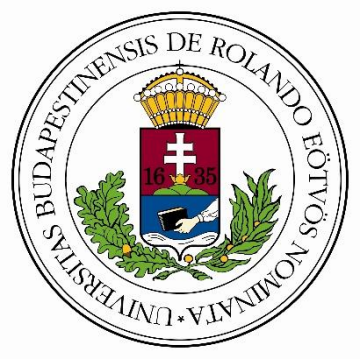


Eötvös Loránd Tudományegyetem

Informatikai kar

Média- és Oktatásinformatika Tanszék

Team Foundation Server - alapú valós idejű nyilvántartás automatizálás

Témavezető:

Dr. Illés Zoltán

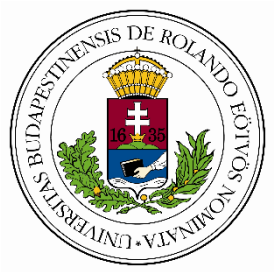
Habilitált egyetemi docens

Készítette:

Nemes László

Programtervező informatikus BSc

Budapest, 2020



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

SZAKDOLGOZAT-TÉMA BEJELENTŐ

Név: **Nemes László** Neptun kód: **RVR55V**

Tagozat: **nappali** Szak: **programtervező informatikus BSc**

Témavezető neve: **Dr. Illés Zoltán**

munkahelyének neve és címe: **Eötvös Loránd Tudományegyetem Informatikai Kar Média- és Oktatásinformatika Tanszék**

beosztása és iskolai végzettsége: **Egyetemi Docens**

ELTE Informatika Doktori Iskola

A dolgozat címe: **Team Foundation Server -alapú valós idejű nyilvántartás automatizálás**

A dolgozat témája:

A szoftver a TFS(Team Foundation Server) alapú szoftverfejlesztés adminisztratív feladataira nyújt egy automatizált megoldást, az ismétlődő, esetleges havi fejlesztési feladatok legenerálása és előkészítése az adott feladatban szereplő fejlesztőknek, akiknek ezt követően csak az adott feladatra fordított fejlesztési idejüket kell vezetniük. Így az adott csapat vezető fejlesztője akár órákat is megspórolhat, (egyetlen gombnyomás vagy ütemezésnek köszönhetően) amit eddig ezen adminisztráció létrehozására és feltöltésére kellett fordítani. A szoftver emellett a teljes általa legenerált adminisztrációt egy .csv file-ban is elkészíti, esetleges papír alapú / excel alapú másodlagos adminisztrációnak előkészítése érdekében. Szükségszerűen akár emailen is továbbíthatja ezen filet/fileokat.

Automatizálás mellett egyéb, alapvetően kimaradt törlési funkciókat (teljes csapat adminisztráció, feladatok(taskok) és dolgozók) is megvalósít az átláthatatlanság és a felhalmozott feladatok/elévült feladatok elkerülésére. A program akár .csv fileból is képes megvalósítani a szükséges törlési műveleteket, így időt spórolva, lerövidítve a hosszadalmas törlési feladatokat. A program ütemezővel történő konfigurálása esetén teljes mértékben felügyelet nélkül is végrehajtja az időszerű valós időben történő automatizált feladatokat.

TARTALOMJEGYZÉK

BEVEZETÉS

A ma ismert felgyorsult, digitalizált világunkban egyre nagyobb teret hódít magának az automatizálás, adott feladatokban, ismétlődő tevékenységekben, bekövetkezendő hibák visszaszorításában, az emberi teher csökkentésében, és ezáltal a feladatok szoftverek által gépeknek történő átadásában kulcsszerepet tölt be az automatizálás. A szoftver ipar közel minden területén fellelhető ezen automatizálásra történő törekvés. Az egyre nagyobb szoftverek készítésének, tesztelésének, a fejlesztők terheinek csökkentésének egyik kiemelkedő fegyvere lett, amely esetleges későbbi felhasználástól, az adott szoftver termék támogatásának is egyik alapkövét jelentheti és jelenti.

Ezen szakdolgozat „Team Foundation Server - alapú valós idejű nyilvántartás automatizálás” célja is ez előbbiekben említett, az iparban a startupoktól egészen a multinacionális cégekig fellelhető nyilvántartás és fejlesztési ütemezés, tevékenység nyomon követés, és ezen teher csökkentése, akár nullára redukálása, mindamellett a hibafaktor jelentős ismételten közel a nullára való csökkentése azon cégek és csapatok körében, akik a munka és feladatok rendszerezésére és akár a fejlesztés tágabb körében (verziókezelés és akár buildek futtatására is) a Team Foundation Servert használják.

Megemlíthetjük itt az Agilis szoftverfejlesztést és az egyik gyakran használt Scrum szoftverfejlesztés módszerét is, amelyek elsősorban a multinacionális cégeknél töltenek be nagyobb szerepet, de a cég méretétől függetlenül, e módszerek kérdés nélkül segítik magának a terméknek és annak minőségének fejlesztését, valamint a fejlesztőcsapatok nap, mint napi munkájában is pozitív előrelépést jelenthetnek ezen módszerek megfelelő használatai. Itt jön a képbe a nyilvántartás, tevékenység kezelő platform és annak megfelelő hibamentes kezelése, frissen tartása, amely időnként egyfajta „mankóként” segítheti a csapatokat az egyre bonyolultabb/bonyolódó termék előállításában. Példaként, hogy az aktuálisan végzett feladatokon kik dolgoznak, ez a folyamat, hogy áll, mennyi ideje dolgoznak bizonyos problémák megoldásával, mire kellhet több idő, esetleg plusz segítség.

Motiváció

Megkérdőjelezhetetlen tény, hogy ezen nyilvántartó rendszerekre a cég méretétől függetlenül szükség van a megfelelő minőségű munka elvégzéséhez és az adott cég megfelelő működéséhez, szervezeti és szerkezeti struktúrájának egészséges fenntartásához.

Természetesen az ilyen rendszerek frissen tartása, megfelelő kezelése, hogy a fejlesztőknek minél kényelmesebb elérést és a lehető leggyorsabb használatot lehessen biztosítani komoly időt és koncentrált munkafolyamatot kíván. Ez a tevékenység, amelyet akár havi ismétlődésekkel is el kell végeznie az aktuális Team Managernek vagy Project Managernek esetlegesen a Vezető Fejlesztőnek, Vezető Fejlesztőknek jelentős, akár több órás extra munkafolyamatot is jelenthet a fejlesztés és egyéb teendőjük, munkaköri feladataik mellett, (elsősorban ezen feladat a Vezető Fejlesztőre hárul csapatszinten) amely időnként egy-egy termék utolsó „Sprintjében", jelentős extra terhet is jelenthet ezen embereknek, ahol a hibák komoly problémákat, könyvelési gondokat és egyéb, akár anyagi problémák forrásai is lehetnek.

Így a fő cél/ célom ezen fejlesztéssel ezen tevékenység teljes körű automatizálása és minden TFS-t (Team Foundation Server-t) érintő adminisztrációs munkafolyamatot a lehető legegyszerűbbé tenni, amely adott esetben nulla az-az nulla percet vegyen igénybe (a hiba faktor minimálisra csökkentése mellett) az adott embertől, akinek eddig ezen tevékenység is a hónap eleji tevékenységei közé tartozott. Másrészt, egyéb TFS adminisztrációs feladatok elvégzését a lehető legnagyobb mértékben le lehessen egyszerűsíteni pár kattintásra (havi feladatokban esetleges változásoktól/felülírástól a teljes server project formázásig) a TFS hiányosságok pótlásával, elsősorban a törlési és egyéb áttekinthetőségi lehetőségek megteremtésével és egyszerűsítésével, gyorsításával. (Grafikus kezelőfelületen keresztül.) Emellett extra lehetőségeket is igyekszik biztosítani az alkalmazás. Mint szerver operációk dokumentálása, (elsősorban az automatizált tevékenység és ezen eredményei, ember általi törlési tevékenység, archiválás) havi szinten az adott szerveren projektéként történő külső fájlban történő dokumentálás, és ezen automatikus dokumentum tovább küldés a cégen belül további részlegek számára. Ami akár a gazdasági és egyéb csapatok munkáját is felgyorsíthatja, egyszerűbbé teheti, hiszen senkinek sem kell a „másik félre várnia" a munkájának ezen tevékenységi körébe tartozó feladatok elvégzésének terén, hiszen ütemezetten minden alkalommal megadott időpontban történik a folyamat.

Összegezve, a cél ezen szoftver segítségével, az általa biztosított automatizált nyilvántartáskezelés akár havi szinten ismétlődő hibamentes lebonyolítása, és szerveren történő emberi módosítások/beavatkozások egyszerűsítése egy grafikus alkalmazás által nyújtott biztonságos „platformon” keresztül. Adott feladatok külső dokumentálása, és a céges struktúrában való tovább küldése, információ megosztása és hatékonyabb globális munkavégzés céljából, az adatok és tevékenységek archiválása.

FELHASZNÁLÓI DOKUMENTÁCIÓ

Program rövid leírása

Az alkalmazás kizárólag Microsoft Windows operációs rendszeren használható. Ajánlott Microsoft Windows 10 „1903” vagy újabb verzió, „Home” vagy bővebb kiadás.

A „Team Foundation Server – alapú valós idejű nyilvántartás automatizálás” alkalmazás, („TFS_Server Operator”) mind a tejes körű automatizálási lehetőségivel, mind egyéb kiegészítő funkcióival törekszik a hibák minimalizálására, az adminisztratív és egyéb szerveren történő műveletek gyorsabb, hatékonyabb és egyszerűbb megoldásainak biztosítására az adott fejlesztőknek, akik Microsoft tevékenység követő és adminisztratív platformját használják. Team Foundation Server számos más alkalmazás közül (pl. Trello, Jira) a mindennapi munkák során extra lehetőségeivel, (egyedi fejlesztői bővíthetőséggel) központosított, számos funkciót magában foglaló platformként (verziókezelés, nyilvántartás, tesztelés, buildelés), fejlesztői csapatok „bázisaként” funkcionálhat.

Az alkalmazás két elkülönülő irányban nyújt szolgáltatásokat. Elsődlegesen teljes körű automatizálás biztosítása a TFS-t (Team Foundation Server-t) érintő adminisztrációs munkafolyamatok emberi beavatkozás nélküli, havi szinten történő ismétlődő tevékenységeknek, amelyek: feladatok generálása, fejlesztői könyvelés teljes előkészítése, archiválás és jelentés készítése, amely szükség esetén cégen belüli más csapatok/részlegek, alkalmazottak számára való továbbítást is magában foglal egyéb munkafolyamatok elvégzésére a csapatok egymásra való várakozása nélkül. Amely jelentős időbeli nyereség mind a csapatok, csapattagok számára, akiknek korábban havi szinten számos munkaórát emésztett fel ezen adminisztráció kezelése. Másodlagosan egy grafikus felület biztosítása, ahol számos, szerveren történő tevékenységet lehet egyszerűen és gyorsan elvégezni a hiba lehetőségének minimalizálása mellett, ahol lehetőség van az automatizált folyamat felülírására, módosítására, lefuttatására, - a jelzés, hogy automatizált futtatás vagy emberi futtatás történt megoldott minden esetben - az archiválás áttekintésére, logok és korábbi tevékenységek, mind automatizált, mind emberi futtatások áttekintésére, és a hiányzó számos törlési és szerver kondíciós folyamatok elvégzésére a TFS szervereken és adott szerveren található számos csapat projekten külön-külön.

Telepítés

Az alkalmazást telepíteni nem szükséges, azonnal használható verzióként van létrehozva. Az indító fájl „UI_TFS_ServerOperation.exe” (UI_TFS_ServerOperation \bin\Debug) mappaszerkezeten belül található, esetleges kicsomagolás után. Parancsikonnal a felhasználó tetszőleges helyről indíthatja az alkalmazást, amely az előbb említett állományt indítja és futtatja. Ezen lehetőséggel az emberi beavatkozást segítő, könnyítő grafikus alkalmazás nyílik meg lehetőségeivel.

Az automatizálásra, beütemezésre (pl. Windows Scheduler/Feladatütemező) a fent említett helyen a „TFS_ServerOperation.exe” nyújt egyszerű megoldást, amely használata a feladatütemezőben a megfelelő beállításokkal történik, legegyszerűbben egy „.bat” fájl létrehozásával és indításával. (Erről és beállításáról bővebben a „Program automatizált felhasználása” pontban olvashat.)

Program felépítése

Ahogy a korábbi pontokban is említésre került, a program két fő szegmensre és felhasználásra bontható. Elsősorban az ismétlődő havi szintű munka támogatás, ami az adminisztráció és tevékenységkövető feladatok teljes körű elvégzése és jelentésküldése, bármilyen emberi beavatkozás nélkül, amely jelentős idő nyereséget és hibafaktor csökkentést jelent ezen „kényes” műveleteket illetően. Amelyhez a „Telepítés” pontban említett módon egy külön indítófájl és konfigurációs lehetőség nyújt megoldást, így elválasztva a grafikus változattól. Természetesen az automatizálás egy adott TFS szerveren belül számos Team Projektre[1] nyújthat használati lehetőséget, ezen egyedi külön beállítási lehetőségével, amely célja szintén a hibafaktor csökkentése és a problémák elkerülése. Így egy adott szerveren lehetőségünk van beütemezni az összes kívánt Team projectet külön-külön, hála a mindegyik ütemezéshez egyedileg a számára hozzárendelhető egyedi konfigurációnak. (Például: Feladatütemezőben minden Team Projectre külön indítható .bat fájlal hozzárendeljük az indítandó állományhoz a hozzá kívánt beállítási fájlt.) Egy adott szervergépen vagy fejlesztői gépen így lehetőség van egy vagy akár több TFS szerver kívánt Projectjeinek az automatizálására.

[1] Adott TFS szerveren, egy csapat/fejlesztés számára létrehozott „részleg”. Saját adminisztrációs, adatbázis lehetőségekkel.

Emellett nyújt egyéb szolgáltatásokat a grafikus felülete a programcsomagnak. A grafikus felület számos beállítási és konfigurációs feladat elvégzésére nyújt egy egyszerű letisztult könnyen áttekinthető és használható felületet. Itt meg kell említeni a szolgáltatás egyediségét, mivel ezen fejlesztés felhasználói elsősorban az IT iparban dolgozó emberek, így a grafikus felületen keresztüli konfiguráció egy bővebb eszközkészletet, modifikációs lehetőséget biztosít, mellőzve az egyéb megkötéseket megszorításokat, természetesen biztonságos keretek között. Itt elvégzett konfigurációval csak úgy, mint az automatizált futtatások, lefuttathatjuk szabad kézi feltöltés indítással is a módosításainkat a grafikus felületről, amelyet a szerver adminisztrációs, tevékenységkövető pontjaiban szeretnénk eszközölni. Ezen extra módosítások, feltöltések nem befolyásolják az automatizált működést, következő automatizált futtatás már az egyedi emberi feltöltést fogja kezelni és azokat archiválni és létrehozni a következő havi adminisztrációt! (A grafikus felületről indított feltöltés archiválja az automatizált aktuális havi adminisztrációt.) Természetesen minden automatizált és egyéb kézi futtatásokat, mind a logokban mind a szerveren egy egyedi „tageléssel” láthatóvá teszi az alkalmazás! Emellett kapunk különböző felhasználói felületeket, a hiányzó törlési és egyéb szerver állapot kezelési feladatok elvégzésére, archiválás és logok áttekintésére.

Fontos pont, hogy az automatizálás vagy grafikus felületről indított feltöltés esetén, ha valamely adminisztrációs egységet létrehozták magán a szerveren, amit ezen automatizált feltöltésnek kellene, vagy ezen automatizált feltöltést kellene módosítania, (esetlegesen a grafikus felületen keresztül) akkor ezen folyamat nem fog lefutni, mivel jelen esetben valami okból „kézzel” lett létrehozva ember által a szerver felületen, ami nem kerülhet felülírásra!

Program automatizált felhasználása

Az automatizálás az egyik sarkalatos pont. A „Telepítés” pontnak megfelelően, az aktuális ütemezésre használható „TFS_ServerOperation.exe” futtatható állományt használjuk. Mivel egy szerveren több Team Projectet vagy akár több szerveren több Team Projektet is szeretnénk automatizálni, a futtatható állományt egy „/config” kapcsolóval ellátott egyedi konfigurációs fájlal láthatjuk el. Ajánlott .bat fájlok segítségével megoldani az aktuális feladatütemezést, ahol minden egyes létrehozott automatizált futtatásra a saját egyedi Team Projecthez tartozó konfigurációs fájlt állíthatjuk be a

„/config” kapcsolót követően. Így elősegítve a jövőbeli módosítások gyorsaságát, pontosságát a hibalehetőség minimalizálása mellett.

```
TFS_ServerOperation.exe /config TFS_ServerOperation.exe.config
```

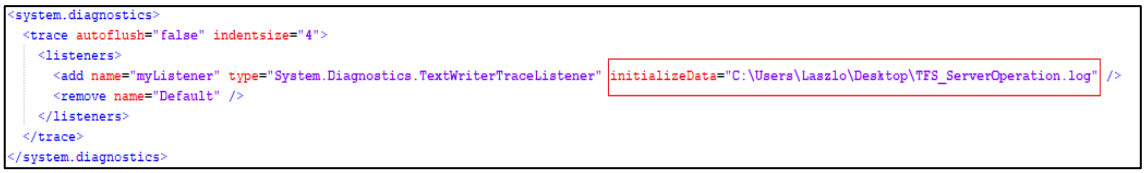
1. ábra
A „.bat” file tartalma.

Az 1. ábrának megfelelő módon beállíthatjuk minden automatizált Team Projektre az adott szervereken, hogy mit és hogyan szeretnénk, ha elvégezne, itt elsősorban a feltöltendő, automatizált kezelésben létrehozandó és archiválást végzendő feladat egységeket, alfeladati kiosztásokat, és jelentés küldési beállításokat, (emailen keresztül milyen Smtp-n keresztül, milyen Porton keresztül) logolási beállításokat, elérési helyeket kell érteni.

Az 1. ábrán láthatjuk, hogy a konfigurációhoz használandó .bat fájlunkban elsősorban, jelen esetben kékkel látható futtatható állományt „TFS_ServerOperation.exe” vesszük fel, ezt követően szóközzel elválasztva a „/config” kapcsoló segítségével megadjuk ismételt szóközt követően az adott futtatható állomány mellé helyezett konfigurációs fájl nevét. (Konfigurációs sablon, üres beállításra váró config fájl az alkalmazás mappájában a futtatható állomány mellett található.)

```
<Connection>
  <add key="TfsCollection" value="http://localhost:8080/tfs/SzakdolgozatCollection" />
  <add key="TeamProjectName" value="Szakdolgozat_Other_Project" />
  <add key="AreaPath" value="Szakdolgozat_Other_Project" />
  <add key="Iteration" value="Szakdolgozat_Other_Project" />
</Connection>
<MailInformation>
  <add key="mail_address" value="wow.laszlo@gmail.com" />
  <add key="smtp_host" value="smtp.gmail.com" />
  <add key="port" value="587" />
</MailInformation>
<PBICollectionSection>
  <PBIS>
    <PBI Title="Month Regular Meeting1 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="2" />
      </Tasks>
    </PBI>
    <PBI Title="Month Regular Meeting2 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="12" />
      </Tasks>
    </PBI>
    <PBI Title="Month Regular Meeting3 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="22" />
      </Tasks>
    </PBI>
  </PBIS>
</PBICollectionSection>
```

2. ábra
Egy beállított automatizált config fájl részlet.



3. ábra
Log fájl beállítás config fájl részlet.

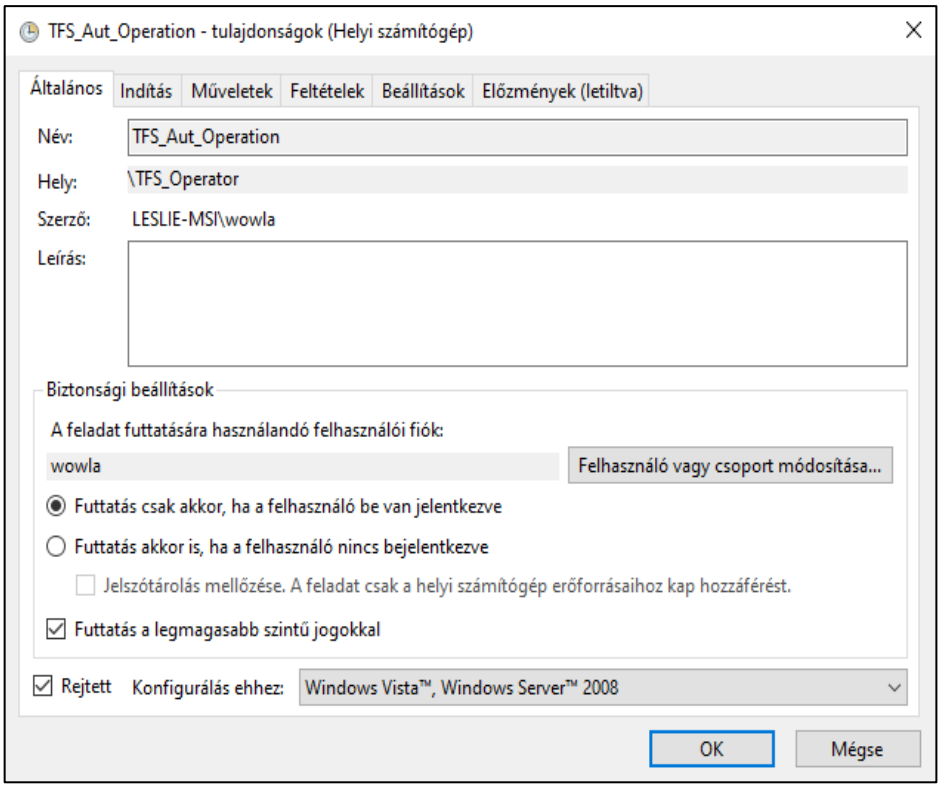
A 2. ábrán láthatunk egy példát egy teljesen bekonfigurált automatizált futtatásra kész fájlról. Észrevehető, hogy három fő beállítási szekciót kell kitöltenünk. Az alkalmazás egyedi XML tagek és szekciókkal segíti a minél könnyebb és átlátható olvasást és beállítást. Elsősorban a „Connection” szekció, ahol az adott szerverhez adjuk meg az eléréshez és kapcsolódáshoz szükséges adatokat, beállításokat. A „TfsCollection” maga a szerver eléréséhez szükséges url. Azt követően az adott szerveren meg kell adni, hogy melyik Team Project nyilvántartását, tevékenységfigyelését szeretnénk innentől automatizálva végrehajthatni havi szinten. Ez a „TeamProjectName” és a hozzá tartozó „AreaPath” és „Iteration”. Következő egység a „MailInformation”. Itt az aktuális automatizálásról való jelentés és általa létrehozott, és módosított feladat egységekről (Később WoktItem[2] ként történik ezen egységekre a hivatkozás.) készült fájlt és értesítést továbbítja a megadott címre, az adott Smtp-n és Porton keresztül. Lehetőség van külső vagy akár cégen belüli Smtp használatára is, ezen szabadságot nem korlátozza az alkalmazás. Majd a „PBICollectionSection” követhetik, ahol PBI – ként (Product Backlog Item idővel User Story[3] – ra történt egy átnevezése) létrehozandó havi ismétléses feladatköröket, munkákat és a feladatkörökhöz tartozó emberek saját feladatainak, munkáinak vezetéséhez szükséges WorkItemeket hozza létre, az ehhez szükséges és felhasználandó időmennyiségek beállításával és aktiválásával. Teljes mértékben emberi beavatkozás nélkül munkára alkalmas és rögtön használható szerver körülményeket hozva létre. Meg kell említeni az egyedi beállítandó mezőket, ezen „PBICollectionSection” – on belül, az adott PBI/User Story- hoz tartozó beállítandó értékek közül a „ParentID” lehet elsőre, nem egyértelmű. Ezen beállítás az aktuális fejlesztendő gyökér feladat „Feature WokrItem” azonosítóját takarja, hogy a szerveren melyik fejlesztési vagy tevékenységi szekcióba tartozik ezen feladatkör, (User Story) ami természetesen további bontásban tartalmazza az aktuális tevékenységi szekción dolgozó embereket (Adott emberhez tartozó Task). Így épül fel egyfajta fa struktúra.

[2] A szerveren lévő munkaegységek átfogó neve. Ez lehet PBI, Task, Bug, TestCase stb.
[3] Egy adott cselekményt/feladatot/munkát magában foglaló egység, amihez az emberek ehhez tartozó feladatai (Taskjai) vannak rendelve, amire végzi az idő könyvelését.

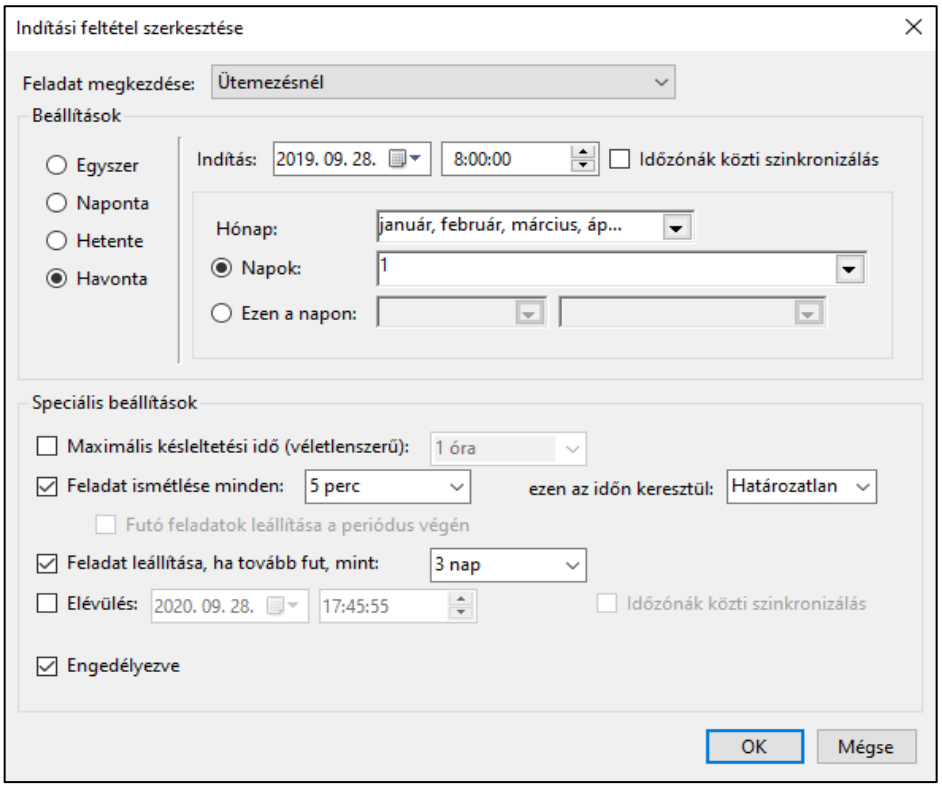
Feature – User Story (PBI) és az adott User Story-n dolgozó ahhoz hozzárendelt dolgozók egyedi feladatai. A többi beállítás elnevezése elég beszédes. „Title” – az aktuális címe, neve a feladatnak, időnként az adott személy neve, aki egy feladaton dolgozik a User Storyn belül. Az „AssignedTo” – a tulajdonosa, végrehajtója a létrehozott feladatkörnek vagy feladatnak az „Effort” ami a Taskoknál (Emberek adott feladatkörön belüli tevékenységeinél szükséges a megadása, de lehet üres is.) maga az idő, amely ezen feladat végrehajtására áll rendelkezésre. Természetesen adott események mellett ezen időt rugalmasan érdemes kezelni!

A 3. ábrán, pedig az elérési utat állíthatjuk be, hogy hova szeretnénk, ha az alkalmazásunk elkészítse futtatásonként a log jelentést. Fontos, hogy minden futtatáskor egyedi időbélyeggel, a logfile nevében ellátott állomány kerül létrehozásra. Így elkerülve egy óriási fájl létrejöttét, ami idővel jelentős tárhelyet is foglalhat, és közel átláthatatlan lehet. Grafikus felületen is lehetőségünk lesz ezen beállításokra, ahol minden egyéb külső fájlt egyszerű módon elérhetünk, így akár az automatizált futtatás esetén is igénybe vehetjük a grafikus állomány ezen ellenőrzési és menedzselési szolgáltatásait!

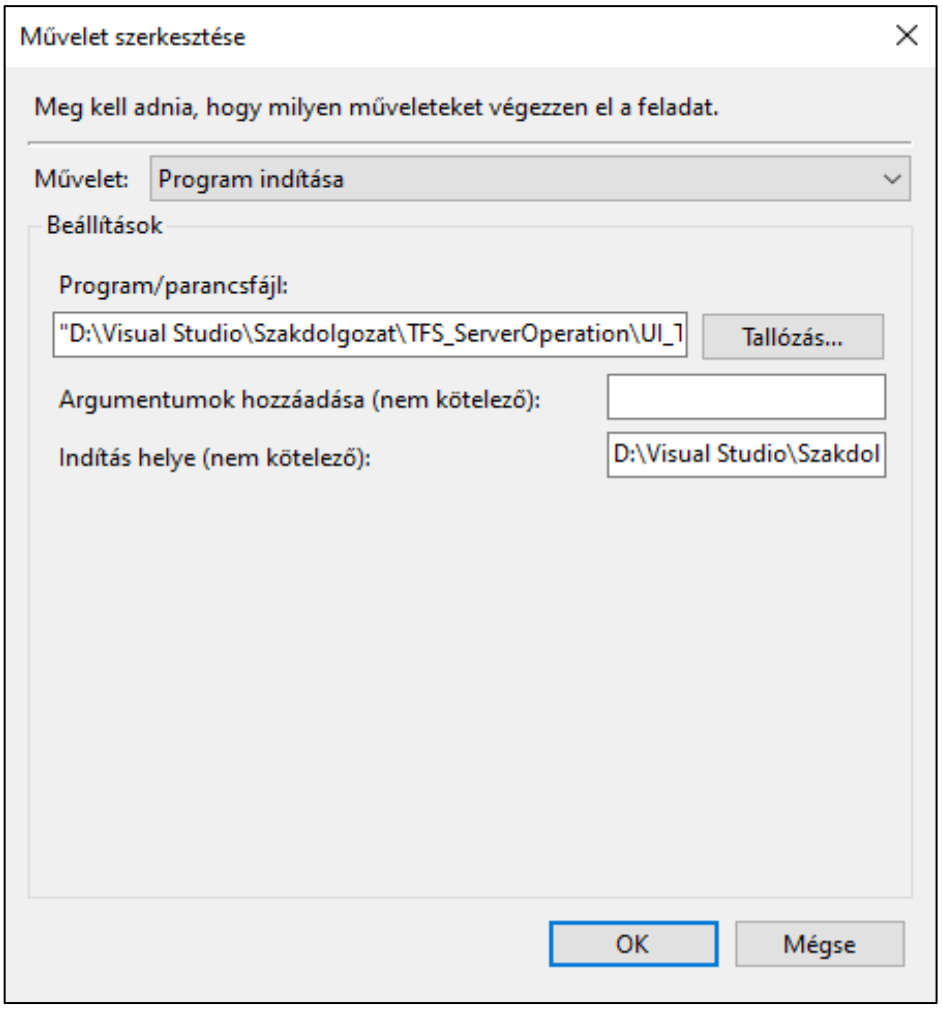
Miután beállítottunk minden szükséges értéket a jövőbeli automatizált futtatásokhoz, és létrehoztuk az 1. ábrának megfelelően a beállításunkat tartalmazó fájlt „/config” kapcsolóval megadtuk a futtatható állománynak a nevet és elmentettük, már csak a Feladatütemezőben/ Windows Scheduler – ben kell létrehoznunk egy automatizált futtatási parancsot. Ahogy korábban említésre került, érdemes minden Team Projecthez külön .bat fájlt és beállításokat tartalmazó konfigurációs fájlt létrehozni a hibák és későbbi átláthatatlanság elkerülése érdekében!



4. ábra
Feladatütemező Új feladat beállítása.



5. ábra
Ütemezési beállítások



6. ábra
Indítási beállítások.

A 4, 5 és 6. ábrának megfelelően elkészített ütemezések innentől havi szinten végrehajtásra kerülnek az operációs rendszer Feladatütemező szolgáltatásának köszönhetően. Ezen a ponton elkészültünk az automatizálás valós idejű ütemezésének teljes körű konfigurálásával, innentől szerverünkön havi szinten megtörténik a feltöltés, aktiválás, archiválás és jelentés küldés, bármi féle beavatkozás szükséglete nélkül!

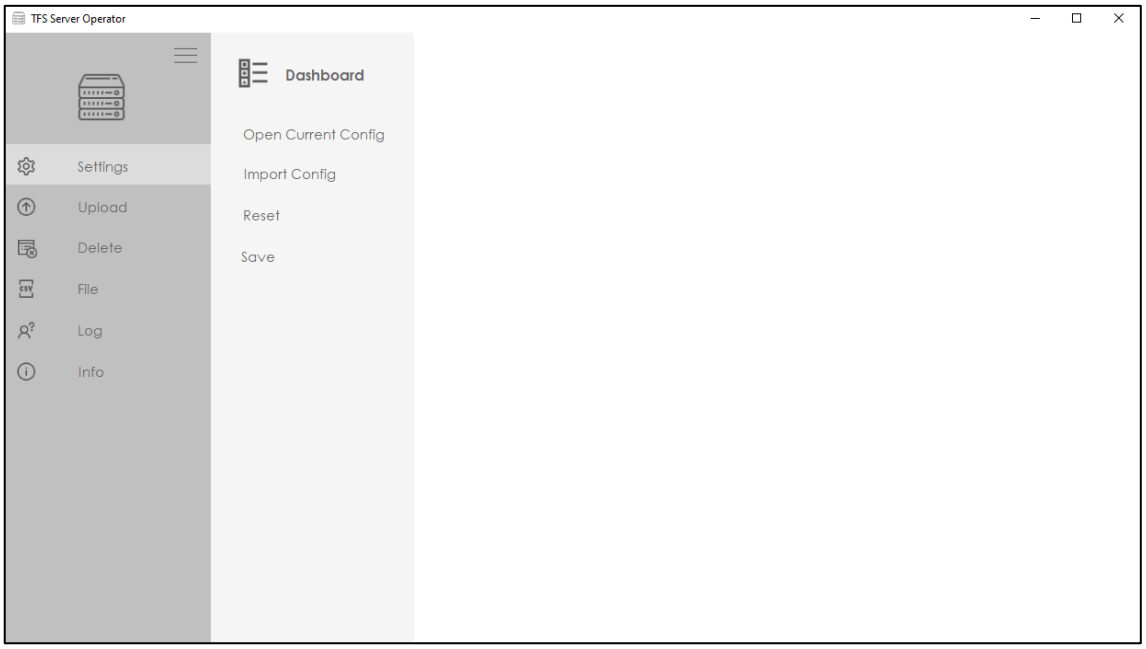
Program grafikus felhasználása

Általános ismertetés

A programcsomag grafikus felhasználói felületen keresztül nyújt lehetőséget a TFS szerverekkel való könnyed és egyszerű interakcióba lépéshez. Ahogy az automatizált és beütemezett adminisztráció és tevékenységkövető előkészítések és generálások történnek, ahhoz hasonlóan lehetőséget biztosít ezen automatizált folyamatok által

létrehozott adatok (User Storyk[4], Taskok) és ezek közötti kapcsolatok felülírására, módosítására szükségszerűen. Emellett nyújt számos hiányzó szolgáltatásra egyszerű és biztonságos lehetőséget, bővítve magának a TFS szervernek a produktív használati funkcióit. A törlési lehetőségek egy vagy több, akár fájlból való nagyobb WorkItem[5] mennyiséget vagy teljes szerver törlésre, resetre nyújtanak könnyű és körültekintő hatékony lehetőséget. Természetesen ezek a grafikus kiegészítő áttekintő interakciók, mindenekelőtt a feltöltéseknek, generálásoknak és egyéb műveleteknek, törléseknek biztosítanak egy egyszerű átfogó nyomon követési és ellenőrzési lehetőséget, amelyek kiterjednek, mind a logok, mind az archivált és aktív fájlokra egyaránt. Az átfogó beállítási és konfigurálási lehetőséget is az automatizálásnál olvasható szabad módon egy külön pontban taglalja, számos extra lehetőséget biztosítva a kényelmes, hibamentes konfiguráció érdekében. Ezen menüpontok és funkciók kifejtésével az adott alcímen belül bővebben is megismerkedhet.

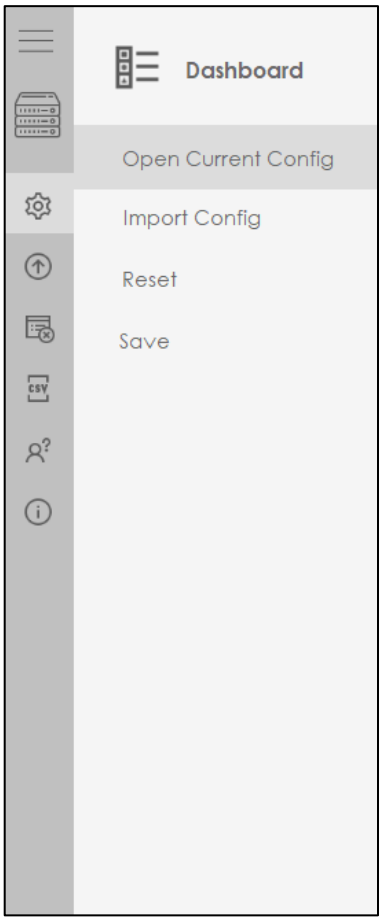
A program indítását („UI_TFS_ServerOperation.exe”) követően, egy egyszerű letisztult, formavilágában, megjelenésében és ikonjaiban is a Microsoft Windows 10 dizájnját követi. A program a kijelző közepén jelenik meg szabadon méretezhető ablak kereteiben.



7. ábra
Indítást követő alkalmazás.

[4] Egy adott cselekményt/feladatot/munkát magában foglaló egység, amihez az emberek ehhez tartozó feladatai (Taskjai) vannak rendelve, amire végzi az idő könyvelését
[5] A szerveren lévő munkaegységek átfogó neve. Ez lehet PBI, Task, Bug, TestCase stb.

A program grafikus felületének bal szélén egy szürke mezőben található a fő menü szekció. Ezen szürke főmenüszakaszban fent középen az alkalmazás ikonjával, és a mező szekció jobb felső sarkában, egy a Microsoft Windows 10 operációs rendszer „Gépház” és egyéb alapértelmezett alkalmazásaiban is népszerű animációs menüfelület összezárás kapcsoló gombja található, amely során az írott menüpont szövegek „bezáródnak” és az ikonok maradnak meg. A fő menüpontokhoz tartozó a főmenü szürke paneljától „bentebb” egy füstös szürke felületen találhatjuk az almenüpontokat és egyéb menüponton belüli szolgáltatások gombjait. (Irányítópult, Dashboard ikonnal és felirattal ellátott terület.) Ezt követő „fehér” felületen, munkafelületen jelennek meg az aktuális menüpontok, almenüpontokban kiválasztott szolgáltatás teljes felhasználóbarát, letisztult dizájnú funkció palettája.



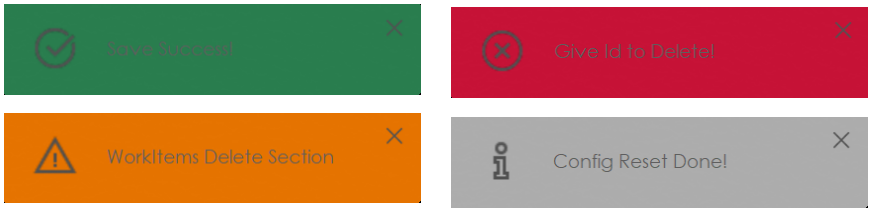
8. ábra
Összezárt főmenüszakasz.

Mind a főmenüben, mind az irányítópult almenüpontjaiban, egyértelműen és jól láthatóan jelzésre kerül, hogy melyik menüpont van aktuálisan kiválasztva.

A program számos alkalommal jól látható, egyértelmű és egyedi értesítésekkel segíti a felhasználók jól informáltságának fenntartását. Például a program legelső indításnál, ahol értesíti a felhasználót, hogy adjon meg egy beállítást, hogy mely szerveren kívánja használni az alkalmazás szolgáltatásait. Az egyedi értesítési rendszer speciális értesítési panelja négy különböző típusú értesítést különböztet meg, mind színben, mind ikonban, mind természetesen az adott helyzethez megfelelő megfogalmazással, amely mindig a legegyértelműbben és célra vezetően igyekszik informálni a felhasználókat. A négy értesítési típus: Siker (Success), Hiba (Error), Figyelmeztetés (Warning) és az Információ (Info). Az értesítés a kijelző jobb felső sarkában jelenik meg fentről animáltan „lecsúszó, leereszkedő” modern értesítési dizájn vonalat követve. Természetesen szabadon mozgatható az értesítési panel, amely megjelenését követően egy elhalványodó animációval tűnik el. Ezen animációs és színeiben élénk értesítési rendszer egyértelműen jelzi a sikeres vagy sikertelen

végkimenetelt, extra információt szolgáltat, vagy éppen felhívja a figyelmet az esetlegesen bekövetkező hibák elkerülésére érdekében.

Minta a négy értesítéstípusra:

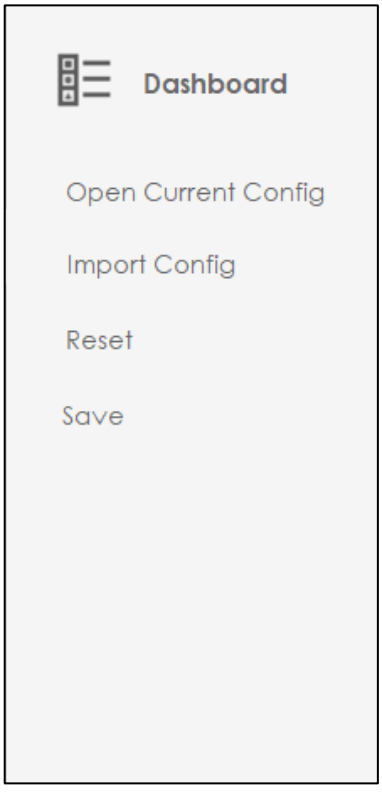


9. ábra
Értesítések, Success, Error, Warning, Info.

Beállítások menüpont

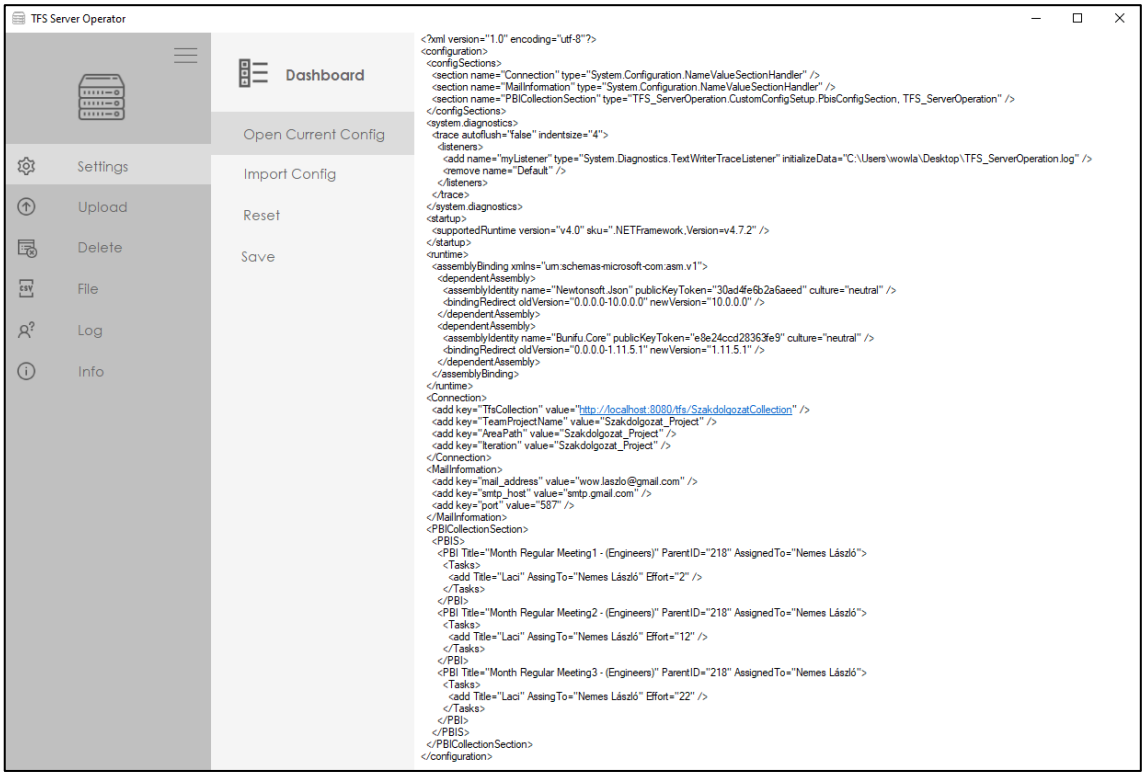
A beállítások (Settings) menüpont az első a főmenüben. Ezen ponton belül nyílik számos lehetőség a különböző konfigurációk elvégzésére. Mint például az adminisztráció és tevékenységkövető WorkItem feltöltés beállításai, különböző szerverekre, vagy szerveren belüli Team Projectekre való csatlakozás és az adminisztrációs jelentésküldés.

A beállítások (Settings) főmenüponthoz tartozó irányítópulton (Dashboard-on) keresztül négy funkció érhető el, az „aktuális config, beállítás megnyitása” - amely a program legelső indítása esetén természetesen konfiguráltalan, és egy értesítésben informálja a felhasználót ezen beállítás elvégzésére. Addig semmiféle szerver kapcsolat nem áll fel, inaktívak a funkciók. A „a külső beállítási fájl beimportálása” – ezen tevékenység hasonló lehet az automatizált beállítási fájlokhoz. Itt lehetőség van ezen külső, esetlegesen más által elkészített vagy korábban használt beállítás egyszerű és azonnali felhasználására az alkalmazáson belül. Plusz a „reset” és a „mentés”



10. ábra
Settings Dashboard szekció.

„Open Current Config” – avagy „aktuális config, beállítás megnyitása**”**. Ezen menüpont nyitja meg nekünk az aktuálisan használatban lévő beállított konfigurációnkat. A korábbi említésnek megfelelően legelső indítás esetén ezen szekció egy beállítatlan állapotot jelenít meg nekünk, amelyről informál is minket az alkalmazás értesítési rendszere, hogy végezzük el a szükséges beállítási műveleteket. Azt követő indítások esetén a korábban beállított adatoknak megfelelően felcsatlakozik az adott TFS szerverre és Team Projectre, úgy várva a további felhasználói beavatkozást. (Hibás beállítás esetén „Inactive” státuszban jelzi az alkalmazás, hogy semmilyen szerver interakciós szolgáltatás nem elérhető!) A grafikus alkalmazás törekszik a minél bővebb és sokszínű beállítási lehetőségek biztosítására, és a teljes konfiguráció átláthatóságára, ezért ezen beállítás az automatizáláskor elvégzendő beállítási szegmenseket tárja elénk, most már az alkalmazáson belüli szerkeszthető felületen keresztül. Meghagyva minden olyan egyéb beállítási lehetőséget, amelyet szinte soha vagy nagyon ritkán speciális esetekben módosítanánk.



11. ábra
Az egyedi XML beállítási felület.

Három fő beállítási szekciót kell kitöltenünk, módosítanunk legtöbb esetben. Az alkalmazás egyedi XML tagek és szekciókkal segíti a minél könnyebb és átlátható olvasást és beállítást. Elsősorban a „Connection” szekció, ahol az adott szerverhez adjuk meg az eléréshez és kapcsolódáshoz szükséges adatokat, beállításokat. A „TfsCollection”

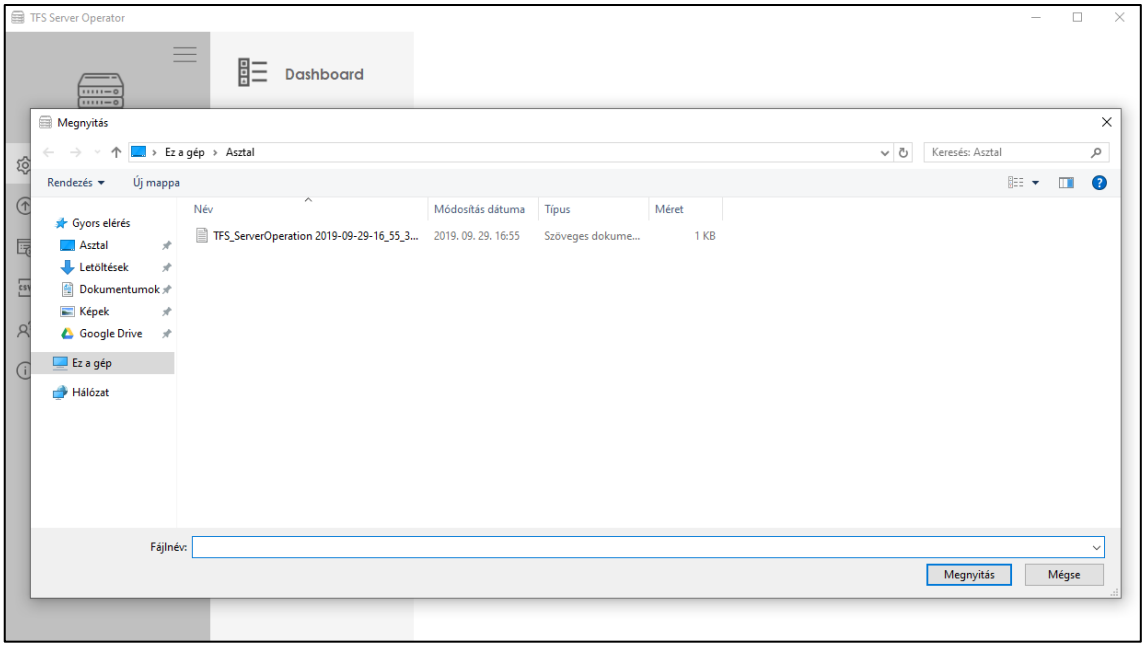
maga a szerver eléréséhez szükséges url. Azt követően az adott szerveren meg kell adni, hogy melyik Team Projectre szeretnénk csatlakozni. Ez a „TeamProjectName” és a hozzá tartozó „AreaPath” és „Iteration”. Következő egység a „MailInformation”. Itt az aktuális beavatkozásról való jelentés és általa létrehozott, és módosított feladat egységekről, WoktItemekről[6] készült fájlt és értesítést továbbítja a megadott címre, az adott Smtp-n és Porton keresztül. Lehetőség van külső vagy akár cégen belüli Smtp használatára is, ezen szabadságot nem korlátozza az alkalmazás. Majd a „PBICollectionSection” következik, ahol PBI – ként (Product Backlog Item idővel User Story[7] – ra történt egy átnevezése) létrehozandó feladatköröket, munkákat és a feladatkörökhöz tartozó emberek saját feladatainak, munkáinak vezetéséhez szükséges WorkItemeket hozza létre feltöltés funkció használata esetén, az ehhez szükséges és felhasználandó időmennyiségek beállításával és aktiválásával Meg kell említeni az egyedi beállítandó mezőket, ezen „PBICollectionSection” – on belül, az adott PBI/User Story- hoz tartozó beállítandó értékek közül a „ParentID” lehet elsőre, nem egyértelmű. Ezen beállítás az aktuális fejlesztendő gyökér feladat „Feature WokrItem” azonosítóját takarja, hogy a szerveren melyik fejlesztési vagy tevékenységi szekcióba tartozik ezen feladatkör, (User Story) ami természetesen további bontásban tartalmazza az aktuális tevékenységi szekción dolgozó embereket (Adott emberhez tartozó Task). Így épül fel egyfajta fa struktúra.

Feature – User Stroy (PBI) és az adott User Sotry-n dolgozó ahhoz hozzárendelt dolgozók egyedi feladatai. A többi beállítás elnevezése elég beszédes. „Title” – az aktuális címe, neve a feladatnak, időnként az adott személy neve, aki egy feladaton dolgozik a User Storyn belül. Az „AssignedTo” – a tulajdonosa, végrehajtója a létrehozott feladatkörnek vagy feladatnak az „Effort” ami a Taskoknál (Emberek adott feladatkörön belüli tevékenységeinél szükséges a megadása, de lehet üres is.) maga az idő, amely ezen feladat végrehajtására áll rendelkezésre. Természetesen adott események mellett ezen időt rugalmasan érdemes kezelni!

Továbbá pedig az elérési utat állíthatjuk be, hogy hova szeretnénk, hogy az alkalmazásunk elkészítse futtatásonként a log jelentést. Fontos, hogy minden futtatáskor egyedi időbélyeggel, a logfile nevében ellátott állomány kerül létrehozásra. Így elkerülve egy óriási fájl létrejöttét, ami idővel jelentős tárhelyet is foglalhat, és közel átláthatatlan lehet. (system.diagnostics – szekcióban.)

[6] A szerveren lévő munkaegységek átfogó neve. Ez lehet PBI, Task, Bug, TestCase stb.
[7] Egy adott cselekményt/feladatot/munkát magában foglaló egység, amihez az emberek ehhez tartozó feladatai (Taskjai) vannak rendelve, amire végzi az idő könyvelését.

„Import Config” – avagy a külső beállítási fájl beimportálása”. Itt van lehetőségünk konkrét külső beállítások egyben történő betöltésére. Egy fájl dialógus ablakon keresztül kiválaszthatjuk az aktuális beállításokat tartalmazó konfigurációs fájlt, és betölthetjük az alkalmazásunknak.



12. ábra
Külső beállítási config fájl betöltés.

„Reset” – „Módosítások visszavonása”. Ez a gomb biztosít számunkra egy „backup” -ot, legyen az egy aktuális beállítás módosítás, vagy egy új beállítás létrehozása, bármikor olyan helyzetben találhatja magát a felhasználó, hogy már nem tudja javítani, amit módosított, nem látja már át a feladatot, amit elkezdett a beállításokban megvalósítani, vagy csak egyszerűen nem találja a hibát, ami miatt a kapcsolatfelvétel, csatlakozás a TFS szerverrel nem következik be. Esetlegesen csak újra szeretné kezdeni a beállítási folyamatot. Ekkor jöhet képbe a „Reset” gomb használata. Az alkalmazás minden indítás esetén az aktuális beállításokat tartalmazó konfigurációs állapotot el cache-eli az operációs rendszer által nyújtott felhasználói temporális mappába, (C:\Users\UserName\AppData\Local\Temp\) így bármikor vissza tudunk állni az „aktuális” kiinduló állapotunkba, mellyel számos problémát, bonyodalmat előzhetünk meg.

13. ábra
Reset végrehajtásának értesítése.

„Save” – avagy „Mentés”. A beállítások konfigurálásának végeztével el kell mentenünk az aktuális módosításainkat, hogy ezt követően ennek megfelelően működjenek az alkalmazás által nyújtott szolgáltatások. Az alkalmazás használata esetén, ha csak betöltjük az „Open Current Config” gombbal az aktuális beállításokat, amellyel indult az alkalmazásunk, természetesen, ha semmi féle módosítást nem végzünk nem szükséges a mentés. Ha módosítunk is de azt nem mentjük el, akkor mint legtöbb esetben a korábbi beállítások maradnak továbbra is életben. Mentések esetén mindig egy új időbélyeggel ellátott log fájl keletkezik a beállításokban megadott elérési helyen. Kiemelendő, ha semmi féle beállítás nincs betöltve a munkafelületre, akkor egy figyelmeztető hibaüzenettel jelzi ezt az alkalmazás a felhasználó felé!

14. ábra
Sikeres mentés értesítés.

15. ábra
Mentés beállítás betöltés nélkül.

Feltöltés menüpont

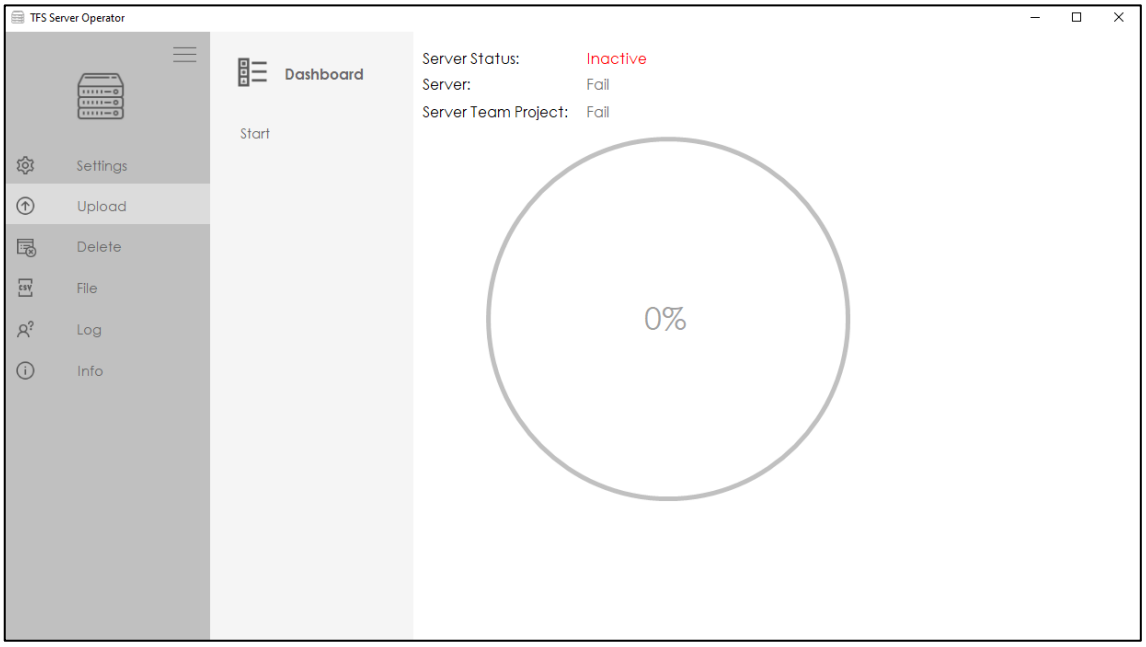
A feltöltés – „Upload” menüpont biztosítja számunkra a saját kézi feltöltés indításának lehetőségét. Így van lehetőségünk az automatizáláson kívüli események egyszerű egy időben történő, hibamentes feltöltésére, létrehozására. (Adminisztráció és tevékenységkövető előkészítések a szerveren, ezek archiválása és dokumentálása egyszeri feltöltés futtatással.) Kiemelendő, hogy ezen feltöltés futtatás biztosítja a lehetőséget az aktuálisan, automatizáltan elvégzett feladatok módosítására. Egyszerűen csak arra az adott TFS szerverre és Team Projectre kell csatlakozunk, aminél szeretnénk ezen automatizált futtatás által létrehozott WorkItemeken elvégezni a módosításainkat. Ezen esetben egy újra feltöltéssel vagyunk kénytelenek operálni, hogy a módosításaink, kiegészítéseink a szerveren megfelelő – aktív állapotban, a korábbi automatizált feltöltést lezárva, archiválva és lejelentve tudjuk megfelelően lekezelt státuszba helyezni. Így az aktuálisan aktív WorkItemek archivált fájlban történő menedzselése, jelentése megfelelő státuszban marad, és a következő hónapban ennek az információknak „tudatában” hajtódhat végre a következő havi adminisztratív automatizált előkészítése a munkának.

(Például: A mi általunk kézzel indított feltöltés az aktív és azokat kell majd archiválnia az automatizált futásnak. Mivel a kézi feltöltésünk archiválta az automatizált futás eredményeit.)

Mindenképp meg kell említenünk azt a tényt, hogy ha egy felhasználó, egy adott WorkItemet, adminisztrációs egységet létrehoz a szerveren, azt sem az automatizált futtatás, sem a grafikus felületen indított feltöltés futtatás nem módosíthat, hiszen az valami okból maga a szerver felületén lett létrehozva egy felhasználó által, amit nem írhatunk felül!

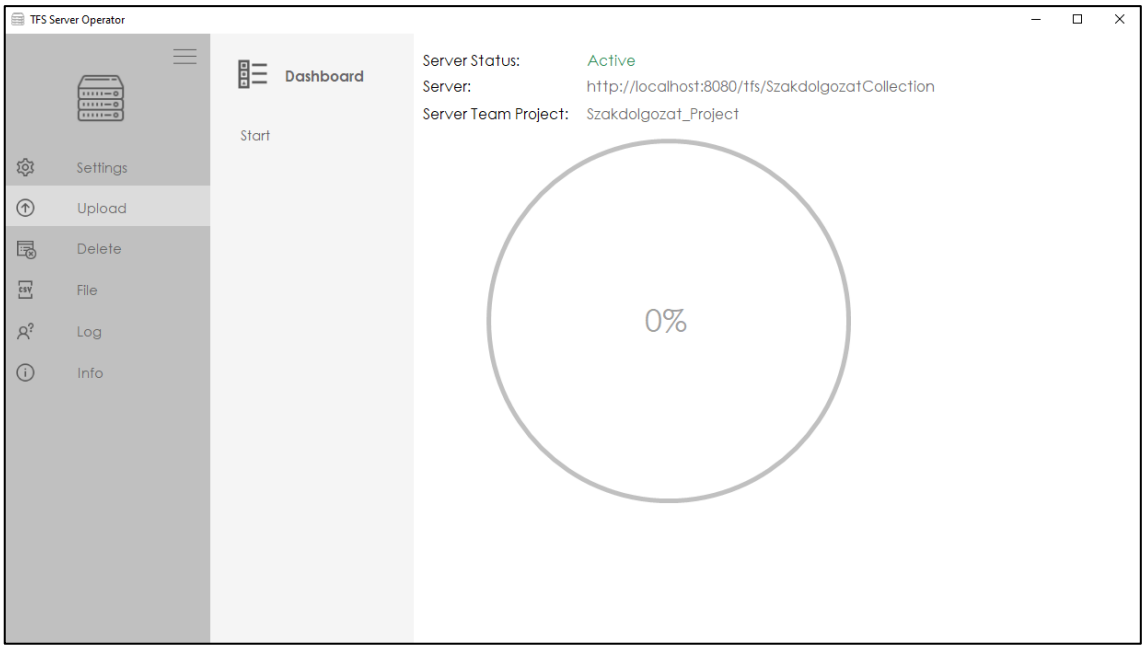
A főmenü „Upload” menüpontjára kattintva, az irányítópulton – „Dashboard” -on egyetlen gomb jelenik meg, maga a feltöltés indítására szolgáló „Start” gomb.

Legelső indítás esetén, és minden olyan esetben amikor a beállítások „Connection” – Kapcsolódás szekciójában szereplő adatok alapján nem képes a kapcsolat felvételére és a csatlakozásra az adott TFS szerverrel és/vagy Team Project-el „Inactive” státuszjelzés fogad miket a feltöltés oldalon. Ekkor a feltöltés mellett a további szolgáltatások is „Inactive”, végrehajtás nem lehetséges státuszban lesznek.



16. ábra
Legelső indítás, vagy hibás TFS szerver url és/vagy Team Project beállítás.

Ahogy a 16. ábrán is láthatjuk, az alkalmazás egyértelműen jelzi a problémát vagy a konfiguráltalan állapotot, gátolva minden nemű károkozást.



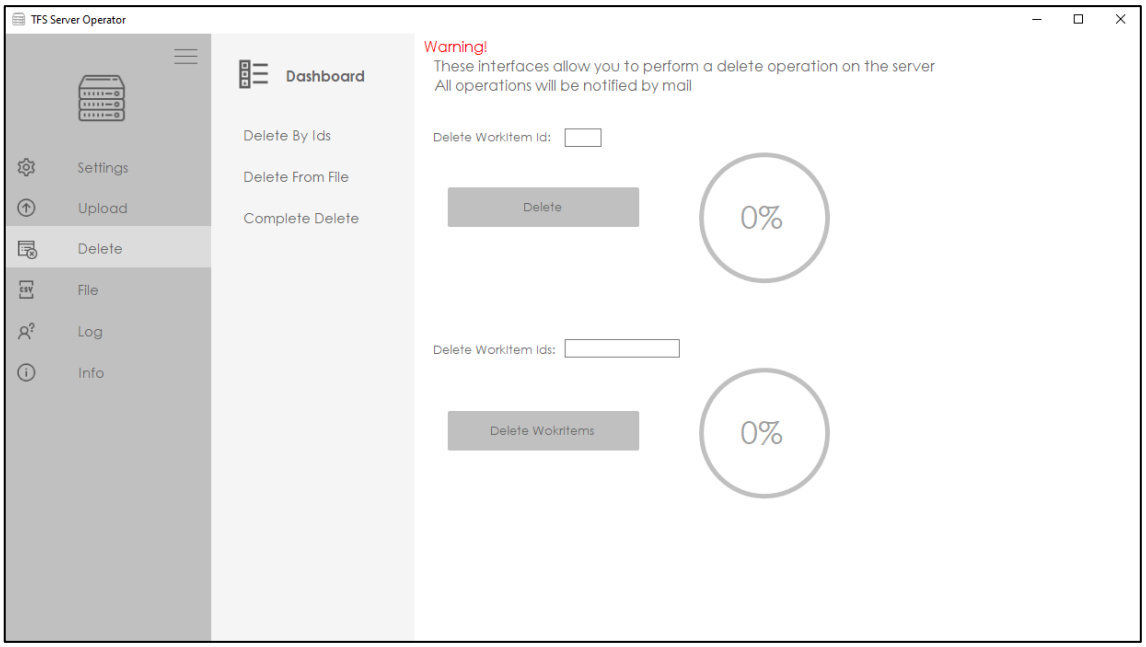
17. ábra
Megfelelő beállításokkal készen a feltöltés indításra.

A 17. ábrának megfelelően, a beállításokat követően, vagy korábban beállított értékeknek megfelelően, a TFS szerverhez sikeresen kapcsolódott, „Active” státuszban van az alkalmazásunk. Lényeges, ha esetlegesen a szerveroldalon lévő problémák, sikertelen kapcsolódás esetén is „Inactive” státuszt vesz fel az alkalmazás.

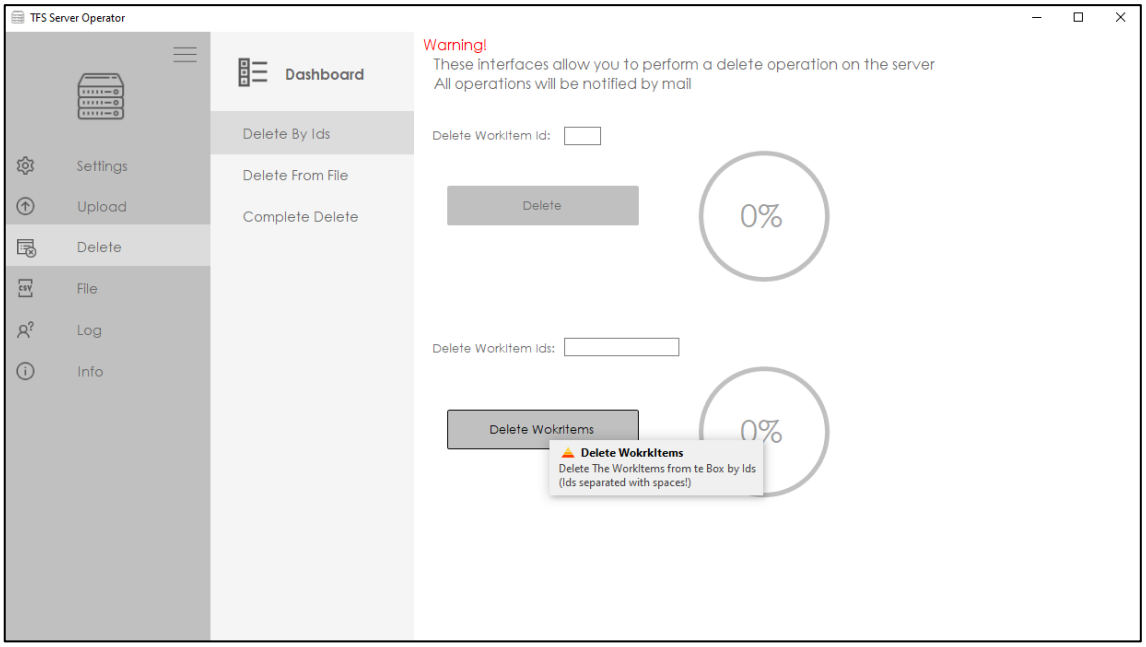
Törlés menüpont

A törlés funkció, ami a TFS 2017 -es verziójáig teljes mértékben hiányzott, mint funkció a szerver kezelő felületéről. Ezt követően is csak egyesével, minden törlendő WokrItemre külön kattintva van lehetőség törlést végezni. Ezen funkciót, és egyéb törlési lehetőségeket, kombinációkat nyújt a menüpontunkba tartozó, irányítópulton (Dashboard-on) keresztül elérhető három funkció.

A „Delete” – Törlés főmenüpontot kiválasztva, az alkalmazás egyedi értesítés rendszere külön, nyomatékosan felhívja a felhasználó figyelmét, hogy ezen szolgáltatásokkal a szerveren törlési műveleteket hajthat végre, és ez fokozott odafigyelést igényel. Természetesen mind a három funkció használata esetén az alkalmazás az irányítópult melletti munkafelületen folyamatosan felhívja a figyelmet az óvatos használatra és hogy minden végrehajtott műveletről jelentés készül, ami továbbításra is kerül, a beállításokban megadott címre! (18. ábra)



18. ábra
A Törlés menüpont munkafelülete. Amely alapértelmezetten a „Delete By Ids” oldalt nyitja meg.



19. ábra
Minden gombra való navigálás esetén extra információs ablak jelenik meg.

„Delete By Ids” - Id (Azonosító) alapján való törlés. Ezen almenüpontban van lehetőség a WoktItemek egyesével való törlésére, mint a szerver kezelőfelületén. Ezt a lehetőséget a felső „blokk” biztosítja a figyelmeztető szöveg alatt. Az „alsó blokkja” az adott almenüpont munkafelületének adja meg a lehetőséget, hogy azonosító alapján, ezen azonosítókat felsorolva, szóközökkel elválasztva egyszerre is törölni tudjuk biztonságosan.

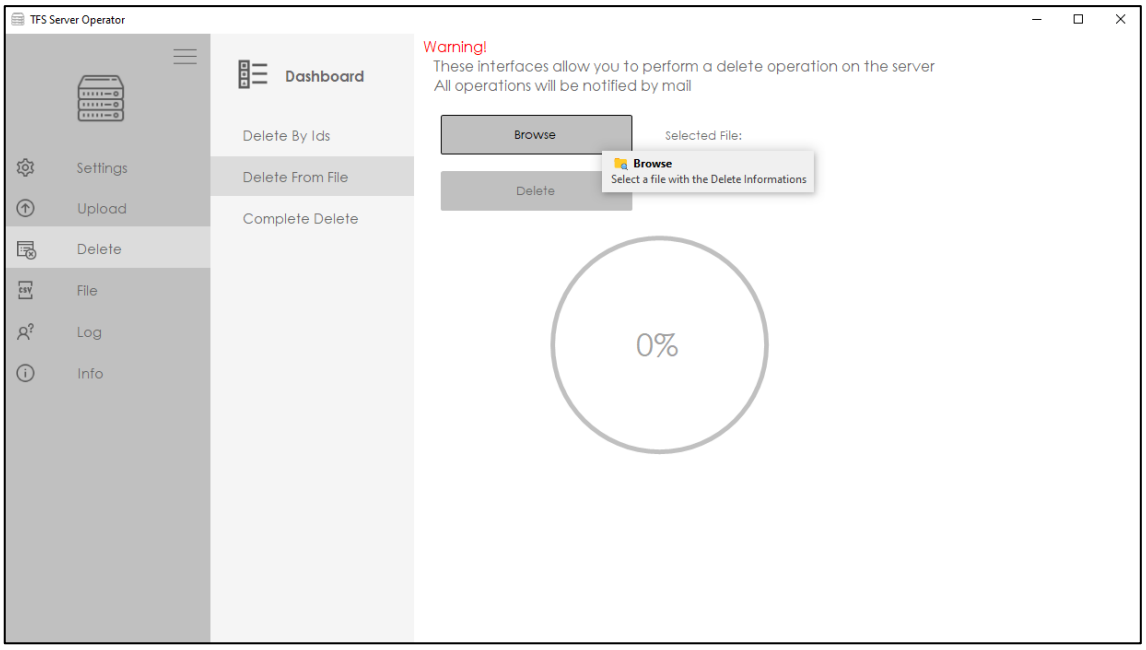
Üres értékmezővel és esetleges „Inactive” szerver kapcsolat esetén természetesen nem következik be törlés és figyelmeztető hibaüzenettel jelez az értesítési rendszer. Hibaesetek bekövetkeztekor erősen ajánlja a log fájl megtekintését is. (Több érték beírása az egy WorkItem törlésére lehetőséget adó mezőbe, és a több WorkItem törlés mezőjébe való egyetlen érték beírása esetén nem történik végrehajtás, figyelmeztető hibaüzenetet kapunk!)

Fontos kiemelendő tény, hogy esetleges olyan törlés indításakor amikor az azonosítóval hivatkozott WorkItem nem létezik a szerveren, a grafikus felületen a folyamat és a törlési kísérlet végbe megy, és egy extra log fájlba jegyzi ezen folyamat végkimenetelét! Mivel a lehető legbiztonságosabb és leghatékonyabb módon végezzük a törlést egy Microsoftos .dll fájl által nyújtott lehetőséggel, hogy garantáltan minden függőség megszűnjön és egyetlen WoktItembe se keletkezzen olyan hiba, amely menthetetlenségét okozná, kénytelenek vagyunk minden alkalommal ezen törlési folyamatot, kísérletet elindítani!

„Delete From File” – Fájlból való törlés. Nagyobb volumenű törlések esetén vagy esetlegesen mások által elkészített fájlok, jelentések alapján is van lehetőségünk, hogy időt és energiát spórolva ezen törlésre szánt WorkItemek egy listáját (fájlt) megadjuk alkalmazásunknak és elvégezzük a törést egyszerre. Megkötések szintjén annyi elvárása van a szoftvernek, hogy az automatizált vagy a grafikus felület feltöltő műveletének megfelelően az általa készített jelentésnek megfelelően formázásban az adott WorkItemek azonosítóival kezdődjenek a sorok a fájlunkban. Ezen rendszerezés mind a szoftvernek mind a későbbi adminisztratív munkák esetén egy sablonként szolgálhat az egyszerű és könnyű, egységes jelentések mindenki számára érthető formában való kezelésére.

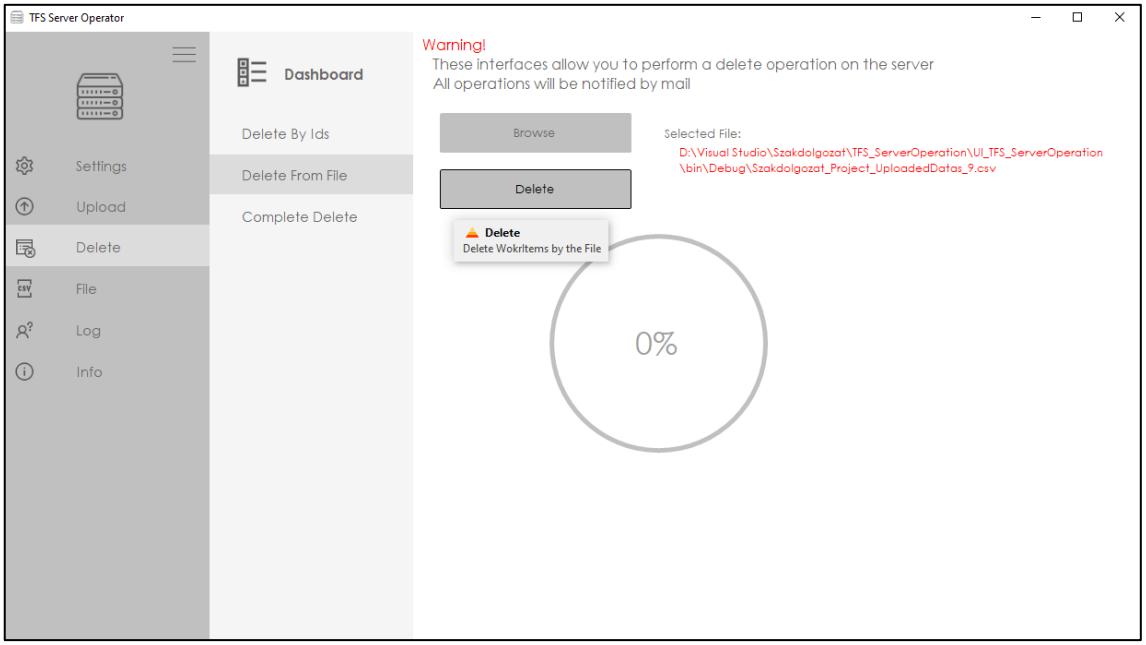
Hibás fájl esetén értesítést kapunk, hogy nem megfelelő a kiválasztott fájlunk, és a törlés nem végrehajtható a tartalmazott adatai alapján!

Ha a fájl olyan értékeket tartalmaz, amely nem létezik a szerveren, de emellett van, vannak olyan értékek, amelyek igen és törölhetőnek lennének, nem fognak törlődni! Biztonsági okokból, ha egy érték nem törölhető akkor a többi sem fog törlődni.



20. ábra
Fájlból törlés felülete, extra gomb információval, mivel a kurzorunkat rámanővereztük.

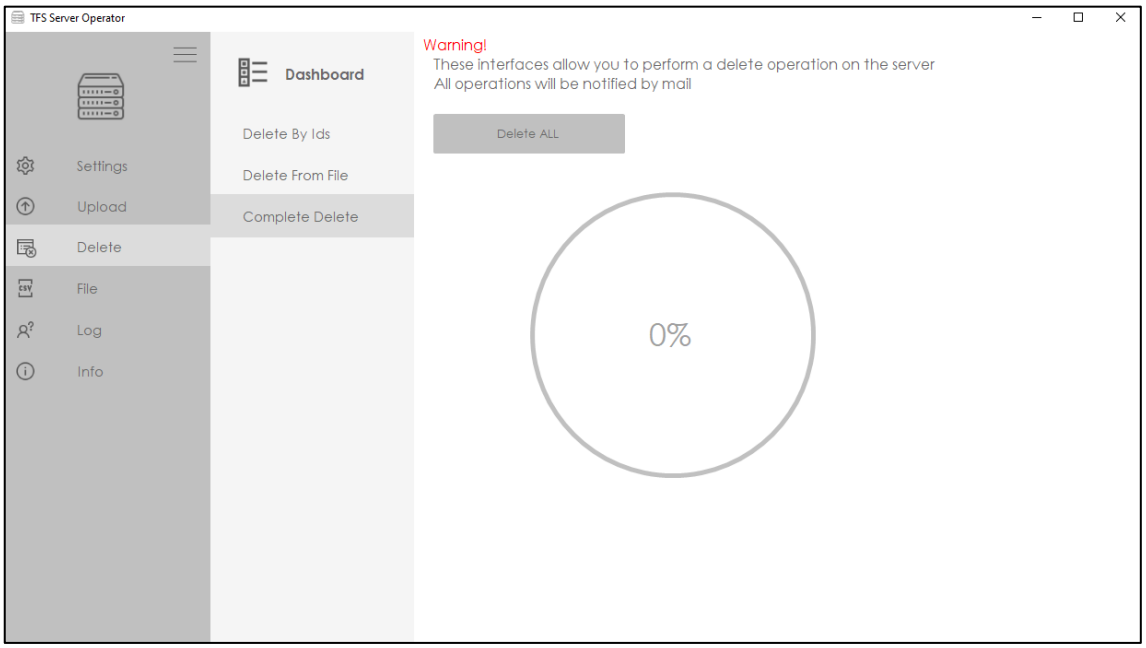
A 21. ábrának megfelelően, a kiválasztott fájl esetén az adott fájl nevét és annak elérési útját is láthatjuk, a „Selected File:” – Kiválasztott fájl szekciónál, így megbizonyosodva, melyik fájl van megadva a törlési folyamatnak.



21. ábra
Kiválasztott fájl elérése látható, törlésre készülve.

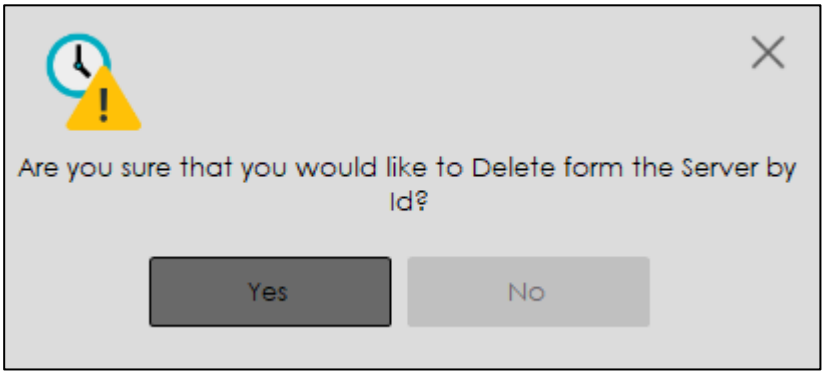
„Complete Delete” – Teljes Szerver Törlés. Az alkalmazás lehetőséget biztosít egy szerver tartalmának teljes törlésére. Ez számos lehetőséget biztosíthat egy adott cégnek, csapatnak, hiszen egy nagy projekt számára külön fenntartott TFS szervert és ezen nagy projekt összes WorkItemét egyszerre törölhetjük, így újra felhasználhatóvá

téve az adott szervert. Másik felhasználási lehetősége lehet ezen funkciónak, ha az adott csapat saját TFS szerverét a munka befejeztével teljesen resetelheti, frissé teheti a jövőbeli munkák számára, így fenntartva a könnyed és egyszerű áttekintést a sok régi használaton kívüli WorkItem jelenléte, zavaró hatása nélkül. Természetesen mind az automatizált és mind a grafikus feltöltő szolgáltatás archiválása és jelentés küldése miatt, minden korábbi projektről, szerver tartalomról van egy havi bontásban lévő áttekinthető archivált jelentéscsokor.



22. ábra
Teljes törlési felület.

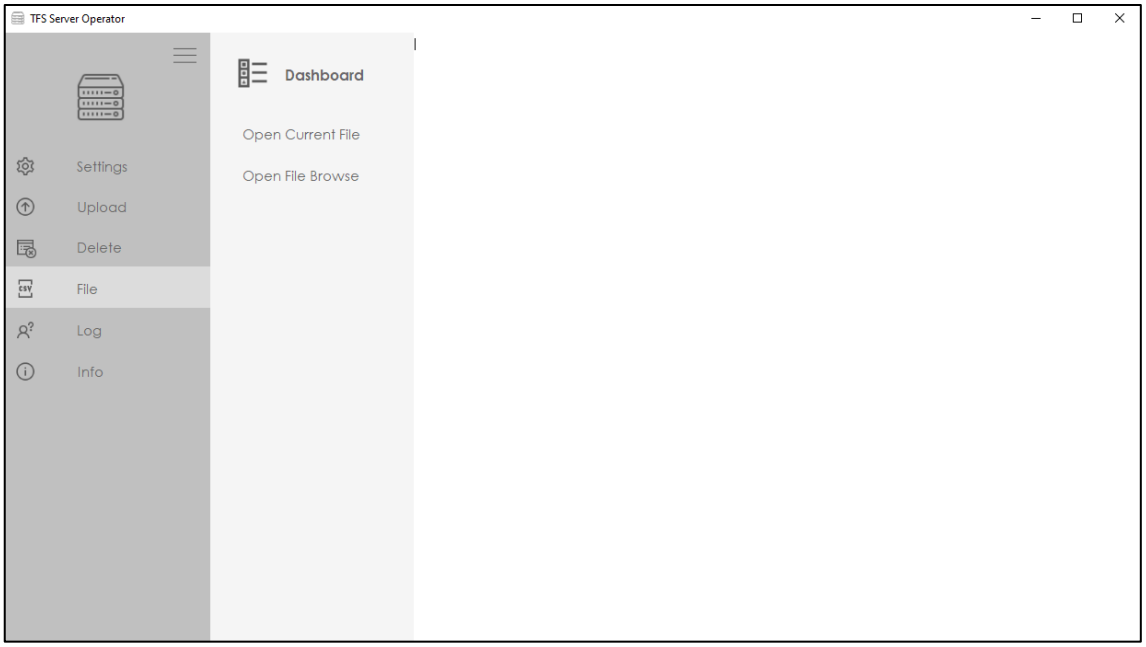
Minden esetben a törlés tényleges végrehajtása előtt van egy felbukkanó információ doboz, amely a folyamat futtatásának még egyszeri egy és egyben utolsó megerősítését kéri. Itt még van alkalmunk visszavonni a folyamatot.



23. ábra
Utolsó megerősítést kérő Message Box – Információ Doboz.

Fájl menüpont

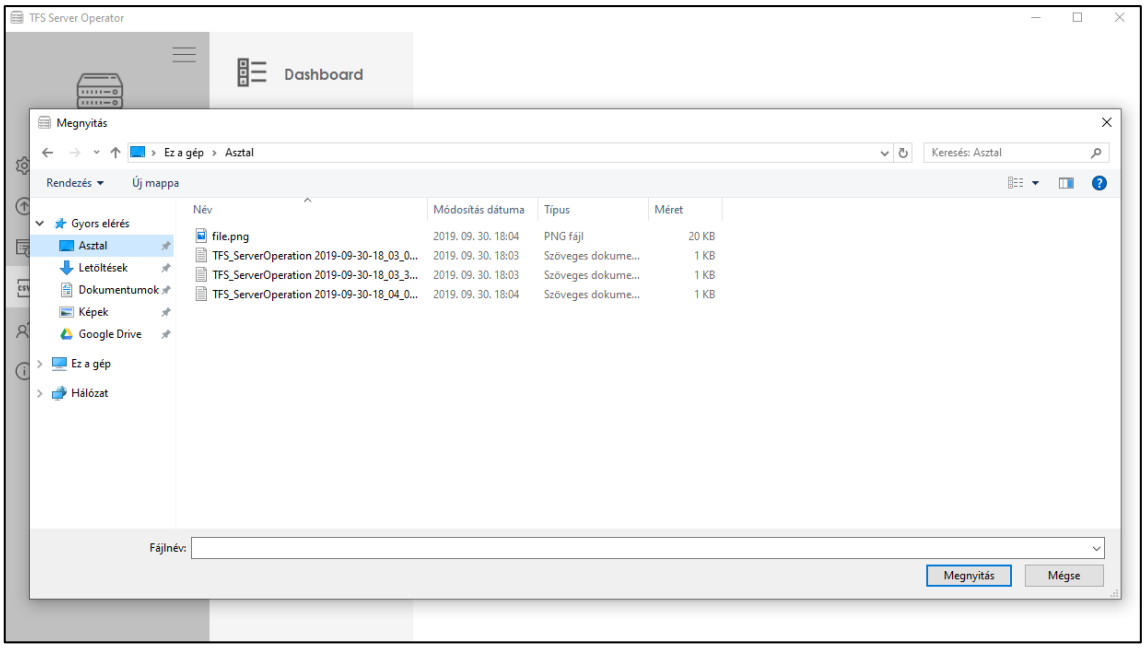
A fájl menüpont biztosítja számunkra az automatizált és a grafikus felületen indított feltöltések, módosítások által létrehozott vagy módosított fájlok egyszerű, egy vagy két kattintással való elérését. Természetesen számtalan módon elérhetjük, megkereshetjük a generált fájlokat (amelyek az alkalmazás fő mappájába kerülnek), de talán ez a legegyszerűbb és legkönnyebb módja az elérésnek, főleg, ha a grafikus felületről indítottunk feltöltést, szerver tartalom módosítást.



24. ábra
File menüpont.

24. ábrán látható módon a „File” menüpont két almenüpontba foglalt lehetőséget biztosít. Első opció az „Open Current File” – Aktuális fájl megnyitása, ha az adott TFS szerver adott Team Projektjén nem történt még automatizált vagy grafikus felületről indított feltöltés a hónapban, amely művelet létrehozott volna egy fájlt az adott WorkItem feltöltésről, akkor egy értesítés formájában kapunk információt, hogy jelen esetben nem tud betölteni semmit sem. (Adott havi fájlt.)

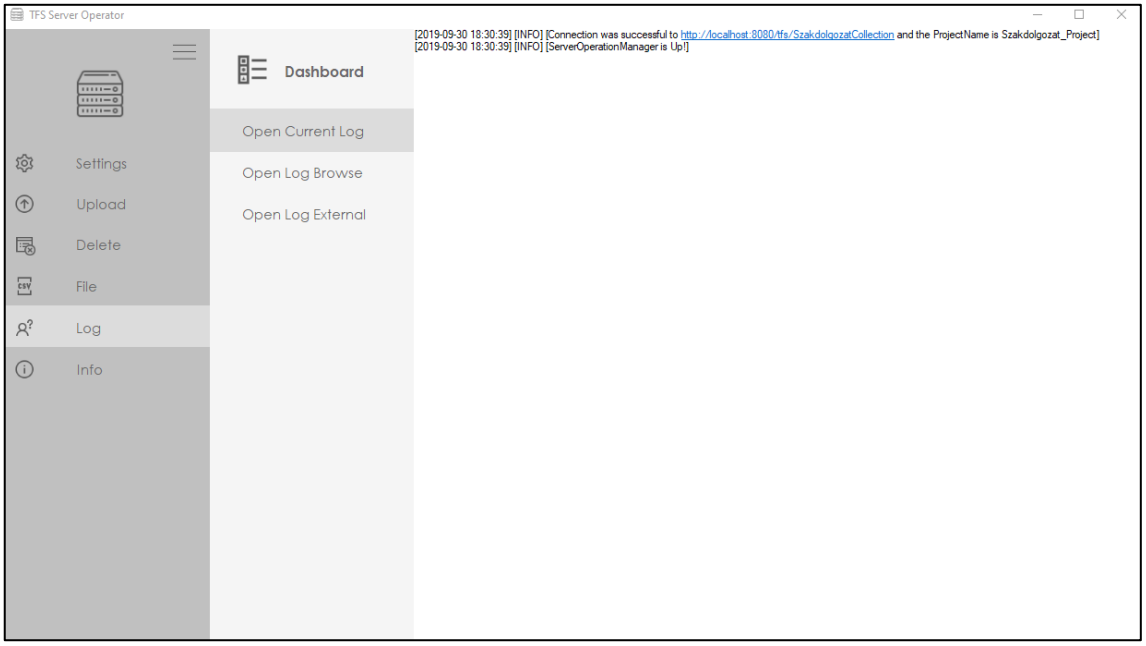
A másik opció itt a „Open File Browse” – Külső fájl megnyitás, amely során kikereshetjük az éppen megnyitni kívánt fájlunkat és az alkalmazáson belül megnyithatjuk. Példaként, ez egy tökéletes lehetőség korábbi havi fájlok gyors megnyitására, áttekintésére. (25. ábra)



25. ábra
„Open File Browse” alkalmazása.

Log menüpont

Ezen fülön belül van lehetőségünk áttekinteni az aktuális és esetleges korábbi log fájljainkat. Ahogy korábban is említésre került, az alkalmazás minden futtatás esetén a beállításokban megadott névvel, a megadott helyen hozza létre a futtatáshoz tartozó log fájlt, a nevében egy idő bélyeget is elhelyezve az egyértelmű megkülönböztetés céljából. Így elkerülve egy óriási átláthatatlan, sok helyet foglaló log fájl létrejöttét.



26. ábra
Log menüpont, „Open Current Log”

A log menüponton belül három szolgáltatás van biztosítva. Első sorban az aktuális éppen elindított alkalmazásunk pillanatnyi log fájljának megnyitása, az „Open Current Log”. Ezen menüpont használatával megnyithatjuk, vagy egy szolgáltatás használatát követően frissíthetjük az alkalmazáson belül is, hogy ellenőrizhessük, hogy mi történt az adott szolgáltatás futtatásával, futtatása közben.

„Open Log Browse” – Külső log megnyitás. Csak úgy, mint a „File” menüpont esetén, lehetőséget biztosítunk külső, esetleges korábbi logfájlok alkalmazáson belüli megnyitására, betöltésére, így a kényelmet és a gyorsaságot támogatva, egyszerűen tudjuk ellenőrizni nem csak az aktuális, hanem bármely korábbi vagy más logfájljainkat. Az aktuális futás logfájlját nincs lehetőség betölteni, erről egy figyelmeztetésen keresztül tájékoztatja az alkalmazás az aktuális felhasználót, hogy ez a fájl az aktuális „futtatásé”, és használja az „Open Current Log” almenüpont lehetőségét.

27. ábra
Aktuális log külső betöltés esetén lévő figyelmeztetés.

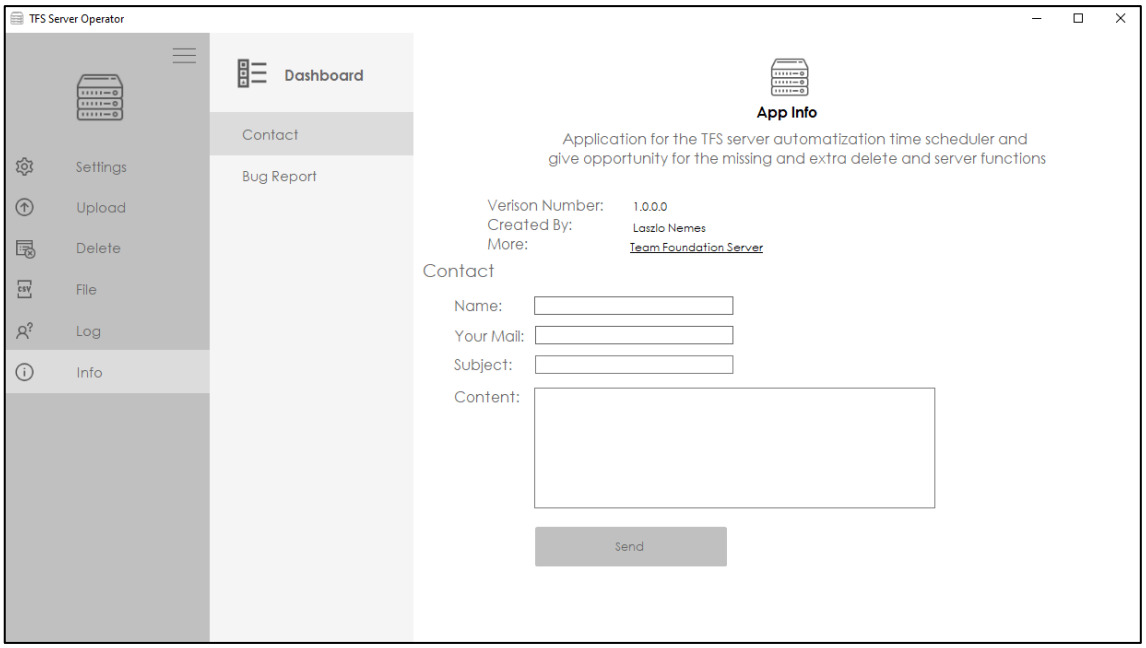
„Open Log External” – Log külső alkalmazásban való megnyitása. Lehetőség van az aktuális log fájl nem az alkalmazáson belüli, hanem egy külső Notepadban – Jegyzettömbben való megnyitására, esetleges speciális ok, okok miatt. Egy kattintás és az alkalmazás „felé” megnyíló jegyzettömbben láthatjuk aktuális fájlunk tartalmát.

Info menüpont

Utolsó menüpontunk a listán, az Info. Ezen fülön belül kaphatunk egy általános összefoglaló pár mondatból álló leírást az alkalmazásunkról. Plusz pár, egyéb általános információt, mint például az aktuálisan telepített és használt applikáció verziószáma, a fejlesztője és egy hivatkozás, link, amely egyfajta a „témáról bővebben mind felhasználói mind fejlesztői szemmel” kiegészítő dokumentáció.

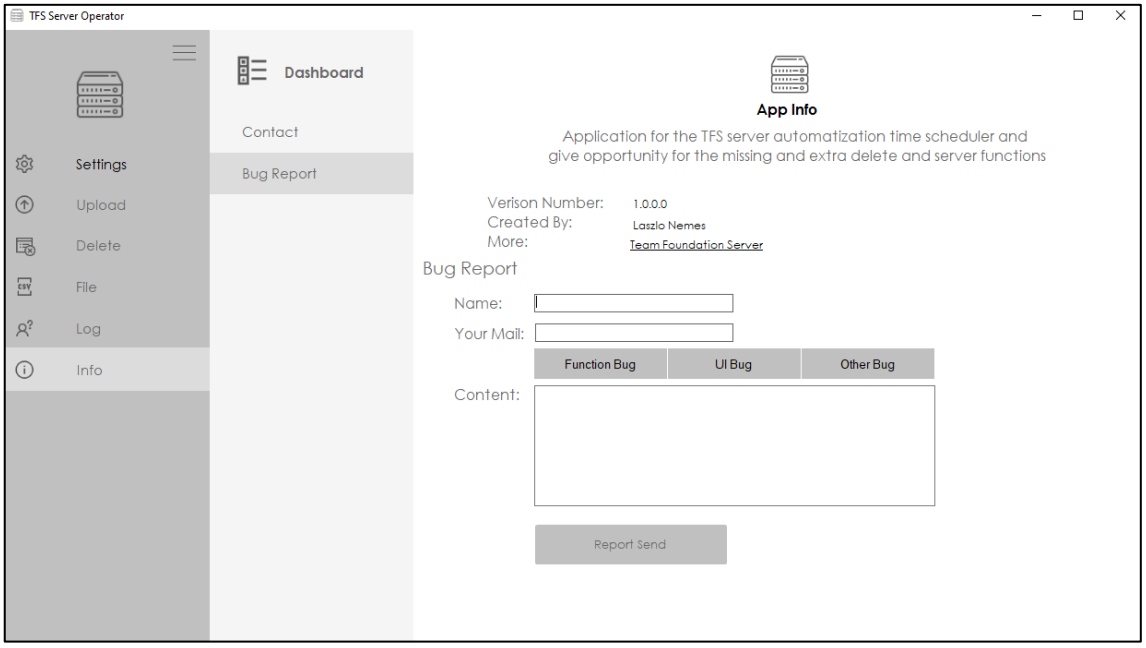
Két funkciót nyújt ezen felül a menüpont számunkra. Egy kapcsolatfelvétel – „Contact” és egy Hiba bejelentő – „Bug Report” felületet.

A „Contact” – Kapcsolatfelvétel során lehetőségünk van az alkalmazáson belül üzenetet írni a fejlesztőnek, fejlesztőknek, szerviz szolgálatot ellátó embernek, csapatnak. Itt leírhatjuk esetleges véleményünket vagy feltehetjük akár specifikus az alkalmazással kapcsolatos kérdéseinket is.



28. ábra
Info menüpont, alapértelmezetten a „Contact” almenüponttal.

A másik almenüpontunk a „Bug Report” – Hibabejelentés. Ezen keresztül informálhatjuk a fejlesztőt, fejlesztő csapatot az esetleges „Side Effectekről” és problémákról, amit az alkalmazás használata közben tapasztalhattak sajnálatos módon.



29. ábra
Info menüpont, „Bug Report”.

A 28 és 29-es ábrákon észrevehető, hogy a „Contact” és a „Bug Report” esetén a küldendő üzenet tárgyának megadása jelentősen eltér. A „Bug Report” esetén három alternatíva van megadva, amelyből kell választani így jobban kategorizálható a beérkezendő hibajegy.

FEJLESZTŐI DOKUMENTÁCIÓ

Program bővebb ismertetése

Átfogó leírás

A „Team Foundation Server – alapú valós idejű nyilvántartás automatizálás” alkalmazás, („TFS_Server Operator”) mind a tejes körű automatizálási lehetőségivel, mind egyéb kiegészítő funkcióival törekszik a hibák minimalizálására, az adminisztratív és egyéb szerveren történő műveletek gyorsabb, hatékonyabb és egyszerűbb megoldásainak biztosítására az adott fejlesztőknek, akik Microsoft tevékenység követő és adminisztratív platformját használják. Team Foundation Server számos más alkalmazás közül (pl. Trello, Jira) a mindennapi munkák során extra lehetőségeivel, (egyedi fejlesztői bővíthetőséggel) központosított, számos funkciót magában foglaló platformként (verziókezelés, nyilvántartás, tesztelés, buildelés), fejlesztői csapatok „bázisaként” funkcionálhat.

A program két fő szegmensre és felhasználásra bontható. Elsősorban az ismétlődő havi szintű munka támogatás, ami az adminisztráció és tevékenységkövető feladatok teljes körű elvégzése és jelentésküldése, bármilyen emberi beavatkozás nélkül, amely jelentős idő nyereséget és hibafaktor csökkenést jelent ezen „kényes” műveleteket illetően. Emellett nyújt egyéb szolgáltatásokat a grafikus felülete a programcsomagnak. A grafikus felület számos beállítási és konfigurációs feladat elvégzésére nyújt egy egyszerű letisztult könnyen áttekinthető és használható felületet.

Az alkalmazás kizárólag Microsoft Windows operációs rendszeren használható. Ajánlott Microsoft Windows 10 „1903” vagy újabb verzió, „Home” vagy bővebb kiadás.

Ezek alapján a programunkat az MVC (Modell – Nézet - Vezérlő) architektúrájának megfelelően három fő részre bonthatjuk, és ezeket a pontokat bontjuk tovább, főként a Modell szegmensét, mint connection, szerveren végrehajtandó műveletek, biztosítás, hogy külön Feladatütemezőből automatizáltan is kényelmesen, egyszerűen lehessen futtatni, műveletek összefogása egy vezérlőben így megteremtve az egyszerűen modifikálható grafikus felületeket stb.

Az alkalmazás kettőssége, maga az automatizált valós idejű feladat elvégzés, és a grafikus felületen keresztüli műveleteknek köszönhetően a további fejlesztési és

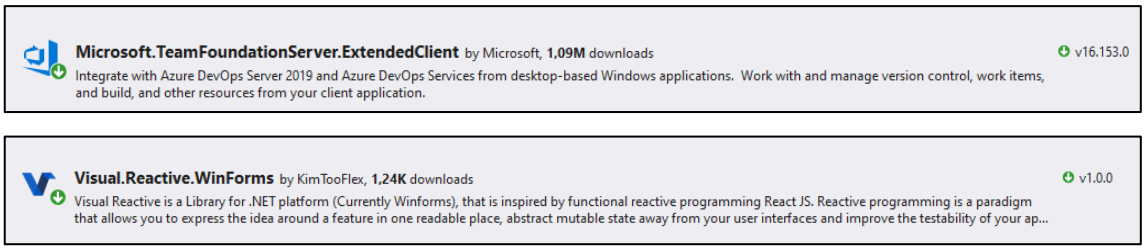
kiegészítési lehetőségek a jövőbeli, mind Microsoft (reagálás a Microsoft frissítésekre, ha szükséges) mind esetleges cég specifikus módosítások, fejlesztések előtt nyitva hagyja a kaput, így könnyen és egyszerűen fejleszthető minden téren alkalmazásunk bármely funkciója. A kulcs az újra felhasználhatóság és korlátozottság mentes módosítási lehetőség. A továbbiakban ezen MVC architektúrának megfelelően végig követhetjük a Modell összes funkciójának a mivoltát és továbbfejlesztéshez szükséges esetleges eszközeit, a Vezérlő felhasználhatóságát, a Modell által nyújtott funkciók használatának biztosítását, hogy is van lehetőség az architektúrának megfelelően a Vezérlőn keresztül biztonságosan és célravezetően alkalmazni a szolgáltatásokat, és a Nézet „panelos" teljes mértékben módosítható cserélhető, leválasztható felépítését.

Fejlesztői eszközök

Maga az alkalmazás C# .Net Framework (v4.7.2) eszközeivel készült különböző NuGet package kiegészítéseket igénybe véve. A fejlesztő környezet Visual Studio 2017 Community Edition és Git verziókezelő menedzsmenttel.

Az alkalmazás egy Microsoft által fejlesztett platform (TFS Server) kiegészítése, amely megvalósításához a C# és a .Net Framework lehetőségei kiváló környezetet biztosítanak a fejlesztés kivitelezéséhez, ezáltal a Microsoft univerzumában maradva. Számos olyan implementáció valósult meg ezen kiegészítésben, amely más környezetben kivitelezhetetlen lehetett volna, a szükséges Microsoft támogatás nélkül a különböző package -ek, library -k és .dll -ek által nyújtott lehetőségek tényében.

Így a további fejlesztéshez szükséges eszközök a fejlesztő környezet mellett (Visual Studio ajánlott) a .Net Framework v4.7.2 -es verziója, amely Visual Studio esetén a telepítéskor kiválasztható, mint opció, más esetben a Microsoft weboldaláról ajánlott letölteni az adott dotnet verziót. A szükséges NuGet packagek pedig a „Microsoft.TeamFoundationServer.ExtendedClient" aktuális legfrissebb verziója, ezen package, számos egymással függőségben lévő NuGet packaget tartalmaz. Ezen csomag biztosítja a kommunikációt a szerverrel és számos TFS specifikus Microsoft által fejlesztett osztályt biztosít, amely nélkül a fejlesztés kivitelezhetetlen lenne. A másik NuGet package a grafikus megfejelés fejlesztéséhez biztosít kényelmes és egyszerű váltásokat a menüpontok és almenüpontok között, így fenntartva a „panelos, minden elem könnyedén cserélhető" elvet, és magában az implementációban is modern egyedi megoldásokat biztosít, ez pedig a „Visual.Reactive.Winforms" package.



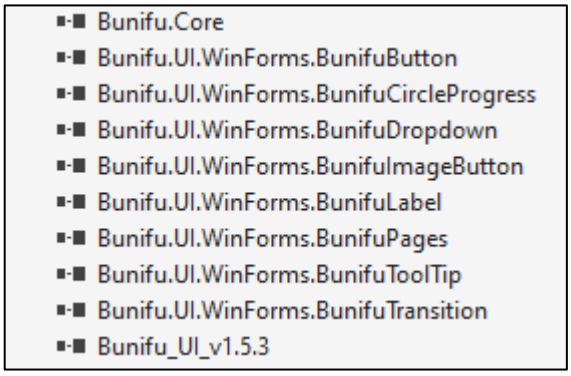
30. ábra
NuGet package -ek.

Természetesen a mellékleten szereplő alkalmazás ezen packageket tartalmazza, és a „packages.config” fájl alapján letölti magának a szükséges NuGet packageket!

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Ben.Demystifier" version="0.1.3" targetFramework="net472" />
  <package id="Microsoft.AspNet.WebApi.Client" version="5.2.3" targetFramework="net472" />
  <package id="Microsoft.AspNet.WebApi.Core" version="5.2.3" targetFramework="net472" />
  <package id="Microsoft.IdentityModel.Clients.ActiveDirectory" version="3.19.4" targetFramework="net472" />
  <package id="Microsoft.IdentityModel.JsonWebTokens" version="5.2.4" targetFramework="net472" />
  <package id="Microsoft.IdentityModel.Logging" version="5.2.4" targetFramework="net472" />
  <package id="Microsoft.IdentityModel.Tokens" version="5.2.4" targetFramework="net472" />
  <package id="Microsoft.TeamFoundation.DistributedTask.Common.Contracts" version="16.153.0" targetFramework="net472" />
  <package id="Microsoft.TeamFoundationServer.Client" version="16.153.0" targetFramework="net472" />
  <package id="Microsoft.TeamFoundationServer.ExtendedClient" version="16.153.0" targetFramework="net472" />
  <package id="Microsoft.VisualStudio.Services.Client" version="16.153.0" targetFramework="net472" />
  <package id="Microsoft.VisualStudio.Services.InteractiveClient" version="16.153.0" targetFramework="net472" />
  <package id="Newtonsoft.Json" version="10.0.3" targetFramework="net472" />
  <package id="System.Collections.Immutable" version="1.4.0" targetFramework="net472" />
  <package id="System.IdentityModel.Tokens.Jwt" version="5.2.4" targetFramework="net472" />
  <package id="System.Reflection.Metadata" version="1.5.0" targetFramework="net472" />
  <package id="Visual.Reactive.WinForms" version="1.0.0" targetFramework="net472" />
</packages>
```

31. ábra
A grafikus projekt „packages.config” fájlja

Emellett mindenképp meg kell említeni a grafikus felület fejlesztéséhez használt egyéb referenciákat, .dll fájlokat. A fejlesztéshez felhasználásra kerültek a „Bunifu Framework” letisztult és modern kiegészítéseket tartalmazó szolgáltatása.



32. ábra
Grafikus fejlesztéshez használt Framework.

Ezen .dll fájl referenciákat sajnos „kézzel” kell hozzárendelni, a bereferált .dll fájloknak megadni az aktuális elérési helyüket a mellékleten található alkalmazás esetén. (Első fejlesztési indításnál szükséges művelet!) Ezen .dll fájlok a mellékleten megtalálhatóak.

Emellett sajnálatos módon a Team Foundation Server és Azure Devops Server fejlesztési dokumentációk Microsoft részről alig fellelhetőek, azok is nagyon kevés néha 1-2 mondatot tartalmazó leírások, amelyek gyakran nem elegendőek a fejlesztés kivitelezéséhez, plusz online, egyéb forrásokból sincs túl sok témába vágó információ, így a különböző konfigurációk mivoltát kénytelenek vagyunk az aktuálisan általunk használt TFS Szerver Process Template – eiből előkeresni. Melléklet tartalmazza a fejlesztés során használt Process Templateket és a dokumentáció Modell szükséges részeinél ki is tér azon referenciákra és ismertekre.

Összességében az alkalmazás egy fejlesztő csomag részeként lehetne elképzelni. Minden fejlesztő megkapja az adott csomagot a fejlesztés és egyéb munkáját érintő alap alkalmazásokkal, mint a fejlesztő környezet, verziókezelő, teszt menedzser és jelen esetünkben a TFS szerver adminisztrátor konzol alkalmazás, és a TFS_ServerOperator alkalmazásunk, amellyel egy teljes csokrot kaphatunk, sok-sok kényelmet és produktivitást támogató lehetőséggel.

Kiegészítő fogalmak

A fejlesztői dokumentáció további olvasása előtt érdemes lehet pár fogalmat részletesebben taglalni, amelyről a felhasználói dokumentációban lábjegyzetben történt meg a szöveg bizonyos részeinek értelmezéshez szükséges definiálás.

Elsősorban maga a WokrItem. Egy munkaelem, projekt elem, amely jelzi a munka típusát, az adott elvégzendő tennivalókat, és megadja a kitűzött célt. Az alapvető munkaegységek a következők lehetnek: User Story – korábban PBI (Product Backlog Item), Task, Issue és Epic.

User Story – korábban PBI. Az Agilis szoftverfejlesztéshez tartozó fejlesztési módszertan, ahol a termék minden olyan tulajdonságát, amely hozzáadott értéket jelent az ügyfél számára, felismerik, prioritássá teszik és fejlesztik az ügyfél igényei alapján.

A Task. Fejlesztőknek elvégzendő, szétosztott megoldandó feladatok pontjai. A User Story több alpontra bontva.

Emellett megemlíthetjük még a Bug – Hibákat, amellyel a TFS szerveren jelezhetjük a problémákat, amelyek megoldásra várnak, és természetesen a Feature-k, amelyekbe szervezhetjük a fejlesztéseinket, ezek nagyobb egységek, amelyeket szétbontunk User storykra azon belül pedig Taskokra, így felépítve egy „fát”. Ezen egységek – WorkItemek között úgynevezett RelatedLink -ek összefüggő kapcsolatok vannak.

Ezen WorkItemek felépítést, mezőit, értékeit mind-mind a Process templatekből tudjuk kiolvasni, és így felépíteni őket, hogy megfelelő „validált” és menthető WoktItemeket kaphassunk.

WIQL. A TFS szervereken Work Item Query Language (WIQL) lekérdezési nyelv, adatbázis található.

Specifikáció

Az adott alkalmazásnak két területen kell helyt állnia. Elsősorban az automatizált WorkItem létrehozás és feltöltés kezelés. Minimum elvárásnak tekinthető, hogy megvalósítson egy automatizált feltöltési rendszert, amely emberi beavatkozás nélkül képes feltöltést végezni havi rendszerességgel, miközben fájlban vezeti az általa létrehozott és feltöltött elemeket, és szükség esetén az előző havi futás eredményét archiválja, ha futott előző hónapban. Emellett egy grafikus felületen keresztül egy teljes különálló alkalmazásként egyszerűsít bizonyos szerver interakciókat. Grafikus felülettel szembeni minimum elvárásnak tekinthetjük, hogy a hiányzó törlés funkciókra adjon megoldást egy vagy több elem egy idejű törlésének megvalósítására és biztosítson egy feltöltési és módosítási lehetőséget, amely képes pótolni, módosítani az automatizált futtatást emberi indítással. Mindeközben egységesíti a log és aktuális és archivált file kezeléseket, áttekintéseket.

Automatizálás vagy grafikus felületről indított feltöltés esetén, ha valamely adminisztrációs egységet létrehozták magán a szerveren, amit ezen automatizált feltöltésnek kellene, vagy ezen automatizált feltöltést kellene módosítania, (esetlegesen a grafikus felületen keresztül) akkor ezen folyamat nem fog lefutni, mivel jelen esetben valami okból „kézzel” lett létrehozva ember által a szerver felületen, ami nem kerülhet felülírásra!

Beállítások terén egy konfigurációs fájl (App.config) alkalmazása biztosítja a legalaposabb beállítási lehetőségek kivitelezését. A beállítás átláthatóságának és egyszerűsítésének érdekében egy egyedileg írt XML „tagelési” rendszer vezeti végig a felhasználót a beállítások legfontosabb szekcióin. Ezen beállítások módosulnak a legtöbb esetben, de nyitva marad a lehetőség minden egyéb beállítás megváltoztatására is, de előre láthatóan nagyon specifikus és rendkívüli esemény idézheti elő a többi szekcióban a változtatás szükségességét. Ezen XML alapú konfiguráció biztosítja a különböző

beállítások széles körű felhasználását, specifikusságát és továbbadását. Biztosítva, hogy akár más ember által készített konfigurációt is importálhatunk az éppen a saját gépünkön lévő alkalmazásban, mind grafikus, mind automatizált futtatási környezetben.

Három fő beállítási szekciót kell kitöltenünk. Az alkalmazás egyedi XML tagek és szekciókkal segíti a minél könnyebb és átlátható olvasást és beállítást. Elsősorban a „Connection" szekció, ahol az adott szerverhez adjuk meg az eléréshez és kapcsolódáshoz szükséges adatokat, beállításokat. A „TfsCollection" maga a szerver eléréséhez szükséges url. Azt követően az adott szerveren meg kell adni, hogy melyik Team Project nyilvántartását, tevékenységfigyelését szeretnénk innentől automatizálva végrehajthatni havi szinten vagy feltöltést indítani a grafikus felületről. Ez a „TeamProjectName" és a hozzá tartozó „AreaPath" és „Iteration". Következő egység a „MailInformation". Itt az aktuális automatizálásról való jelentés és az általa létrehozott, és módosított WorkItemekről készült fájlt és értesítést továbbítja a megadott címre, az adott Smtp-n és Porton keresztül. Lehetőség van külső vagy akár cégen belüli Smtp használatára is, ezen szabadságot nem korlátozza az alkalmazás. Majd a „PBICollectionSection" követhetik, ahol User Story, (PBI) – ként létrehozandó havi ismétléses feladatköröket, munkákat és a feladatkörökhöz tartozó emberek saját feladatainak, munkáinak vezetéséhez szükséges WorkItemeket hozza létre, az ehhez szükséges és felhasználandó időmennyiségek beállításával és aktiválásával. Teljes mértékben emberi beavatkozás nélkül munkára alkalmas és rögtön használható szerver körülményeket hozva létre vagy a grafikus feltöltés indításával módosítja azt, minden esetben teljesen munkára alkalmas helyzetet fenntartva. Meg kell említeni az egyedi beállítandó mezőket, ezen „PBICollectionSection" – on belül, az adott PBI/User Story- hoz tartozó beállítandó értékek közül elsősorban a „Title" – az aktuális címe, neve a User Sory -nak. Ezt követi a „ParentID". Ezen beállítás az aktuális fejlesztendő gyökér feladat „Feature WokrItem" azonosítóját takarja, hogy a TFS szerveren melyik fejlesztési vagy tevékenységi szekcióba tartozik ezen feladatkör, (User Story) ami természetesen további bontásban tartalmazza az aktuális tevékenységi szekción dolgozó embereket. (Adott emberhez tartozó Task). Így épül fel egyfajta fa struktúra. Utolsó beállítandó érték a User Storyban pedig az „AssignedTo" – a tulajdonosa, végrehajtója a létrehozott feladatkörnek vagy feladatnak.

A fa struktúra így, Feature – User Stroy (PBI) és az adott User Sotry-n dolgozó ahhoz hozzárendelt dolgozók egyedi feladatai. A Taskok felépítése, elsősorban a „Title" – az aktuális címe, neve a feladatnak, időnként az adott személy neve, aki egy feladaton

dolgozik a User Storyn belül. Az „AssignedTo” – a tulajdonosa, végrehajtója a létrehozott feladatkörnek vagy feladatnak és az „Effort” ami az emberek adott feladatkörön belüli tevékenységeinél szükséges a megadása, de lehet üres is. Ez maga az idő, amely ezen feladat végrehajtására áll rendelkezésre.

```
<Connection>
  <add key="TfsCollection" value="http://localhost:8080/tfs/SzakdolgozatCollection" />
  <add key="TeamProjectName" value="Szakdolgozat_Other_Project" />
  <add key="AreaPath" value="Szakdolgozat_Other_Project" />
  <add key="Iteration" value="Szakdolgozat_Other_Project" />
</Connection>
<MailInformation>
  <add key="mail_address" value="wow.laszlo@gmail.com" />
  <add key="smtp_host" value="smtp.gmail.com" />
  <add key="port" value="587" />
</MailInformation>
<PBICollectionSection>
  <PBIS>
    <PBI Title="Month Regular Meeting1 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="2" />
      </Tasks>
    </PBI>
    <PBI Title="Month Regular Meeting2 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="12" />
      </Tasks>
    </PBI>
    <PBI Title="Month Regular Meeting3 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="22" />
      </Tasks>
    </PBI>
  </PBIS>
</PBICollectionSection>
```

33. ábra
A beállítási szekciói az App.confignak.

Automatizált, Ütemezhető szolgáltatás

Az automatizált feltöltés kezelése egy sarkalatos pont. Minimum követelményként biztosítani kell, hogy emberi beavatkozás nélkül havi rendszereséggel automatizáltan létrehozásra és feltöltésre kerüljenek a szükséges munkát támogató WorkItemek – User Storyk és Taskok, amelyek megfelelő Feature -ökhöz vannak linkelve és természetesen aktívak, azonnal használatra készek. Az automatizálás ütemezéséhez egy külön a Feladatütemezőbe használandó „TFS_ServerOperation.exe” futtatható állományt biztosítunk. Mivel egy szerveren több Team Projectet, vagy akár több szerveren több Team Projektet is szeretnénk automatizálni, a futtatható állományt ezért a hibafaktor minimalizálása és az egyszerű felhasználóbarát áttekinthetőség miatt, ezen ütemezések Feladatütemezőbe való felvételéhez minden Team Projecthez egyedileg megadható, /config kapcsoló után egy az aktuális beállításokkal ellátott konfigurációs fájl. Így könnyen cserélhető és áttekinthető rendszert hozva létre a Feladatütemezőn belül.

Célszerű ezért egy .bat fájlban felvenni az adott futtatható állományt a /config kapcsolót és az adott beállításokat tartalmazó fájlt.

Archiválás és fájlkezelés esetén az aktuális hónap számával ellátott fájlok létrehozása biztosítja, hogy az adott havi műveletek eredményei hosszú távon elérhetőek maradjanak, és hónap első napján automatizált futás esetén is ezen fájl tartalmát felhasználja, mint az előző havi archiválásra váró WorkItemek listája. Így a grafikus felület feltöltési, módosítási lehetősége egy ember általi indítás, így felülírja az aktuálisan az automatizálás által létrehozott WorkItemeket, mind a TFS szerveren, mind a fájlban. Így a grafikus felület műveleti eredményei (WorkItemei) lesznek az új aktuális havi értékek, amit vagy ismételt emberi, vagy automatizált futás archivál, automatikus futtatás esetén következő hónap első napján. (Mind a TFS szerveren mind a fájlkezelésben.)

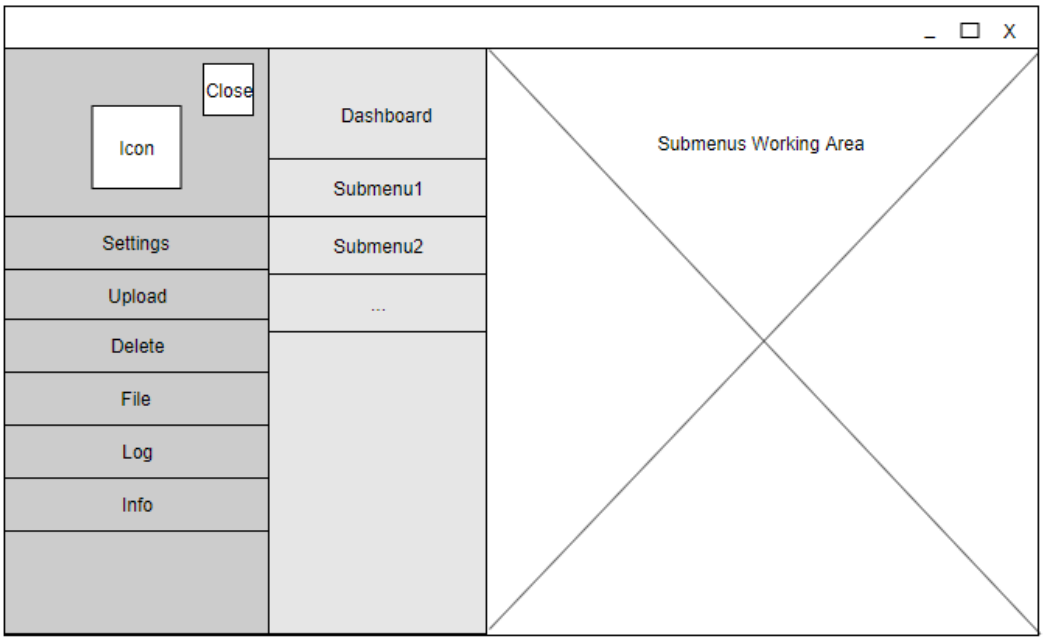
Minden automatizált futásról és annak eredményéről, elsősorban az általa készített, az archiválást követő aktuális havi WorkItemek leírását tartalmazó fájl e-mailben kerül tovább küldésre a beállításokban megadott címre, a szintén a beállításokban megadott Smtp szerveren és Porton keresztül. Így támogatva esetleges más csoportok munkáját cégen belül.

Ezen konfigurációs lehetőségekkel több, különféle beállítások használatával egyszerre több szerverre és több Team Projectre történhet egy adott beütemezett időben a feltöltés kezelés, és mindegyiknek saját specifikus beállítása lehet. Ezen létrehozott beállításokat tartalmazó fájlok, a grafikus felület beállítás importálás során gyors beállítási és sokoldalú modifikációs végrehajtást biztosít időt megtakarítva, hiszen a teljesen bekonfigurált beállításokat használhatjuk módosítás nélkül, vagy apró módosításokat eszközölve a hiba lehetőségének további minimalizálása mellett.

Grafikus szolgáltatás

A teljes körű felhasználás megfelelő biztosításának Minimum feltételeiként meg kell valósítani a következő felhasználási pontokat. A programcsomag grafikus felhasználói felületen keresztül nyújt lehetőséget a TFS szerverekkel való könnyed és egyszerű interakcióba lépéshez. Ahogy az automatizált és beütemezett adminisztráció és tevékenységkövető előkészítések és generálások történnek, ahhoz hasonlóan lehetőséget biztosít ezen automatizált folyamatok által létrehozott adatok (User Storyk, Taskok) és ezek közötti kapcsolatok felülírására, módosítására. A törlési lehetőségek egy vagy több, akár fájlból való nagyobb WorkItem mennyiséget, vagy teljes szerver törlésre, resetre nyújtanak lehetőséget. Természetesen ezek a grafikus kiegészítő áttekintő interakciók,

mindenekelőtt a feltöltéseknek, generálásoknak és egyéb műveleteknek, törléseknek biztosítanak egy egyszerű átfogó nyomon követési és ellenőrzési lehetőséget, amelyek kiterjednek, mind a logok, mind az archivált és aktív fájlokra egyaránt.



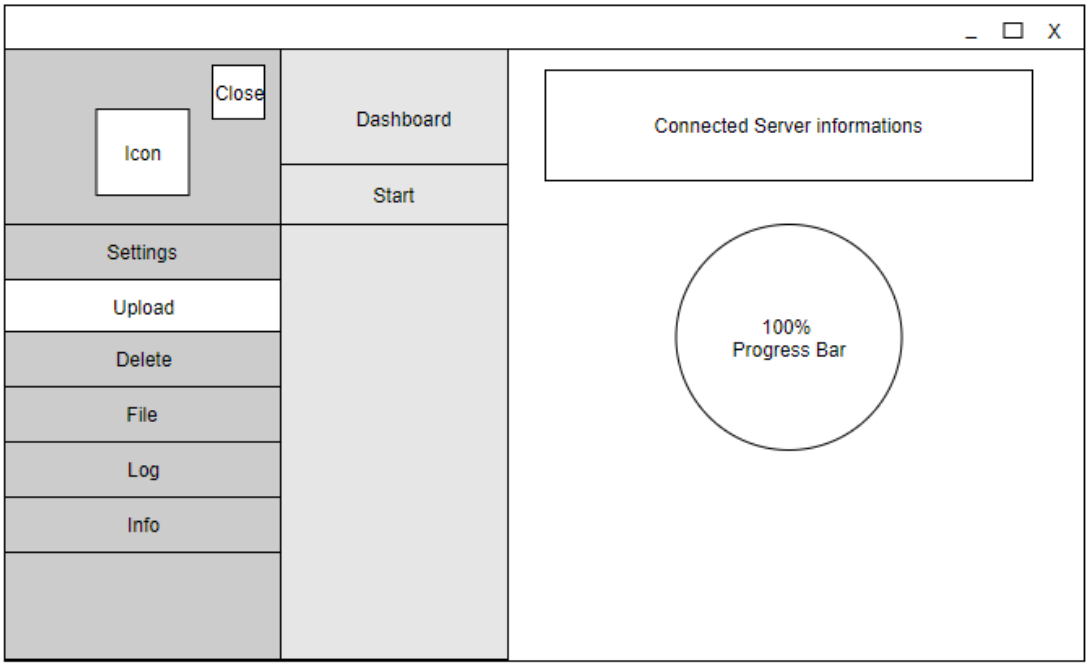
34. ábra
Képernyőterv, Menüterv.

A 34. ábrának megfelelő tervnek, terveknek felépítését követve. A program grafikus felületének bal szélén egy szürke mezőben található a fő menü szekció. Ezen szürke főmenü szakaszban fent középen az alkalmazás ikonjával, és a mező szekció jobb felső sarkában, egy a Microsoft Windows 10 operációs rendszer „Gépház” és egyéb alapértelmezett alkalmazásaiban is népszerű animációs menüfelület összezárás kapcsoló gombja található, amely során az írott menüpont szövegek „bezáródnak” és az ikonok maradnak meg. A fő menüpontokhoz tartozó a főmenü szürke paneljától „bentebb” egy füstös szürke felületen találhatjuk az almenüpontokat és egyéb menüponton belüli szolgáltatások gombjait. (Dashboard) Ezt követő „fehér” felületen, munkafelületen jelennek meg az aktuális menüpontok, almenüpontokban kiválasztott szolgáltatás teljes funkció palettája.

A főmenü 6 pontból áll. Első a beállítások, (Settings) ezen főmenü ponthoz tartozó irányítópulton (Dashboard-on) keresztül négy funkció érhető el, az „aktuális config, beállítás megnyitása” amely a program legelső indítása esetén természetesen konfiguráltalan, és egy értesítésben informálja a felhasználót ezen beállítás elvégzésére.

Addig semmiféle szerver kapcsolat nem áll fenn, inaktívak a funkciók. A „a külső beállítási fájl beimportálása” –Itt lehetőség van ezen külső, esetlegesen más által elkészített vagy korábban használt beállítás egyszerű és azonnali felhasználására az alkalmazáson belül. Plusz egy „Reset” a hibák visszavonása, és konfiguráció újrakezdés miatt. Végül természetesen a „Save” – Mentés funkcióval, amelyet követően a beállításoknak megfelelően megpróbál az adott TFS szerverre kapcsolódni.

A feltöltés (Upload) menüponton keresztül lehet indítani feltöltést, automatizált feltöltés módosítást. Feltöltés, módosítás során a konfigurációnak megfelelően létrehozza, feltölti és aktiválja a WorkItemeket. A grafikus feltöltés futtatás biztosítja a lehetőséget az aktuálisan, automatizáltan elvégzett feladatok módosítására. Csak arra az adott TFS szerverre és Team Projectre kell csatlakoznunk (Beállítások menüpontban elvégzett konfiguráció.), aminél szeretnénk ezen automatizált futtatás által létrehozott WorkItemeken elvégezni a módosításainkat. Így az aktuálisan aktív WorkItemek archivált fájlban történő menedzselése, megfelelő státuszban marad, és a következő hónapban ennek az információknak megfelelően hajtódat végre a következő havi adminisztratív automatizált előkészítés, felöltés a TFS szerveren. Archiválás és fájlkezelés esetén az aktuális hónap számával ellátott fájlok létrehozása biztosítja, hogy az adott havi műveletek eredményei hosszú távon elérhetőek maradjanak, és hónap első napján automatizált futás esetén is ezen fájl tartalmát felhasználja, mint az előző havi archiválásra váró WorkItemek listája. Aktuális aktív WorkItemek zárásra kerülnek a szerveren, emellett módosul az aktuális hónapszámmal ellátott fájl tartalma. Ha egyszerűen grafikus felületről indított feltöltést végzünk olyan TFS szerverre, ahol korábban nem volt interakció az alkalmazás grafikus vagy automatizált WorkItem létrehozás és feltöltés szolgáltatásával. A folyamat megegyezik az előzetes leírásban rögzítettekkel az archiválás kihagyásával. Ezt követően pedig, a feltöltés által létrehozott fájl miatt már következő futásnál az archiválás és fájlkezelés is végbe megy.

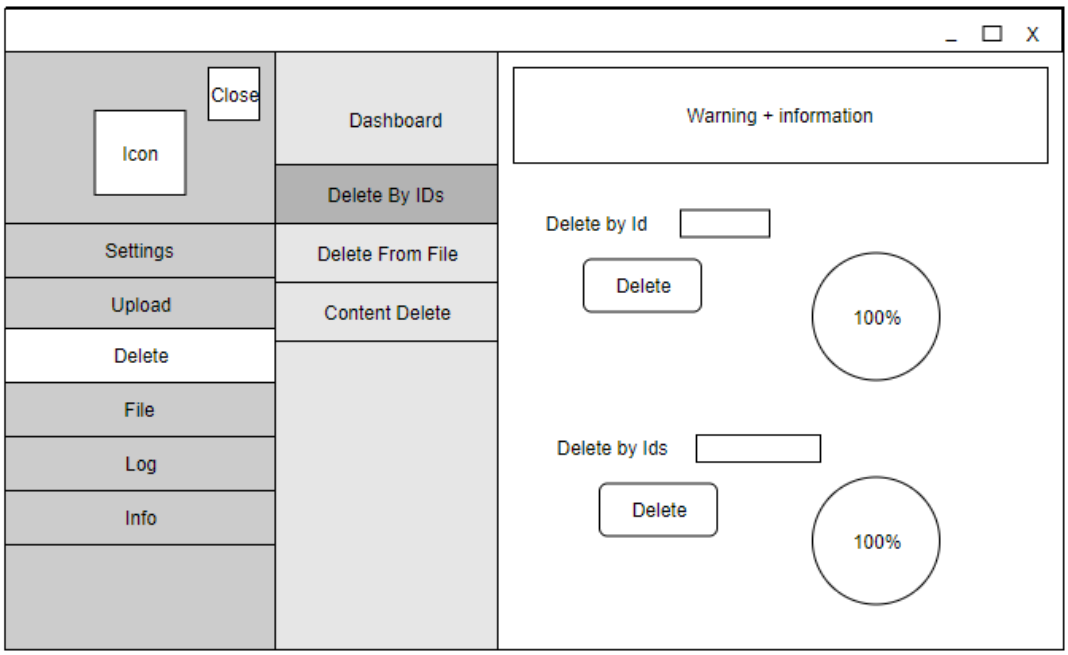


35. ábra
A feltöltés felület egyedi munkaterület terve.

A törlés (Delete) és azon három funkciója, azonosító (Id) alapján való egy vagy több WorkItem törlése, megfelelő fájlból való törlés, itt kiemelendő a fájl tartalma, melynek az alkalmazás általi könyvelési struktúrát kell követnie, vagyis azonosítóval kezdődő sorok majd azt követően a részletes leírások a WokrItemekről (A kulcs a sor elei azonosító, amely alapján töröl a fájlból az adott menüpont szolgáltatása.). Utolsó sorban pedig a teljes szerver tartalom törlése, így biztosítva az újra felhasználást. Minden esetben többszöri megerősítés szükséges a törlés végrehajtásához, és extra log fájlban történik a szerveren nem létező WorkItemek törlési kísérlete.

A „Delete” – Törlés főmenü pontot kiválasztva, az alkalmazás értesítés rendszere külön felhívja a felhasználó figyelmét, hogy ezen szolgáltatásokkal a szerveren törlési műveleteket hajthat végre, és ez fokozott odafigyelést igényel. Természetesen mind a három funkció használata esetén az alkalmazás az irányítópult melletti munkafelületen folyamatosan felhívja a figyelmet az óvatos használatra, és hogy minden végrehajtott műveletről jelentés készül, ami továbbításra is kerül, a beállításokban megadott címre!

Ez alapján a törlési felületek egyedi munkaterület terve:



36. ábra
Delete szekció terve.

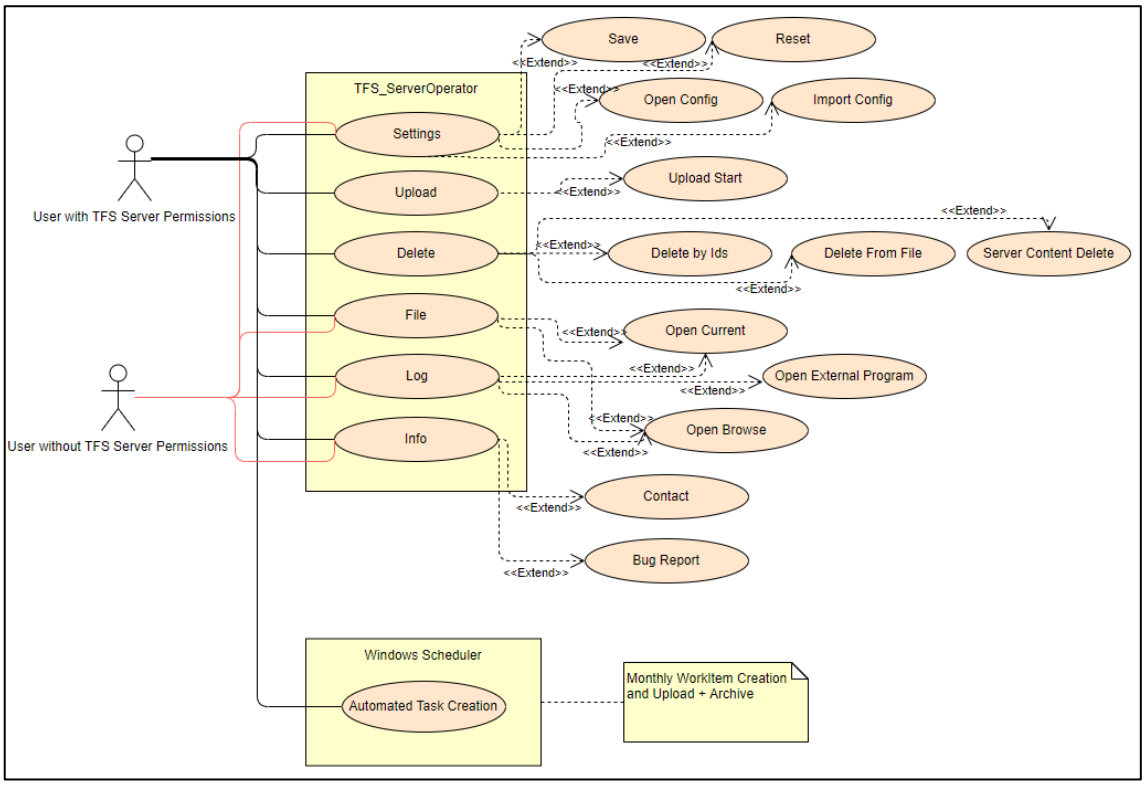
A fájl (File) menüpont biztosítja a 34. ábra terveknek megfelelő kinézettel és munkaterülettel a lehetőséget, hogy az aktuális automatizált feltöltés fájlját vagy a grafikus felületen indított műveleteket követően megnyithatjuk az aktuális aktív havi fájlt. Emellett biztosít lehetőséget egyéb fájlok betöltésére fájldialog ablakon keresztül, így egyszerűsítve korábbi esetleges archivált fájlok betöltését, áttekintését.

A Log menüpont. (34. ábra tervein alapul.) Ezen fülön belül van lehetőségünk áttekinteni az aktuális és esetleges korábbi log fájljainkat. Az alkalmazás minden futtatás esetén a beállításokban megadott névvel, a megadott helyen hozza létre a futtatáshoz tartozó log fájlt, a nevében egy idő bélyeget is elhelyezve az egyértelmű megkülönböztetés céljából. Így elkerülve egy óriási átláthatatlan, sok helyet foglaló log fájl létrejöttét. A menüpont három almenüpontban biztosít a fájlhoz hasonlóan, aktuális éppen a futó műveletekhez tartozó logfile megnyitását, emellett egy külön pontban fájldialog ablakon keresztüli korábbi logok megnyitására ad lehetőséget, kivéve az aktuálisan futó alkalmazás fájlja, ezen esetben értesítést kapunk. Végezetül, pedig lehetőséget ad, az aktuális logfájlunk külső programmal való megnyitására is.

Az Info – Kapcsolatfelvétel (34. ábra tervein alapul.) során lehetőségünk van az alkalmazáson belül üzenetet írni a fejlesztőnek, fejlesztőknek, szerviz szolgálatot ellátó embernek, csapatnak. Itt leírhatjuk esetleges véleményünket vagy feltehetjük akár specifikus az alkalmazással kapcsolatos kérdéseinket is. Emellett pedig az alkalmazás

használata során jelentkező hibák, problémák jelentésére is egy külön kapcsolatfelvételre alkalmas a hibajegyeket prioritásban kezelő felületre írhatunk bejelentést.

Összességében a specifikáció menüpontban általánosan leírt átfogó kritériumok, tervek és mind az automatizálás, mind a grafikus felület funkcióinak leírását, felhasználói lehetőségeit egy „Use Case” diagramm keretein belül is szemléltethetjük.



37. ábra
Use Case diagramm.

Megvalósítás

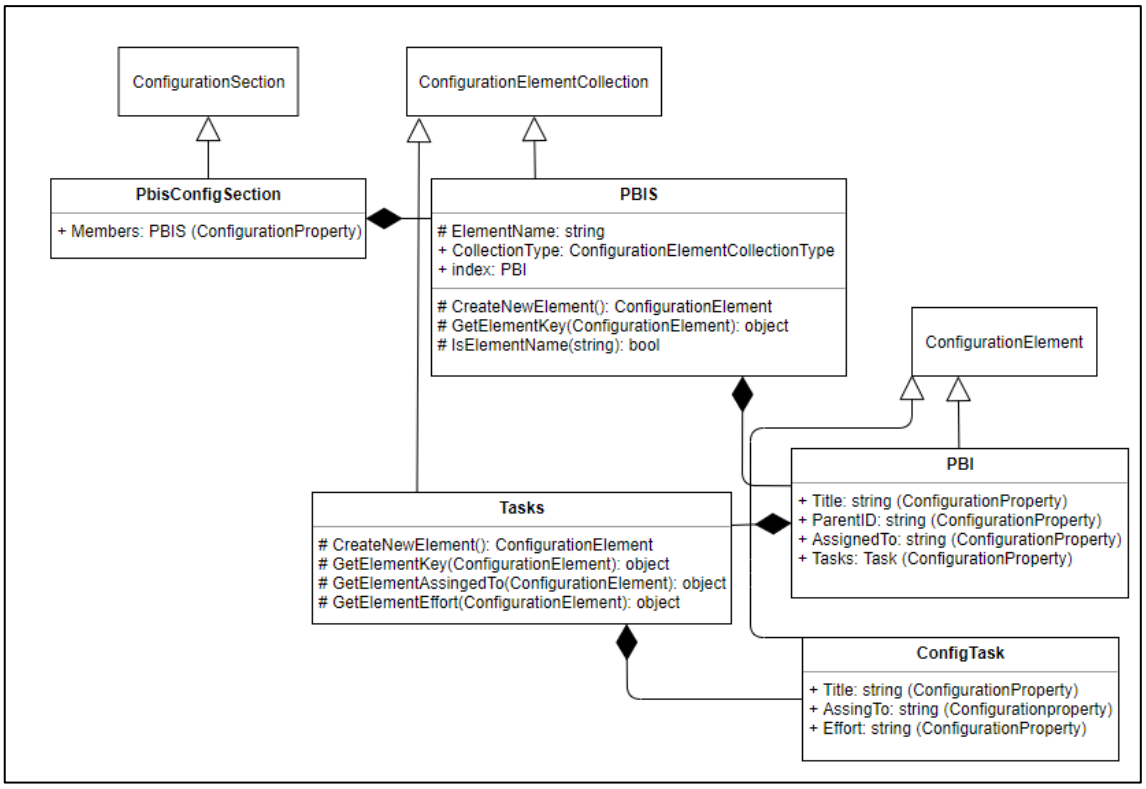
Az alkalmazás felépítése az MVC (Modell – Nézet - Vezérlő) architektúrájának megfelelően három fő részre bonthatjuk. A Modell egység, amely a különböző TFS szerveren végrehajtható műveleteket valósítja meg, mint a törlési módok, archiválások, WorkItem műveletek, (WorkItem létrehozások, kapcsolat menedzsment, feltöltés) kapcsolatfelvétel a TFS szerverrel stb. Alapvetően ezen rétegben található az üzleti logika jelentős része. Második a Vezérlő egység, réteg, amely megteremti a kapcsolatot a modell és a nézet rétegek között, feldolgozva az inputot a nézet réteg elemeitől és annak megfelelő feldolgozása a vezérlői utasításokon keresztül a modell réteg megfelelő szolgáltatásaihoz kerülve biztosítják az eredményt, „válaszreakciót”, így biztosítva az interakciókat megfelelően szeparált réteges, moduláris struktúrában. Harmadik pedig a

Nézet réteg, amely a felhasználói felületet biztosítja, amelyen keresztül a felhasználó képes interakcióba lépni a különböző szolgáltatásokkal, mindezt egy letisztult modern designon keresztül, amely nagyban megkönnyíti a különböző szolgáltatások igénybevételét. A nézet rétegen a „panelos felépítés" dominál, így magának az MVC architektúrának a modularitásán felül, maga a nézet réteg modularitása is jelentős, hiszen a teljes nézet csere mellett, amelyhez csak a vezérlőt kell egyszerűen „lekapcsolni" lehetőség van a nézet réteg különböző paneljainak a cseréje, mint például a Specifikáció grafikus terveinél látott ábráknál, (34,35,36) egyszerűen cserélhető a főmenü, almenü, és a munkaterületek különböző megvalósítása, ezen panelos felépítésnek köszönhetően.

Maga a szolgáltatásokat tartalmazó üzleti logika 4 interface megvalósításán, implementációján alapszik. Ezen interface-k egy a külön mappában („Interfaces") találhatóak és a következő nevekkel rendelkeznek: „IConnection", „IFileInteraction", „IMailInteraction" és „IServerInteraction". Emellett mindenképp meg kell említeni a következő „CustomTraceLogging" mappa tartalmát. Ezen mappában található kettő osztály a „Logger" és a „TextLogTraceListener" valósítja meg a teljesen egyedi logolási szolgáltatást. Harmadik kiemelendő mappánk pedig a konfigurációs XML fájlunk egyedi „tagelési" rendszerét és szisztémáját valósítja meg. Ez a „ConstomConfigSetup" nevezetű mappa a modellt és vezérlőt megvalósító projektben található, amely a teljes program Solution-ének a része értelemszerűen, mint az ezt megelőzően említett mappák, a tartalma pedig, „ConfigTask", „PBI", „PBIS", „Tasks" és „PbisConfigSection" ezen öt osztály valósítja meg ezen tagek szerkezetét, felépítését, használandó adatábrázolását. Amely a Fejlesztői Dokumentáción belül a 33. és 39. ábrán is látható szerkezetet eredményezi.

Beállítás Részletes Megvalósítása

Ahogy korábban is említésre került, a konfigurációnak egy egyedi „tagelési" rendszer van felépítve, amely során minden szükséges beállítás megfelelő tagolásban adható meg. Ezen rendszer különböző részei három különböző osztály öröklődésével valósul meg. (Mindegyik osztály az adott részhez szükséges osztálytól örököl) ezen osztályok a „ConfigurationSection", „ConfigurationElementSection" és a „ConfigurationElement".



38. ábra
Beállítás szekció UML Diagrammja.

A 38. ábrának megfelelően a „PbisConfigSection” osztály a beállítási rendszer első „eleme” a teljes szekciót összefogó rész, amely „PBICollectionSection” névvel található a beállítás leírásában. (Config fájl) Ezen belül található a „PBIS” osztálynak megfelelően a különböző PBI / User Storykat összefogó tag, amelyek a különböző PBI / User Storykat és a „hozzájuk tartozó” különböző Taskokat, ölelik fel, ha tartoznak a User Storyhoz, User Storykhoz Taskok. Főként a „PBIS” osztály egy plusz leíró szerkezeti elem a jobb tagoltság, átláthatóság és felépítés miatt. A „PBI” és a „Tasks” osztályok határozzák meg ezen belső szerkezetet. A „PBI” osztály meghatározza hogyan is épül fel egy létrehozandó, majd feltöltésre szánt PBI / User Stroy az alapvető és kulcsfontosságú változó adattagokkal, amelyet mindenképp egyedileg kell megadnia a felhasználónak a beállításokban. A „Tasks” osztály pedig ezen PBI / User Story tagen belüli nulla, egy, vagy akár több hozzá tartozó Task leírásának és felépítésének a szerkezetéért felel. A „ConfigTask” az az osztály, amely az adott PBI / User Stroy -hoz tartozó Taskok felépítését, létrehozását határozza meg, definiálva a szükséges ismételten a felhasználónak beállítandó kulcsfontosságú mezőket megadva. A következő oldalon látható ezen leírt beállítási felépítés szemléltetése.

```
<PBICollectionSection>
  <PBIS>
    <PBI Title="Month Regular Meeting1 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="2" />
      </Tasks>
    </PBI>
    <PBI Title="Month Regular Meeting2 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="12" />
      </Tasks>
    </PBI>
    <PBI Title="Month Regular Meeting3 - (Engineers)" ParentID="467" AssignedTo="Nemes László">
      <Tasks>
        <add Title="Laci" AssingTo="Nemes László" Effort="22" />
      </Tasks>
    </PBI>
  </PBIS>
</PBICollectionSection>
```

39. ábra
A Beállítás UML Diagram szerkezeti szemléltetése.

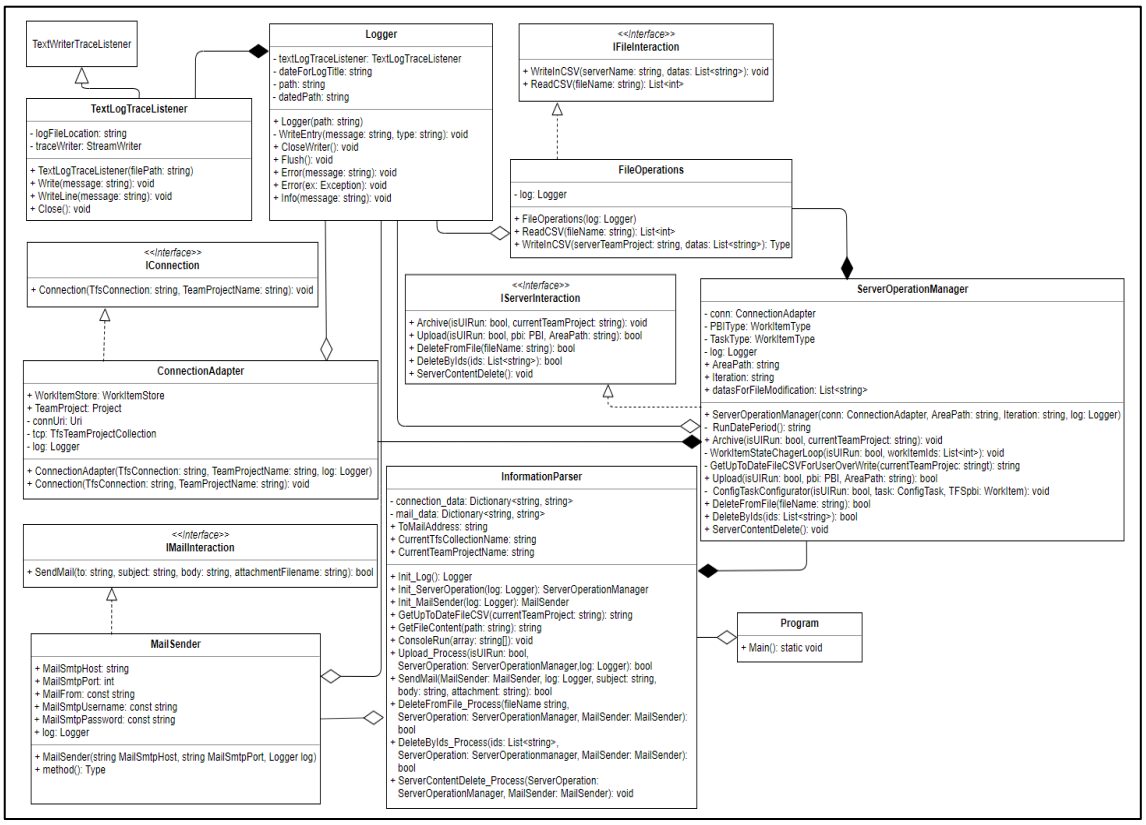
A TFS szerver connection és jelentés küldés beállításai, egy név – érték páron alapuló XML konfigurációs szekcióként valósul meg.

A „Connection” szekció, ahol az adott szerverhez adja meg a felhasználó az eléréshez és kapcsolódáshoz szükséges adatokat, beállításokat. A „TfsCollection” maga a szerver eléréséhez szükséges url. Azt követően az adott szerveren meg kell adnia, hogy melyik Team Project nyilvántartását, (User Story és Taskok) szeretné innentől automatizálva végrehajthatni havi szinten vagy feltöltést indítani a grafikus felületről. Ez követi a „TeamProjectName” és a hozzá tartozó „AreaPath” és „Iteration”. A következő egység a „MailInformation”. Itt az aktuális automatizálásról való jelentés és az általa létrehozott, és módosított WorkItemekről készült fájlt és értesítést továbbítja a megadott címre, az adott Smtp -n és Porton keresztül. Lehetőség van külső vagy akár cégen belüli Smtp használatára is, ezen szabadságot nem korlátozza az alkalmazás.

```
<Connection>
  <add key="TfsCollection" value="http://localhost:8080/tfs/SzakdolgozatCollection" />
  <add key="TeamProjectName" value="Szakdolgozat_Other_Project" />
  <add key="AreaPath" value="Szakdolgozat_Other_Project" />
  <add key="Iteration" value="Szakdolgozat_Other_Project" />
</Connection>
<MailInformation>
  <add key="mail_address" value="wow.laszlo@gmail.com" />
  <add key="smtp_host" value="smtp.gmail.com" />
  <add key="port" value="587" />
</MailInformation>
```

40. ábra
Connection és MailInformation beállítás szerkezeti szemléltetése.

Üzleti Logika Általános Megvalósítása



41. ábra
Üzleti Logika UML Diagrammja.

Ahogy korábban is említésre került az üzleti logika négy interface megvalósításán alapszik. Ezen megvalósító osztályok pedig a „ConnectionAdapter”, „MailSender”, „FileOperations” és a „ServerOperationManager”. A logolási rendszer is teljesen egyedi implementáció, mivel meg kell valósítani egy egyszerű, egységesen formázott információt és error – hibaüzeneteket leíró szisztémát, amely időbélyegekkel és vezető INFO és ERROR típusokkal adja meg az aktuális eseményeket. Ezen megvalósítás a „TextWriterTraceListener” osztályból származó „TextLogTraceListener” osztályon keresztül valósul meg, amely magát a „Logger” osztályt támogatja. Ezen két osztály a „TextLogTraceListener” és „Logger” között így Kompozíció figyelhető meg, mivel a „Logger” képtelen ellátni a „TextLogTraceListener” nélkül a feladatát. Az „InformationParser” osztály látja el az architektúra Vezérlő szerepét. Itt összpontosul a modell szolgáltatásainak a nézet felé esetlegesen az automatizált futtatás felé történő utasításkezelés és a modell megfelelő szolgáltatásának indítása, alkalmazása. Mind az automatizált felhasználás mind a grafikus felület ezen osztály metódusainak hívásával hajtja végre a műveleteket. (Az adott inputra reagál a vezérlő és a megfelelő modell béli implementált szolgáltatást biztosítja a hívás felé.)

Az „InformationParser” osztály ezen vezérlő feladatához elengedhetetlen a Kompozíciós kapcsolat a „ServerOperationManager” osztállyal. Az interfaceket implementáló szolgáltatások, műveletek osztályai, mint például a „MailSender” biztosítja az automatizált futtatás követő e-mailen keresztüli értesítést és jelentés küldését (létrejött fájl) a beállításoknak megfelelő helyre, és a különböző törlési eseményekről való értesítések küldését egyaránt. A „ConnectionAdapter” biztosítja a különböző TFS szerverekre és Team Projectekre való kapcsolódást. A „FileOperations” osztály biztosítja a különböző fájl műveleteket, (írás, olvasás, feldolgozás) ami a feltöltésnél létrejövő új aktuális fájl előállítása, az archiválás kezelése és egyéb szolgáltatások, mint a fájlból való törlés esetén való fájlhasználatot. A „ServerOperationManager” osztály a modell egyik fő osztálya, amely a korábban említett osztályokkal, mint a „ConnectionAdapter” és „FileOperations” Kompozíciós viszonyban áll, hiszem a megfelelő működéshez elengedhetetlen ezen osztályok szolgáltatásai, és a „Logger” osztállyal, mint a többi korábban említett interface -eket megvalósító osztályok is Aggregációs viszonyban áll.

Modell szolgáltatásainak megvalósítása

Team Foundation Server kapcsolódás

A Különböző TFS szerverekre és azon belüli Team Projectekre való kapcsolódáshoz szükséges adatokat, beállításokat a konfigurációs fájl „Connection” szekciója biztosítja, így fenntartva az egységes beállítás kezelést, és a modifikációt. a 40. ábrának megfelelően, a felhasználónak meg kell adnia a szerver eléréséhez szükséges URL -t, amelyre egy Uri osztály segítségével fog kapcsolódni. Emellett a szerver megfelelő Team projectéhez való kapcsolódóhoz szükség van ezen TeamProject névre, majd Iteration és AreaPath értékekre, amellyel meghatározzuk teljes mértékben a kapcsolódást.

A Modellünknek külön az „IConnection” interface implementálásával létrehozott „ConnectionAdapter” osztály biztosítja kapcsolat felvételt. Ezen „ConnectionAdapter” osztály konstruktora által hívott, az interface implementálásának eleget tevő metódus, a „Connection(TfsConnection, TeamProjectName)” biztosítja a kapcsolódást. Ezen metódus a Fejlesztői eszközöknél említett NuGet package által biztosított osztályelérések és .dll fájlok segítségével („Microsoft.TeamFoundationServer.ExtendedClient”) csatlakozik. A NuGet package nélkül elérhetetlenek a kapcsolathoz szükséges lépések.

Az osztályok mellett külön kiemelendő a „Microsoft.WITDataStore64.dll” és „Microsoft.WITDataStore32.dll” –ek amelyek nélkül nem jöhet létre kapcsolat, nem fut le a megfelelő folyamat. Ezen .dll fájlok a buildeléskor jönnek létre az alkalmazás bin/Debug mappájában, külön fejlesztő környezeti referencia megjelölésük nincs. A TFS szerverekre való kapcsolódás teljesen egyedi és eltérő, más szerverekhez képest a különböző WorkItem kezelés és elérések miatt. Ahogy említve lett, első lépésben egy Uri() segítségével elkezdődik a „kapcsolatfelvétel”, majd egy TfsTeamProjectCollection osztály inicializálása következik, ezen Uri paraméterrel.(Hiszen az URL utolsó tagja maga a collection neve amelyen belül szeretnénk elérni az adott TeamProjectet.) Így az adott szerver Uri() paraméterében lévő szerver collectionra történik a csatlakozás. Ezután egy WorkItemStore szervíz kérés következik, hogy az aktuális WorkItemStore -t megkapjuk, (WorkItem elésérekhez) végül pedig egy Projekt inicializálás következik, ahol a beállításokban kapott TeamProjectName – nek megfelelő Storet határozzuk meg. Ezen lépések sorozata, „Uri” majd „TfsTeamProjectCollection” ezt követő „WorkItemStore” és „Project” inicializálást követően jött létre a valós kapcsolat, és érjük el a szerveren lévő WorkItemeket. Ezen folyamatot követően van alkalmunk bármiféle lekérdezésre, query -k futtatására (Query -k esetén is az adott querynek eleget tevő WorkItemek egy collectionét kapjuk meg.) és módosításokra a szerveren, természetesen, ha az adott felhasználó Microsoft fiókja rendelkezik a szerveren megfelelő jogosultságokkal.

Ezen összetett kapcsolódási folyamat kód szinten:

```
/// <summary>
/// Based on the App.config file this method will connect to the TFS server.
/// </summary>
/// <param name="TfsConnection">The server URL, where we would like to connect</param>
/// <param name="TeamProjectName">The Project name on the server, where we would like to do something</param>
public void Connection(string TfsConnection, string TeamProjectName)
{
    if (!(TfsConnection == null || TeamProjectName == null))
    {
            connUri = new Uri(TfsConnection);
            tpc = new TfsTeamProjectCollection(connUri);
            WorkItemStore = tpc.GetService<WorkItemStore>();
            TeamProject = WorkItemStore.Projects[TeamProjectName];
            string message = string.Format("Connection was successful to {0} and the ProjectName is {1}", TfsConnection, TeamProjectName);
            log.Info(message);
            log.Flush();
    }
    else
    {
        log.Error("ConnectionException in the ConnectionAdapter class");
        log.Flush();
        throw new ConnectionException();
    }
}
```

42. ábra
Connection megvalósítása kód szinten.

Bármilyen esetleges köztes hiba esetén, vagy a metódus, vagy a megfelelő hívó környezete kezeli a hibát. Ezen kezelés elsődleges oka a Nézet felületen megfelelő

értesítések, visszajelzések beállítása, mint például hibás csatlakozási adatok esetén a nézet felület feltöltés szolgáltatásánál „Inactive” jelzés látható, a szerver információknál pedig kapcsolódási hiba feliratok. Bármely más szolgáltatás, mint a törlések is jeleznek, hogy a szerverkapcsolat nem jött létre. Automatizált futtatás esetén is, mind az értesítés kiküldésének hiánya, mint a log fájl tiszta a kivétel teljes szövetét is tartalmazó hibaüzenetet biztosít.

Feltöltés – Process Templates

A feltöltés az egyik legnagyobb és legösszetettebb folyamat. Az architektúrában vezérlő szerepet betöltő „InformationParser” osztály két metódusban biztosítja ezen feltöltés kivitelezését. Egyik az automatizált futtatás esetén használandó, másik pedig a grafikus felület által használt szolgáltatás.

Először a grafikus futtatás, az „InformationParser” osztály logikai értékkel visszatérő Upload_Process(bool isUIRun, ServerOperationManager ServerOperation, Logger log) metódusa valósítja meg a feltöltést. Ezen metódus első paramétere egy logikai flag, (isUIRun) amely azt határozza meg, hogy kézi az-az grafikus felületről indított szolgáltatás igénybevétel következik be vagy automatizált futtatás történik. (Ezen metódus hívása történik automatizált esetben is, kissé eltérő környezetből, mivel ott egy ConsoleRun(string[] array) metóduson belül történik a hívás.) Ezt követő ServerOperation paramétere az Upload_Process metódusnak egy korábban számos alkalommal említett „SzerverOperationManager” osztályt jelent, amely magában foglalja ezen Modell réteg által biztosított szolgáltatásokat, amelyek a vezérlőn keresztül elérhetőek. Feltöltés esetén a fájl dokumentum elkészítéséhez használandó Lista (datasForFileModification) tisztítása következik be elsősorban. (Ezen listába kerülnek a feltöltés során elkészített WorkItemekről az információk, amelyek kiírásra kerülnek a fájl dokumentálás részeként.) Majd az Archive(bool isUIRun, string currentTeamProject) metódus kerül meghívásra a „SzerverOperationManager” osztályból. (Ezen Archive metódus gondoskodik a TFS szerveren aktuálisan aktív WorkItemek zárásáról és ezen zárás jelzéséről magukon a WorkItemeken, tagek formájában. Automatizált esetben az előző havi fájl tartalma alapján végzi a műveletet a TFS szerveren, grafikus futtatáskor pedig az aktuális havi fájl alapján, ha van, amely feltöltéskor újraírásra kerül, hogy továbbra is az aktuálisan aktív WorkItemeket tartalmazza.) Ahol szintén megtalálható, átadásra kerül ezen logikai flag, (isUIRun) hogy ezen futtatás automatizált vagy grafikusan indított. (Archive metódus paraméterlistája) Hiszen a grafikusan indított

esetben a korábban leírtak alapján, nem az előző havi fájl alapján kell végrehajtania az archiválást, hanem az aktuális havi fájl tartalmát kell archiválni, természetesen, ha előzetesen volt valami futtatás és van ilyen fájl. Ellenkező esetben ezen művelet kimarad, logban jelzésre kerül és megtörténik a feltöltés, ha minden rendben van az adatokkal, a beállítás szekcióban. Az archiválás második paramétere az adott TeamProject neve, hiszen egy TFS szerveren számos TeamProject lehet, amelyhez mind külön történik a fájl létrehozás, így elkülönítve minden TeamProjectet is egy adott TFS szerveren. Ezen archiválást és lista kezelést követi egy ciklikusan meghívott feltöltés, Upload (bool isUIRun, PBI pbi, string AreaPath) metódus. (Upload_Process metóduson belül) Maga az Upload_Process olvassa a beállításokban rögzített feltöltési szekciót, és azon beállítások alapján hívja meg a megfelelő PBI -okra / User Storykra és hozzájuk tartozó (linkelendő) Taskokra a feltöltést. Az isUIRun itt is az automatizált vagy a grafikus felületről indított feltöltés indítást jelzi. Az Upload metódus „AreaPath" paramétere az adott TeamProject alapú, egy TFS szerveren minden TeamProject számára külön fájlban történik a jelentés készítés, ezért havi szintű megfelelő elkülönítés miatt került a paraméterlistába az „AreaPath". (Feltöltés folyamán, amikor ellenőrzésre kerül, hogy ezen WorkItem aktívan létezik-e már a TFS szerveren az adott TeamProjecten, esetleges más TeamProjecten ugyanezen a néven létrehozott WorkItem ne okozzon problémát a feltöltés során, hiszen teljesen különböző két TeamProject, csak egyező nevű WorkItem, amely előfordulhat éles használat esetén egy TFS szerveren.) Ha sikeresen feltöltésre és aktiválásra került minden WorkItem, igaz értékkel tér vissza az Upload_Process, ellenkező esetben hamissal.

Automatizált esetben ezen feltöltés egy ConsoleRun metóduson belül kerül meghívásra, ahol a megfelelő /config kapcsoló utáni beállításokat tartalmazó fájl alapján fog megtörténni az automatizált feltöltés. (.bat fájlként érdemes ezen automatizált futtatásokat felvenni Feladatütemezőben, ahogy korábban is kifejtésre került, mind a beállítás megvalósítás mind a specifikációnál) A korábban kifejtett Upload_Process metódus kerül meghívásra ugyan úgy, a logikai flag (isUIRun) -ben jelezve, mint hamis érték, hogy ezen futtatás automatizált, így fog lefutni az Upload_Process és a benne található Archive metódus is, hogy ezen futtatás gépi automatizált. Utolsó lépésként pedig a „MailSender" osztály „SendEmail" metódusa mindenről jelentést küld a csatolva a létrehozott friss fájlt.

Ezen automatizált és grafikus futás által végzett WorkItem létrehozás és feltöltés, a WorkItem tagelésekben jelzésre kerülnek, hogy ezen WorkItemet grafikus felületről

indított feltöltés vagy automatizált futás hozta-e létre, így megkülönböztetve mindenki számára a TFS szerveren.

Maga a feltöltés metódus (Upload) végzi a legnagyobb munkát, hiszen ezen metódus készíti el a WorkItemeket a beállítások alapján, azon beállításokat kiegészítve alapértelmezett értékekkel, hogy valóban létre tudjon hozni egy WorkItemet. Az adott beállított értékeknek megfelelően, mielőtt elkezdené felépíteni és megadni a WorkItemnek a szükséges Field értékeit, futtat egy ellenőrző Query-t, hogy biztos nincs-e az adott teamprojecten már létező és aktív változata a létrehozni kívánt WorkItemnek. (Itt jön képbe a korábban leírt AreaPath szükségessége.) Ha a Query eredményeként létrejövő WorkItemStore üres, akkor azt jelenti nincs ilyen WorkItem így elkezdődhet a létrehozás. Megadásra kerül a beállításokban felvett név, kiegészítve egy havi intervallum, dátum jelzéssel, megtörténik a feltagelése a WorkItemnek (Itt jelzésre kerül gépi vagy grafikus futtatás során létrejövő WorkItemről van-e szó.) Beállításra kerül a WorkItem leendő tulajdonosa az „AssignedTo” beállításnak megfelelően és beállításra kerül az „Iteration” és az „AreaPath” is. Meghatározásra kerül az alapértelmezett kockázati mező érték, amely jelen esetünkben alacsony lesz, plusz elkezdi a WorkItem leírás mezőjét, amelyet a felhasználó használat során folytathat. Megtörténnek a linkelések mind az adott User Story típusú WorkItem esetén a szükséges Feature WorkItemhez, és User Story esetén a hozzá tartozó Taskok létrehozását követően az adott User Storyhoz. (ConfigTaskConfigurator metódus, amely a beállítások alapján létrehozza a szükséges User Storyhoz tartozó Taskokat és létrehozza a kapcsolati linkeket. Kiemelendő, hogy Taskok esetén a beállítás „Effort” tagbéli értéke alapján felvételre kerül az adott Taskra szánt munkaórák száma, és a hátralévő felhasználható órák száma, a teljesített vagy már felhasznált órák száma pedig null értéket kap.) A linkeléseket követően majd egy Validate() metódussal ellenőrzésre kerül, hogy minden szükséges érték megadásra került-e a megfelelő WorkItem szabályoknak megfelelően (Process Template), hogy „New” státuszú WorkItemként el lehessen menteni és befejezni az elkészítését. Ezt követően egy státuszváltás végrehajtódik és egy újbóli Validate() ellenőrzést követően „Aktív” státuszba kerülve kerül mentésre. Így teljesen elkészülve és átadva a használatra.

Ezen folyamatok mellett kerül folyamatos írásra az elkészült aktív státuszba kerülő User Stroy és Task WorkItemek a „datasForFileModification” nevű listába, amelyből elkészül a folyamatok végén a .CSV fájl, a feltöltött adatok alapján. Minden feltöltés esetén azért kerül tisztításra a lista, hogy grafikus esetben, ha egymás után több

futtatásra kerül sor, minden esetben egy tiszta, üres listába kerüljenek a kiírandó értékek, így megelőzve az esetleges tartalmi hibákat.

Process Teampaltek, sajnálatos módon ezen WorkItem létrehozások, egyéb WorkItem Field (Mezők) megadások, és ezek típusai, felvehető értékei nincsenek megfelelően dokumentálva Microsoft részről, így a mellékleten megtalálható Process Templatekből lehet az éppen használni kívánt WorkItem típus XML fájljából megszerezni az információkat, hogy hogyan is kell felépíteni egy WorkItemet. Milyen hivatkozási (refname) névvel utalhatunk az egyes mezőkre a kódban, milyen értékeket, és ezen értékeknek milyen típusai elfogadottak egy adott Field valid megadásához, milyen egyéb megkötéseknek kell megfelelni különböző esetekben stb. (Egyes mezők, csak előre a Process Templateben rögzített értékeket vehetik fel.) Természetesen ezen Process Templatek letölthetőnek Visual Studio -n keresztül, ha csatlakozunk egy adott TFS szerver Collectionre (Team Explorer), akkor a „Team” beállítási fülön magának a Visual Studionak, azon belül „Team Project Collection Settings” és „Process Template Manager” -en keresztül letölthető az aktuális Process Template fájlkötet.

```
<?xml version="1.0" encoding="utf-8"?>
<witd:WITD application="Work item type editor" version="1.0" xmlns:witd="http://schemas.microsoft.com/VisualStudio/2008/workitemtracking/typedef">
  <WORKITEMTYPE name="User Story" refname="Microsoft.VSTS.WorkItemTypes.UserStory">
    <DESCRIPTION>Tracks an activity the user will be able to perform with the product</DESCRIPTION>
    <FIELDS>
      <FIELD name="Id" refname="System.Id" type="Integer" reportable="dimension" />
      <FIELD name="Title" refname="System.Title" type="String" reportable="dimension">
        <REQUIRED />
        <HELPTEXT>What the user will be able to do when this is implemented</HELPTEXT>
      </FIELD>
      <FIELD name="Description" refname="System.Description" type="HTML">
        <HELPTEXT>Description or reference to the story that must work for this work to be considered complete</HELPTEXT>
      </FIELD>
      <FIELD name="Story Points" refname="Microsoft.VSTS.Scheduling.StoryPoints" type="Double" reportable="measure" formula="sum">
        <HELPTEXT>The size of work estimated for implementing this user story</HELPTEXT>
      </FIELD>
      <FIELD name="Assigned To" refname="System.AssignedTo" type="String" reportable="dimension" syncnamechanges="true">
        <ALLOWEXISTINGVALUE />
        <VALIDUSER />
        <HELPTEXT>The person currently working on this story</HELPTEXT>
      </FIELD>
      <FIELD name="Area Path" refname="System.AreaPath" type="TreePath" reportable="dimension">
        <HELPTEXT>The area of the product with which this user story is associated</HELPTEXT>
      </FIELD>
      <FIELD name="Iteration Path" refname="System.IterationPath" type="TreePath" reportable="dimension">
        <HELPTEXT>The iteration within which this user story will be implemented</HELPTEXT>
      </FIELD>
      <FIELD name="History" refname="System.History" type="History">
        <HELPTEXT>Discussion thread plus automatic record of changes</HELPTEXT>
      </FIELD>
      <FIELD name="State" refname="System.State" type="String" reportable="dimension">
        <HELPTEXT>New = Not started yet; Active = work remains to be done; Resolved = awaiting acceptance tests; Closed = acceptance tests passed</HELPTEXT>
      </FIELD>
      <FIELD name="Reason" refname="System.Reason" type="String" reportable="dimension">
        <HELPTEXT>The reason why the story is in its current state</HELPTEXT>
      </FIELD>
      <FIELD name="Changed Date" refname="System.ChangedDate" type="DateTime" reportable="dimension" />
      <FIELD name="Changed By" refname="System.ChangedBy" type="String" reportable="dimension" syncnamechanges="true">
        <VALIDUSER />
        <ALLOWEXISTINGVALUE />
      </FIELD>
      <FIELD name="Created Date" refname="System.CreatedDate" type="DateTime" reportable="dimension" />
      <FIELD name="Created By" refname="System.CreatedBy" type="String" reportable="dimension" syncnamechanges="true" />
      <FIELD name="State Change Date" refname="Microsoft.VSTS.Common.StateChangeDate" type="DateTime">
        <WHENCHANGED field="System.State">
```

43. ábra
User Story Process Template Részlet.

A hosszú tárgyias megvalósítás leírása mellett, a kód ezen metódusai is, mint az összes többi programban lévő metódus kommentelve van és olvashatóságot segítő változóneveket használ. Kiemelt részként kezelve ezen „Upload” és a hozzá tartozó privát „ConfigTaskConfigurator” metódus talán nagyobb részleteséggel is taglalva van, így

felhívva a figyelmet a trükkösebb kódsorokra, részletekre, bővebb és részletesebb magyarázatokkal.

Törlési lehetőségek (Egy, Több, Fájlból, Szerver)

Törlési funkciók, három kategóriába sorolható törlési funkció áll rendelkezésre, amelyek a grafikus felületről indíthatóak el kizárólagosan. A főmenü „Delete" menüpontján belül, az almenü (Dashboard) három gombon keresztül biztosítja ezen törlési funkciók igénybevételéhez szükséges munkaterületet. Első sorban az Azonosító (Id) alapú törlés, törlések. Lehetőség van azonosító alapján egy WorkItem törlésére (ami a TFS 2017 es verziójától az online kezelőfelületen is elérhető) és lehetőség van egyszerre több WorkItemet azonosító alapján való törlésére. Ezen lehetőségeket a „DeleteByIds_Process" metódus hívásával hajtódnak végre, amely a vezérlő osztály, az „InformationParser" metódusa. Ezen metódus hívja meg a „ServerOperationManager" – nek a „DeleteByIds" metódusát, amely paraméterül egy string listát vár. Ezen paramétert a metódus többszöri felhasználása indokolta. Így akár egy, akár több WorkItemet szeretnénk törölni ez az egy metódus képes elvégezni a feladatot. Mivel a Nézet felületről, Vezérlőn keresztül érkezik be a törlési információ és az azonosító(k) a megfelelő adatbevitel ellenőrzés nem ezen metódus feladata. Maga a „DeleteByIds" metódus, ahogy említve lett akár egy, akár több WorkItem törlése a feladata és egy listán keresztül kapja meg a paramétereket. Mivel string lista, elvégzi a megfelelő Parsolásokat és ezt követően a „Microsoft.TeamFoundationServer.ExtendedClient" NuGet Package általi .dll referenciákat használva, az adott TFS szerver kapcsolaton, a szerver WorkItemStore-án keresztül végzi el a törlést az azonosító(k) alapján a „WorkItemStore.DestroyWorkItems" hívással. Ennek kulcsfontosságú jelentősége, van, hogy ezen metodika alapján történt az implementáció, mivel egy WorkItem törléséhez minden függőségét megfelelő módon meg kell szüntetni User Story esetén az adott Feature WorkItem RelatedLinkjét, és minden az adott User Storyhoz tartozó Taskal is meg kell szüntetni a RelatedLink kapcsolatot, ez követően hajtható végre maga a WorkItem törlése. Ezen linkelési kapcsolat megszüntetésének fő oka a „WorkItemStore.DestroyWorkItems" használata, így elérhető, hogy a törlést követően az összes többi WorkItem, ahol a linkelési kapcsolat megszűnt az adott WorkItemmel, megfelelő valid státuszban, menthető formában maradjanak, így fenntartva a működésüket. Több WorkItem egyidejű törlése esetén is ezen metodika játszódik le, csak az adott paraméterlistában érkező adatmennyiség különbözik (több).

A hibakezelés, itt szintén a „WorkItemStore.DestroyWorkItems” által biztosított megoldás köré építkezik. Ha olyan azonosító kerülne törlésre, vagy olyan azonosító is van a törlendő listában, amelyhez nem tartozik a szerver oldalon WorkItem, a törlés teljes folyamata nem hajtódik végre, hiszen nem tud minden megadott elemet törölni, így egyet sem fog. Erről egy extra log fájl készül a hiba okával és a hibát okozó WorkItem azonosítójával, azonosítóival, amely nem létezik, nem léteznek.

Fájlból való törlés. A fájlól való törlés esetén az „InformationParser” „DeleteFromFile_Process” metódusán keresztül történik meg a modellből való törlés művelet hívás. A „DeleteFromFile” metódus („ServerOperationManager” metódusa) hajtja végre ezen törlést, amely paraméterül az adott fájlnak a nevét kapja meg, amelyet feldolgozva el kell végeznie az abban szereplő WorkItemek törlését. A „DeleteFromFile” metódus első sorban inicializál egy „FileOperations” osztályt, objektumot, aminek a „ReadCSV” metódusával dolgozza fel a fájlt, és ad vissza a feldolgozását követően egy listát az adott azonosító értékekkel. Ezen oka, hogy a korábban az azonosító törléseknél leírtaknak megfelelően „WorkItemStore.DestroyWorkItems” hívást tudjuk ismételten használni a biztonságos, és minden a törlés során érintett (RelatedLinkek) WorkItem megfelelő és valid állapotban maradjon. Hiszen minden érintett WorkItem esetén itt is el kell végezni a linkek eltávolítását ás ezt követően menteni kell a magváltozott helyzeteket.

A fájl szerkezetileg a „FileOperations WriteInCSV” metódusának megfelelő felépítéssel kell rendelkeznie. Ilyen struktúrájú fájlokat állít elő az alkalmazás feltöltések esetén is, mind automatizált, mind grafikus feltöltés indításkor. A CSV fájl szerkezeti felépítése a következő, „Id - Title - AssignedTo - State”, ennek megfelelő fájltartalom szükséges a törlésnél megadott fájlnak is, hiszen az „Id” – Azonosítók alapján állít össze egy törlendő listát a fájlból a ReadCSV metódus a „WorkItemStore.DestroyWorkItems” hívása előtt.

A hibakezelés, nem megfelelő fájlformátum esetén, mind logban, mind a grafikus felületen megtörténik ezen értesítés. Természetesen, ha olyan azonosító kerülne törlésre, vagy olyan azonosító is van a törlendő listában, amelyhez nem tartozik a szerver oldalon WorkItem, a törlés teljes folyamata nem hajtódik végre, hiszen nem tud minden megadott elemet törölni, így egyet sem fog. Ezt a „WorkItemStore.DestroyWorkItems” használata biztosítja. Erről egy extra log fájl készül a hiba okával és a hibát okozó WorkItem azonosítójával, azonosítóival, amely nem létezik, nem léteznek.

Teljes szerver tartalom törlése. Előfordulhatnak olyan esetek, hogy egy adott TFS szerver teljes tartalmának a törlése mellett születik meg a döntés, esetleges leállítás esetén, vagy amikor az adott TFS szerver korábbi verziója archív backupként sem maradhat meg migrációt követően például. (Esetleges verziók közötti eltérő Process Template felépítés, különbségek miatt.) Ekkor egy teljes szerver törlés funkció jelentős időt takaríthat meg, és minden törlés megfelelő, biztos módon következik be. Az „InformationParser ServerContentDelete_Process" metódussal látja el a teljes törlés feladatát, amely a „ServerOperationManager" –nek a „ServerContentDelete" paraméter nélküli metódusát hívja meg. Ahogy a korábbi fájlból való, vagy megadott azonosító(k) szerinti törléseknél, itt is a kulcs szerepet a „WorkItemStore.DestroyWorkItems" játssza, a megfelelő és ellenőrzött törlések érdekében. A „WorkItemStore.DestroyWorkItems" használata, hívása előtt a „ServerContentDelete" metódus egy Query -t futtat, így a szerveren lévő összes WorkItemet egy WorkItemCollection-be összefogva. Ezen WorkItemeken végig iterálva készíti el a „WorkItemStore.DestroyWorkItems" hívásához szükséges azonosító listát, majd hajtja végre a teljes TFS szerver állománytisztítást.

A hibakezelés ezen esetben a Query alapján történik, hogy üres-e az adott szerver, vagy sem, amelyről log és grafikus felületen keresztül is értesítve lesz a felhasználó.

Minden törlési variánsról elmondható, hogy valid (Minden mentés megtörténik a módosítások után, nincs validálatlan és nem mentett WorkItem.), a TFS szervert használható állapotba tartva végzik el a szükséges törléseket, fokozott ellenőrzések mellett. A grafikus felületen is több lépcsőben kell megerősítenie a felhasználónak a törlési szándékát, és csak azon felhasználók képesek törlést vagy feltöltést indítani, ha az adott felhasználó Microsoft fiókja rendelkezik a szerveren megfelelő jogosultságokkal.

Minden törlés végrehajtásról a beállításokban megadott címre jelentés küldés történik!

Havi Fájlkezelés

A fájlkezelés biztosítja a grafikus felületen az áttekintések, és „ellenőrzések" egy lehetőségét. A Fájl menüpont almenüpontjaiban szereplő gombok, az aktuális havi fájl megnyitása, ha van és egyéb külső fájl megnyitása az alkalmazáson belül biztosítja ezen áttekintési lehetőségeket.

Az aktuális fájl betöltése esetén az „InformationParser" osztály két metódusán keresztül történik a végrehajtás. Az aktuális fájl megnyitása előtt a vezérlő „GetUpToDateFileCSV" metódusa fut le az aktuálisan csatlakozott TFS szerver

TeamProjectjének a nevével. Így ellenőrzésre kerül, hogy az adott TFS szerver ezen TeamProjectjén történt-e már automatizált vagy grafikus felületről indított feltöltés, amikor létrejöhetett az aktuális havi fájl a feltöltött WorkItem információkkal. Ezen fájlok alapértelmezetten az alkalmazás mappájában jönnek létre, maga a keresés is alapértelmezetten úgy van megvalósítva, hogy ezen mappamozgatások esetén semmilyen probléma ne történjen. (AppDomain.CurrentDomain.BaseDirectory) A metódus vagy, üres stringgel, vagy az aktuális havi fájl path információjával tér vissza. Ezen path elérés kerül átadásra a második metódusnak, ami a „GetFileContent", amely „FileMode.Open", „FileAccess.Read" kapcsolókkal ellátva olvassa el a fájlt és adja át a tartalmat a nézet rétegnek, hogy az megjeleníthesse. Így elkerülve a különböző folyamatok által „megnyitott, tartott" fájlok olvasási, vagy megjelenítési problémáját.

Egyéb külső fájl megnyitása az alkalmazáson belül. Ezen metódus fő létjogosultsága az, hogy ne kelljen esetleges korábbi és egyéb fájlokat Fájlkezelőben, vagy egyéb eszközökkel keresgetnie a felhasználónak, hanem „instant" egy helyről meg lehessen nyitni és megtekinteni a tartalmakat. Főként a korábbi havi, már archív fájlok megnyitására történt meg a funkció beépítése. Megvalósítása csak úgy, mint az aktuális fájl megnyitásakor az „InformationParser" osztály „GetFileContent" metódusát használja. Amelynek a paramétere ez esetben a Nézettől származó, a felhasználó által kiválasztott fájl path, amit az „OpenFileDialog" osztály segítségével tud megadni.

Loggolás

A loggolás menüpont szintén három gombon keresztül nyújt szolgáltatásokat a grafikus felületen. Cél az aktuális és korábbi logfájlok egyszerű, és gyors elérése, megtekintése, akár egy művelet futtatása után rögtön az alkalmazáson belül, így semmi keresgetés, vagy egyéb alkalmazás nem szükséges. (A beállításokban adjuk meg a pontos helyet, hogy hova szeretnénk, ha elkészülnének ezen logfájlok, minden futást követően.)

Az aktuális log megnyitása, az egyik legegyszerűbb művelet. Itt a Nézet réteg csak lekérdezi az aktuálisan létrehozott logfájl nevét és elérését, ami alapján megjeleníti a tartalmat.

Nem aktuális logfájl megnyitása az alkalmazáson belül. Ezen funkció esetén a fájl külső megnyitáshoz hasonló módon tölthetünk be „OpenFileDialog" osztály segítségével korábbi logfájlokat, amelyeket meg szeretnénk nézni. Az „InformationParser" osztály „GetFileContent" metódusa kerül használatba ez esetben is paraméterként pedig a korábban említett „OpenFileDialog" – on keresztül a felhasználó által kiválasztott fájl

path -ot kapja meg. Kiemelendő, az aktuális log fájlt nem lehetséges betölteni ezen módon. Ekkor a felhasználó egy figyelmeztetést kap, hogy ez az aktuális futásé, és használja a korábbi aktuális log megnyitás funkciót.

Külső programon keresztül történő aktuális log megnyitása. Ezen funkció számos kérdést vethet fel, hogy miért is kapott egy külön dedikált gombot egy ilyen funkció az aktuális nézeten. Ezen funkció szeretne egy lehetőséget biztosítani esetleges „kedvenc" szövegszerkesztőnk használatához, és gyors megoldást biztosítani esetleges karakterkódolások esetén jelentkezhető problémák megoldására. A működés egy egyszerű Process.Start híváson keresztüli alapértelmezett „Notepad" indítás. Összességébe főként a karakterkódolások, és nyelvi problémák kiküszöbölése miatt került be a funkció, ha egy olyan országban történik az alkalmazás felhasználása, ahol ezen említett problémák kérdéseket vethetnek fel.

Maga a loggolás megvalósítása. A „CustomTraceLogging" mappa a modell réteget és a vezérlőt tartalmazó projektben található. Ezen belül van az implementációt tartalmazó, és a teljes logging rendszert megvalósító kettő osztály, a „TextLogTraceListener" és a „Logger". A „TextLogTraceListener" osztály a „TextWriterTraceListener" osztálytól öröklődik. Ezen osztály valósítja meg „override" metódusokon keresztül az alapvető logfájlba való írást és az új sorba írást, zárást. Az osztály egy „StreamWriter" objektumot példányosít kettő paraméterrel, egyik a fájl path, másik egy logikai igaz a fájl további írását lehetővé téve. Ez az osztály kerül felhasználásra a „Logger" – ben. A „Logger" osztály konstruktorában kerül felhasználásra a beállításokból származó, a felhasználó által megadott path, a logfájl kívánt nevével. Ezen adat lesz kiegészítve minden futtatás esetén egy időbélyeggel a megfelelő áttekinthetőség miatt. A „TextLogTraceListener" osztály „WriteLine", „Close" és „Flush" metódusai kerülnek felhasználásra elsősorban a „Logger" -ban. A fájlba írása ezen „TextLogTraceListener" által biztosított kiírásnak ad első sorban egy formatot ami a dátum, az adott log sor típusa (INFO vagy ERROR) és maga a kiírandó szöveg. Ez kerül a teljes alkalmazás során felhasználásra. Ezen kiírás formatot rögzítő metódus neve, „WriteEntry", ez a privát metódus kerül hívásra log írás esetén.

Maga a logolás használata. Egy Info metódus szolgálja azon feladatot, hogy a korábban említett formot megadó „WriteEntry" privát metódus hívással, közlő információt rögzíthessen a logfájlban (fejlesztő által megadott szöveg), emellett van kettő errort, hibát közlő metódus lehetőség, paraméter túlterhelés lehetőségének biztosításával.

Az egyik esetén a fejlesztő adhatja meg a hiba szövegét, másik esetben pedig a kivétel (Exception) szövege kerül a logba.

Mail Jelentés

A program számos alkalommal küld jelentést automatizált feltöltésektől minden törlési változat használatáig. A kulcs itt az információ gyors és megfelelő áramlásának fenntartása, és ez által esetleges céges környezetben való további munkák elősegítése, hatékonyság növelése. Hiszen automatizált futtatás esetén minden hónapban adott időpontra megérkezik a jelentés további (esetleges) adminisztratív munkára. Törlés esetén pedig maga a funkció magas felelőssége, fontossága váltja ki ezen jelentésküldéseket.

Az automatizált feltöltés jelentéstovábbítása mellett (Értesítés az aktuális friss havi fájlal.) és a törlési funkcióknál taglalt minden nemű művelet végrehajtása esetén küldésre kerülő jelentés, e-mail mellett, a grafikus felület „Info" menüpontja is kettő funkciót kínál üzenetküldésre. Ezen funkciók az „Info" menüpont általános tájékoztató szerepe mellett, pedig a „Contact" és a „Bug Report", amely hozzátartozik egy alkalmazás karbantartási és esetleges egyéb általános kapcsolatfenntartási részéhez, ezért is került ezen menüpont „perkeibe" eme kettő funkció.

„Contact" – Kapcsolatfelvétel esetén a nézet réteg megfelelő mezőinek kitöltését követően van lehetőség üzenetünk elküldésére. Ezen küldési funkciót is az „InformationParser" vezérlő réteget ellátó osztály egyik, jelen esetben a „SendMail" metódusa látja el. Amelyen belül a „MailSender" osztály „SendMail" metódusa kerül hívásra. Minden üzenet küldés az alkalmazáshoz tartozó e-mail címen keresztül kerül elküldésre, így a grafikus felületen megadott név és e-mailcím ezen üzenet része lesz, az üzenet szintúgy a beállított e-mail címre kerül küldésre, így biztosítva, hogy a cég megfelelő módon felvehesse a kapcsolatot a kiadóval, vagy a szoftver license partnerével. A „SendMail" metódus esetén a paraméterlistában meg kell adnunk a mail címet, ahova továbbítani szeretnénk levelünket (jelen esetben ez a beállításokból kapott érték), a tárgyat, a levél „törzsét", a szöveget és az esetleges csatolmányfájlt (fájl path). Természetesen a csatolmány lehet null is, amikor nincs semmi küldendő egyéb fájlunk, mint például a törlés jelentések esetén is (automatizált üzenetküldés a törlés használatát követően azonnal). Mindezt követően a „SendMail" a paramétereknek megfelelően összeállít egy „MailMessage" objektumot, szükség esetén „Attachment" objektumot is, amit csatol a „MailMessage" objektumhoz. Ezek után egy „SmtpClient" példányosítás és

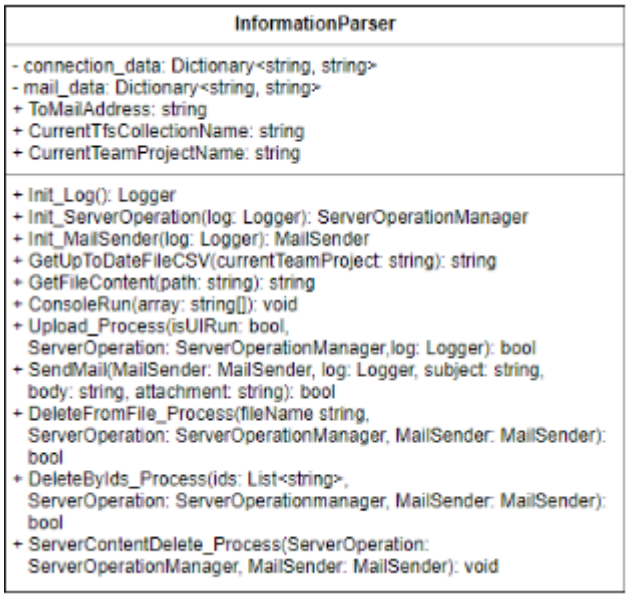
egy „NetworkCredential” beállítása történik, majd a levél küldése mindezek alapján. A megadott adatok közül a Nézet réteg ellenőrzi a felhasználó által megadott e-mailcím helyes formátumát egy Regexen keresztül.

A „Bug Report” – Hibajelentés. Esetén a megoldás menete teljesen megegyezik a „Contact” – Kapcsolatfelvétellel. A fő különbség a tárgy kiválasztásánál jelentkezik az aktuális Nézet rétegen. Mivel itt nem szövegesen megadható, hanem három megadott opció közül kell választani a megfelelőt üzenetkezelés és rendszerezés miatt. („Function Bug”, „UI Bug”, „Other Bug”) Ezalapján a nevet, az e-mail címet és magát az e-mail „törzsét”, a szöveget kell megadni a megfelelő tárgyválasztó gomb megnyomásával. Így ezen információkkal kerül meghívásra az „InformationParser” -nek a „SendMail” metódusa.

Controller funkciója

A „Controller”, avagy a Vezérlő. Ahogy korábban is említésre került az MVC (Modell – Nézet - Vezérlő) architektúrának megfelelően történik meg az alkalmazás rétegeinek elválasztása, és ezen funkciók kezelésének a folyamata.

A vezérlő adattagjai a modell feladatok, hívások inicializálások megfelelő módú felhasználását segítik a szükséges paramétereket is biztosítva. (A beállításokat tartalmazó config olvasásnak megfelelően, amelyet az egyedi config szerkezeti egység tesz teljessé.)



44. ábra
Controller UML diagramm.

A 44.ábrán látható diagramm alapján az első kettő privát adattag a „connection_data” és a „mail_data” fő feladatai a későbbiekben látható „Init_ServerOperator” és „Init_MailSender” metódusoknak biztosítják a megfelelő beállítási értékeket. A megfelelő config olvasáshoz a „System.Configuration referencia” (.dll) biztosítja a megfelelő, a feldolgozásához szükséges osztályokat, metódusokat, mint

például a „ConfigurationSection”, „ConfigurationManager”, „ConfigurationElement” és a „NameValueCollection”. Megjegyzésként megemlíthető, hogy a feltöltés esetén használandó (User Storykra és Taskokra) config olvasásánál az egyedileg implementált „PbisConfigSection” osztály használatos. (38. ábrán látható az UML diagrammja.)

Az azt követő három publikus adattag, („ToMailAddress”, „CurrentTfsCollectionName” és „CurrentTeamProjectName”) amelyek a különböző modell osztályok vezérlő általi inicializálásainál kapnak értékeket, a későbbi paraméterként való egyszerűbb használatra, felhasználásra. Amelyre példaként szolgálhat a nézet réteg feltöltés oldalán lévő szerver információk esete, ahol ezen adatok kerülnek lekérdezésre a nézettől a vezérlő felé, ami biztosítja az adattagokat a felhasználói felületen való megjelenítés számára. Vagy az adott vezérlő metódusain belül kerülnek felhasználásra, mint paraméter. („ConsoleRun” metóduson belüli jelentés küldést ellátó „SendMail” metódushívás esete.) A vezérlő szerepet betöltő osztály metódusai esetén az első három metódus az inicializálásokat végzi. (Init_ …) A szükséges modell osztályokat példányosítja, így egy esetleges új nézet réteg esetén is minden továbbra is a vezérlőn keresztül elérhető és példányosítható, nincs szükség egyéb osztályok felhasználására, mivel az „InformationParser” inicializálja szükség esetén a beállításokból származó adatoknak megfelelően. Így fenntartva az architektúra megfelelő réteges szerkezetének az elvét. Ezt követő metódusok a modell rétegen keresztüli szolgáltatások „összeállítását” és hívásait tartalmazza, amelyet a nézet réteg felhasználói interakciója hívhat meg a vezérlőtől, a vezérlőn keresztül.

Ütemezett felhasználás

Ütemezett felhasználás, avagy az automatizált felhasználás. A Modell szolgáltatásainak megvalósítása menüponton belül a „Feltöltés – Process Templates” almenüpont taglalja részletesen ezen megvalósítását a „ConsoleRun” metóduson keresztül történő „Upload_Process” és „SendMail” hívásokat, ezek pontosabb leírásait.

Maga a .bat fájlon keresztül történő megfelelő, a vezérlőt tartalmazó projekt indításakor, az adott projekt „Program” osztályán belüli „Main” metódusban történik meg ezen vezérlő - „InformationParser” inicializáció és ezen „ConsoleRun” metódus hívása a „Main” –től átadott „args” tömbbel, ami tartalmazza az adott kapcsolót (/config) és a beállításokat tartalmazó egyedi config fájlt, minden automatizált használati esetre külön-külön az áttekinthetőség és a szeparáció miatt.

View szolgáltatása (Reactive funkciók és értesítések)

A nézet réteg feladata a felhasználó számára egy egyszerű és „dizájnos” kezelő felület biztosítása a különböző feladatok végrehajtása érdekében.

Ahogy korábban is említésre került az aktuális nézet réteg egy „panelos” felépítés alapján készült. Így nem csak az alkalmazás nézet rétegét van lehetőség cserélni, hanem esetleges panelokat. Az alkalmazás grafikus felülete egy Winform alkalmazásként testesül meg, ahol a „Form1” tartalmazza az alapot, amire az „építkezés” megtörténik. Emellett még a főmenü sort és azon animált menüzárásokat tartalmazza a „Form1”, plusz kód szinten itt kerülnek felhasználásra a „Visual.Reactive.Winforms” NuGet package által biztosított lehetőségek.

A „Visual.Reactive.Winforms” segítségével van megvalósítva az adott főmenüpontok közötti váltáskor az „almenü és munkafelület panelon” („subMenu”) való, az adott főmenüponthoz tartozó almenü „Dashboardra” való váltás, és hasonlóan ezen módszer biztosítja emellett az adott munkaterületen a megfelelő oldalra való váltást is. Ezen oldalváltás az alapértelmezett munkaoldal megnyitását foglalja magában, a különböző almenüpontok esetén, ha azon funkciókhoz külön munkaoldalak tartoznak, mint jelen esetünkben a „Delete” szekció (Id alapján való törlések, Fájl alapján, Teljes törlés) és az „Info” szekció (Contact, Bug Report), ezek az oldalváltások már kizárólagosan a „subMenu” panelon belül vannak megvalósítva egy egyszerű „SetPage” hívással. A NuGet packegenek hála, így egyszerűen cserélhetünk panelt, hiszen nincs szükség jelentős kód módosításra, mivel a „Form1”, amely a főmenüt tartalmazza és a „subMenu” panel, amely a jelentős munkafolyamatokat tartalmazza kizárólag a „Visual.Reactive.Winforms” által megvalósított kód tartja fenn a kapcsolatot, nincs szükség semmi más módosításra, így jelentősen növekszik a modifikáció lehetősége. A „subMenu” panelon belül szintén fent van tartva ezen panelos szerkezet, így az információs szekcióktól az adott ProgressBar -okig minden nagyon egyszerűen és gyorsan cserélhető.

```
VSReactive<int>.SetState("menu", int.Parse(((Control)sender).Tag.ToString()));
VSReactive<int>.SetState("ContentControllerPages", int.Parse(((Control)sender).Tag.ToString()));
```

45. ábra
A „Visual.Reactive.Winforms” megvalósítás részlet.

A 45. ábrának megfelelően vannak implementálva az adott főmenü gombokhoz, a szükséges váltást kivitelező funkciók. Az első sora maga a főmenü adott menüpontjának gombjához szükséges „subMenu” -n lévő „Dashboard” almenü szekciót választja ki. Itt

szükséges az adott gombjait a főmenünek a „Data – Tag” -ein keresztül megszámozni nullától, ahány főmenügombunk van. A második sor a „subMenu” megfelelő munka szekció oldalát választja ki. A „Form1” így képes lesz megnyitni a szükséges „subMenu” „Dashboard” és „Munkafelületeket”, amelyek az adott főmenüponthoz tartoznak.

```
VSReactive<int>.Subscribe("menu", e => tabControl1.SelectedIndex = e);
VSReactive<int>.Subscribe("ContentControllerPages", e => ContentControllerPages.SelectedIndex = e);
```

46. ábra
A „Visual.Reactive.Winforms” megvalósítás további részlet.

A 46. ábrán látható a „subMenu” -n belüli kettő sor, amely kezeli ezen főmenü választásokat. Az első sor kezeli a megfelelő „Dashboard” almenüre történő váltást. A második sor pedig a megfelelő munkaoldalt nyitja meg az adott főmenüponthoz. Itt megemlítendő, hogy az aktuális „Pages” szekciót szintén meg kell számozni a korábbi „Data – Tag” -en keresztül. (nullától) Így lesz képes az adott „Dashboard” panel szekció váltások mellett az adott munkaoldalak váltására is az adott egyetlen főmenüpontunk a „subMenu” panelon.

Összességében, így egy nagyon magas „színvonalú” implementációt elérve biztosíthatjuk az interakciót, fenntartva a modularitást, hiszen nagyon minimális kódváltoztatás szükséges, ha nem nézet réteget (View), hanem csak az adott nézet rétegen szeretnénk nagyobb volumenű panelcserét végrehajtani. („Form1” és „subMenu”, ami tartalmaz minden egyéb szolgáltatást panelos bontásban, így ott nem jelentkezhet semmilyen speciális és kiemelendő téma a módosítás alatt.)

Az értesítési rendszer értesítési paneljai az „Alert” -ben kerül implementációra. Az adott értesítés esetén egy „AlertType” enum segítségével választhatjuk ki milyen értesítésre is van szükség. Minden az értesítésekhez tartozó implementációt, animációt, és időzítőket felölel ez az egy fájl. Az adott nézeten csak egyszerűen az „AlertCreation” publikus osztályszintű metódust szükséges használnunk, mint „Alert.AlertCreation” első paraméterként megadva az értesítés által megjelenítendő szöveget, második és egyben utolsó paraméterként pedig a megfelelő „AlertType” típust.

47. ábra
Lehetséges Alertek – Értesítések. (Success,Error,Warning,Info)

Adatszerkezet

A Team Foundation Server -ek által kezelt adatok, WorkItemek esetén egy speciális lekérdezési nyelvet szükséges használnunk, aminek a neve WIQL avagy Work Item Query Language. Mind kód szinten, mind az adott szerver webes kezelőfelületén, ennek megfelelően kell összeállítani a lekérdezéseket.

Összességében egy munka elem lekérdezése a WIQL lekérdezési nyelv segítségével definiálható. Ami egy SELECT utasítást tartalmaz, amely oszloponként felsorolja a visszaadni kívánt eredménykészlet mezőit. Az eredménykészletet logikai kifejezések használatával tovább lehet módosítani és meglehet adni különböző rendezési sorrendeket is.

A TFS szerver webes felületén keresztül egyszerű módon a megfelelő legördülő menükön keresztül kiválasztott mezőértékek segítségével lehet query-ket, lekérdezéseket létrehozni. Amelyeket a webes felületen lehetőség van menteni, esetleges későbbi, újra felhasználás céljából. Lehetőség van saját és megosztott lekérdezésként is menteni a szerveren.

Kód szinten történő lekérdezések esetén a különböző mezőneveket (refname), amelyeket a lekérdezéshez használnánk, a Process Template megfelelő „refname” -e alapján kell megadni. Kiemelendő, hogy a szintaxis és különböző operátorok szintjén vannak eltérések a nagyon hasonló és népszerű sql -hez képest, például az egyik legfeltűnőbb eset a „between” hiánya lehet. További ismeretek és sajátosságok áttekintésére ajánlott a Microsoft WIQL dokumentációja.

Tesztelés

A tesztelési tervet elsősorban kettő részre lehet bontani. Functional Testing és Non-Functional Testing. Így lefedve a teljes modern szoftver tesztelési modellt megbizonyosodva a termék megfelelő minőségéről és használhatóságáról.

A functional testing azon szoftver tesztelési típus, ahol az adott szoftvert a megadott specifikációk és elvárásoknak megfelelő funkcionalitását vizsgáljuk, hogy minden, aminek működnie kell megfelelően hibamentesen működik és minden specifikációban rögzített pontnak eleget tesz. Első fázis a Unit Testing. A tesztelés legkisebb egysége, ahol az adott metódusok funkciók kerülnek tesztelésre, hogy

megfelelően működnek-e, mint legkisebb elemek. A „TFS_ServerOperation.Tests” projekt foglalja magába ezen Unit test fázist. A tesztelés osztályról osztályra szeparálva történik minden esetben a szoftver működésének kulcsfontosságú metódusainak tesztelésével. Az adott projektben találhatunk egy mappát is „TestFiles” ahol, ha az adott metódusnak bemeneti fájlra van szüksége, mint például a „FileOperatios” osztály metódusai esetén, ahol a szoftver által előállított dokumentum olvasása és ezen írása kerül tesztelésre a végrehajtandó metódusok szintjén. Ezen tesztek az „Arrage – Act - Assert” tagolásban kerültek megírásra. Első teszt osztály a projekten belül a „ConnectionAdapterTest”, ahol a „ConnectionAdapter” osztály metódusai, elsősorban a „Connection” metódusa kerül tesztelésre, mind megfelelő lefutással, mind paraméter probléma, és mind esetleges hiányosság esetén történő futtatással. A következő, a példaként említett „FileOperationsTest” osztály, amely során a fájl létrehozása és feldolgozására írt metódusok kerülnek tesztelésre. Kiemelendő minden teszteset esetén törekedve volt a legszélsőségesebb és hibás hívások esetén való reakció feltérképezésére. Ennek köszönhetően a Unit Test fázist követően előfordult, hogy egyes metódusok hibakezelésén módosítani kellett eddigi rejtett és még nem implementált esetek megfelelő kezelése érdekében, amely a Regression Test fázisban került megvalósításra és újra tesztelésre, hogy ezen módosítások megfelelőek voltak-e, és semmi egyéb „Side effectet” nem okoztak-e az integrációs és system tesztek esetén, vagyis a program felhasználásának más területén. A „MailSenderTest” teszt osztály foglalja magában az email kezelési, létrehozási, attachment kezelési és küzdéshez szükséges metódusok tesztelését. A „ServerOperationManagerTest” a modell fő osztályának tesztje során pedig a funkciók metódusai, a különböző feladatok, amelyek részfeladatokká lettek bontva probléma dekompozíció elve alapján megfelelően működnek-e önálló egységként is, hogy a megvalósítandó integrációs esetben ezen metódusok önmagukban biztosan megfelelően ellátják a feladatukat. Végül pedig a „InformationParserTest” tesztosztály kerül megemlítésre a vezérlő szerepet betöltő osztály metódusainak megfelelő működésének a tesztelésével, amely főként az integrációs tesztekben vállal nagyobb szerepet a különböző modelltől való szolgáltatás hívások esetén megfelelően működnek-e azon metódusok, amelyek teljesítik ezen funkciók specifikáció és elvárás szerű végrehajtását.

Integrációs teszt fázis. Integrációs tesztek foglalják össze egy csoportba a korábbi Unit test során említett metódusokat, amelyek az adott szolgáltatás végrehajtásáért felelnek, így tesztelve a különböző funkciók megfelelő működését, mint például a vezérlő által a nézet réteg felé biztosított „Upload_Process” metódusa, amely több metódus hívást

magában foglalva végzi el a TFS szerverre a feltöltést, szükség esetén az archiválást és az annak megfelelő fájlkezelés műveleteit. Az integrációs teszteket a „TFS_ServerOperation.IntegrationTests” project foglalja magába, ahol a korábban említett „Upload_Process” mellett az inicializáló – config feldolgozó metódusok és a különböző törlési metódusok, funkciók is tesztelésre kerülnek, mind megfelelő végrehajtható állapotban, mind hibás esetekben egyaránt.

System testing esetén, vagyis a teljes működés egyben való tesztelésére a Non-Functional testing leírásánál részletesebben kifejtésre kerül. Ami az MTM (Microsoft Test Manager) -be integrálható és végrehajtható, a jövőben akár automatizálható teszteseteken keresztül kerül végrehajtásra. Interface testingre nem került sor.

Regression testing, ahogy a Unit test fázisnál is említésre került, azon egysége a tesztelési fázisoknak, ahol az esetleges hozzáadott új funkciók, feature -ök és az esetleges változtatások kerülnek tesztelésre amit jelen esetben a Unit test fázis eredményezett. Bizonyos metódusok kivételkezelésének módosítása és esetleges egyéb paraméter értékek plusz külön ellenőrzése nagyban segítik a metódusok működési hatékonyságát. Hiszen kivételek kiváltását lehet elkerülni ezen extra ellenőrzésekkel és maga a metódus végrehajtásának idején is javítanak ezen módosítások, amelyek a Unit test fázis során leltek napvilágot. Fontos tényező volt ezen tesztelési periódusban, a további metódusok és integrációs tesztesetek újbóli áttekintése, hogy semmilyen rossz kölcsönhatást nem vált-e ki más területén a kódnak, ezen eszközölt módosítások valamelyike.

A végső eleme ezen Functional testing egységnek pedig a UAT (User Acceptance Testing). Ahol személyesen a dokumentációban rögzített specifikációknak, diagrammoknak és képernyőterveknek megfelelően történt egy áttekintés, hogy minden megfelelő funkció implementálásra került-e, és hogy „én” mint ügyfél elégedett vagyok-e a látottakkal.

A Functional testing során a „BlackBox” test elv került felhasználásra, vagyis a specifikáció alapján készülnek a tesztesetek. (Leggyakoribb formája, hogy egy adott bemenetre tudjuk, milyen kimenetet kellene adni a programnak. Lefuttatjuk a programot a bemenetre és összehasonlítjuk a kapott kimenetet az elvárttal.) Amire említés is kerül a Unit Testing bekezdés esetén mivel „Arrage – Act - Assert” tagolásban kerültek megírásra ezen tesztesetek. Teszteket TFS, Azure vagy Travis CI pipeline-on keresztül futtathatjuk buildek esetén is természetesen.

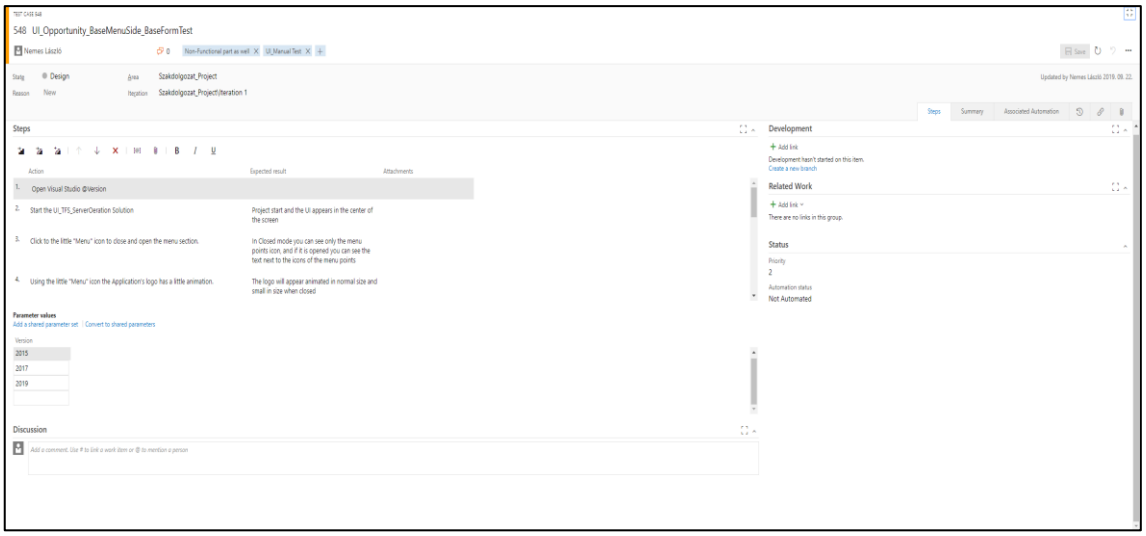
Összességében a Functional testelés során a tervezettnek megfelelően ellenőrzésre kerülnek a legkisebb felépítési egységektől a teljes rendszering a funkciók, hogy miket kell a specifikáció és elvárások szerint „tudniuk, leírniuk” a funkcióknak.

Második nagy fázisa a tesztelési tervnek a Non-Functional Testing. Amely tesztelése során olyan pontokra történik meg a kitérés, mint a performance, olvashatóság, felhasználhatóság, végig szem előtt tartva a felhasználói elvárásokat. A tesztelés során a „WhiteBox” test elv kerül előtérbe. (A fehérdobozos tesztelést strukturális tesztelésnek is nevezzük, mert mindig egy már kész struktúrát, pl. program kódot, tesztelünk.)

Ezeknek megfelelően a tesztelés során olyan egységeket kell végig járni, hogy megfelelő-e a jelenlegi User interakció, megfelelő-e a szoftver felhasználása az első indítástól kezdve, egyszerűen ki lehet igazodni a szoftver pontjai között, mindemellett megfelelő a performance és gördülékeny felhasználást biztosít-e a teljes szoftver mindennapos használata. (A Non-Functional testing a Functional testing után következik.)

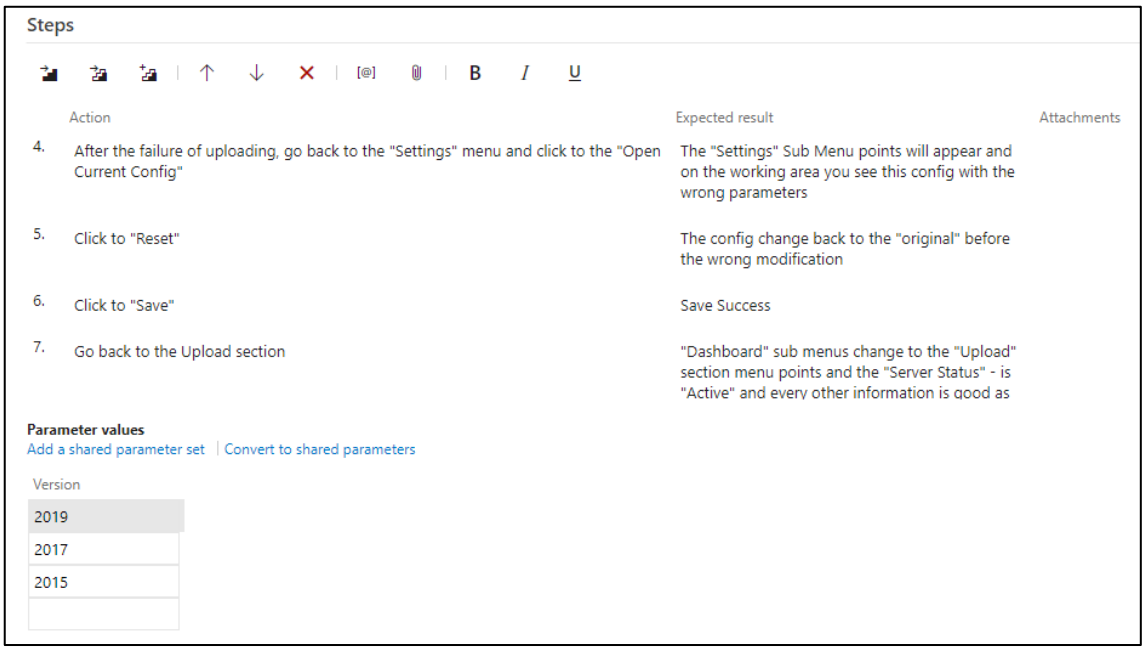
A Non-Functional testing végrehajtására az MTM (Microsoft Test Manager) -be használható, vagy adott Visual Studio Test Manager bővítményével (Visual Studio Test Professional subscription része) elérhető, az adott aktuális, vagy test TFS szerveren is létrehozható és tárolható Test Case -ek segítségével történik. A mellékleten ezen Test Case leírások, adott lépések és a hozzájuk tartozó elvárt eredmények megtalálhatók, így akár azon fájl segítségével bármely TFS szerveren létrehozhatjuk egyszerűen a szükséges Test Case egységet.

A legfőbb indok, ezen MTM Test Case használat mellett, maga a szoftver, (MTM vagy Test Manager) amely egy környezetet biztosít ezen tesztek végrehajtására, amin egyszerűen nyomon követhető, hogy melyik test melyik lépésénél van, vagy lehet probléma, és a tesztelő által megjegyzésben rögzítheti is ezen tapasztalatait. Ezáltal egy teljes felhasználás keretein belül, minden egyes funkció több felhasználási irányból megközelítve bizonyíthatja, hogy teljesen működőképes. Mindemellett egy felhasználói és felhasználási élményt is biztosít kiadás előtt, a fejlesztő vagy az aktuális fejlesztő csapat számára, akik akár service feladatokat is elláthatják a szoftverrel kapcsolatosan kiadást követően.



48. ábra
TFS Szerveren lévő Test Case.

A 48. ábrán látható egy TestCase WorkItem, ami mind TFS szerveren, mint MTM szoftverből létrehozható. Ezen WorkItem tartalmazza magukat a lépéseket, a prioritást, és hogy automatizált vagy szabad kézi végrehajtás szükséges-e.



49. ábra
Test Case Steps.

A 49. ábrának megfelelően épülnek fel a teszteset lépései, („Steps”) ahol látható az adott „Action”, teendő és az azt követő elvárt eredmény („Excepted result”), ha ez teljesül, akkor az adott lépés helyes, ellenkező esetben a tesztet félbe kell szakítani és a tesztelőnek fel kell vennie az adott lépés megjegyzéséhez a problémát.

Work Item...	Title	Assigned To	State
Test Case	UI_Delete Menu Point Test Delete From File	Nemes László	Design
Test Case	UI_Delete Menu Point Test Delete From File WrongFile	Nemes László	Design
Test Case	UI_Delete Menu Point Test Delete From File Alerts Tests	Nemes László	Design
Test Case	UI_Delete Menu Point Test Complete Delete WARNING	Nemes László	Design
Test Case	UI_Delete Menu Point Test Complete Delete Inactive Server	Nemes László	Design
Test Case	UI_File Menu Point Test Open Current File Previous Month File Does Not ...	Nemes László	Design
Test Case	UI_File Menu Point Test Open Current File After a Success Upload	Nemes László	Design
Test Case	UI_File Menu Point Test Open File Browse	Nemes László	Design
Test Case	UI_Log Menu Point Test Log External Start Test	Nemes László	Design
Test Case	UI_Log Menu Point Test Open Log Browse Test	Nemes László	Design
Test Case	UI_Log Menu Point Test Open Current Log	Nemes László	Design
Test Case	UI_Info Menu Point Test Contact Test	Nemes László	Design
Test Case	UI_Info Menu Point Test Bug Report Test	Nemes László	Design

50. ábra
Különböző Test Case-ek részlet.

Az 50. ábrán látható részlet a TestCasekről. Összességében 24 darab TestCase fedi le az alkalmazás teljes Non-Functional tesztelését, amelyek során minden menüpont minden egyes funkciója egy helyes használat mellett számos hibát kiváltó felhasználási módot is végigkísér a megfelelő hibakezelések és értesítések tesztelése érdekében. Így egy tiszta képet kapva minden funkció felhasználatóságáról, egyértelműségéről és megfelelő teljesítményéről. (performance- ről)

Performance teszt kifejtése. Az alkalmazás felhasználhatósága ki lett próbálva nagyobb adatmennyiség mellett is, szimulálva egy közepes 8-10 fős csapat általános munka WorkItemeivel, (Automatizált és grafikus felületről egyaránt.) és egy nagyobb volumentű, több TeamProjectet felhasználva történő feltöltést a grafikus felületen keresztüli szerverváltással. Ahogy várható volt nagyobb adatmennyiség alatt a feltöltés lassabb volt, de nem számottevő a performance visszaesés, ami a nagy mennyiségű WorkItem létrehozás és a megszaporodott linkelési műveletek miatt jelentkezik. De továbbra is felhasználóbarát, gördülékeny a felhasználása a funkciónak. (Természetesen, ha a WokrItem mennyiséget nagyon magasra emeljük (100-200+) jelentősebb lassulás is észrevehető a feltöltések esetén. De nagyon kicsi a valószínűsége ekkora mennyiség egyidejű feltöltésének, hiszen TeamProject szeparációban egyszerre ekkora mennyiség nem nagyon fordulhat elő.) Törlési műveleteknél nagyobb mennyiségű törlés esetén sem nagyon lehet érzékelni az adott implementálási mód mellett változást.

Tesztelési eredmény

Összességében a szoftver 1.0.0.0 kiadásának tesztelése sikeresnek értékelhető, hiszen mind a Functional mind a Non-Functional testek folyamán is megfelelő működést, hibakezelést produkált, a tesztek során jelentkező esetek javításait követően. A Functional testek esetén javasolt CI pipeline-t használni buildelések előtt, releasek előtt, és minden egyes újabb verziófrissítés, release előtt a fejlesztőnek, fejlesztő csapatnak mindenképp ajánlott a Non-Functional testelések (TestCase -ek) végrehajtása. Hibák természetesen előfordulhatnak bármikor egy szoftver életében, de a jelenlegi 1.0.0.0 verzió esetén nincs tudomás bármilyen problémáról a kiadás időpontjában. Minden teszt sikeresen lefutott, átment.

ÖSSZEGZÉS

A dokumentáció végére évre elmondhatjuk, hogy a specifikációnak és terveknek megfelelően sikerült elkészíteni a szoftvert, minden tervezett funkciójával együtt.

A szakdolgozat készítése, írása folyamán számos új ismerettel gyarapodtam ezen „specifikus” téma elemeket is beleértve. Mind a DevOps, mind a Fejlesztési folyamatokban sikerült elmélyülni és hiszem, hogy a jövőben ezen új ismeretek jelentősen segítik a további karrierfejlődésemet.

A dolgozatban mindvégig törekedtem újabbnál újabb technológiákat bevonni és megjelenítésben is napjainkban nagy népszerűségnek örvendő letisztult „flat design” -t követni, dokumentáció során pedig a lehető legrészletesebben átadni a szükséges ismereteket a témával kapcsolatosan.

Ábrajegyzék

Irodalomjegyzék, Linkek

Minden hivatkozott linket 2019. október 10 -én elértem.

- Microsoft Team Foundation Server dokumentáció: https://docs.microsoft.com/en-us/azure/devops/server/tfs-is-now-azure-devops-server?view=tfs-2017
- NuGet - Microsoft.TeamFoundationServer.ExtendedClient: https://www.nuget.org/packages/Microsoft.TeamFoundationServer.ExtendedClient
- NuGet - Visual.Reactive.WinForms: https://www.nuget.org/packages/Visual.Reactive.WinForms/