Pannon Egyetem

Matematikai Tanszék

Programtervező Informatikus BSC

SZAKDOLGOZAT

KPI\_Tool

Suborits Péter

Témavezető: Lipovits Ágnes

Konzulens:Both Botond

2018

Contents

[1. Felhasznált technológiák 4](#_Toc508275878)

[1.1 IMS Intergrity Client (MKS) 4](#_Toc508275879)

[1.2 Microsoft Excel-Visual Basic 6](#_Toc508275880)

[1.3 Visual Studio-C# 6](#_Toc508275881)

[1.4 Perl script 6](#_Toc508275882)

[2. Feladat ismertetése 7](#_Toc508275883)

[3. Felhasznált szűrések, és jelentések 8](#_Toc508275884)

[3.1 Code Review 8](#_Toc508275885)

[3.2 Code Review Report 9](#_Toc508275886)

[3.3 CVEH SC Release Task 9](#_Toc508275887)

[3.4 CVEH SC Release Task Report 10](#_Toc508275888)

[4. Létrehozott fájlok bemutatása 12](#_Toc508275889)

[4.1 Pre-Analysis 12](#_Toc508275890)

[4.2 Code Review 12](#_Toc508275891)

[4.3 Release Task 12](#_Toc508275892)

[4.4 Expired Items 12](#_Toc508275893)

[4.5 Bonus Files 12](#_Toc508275894)

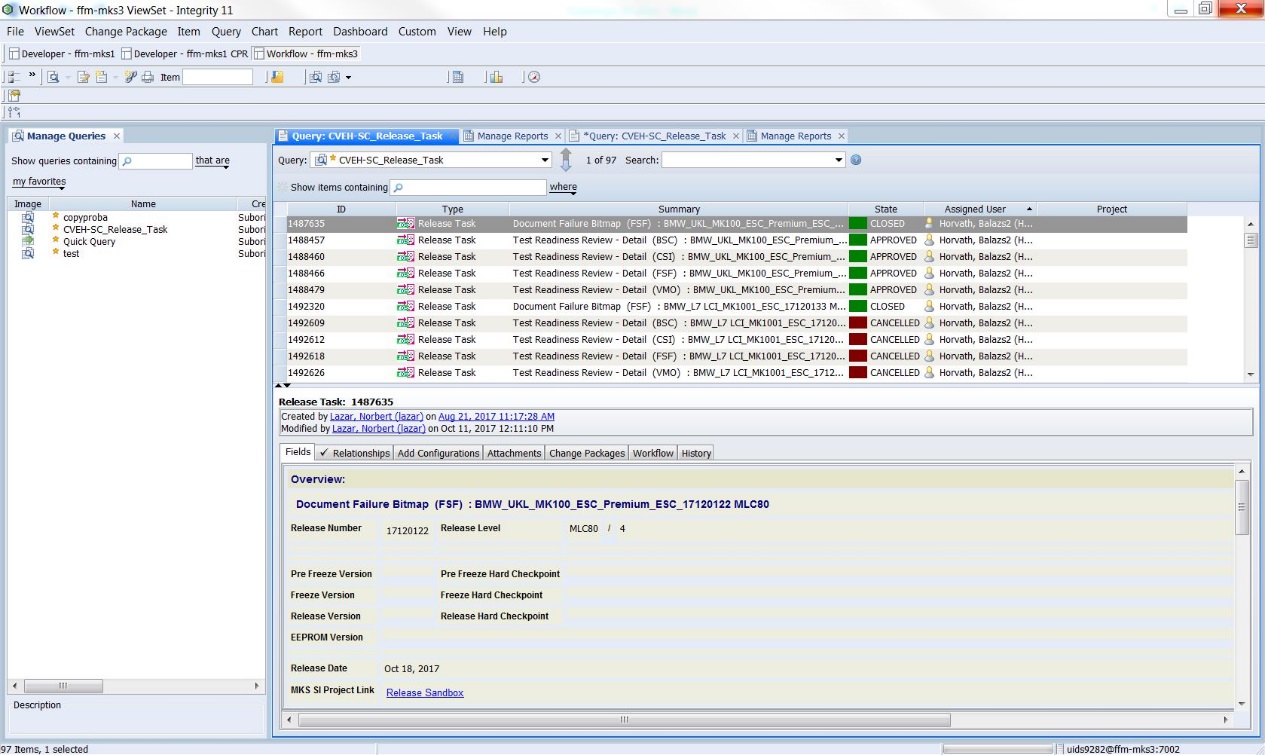
[4.6 Personal Results 12](#_Toc508275895)

# Felhasznált technológiák

## 1.1 IMS Intergrity Client (MKS)

A szoftverfejlesztők egy verziókövető rendszert (MKS) használnak a dokumentációk és programkódok tárolására. Továbbá ezt az egész rendszert nem csak itt a magyarországi vállalatnál használják, hanem a többi lokáción is.

A rendszert a PTC fejlesztette ki még 2001-ben. Java nyelven íródott. Többféle program is tartozik az Integrity családba, a cégnél a PTC Integrity Lifecycle Manager-t használják, mivel ez alkalmas a verziókövetésre és tárolásra.



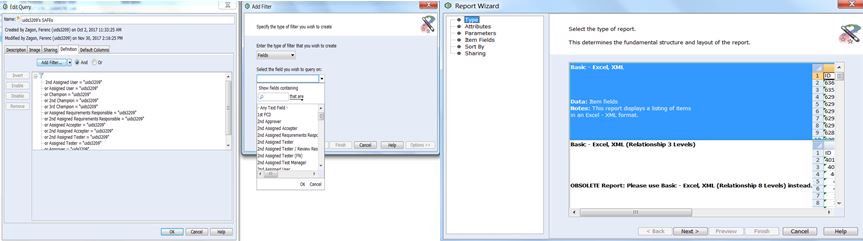
*1.1 ábra*

A rendszer tartalmazza a felhasználói felületet, amely az 1.1-es ábrán látható és egy kis gyakorlással egyszerűen lehet használni. Két szervert használnak a projekt munkával kapcsolatosan az adatok tárolásához, az egyiken vannak a dokumentációk, a másikon pedig a programkódok, módosítások. Ebből a szempontból is érdekes lesz az adatok megfelelő módon történő kinyerése, mivel mind a két szerverrel fel kell építeni a kapcsolatot.

Nekem csak arra lesz szükségem, hogy a szükséges fájlokat kiexportáljam és feldolgozzam, ezért előreláthatólag csak két funkcióját fogom használni a verziókövető rendszernek. Az egyik ilyen funkció a lekérdezés (query) lesz, ami a rendszeren belül egy szűrő. Logikai műveletekből épül fel amit tudunk kombinálni a mezőkkel, hogy milyen sorrendben keresse a megadott feltételeket. A program szempontjából ezeknek a szűréseknek félig dinamikusaknak kell lenniük, mivel lesz bennük, ami fix, és lesz, ami a felhasználótól fog függni, ilyen például hogy milyen időintervallumon belül szeretne keresni. A felhasználok, mint korábban említettem a csoportvezetők lesznek, és a dinamikusság alatt arra kell gondolni, hogy minden vezetőhöz más-más csoport vagy csoportok tartoznak, ezáltal más fejlesztők tartoznak hatáskörük alá.

A másik funkció a generálás (report) lesz, amely lényegében az exportálásért felelős. Itt meg tudjuk adni beállításként, hogy milyen formátumba szeretnénk exportálni (xml, html), formázást tudunk adni, számolt cellákat tudunk hozzáadni, de ezekre nagyon kell figyelni, mivel a futási időre nagy hatással van. Például ha a számított celláknál nem a megfelelő képleteket használjuk, akkor ez a lefutás megszakításához is vezethet, mivel a rendszernek van egy időkorlátja, amit ha túllép, automatikusan leállítja a futó folyamatot. Tehát nagyon fontos az optimalizálás, mivel ha csak egy report nem fut le az már hatással lehet az egész kiértékelésre, hibás adatokat, információkat kaphatnak a csoportvezetők.

Az adatok mappás elrendezésben vannak jelen a rendszerben, nekem ezeknek a mappáknak az információtartalmát kell szelektálnom és a szűréseimnek megfelelő mappa információit kiexportálni. Alább az 1.2-es ábrán látható egy-egy példa a lekérdezésre (query), és a generálásra (report), hogy a készítésük során mennyi funkció közül lehet választani. Ezt természetesen majd a program fogja átadni parancssoros formában.



*1.2 ábra*

## 1.2 Microsoft Excel-Visual Basic

A Microsoft Excel 2007-es verziójában kezdtem el a fejlesztést, viszont időközben a cég váltott a 2013-as verzióra. Az átállás során nem volt verziókülönbségbeli probléma az addig megírt kód ugyanúgy futott az újabb verzióban is. A választás azért esett az Excel-re mivel a programnak egy részfeladata már meg volt írva Visual Basic-ben, csak éppen nem működött megfelelően. Eleinte ezt kellett volna orvosolnom, majd utána a vezetők kitalálták, hogy akkor legyen egy külön makró, amelyet alább a 2. pontban részletezek. Továbbá én is ezt tartottam az ésszerű választásnak, mivel Excel fájlokkal kell dolgoznom. Az Excel-en belül is inkább a fejlesztőeszközök menüpont alatt tevékenykedek, ezen belül is legtöbbször a beépített programban a Visual Basic-ben. Ez többek között azért is jó, mert a beépített függvények mellett ugyanúgy tudom használni a programozásban tanult dolgokat, mint például a tömbkezelés.

## 1.3 Visual Studio-C#

Mivel az Excel-ben a felhasználói felület készítése elég szegényes, és mégiscsak elegánsabb egy külön futtatható fájl készítése, ezért döntöttem a c# programozási nyelv mellett, amelyben a kezelőfelület fog készülni. Fejlesztési környezetnek a Visual Studio-t választottam, mivel egy kevés tapasztalatom már van benne és elég sok lehetőséget is tartalmaz. A 2013-as verziót fogom használni itt is. Visual Studio-n belül egy WindowsFormsApplication-t fogok készíteni, amellyel tulajdonképpen egy asztalról futtatható grafikus kezelőfelületű alkalmazást kapok.

## 1.4 Perl script

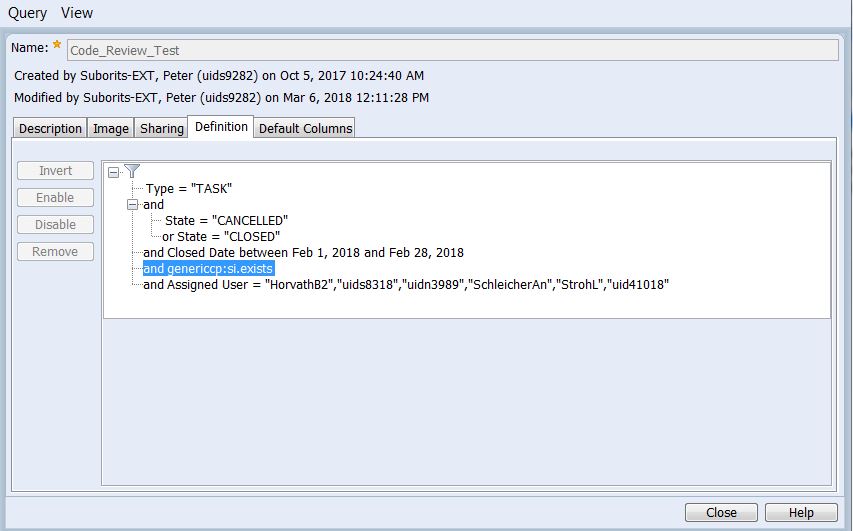
Egy külön programozási nyelv, amelyet Larry Wall fejlesztett ki még az 1980-as években. Stílusában nagyon hasonlít a C nyelvhez. Mivel ez a script is elengedhetetlen a program megfelelő működéséhez és viszonylag készen kisebb hibákkal kaptam meg, ezért ezeket a kisebb hibákat javítva építettem bele az alkalmazásba. Notepad++ használtam fejlesztési környezetként.

# 2. Feladat ismertetése

Egy olyan program/ script készítése, amellyel automatizálható, különböző formájú excel-ek kiértékelése és egy összesítő táblázatba való átírása. A rendszernek elsősorban képesnek kell lennie a félévi táblázatokat létrehozni, illetve adott féléven belül havi futtatáskor ellenőrizni azt, hogy van-e új tag valamely csoportban és, ha igen hozzáfűzni a táblázathoz. A dolgozók információit egy webes felületről lehet elérni amely alapján ellenőrizhető lesz az új tag kérdés. A kiértékelni kívánt dokumentumokat/adatokat egy verziókövető rendszerből kell lekérdezni szűrések alapján, majd letölteni és egy excel táblázatba rendszerezni. A verziókövető rendszer egy API alkalmazás. A verziókövető rendszerhez személyes azonosítás szükséges (felhasználónév, jelszó), amit a programnak kell átadnia, továbbá a szűrési feltételek és a generálni kívánt jelentéseket is a táblázatkezelő szoftverével kell továbbítani. Néhány dokumentum esetében további lekérdezés is szükséges, majd ez alapján formázni a belőle készített excel fájlokat. Havi szinten szeretnék futtatni a programot, tehát dokumentumok is havi bontásban generálódnak, és a kiértékelésnek is így kell történnie, továbbá az összesítő táblázatba is a megfelelő hónaphoz és személyhez kell kerülnie az adatoknak. Kell egy beépített modul is ezekhez az összesítő táblázatokhoz, amivel félévek végén tudnak generálni személyre szabott féléves összesítő táblázatokat, illetve pdf-eket. Havonta körülbelül 15-20 excel-el kell dolgozni, és ezeket az excel-eket előtte legenerálni. Ez a folyamat manuálisan minden egyes csoportvezetőnek havi szinten körülbelül 5-6 óra munkával járt, terveim szerint ezzel a script-el 3-4 perc lesz mindössze, ez is többnyire a verziókövető rendszerrel való kommunikáció miatt. A megvalósítást a Microsoft Excel, Visual Basic környezetében tervezem.

# Felhasznált szűrések, és jelentések

## 3.1 Code Review



*1.3 ábra*

Az 1.3-as ábrán látható, hogy a Code Review-hoz milyen szűrések kellenek hogy a megfelelő adatokat kaphassuk. Kell típusra szűrni, ahol csak a „TASK” típusú fájlok elfogadottak, a státusznak „CANCELLED” vagy „CLOSED” állapotban kell lennie. Továbbá kell a dátum amelyet a felhasználó a program indítása után ad meg hogy melyik hónapot szeretné futtatni, és az alkalmazás ebből számolja ki azt hogy mit kell átadnia, illetve hogy mely felhasználókra keressen, ezt megint csak a program számolja az alapján hogy éppen melyik csoporttal dolgozik. A dátumnál fontos még megemlíteni, hogy azt nézi a szűrő, hogy mikor került „CANCELLED” vagy „CLOSED” állapotba, tehát ennek a dátumára keres. A „genericcp” azt vizsgálja hogy létezik e checkpoint az adott dokumentációhoz. Ezek a feltételek és kapcsolatban vannak, tehát mindegyiknek teljesülnie kell.

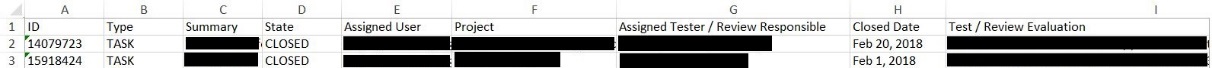
## 3.2 Code Review Report

A Report-nál azt kell elsősorban megadnunk hogy melyik szűrést akarjuk használni ami esetünkben a Code\_Review lesz. Továbbá be lehet illetve kell is állítani azt, hogy a generálás során elkészült fájlba milyen adatokat szeretnénk belerakni, amelyek fontosak lehetnek ahhoz, hogy a csoportvezetők is át tudják nézni az adott eredményeket, és ne csak két szám legyen ott.



*1.4 ábra*

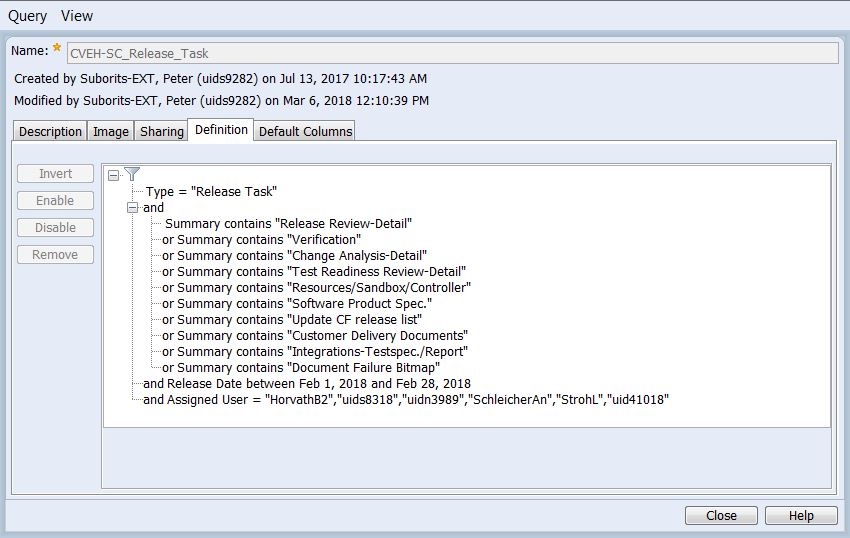
A Code Review Report-nál az 1.4-es ábrán látható adatok lesznek megjelenítve a fájlban amit legenerál, vagyis az azonosító, típus, összegzés, státusz, a task-hoz tartozó fejlesztő, hogy milyen projekt, ki tesztelte az adott task-hoz tartozó kódot, mikor lett lezárva a task, a teszthez tartozó komment. Alább az 1.5-ös ábrán látható a kész fájlba írt adatok.



*1.5 ábra*

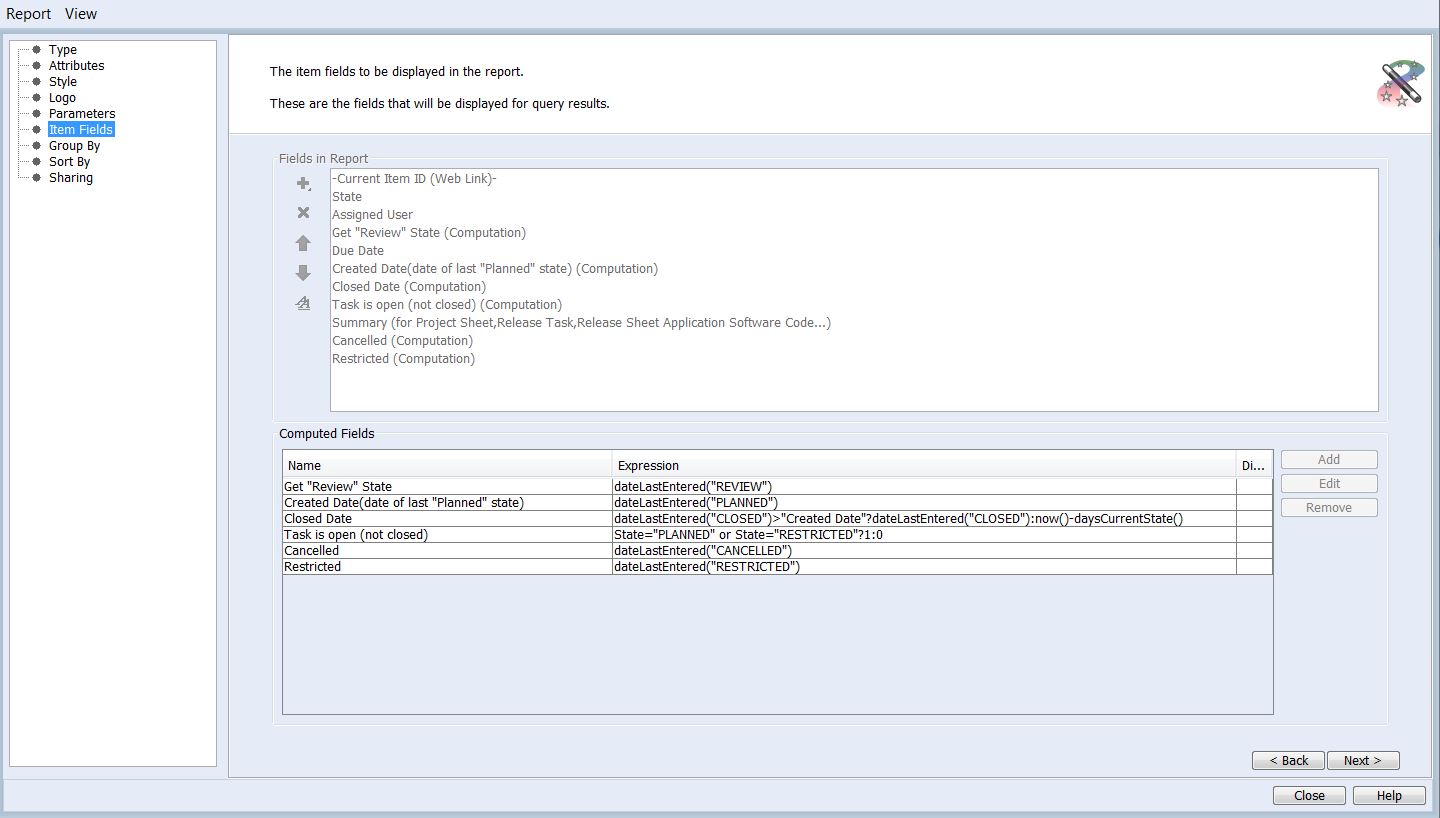
## 3.3 CVEH SC Release Task

Az alábbi 3.3-as ábrán látható, hogy milyen szűrési feltételeket kell alkalmazni a Release Task-ok lekérdezéséhez. Elsőként is a típusának „Release Task”-nak kell lennie. Az összegzés vagy összefoglaló résznek a felsorolt elemekből valamelyiket tartalmaznia kell, ahhoz, hogy megfelelhessen a kritériumoknak. Továbbá mint a Code Review-nál itt is kell neki dátumot adni, hasonló módon, csak itt a program azt a dátumot nézi amikor ki lett adva az adott task, illetve kellenek a felhasználók is ugyancsak hasonló módon mint a Code Review-nál.



*3.3 ábra*

## 3.4 CVEH SC Release Task Report



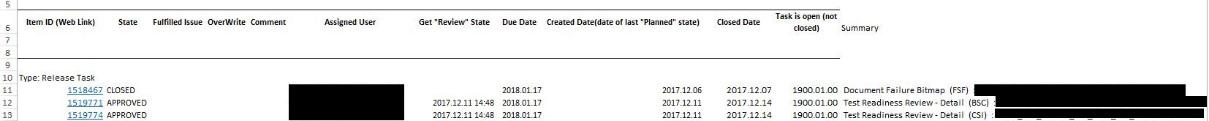
*3.4 ábra*

A fent látható 3.4-es ábrán a generáláshoz szükséges adatok, mezők, vannak felsorolva. A Code Review-hoz képest ez egy sokkal bonyolultabb és összetettebb generálás, mivel itt számított mezőket is kellett beletenni. Ez talán az egyik legkényesebb pontja a verziókövető rendszernek, mivel ezek a számított mezők nagyon időigényesek és meg kell válogatni, hogy milyen parancsokkal is szeretnénk a kívánt adatokat elérni, tehát fontos a jól optimalizáltság. Ezekkel a mezőkkel olyan adatokat tudunk elérni, amelyek nincsenek benne az alapból elérhető mezők közt, de különböző parancsokkal és kitételekkel előhívható.

A fájlba kiírt mezők pedig a következők:

* Azonosító, amely linkként is funkcionál
* Állapot, az adott task milyen státuszban van
* Felhasználó, amelyik fejlesztőhöz tartozik az adott task
* Mikor volt elbírálás alatt, ez egy dátumot ad vissza
* Határidő, mi az elvárt időpont, amire kész kell lennie az adott task-nak
* Mikor készült az adott task, dátumot ad vissza
* Mikor lett lezárva, dátumot ad vissza
* A task-ok nyitva vannak-e, ha igen akkor 1, ha nem akkor 0-t ad vissza itt a „PLANNED” vagy a „RESTRICTED” állapotokat fogadja el nyitottnak
* Összegzés
* Ha volt ilyen akkor mikor lett elvetve a task, dátumot ad vissza
* Korlátozott, ez azt jelenti, hogyha éppen nem tudtak valami más rész miatt haladni vele, akkor került ebbe az állapotba, dátumot ad vissza

A sok számított mezőre azért van szükség, mert későbbiekben az Excel és a makró ezekből határozza meg megadott feltételekből, hogy mi az, ami elfogadott és mi az, ami nem. Az alább látható 3.41-es ábra szemlélteti, hogy milyen a nyers fájl még a makró futtatása előtt.



*3.41 ábra*

# 4. Létrehozott fájlok bemutatása

## 4.1 Pre-Analysis

## 4.2 Code Review

## 4.3 Release Task

## 4.4 Expired Items

## 4.5 Bonus Files

## 4.6 Personal Results