HÁZI FELADAT

Kliensoldali technológiák

Kővári Bence

2023.

Verziótörténet:

1.0	2018.02.19.	Első verzió, követelmények ismertetése, 5 példa feladat
1.1	2018.02.27.	További 2 példa feladat
1.2	2019.03.11.	iMSc követelmények pontosítása
1.3	2019.04.30.	Oxford API helyett Words API, platformspecifikus követelmények pontosítása
1.4	2020.04.09.	COVID-19 módosítások (módosult pontrendszer, kötelező dokumentáció)
1.5	2020.04.14.	Szótár feladat pontosítása
1.6	2021.02.10.	Y2021 update
1.6	2022.04.05.	Y2022 update
1.7	2022.04.13.	Y2023 update

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

1117 Budapest, Magyar tudósok körútja 2.
Q. ép. B. szárny 207. • www.aut.bme.hu
Telefon: 463-2870 • Fax: 463-2871

Bevezetés

Az aláírás megszerzésének feltétele a házi feladat határidőre történő beadása és legalább elégséges szint elérése. Az alábbiakban összefoglaljuk ennek részkövetelményeit

Témaválasztás

E dokumentum végén felsorolt feladatok valamelyikét kell megoldani. A kiválasztott feladatot nem szükséges külön egyeztetni a gyakorlatvezetővel.

Lehetőség van egyedi feladat készítésére is. Ez esetben a témát a 8. oktatási hét végéig jóvá kell hagyatni a gyakorlatvezetőddel. Az egyedi házi feladatok az előre kiírtakkal azonos pontrendszer szerint kerülnek értékelésre.

Technológia

A feladatokat a tárgyban oktatott 3 fő technológia **egyikével** kell megvalósítani. Ezek:

- Universal Windows Platform
- MAUI
- Angular (nem AngularJS)

A technológiák a pontozás szempontjából egyenértékűek. A döntést befolyásolhatja, hogy egyes technológiákkal már a tárgy elején, másokkal csak a tárgy vége felé ismerkedünk meg.

Kidolgozás

A házi feladat kidolgozása a félév során **önálló** feladat. **Az elkészült feladat másról való másolása és másokkal történő megosztása egyaránt tilos**. Az újrahasznált/másolt kódrészleteket automatizált eszközökkel vizsgáljuk, **bizonyítható plágium esetén minden érintett fél háziját érvénytelennek tekintjük és az aláírást megtagadjuk.**

Beadás

Az elkészült házi feladat forráskódját a tanszéki portálon keresztül fel kell tölteni, majd az utolsó gyakorlaton a gyakorlatvezetőnek az alkalmazást működés közben bemutatni. A demonstrációhoz a laborgép, illetve saját számítógép (laptop, távoli asztal stb.) egyaránt használható. A bemutató során a laborvezetőnek be kell mutatni:

- A működő alkalmazást
- Az alkalmazás forráskódját

A gyakorlatvezető a pontozást helyben végzi, ehhez a beadáskor kérheti a forráskód értelmezését és akár kisebb módosítását is.

A pótbeadáson (melyet a pótlási héten tartunk) lehetőség van azok számára is részt venni, akik az előző beadáson elért pontszámukat (az alkalmazáson időközben elvégzett módosításokkal) javítani szeretnék.

Pontozási rendszer

A gyakorlatvezetők a beadott feladatokat az alábbi pontozási rendszer szerint értékelik.

Követelmény	Elérhető pontszám
Funkcionalitás	6
Szerver oldali API használata	2
Platform konvenciók követése	3
Aszinkronitás	2
Platformspecifikus megoldások	3
Kommentezés	2
Design, ergonómia	2

A pontozási szempontokhoz az alábbi útmutatót adjuk. Az értékelők a teljes skálán értékelik a feladatokat, a lenti példák az egyes pontszámokért elvárt tartalmat illusztrálják.

Funkcionalitás (0-6)

- 0 – Az alkalmazást nem sikerült működés közben bemutatni, vagy a forráskódja nem elérhető
- 1 – Az alkalmazás elindul, de alapvető funkciók működésképtelenek benne
- 2-3 – Az alkalmazás elindul, egy-két fontos funkció nincs megvalósítva
- 4-5 – Az alkalmazás elindul, egy-két funkció bemutatása során hibára fut (vagy a funkció lefut, de a háttérben vannak pl. Javascript futtatási hibák, vagy adatkötési hibák)
- 6 – Az alkalmazás teljes mértékben megvalósítja a feladatkiírásban foglalt funkcionális követelményeket

Szerver oldali API használata (0-2)

- 0 – Az alkalmazás nem használ hálózati kommunikációt
- 0 – Az alkalmazás kommunikál hálózaton, de annak tartalmát a használt külső osztálykönyvtár elfedi
- 1 – Az alkalmazás hálózaton keresztül, JSON/XML formátumban küld/fogad adatot, de a megvalósítás nem elég robosztus (pl. nincs hibakezelés, ékezetes karaktereket rosszul kezeli stb.)
- 2 – Az alkalmazás hálózaton keresztül, JSON/XML formátumban küld/fogad adatot, a megvalósítás robosztus.

Platform konvenciók követése (0-3)

Az egyes platformok saját konvenciókkal rendelkeznek, melyek követését elvárjuk:

- 3– Angular: Module, component, service, template, metadata, data binding helyes használata (pl. üzleti logika és hálózati kommunikáció ki van csoportosítva szolgáltatásokba, a felület használ adatkötést, projektszerkezet megfelelő stb.)
- 3 – UWP, MAUI: MVVM minta helyes használata (nézetek nem tartalmazhatnak üzleti logikát, vagy hálózati kommunikációt, adatkötések helyes használata, jól strukturált projekt stb.)
- 2 – UWP: MVVM minta használata kisebb hibákkal
- 1 – UWP: MVVM minta használata nagyobb hibákkal
- 0 – UWP: MVVM használatának mellőzése, vagy jelentős hibákkal történő használata

Aszinkronitás (0-2)

- 0 – Van olyan I/O, vagy számításigényes funkció, mely a felhasználó felületet érzékelhető ideig blokkolja (akkor is 0 pont, ha az alkalmazás egyébként használ aszinkron mintákat)
- 1 – Az alkalmazás kisebb hibákkal használja a platformspecifikus aszinkron mintákat
- 2 – Az alkalmazás tökéletesen használja a platformspecifikus aszinkron mintákat és nyelvi elemeket

Platformspecifikus megoldások (0-3)

Itt egyedi, a platformra jellemző megoldások használatát értékeljük (az alacsonyabb pontszámú elemekből több is választható, ilyenkor pontszámuk összeadódhat, de legfeljebb 5 pont szerezhető). Néhány példa, és értékük

- MAUI:
  - 3 – az alkalmazás (Android, Windows, iOS) platformból legalább 2-n fut és tartalmaz (érdemi) natív kódrészleteket mindkét platformra
  - 2 – Adott operációs rendszerre jellemző Natív API/osztálykönyvtár használata
- UWP:
  - 1-2 – WinRT API-k használata (isolated storage, fájlkezelés, fénykép készítése, 3D-s megjelenítés stb.) (minden API külön pontot érhet)
- Angular:
  - 2 – Együttműködés natív JavaScript függvényekkel egyedi típusdefiníciós fájlokkal
  - 1-2 – böngésző local storage api használata,
  - 2 – architekturális elemek komplex használata (pl. összetett, több modulon átívelő routing szabályok)

Kommentezés (0-2)

Fontos, hogy a kódban a nyilvános (public, internal, vagy protected láthatóságú) függvények a platform konvencióinak megfelelően kommentezve legyenek (.NET-ben XML komment, TypeScriptben TSDoc), illetve a komplexebb függvények logikájának megértését is külön kommentek segítsék. Alacsony komplexitású házi esetén (kevesebb mint 15 kommentezett függvény) pont nem adható.

- 2 – Saját készítésű publikus elemek 90%-a kommentezett platformkonvenciók szerint
- 1 – Saját készítésű publikus elemek 50%-a kommentezett (platformkonvenciókat nem mindig követve)
- 0 – Kevesebb mint 15 függvény van kommentezve, vagy platformkonvenciókat nem követi a kommentezés

Design, ergonómia (0-2)

- 0 – Az alkalmazás alapvető ergonómiai/esztétikai hibákat tartalmaz (pl. túl kis gomb/közeli gombok az érintőképernyőn, nem elég kontrasztos színek, visszajelzés elmaradása stb.)
- 1 – Az alkalmazás szép, funkciói letisztultak, de nem követ különösebb UI irányelvet
- 2 – Az alkalmazás valamely nagy szoftvergyártó UI konvenciót hűen követi. Pl. Material design, Fluent design. Ennek érdekében külső osztálykönyvtárak felhasználhatóak.

Tipp: a fenti konvenciók nincsenek feltétlenül platformhoz kötve, a Material design esetében mindhárom környezethez találhatunk osztálykönyvtárakat.

iMSc

iMSc pontozás esetén az elvárás, hogy a házi feladat a 3 platformból kettőre megvalósításra kerüljön, melyek közül az egyik az Angular kell legyen. Az iMSc pontszám a két feladat közül a gyengébbre értékelt platform pontszáma lesz.

Feladatok

A feladatok listáját (részben a beérkező ötletek alapján is) folyamatosan bővítjük. A feladatok mindegyike nyílt szerveroldali API-ra épít, melyek regisztráció nélkül, vagy ingyenes regisztrációval elérhetők. Az előírt API-k csak ajánlások, a gyakorlatvezetővel egyeztetve más API is alkalmazható a kijelöltek helyett, vagy mellett. Elképzelhető, hogy egyes API-k nem, vagy nem pont úgy érhetők el, mint a feladat kiírása idején. Ilyen esetben bármilyen további, hasonló célú API használható a célra. Szintén elfogadható, ha a hallgató saját megvalósítású (pl. önálló laboratórium, vagy más házi feladat keretében készített szerver oldali API-ját használja)

Meghajtó

Készíts egy klienst valamely népszerű felhő tárhelyhez a következők közül:

- Dropbox: https://www.dropbox.com/developers
- Google Drive: https://developers.google.com/drive/
- OneDrive: https://dev.onedrive.com/

A kliensnek képesnek kell lennie a bejelentkezés kezelésére, a mappaszerkezet böngészésére, az alapvető fájl és mappainformációk megjelenítésére, továbbá fájlok le és feltöltésére.

Szótár

Készíts egy egyszerű szótárprogramot például a következő API-kra építve:

- Words API: https://www.wordsapi.com/[1]
- Yandex dictionary API : https://tech.yandex.com/dictionary

A kliens lehetővé teszi, hogy a felhasználó külön-külön kiválassza a forrás és célnyelvet (de csak olyan párokat, amelyekhez tényleges szótár is létezik, lásd „/languages”). A fordítás mellett az alkalmazás képes szinonimák lekérdezésére is (ehhez szükség lehet további API használatra is http://thesaurus.altervista.org/service).

Arcfelismerés

Készíts egy alkalmazást, mely képes fényképről felismerni egy adott embert. UWP és MAUI esetében legyen lehetőség az eszköz kamerájával készíteni a fényképeket, Angular alapú megoldásnál elfogadható, hogy a képfájlokat a felhasználó választja ki. Mind sikeres, mind sikertelen azonosítás után jelölje az alkalmazás, hol talált emberi arcokat a képeken, illetve a talált arc fontosabb jellemzőit (pl. férfi/nő). Az alkalmazás a Microsoft Face API-t használja az azonosításhoz.

https://azure.microsoft.com/hu-hu/services/cognitive-services/face/

Könyvkereső

Készíts egy alkalmazást melyben a felhasználó kulcsszavak megadásával kereshet rá könyvekre. Az alkalmazás ezt követően listázza a találatokat, melyeket kiválasztva további információkat tudhat meg a könyvről. Fontos, hogy az alkalmazás a találatoknál minden esetben jelenítse meg az egyes könyvek borítóképeit is (amennyiben az rendelkezésre áll). A találat részleteit az alkalmazás gondos részletességgel jelenítse meg egy, az eszköznek megfelelő elrendezésben. (pl. ha a szerző honlapjára van url, akkor a szerző neve linkként működjön, a borítóképre kattintva a nagyobb felbontás töltődjön

[1] A korábban ajánlott Oxford Dictionaries API (https://developer.oxforddictionaries.com) számos funkciója fizetőssé vált

le stb.). Az alkalmazás adjon egyszerű lehetőséget többféle szempont szerinti keresésre is (pl. cím alapján, téma alapján stb.). Megoldásában az Open Library API-t használja!

https://openlibrary.org/developers/api

TRÓNOK HARCA

Készíts egy alkalmazást, mely a népszerű „Trónok harca" könyvsorozat kiadásait és szereplőit mutatja be. Az alkalmazás az Ice And Fire API-ra (https://anapioficeandfire.com/) épüljön. A kész alkalmazásnak képesnek kell lennie interaktív és ergonomikus módon bejárni a Trónok harca világát, vagyis az alkalmazás legyen képes a könyvek, a karakterek és családok listázására azok tömör megjelenítésére (pl. keresési találatnál) és azok **összes** adatának megjelenítésére egy részletező oldalon/felületen. A kapcsolódó entitások adatai (pl. egy szereplő rokonai) könnyen legyenek elérhetőek/bejárhatók.

FILM VILÁG

Készíts egy alkalmazást, mely általános segítséget ad az esti filmvadászatban. Az alkalmazás tegye lehetővé, hogy kulcsszavak alapján megkeressünk filmeket, sorozatokat. A találati listából kiválasztott elemhez jelenítse meg a részletes adatlapját legalább 5-6 fontosabb adatot kiemelve, továbbá sorolja fel a szereplőket, sorozatok esetében pedig az epizódokat (amennyiben az API biztosítja ezt). Az alkalmazás tegye lehetővé a böngészést, így pl. legyen a színészeknek is saját adatlapjuk, melyen a színész fontosabb adatai és a szerepei jelennek meg. Ezen felül az alkalmazás adjon tippeket a filmválasztáshoz, így például tegye lehetővé általánosan a legnépszerűbb filmek, vagy éppen egy adott filmhez/témához/műfajhoz kapcsolódó legnépszerűbb filmek listázását is (az API lehetőségei szerint). A feladat megoldásához a következő 2 API valamelyikét ajánljuk:

- http://docs.trakt.apiary.io/
- https://www.themoviedb.org/documentation/api?language=hu

KÉPNÉZEGETŐ

Készíts egy képnézegető alkalmazást a Flickr API-ra, vagy nyilvános Feed-ekre alapozva!

- https://www.flickr.com/services/feeds/
- https://www.flickr.com/services/api/

A felhasználó az alkalmazásban képes kell legyen kulcsszavak alapján keresni a nyilvános fotók között. A fotók megjelenítése mellett az alkalmazás egy-egy fotóhoz jelenítse meg az összes hozzá tartozó metaadatot is (leírás, tagek, kategória, szerző stb.). Az alkalmazás adjon lehetőséget a kapcsolódó információk intuitív bejárására, így például a felhasználó nevére kattintva az adott felhasználó képei, adott tagekre a velük megjelölt képek jelenjenek meg, de legyen lehetőség pl. egy felhasználó nyilvános kedvenc képeinek a megjelenítésére is. Az alkalmazás tegye lehetővé, hogy a megjelenített információkat adott nyelvű tartalomra szűrjük.