמה את הנתונים בקובץ ENV

```

NODE_ENV=development
DATABASE_HOST=localhost
DATABASE_USER=root
DATABASE_PASSWORD=*****
DATABASE_DB=notes_app

```

תחת תקיית MODEL ניצור קובץ index.js

```

const dbConfig = require('../config/dbConfig');
const {Sequelize, DataTypes} = require('sequelize');

const sequelize = new Sequelize(
  dbConfig.DB,
  dbConfig.USER,
  dbConfig.PASSWORD, {
    host: dbConfig.HOST,
    dialect: dbConfig.dialect,
    operatorsAliases: false,

    pool: {
      max: dbConfig.pool.max,
      min: dbConfig.pool.min,
      acquire: dbConfig.pool.acquire,
      idle: dbConfig.pool.idle
    }
  }
)

sequelize.authenticate().then(() => {
  console.log('Connection has been established successfully.');
```

```

}).catch((error) => {
  console.error('Unable to connect to the database: ', error);
});

```

```

const db = {}

db.Sequelize = Sequelize
db.sequelize = sequelize

db.notes = require('./note')(sequelize, DataTypes)
db.sequelize.sync({ force: false })
.then(() => {
  console.log('yes re-sync done!')
})
module.exports = db

```

נוסיף את הקובץ note.js תחת תקיית models  
 הסוגים השונים של DATATYPE

<https://sequelize.org/docs/v6/core-concepts/model-basics/#data-types>

```

module.exports = (sequelize, DataTypes) => {
  const Note = sequelize.define(
    "note",
    {
      title: {
        type: DataTypes.STRING,
        allowNull: false,
      },
      contents: {
        type: DataTypes.TEXT,
      },
      created: {
        type: DataTypes.DATE,
      },
    },
    {
      timestamps: false,
    }
  );

```

```
    return Note;
};
```

## 06 יצירת ROUTES & CONTROLLERS

ניצור קובץ noteController.js תחת תקיית controllers שניצור

```
const db = require('../models/index')
const Note = db.notes

// @desc Get all notes
// @route GET /notes
// @access Private
const getAllNotes = async (req, res) => {
  // Get all notes from DB
  const notes = await Note.findAll({})

  // If no notes
  if (!notes?.length) {
    return res.status(400).json({ message: 'No notes found' })
  }

  res.json(notes)
}

// @desc Get all notes
// @route GET /notes
// @access Private
const getOneNote = async (req, res) => {
  const id = req.params.id
  const note = await Note.findOne({where:{id:id}})
  res.json(note)
}

// @desc Create new note
// @route POST /notes
// @access Private
const createNewNote = async (req, res) => {
```

```

    const { title, contents } = req.body
    // Confirm data
    if (!title) {
        return res.status(400).json({ message: 'All fields are required'
    })
    }
    const note = await Note.create({ title, contents })

    if (note) { // Created
        return res.status(201).json({ message: 'New note created' })
    } else {
        return res.status(400).json({ message: 'Invalid note data
received' })
    }
}

// @desc Update a note
// @route PATCH /notes
// @access Private
const updateNote = async (req, res) => {
    const { id, title, contents } = req.body

    // Confirm data
    if (!id || !title) {
        return res.status(400).json({ message: 'All fields are required'
    })
    }
    const note = await Note.update({title,contents},{where:{id:id}})

    if (!note) {
        return res.status(400).json({ message: 'note not found' })
    }

    res.json(note)
}

// @desc Delete a note
// @route DELETE /notes

```

```

// @access Private
const deleteNote = async (req, res) => {
  const { id } = req.body

  // Confirm data
  if (!id) {
    return res.status(400).json({ message: 'note ID required' })
  }

  await Note.destroy({
    where: {
      id: id
    }
  });
  res.json( `Note with ID ${id} deleted` )
}

module.exports = {
  getAllNotes,
  createNewNote,
  getOneNote,
  updateNote,
  deleteNote
}

```

ניצור קובץ noteRoutes.js תחת תקיית routes שיצורנו

```

const express = require('express')
const router = express.Router()
const noteController = require('../controllers/noteController')

router.route('/')
  .get(noteController.getAllNotes)
  .post(noteController.createNewNote)
  .patch(noteController.updateNote)
  .delete(noteController.deleteNote)
router.get('/:id', noteController.getOneNote)
module.exports = router

```

<https://sequelize.org/docs/v6/core-concepts/model-querying-basics/>

נוסיף הפנייה לROUTE שיצרנו בקובץ הSERVER

```
app.use("/api/notes", require("../routes/noteRoutes"));
```

והקובץ המלא של הSERVER



```

const express = require('express')
const app = express()
const path = require('path')
const cookieParser = require('cookie-parser')
const cors = require('cors')
const corsOptions = require('./config/corsOptions')
const PORT = process.env.PORT || 3600

//middleware
app.use(cors(corsOptions))
app.use(express.json())
app.use(cookieParser())
//routes
app.use('/', express.static(path.join(__dirname, 'public')))
app.use('/', require('./routes/root'))

app.use("/api/notes", require("./routes/noteRoutes"));

app.all('*', (req, res) => {
  res.status(404)
  if (req.accepts('html')) {
    res.sendFile(path.join(__dirname, 'views', '404.html'))
  } else if (req.accepts('json')) {
    res.json({ message: '404 Not Found' })
  } else {
    res.type('txt').send('404 Not Found')
  }
})
app.listen(PORT, ()=> console.log(`Server running on port ${PORT}`))

```



הללויה - אפשר להריץ גם מהPOSTMAN

node app / get all notes

Save

GEThttp://localhost:3600/api/notes

Send

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (9)Test Results

200 OK169 ms846 BSave Response

PrettyRawPreviewVisualizeJSON

```
1  {
2    {
3      "id": 1,
4      "title": "My First Note",
5      "contents": "A note about something",
6      "created": "2023-01-09T17:17:34.000Z"
7    },
8    {
9      "id": 2,
10     "title": "My Second Note",
11     "contents": "A note about something else",
12     "created": "2023-01-09T17:17:34.000Z"
13   },
14   {
15     "id": 3,
16     "title": "test updated",
17     "contents": "test2",
18     "created": "2023-01-10T00:37:23.000Z"
19   }
20 }
```