NODE.JS FULLSTACK FEJLESZTÉS

3. forduló

A kategória támogatója: Emarsys

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

3.forduló - HTTP

Felhasznált idő: 10:05/15:00 **Elért pontszám:** 6/12

1. feladat 0/2 pont

Mely **függvénynevek** kerülnek a számmal jelölt helyekre?

```
const http = require('node:http');
const server = http.__1__();

server.on('request', (request, response) => {
  response.__2__(200, { 'Content-Type': 'application/json' });
  response.__3__(JSON.stringify({
    data: 'Hello World!'
  }));
});

server.__4__(8080);
```

Válasz

- [ ] 1. listen
  2. setHeader
  3. end
  4. createServer

- [x] 1. createServer
  2. addHeader
  3. send
  4. listen

  Ez a válasz helytelen, de megjelölted.

- [ ] 1. createServer
  2. writeHead
  3. end
  4. listen

  Ez a válasz helyes, de nem jelölted meg.

- [ ] 1. createServer
  2. send
  3. addHeader
  4. listen

- [ ] 1. createServer
  2. setHeader
  3. end
  4. listen

Magyarázat

https://nodejs.org/api/http.html

2. feladat 0/4 pont

Mely állítások igazak a **http.ServerResponse.end()** metódusra?

Válaszok

- [x] Egészen addig, amíg nem hívjuk meg, addig a válasz szabadon módosítható.
  Ez a válasz helyes, és meg is jelölted.

- [ ] Amennyiben a **.send()** metódust meghívtuk, úgy már nincs szükség a hívására.

- [x] Nem kötelező ebben megadni az adatot, amivel vissza szeretnénk küldeni a választ.
  Ez a válasz helyes, és meg is jelölted.

- [ ] Csak annak hívásakor tudjuk meghatározni a visszaküldendő adatot.

- [x] Deprecated, használjuk helyette a **.send()**-et.
  Ez a válasz helytelen, de megjelölted.

Magyarázat

1. válasz: Igaz. Az .end() meghívása előtt még lehet módosítani a választ, viszont utána már nem.
2. válasz: Hamis. A .send() metódus nem létezik nodejs-ben. Az ismert frameworkok közül pl. az express-ben van használva.
3. válasz: Igaz. Lehetőség van a write() metódusban is megadni.
4. válasz: Hamis. Lehetőség van a write() metódusban is megadni.

5. válasz: Hamis. A .send() továbbra sem létezik.

3. feladat 6/6 pont

Mely állítások **igazak** a következő kódrészletre?

```
const jwt = require('jsonwebtoken');
const express = require('express');

const USER_DB = {
    'user1': 'Secret1',
    'user2': 'Secret2'
};

function authenticate(req, res, next) {
    const authHeader = req.headers['authorization'];
    const token = authHeader && authHeader.split(' ')[1];

    if (token === null) {
        return res.sendStatus(401);
    }

    const authData = jwt.decode(token);

    if (!(authData.user in USER_DB)) {
        return res.sendStatus(401);
    }

    req.user = authData;
    next();
}

const app = express();

app.get('/endpoint/:user', authenticate, function(req, res) {
    const user = req.params.user;
    res.send(`Hi ${user}! Your secret is ${USER_DB[user]}`);
});

app.listen(5000);
```

Válaszok

- [x] Hiányzik az autentikáció, mert nem ellenőrizzük a JWT token hitelességét.
  Ez a válasz helyes, és meg is jelölted.
- [x] Minden bejelentkezett felhasználó el tudja érni az összes felhasználó adatait.
  Ez a válasz helyes, és meg is jelölted.
- [ ] Az alapértelmezett token aláírási algoritmus nem biztonságos, ezert kriptográfiai támadással egyszerűen lehetséges a felhasznalók megszemélyesítése.
- [ ] Hibás tokennel is be lehet jelentkezni, mert a decode aszinkron függvény, és a hibák nem propagálódnak.

Magyarázat

Hiányzik az autentikáció, mert nem ellenőrizzük a JWT token hitelességét: IGAZ. Jwt decode helyett verify-t kell használni

Minden bejelentkezett felhasználó el tudja érni az összes felhasználó adatait: IGAZ. Azon felül, hogy nem hitelesítünk, nem az ellenőrzött (tehat megbízhatatlan) user ID alapján adjuk vissza az adatot.

Az alapértelmezett token aláírási algoritmus nem biztonságos, ezert kriptográfiai támadással egyszerűen lehetséges a felhasznalók megszemélyesítése: HAMIS. A jelenlegi alapértelmezett algoritmus a HS256 (HMAC with SHA-256), ami biztonsagos.

Hibás tokennel is be lehet jelentkezni, mert a decode aszinkron függvény, és a hibák nem propagálódnak: HAMIS. A decode függvény szinkron.