Programrendszerek fejlesztése gyakorlat

(IMN109L-2)

html_mastery

(Beadandó feladat dokumentációja)

Feladatkiírás

Angular+NodeJS+MongoDB stackkel kell egy komplex webalkalmazást összehozni. Alapvetően 2-3 fős csapatoban. 2 fő esetén a webalkalmazás designjára, reszponzivitására nem kell annyi időt szánni, 3 fő esetén elvárás, hogy a böngészős frontend is színvonalas legyen. Amennyiben valaki egyszemélyes seregként szeretne dolgozni a projekten, akkor a frontend design szintén nem elvárás és extra segítségként ekkor lehetőség van a stack egy elemét kicserélni, tehát pl. Springet használni NodeJS helyett vagy Reactet Angular helyett.

Html_mastery leírása

A backend különböző HTML feladatok leírását tárolja, minden feladathoz tartozzon egy leírás, egy stringként tárolt forráskód részlet és egy másik forráskód részlet, amely a sikeres megoldást ellenőrzi. A szerver kezelje felhasználók regisztrációját is, minden felhasználó rendelkezzen egy “mastery” szinttel, amely jelzi a felhasználó által sikeresen leküzdött feladatok számát.

A frontenden a felhasználó bejelentkezés után tudjon indítani feladatokat, kihívásokat, amelyek tartalmát a frontend a szerverről kéri le, és a usernek egy megadott idő lejárta előtt kell végrehajtania. A megoldófelület egy kétpaneles elosztást követ, az egyik oldalon HTML forráskód legyen szerkeszthető, a másik oldalon pedig ez a forráskód legyen az Angular ngSanitize modulja segítségével a komponens részeként megjelenítve. A megoldó felületen legyen egy tesztelési gomb is, amely meghív egy funkciót és leellenőrzi, hogy sikeres megoldásra került-e a feladat. (Pl. hozz létre egy gombot egy div tagen belül úgy, hogy a div id-ja “protector” legyen, a gombé pedig “greetButton”)

Csappattagok

1. Név: Daniel János Róbert
   Neptun: HSHAOO
   EHA: DAJYAAT.SZE
   h-s azonosító: h650011
   Vállalt feladatok: Fullstack

Megjegyzések a feladathoz

Egyszemélyes csapatként indultam. Nem cserélek le stack-elemet.

Adatbázis

MongoDB, melyet mongoose segítségével kezelek. Az adatbázis a következő kollekciókat tartalmazza (sémaleírás server/src/models):

Tasks: A HTML feladatokat tárolja. Megtalálható benne a feladatok címe, leírása, a kiindulási forráskód, a feladat megoldásának kódja, illetve egy időkorlát, ami alatt a feladatot megoldó felhasználó meg kell oldja a feladatot (1. Ábra).

Users: A felhasználók kollekciója. Megtalálható benne a felhasználónév, a jelszó (titkosítva), a felhasználó mastery-szintje (minden helyes megoldásért a mastery-szint értéke egyel nő), illetve a felhasználó által már megoldott feladatok címeit tartalmazó tömb (1. Ábra).

```
const taskSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
    unique: true
  },
  description: {
    type: String,
    required: true
  },
  base_source_code: String,
  solution: {
    type: String,
    required: true,
  },
  max_duration: Number
})
```

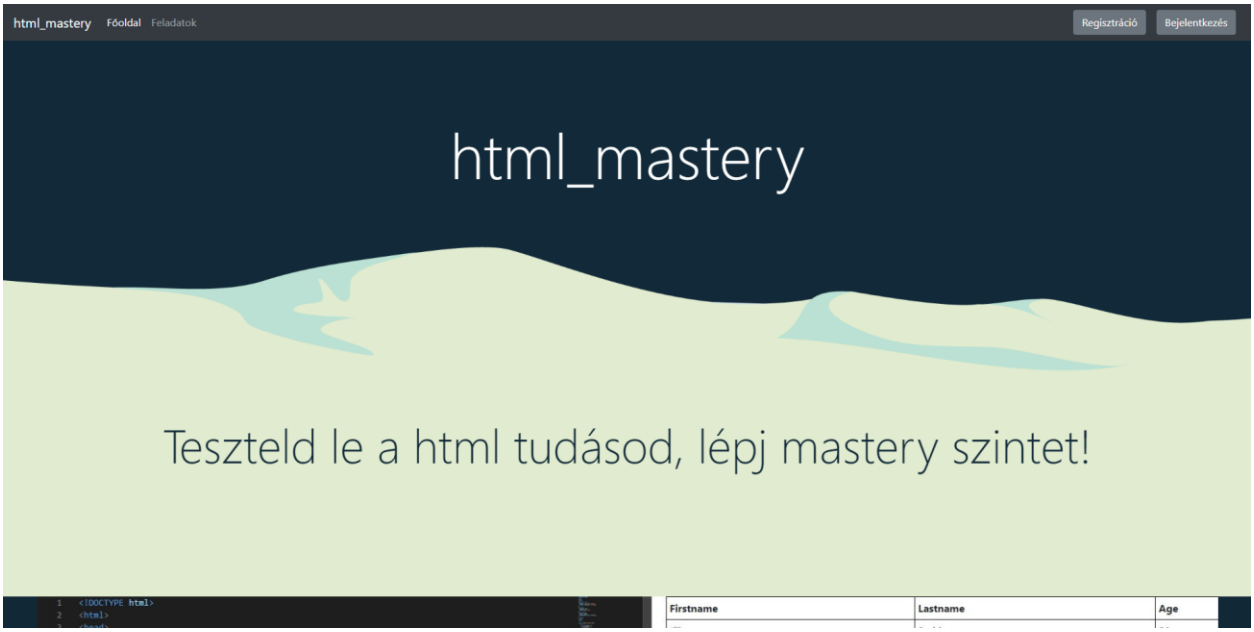
```
const userSchema  = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true,
  },
  password: String,
  mastery: Number,
  resolved_tasks: [String]
})
```

1. Ábra: Task és a User kollekciók sémája

Frontend

A frontend megvalósításához AngularJs keretrendszert alkalmaztam, ennek 9-es verzióját. A megjelenítéshez és az oldal design-jához az ngBootstrap-et, egy bootstrap alapú keretrendszert használtam, amelyet kifejezetten Angular-hoz fejlesztettek. A rendszer asztali felhasználásra lett tervezve.

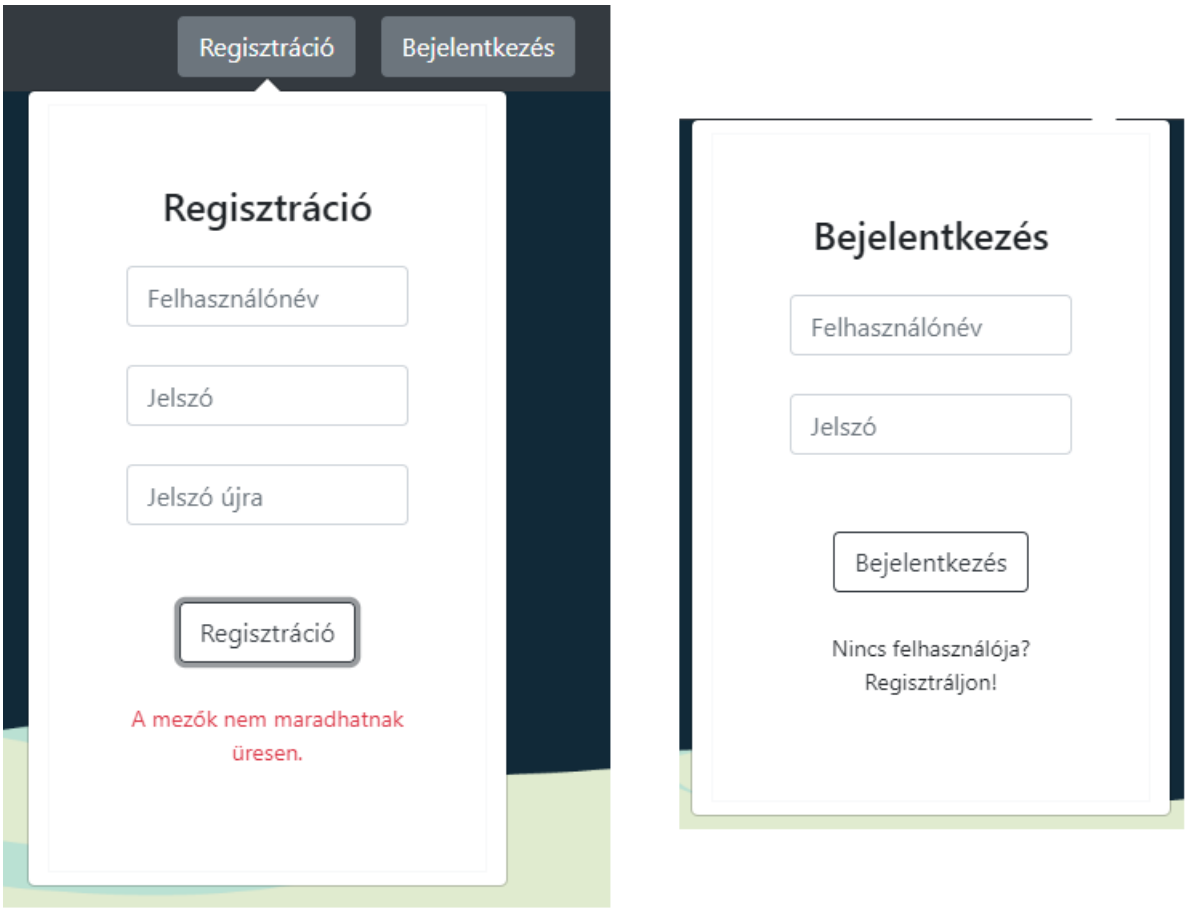
A html_mastery egy egyszerű, néhány oldalból álló SPA. Letisztult, minimalista design-t követtem, ennek köszönhetően az oldal használata szép és nem túlzsúfolt, használata nagyon egyszerű. A főoldal funkcionalitást nem tartalmaz. Ha a felhasználó olyan oldalt látogatna meg, melyhez nincs jogosultsága, akkor ide lesz irányítva. (2. Ábra)



2. Ábra: A főoldal

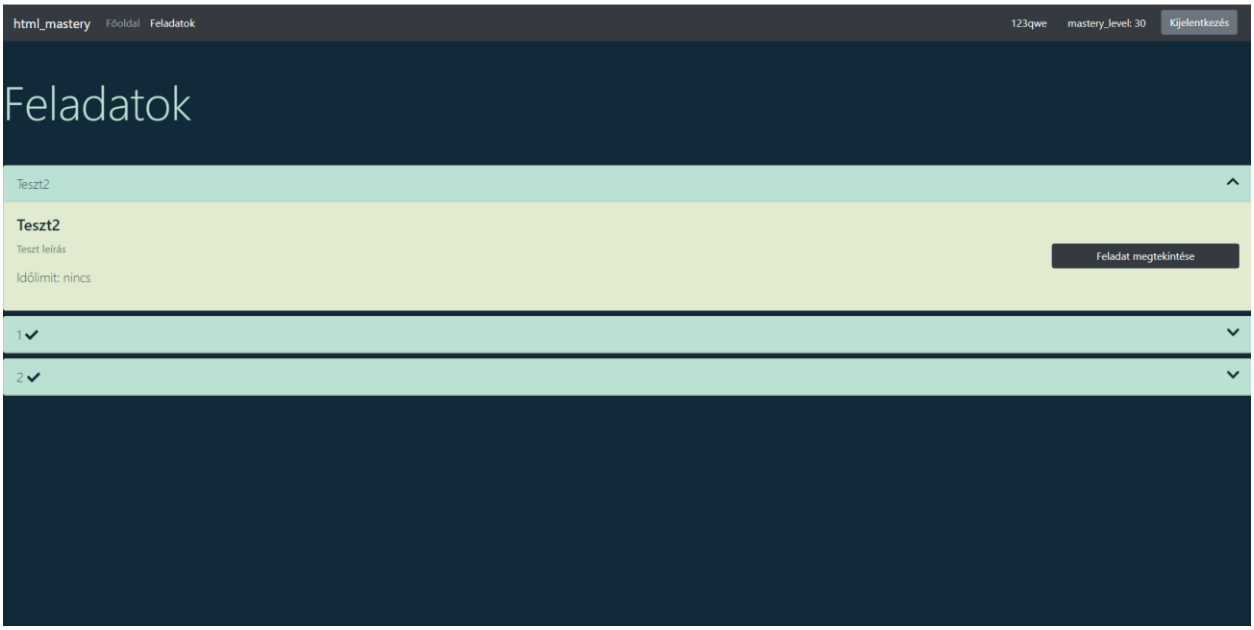
A főoldalon viszont már látható a menüsor, melynek köszönhetően navigálhatunk az oldalon belül. A menüsor egy igen komplex komponense a weboldalnak, ez később fog látszódni.

A főoldalon található a regisztráció, illetve a bejelentkezés. A két komponens hasonlóan épül fel. Hibás adatok megadása esetén hibaüzenet jelenik meg, amely tájékoztatja a felhasználót az esetleges elgépelésért, vagy ha nem érhető el a weboldalt kiszolgáló szerver. (3. Ábra)

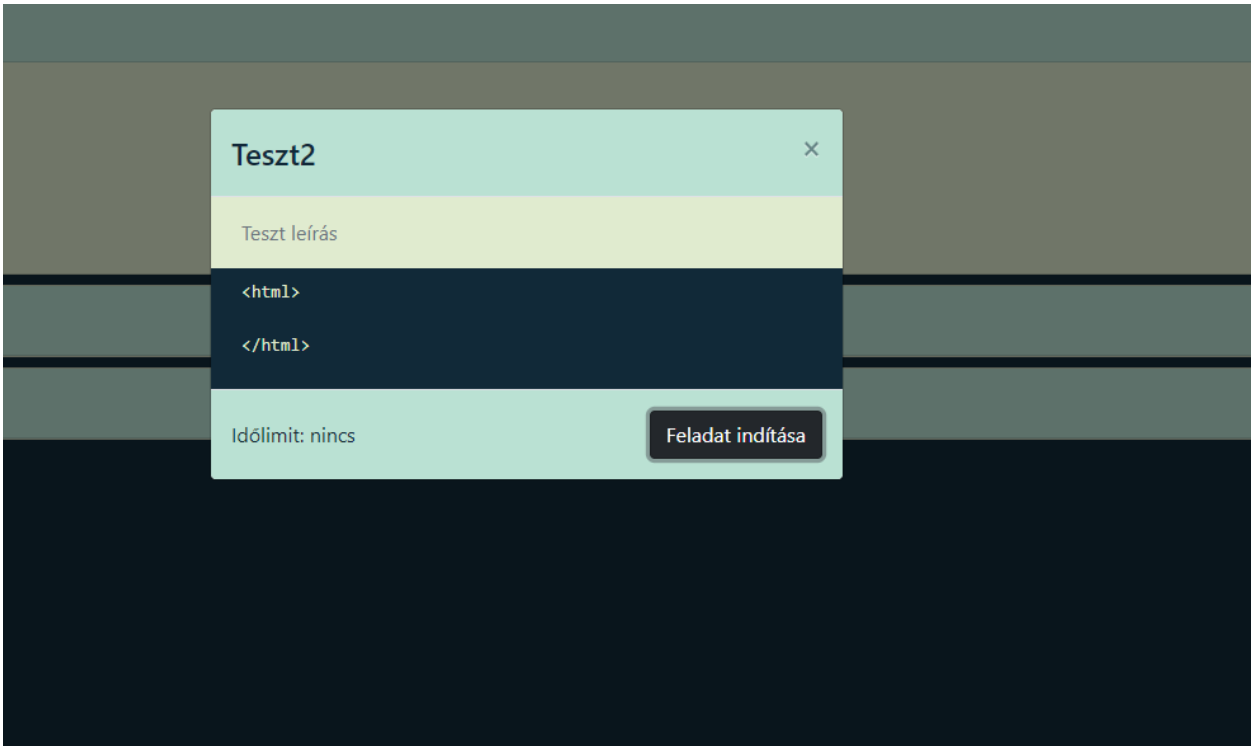


3. Ábra: A regisztrációt és a bejelentkezést szolgáltató komponensek.

Amennyiben a felhasználó már be van jelentkezve, elérhető számára a feladatok menüpont. Eddig a menüpontra kattintva egy értesítés ugrott fel, miszerint bejelentkezés szükséges a feladatok oldalra való navigáláshoz. A feladatok oldalon megjelenik az összes elérhető feladat, melyek közül válogathatunk, megtekinthetjük, hogy hogy néz ki a kiindulási kód, illetve megtekinthetjük, hogy mi a kiindulási kód. Amennyiben egy feladatot már megoldottunk, egy pipa jelenik meg a feladat címe mellett. A feladat továbbra is megoldható számunkra, viszont többszöri helyes megoldás esetén nem nő a mastery-szintünk. (4, 5. Ábra)

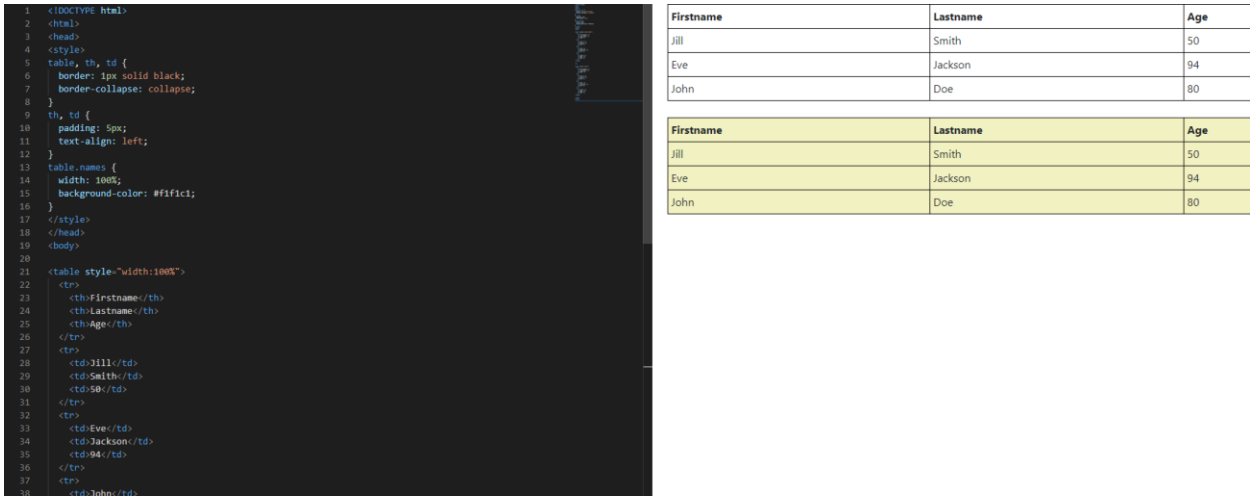


4. Ábra: A feladatok menüpont



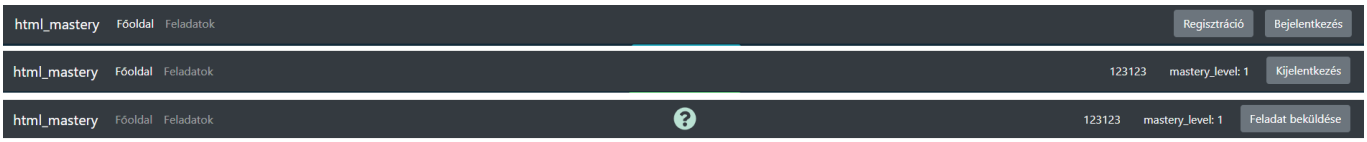
5. Ábra: Egy feladat részleteinek megtekintése

Feladat megoldásakor egy szerkesztőfelületre kerülünk, ahol elkezdhetjük a kódolást. A képernyő bal oldalán kódolunk, a jobb oldalon pedig az eredményt látjuk. Felül láthatjuk a hátralévő időt (amennyiben a feladat rendelkezik időkorláttal), illetve egy kérdőjelet, amelyre kattintva az 5. Ábrán látható módon a részletek innen is elérhetők. A jobb felső sarokban található gombbal adhatjuk be a megoldást. Helyes megoldás esetén (és amennyiben a feladatot még nem oldottuk meg) nő a mastery szintünk. Helytelen megoldás esetén folytathatjuk a feladat megoldását, viszont ha az időkorlát lejár, akkor navigáció történik a feladatok oldalra. (6. Ábra)



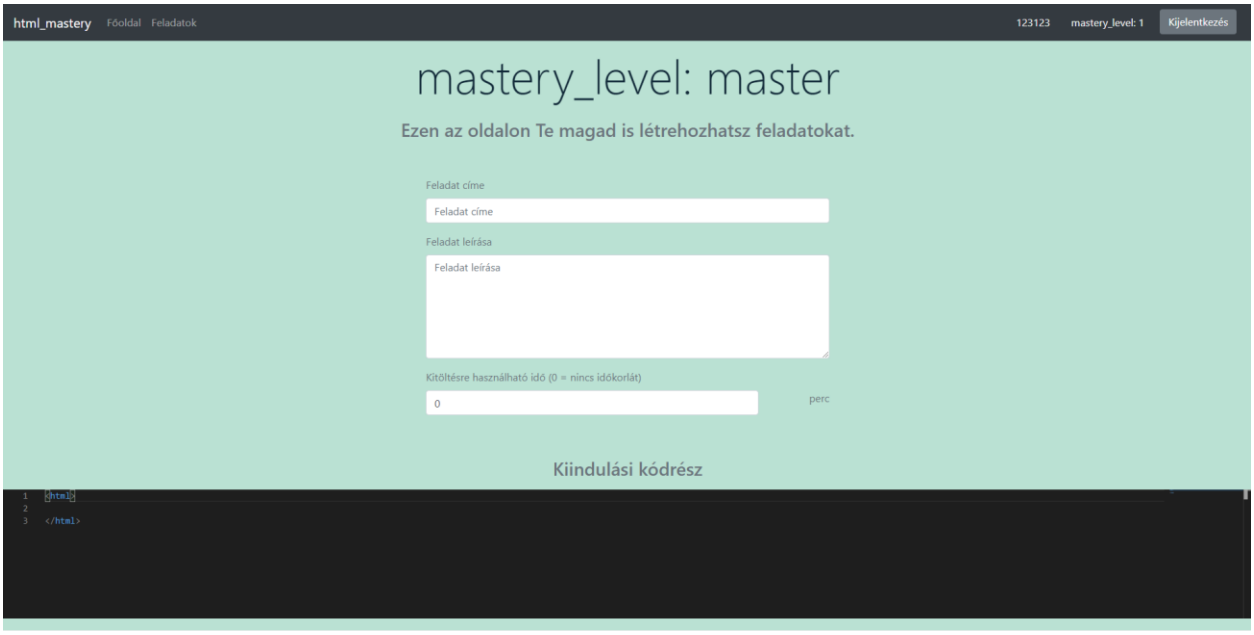
6. Ábra: A kódszerkesztő oldal.

A menüsor komponens tartalmazza a legtöbb funkcióbeli elemet. Ha a felhasználó nincs bejelentkezve innen léphet be, itt regisztrálhat. Bejelentkezett felhasználó láthatja a felhasználónevét, illetve a mastery-szintjét, feladatot itt adhatunk be és itt informálódhatunk róla, illetve természetesen itt történik a navigáció is. (7. Ábra)



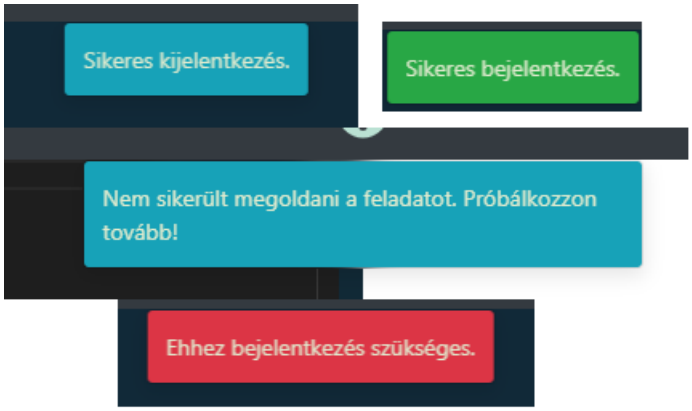
7. Ábra: A menüsor kijelentkezett, bejelentkezett felhasználó esetén, illetve feladatmegoldáskor

Az alkalmazás tartalmaz még egy feladatot benyujtó és szerkesztő oldalt, amely csak elérési út megadásával érhető el. Ennek útja ‘/i/can/now/give/others/tasks/because/i/am/at/master/level’, illetve szerkesztés esetén a végére illesztendő még a következő karaktersorozat ‘?title=<<feladat_címe>>’. (8. Ábra)



8. Ábra: Feladatszerkesztő easter egg

A felhasználó informálása toasterekkel történik, amelyek kis notifikációk a weboldal tetején, a menü alatt. (9. Ábra)



9. Ábra: különböző interakciókkor megjelenő értesítőablakok (toasterek)

Backend

A backend egy NodeJs alapú express alkalmazás, amely a frontend-del http kéréseken keresztül kommunikál. A felhasználók autentikálása a backend oldalon passport segítségével történik. A felhasználók a localStorage-be kerülnek tárolásra. Egy felhasználó tokenje 10 óráig érvényes.

Két fő útvonallal rendelketik: ezek a users és a task. A users útvonalon regisztrálhatunk új felhasználót, léphetünk be a webalkalmazásba. Itt történik még az aktuális felhasználó mastery_leveljének, illetve az általa megoldott feladatok lekérése. A task útvonalon kérhetjük le az összes feladatot, tölthetünk fel új feladatot, illetve szerkeszthetünk meglévőt. Ezen az útvonalon történik a feladat ellenőrzése is.

Nehézségek

Két komplikáltabb feladattal kellett megküzdenem. Előszöris meg kellett oldanom, hogy egy feladatot meg tudjunk oldani és ez renderelődjön ki, viszont a kiindulási forráskód úgy jelenjen meg, ahogy az megíráskor történt (indentálás, space-ek száma, sorvége karakterek). Szerencsére a kódszerkesztőben alapból úgy szerepelnek a kódrészek, mintha azok <pre> tag-ek között lennének találhatóak, így egyszerűen string-ként le tudtam menteni adatbázisba, ahonnan ugyanolyan szerkesztéssel kerülnek elő a kódok. Kódszerkesztőnek az ngx-monaco-editort használtam. Próbálkoztam még CodeMirrorral is, amely megemlíthető, illetve még több más kódszerkesztővel, de mindegyikkel valamilyen hibába ütköztem. Sajnos a monaco editor is hibás Angular 9-hez, mivel a kódajánlások nem jelennek meg, viszont ott vannak, illetve ctrl+space-el olvasható is minden ajánlás leírása.

A másik probléma, amelyet meg kellett oldanom az a szerkesztett html kód összehasonlítása az elvárt megoldásként megadott kóddal. Ennek megoldása az alábbi módon történik: (10. Ábra)

```
const trimHtml = function(str: string) {
  return str.replace(/(\r\n|\n|\r|\t)/gm, '')
    .replace(/  +/g, ' ').replace('> <', '><')
}
```

10. Ábra: összehasonlítás megoldása

Először kiveszem a tabulálásokat, illetve a sorvége karaktereket. Szóköz karaktereket nem lehet ezzel a módszerrel kivenni, hiszen ekkor kivételre kerülne az egyes megjelenítési osztályok közötti szóköz is, illetve a tag-eken belül található összes attribútum összeolvadásra kerülne. A következő lépésben az egymás után következő szóközöket cserélem re egy darab szóközre, utoljára pedig a tag-ek közt található szóközöket szűntetem meg.

Viszont egyszerű szöveg alapján nem határozható meg, hogy két html kód ugyanazt eredményezi-e, ezért html2json segítségével a két html kódot (megoldás és elvárt megoldás) json-né alakítottam. Ehhez a lépéshez kellett egyébként az alap szöveg html-t is trimmelnem, mert a tag-ek közt található szóközöket is node-okként érzékelte. Ezután a lodash könyvtár isEqual

metódusának köszönhetően a két json-ről meg tudtam mondani, hogy megegyeznek-e, így ennek eredménye határozza meg, hogy a feladat sikeresen lett-e megoldva. (11. Ábra)

```
export const reviewTask = async (req: any, res: any) => {
  const completeTask: ReviewableTask =  {
    username: req.body.username,
    task_title: req.body.task_title,
    source_code: req.body.source_code
  }
  Task.findOne({title: completeTask.task_title}, (err, task) => {
    if (err) {
      return res.status(constants.HTTP_STATUS_INTERNAL_SERVER_ERROR)
        .send({msg: 'Error occured'})
    }
    if (!task) {
      return res.status(constants.HTTP_STATUS_BAD_REQUEST)
        .send({msg: 'No task found with given name'})
    }
    let comp = trimHtml(completeTask.source_code)
    let sol = trimHtml(task.solution)
    const result = _.isEqual(html2json(comp), html2json(sol)) ? "success" : "fail"
    if (result === "success") {
      userController.giveMasteryPointToUser(completeTask.username, completeTask.task_title,  res)
    }
    return res.status(constants.HTTP_STATUS_OK)
      .send({result: result})
  })
}
```

11. Ábra: Feladat ellenőrzésének menete