

BMSZC Petrik Lajos Szakgimnázium
Budapest, 2022

2022.05.03.

UPDF

Webfejlesztő szakképzés 06135001

Rekenei Zolta

Nyilatkozat

Alulírott Rekenei Zolta kijelentem, hogy ez a szakdolgozat saját tudásom, önálló munkám terméke

Aláírás

Rekenei Zolta

Budapest 2022. 05. 03.

1. Bevezető	4
2. Témaválasztás	4
3. Üzletmenet-felmérés	5
4. Fejlesztői dokumentáció	6
4.1 Tervezés	6
4.2 Adatbázis	6
4.2.1 Alkalmazott fejlesztői eszközök	6
4.2.2 Felépítés	7
4.2.3 Táblák	8
4.3 Backend	9
4.3.1 Alkalmazott fejlesztői eszközök	9
4.3.2 Adatmodellek	9
4.3.3 Részletes feladat specifikáció	9
4.3.4 Továbbfejlesztési lehetőségek	10
4.4 Frontend oldal	10
4.4.1 Alkalmazott fejlesztői eszközök	10
4.4.2 Felület-terv	11
4.4.3 Részletes feladatspecifikáció	11
4.5 Kiegészítő fejlesztői eszközök	13
4.5.1 SonarCloud	13
4.5.2 DoxyGen	14
4.5.3 Selenium IDE	15
4.5.4 GitHub Copilot	16
4.5.5 Swagger	17
4.4.6 Továbbfejlesztési lehetőségek	18
5. Felhasználói dokumentáció	19

5.1 Program általános specifikáció	19
5.2 Rendszer követelmények	19
5.3 Telepítés és konfigurálás	20
5.4 A program felhasználói csoportok.....	20
5.4.1 Admin.....	20
5.4.2 User	21
6. Összegzés	21

1. Bevezető

Az általam megvalósítani kívánt rendszer egy online nevezése az UPDF, azaz Unity for Passion Dance Fest névre hallgató táncverseny részére. Itt létre kell tudni hozni egy felhasználót egyszerű autentikációval, hogy aztán email cím és jelszó párossal be tudjon lépni a rendszerbe, és utána tudjon hozzáadni versenyzőket, akikhez ezután nevezést tud rögzíteni. Az egyéb adatok a rendszerben statikusak, mert jellemzően minden versenyenben ugyanazok a korcsoportok, kategóriák, versenyszámok vannak, így a rendszerbe fixen rögzíthetőek. Ezek az adatok nagyon ritkán módosulnak egy ilyen táncversenynél, mivel általában 2-4 évente van szabályzat módosítás, amiket jellemzően a rendszerben le kell követni, de ilyen esetekben a módosítás szükségzerű. A statikus adatokat Adminisztrátor joggal rendelkező felhasználó tudja csak módosítani, minden egyéb más jogosultságú felhasználó pedig csak láthatási vagy felületen végzett funkcionális jogokkal bír a rendszerben és a rendszer működésében.

2. Témaválasztás

Munkám során, ahol hangtechnikusként, leggyakrabban táncversenyeken látom el a feladatomat és támogatom a versenyek lebonyolítását nagyon sok olyan problémával találkoztam, amire egy célszoftver megoldással tudna szolgálni, így elsősorban onnan merítettem példákat. Táncverseny lebonyolításához sok helyen papíron vezetik a nevezőket, a pontokat, a versenyszámokat, az aktuális versenyzőket, sorrendet és manuálisan rögzítik papír alapon, számolják ezeket össze, ami elég sok emberi erőforrást, időt igényel, és nem utolsósorban sok hibalehetőséget is rejt magában, hiszen több ember több dolgot tud félreértelmezni egy esetleges félreolvasás, vagy olvashatatlan rögzítés miatt.

De kezdjük el az elején; a verseny szervezésének megkezdésével, a nevezéssel, ahol a szervezők szempontjából az egész versenyszervezési folyamat indul. Alapvetően nem tűnik bonyolultnak ez a regisztráció. A jelenleg általam megismert estek többségében ez vagy email alapon, vagy egy kissé professzionálisabbnak vélt esetekben egy google form segítségével van megoldva. Az általam gondolt legszofisztikáltabb megoldás, egy, az internetről bárhonnán elérhető webes felület lenne. Ez a felület -mint megtudtam- az UPDF

táncversenynek eddig még nem áll rendelkezésére, ezért felvettem a versenyek fő szervezőjével a kapcsolatot ezügyben, hogy segíthessük és támogassuk egymás munkáját. Szerencsére neki nagyon tetszett az ötlet, így meg is lett a szakdolgozatom témája és meglett a megrendelőm is. Ez nagyon fontos volt számomra, mert motiváló volt, hogy egy olyan fejlesztéssel foglalkozhatok, ami számomra is értéket teremt és mások számára is felhasználható és értékes eszköz lehet a jövőben.

3. Üzletmenet-felmérés

Miután az igény felmérési szakaszában a google form megoldással kellett szembesülnöm, így ennek elemzése adta az alapot, amivel az első tervezési fázisba léphettem. Ennek részletes elemzését és a korábbi versenyszervezéssel kapcsolatos ismereteimet felhasználva, sikerült eljutnom az igény kiszolgáláshoz szükséges weblap jövőbeni koncepciójának elméleti tervéhez. Mielőtt nagy lendülettel elkezdhetem volna az alkalmazás működésének tervezési fázisát, úgy gondoltam, hogy alaposan végiggondolom a fejlesztés kereteit, illetve összeállítom a fejlesztéshez általam ismert legmegfelelőbb környezetet, mivel számomra nagyon fontos a biztonság és a dokumentáltság. Ehhez több segédprogramot, szolgáltatást is beszereztem, amik ezeket az elvárt támogatási elemeket a lehető leghatékonyabban és legértelmesebben szolgálják ki a fejlesztésem során. Az alkalmazott elemek kiválasztásánál a Doxygen és a SonarCloude eszközökre esett a választásom, amikről később szót fogok ejteni. Szükségessé vált a szolgáltatások összefogására egy új elemre is, ami szintén a kiválasztás egy új lépése volt. Véleményem szerint erre legalkalmasabb az AzureDevOps környezet lenne, de sajnos az egy fejlesztendő alkalmazás költségvetéséhez képest költséges megoldás lett volna, így további kutatásokat végeztem. Beleástam magam a GitHub által nyújtott ci/cd megoldásaiba, amely nagyon tetszett és az aktuális projecthez szükséges funkcionalitással bírt. Természetesen abban is csak addig a funkció csomagig építkeztem, amíg a diák licenz érvényessége elérhető volt számomra. Miután ezek a kezdeti lépései megvoltak az üzletmenet felmérésnek, jöhetett az ügyféllel való pontos funkció-specifikálás. Ebben a szakaszban olyan dolgok is letisztázódtak, amelyeket én gondoltam másképp és azok is, melyekre az igénylő nem gondolt az esetleges informatikai ismeret, illetve a lehetőségek rendelkezésére állásnak hiánya, vagy csak egyszerűen az eddig megszokott keretek által beszorított gondolkodás

miatt. A felmérést követően egy második körben a tisztázatlan és a felmerülő további kérdések egyeztetése után kezdődhetett az igazi tervezése az alkalmazásnak. A korábbi lépéseknek köszönhetően várható volt, hogy az igény teljes és még az első körben fel sem merült igényeket is teljeskörűen ki fogja elégíteni.

4. Fejlesztői dokumentáció

4.1 Tervezés

Először is végig kellett gondolni az általam használni kívánt környezeteket és azok kompatibilitását. Ez egy igen nehéz kérdés mindig és egyéb esetekben, amikor nem egy szakdolgozat témáját kell kidolgozni, akkor sosem egyedül végzem el. Az gondolom, hogy a fullstack mint fejlesztési modell elavult, legalábbis a tapasztalataim ezt mutatják, illetve akikkel beszéltem erről a témáról, azok is ezen az állásponton vannak. Az informatika szerteágazósága, a sok újabb és újabb fejlesztési lehetőségek és megjelenő trendek és elemek sokasága miatt jellemzően az informatikusok nagy része specializálódik egyegy szakterületre. Ennek köszönhető, hogy a szakemberek nem teljesen professzionálisan mozognak a backend és a frontend témában, illetve szintén nem jellemző, hogy mindkettő esetben teljes mélységben naprakész és komplex tudással rendelkezzenek. Én fejlesztői pályafutásomat a backend irányába tervezem és jelenleg is ebben fejlesztem magam, mert a megjelenítési, vagy design részben nem találtam kihívást és nem érzem ebben a számomra ideális fejlődési, vagy számomra komfortos irányt. Sok időt és energiát szántam a frontend fejlesztés elsajátításába, de továbbra sem érzem az általam kényelmes és megfelelő útnak, ezért úgy véltem, hogy a feladat mérete, és a feladat pontos specifikációja, illetve a leadási határidők miatt nem használok kliens oldali renderelt frontend frame workot, hanem az általam kedvelt ASP.NET környezet nyújtotta lehetőséggel élek, mivel a frontend framework szerver oldalon van rendereleve Razor megoldásokkal, így a html fájlok helyett cshtml-t használ, ezzel kihasználva a mély backend logika és technológia adta lehetőségeket.

4.2 Adatbázis

4.2.1 Alkalmazott fejlesztői eszközök

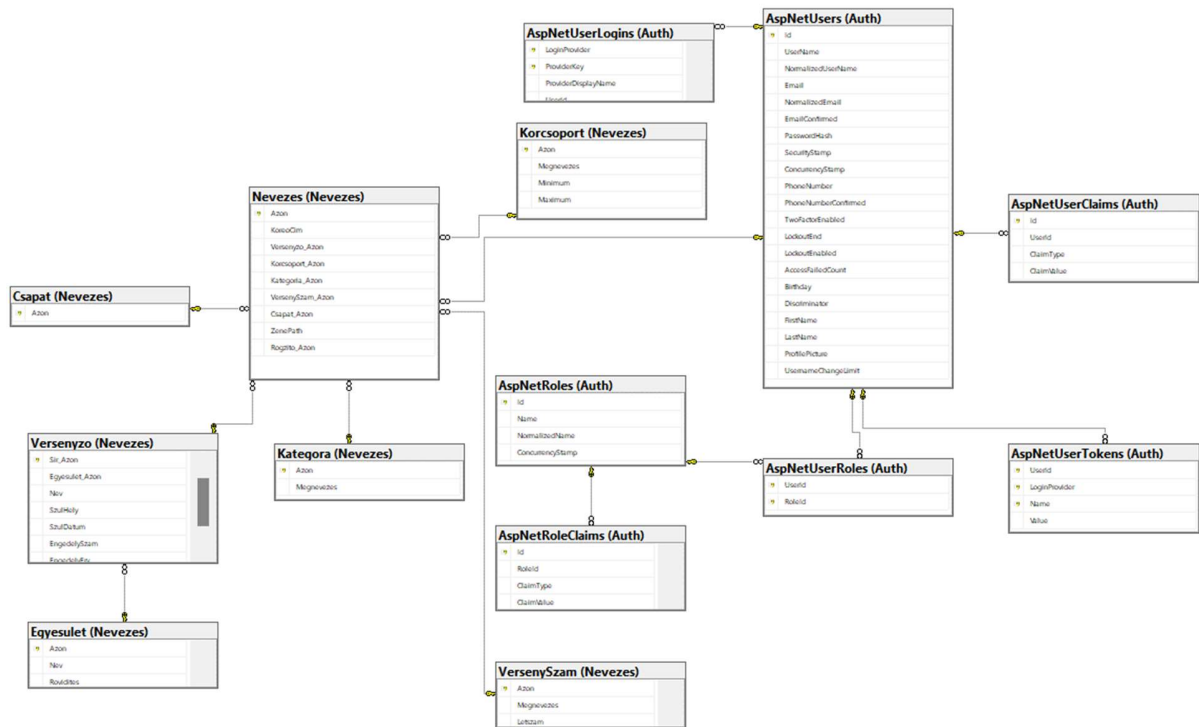
Mivel a backend ASP.NET környezetében fejlesztem, így kézenfekvő volt a tervezésénél, hogy akkor maradok a Microsoft termékcsaládnál a könnyebb és

biztosabb kompatibilitás miatt, így a Microsoft SQL server Developer mellett döntöttem az adatbázis kiválasztásánál. Nap mint nap dolgozok ezzel az adatbázissal az SQL Server Management Studio felületén keresztül, de ebben a projektben nem igazán volt szükségem arra, hogy ennek az eszköznek az erősségeit kihasználjam. A projektben előnyben részesítettem az O/RM az Entity Framework 6 eszközt, amin keresztül nem csak csatlakozni tudtam az adatbázishoz, hanem a teljes adatbázis létrehozást és migrációt is végre tudtam hajtani. Jelen feladatnál egy jól paraméterezett context segítségével, illetve egy speciális kommanddal létre tudtam hozni a migrációs fájlt. Ez a kommand az eszközben így elérhető „Add-Migration <egy név>”. Ezek után szükséges az adatbázis feltöltése, melyet szintén egy kommand segítségével egyszerűen és gyorsan el lehet végezni az eszközben. Ehhez a kommandon kívül az SQL Server manager Studióban használt, illetve megszokott SQL parancsokra nincs szükség, hanem egy egysoros kóddal ez végrehajtható egyszerűen és gyorsan. Az ehhez szükséges parancs a következő: „Update-Database”

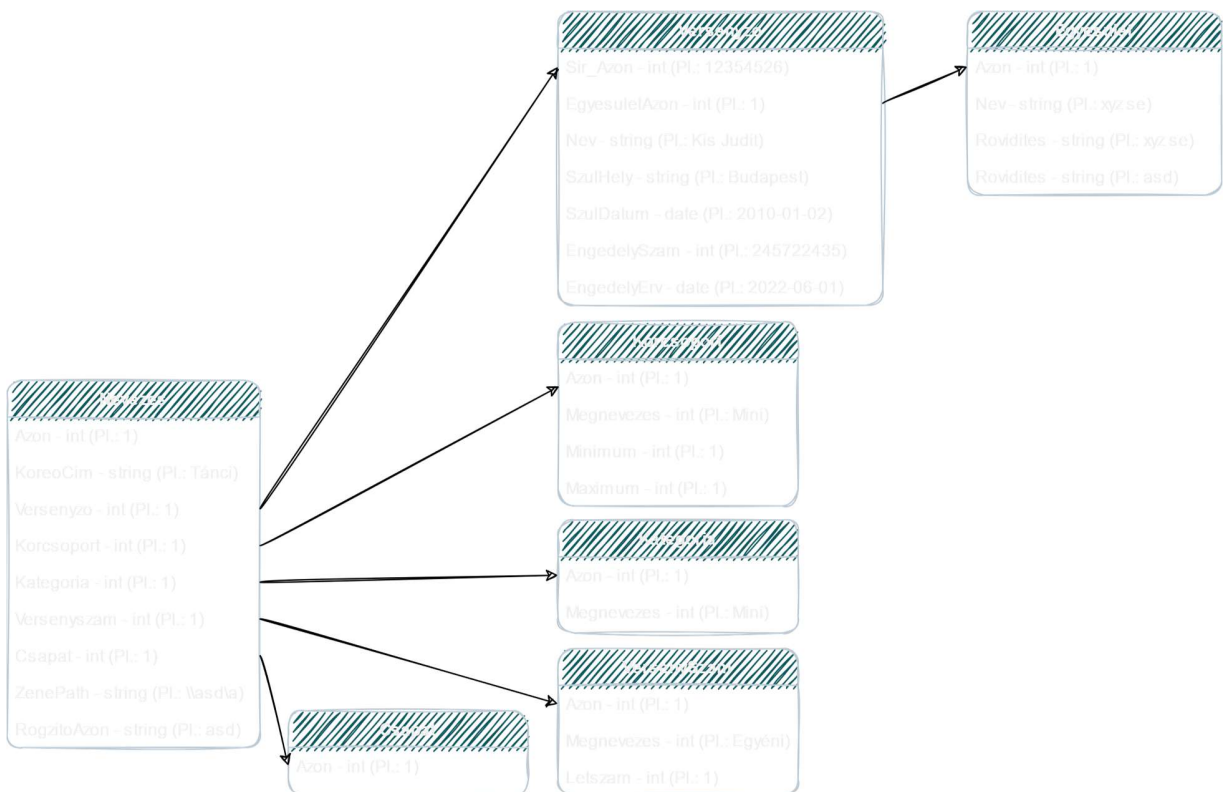
4.2.2 Felépítés

Egy-egy versenyhez létrehozandó adatbázis nem túl nagy, de úgy gondoltam, hogy két jól elkülöníthető logikai név térre szedem szét az adatbázist. Az egyik az „Auth”, mely tartalmazza az összes autentikációval kapcsolatos funkcióhoz szükséges adattáblát, indexet és minden ehhez szükséges kulcsot - Itt elérhetőek olyan elemek is, amik a továbbfejlesztési lehetőségek támogatása miatt lettek betervezve az adatbázisba, de jelenleg nem bírnak funkcionalitással -, illetve van a „Nevezes” név tér, ahol minden olyan tábla helyet kapott, ami már a verseny lebonyolításához és az azzal kapcsolatos funkciók ellátáshoz kötődik.

Felépítése az alábbi:



4.2.3 Táblák



4.3 Backend

4.3.1 Alkalmazott fejlesztői eszközök

Mivel a tervezési folyamat során a *c# ASP.NET* eszköz mellett döntöttem, ezért kézenfekvő volt, hogy a *Visual Studio* egyik változatát fogom választani. Mivel nagyon jó tapasztalataim vannak a *Visual Studio 2022* fő verzióval, ezért ebből is az *Enterprise* verziót választottam, mert az a legkomolyabb és a leghasználhatóbb változata. Az amúgy is jó fejlesztői környezetet megfűszereztem még a *CodeMaid* nevű kód formázó tool-lal és egy *GitHub* által még csak bétában elérhető, várólistás mesterséges intelligencia alapú kód kiegészítő extension-nel. Az adatbázis csatlakozásra a nem natív sql parancsok küldése mellett döntöttem, mivel azt elavult megoldásnak gondolom, hanem a *Microsoft* saját *O/RM* keretrendszerét használtam fel, amely sokban megkönnyíti a fejlesztést.

4.3.2 Adatmodellek

Az adatmodellek szinte teljes mértékben megegyeznek az adatbázisban fellelhető táblákon találhatóakkal, hiszen ezekből lett generálva az adatbázis. Annyi az eltérés, hogy itt külön „*Virtual*” nevű változókkal lehet a táblák közti összeköttetést lemodellezni. Ennek segítségével nem kell külön lekérdezéseket csinálni, ha a tábla csak egy azonosító kulcsot tárol, hanem arra az összekötő virtual változóra hivatkozva könnyedén le lehet kérni a hozzá tartozó tábla bármely paraméterét.

4.3.3 Részletes feladat specifikáció

4.3.3.1 Nevezés névtér

Mivel az MVC alapot követi a projekt, ezért minden modellnek van egy kontrollere és egy megjelenítése, vagyis egy view-ja. Ezeket nem csak a projekten belüli elhelyezkedése különbözteti meg, illetve a funkciója, hanem a név konvenciók is, ami alapján a .cs fájlok el vannak nevezve. Például a „*NevezesController.cs*” névből pontosan lehet tudni, hogy az a controller.

4.3.3.2 Auth névtér

Ez egy olyan terület, ami egy külön „area”-ba lett elhelyezve, mivel az logikailag teljesen elkülönül a nevezéstől, hiszen az csak egy előfeltétele rendszer teljes funkcionalitásának. Itt lelhető fel az összes olyan funkció, ami a

bejelentkezéshez szükséges és ehhez kapcsolódik, vagy fog kapcsolódni a továbbfejlesztési lehetőségek között.

4.3.4 Továbbfejlesztési lehetőségek

Mint minden rendszert, így ezt is végtelen mennyiségű funkcionalitással lehet bővíteni, melynek csak az igények szabnak határt. Mint tudatos fejlesztő, igyekeztem a rendszert ennek szellemében tervezni és fejleszteni. Az ötletek hosszú listája már a terveim között van, amikből a legfontosabbakat a lentebbi lista tartalmazza.

- Zene feltöltési felület a nevezéshez. Ennek segítségével nem kellene a versenyre minden egyesületnek egy pendrive-val érkezni és ott másolgatni, illetve feltölteni az mp3 fájlokat, hanem a nevezés mellé egyszerűen le lehetne adni és ezt egyben át tudná adni a verseny szervezője az aktuális technikus kollégának. Ezzel is könnyebben, zökkenőmentesebben indulhatna a verseny, zavartalanabb lenne a szervezés. Erre az opcióra az adatbázist a tervezési fázisban már felkészítettem, hiszen a nevezés közben megtalálható a ZenePath, ami a serveren lévő elérési út tárolására lenne használható. Az itt tárolt információ írná le, hogy az alkalmazás hol tárolja a feltöltött zenéjét a nevezőnek.
- Egy összetett versenyszervezői adminisztrátori felület. Ennek a megvalósítását a szakdolgozat utáni, továbbfejlesztési fázis fogja tartalmazni. A funkcionalitás teljeskörűségét, egy erre összpontosított igényfelmérés eredményeképpen fogom megtervezni.

4.4 Frontend oldal

4.4.1 Alkalmazott fejlesztői eszközök

Mivel nem egy külön keretrendszert használtam, hanem az ASP.NET-be foglalt Razor megoldást, így a fejlesztői eszközök nem tértek el a backendnél használt *Visual Studio 2022 Enterprise-től*. A Razor egy server oldalon renderelt speciális html fájl, melynek a kiterjesztése cshtml, ez mondhatni egy hibridje a HTML-nek és a C#-nak. Ezzel könnyedén lehet a benne létrehozott backend kódra hivatkozni és nem kell feleslegesen modelleket duplikálni és ide-oda hivatkozni, hiszen egy eszközön belül meg lehet oldani benne mindent, amire csak

szükségünk lehet. Ugyanúgy lehet használni benne a html összes adottságát, de a C# professzionalizmusával fűszerezve azt.

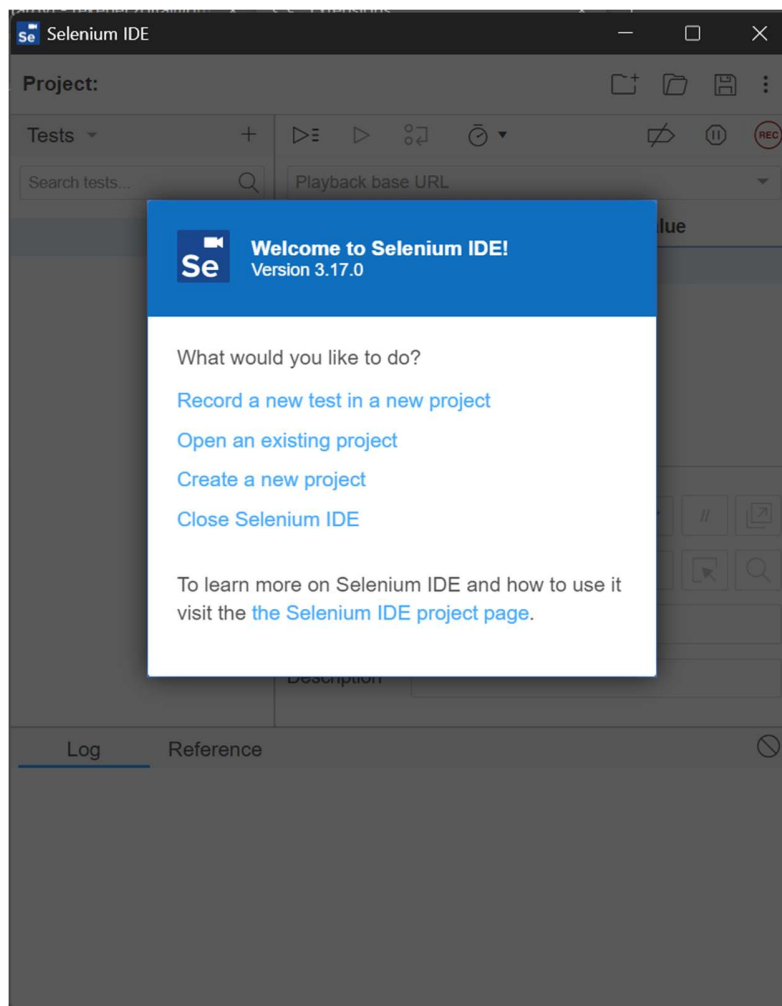
4.4.2 Felület-terv

Mivel már korábban is említettem, hogy a design nem a kedvelt szakterületem, ezért nem készült felület-terv. A jövőben, amikor ennek a tervezése sorra kerül, akkor terveim szerint nem én fogom elkészíteni, hanem az erre szakosodott designer kollégám, akivel ezekben a projektekben együtt szoktam dolgozni. Alapvetően a tervek szerint kétféle nézet lesz az alkalmazásban elérhető. Egy sötét és egy világos nézet. Ennek az az oka, hogy sokak -leginkább az idősebb korosztály- számára a világos sokkal olvashatóbb és átláthatóbb a nagyobb kontraszt különbség miatt, mint a sötét, viszont a verseny stílusát, témáját jobban komplementálja a sötét tónus.

4.4.3 Részletes feladatspecifikáció

A legnagyobb feladat, aminek a frontend fejlesztés során meg kell felelni, az a felhasználóbarát felület. Azt tudni kell ezekről a táncoktatókról, akik a nevezéseket intézik, hogy nagyon végfelhasználó tudással bírnak, tipikusan nem informatikai vénával rendelkeznek, így az általános informatikai megoldások megértése is néha nehézséget okoz számukra. A felhasználói felületeknek ezeknek a kihívásoknak kell megfelelniük, vagyis a lehető legegyszerűbbeknek és a lehető leginkább felhasználóbarát megoldásoknak kell lenniük, hogy a verseny szervezésében részt vevő emberek komfortosan érezzék magukat és az alkalmazással elégedettek legyenek, de ezzel egy időben a rendszer biztosítsa a hibázási lehetőségek kiküszöbölését is.. , 4.4.4 Tesztelés

Tesztelésre a sok szempontból híres [Selenium IDE](#)-t használtam, és a megfelelő használhatósága miatt a továbbiakban is ezt fogom használni hozzá. Az általam legjobban preferált ok a sok jó funkció mellett az, hogy az eszköz rendkívül felhasználóbarát. A Selenium egy olyan automatizálható felületen keresztüli funkcionális tesztelő program, amihez foghatóval még sosem találkoztam. Ha egyszer manuálisan végig kattintja a fejlesztő a tesztelni kívánt felületet és az lekódolja ezeket a lépéseket, vagyis minden



kattintásnál lementi az elem xpath értékét, akkor később ezen keresztül tudja újrafuttatásnál emulálni a kattintást. Természetesen a beviteli mezők sem okoznak az eszköznek gondot, és ami a számomra a legpraktikusabb a használatában, hogy nem kell hozzá semmi plusz informatikai elem, csak egy böngésző, ami vagy Chromium, vagy Firefox alapon működik és máris hozzá lehet adni bővítményként. Ezzel aztán gyorsan végig tesztelni a fejlesztett oldalt. Egy gombnyomással tudunk menteni és onnantól bármikor vissza tudjuk tölteni és újra futtatni azt a tesztet, anélkül, hogy egy sort is kellett volna kódolni. Viszont, ha mi már egy komolyabb műveletre vágyunk, akkor lehet az általa generált utasítás sorozatba -minimálisan ugyan-, de beleprogramozni és még mindig nem hagytuk el a böngészőt-. Ciklusokkal és tömbbel is bővíthetjük a tesztelési folyamat lépéseit, amivel megadjuk a bemeneti adatokat. Amennyiben még ennél is komplexebb tesztet szeretnénk végrehajtani, úgy

12 ■

kihasználhatjuk az eszköz exportálási lehetőségéből adódó, több programnyelvben felhasználható kódgenerálási lehetőségét.

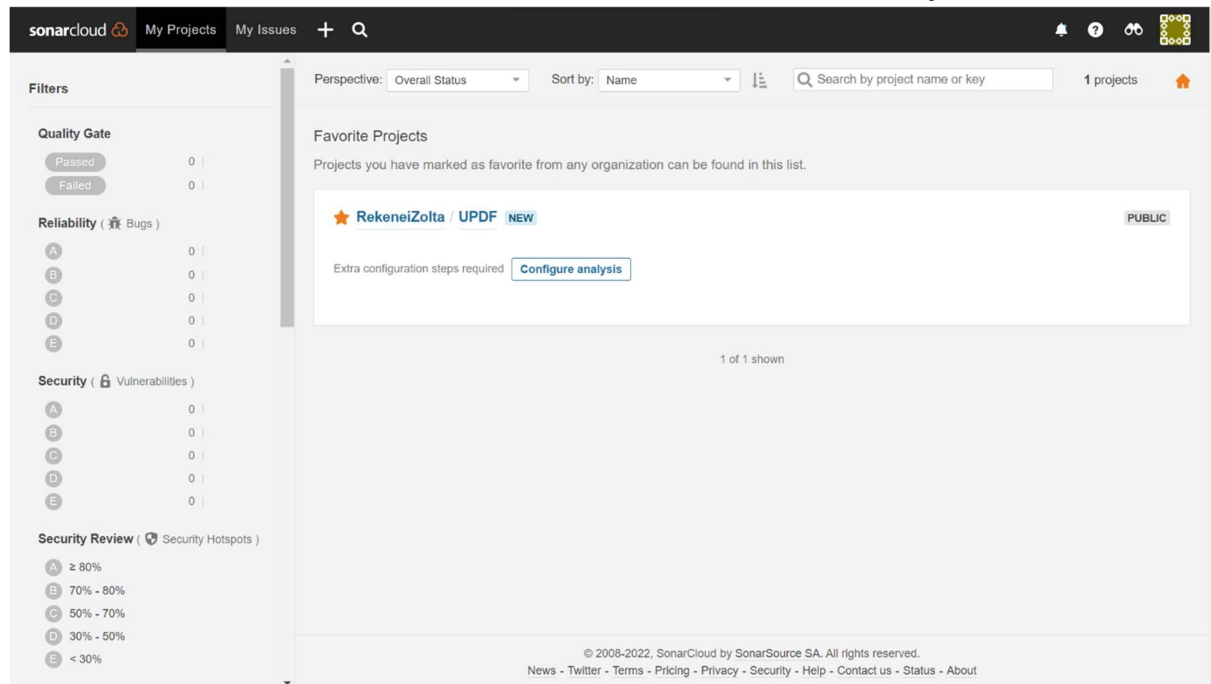
4.5 Kiegészítő fejlesztői eszközök

Korábban többször említettem már olyan eszközöket, melyek se nem frontend se nem backend igazán, hanem a fejlesztés megkönnyítésére hivatottak. Ezeknek a felhasználását és az automatizálását fejteném ki a következő fejezetben. Ezeknek a felhasználását és az automatizálását fejteném ki a következő fejezetben.

4.5.1 SonarCloud

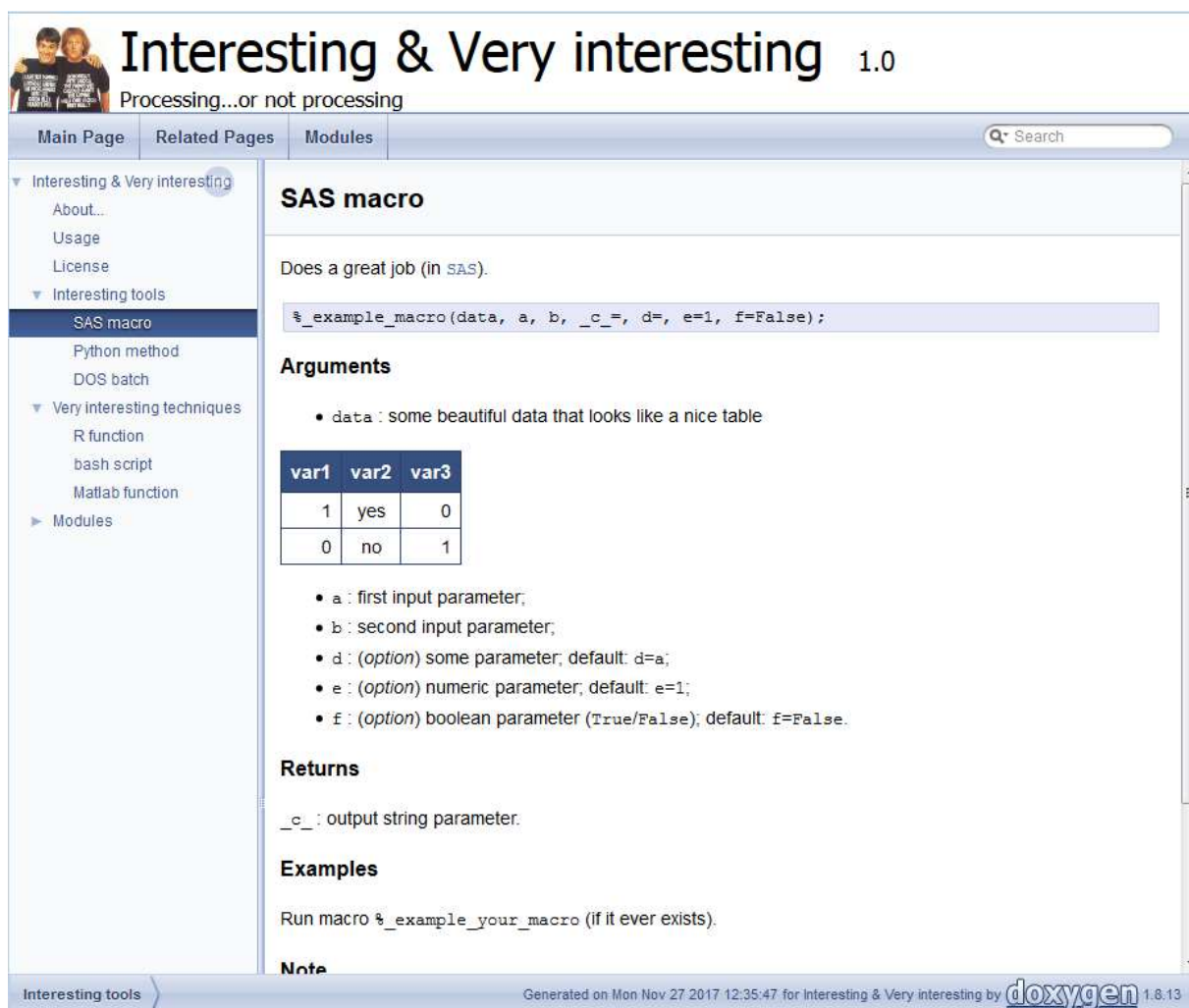
Ez egy olyan kód analizáló tool, amely a sok fejlesztő által elhanyagolt biztonságos fejlesztést segíti elő. A SonarCloud egy felhő alapú verziója kezelőkbe is beköthető pl. Azure, GitHub vagy egyéb CI tool valamelyikébe, így alkalmas arra, hogy például minden main szálban lévő változásnál lefusson és ellenőrizze a kódot. Amennyiben valaki nem szeretné, hogy a rendszere az internetre csatlakozva működjön, vagy biztonsági okokból ezt a fejlesztési project nem támogatja, abban az esetben a SonarQube lokális megoldását is választhatja, mely szintén teljes funkcionalitással bíró a távoli internetalapú elérések verzió kezelésének funkciói hiányától eltekintve. Hogy kimenjen a netre arra is van megoldás a SonarQube nevű tool-lal, ami ugyanezt a feladatot látja

el, csak lokálisan is futtatható saját szerveren.



4.5.2 DoxyGen

Ez egy régebbi nagy kedvencem. Nagy általánosságban elmondható, hogy a fejlesztők nem szeretnek dokumentálni, illetve amikor a dokumentálására sor kerül, már régen kész az alkalmazás. Az utólagos dokumentálásnak lehetséges hátulütője lehet, hogy egyes részek már nem lesznek pontosan kifejtve, vagy a múlt miatt már pontatlanul kerülnek a dokumentációba. Ezt támogatja a megfelelően formázott kommentekkel ellátott kódra, ha alkalmazzuk a DoxyGen-t funkcionalitását. Az eszközből és az általa látott file struktúrából egy tökéletes fejlesztői dokumentációt képes csinálni, amit utána akár html weblap formájában is képes exportálni, és maximálisan testre szabható.



Interesting & Very interesting 1.0
Processing...or not processing

Main Page | Related Pages | Modules | Search

▼ Interesting & Very interesting

- About...
- Usage
- License
- ▼ Interesting tools
 - SAS macro**
 - Python method
 - DOS batch
- ▼ Very interesting techniques
 - R function
 - bash script
 - Matlab function
- Modules

SAS macro

Does a great job (in *SAS*).

```
%_example_macro(data, a, b, _c_, d=, e=1, f=False);
```

Arguments

- data : some beautiful data that looks like a nice table

var1	var2	var3
1	yes	0
0	no	1

- a : first input parameter;
- b : second input parameter;
- d : (option) some parameter; default: d=a;
- e : (option) numeric parameter; default: e=1;
- f : (option) boolean parameter (True/False); default: f=False.

Returns

c : output string parameter.

Examples

Run macro %_example_your_macro (if it ever exists).

Note

Generated on Mon Nov 27 2017 12:35:47 for Interesting & Very interesting by **doxygen** 1.8.13

4.5.3 Selenium IDE

Igazából a 4.4.4 es pontban már kifejtettem, hogy a Selenium IDE egy funkcionális tesztelést automatizáló eszköz, melynek segítségével a tesztelés könnyebbé válik, nem kell többé id-kat keresni, vagy minden élesítés előtt

egyesével végig nyomkodni a felületeket, elég, ha csak az előre összeállított tesztelést elindítjuk.

4.5.4 GitHub Copilot

Selenium IDE - <https://www.microsoft.com/hu-hu/>

Project: <https://www.microsoft.com/hu-hu/>

Executing ▾

teszt

<https://www.microsoft.com>

	Command	Target	Value
1	open	/hu-hu/	
2	set window size	1280x720	
3	click	id=shellmenu_0	
4	click	css=.x-hidden-focus	
5	click	css=.collapsed > .x-hidden	

Command: //

Target:

Value:

Description:

Runs: 1 Failures: 1

Log Reference

Running 'teszt' 21:17:44

1. open on /hu-hu/ OK 21:17:45

2. setWindowSize on 1280x720 OK 21:17:45

3. click on id=shellmenu_0 OK 21:17:45

4. Trying to find css=.x-hidden-focus... Failed: 21:17:47
Playback aborted

'teszt' was aborted 21:17:56

Sajnos ezt a toolt teljes mértékben nem sikerült elsajátítani, mivel a fejlesztés elejétől nem tudtam alkalmazni. A fejlesztés közben a vendor várólistáján a sorra kerülésem után tudtam kezdeni a felhasználását. Az eszköz kiváló. Ez egy központi mesterséges intelligencián alapuló kód kiegészítő, ami minden fejlesztőnek a programozási szokásait elemzi, aki ezt használja. Az elemzés után az optimálisabbat elkezd tanítani a többi felhasználónak is. Többször javasolt jobb megoldást, mint amit én alkalmaztam volna. Sok magyar felhasználója van, így a magyar támogatása is igen kielégítő.

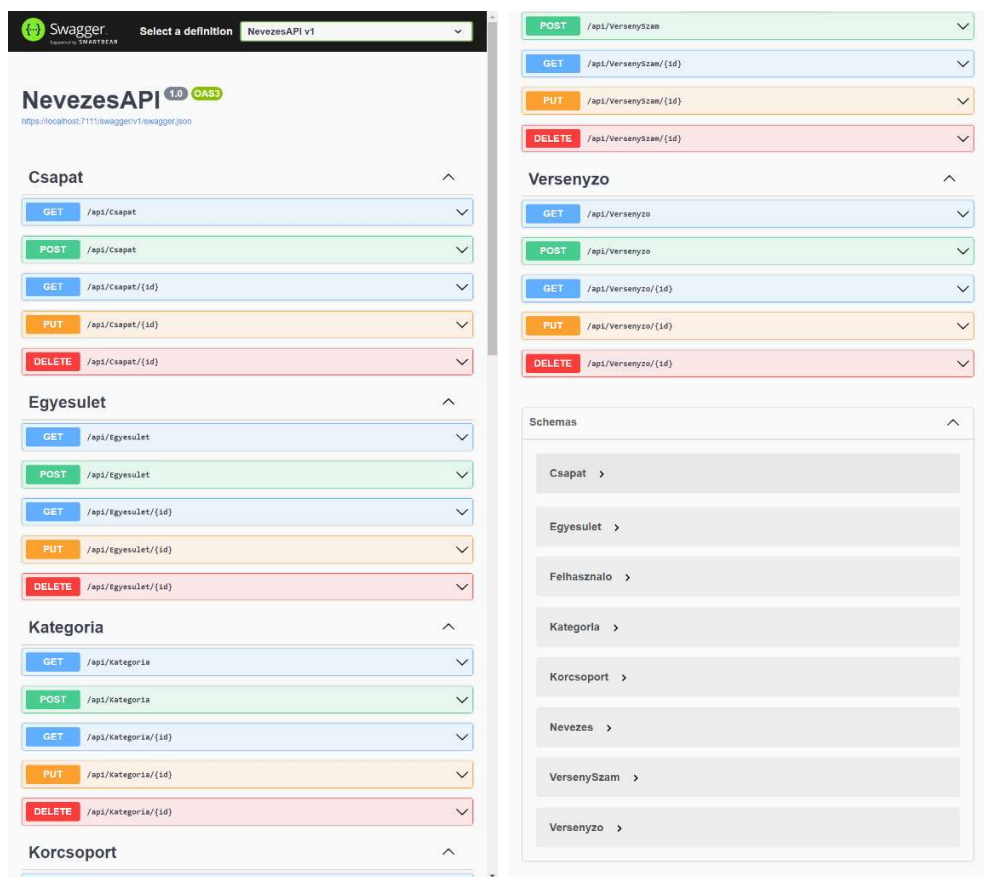
```

1  #!/usr/bin/env ts-node
2
3  import { fetch } from "fetch-h2";
4
5  // Determine whether the sentiment of text is positive
6  // Use a web service
7  async function isPositive(text: string): Promise<boolean> {
8    const response = await fetch('http://text-processing.com/api/sentiment/', {
9      method: "POST",
10     body: `text=${text}`,
11     headers: {
12       "Content-Type": "application/x-www-form-urlencoded",
13     },
14   });
15   const json = await response.json();
16   return json.label === "pos";
17 }

```

5.4.5 Swagger

Ez egy kódba építhető tool, mellyel ha van, akkor az API végpontokat lehet tesztelni, listázni, ellenőrizni, emellett értelmét veszti a Postman funkcionalitása is.



4.4.6 Továbbfejlesztési lehetőségek

A front-end elkészült funkcióit, jelen állapotában a back-end funkciók tesztelhetőségének segítségével korlátozódnak, így ebben még nagyon sok kiaknázandó lehetőség rejlik:

- Kétnyelvű felhasználói felület; angol és magyar. Mivel elképzelhető, hogy a közeli országok felé is nyit a verseny, így a többnyelvű funkciók valószínűleg hamarosan szükséges lesz.
- Téli versenyekre való jelentkezés idején hóesés animáció
- Idővel a verseny főoldalának kiváltása, ami jelenleg is csak egy statikus oldal. Az oldal folyamatosan frissülő versenyidőpontokat képeket tartalmaz. Lásd:

AKTUÁLIS


RÓLUNK

SZABÁLYZAT

BÍRÓK

GALÉRIA

KAPCSOLAT



AKTUÁLIS

Következő versenyeink:

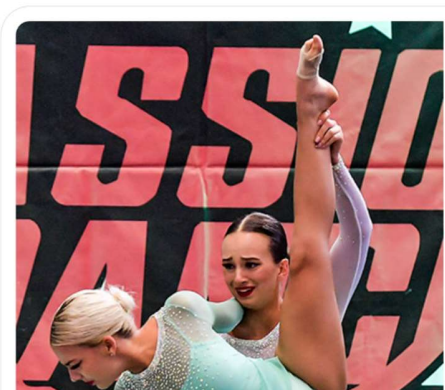
Május 22

Barátság Kupa UPDF

Június 24-26

Summer Section UPDF

Nevezés



- Teljes felülvizsgálata minden bemeneti mezőnek [XSS](#) ellen. Ebben nagyon sokat segít a Entity Framework. Az eszköz a bemeneti mezőket string-ként kezeli, így az ártó szándékú kódok bejutásának egy részét megakadályozza.
- A legnehezebb feladat egy weblapnál a felület folyamatos változtatása és fejlesztése anélkül, hogy ezt a felhasználó hátrányként fogná fel, mivel egy ilyen weblapot, ami a művészetről szól, nem szabad engedni egy elavult designnal üzemelni hagyni a nagyvilágban.

5. Felhasználói dokumentáció

5.1 Program általános specifikáció

Az alkalmazásnak szüksége van egy Microsoft SQL Szerver 2019re, amiben tárolja az általa kezelt adatokat, illetve szüksége van egy ASP.NET 6-ot futtatni képes környezetre, melyet később kifejtek. Szintén szüksége van egy internet böngészőre, melyen keresztül megjeleníti az alkalmazás adatait.

5.2 Rendszer követelmények

Az alkalmazás futtatásához szükséges egy Microsoft SQL szerver 2019 az mely már [Linux alatt is telepíthető natívan](#), így nem függ semmilyen rendszertől.

Ugyan Mac OS alatt nem működik, de natívan lehet Linux, Microsoft Windows rendszeren, vagy akár egy Docker konténerben is futtatni.

Az ASP.Net 6 web projekt is hasonlóan jó tulajdonságokkal bír, hiszen összesen egy operációs rendszerre van szüksége, ami képes a dotnet6 futtatására, vagy rendelkezik IIS szerverrel.

5.3 Telepítés és konfigurálás

Weblap futtatására több opció is lehetséges, ezeket felsorolom most az általam ajánlott sorrendben:

1. Windows operációs rendszeren [IIS](#), vagy [IIS Express](#) szerveren keresztül. Ezután az IIS re fel kell rakni a publish mappát egy új pullba, [és ezt végzi el a DevOps vagy az üzemeltető szakember](#).
2. Bármilyen olyan operációs rendszerről, ahová telepíthető a [dotnet](#) megfelelő verziója, ami jelen esetünkben a [.Net 6](#). Egyszerűen csak elnavigálunk a publish mappába command prompt segítségével, majd ott az alábbi kommandot tudjuk futtatni „dotnet UPDF.dll” és az alkalmazás elindul.
3. Egy erre felkészített Docker konténerbe behelyezni és elindítani
4. Ha nem éles környezetben, csak tesztelni szeretnénk és van visual stúdió eszközünk, akkor a projektet azon keresztül könnyen lehet futtatni a „futtatás gombbal”

Mindenekelőtt a publish mappában lévő exe fájlt kell futtatni, mert az megcsinálja magának az adatbázist, ha jól van beállítva az „appsettings.json” ben található adatbázis elérés.

5.4 A program felhasználói csoportok

Ennek a web alkalmazásnak két felhasználói csoportja van. Az egyik az Admin azaz a verseny szervező/szervezők és a nevezést leadó felhasználók, akik a User-ek.

5.4.1 Admin

Az adminoknak két feladatuk van. Az első, hogy ellenőrzik, hogy minden nevezéshez szükséges kategória, vagy egyéb feltétel megvan-e, majd a verseny előtt a leadott nevezéseket lekérni az alkalmazásból és átadni a pontozó rendszernek.

5.4.2 User

Ők az egyszerűbb felhasználói kör, hiszen nekik kell az alkalmazás funkciónak megfelelni, ők a végfelhasználók. Ha már van egy regisztrációjuk, akkor látják az általuk rögzített versenyzőket és a hozzájuk tartozó nevezéseket. Ha ez megvan, nincs más hátra, mint kijelentkezni és hátradőlni, főleg, ha a versenyzőik már tökéletesen tudják a koreográfiát.

6. Összegzés

Mivel ez egy hiánypótló és remélhetőleg a véglegesítés után sokat használt webalkalmazás lesz, így nagyon élveztem ennek a tervezését. Az ehhez tartozó fejlesztést segítő eszközök otthoni felhasználásra elérhető variációinak összegyűjtését, melyek nagyvállalati környezetben is jól beváltak. Ellenben mikor szembesültem azokkal a korlátokkal, melyeket a szakdolgozat állított elő, teljesen lelombozódtam, úgy vélem, hogy azok a megvalósítási elvek, azok az elvárások, teljesen addig validak, amíg régebbi, kötöttebb környezetekben gondolkozunk. Azt értem, hogy a létszámhiány miatt muszáj volt ragaszkodni a fullstack fejlesztéshez -habár számomra ez elég nagy negatívum volt-, viszont úgy gondolom, hogy azok az irányok, amiket tanultunk nem feltétlenül rosszak, de nagyon nem korszerűek. Már vannak ennél sokkal könnyebben és sokkal produktívabban fejleszthető rendszerek, környezetek és én éppen ezen véleményem miatt változtattam kicsit a Szakdolgozatom irányán és inkább az innovatív eszközökre fektettem nagyobb hangsúlyt, amik segítik és jelentősen támogatják a fejlesztést.

Források:

- [Unity for Passion Dance Fest \(updf.hu\)](#)
- [GitHub](#)
- [Stack Overflow - Where Developers Learn, Share, & Build Careers](#)
- [Visual Studio 2022 IDE - Programming Tool for Software Developers \(microsoft.com\)](#)
- [SQL Server 2019 | Microsoft](#)
- [Projects \(sonarcloud.io\)](#)
- [Selenium IDE · Open source record and playback test automation for the web](#)
- [Doxygen: Doxygen](#)
- [API Documentation & Design Tools for Teams | Swagger](#)
- [Entity Framework 6](#)
- [C# Corner - Community of Software and Data Developers \(c-sharpcorner.com\)](#)