JEGYZŐKÖNYV

Web technológiák 2

Féléves feladat/beadandó

Webapplikáció készítés

Készítette: **Simkó Levente**

Neptunkód: **YG5YAO**

Dátum: **2024. 06. 16.**

**Feladat:**

Az elvárt feladat szerint valamilyen termék nyilvántartást választottam azon belül is egy Legókat nyilvántartó weboldat készítettem, ami Materiált használ megjelenítéshez és MongoDB-t az adatbázishoz. A weboldal belépéi oldallal nyílik meg először, ami, ha nem beregisztrált felhasználó esetén át lehet térni egy regisztráló oldalra. De miért Lego? A Lego egy kreativitást fejlesztő játék, ami kis egymásba illő és kapcsolódó darabok segítségével lehet különböző dolgokat alkotni. Egészen kis kortól (akár 1 éves) bárki hozzáférhet (Lego szerint 99 éves korig bezárólag) különböző Témákban és egyéb Lego termékekhez.

**Indítás:**

A programkód sikeres cloneozása után a futtatáshoz összesen 3 parancs szükséges, és két terminál ablak. Az első parancs a *npm install* ami a kódhoz tartozó packege-ket leszedi és telepíti, ilyenkor mind a kliens oldali mind a szerver oldali package-ek települnek. Ezek után a *npm run start-backend* parancsot kiadva indítható el a szerver, ami a MongoDB-hez csatlakozik, illetve a localhost 8080 portján fut, a szerver kezeli a különböző klienstől beérkező adatokat, illetve az továbbítja szervertől a felhasználó felé az adatbázisból a lekért adatokat. Kliens indítása pedig a *npm run start* segítségével történik, ami a felhasználói felület futtatásáért felel és a localhost 4200 portján fut. A kliens oldalon történik adatok validálása az adatoknak mint például belépési vagy a Legókhoz kapcsolódó adatok.

**Belépési oldal/Regisztrációs oldal:**

A kliens oldal alapvetően a belépési oldal felé irányít, mert azt feltételeztük, hogy a felhasználó nem először van ott viszont könnyen hozzáférhető egy linken keresztül a regisztrációs oldal. A regisztrációhoz szükséges megadni egy Nevet, Email címet, Telefonszámot, Jelszót, A jelszót a szerver elhasheli majd az a hash kerül eltárolva az adatbázisban. A belépéshez szükséges email cím és jelszó kombináció, amit átalakít tokenné majd elkapja egy elkapó azt nézi, és az alapján lehet továbbmenni a további oldalakra úgy, hogy nem kell mindegyik oldalra a jelszó emailcím kombinációt., van egy guard ami azért felel hogy csak akkor lehessen az oldalt használni ha van token szóval be kell hozzá lépni egyébként átnavigálja belépési oldalra a felhasználót.

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

**Belépési/Regisztrációs oldallal kapcsolatban megkötések:**

A regisztrációhoz kötelező megadni a nevet, telefonszámot és az e-mail címet, valamint a jelszót a jelszón és az email címen további validátorok vannak jelen, jelszóra van egy megkötés, hogy minimum 6 karakternek kell lennie, míg e-mailhez van a beépített validátor. A Telefonszámra pedig normál regex felel.

**Regisztrációs oldalon:**

registerForm: FormGroup = this.formBuilder.group({

name: ['', Validators.required],

email: ['', Validators.compose([Validators.required, Validators.email])],

phone\_number: ['', Validators.compose([Validators.required, Validators.pattern(/((?:\+?3|0)6)(?:-|\()?(\d{1,2})(?:-|\))?(\d{3})-?(\d{3,4})/g)])],

password: ['', Validators.compose([Validators.required, Validators.minLength(6)])]

});

**Belépési oldalon:**

loginForm: FormGroup = this.formBuilder.group({

email: ['', Validators.compose([Validators.required, Validators.email])],

password: ['', Validators.compose([Validators.required, Validators.minLength(6)])]

});

Ezek a validátorok a html fájlba vannak bekötve, valamint a materiál miatt elvégzi a hibák kijelzését, ha nem megfelelő formátum vagy nincs kitöltve valamelyik mező. Amíg valamelyik validátor nem megfelelő nem tud a felhasználó belépni, vagy nem tud regisztrálni az oldalra.

**Belépés után:**

A belépést követően egy Kezdőlap fogadja a felhasználót, amin egy menürendszer található. Ezen a menürendszeren 3 további oldalról lehet választani, első a kezdőlap, amin a felhasználó alapból is landol második a Legók névvel ellátott lehetőség, ahol a rendszerben lévő Legók kerülnek kilistázásra, illetve, ha harmadik az Új Legó opció, ahol új Legót lehet felvenni az adatbázisba. Valamint lehetőség van a menüsor végén egy kijelentkezésre, ilyenkor visszatér a bejelentkezési oldalra.

A képen szöveg, Betűtípus, fehér, Grafika látható

Automatikusan generált leírás

A Legók menüpont kiválasztásával az alábbi kép fogad bennünket, itt egy táblázatban fel van tűntetve az összes adat, és ha egy bizonyos termékszámot elér akkor további lapokat csinál elkerülve ezzel a hosszú táblázatot, ami miatt a végtelenségig lehetne tekerni az oldalt. Lehetővé teszi létező Lego módosítását, valamint annak törlését.

A képen szöveg, képernyőkép, sor, Betűtípus látható

Automatikusan generált leírás

Ha a módosításra kattintunk akkor ugyan azt az oldalt kapjuk meg mintha új Lego-t adnánk meg azzal az eltéréssel, hogy a mezői üres helyett kitöltve jelennek meg így azokat tudjuk módosítani majd menteni őket az adatbázisba, illetve ezáltal ugyan azok a validátorok kerülnek alkalmazásra, mint ha új Lego-t vennék fel.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

Új Lego-t a menüből a neki megfelelő menüpont kiválasztása után férhetünk hozzá az alábbi laphoz.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

Itt megadhatjuk a Lego nevét, árát leírását és az ajánlott kort. A Lego minden egyes termékét ellátja egy korhatár besorolással, ami lehetővé teszi azt, hogy a megvétel előtt meggyőződjünk a korcsoportról, hogy kinek ajánlják. Vannak kis gyerekeknek készülő termékeik (Lego Duplo) illetve nagyobb korosztálynak szánt (Lego Technic) amik már inkább ilyen technikai irányba megy el és igyekeznek vele akár mechanikus dolgokat is megoldani.

**Új Lego oldal formja:**

legosForm: FormGroup = this.formBuilder.group({

name: ['', Validators.required],

price: ['', Validators.compose([Validators.required, Validators.min(500)])],

description: ['', Validators.required],

recomendedAge: ['', Validators.required],

});

Mint látható a Formhoz kötődik egy pár validátor, a recomendedAge esetén lehetne még egy normál regexet bevezetni, hogy ellenőrizzük, hogy például 1-99 közötti-e a korbesorolás, de az majd csak későbbiekben fog megtörténni. Az ár alapján látszik, hogy minimum 500 Ft-nak kell lennie egy terméknek, erre azért van szükség, hogy még „leárazva” se legyen kevesebb mint 500 Ft.

**Fontosabbnak ítélt kódrészletek, illetve adatbázisból képek:**

A validátorok nagytöbbsége a lentebbi kód vagy annak módosított verziója alapján jeleznek vissza a felhasználónak:

getErrorMessage(err: any) {

    if (err.hasError('required')) {

      return 'Kötelezően kitöltendő mező!';

    }

    return '';

  }

  getErrorMessagePrice(err: any) {

    if (err.hasError('required')) {

      return 'Kötelezően kitöltendő mező!';

    }

    if (err.hasError('min')) {

      return 'Minimum ár 500 Ft';

    }

    return '';

  }

Ezek a html oldalon vannak bekötve mar-error tagek között így a Material tudja, hogy azt kell megjeleníteni hiba esetén.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Látható, hogy a jelszavak hashelt módon van eltárolva, de akár az email is lehetne hashelni így az emailek is védettebek lennének, de ugyan ez nem igaz telefonszámokra mert azt percek alatt le tudja generálni egy program így felesleges lenne vele törődni. Illetve látni, hogy az adatbázis hozzárakott még két adatot azzal kapcsolatban mikor volt létrehozva vagy módosítva.