Fejlesztői dokumentáció

2024

*Készítette: Zsiga Gergely és Szávai Balázs Külker Technikum*

Cél

Célunk egy program megírása volt amely segítség lehet az iskolánknak/iskoláknak.

A program a tanároknak illetve a vezetőségnek való visszajelzésben segít a diákok részéről, mint tanulási mint az óra illetve egyéb teljesítések kapcsán.

A programban regisztrálás majd belépés, illetve osztályunk választása után kiválaszthatjuk tanárunkat akit értékelni szeretnénk.

Miután kiválasztottuk, számos kérdésre csúszka segítségével adhatunk pontszámot.

A backend-t Angular keretrendszer, az adatbázist szerver biztosítja.

Frontendnek egy asztali, webes alkalmazás áll rendelkezésre.

A kitöltés után a böngészőben érhetőek el a visszajelzések. Az asztali alkalmazásból

adminisztrálhatók az eredmények. A webes felületen kezelhetők a tanárok és lekérdezhetők

osztályok szerint.

Felhasznált technológiák

-Backend

-Angular ( keretrendszer )

-Készítéshez használt programok

-dia

-Visual studio code 1.63.0

-Isomnia 2021.5.3

-phpmyadmin

-Frontend

-Asztali alkalmazás

Kódolási konvenciók

A kódot git verziókezelővel használjuk.

Alapkönyvtárak

- api

- database

- desktop

- doc

- mobil

- web

Az api backend, a web a webes frontend.

Végpontok

Pl.:

|  |  |  |  |
| --- | --- | --- | --- |
| Végpont | Metódus | Azonosítás | Leírás |
| /login | post | nem | Felhasználói bejelentkezés |
| /logout | post | igen | Felhasználói kijelentkezés |
| /groups | get | nem | Tanárok/Osztályok lekérdezése |
| /register | post | nem | Felhasználó regisztrációja |
|  |  |  |  |
|  |  |  |  |

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'trs\_app2';

}

Az Angular keretrendszerben az @Component dekorátor egy osztályt rendel egy komponenshez. Ebben az esetben az AppComponent osztályt rendeli a fő komponenshez.

A selector tulajdonság meghatározza, hogy az alkalmazás melyik elemében jelenjen meg ez a komponens. Ebben az esetben az app-root elemet választja ki.

A templateUrl tulajdonság a komponens HTML sablonjának az elérési útját adja meg. Ez a fájl tartalmazza a komponens HTML kódját.

A styleUrls tulajdonság pedig egy tömb, amely azoknak a CSS fájloknak az elérési útvonalait tartalmazza, amelyek stílusinformációkat tartalmaznak a komponenshez.

Az AppComponent osztályban található title változó az alkalmazás címét tárolja.

Általános működés:

A regisztráció, a bejelentkezés, a tanárok és

csoportok lekérdezésének végpontjai publikusak, ezek használatához autentikáció nem

szükséges. Az adatbázist a Angular keretrendszer biztosítja illetve innek nyerhetők ki aza adarok.

A kontrollerek modellekkel vannak kapcsolatban amelyek az adatkezelésért felelősek.

Osztályok

A képen szöveg, képernyőkép, képernyő, szoftver látható

Automatikusan generált leírás

Egy Angular egységteszt (unit test), amely a LoginComponent nevű komponenst teszteli.Az egységteszt célja, hogy ellenőrizze, hogy a LoginComponent helyesen működik-e, és megfelelően van-e inicializálva.

Az egységteszt folyamata a következő lépésekből áll:

Fixture és komponens létrehozása: A TestBed.createComponent() metódus létrehoz egy új komponenst a megadott deklarációk alapján.

A fixture.detectChanges() metódus meghívja az Angular változásdetektálás mechanizmusát, amely beállítja a komponens és a hozzá tartozó nézet kezdeti állapotát.

Teszteset futtatása: Az it() blokkban található teszteset meghívja az expect függvényt, hogy ellenőrizze, hogy a komponens helyesen létrejött-e. Ha a komponens helyesen inicializálódott, akkor az expect(component).toBeTruthy() kifejezés igaz értéket ad vissza.

it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

TestBed.createComponent: Ez a függvény létrehozza az AppComponent komponens példányát a tesztesetekben. Ezt használják a tesztesetekben az AppComponent osztály tagjainak (pl. title) ellenőrzésére.

A képen szöveg, képernyőkép, képernyő, szoftver látható

Automatikusan generált leírás

A Testbed.createcomponent() metódus létrehoz egy új komponenst. Egy Componentfixture objektumot állít elő, amely lehetővé teszi a komponens tesztelését.

Komponens inicializálása: A fixture.detectChanges() metódus meghívja az Angular változásdetektálás mechanizmusát, amely beállítja a komponens és a hozzá tartozó nézet kezdeti állapotát.

Teszteset futtatása: Az it() blokkban található teszteset meghívja az expect függvényt, hogy ellenőrizze, hogy a komponens helyesen létrejött-e. Ha a komponens helyesen inicializálódott, akkor az expect(component).toBeTruthy() kifejezés igaz értéket ad vissza.

Ez a tesztkód biztosítja, hogy a QuestionScComponent megfelelően inicializálódik és létrejön, ami fontos az alkalmazás stabilitása és megbízhatósága szempontjából. Az egységtesztek segítenek azonosítani a potenciális hibákat és problémákat a komponens fejlesztése során, és segítenek megelőzni a hibákat a későbbi fejlesztési fázisokban.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

Egy TypeScript interfész definíciója, amely egy osztálycsoportot reprezentál. Az interfész meghatározza az osztálycsoport tulajdonságait és típusait.

* class: Az osztálycsoport nevét reprezentáló string típusú tulajdonság.
* registrations: Az osztálycsoportban regisztrált diákok számát reprezentáló egész típusú tulajdonság.
* totalSchoolPoints: Az osztálycsoport összesített iskolapontjait reprezentáló egész típusú tulajdonság.
* averageSchoolPoints: Az osztálycsoport átlagos iskolapontjait reprezentáló egész típusú tulajdonság.
* classification: Az osztálycsoport osztályozását reprezentáló string típusú tulajdonság.
* teacherNames: Az osztálycsoport tanárainak neveit tartalmazó string tömb típusú tulajdonság.

Ez az interfész lehetővé teszi, hogy egy osztálycsoportot leíró objektumokat hozzunk létre, amelyeknek minden tulajdonsága meg kell feleljenek az interfész definíciójának.

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

Ez egy Angular szolgáltatás, amely az autentikációt és a felhasználói adatok kezelését végzi. Néhány kulcsfontosságú funkciója a következő:

getAllData(): Ez a metódus egy HTTP GET kérést küld az adatbázis szervernek a felhasználók összes adatainak lekérésére.

registerUser(userDetails: User): Ez a metódus egy HTTP POST kérést küld az adatbázis szervernek, hogy regisztrálja az új felhasználót a megadott felhasználói részletekkel.

getUserByEmail(\_email: string): Ez a metódus egy HTTP GET kérést küld az adatbázis szervernek, hogy lekérdezze a felhasználót az email cím alapján.

saveUserResults(results: { result\_tc: number; result\_sc: number }): Ez a metódus egy HTTP POST kérést küld az adatbázis szervernek, hogy mentse a felhasználó eredményeit.

updateAllData(): Ez a metódus egy HTTP GET kérést küld az adatbázis szervernek, hogy frissítse az összes felhasználó adatait.

logOut(): Ez a metódus kijelentkezteti a felhasználót. Először törli a felhasználói adatokat a helyi tárolásból (localStorage), majd átirányítja a felhasználót a kijelentkezési oldalra.

Ezek a metódusok az Angular HttpClient modulját használják az HTTP kérések küldéséhez és fogadásához. A szolgáltatás a gyökér injektálóval van deklarálva, így globálisan elérhető az alkalmazásban.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Ez egy TypeScript konfigurációs fájl (tsconfig.spec.json), amely specifikusan az Angular alkalmazás egységtesztjeinek (spec fájlok) fordítására szolgál.

extends: "./tsconfig.json": Ez az opció arra szolgál, hogy az aktuális konfigurációs fájl a tsconfig.json fájltól örökölje a beállításokat. Ez azt jelenti, hogy a specifikus beállítások mellett az összes általános beállítás is érvényes lesz.

compilerOptions: Ezek a fordító beállításokat határozzák meg. A outDir beállítás megadja, hogy hová kerüljenek az egységtesztek fordítás utáni kimeneti fájljai. A types beállításban felsoroljuk azokat a típusokat, amelyeket a fordítónak fel kell használnia a kódban.

A jasmine típus a Jasmine keretrendszer típusait adja meg.

include: Itt adjuk meg, hogy mely fájlok kerüljenek bele a fordításba.

Ez a konfigurációs fájl segít az Angular alkalmazás egységtesztjeinek fordításában és futtatásában, és biztosítja, hogy a fordító megfelelően kezelje a tesztelést.

Tesztelés

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Regisztrálást (e-mail illetve jelszó választása után enged be a rendszer) követően a rendszer tovább visz a „leírás” részhez

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A regisztrálást követően visszaigazoló üzenet érkezi.

A képen szöveg, sor, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

Itt történik a tanárunk kiválasztása, akit szeretnénk értékelni.

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

A képen szöveg, Betűtípus, szám, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, Betűtípus, szám, képernyőkép látható

Automatikusan generált leírás

Fejlesztői környezet

* XAMPP
* Adatbázis kezelése a mysql nevű klienssel.
* Visual Studio Code
* Laravel
* Angular
* Java FX

Fejlesztési lehetőség

Mint minden programozó mi is úgy állunk hozzá művünkhöz, sose lesz kész.

Mindíg van hova fejleszteni a programot. Számos újításon gondolkodtunk illetve tovább is lehene fejleszteni a dolgot. Mint kiegészítve nem csak a tanárok visszajelzésére hanem az iskolai dolgokra például hogy hogyan értékelik az iskolai előadásokat, különböző iskolán kívüli programot amit az iskola rendez. Mint például az Erasmus+ program a +1 szendvics program vagy a megemlékezésekről.