Node.Js

Authentication & Authorization

01/01/2025

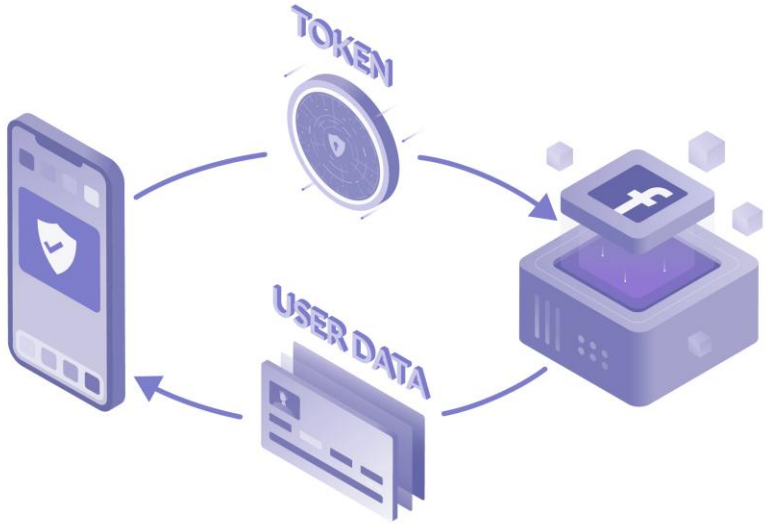
רמות הרשאה וניהול גישה למסד הנתונים

OAuth2

בחלק זה נכיר את פרוטוקול **OAuth 2.0**, נבין כיצד הוא מאפשר הרשאה מאובטחת ונלמד על המרכיבים המרכזיים שלו.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה זה "**OAuth 2.0**"?
- באילו תרחישים יהיה לנו כדאי להשתמש בפרוטוקול זה?
- מה ההבדל בין "**OAuth 2.0**" לאימות רגיל עם סיסמה?
- מה היתרון המרכזי של שימוש ב-"**OAuth 2.0**" עבור אפליקציות צד שלישי?
- מהו "**Access Token**" וכיצד משתמשים בו?
- מדוע מומלץ להגדיר תוקף ל-**Access Tokens**?
- מהם הסיכונים האפשריים בשימוש לא נכון ב-**OAuth 2.0**?

OAuth2



JWT

בחלק זה נלמד על הספרייה **jsonwebtoken** וכיצד היא משמשת לאימות והעברת נתונים מאובטחים בין צדדים שונים. נבין את מבנה ה-**JWT**, נכיר את יתרונותיו ונבחן דוגמאות לשימוש. בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה זה "**JWT**"?
- ממה הוא מורכב?
- מה מכיל כל אחד מהחלקים?
- מה היתרון של שימוש ב-**JWT** לאימות?
- היכן נרצה לשים את הסיסמה שיש להוסיף לחתימת ה-"**token**"?
- לשם מה נשתמש במתודה "**sign**" ואיזה פרמטרים ניתן להעביר לה?
- לשם מה נשתמש במתודה "**verify**" ואיזה פרמטרים ניתן להעביר לה?

JWT – תרגול – חלק א'

Welcome/biz_cards_dev.rar

```
import jwt from "jsonwebtoken";
import { SECRET_KEY } from "./env.service.js";

const generateToken = (user) => {
    const { _id, authLevel } = user;
    const payloadData = { _id, authLevel };
    const token = jwt.sign(payloadData, SECRET_KEY, { expiresIn: "1d" });
    return token;
};


const verifyToken = (tokenFromClient) => {
    try {
        const userData = jwt.verify(tokenFromClient, SECRET_KEY);
        return userData;
    } catch (error) {
        return null;
    }
};

export { generateToken, verifyToken };
```

```
import { verifyToken } from "../services/auth.service.js";

export const auth = async (req, res, next) => {
    try {
        const tokenFromClient = req.header("x-auth-token");
        if (!tokenFromClient)
            throw new Error("Authentication Error: Please Login");
        const userInfo = verifyToken(tokenFromClient);
        if (!userInfo) throw new Error("Authentication Error: Unauthorize user");
        req.user = userInfo;
        return next();
    } catch (error) {
        return res.status(401).json({ message: error.message });
    }
};
```

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	צרו **service** גלובלי חדש אשר מכיל פונקציות בשם "**generateToken**" ו-"**verifyToken**" עבור יצירת **token** ווידוא תקינות **token** בהתאם.
Ex-2	שנו את הלוגיקה של התחברות משתמש חדש כך שבמידה וההתחברות תקינה, הנתיב יחזיר **token** חדש אשר מורכב מפרטי המשתמש הרלוונטיים.
Ex-3	שנו את ה-**middleware** שקראתם לו "**auth**" כך שישתמש בפונקציה "**verifyToken**" שיצרתם עבור וידוא תקינות **token** במקום הפונקציונליות הקודמת שהיתה ל-**middleware** זה.
Ex-4	בידקו את תיפקוד ה-**API** בהתאם לשינויים שביצענו. (תרצו לבצע **login** ולאחר מכן להשתמש ב-**token** ב-**header** של בקשה לנתיב מוגן).

Welcome/biz_cards_dev.rar

JWT – תרגול – חלק ב'

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	צרו בתיקיית ה-**middlewares** פונקציה חדשה אשר מקבלת רמת הרשאה מינימלית כפרמטר ומחזירה **middleware** אשר בודק את רמת ההרשאה של המשתמש ומחזיר שגיאה אם היא יותר נמוכה מרמת ההרשאה שקיבלנו כפרמטר.
Ex-2	השתמשו ב-**middleware** שיצרתם בנתיב שמחזיר את כל המשתמשים כך שרק משתמש מסוג אדמין יוכל לגשת אליו.
Ex-3	צרו **middleware** חדש אשר בודק אם המשתמש הוא אדמין או שה-**id** שהתקבל ב-**params** תואם ל-**id** אשר מופיע ב-**token** שהתקבל בבקשה, אחרת תוחזר שגיאה (כלומר שזהו וידוא שאכן מדובר באותו המשתמש או שמדובר באדמין שמנסה לגשת שזה גם תקין) .
Ex-4	השתמשו ב-**middleware** שיצרתם בנתיב שמחזיר את פרטי המשתמש לפי **id** ובנתיב למחיקת משתמש לפי **id** כך שרק אותו המשתמש או האדמין יוכלו לגשת אליו.

```
export const checkAuthLevel = (minAuthLevel) => async (req, res, next) => {
    try {
        if (req.user.authLevel < minAuthLevel) {
            throw new Error("Authentication Error: Unauthorized user");
        }
        return next();
    } catch (error) {
        return res.status(403).json({ message: error.message });
    }
}
```

```
export const isUser = (req, res, next) => {
    if (req.user.authLevel < 3 && req.user._id !== req.params.id) {
        return res.status(403).json(
            { message: "Authentication Error: Unauthorized user" }
        );
    }
    return next();
}
```


מחבר: גל לביא

תודה על ההקשבה

אני וצוות המכללה כאן עבורכם