|  |  |  |
| --- | --- | --- |
| oe_cimer_szines_print_res | **NEUMANN JÁNOS**  **INFORMATIKAI KAR** | NIK_cimer.jpg |

**SZAKDOLGOZAT**

|  |  |  |
| --- | --- | --- |
| **OE-NIK**  **2017** | Hallgató neve:  Hallgató törzskönyvi száma: | **Zalán Máriusz**  **T/003794/FI12904/N** |

Tartalomjegyzék

[1. Bevezető 5](#_Toc530393170)

[1.1. Szoftverek az oktatásban 5](#_Toc530393171)

[1.2. Az oktatásszervezés támogatása 5](#_Toc530393172)

[1.3. Szakmai oktatás támogatása 6](#_Toc530393173)

[1.4. Csapatmunka támogatása 6](#_Toc530393174)

[2. Célkitűzések 7](#_Toc530393175)

[3. Lehetőségek csapatmunkára 8](#_Toc530393176)

[3.1. moodle 8](#_Toc530393177)

[3.2. Git 8](#_Toc530393178)

[3.3. Trello 8](#_Toc530393179)

[3.4. TFS (Team Foundation Server) 8](#_Toc530393180)

[3.5. Google Drive 8](#_Toc530393181)

[4. Követelmény specifikáció 9](#_Toc530393182)

[4.1. A mozgatórugók 9](#_Toc530393183)

[4.2. A rendszer használói 9](#_Toc530393184)

[4.3. Hallgatók funkciói 10](#_Toc530393185)

[4.3.1. Szolgáltatások 10](#_Toc530393186)

[4.3.2. Adatmodell 10](#_Toc530393187)

[4.3.3. Folyamatábrák 10](#_Toc530393188)

[4.4. Oktatók funkciói 10](#_Toc530393189)

[4.4.1. Hallgatói munka ellenőrzése, jóváhagyása 10](#_Toc530393190)

[4.4.2. Kurzus nyitása, zárása 11](#_Toc530393191)

[4.5. Rendszerrel szemben támasztott követelmények 11](#_Toc530393192)

[5. A feladat problémáinak kiaknázása a jelenleg rendelkezésre álló eszközökkel 12](#_Toc530393193)

[5.1. Verziókezelők 12](#_Toc530393194)

[5.2. Verziókezelők típusai 12](#_Toc530393195)

[5.2.1. Elosztott verziókezelők 12](#_Toc530393196)

[5.2.2. Központosított verziókezelők 13](#_Toc530393197)

[5.2.3. Nyílt elosztott verziókezelők 14](#_Toc530393198)

[5.2.4. Zárt elosztott verziókezelők 14](#_Toc530393199)

[5.2.5. A verziókezelők és a megvalósítandó rendszer 14](#_Toc530393200)

[5.3. Agilis módszertanok 14](#_Toc530393201)

[5.3.1. SCRUM 15](#_Toc530393202)

[5.3.2. Kanban 16](#_Toc530393203)

[5.4. Agilis módszertanok eszközei 16](#_Toc530393204)

[5.4.1. Trello 16](#_Toc530393205)

[5.4.2. Redmine 17](#_Toc530393206)

[5.5. Az agilis módszertanok és a megvalósítandó rendszer 18](#_Toc530393207)

[5.6. Egyéb, ingyenesen elérhető közösségi eszközök 19](#_Toc530393208)

[6. Összefoglalás 20](#_Toc530393209)

[7. A rendszer elvi felépítése 21](#_Toc530393210)

[7.1. Oktatói funkciók 21](#_Toc530393211)

[7.1.1. Félév kezdés, adminisztráció 21](#_Toc530393212)

[7.1.2. Féléves munka, oktatás 21](#_Toc530393213)

[7.2. Hallgatói funkciók 22](#_Toc530393214)

[7.3. Felhasználókezelés 22](#_Toc530393215)

[8. Adatbázis-szerkezet 23](#_Toc530393216)

[8.1. Adattáblák 23](#_Toc530393217)

[8.1.1. Users 23](#_Toc530393218)

[8.1.2. Student és Teacher 24](#_Toc530393219)

[8.1.3. SessionGroup 24](#_Toc530393220)

[8.1.4. Team 24](#_Toc530393221)

[8.1.5. Semester 25](#_Toc530393222)

[8.1.6. Dictionary 25](#_Toc530393223)

[8.1.7. DictionaryType 25](#_Toc530393224)

[8.1.8. ServiceStore 26](#_Toc530393225)

[8.1.9. ServiceRequest 26](#_Toc530393226)

[8.1.10. ServiceTable 26](#_Toc530393227)

[8.1.11. ServiceTableField 27](#_Toc530393228)

[8.1.12. ServiceStoreParams 27](#_Toc530393229)

[9. A program felépítése 29](#_Toc530393230)

[9.1. WPF 29](#_Toc530393231)

[9.2. MVVM 29](#_Toc530393232)

[9.3. Data Access Layer 29](#_Toc530393233)

[9.4. Entity Framework 29](#_Toc530393234)

[9.5. Database First 29](#_Toc530393235)

[10. Tesztelés 30](#_Toc530393236)

[11. Továbbfejlesztési lehetőségek 31](#_Toc530393237)

[Irodalomjegyzék 32](#_Toc530393238)

[Ábrajegyzék 33](#_Toc530393239)

# Bevezető

## Szoftverek az oktatásban

Napjainkban az informatika által nyújtott lehetőségek egyre nagyobb szerepet játszanak a közoktatás területén. Mind a közép-, mind a felsőoktatásban megtalálhatóak ma már olyan szoftverek, melyek nemcsak az oktatók, hanem a tanulók munkáját is segítik, támogatják. Ezek a programok funkciójuktól, felhasználói körüktől, oktatási renszderben való elhelyezkedésüktől függően igen sokfélék lehetnek. Míg egyrészük a tananyag elsajátítását segítik, másrészük magát az oktatásszervezés területét támogatják, úgy mint a Neptun, az ETR vagy a különféle e-napló rendszerek.

Ezutóbbi rendszerek bevezetésével és használatával, a korábban papír alapon rögzített adatok, központosítva, elektronikusan kerülnek tárolásra, így az ezekhez történő hozzáférés több szempontból is előnyösebb, mint a régi, analóg tárolási módszerek esetében. Annak ellenére, hogy ezek a papír alapú módszerek természetesen nem kerülhetőek ki teljesen - hiszen sok helyen például az index fizikai példánya továbbra is megmaradt - a pénzügyek nyomon követése és kezelése egyszerűbb, az ügymenet sokkal gyorsabb és pontosabb, köszönhetően a rendszerekbe beépített automatizmusoknak és ellenőrzéseknek.

Ahogy azonban fentebb már említettem, a tantárgyak oktatásának informatikai támogatása is egyre jelentősebb lett az elmúlt években. Legfontosabb előnye azontúl, hogy a tananyagokhoz történő hozzáférés egyszerűbb, elsajátítása könnyebb; elősegíti, hogy az oktatás kielégítse azt az igényt, hogy a tanulók/hallgatók ne csak elméleti, hanem tapasztalati, empirikus úton is megszerezzék a munkaerőpiac által elvárt tudást. Ennek a gyakorlati alapú oktatásnak egyik módszere, hogy olyan feladatokat kell a diákoknak a félév során elvégezni, ami csapatmunkát igényel. Sok esetben azonban az ezen feladatokhoz használt szoftverek nem oktatási célra lettek kifejlesztve, mégis, mivel a munkahelyeken, éles környezetekben széles körben alkalmazzák őket, a gyakorlat központú oktatásban nemcsak, hogy jól használhatóak, hanem ezek segítségével a tanulókat fel lehet készíteni a jövőbeli, munkahelyi kihívásokra és feladatokra.

A mérnök informatikus és szoftverfejlesztői képzés területein ez a gyakorlatra épülő oktatás kulcsfontosságú. Esetükben, a tantárgyakhoz használt programok olyan követelményeknek kell, hogy eleget tegyenek, mint: az alkalmazott program szemléletesen mutassa be a felmerülő problémát, annak megoldási lehetőségeit, vagy, hogy a munkafázisok és azok eredményei jól követhetőek legyenek.

A szakdolgozatomban a fenti szempontokhoz kapcsolódó néhány termék elemzésével foglalkozom, és az ebből levont következtetések, tapasztalatok alapján a cél egy, az Óbudai Egyetemen oktatott, tipikusan csapatmunkát igénylő tárgyat támogató szoftver megvalósítása.

## Az oktatásszervezés támogatása

Az oktatásszervezés támogatása egyaránt megkönnyíti a hallgatók és az oktatók munkáját. Oktatásszervezés alatt értendő minden olyan tevékenység, mely nem közvetlenül a tantárgyi elvárásokhoz kapcsolódik, hanem a zavartalan és jól szervezett munkát teszi lehetővé. Az ezen szempontok által támasztott elvárások nagyban függnek például a résztvevők számától (állami szintű, intézményi szintű, kurzus szintű, stb.), a nyújtott szolgáltatások komplexitásától (egy kurzusfelvétel vagy vizsgajelentkezés nem igényel olyan bonyolultságot, mint egy gyűjtőszámla befizetés), a felhasználói körtől (szükséges-e akadálymentesítés, többnyelvűsítés).

A neptun és etr rendszerek a legismertebbek ezen a területen, illetve a középiskolai e-napló is idesorolható. Használatuk az elmúlt évtizedekben általánossá vált. Mindegyik hasonló funkciót lát el.

## Szakmai oktatás támogatása

A szakmai oktatás támogatása nagyban függ a szakterülettől. Ezek általánosan célszoftverek, vagy általános célú szolgáltatások, mint például a Google által nyújtott szolgáltatások. Azon szempontok, melyeknek meg kell felelni, rendszerint változnak attól függően, hogy milyen szakterületről van szó. Egy moodle, ami bár ötvözi a kétféle rendszer tulajdonságait, nem felel meg egy üzleti szimulációs szoftver elvárásainak, és fordítva. Megtalálhatóak még a piacon olyan szoftverek, melyek ingyenes vagy iskolai verziója, esetleg oktatási intézményi szerződés által elérhető változata segíti az oktatást, például Enterprise Architect, Visual Studio, Xilinx.

## Csapatmunka támogatása

Az egyéni munkához a Google szolgáltatásai és egy, a feladat típusához megfelelő segédszoftver általában elégséges megoldásokat nyújtanak. Egy több embert igénylő feladatnál viszont szükség van a csapattagok közti kommunikáció megvalósítására, valamint arra, hogy egymás munkáját nyomon lehessen követni, a felmerülő problémákat meg lehessen tárgyalni egy közös platformon, és hogy a csapattagok naprakészek legyen a feladat aktuális állapotával. Erre a célra a szoftverfejlesztésben számtalan lehetőség van, verziókövetésre ott a Git, a Microsoft TFS szolgáltatása, kommunikációra és feladatlebontásra használható a Trello vagy a Redmine. A Google drive szintén alkalmazható, bár a feladat komplexitásától függően ez veszít a használhatóságából.

# Célkitűzések

Szeretnék egy olyan rendszert létrehozni, amelyik alkalmas a Projektlabor szakirányos, és a Vállalati Információs Rendszerek Modellezése kerettantárgy támogatására, akár teljes vezetésére. A tárgyak során egy fiktív vállalat kerül modellezésre; a szakirányos tárgy keretein belül részletesebben, nagyobb hangsúlyt fektetve a tényleges működésre a nagyolt modellezéssel szemben. Ezeknek a tárgyaknak a célja, hogy a hallgatók megismerkedjenek egy vállalat tényleges belső szerkezetével, annak működésével, miközben aktív csapatmunkában vesznek részt. Az általam készítendő rendszernek alkalmasnak kell lennie mindazon eszközök kiváltására, melyekkel a hallgatók és oktatók dolgoznak, miközben a munkát nem akadályozza. Számba kell venni a lehetséges témaköröket, melyek a probléma megoldásához kapcsolódnak, az ezen témakörök problémáihoz tartozó eszközöket, majd azok előnyei és hátrányai alapján a saját feladatom nyomvonalait ki kell alakítanom. A rendszerrel szemben támasztott felhasználói és funkcionális követelményeket meg kell határoznom. Végezetül az általam választott technológia felhasználásával implementálnom és tesztelnem kell a dolgozatban megfogalmazott rendszert.

# Lehetőségek csapatmunkára

A fejezet néhány olyan lehetőséget vesz számba, melyekkel a csapatmunka egyszerűbbé, hatékonyabbá tehető.

## moodle

A moodle egy nemzetközi, oktatást támogató, nyílt forráskódú szoftver. A legtöbb hazai egyetemen megtalálható. Alkalmas kurzusok felvételére, szerepkörök meghatározására, dokumentumok feltöltésére. Leginkább információ terjesztésére alkalmas, olyan értelemben, hogy az órai anyagot az oktató könnyen elérhetővé teheti minden résztvevő számára, valamint a hallgató is fel tudja tölteni az órai munkáját tetszőleges fájl formátumban. Lehetőséget biztosít még tesztek kitöltésére, mellyel a félévi követelmények ellenőrzése megkönnyíthető.

## Git

A Git a legnagyobb körben elterjedt verziókövető/kezelő rendszer, melyet vállalati és magánszférában egyaránt sokan használnak. Ma már szoftver kódolásához minden tekintetben elengedhetetlen valamilyen verziókövető használata. A Git a legtöbb fejlesztőkörnyezetben integráltan is megtalálható, felhasználóbaráttá téve használatát. A végleges, úgynevezett commitolt változtatásokhoz írt kommentek minden résztvevő számára láthatóak, a változtatások pontokba szedve megtekinthetőek, ezzel egy több résztvevős projekt során a munka átláthatóvá válik, a hibák száma csökken, és a hibákra való reagálás ideje kevesebb.

## Trello

A Trello gyakorlatilag egy Kanban eszköz. Lehetőséget nyújt nagyobb feladatok részfeladatokra bontására, melyeket a Kanban tábla résztvevői magukhoz tudnak venni. A feladatokat lehet kategorizálni állapotuk szerint (fejlesztés alatt, tesztelés alatt, kész, stb.), valamint a Kanban táblán lévő kártyákhoz csatolhatóak Git kommitok is, amivel ellenőrizni lehet a kártya hitelességét.

## TFS (Team Foundation Server)

A Microsoft Team Foundation Server szolgáltatása sokban hasonlít a Gitre. Ez is egy verziókezelő rendszer, a különbség azonban a kettő közt, hogy míg a Git egy elosztott rendszer, tehát a változtatások először egy lokális tárba kerülnek, ahonnan tovább lehet küldeni ezeket a fő termék felé, addig a TFS egy központi architektúrát használ.

## Google Drive

A Google drive az egyik legnagyobb körben használt fájlmegosztó szolgáltatás. Nem olyan specializált, mint a fentebb említett néhány példa, viszont jól ötvözi azok tulajdonságait. Alkalmas több személy által szerkesztett táblázatok, dokumentumok kezelésére, űrlapok létrehozására, bármilyen egyéb fájl megosztására másokkal. Az általam készítendő szoftver a Google szolgáltatásait mintául véve, és azokat kiegészítve valósul meg.

# Követelmény specifikáció

## A mozgatórugók

A szoftver, melyet elkészítek, a fentebb említett előnyök kiaknázására alapul, azokat hivatott egyesíteni. A feladat problémáinak egyediségéből fakadóan tartom a szakdolgozat témáját életképesnek. A hallgatók a szoftver segítségével egy vállalat működését képesek modellezni különböző részletességgel. A rendszert a felhasználókon, és a számukra rendelkezésre álló funkciókon keresztül mutatom be.

## A rendszer használói

A felhasználók az Egyetem hallgatói, valamint oktatói lesznek. A hallgatók csoportja számára biztosítani kell az otthonról való munkavégzés lehetőségét, az egymással való kommunikációt (amennyiben ez nem történik meg önszerveződően, erre biztosítani kell lehetőséget), a határidők és feladatok követhetőségét. Ezek szükségesek ahhoz, hogy a munka gördülékenyen folyjon. A hallgatók a kitűzött feladatok megoldásának megfelelő specializált csapatokba szerveződve végzik a munkát, és minden csoportban van egy vezető, csapatkapitány. A csapatkapitány felelős a határidők betartásáért, valamint hogy a feladatok elvégzése a megfelelő ütemben történjen. Ennek feltétele, hogy a csapatok rálássanak más csapatok munkájára, valamint a feltöltött és elkészült anyagokhoz megjegyzéseket tudjanak fűzni. A vállalati modell szolgáltatásokon keresztül valósul meg, melyek kommunikálnak egymással, és ezeket a szolgáltatásokat lehet részletezve lemodellezni a szoftver segítségével. Egyéb funkció például a csapattársak és a csapat munkájának értékelése, dokumentumok és képek feltöltése.

A rendszer felhasználóinak másik csoportja az oktatók. Az ő lehetőségeik közé tartozik a hallgatók munkáinak ellenőrzése, ebből fakadóan ők is hozzáférnek minden, rendszerben szereplő adathoz, de módosítani csak a saját megjegyzéseiket tudják. Ők tudnak majd csapatokat létrehozni, amelyekhez hallgatók rendelhetőek. A csapatokat teljesen manuális módszerrel is létre lehet hozni, de lehetőségük van előre felvitt sablon alapján ezeket automatikusan generáltatni a rendszerrel. A félév során készült hallgatói értékelésekhez csak a csapat kurzusához tartozó oktatók férnek hozzá, és ezek igény szerint megtekintés után törölhetőek.

Felhasználókat és csoportokat kell tudnia kezelnie a szoftvernek, ebből fakadóan szükség van legalább egy olyan szerepkörre, amellyel azok felvehetőek. Ez legegyszerűbb esetben minden olyan oktató lehet, aki kurzust tart, de alapértelmezetten egy dedikált szerep, melyet egy erre kijelölt felhasználó tölt be. Minden rendszerbeli objektumhoz tartozik valamilyen szerepkör, legyen az egy csoport, egy felhasználó, vagy egy hallgató által létrehozott szolgáltatás. Ezeket a szerepkör-összerendeléseket is kell tudnia kezelnie a rendszernek.

## Hallgatók funkciói

## Szolgáltatások

A fő feladatok közé tartoznak a szolgáltatások, ebből épül fel később a modell. A szolgáltatásoknak vannak igénylői és nyújtói, melyek némely esetben egyeznek, ekkor a szolgáltatást belső szolgáltatásnak nevezzük. A belső szolgáltatások nagyrészt a csapat specialitásának megfelelő fő feladat megvalósításában játszanak szerepet. Minden felvett szolgáltatásnak kötelezően kell tartalmaznia bemenő és kimenő paramétereket. Ezek az adatok a modell mögött lévő adattáblák mezőit reprezentálják. Minden csapat tud készíteni saját adatmodellt, amivel a szolgáltatások funkcionalitását tudja szinkronizálni. A kurzus folyamán minden szolgáltatáshoz szükség lesz egy részletes szöveges leírásra a funkcionalitására vonatkozóan, melyet egy kapcsolódó objektumként, valamint egy folyamatábrára, melyet kép formátumban tudnak a hallgatók létrehozni. A szolgáltatások egy átlátható és szűrhető listában jelennek majd meg a hozzájuk tartozó mérföldkövek állapotaival együtt. Ezek a mérföldkövek például a részletes leírás, igénylő vagy nyújtó általi jóváhagyás, folyamatábra csatolása, oktatói jóváhagyás.

## Adatmodell

Minden csapat készít egy relációs adatmodellt, mely, mint a nevéből adódik, csak egy vázlatos modell, nem tartozik hozzá valós SQL adatbázis. A modellhez tartozó tábláknak kötelezően kell tartalmaznia egy elsődleges kulcsot, és minden mező szerepel legalább egy szolgáltatásban, mint bemenő vagy kimenő paraméter. A csapatok látják egymás adattábláit, de módosítani csak a sajátjukat tudják. A modellhez tartozik külön úgynevezett szótár tábla is, melyet minden csapatnak kötelező elkészítenie, és a lehetséges értékeket összerendelnie különböző táblák státuszmezőivel.

## Folyamatábrák

Minden szolgáltatás részletes leírása mellé tartozik egy folyamatábra, mely eseményvezérelten szemlélteti annak részletes működését. Ezeket az egyetemi signavio szoftverrel készítik a hallgatók, és az abból kimentett képeket lehet feltölteni a rendszerbe. Ezzel egy-egy folyamat helyessége nem ellenőrizhető, az viszont igen, hogy minden szolgáltatáshoz tartozik-e ábra, és ez kényelmesebb és gyorsabb funkcionalitást nyújt, mint dokumentumok kézzel való ellenőrzése.

## Oktatók funkciói

## Hallgatói munka ellenőrzése, jóváhagyása

Minden olyan tevékenység, melyet a hallgatók véglegesítenek a rendszerben, megjelenik az oktatói felületen is. Az oktatók itt tudják ellenőrizni az elkészült munkákat, és azokhoz megjegyzéseket fűzhetnek. Egy szolgáltatás életútjában minden mérföldkövet jóvá tudnak hagyni, így a hallgatók értesülnek a munkájuk helyességéről, valamint később visszatérve láthatják, hogy mely feladatokkal van még elmaradásuk.

## Kurzus nyitása, zárása

Félév elején lehetőség van kurzusok megnyitására. Ezen kurzusokban lehet a csapatokat létrehozni, és azokba hallgatókat felvenni. Igény szerint a rendszer automatikusan létrehozza az előre sablonban meghatározott csoportokat hallgatók nélkül, akár oktatói beavatkozással (egy gombnyomás), akár önmagától, egy előre beállított dátumban. Neptun integráció esetén lehetőség nyílik a kapcsolódó kurzus tárgykódja szerinti csatoláshoz, és így az órarendi időpont alapján a program tudni fogja magától, hogy mikor kell a sablonban megadott csoportokat létrehozni, és akár Neptun azonosítóval is be lehet jelentkezni a rendszerbe. A zárás funkció félév végén történik, ezt szigorúan csak az oktató indíthatja el. A rendszer a féléves adatok alapján különböző riportokat képes létrehozni, amelyből az oktatók átfogó képet kaphatnak akár a hallgatók egyéni, akár a csoportok munkájáról. A program historikusan ment minden adatot. A riportok szempontjai még egyeztetés alatt vannak.

## Rendszerrel szemben támasztott követelmények

Mivel a hallgatóknak otthonról kell dolgozniuk a szoftverrel, ezért a legoptimálisabb megoldás egy iskolai domainen elhelyezett vékonyklienses szoftver. Ennek viszont meg kell felelnie az Egyetem által támasztott biztonsági követelményeknek. Ezzel együtt megoldódna a rendelkezésre állás problémája is.

# A feladat problémáinak kiaknázása a jelenleg rendelkezésre álló eszközökkel

## Verziókezelők

„Verziókezelés alatt több verzióval rendelkező adatok kezelését értjük. Leggyakrabban a mérnöki tudományokban és a [szoftverfejlesztésben](https://hu.wikipedia.org/wiki/Szoftverfejleszt%C3%A9s) használnak verziókezelő rendszereket fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg. Az egyes változtatásokat verziószámokkal vagy verzióbetűkkel követik nyomon.”[[1]](#wiki_CVS)

Egy verziókezelő rendszertől elsősorban az alábbiakat várjuk el:

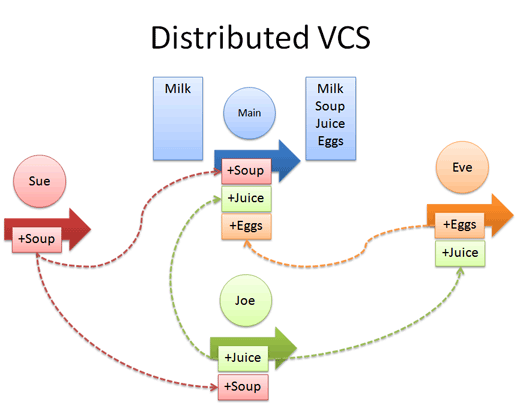
* Több verzió kezelése, követése.
* Több fejlesztő is dolgozhasson egy projekten. Ütközések, együttes módosítások kezelésére képesnek kell lennie.
* Korábbi verziók elérésének és visszaállíthatóságának biztosítása.[[2]](#Central_CVS)

## Verziókezelők típusai

## Elosztott verziókezelők

* Szemben a kliens-szerver (központosított) modellel, az elosztott verziókezelők decentralizált rendszerek. Itt egy központi tároló (angolul repository) helyett minden felhasználó gépe egy-egy külön tárolóként jelenik meg. A szinkronizáció az egyes gépek között küldött [patch](https://hu.wikipedia.org/wiki/Patch)-ek (módosításcsomagok) által valósul meg. Minden munkaállomás legalább egy külön munkamásolattal rendelkezik, nincs központi adatbázis
* A központi adatbázis hiányával a verziókezelési műveletek gyorsak
* Minden munkamásolat biztonsági másolat is egyben, ami természetes védelmet ad az adatvesztés ellen.

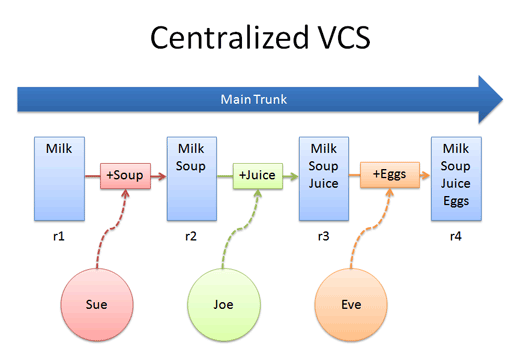
Két fajta elosztott verziókezelő létezik, a nyitott és a zárt. A nyitott rendszereket inkább nyílt forráskódú termékeknél használják, a zártakat inkább a nem nyilvános forráskódú termékeknél.[[3]](#wiki_CVS)



1. ábra Elosztott verziókezelő vázlatos ábrája[[4]](#Central_CVS_Figure)

## Központosított verziókezelők

Minden verziókezelési művelet egy közösen használt szerveren történik. A konkurens használatot az éppen használt fájlra tett zárral oldják meg. A saját változtatását elsőként becsekkelő felhasználó mindenképpen sikerrel fog járni.



2. ábra Központosított verziókezelő vázlatos ábrája[[5]](#Central_CVS_Figure)

## Nyílt elosztott verziókezelők

A nyitott, elosztott verziókezelők támogatják különböző ágak létezését, és erősen függenek az összefésülés (merge) művelettől. Általános jellemzőik a következők:

* Minden munkamásolat gyakorlatilag egy ág.
* Minden ág egy-egy munkamásolatként implementálódik. Az ágak összefésülése patch-ek küldözgetésével történik.
* Lehet válogatni az egyes változtatások között, nem kell feltétlenül minden változtatást letölteni.
* Új tagok bármikor csatlakozhatnak a rendszerhez, nincs szükség szerveroldali regisztrációra.[[6]](#wiki_CVS)

## Zárt elosztott verziókezelők

A zárt, elosztott verziókezelők [adatbázis replikáción](https://hu.wikipedia.org/w/index.php?title=Adatb%C3%A1zis_replik%C3%A1ci%C3%B3&action=edit&redlink=1) alapulnak. Csak egy baseline van, minden becsekkelt változás ebbe kerül bele.[[7]](#wiki_CVS)

## A verziókezelők és a megvalósítandó rendszer

A féléves munka során egy-egy szolgáltatás és minden hozzá tartozó objektum több változata készül el, míg a végleges formáját ölti. A jelenlegi megoldás a google drive-on lévő objektumok folyamatos módosítása, amivel nem érhetőek el az előző verziók, és nem tekinthető meg, hogy ki mit módosított. A napjainkban legelterjedtebb verziókezelő a Git, azonban ennek a használatát – a verziókezelés alapjait általában – sokan nem ismerik. Az általam megvalósított program egy közös helyre gyűjtené a feltöltött vagy ott létrehozott anyagokat, és eltárolja azok feltöltőjét. A dokumentumok különböző verzióival nem kerül felülírásra a régi, hanem azok külön, archív felületen hozzáférhetőek maradnak, egy Admin jogosultságú felhasználó pedig egy gombnyomásra letisztíthatja a rendszerből a régebbi dokumentumokat. Így a lehető legegyszerűbb módszerrel követhetőek maradnak a változások, követhető lesz, hogy ki mit módosított és mikor, valamint minden eddig munka elérhető marad. A signavio (folyamatábra szerkesztő) biztosít revizionálhatóságot, mellyel az ábrák előző verziói megtekinthetőek, de ezeket az ábrákat nem szükséges külön eltárolni, elég a végleges ábrák feltölthetőségét biztosítani.

## Agilis módszertanok

Az agilis módszertanok a szoftverfejlesztés ma használatos módszertanai. Egyik nagy előnyük a rugalmasság, és hogy a hetvenes években megfogalmazott szoftverkrízis problémáira nyújtanak különféle megoldásokat. A legtöbb ilyen módszertan által használt eszköz nem csak szoftverfejlesztésben alkalmazható, hanem bármilyen más területen, ahol szükség van a folyamatos változások követésére és bárminemű implementálására. A legnagyobb előnye, hogy nincs kőbe vésve egy-egy projekt kezdetekor a végtermék specifikációja, hanem – mivel általában az ügyfél nem tudja, hogy pontosan milyen rendszert akar – az ő általa kért módosítások és változtatások a projekt életútja során bármikor megvalósíthatóak anélkül, hogy az a határidő vagy a végtermék rovására menne.

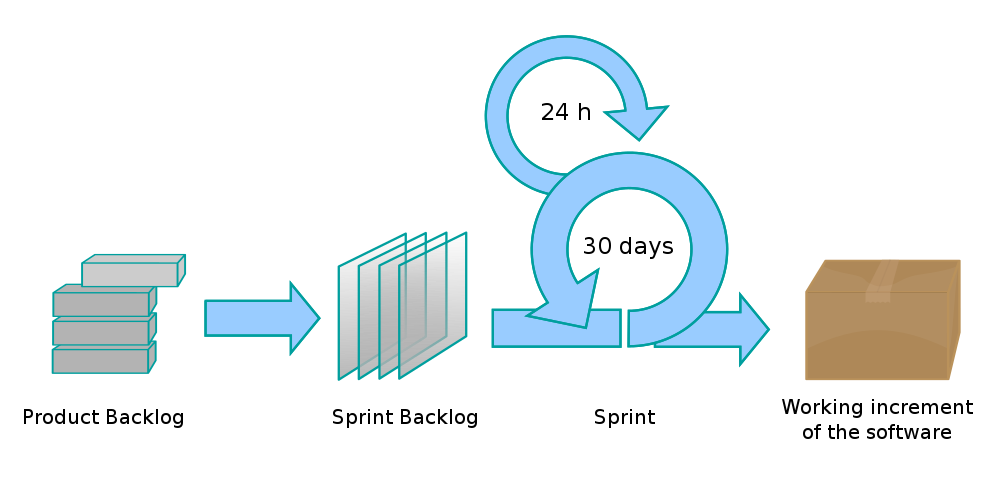
## SCRUM

A scrum egy ajánlás, egy keretrendszer. Egyik fő eleme a szerepkörök, és hogy minden szerepkörnek megvan a jól behatárolt feladata. A projekt mérföldkövekből épül fel, melyek előre megtervezettek, de a belső lefolyásuk rugalmas, és igény szerint alakítható.

A scrum szerepkörök két csoportra oszthatóak. A „csirkék”, akik nem vesznek részt a folyamatban, és bár manapság inkább a „stakeholder”, haszonélvező kifejezést használják rájuk, a scrum köztudatban az előbbi maradt meg. A másik csoport a „disznók”, akik a saját bőrüket viszik a vásárra, ők azok, akik a folyamat megvalósításában effektíven részt vesznek. Közéjük tartoznak:

* Terméktulajdonos (Product Owner): Az összekötő az ügyfél és a megvalósító csapat között. Feladata a csapat számára biztosítani a megfelelő üzleti információkat, és a teendőlistát frissíteni.
* Scrum Master: Feladata a fejlesztőcsapat útjába eső akadályok elhárítása, és hogy a csapat mindig a feladattal tudjon foglalkozni.
* Fejlesztők: A fejlesztő csapat önszerveződő és önellenőrző. Általában 3-9 főből áll, és lehetőség szerint a csapat tagjai különféle területre specializálódtak.[[8]](#WIKI_SCRUM)

A SCRUM folyamata sprintekből áll, melyek során egy működő prototípus készül. Ezt a prototípust az ügyféllel tudjuk véleményeztetni, és ez alapján tudunk változtatásokat eszközölni. A sprintekhez úgynevezett „backlog”, leírás tartozik, mely leírja, hogy a sprint végén lévő terméknek mit kell tudnia. A sprint backlog a „product backlog”, termék leírás alapján készül, mely az egész termék specifikációja.



3. ábra SCRUM folyamat vázlatos ábrája[[9]](#WIKI_SCRUM)

A SCRUM napjainkban a nagyvállalati szoftverfejlesztésben a legnépszerűbb. Kis cégeknél sem a projektek, sem pedig a rendelkezésre álló humán erősforrás mérete nem teszi lehetővé vagy kifizetődővé az alkalmazását, míg egy nagy cégnél, ahol akár száz fejlesztő is dolgozik sok projekten, ez a módszertan rendkívül közkedvelt.

## Kanban

A Kanbant a Toyota fejlesztette ki, és ebből fakadóan a Japán értékek és szemléletmód tükröződik rajta. Az ország, szemben gazdasági versenytársaival nem rendelkezik sok nyersanyaggal, ezért az iparának szükséges alapanyagokat importálniuk kell. A Kanban alapvetően arra keres megoldást, hogy hogyan lehet a rendelkezésre álló (akár import akár saját) nyersanyagainkból a lehető legtöbbet kihozni miközben a lehető legkevesebb megy veszendőbe. Ennek megvalósítási módjai közé tartozik az úgynevezett „Just in time production”, a munkavállalók képességeinek legjobb kihasználása, vagy a munkahelyi balesetek csökkentési a várakozási idő eliminálásával. A Kanban alapvetően nem egy ERP rendszer vagy annak modulja, ami a számítási feladatok meggyorsításával segíti a vállalatot.[[10]](#Kanban_Source)

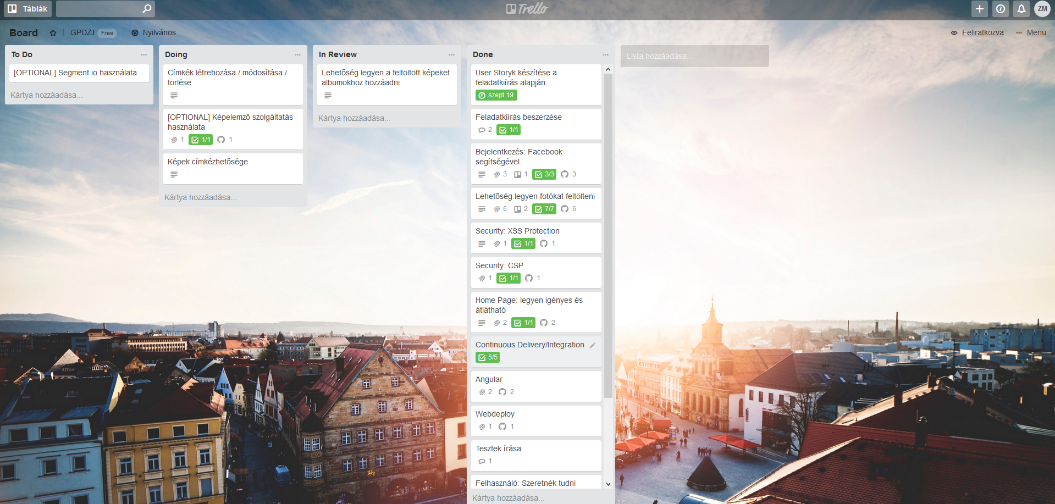
Kanbannal a szoftverfejlesztésben is lehet találkozni. A legtöbb úgynevezett „ticketing”, azaz jegyrendszer a Kanban elveket követi. Ezekben a rendszerekben a fejlesztőknek lehet készíteni jegyeket (pl. SCRUM sprint backlog alapján feladatokat), a jegyekhez lehet leírást készíteni, időtartamot fűzni, nehézséget vagy prioritást állítani, és a rendszer funkcionalitásától függően számos más tulajdonságot hozzáfűzni. Az elkészült jegyeket aztán a fejlesztők magukra tudják venni, munka közben a kijelölt tulajdonságokat tudják állítani. Az egyik ilyen eszköz a Trello.

## Agilis módszertanok eszközei

Ahogy a nevéből adódik, a SCRUM, a Kanban, és minden egyéb módszer csupán egy módszer, egy ajánlás. Ahhoz, hogy az általuk megfogalmazott alapelveket használjuk, valamilyen eszközre lesz szükségünk. Ezek az eszközök, ahogy a verziókövetők, nem csak a szoftverfejlesztésben, hanem bármilyen más területen használhatóak. Előnyük, hogy a munka szervezetté, összefogottabbá válik. A munka menete és a projekt előrehaladása követhető, a változások egyeztethetőek lesznek. Nagyban nő a tervezhetőség, valamint csökken a komplexitásból adódó plusz munka ráfordítás. Röviden kevesebb, de céltudatosabb munkával minőségibb termék állítható elő.

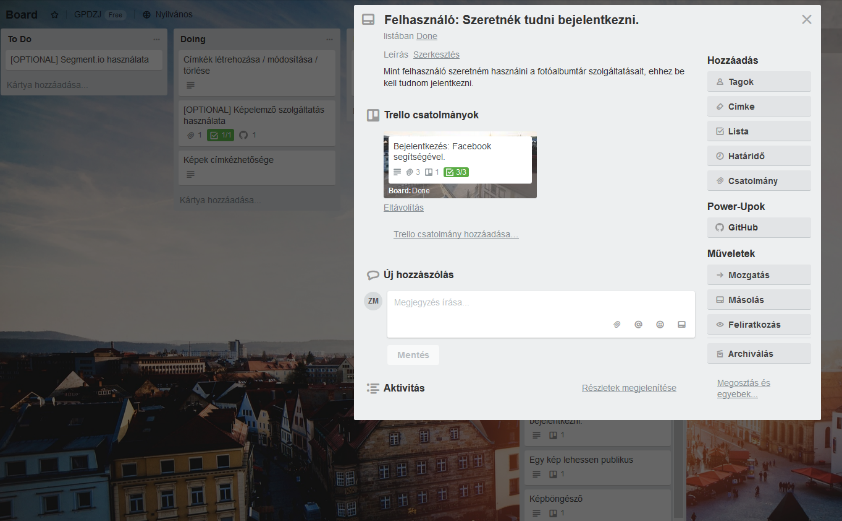
## Trello

A Trello a Kanban táblák létrehozásához és kezeléséhez tipikusan használható eszköz. Népszerűsége az egyszerűségében és ingyenességében rejlik. A felület tetszés szerint testre szabható anélkül, hogy a funkcionalitást csökkentené, vagy a munkavégzést gátolná. A működése és a benne szereplő objektumok analóg változatai is alkalmazhatóak. Létrehozhatóak benne csoportok, melyekhez tartozik egy vagy több tábla. Ezek a táblák általában a projekten dolgozó csapatok munkáit foglalják össze. Minden táblában szabadon létrehozhatóak oszlopok, melyek kategóriákat jelölnek, az oszlopokhoz kártyák vehetők fel, és a kártyák szabadon mozgathatóak az oszlopok között. A legegyszerűbb példa a „To Do”, „In Progress”, és a „Done” oszlopok, azaz azon kártyák kategóriái amelyeket a jövőben meg kell oldani, épp munkafázisban vannak és dolgoznak rajtuk, vagy már készen vannak.



4. ábra Trello tábla

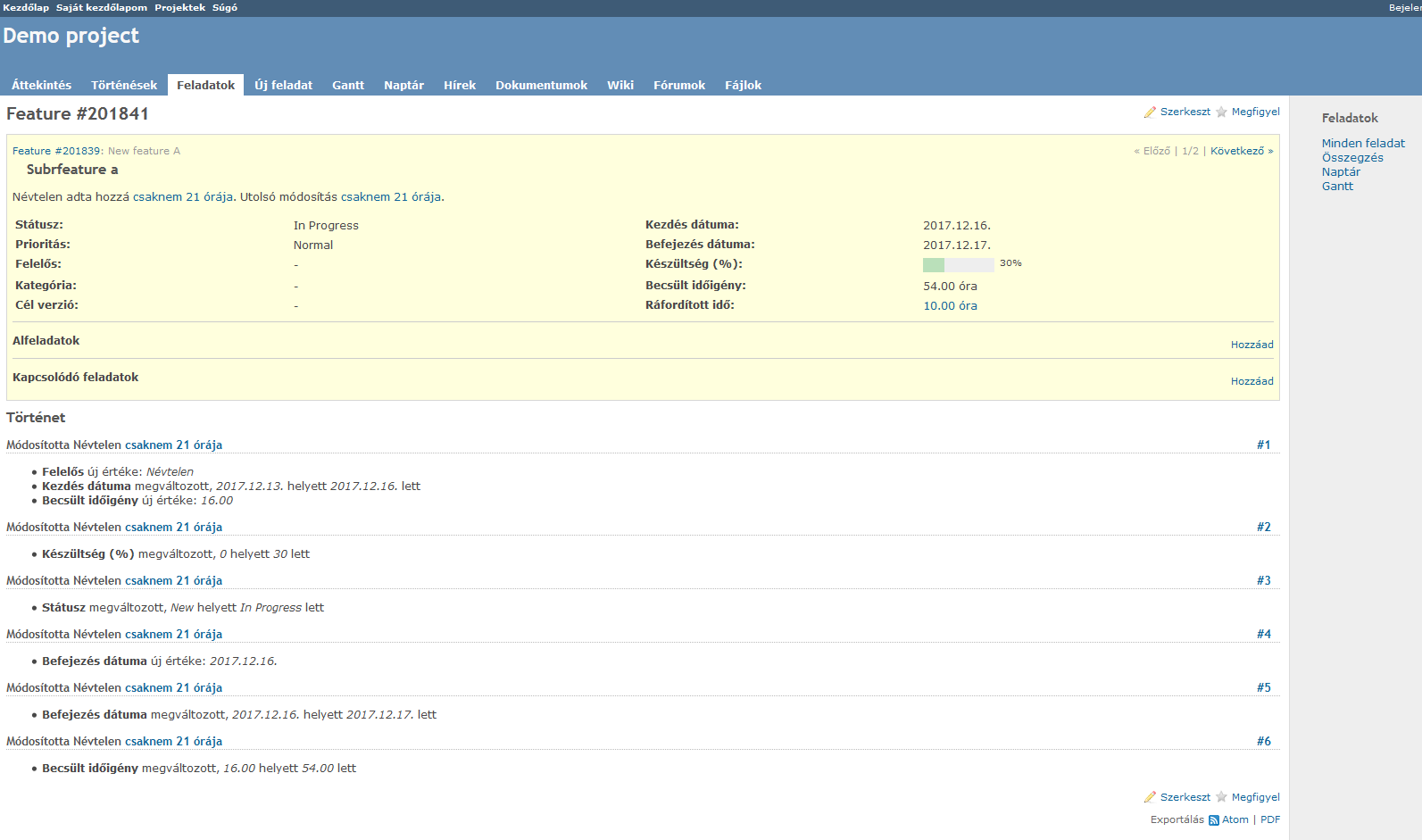
A kártyákon az ajánlás szerint érdemes a lehető legkevesebb szöveggel a lehető legtöbb információt leírni. Amennyiben külső fájl, másik kártya, vagy esetleg Git kommit tartozik a kártyához, az is egyszerűen hivatkozható. Ezen felül lehet listákat létrehozni a kártyákon, tipikusan „Definition of Done” (DoD), a kártyán szereplő feladat készségének definíciója szokott szerepelni. Csatolható határidő is az egyes elemekhez, ami egy SCRUM Sprint definíciójához elengedhetetlen. A rendszerben van lehetőség jogosultságok kiadására, megszabható hogy ki melyik táblához férjen hozzá, és a táblában lévő kártyákat ki tekintheti meg, vagy szerkesztheti.



5. ábra Trello kártya szerkesztése

## Redmine

A Redmine egy projekt menedzsmenthez használható rugalmas, ingyenes alkalmazás. Tipikusan nem alkalmazzák egyik módszertanhoz sem olyan értelemben, mint a Trellot a Kanbanhoz, viszont nagyobb munkák során rendkívül nagy segítséget nyújt, akár valamilyen módszertan által kijelölt munkamenetet próbálunk alkalmazni, akár nem. Lehetőség van több projektet vezetni benne, és minden projektet apró részekre, sprintekre bontani. A sprintekhez aztán jegyeket tudunk felvenni, amiknek beállítható a határideje, a becsült szükséges ráfordítása. Minden jegyhez tartozik egy, a rendszer által generált sorszám, ami könnyen kereshető. A fejlesztők ezeket a létrehozott jegyeket magukra veszik, és a megvalósítás folyamata közben a szükséges mezőket töltik. A jegyeknek állítható az állapota, így például a tesztelés során azonnal értesül a fejlesztő, ha a munkájában hibát találtak. Szintén lehetőséget nyújt a rugalmas jogosultság kezelésre, valamint verziókövetési integrációt is tartalmaz. [[11]](#Redmine)



6. ábra Redmine felület

## Az agilis módszertanok és a megvalósítandó rendszer

A fentebb leírt eszközök egy szoftverfejlesztés során előszeretettel kerülnek elő. Mivel a munka során a fejlesztők és a menedzserek akár egy, akár több projekten dolgoznak huzamosabb ideig, és közben ezeket az eszközöket folyamatosan használják, nem esnek ki a ritmusból. Az ergonómia is sokat számít a munkavégzés során, hogy ne kelljen minden után keresgélni, a lehető legkevesebb kattintással a lehető legtöbb információ kerüljön a felhasználó birtokába. Mind a projektlabor szakirányos tárgy, mind pedig a VirMod során a google drive került használatra, ahol az egyes csapattagok feladatai csak akkor voltak megtekinthetőek, ha elnavigáltunk egy külön mappában lévő PDF-hez, ahol le volt írva a megvalósítandó szolgáltatás neve. Ezután a szolgáltatás leírása ismét egy másik dokumentumban volt egy másik mappában. Majd ha ezekhez az információkhoz hozzájutott a hallgató, elkezdhette a folyamatábra készítését. Amennyiben valami változtatást kellett eszközölni a szolgáltatáson, azt az előbb említett dokumentumban kellett módosítani, erről viszont senki nem értesült a módosítón kívül. Így ha egy másik csapat az aktuális logikára épített egy saját szolgáltatást, de az előbbi változtatása ezt a logikát már nem szolgálná ki, akkor az utóbbi szolgáltatás hibás lesz, de erről értesítést nem kap. Ezeket az eszközöket természetesen lehetne használni önmagukban a félév során, viszont ezzel csak még több kezelendő felület kerül elő, valamint ezen felületek kezeléséhez nem biztos, hogy mindenki ért, tehát plusz ráfordítás lenne a hallgató és az oktató szempontjából is ezek megismerése. Az általam tervezett alkalmazás a fenti problémákra egy helyen próbál megoldást nyújtani, a lehető legkevesebb felhasználói beavatkozással. A lehető legkevesebb felhasználói beavatkozás azt jelenti, hogy érintett módosítás esetén az illetékes e-mailben kap visszajelzést, a határidőkről és a feladat pontos leírásáról is egy helyen tájékozódhat a felhasználó.

## Egyéb, ingyenesen elérhető közösségi eszközök

A kurzus során is használt egyik ingyenesen elérhető eszköz a Google drive, ahol egy megosztott mappában tudunk létrehozni bármikor elérhető dokumentumokat. A létrehozott fájlokhoz vagy az azokat tartalmazó mappákhoz jogosultságokat tudunk létrehozni, megtekinthetőek a régebbi változások, az azokon végzett módosítások, és hogy ki módosította a dokumentumot. Ezeket valós időben több felhasználó is szerkesztheti, azonban ez sokszor ahhoz vezet, hogy – mivel többen is ugyan azon dolgoznak – egymás munkáját írják felül. Sok adatnál az Excel táblázatok egy idő után kevéssé átláthatóak lesznek, és a szűréssel nem csak a mi felületünkön szűrünk, hanem mindenkién. A fentiekből következik, hogy a Drive megosztásra és hozzáférhetőségre rendkívül alkalmas, hisz a felhőben tárolt fájlok bárhonnan hozzáférhetőek, nem kell azokat magunkkal hordanunk, viszont az intenzív csapatmunkában és a nagy adathalmazok kezelésében nem olyan jó.

Egy másik ilyen közösségi eszköz, ami szintén ingyenesen elérhető, a moodle. Ez a platform tartalmaz rengeteg funkciót, amivel segítheti a munkát. Az oktatók tudják követni a hallgatók előrehaladását egy külön felületen. A többnyelvűség lehetővé teszi, hogy egy globális közösség támogassa, fejlessze, és használja. Tevékenységek vehetőek fel, köztük tesztek, visszajelzések, felmérések, chat, és számos egyéb. Az oktató és hallgató közötti információáramlást és fájlcserét is nagyban elősegíti, azonban kevéssé támogatja a nagyobb volumenű csapatmunkát. Leginkább az előadás diáinak, anyagának elérésére, házi feladatok fel- és letöltésére, valamint tesztek kiírására és kitöltésére használatos.

# Összefoglalás

A célkitűzések és a számba vett lehetőségek együttese által a rendszer funkcionális követelményei körvonalazódnak. A rendszerterv és a tárgyalt alternatívák körülhatárolják azokat a funkciókat és paramétereket, melyek segítségével a szoftver gördülékeny működése megvalósítható. Az optimális működéshez ezeken felül még szükség van a felhasználói követelményeknek való megfelelésre, valamint az implementáció előtti és közbeni tervezésre. A felhasználói követelmények alatt értem a rendelkezésre állást, a biztonságot, a felhasználóbarát működést. Ezeknek a tervezése akkor válik időszerűvé, mikor a projekt az implementáció tervezésének fázisába ér.

Mivel a fent tárgyalt eszközök és lehetőségek rendkívül sokszínű funkcionalitással rendelkeznek, melyből a megvalósítandó projekthez nincs szükség mindenre, ezért kizárólag a működéshez elengedhetetlen elemeket tudom felhasználni. A célom továbbra is az, hogy ez az alkalmazás könnyen kezelhető, magától értetődő, a tárgyi követelményeket maximálisan kiszolgáló, és a mindennapi munka során hatékony legyen. Az üzleti életben használt megoldások közül lehetőség szerint a Google drive dokumentum tárolási szolgáltatását szeretném integrálni, tekintettel a kurzusok során keletkező dokumentumok mennyiségére.

A dolgozatom további feladatai közé tartozik a rendszer funkcionális megtervezése, majd a tervek megvalósítása, sorrendben:

* adatbázis szerkezet
* funkciók részletes meghatározása valamint megvalósításuk
* felhasználó-kezelés
* jogosultsági mátrix
* dokumentum-kezelés (megtekintés, letöltés, feltöltés)
* a fentiek tesztelése

# A rendszer elvi felépítése

A rendszert használó felhasználók és azok feladatai alapján két jól elkülöníthető részre lehet osztani a funkciókat. Míg az Oktatók az adminisztrációs és egyéb óravezetési feladatokat látják el, addig a hallgatók a felületen felvett szolgáltatások segítségével építenek fel egy, a tantárgy céljának megfelelő vállalati modellt. A rendszer logikai törlést használ, és amennyiben egy-egy érintett elemre külön szabály nem vonatkozik, csak olyan objektum törölhető a rendszerből, amelynek nincsen kapcsolódása. Később ezek a törölt adatok igényektől függően akár archivált állapotúak is lehetnek, és lehetőség van rá, hogy a rendszerbe visszakerüljenek.

## Oktatói funkciók

## Félév kezdés, adminisztráció

Az oktatóknak lehetőségük van új féléveket létrehozni. Minden munka, ami a rendszerben rögzítésre kerül, egy aktív félévhez tartozik, melyből csak egy lehet a rendszerben. Szemesztertől függetlenül tudnak hozzáadni tanárokat a rendszerhez, akiket később csoportokhoz rendelhetnek. Ezek a csoportok már félévek alá tartoznak, és ezek fogják össze az egy-egy oktató által oktatott hallgatókat. A csoportokat csapatok alkotják, melyek reprezentálják a megvalósítandó vállalat funkcióit, és a feladatkörük is szorosan ahhoz kapcsolódik. Hallgatót csak csapathoz rendelve lehet létrehozni a rendszerben, ezzel könnyítve az adminisztrációt és a követhetőséget. A program rendelkezik import funkcióval, mely segítségével egy egész kurzus rögzíthető. A félévek végén lehetőség van félévet léptetni. Ez a funkció az összes, jelenleg aktív félév adatait átemeli egy, a felhasználó által a felületen megadott új félévhez. Az oktatók férnek hozzá kizárólag a rendszerben szereplő adatszótárakhoz. Ezektől függ a legtöbb automatizált funkció működése. Az adatszótári elemek tulajdonságai közé tartozik a védettség. Egy védett elem mentés után nem törölhető a felületről, csak módosítható. Ezek a funkciók merítik ki az oktatás szervezés feladatát a rendszerben

## Féléves munka, oktatás

A félév során épülő modell a szolgáltatásokból épül fel. Ezen szolgáltatásokat a hallgatók készítik, viszont az oktatóknak is lehetőségük van a munkák megtekintésére, ellenőrzésére. Az oktatók látják más, nem az ő csoportjaikban lévő hallgatók munkáját is, viszont azokhoz csak és kizárólag megtekintési jogaik vannak. Az elkészült szolgáltatásokhoz vagy szolgáltatás igényekhez tudnak megjegyzéseket fűzni, a hibákra és hiányos részekre felhívni a figyelmet. A házifeladatok és határidők nyilvántartásának vezetése, és ezzel a félév ütemezése is a program oktatói funkciói közé tartozik, valamint lehetőség van minden félév végén értékeléseket megadni az egyes hallgatókhoz. Ezek a funkciók merítik ki az szakmai oktatás támogatását a rendszerben.

## Hallgatói funkciók

A hallgatók a félév elején csapatokba szerveződve kezdik meg a munkát. Minden csapathoz tartozik egy csapatkapitány, aki egy külön felületen tud a többi kapitánnyal egyeztetni. A féléves tevékenységek nagy részét a szolgáltatások modellezése teszi ki. Minden szolgáltatást a sorszáma azonosít egy csoporton belül. A csapatok és az általuk készített szolgáltatások egymásba fonódva alkotják majd a felépítendő rendszert, ezért mindegyikhez tartozik egy nyújtó és egy igénybe vevő csapat, ami lehet ugyan az. Amennyiben egy csapatnak olyan szolgáltatásra lenne szüksége, amely nem szerepel a rendszerben, ezt egy kéréssel, „request”-tel jelezheti a funkciót ellátó csapat felé. Egy-egy szolgáltatás leképezhető a programozásban ismert metódussal vagy függvénnyel, ezért lehetőség van be- és kimenő paraméterekkel ellátni azokat, valamint más szolgáltatásokat is igénybe vehetnek. Ezeket a paramétereket nagyban a modellben található, a hallgatók által létrehozott adattáblák mezői teszik ki, de lehetőség van egyedi paraméterek megadására. A félév végén rendelkezésre áll egy formázatlan dokumentáció automatikus legenerálása, valamint ennek formázott változatát elkészítve, később a google drive-ba feltöltve lehetőség nyílik ennek megtekintésére. A felület az oktatók által bekapcsolt ellenőrzéseket, a többi csapat igényeit, valamint egyéb hiányosságokat a hallgatók felé jelzi, azok számosságáról és mibenlétéről minden illetékes felhasználó könnyen értesülhet.

## Felhasználókezelés

A rendszer MSSQL adatbázist használ, és abban valósul meg a felhasználók nyilvántartása is. Oktatónként és hallgatónként az adatbázisban szerepel egy-egy felhasználó rekord. A felhasználó a megadott emailcímen értesül, hogy bekerült a rendszerbe. Az alapértelmezett jelszó és felhasználónév minden esetben a megadott e-mail cím, melyet az első bejelentkezéskor kötelező megváltoztatni. Ezek után minden felhasználónak lehetősége van új jelszót igényelni. Az adatbázisban a felhasználók jelszavai HASH-elve kerülnek tárolásra, így biztonságosabb azok tárolása.

# Adatbázis-szerkezet

A rendszert egy MSSQL szerver szolgálja ki. A fejlesztés során lokális adatbázist használtam, a kiadott verziók teszteléséhez pedig egy bárhonnan elérhető ingyenes Azure SQL szerver állt a rendelkezésemre. Ezek szolgálnak a program használata során keletkező adatok tárolására, itt történik a felhasználók autentikációja. A programomat SQL First struktúrában fejlesztettem. A program SQL2014 vagy afölötti verzióval kompatibilis. Az adattáblák és mezőik elnevezésében angol nyelvet használtam. Az összes, fejlesztés során készül sql script megtalálható a dolgozatomhoz csatolva, valamint egy SQLCMD módban futtatható scripttel (RUN\_ALL.sql) az összes szükséges adatbázis művelet végrehajtásra kerül ami a program működéséhez szükséges. Minden tábla azonosítója az ID mező, amely integer típusú, és az MSSQL Identity függvénye biztosítja az egyediségét, valamint a logikai törlést biztosítandó, egy Deleted nevű bit flag is szerepel minden táblán.

## Adattáblák

## Users

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| NEPTUN | nvarchar | Felhasználó NEPTUN kódja |
| FirstName | nvarchar | Felhasználó keresztneve |
| LastName | nvarchar | Felhasználó vezetékneve |
| FullName | nvarchar | A felhasználó teljes neve |
| Email | nvarchar | A felhasználó E-mail címe |
| UserName | nvarchar | Felhasználó bejelentkezési neve |
| UserPassword | nvarchar | Hash-elt jelszó |
| PasswordChangeRequired | bit | Jelszóváltoztatás szükségességét jelző flag |
| Deleted | bit | Törlést jelző flag |

A *Users* tábla tárol minden felhasználói adatot. Minden oktatóhoz (*Teacher*, ld. később) és hallgatóhoz (*Student*, ld. később) pontosan egy rekord szerepel ebben a táblában. Amennyiben akár az oktató, akár a hallgató törlésre kerül, a *Users* rekord is törlésre kerül. A táblában a *NEPTUN*, *UserName* és *Email* mezők egyediek

A fontosabb mezők:

* *Email* és *UserName*: Alapértelmezetten minden felhasználó belépési neve a megadott E-mail címe, amelyet meg tud változtatni. A rendszerből érkező minden levél a megadott címre érkezik.
* *PasswordChangeRequired*: a felhasználó létrehozása után, valamint új jelszó igénylésénél ez indikálja, hogy a felhasználónak új jelszót kell megadnia a legközelebbi bejelentkezésnél.

## Student és Teacher

A két tábla szerkezete sorrendben:

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| UserID | int | Tanuló felhasználó azonosító |
| TeamID | int | Tanuló csapat azonosító |
| Deleted | bit | Törlést jelző flag |

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| UserID | int | Oktató felhasználó azonosító |
| Deleted | int | Törlést jelző flag |

A két táblában hallgatók és oktatók rekordjai találhatóak. A két entitás szétválasztására azért került sor, hogy a későbbi esetleges módosítások ne okozzanak problémát, valamint hogy a két féle objektum jól elkülöníthető legyen. Minden rekordhoz pontosa egy *Users* rekord tartozik. A diákok csak csapatba sorolva vehetőek fel, ezért a *TeamID* mező kitöltése kötelező.

## SessionGroup

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| SessionGroupName | nvarchar | Csoport neve |
| SemesterID | int | Félév azonosítója |
| TeacherID | int | Csoportot vezető oktató azonosítója |
| Deleted | bit | Törlést jelző flag |

A tábla a féléven belüli csoportokat tartalmazza, melyekben csapatokba szerveződve dolgoznak a diákok. A csoportokat minden esetben egy oktató tartja. A felületen csak az aktív félévhez tartozó csoportok láthatóak. Egy csoport csak akkor törölhető, ha nincs benne csapat.

## Team

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| SessionGroupID | int | Csoport azonosító |
| TeamName | nvarchar | Csapat neve |
| TeamCaptain | int | Csapatkapitány |
| Deleted | bit | Törlést jelző flag |

Ebben a táblában tárolódnak a csapatok. Minden csapat egy csoporthoz tartozik, valamint minden csapatnak lehet egy csapatkapitánya. A csapatkapitányok bonyolítják le a csapatok közti kommunikációt. A csapat neve egyedi kell, hogy legyen csoportonként, de egy félévben több azonos nevű csapat is lehet. Egy rekord akkor törölhető, ha nem tartozik hozzá hallgató.

## Semester

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| SemesterName | nvarchar | Félév neve |
| SemesterType | int | Szemeszter típusa (Őszi/Tavaszi) |
| IsActive | bit | Aktív állapotot jelző flag |
| Deleted | bit | Törlést jelző flag |

A félévek ebben a táblában kerülnek tárolásra. Minden félév egyedi névvel kell, hogy rendelkezzen, és a típusa az adatszótárból választható ki. Egy félév csak akkor törölhető vagy archiválható, ha nem tartoznak hozzá csoportok.

## Dictionary

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| ItemName | nvarchar | Elem neve |
| ShortItemName | nvarchar | Elem rövid neve |
| DictionaryTypeID | int | Szótárelem típusa |
| IsProtected | bit | Védettséges jelző flag |
| Deleted | bit | Törlést jelző flag |

A *Dictionary* tábla tárolja a rendszerben található összes szótárelemet. Minden szótárelem egy szótár típushoz tartozik, és azon belül egyedi. A rövid név csak a csapat típusok neveinél használható érdemben, a szolgáltatásoknál az jelenik meg sok helyen. A védettséget jelző mező feladata, hogy amikor egy elemet a programban védettnek jelölünk, az később nem törölhető, csak módosítható, ezzel megakadályozandó a véletlen törléseket.

## DictionaryType

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| TypeName | nvarchar | Típus neve |
| IsSessionGroupType | bit | Csoport típust jelző flag |
| Deleted | bit | Törlést jelző flag |

Ebben a táblában találhatóak a rendszer által használt szótárelemek típusai. Minden típus neve egyedi, és csak akkor törölhető, ha nem tartozik hozzá szótárelem. A rendszerben lehetőség van sablon alapján legenerálni a csapatokat új csoport létrehozásánál. Hogy melyik sablon alapján készülnek a csapatok, az a szótárban lévő elemektől függ. Minden szótártípus beállítható, mint csoport típus elem (*IsSessionGroupType*), és csapatok generálásakor a benne található szótárelemek alapján jönnek létre a csapatok az újonnan létrehozott csoportban. Ezek a szótárelemek nem kötődnek szorosan a csapatokhoz, csak a nevüket öröklik, azonosító alapján nincs hivatkozás, így akár előremenőleg megváltoztathatók a csoportok sablonjai, anélkül hogy ez a már felvett adatokon tükröződne, vagy törölhetők, amennyiben nincs védettek.

## ServiceStore

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| ServiceName | nvarchar | Szolgáltatás neve |
| CreatorID | int | Szolgáltatás készítője |
| ProviderTeamID | int | Nyújtó csapat azonosítója |
| RequesterTeamID | int | Igénybe vevő csapat azonosítója |
| ServiceDescription | nvarchar | Szolgáltatás rövid leírása |
| Deleted | bit | Törlést jelző flag |

A szolgáltatásokat tároló tábla a *ServiceStore,* „Szolgáltatástár” tábla. A félév során nagy számban készülő szolgáltatásokat tárolja. A félév során modellezni kívánt vállalat több funkcióból (ellátott feladat szerint specifikus) épül fel, melyeknek egymással és a külvilággal szolgáltatásokon keresztül kell kommunikálniuk, így a *ProviderTeamID* reprezentálja a szolgáltatást nyújtó csapatot, a *RequesterTeamID* pedig a szolgáltatást igénybe vevő csapatot. A két azonosító lehet azonos, ebben az esetben beszélünk belső szolgáltatásról. A rövid leírásban szereplő szöveg segíti a csapatokat és az oktatókat a szolgáltatás funkciójának megértésében.

## ServiceRequest

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| ServiceName | nvarchar | Szolgáltatás neve |
| RequesterTeamID | int | Igénylő csapat azonosítója |
| ProviderTeamID | int | Nyújtó csapat azonosítója |
| ServiceID | int | Szolgáltatás azonosítója |
| Deleted | bit | Törlést jelző flag |

Amennyiben egy csapatnak olyan szolgáltatásra van szüksége egy másik csapattól (vagy akár önmagától), amelyik még nem szerepel a rendszerben, egy igényt tud feladni a szolgáltató csapat felé. Ez megjelenik a másik csapat felületén, mint kidolgozandó feladat. Az elkészült szolgáltatást az igényhez lehet csatolni (*ServiceID)*, majd jóváhagyás után az bekerül a rendszerbe, míg a kérés törlődik. Már meglévő szolgáltatás is bármikor csatolható egy-egy kéréshez. A nyújtó és igénylő csapat logikája megegyezik a szolgáltatástár logikájával.

## ServiceTable

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| TableName | nvarchar | Tábla neve |
| TeamID | int | Tulajdonos csapat azonosítója |
| Deleted | bit | Törlést jelzó flag |

Mint a legtöbb rendszer, a hallgatók által modellezett is rendelkezik egy adatbázissal, azonban ez a fizikai megvalósítás helyett szintén egy modell. Ebben a táblában kerül tárolásra minden olyan virtuális adattábla, amelyet a csapatok igénybe tudnak venni. A táblák nevei egyediek a csoportokon belül, így érdemes ezeket prefixekkel ellátni, amennyiben több egy csoportban több csapatnak is ugyan olyan nevű táblákra van szüksége. Szerkeszteni mindenki az ő csapatához tartozó elemeket tudja csak, azonban a szolgáltatásai használhatják a csoporton belül bárki tábláját.

## ServiceTableField

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| TableID | int | Tábla azonosító |
| FieldName | nvarchar | Mező neve |
| FieldTypeID | int | Mező típus azonosító |
| Deleted | bit | Törlést jelző flag |

A fenti *ServiceTable* rekordok egy táblát reprezentálnak, ezért kell, hogy tartozzanak hozzájuk mezők. Ezek a mezők a *ServiceTableField* táblában kerülnek tárolásra. Itt megadható a nevük, valamint hogy milyen a típusuk. Ezek a típusok az adatszótárban találhatóak, egy védett szótárgyűjtemény, az „Adattípusok” alatt. Tetszés szerint vehetőek fel új típusok, de a program implementálása során bekerülnek a legfontosabbak: *int, bool, string, float, double, char, date, datetime, time.*

## ServiceStoreParams

|  |  |  |
| --- | --- | --- |
| Oszlop neve | Oszlop típusa | Rövid leírás |
| ID | int | Azonosító |
| ServiceID | int | Szolgáltatás azonosító |
| InOut | bit | Be/kimeneti viselkedést jelző flag |
| ServiceTableFieldID | int | Mező azonosító |
| Deleted | bit | Törlést jelző flag |

A szolgáltatások be-, és kimenő paraméterei ebben a táblában találhatóak. A felületen az elkészült táblák (*ServiceTable*), és azok mezői (*ServiceTableField*) választhatóak ki. Egy irányba (be-, vagy kimenet) egy mező csak egyszer szerepelhet. Lehetőség van egyedi paraméter megadására is, ezek olyan *ServiceTableField* rekordok, melyek nem tartoznak egy táblához sem, így ezek szolgáltatásonként egyediek.

ÁBRA MAJD HA KÉSZ A DB

# A program felépítése

A program az SMaP fantázianevet kapta a „Service Management Program” rövidítéséből. Tekintve, hogy az éles futáshoz használható adatbázis biztosítása kérdéses, hogy nem rendelkezem saját webszerverrel, valamint, hogy a vastagklienses technológiákban és az SQL-ben vagyok jártas a programozás területén, ezért a választásom a Microsoft WPF technológiájára esett. Az adatbázis egy SQL 2016 szerver, amelyet a fejlesztés során egy lokális szerver szolgált ki, a kiadott verziók teszteléséhez és bemutatásához pedig egy Microsoft felhőben található ingyenes Azure adatbázist hoztam létre. Az Azure-ban lévő szolgáltatás hátránya, hogy ingyenességéből fakadóan rengeteg funkció le van tiltva benne (SQL Job-ok, E-mail küldés, Backupok, stb.), ezért ez a továbbfejleszthetőség és hosszú távú használat lehetőségeit leszűkíti. Munkámból fakadóan adatbázis szemlélettel szeretek dolgozni, ezért az adatbázis és a program között egy Database First Entity Framework modellt használok.

## WPF

## MVVM

## Data Access Layer

## Entity Framework

## Database First

# Tesztelés

# Továbbfejlesztési lehetőségek

# Irodalomjegyzék

[1,3,6,7] Wikipédia: Verziókezelők  
https://hu.wikipedia.org/wiki/Verzi%C3%B3kezel%C3%A9s   
utoljára megtekinve: 2017. 12. 10

[2] Miért kell a verziókezelés?  
http://bmeaut.github.io/snippets/snippets/0114\_VerziokezelokOsszehasonlitasa/  
utoljára megtekinve: 2017. 12. 12

[4,5] Intro to Distributed Version Control (Illustrated)  
https://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/  
utoljára megtekinve: 2017. 12. 13

[8,9] https://hu.wikipedia.org/wiki/Scrum  
 utoljára megtekinve: 2017. 12. 14

[10] Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system, Y. SUGIMORI , K. KUSUNOKI , F. CHO & S. UCHIKAWA  
<http://www.tandfonline.com/doi/abs/10.1080/00207547708943149>  
utoljára megtekintve: 2017. 12 . 14

[11] Redmine  
<https://www.redmine.org/>  
utoljára megtekintve: 2017. 12. 14

# Ábrajegyzék

[1. ábra Elosztott verziókezelő vázlatos ábrája[4] 12](#_Toc529906964)

[2. ábra Központosított verziókezelő vázlatos ábrája[5] 12](#_Toc529906965)

[3. ábra SCRUM folyamat vázlatos ábrája[9] 14](#_Toc529906966)

[4. ábra Trello tábla 16](#_Toc529906967)

[5. ábra Trello kártya szerkesztése 16](#_Toc529906968)

[6. ábra Redmine felület 17](#_Toc529906969)

[7. ábra Az adatbázisszerkezet 27](#_Toc529906970)