Munkanapló

Benák Ádám – JQDQ5I

**Első hét:**

**Órák száma: 23, 2022.03.07-2022.03.10**

Először is megkaptam az asztalom, és a számítógépem. Beállítottam és feltelepítettem a szükséges programokat, többek között a Microsoft Visual Studio Community 2022-t, a Git Bash-t és a Git Sourcetree-t. Kaptam saját fiókot melyen a továbbiakban megkaphatom a rám kiosztott feladatokat.

Megtanultam a Git verziókövető rendszer működését, és használatát egy online könyv segítségével. Konzolos formában a Git Bash-t használtam, valamint megismerkedtem a Git használatát könnyítő grafikus felülettel, a Git Sourcetree-vel. Amit tanultam arról példákat készítettem és gyakoroltam, hogy működne élesben. Csináltam saját Git repository-t branch-ekkel, és generáltam conflictokat és a Stash használatát is kipróbáltam.

A Git elsajátítása után elkezdtem tanulni a C# nyelvet. Visual Studio-t használtam a gyakorlásához. Itt is példákat készítettem. Mindenhez külön osztályt hoztam létre és ezeket példányosítom a Main függvényben a példák futtatásához. Ezzel gyakorlatilag egy példatárat is készítettem, ami hasznos lehet a későbbiekben, ha valami nem jut eszembe.

Tehát ezen a héten a C#-ban megismerkedtem az alapokkal. A .NET keretrendszertől és egy szimpla Console.Writeline() függvénytől kezdve az osztályok létrehozásáig és példányosításáig. Megtanultam továbbá az öröklődést és a láthatóság szabályait, a destruktort és a static kulcsszó használatát függvények és változók esetén is, a nullable használatát és a két dimenziós tömböt.

**Második hét:**

**Órák száma: 12, 2022.03.17-2022.03.18**

A hosszúhétvége miatt csak két napot voltam a héten. Ez idő alatt folytattam a C# tanulását. Megismerkedtem a következő fogalmakkal: sealed class, internal class, const és readonly változó közötti különbség, virtuális metódus és ősosztály metódusainak felülírása override kulcsszó használatával.

Reference type és Value type közötti különbség, itt szóba jött természetesen a ref kulcsszó, de az in és out paraméter kulcsszavakat is áttekintettem, mert relevánsak a témában. Létrehoztam saját kivétel osztályt és megtanultam a hibakezelést, valamint a namespace-ek használatát projekthez hozzákapcsolt másik projekt esetében is, melyet íródhatott akár más nyelvben is például Visual Basic-ben. Ide kapcsolódik az internal kulcsszó. Megtanultam még a Stringbuilder, Listák és Dictionary-k működését, és ez után jöhettek a nagyobb témakörök C#-on belül.

Az egyik ilyen a Generikus típus. Létrehoztam generikus osztályokat, generikus listát és generikus Delegate-et, mely pedig a másik nagyobb témakör. Itt az „egyszerű” delegate létrehozása után készítettem Action, Func és Predicate speciális delegate típusokat, melyek egyfajta callback function-nek feleltethetőek meg. És természetesen összekötöttem a két témakört és Generic Delegate-et is létrehoztam.

**Harmadik hét:**

**Órák száma: 22, 2022.03.21-2022.03.25**

Ezen a héten megtanultam két nagyobb témakört: a LINQ lekérdezéseket és a párhuzamos programozás alapjait. A LINQ lekérdezéseket megnéztem query comprehension, és lambda formában is, de a továbbiakban az utóbbit fogom használni, mert az a szimpatikusabb és azt is használják a cégnél. Írtam összetettebb lekérdezéseket is, allekérdezésekkel, és kipróbáltam a legtöbb és leggyakoribb LINQ metódusokat.

Ezek után megnéztem a párhuzamos programozást, mely során érintettem a thread-eket, threadpool-t párhuzamos for-ciklust és a task-ok létrehozását és használatát. Szóba jött még a ThreadStatic változó, a kritikus régió és a thread-ek adatait lekérdező és módosító metódusok (prioritás, név, háttérben futás lekérdezése). Még ehhez a témakörhöz tartozik, de külön nagy témaként is említhetném az aszinkron programozást, mivel szorosan összekapcsolódik a kettő. Ide tartozik természetesen az async és az await kulcsszó, és itt vesszük hasznát a task-oknak.

Majd elkezdhettem a következő nagyobb témát, mely már túlmutat a C# nyelven, de a .NET környezeten még nem, ez pedig az adatbázis kezelés ADO.NET segítségével. Kezdésként telepítettem a Microsoft SQL Server Management Studio-t és készítettem egy adatbázist Visual Studio-ban hogy kipróbáljam. Összekapcsoltam a kettőt, és írtam egyszerű lekérdezéseket MSSMS-ben.

**Negyedik hét:**

**Órák száma: 23, 2022.03.28-2022.03.30**

Készítettem egy alkalmazást, a MSSMS-ben létrehozott adatbázis tábláinak lekérdezésére, módosítására. Ezt kiegészítettem prepared statement használatával, és a korábban megismert szálkezeléssel, hogy biztonságos és reszponzívabb legyen. MSSMS-ben írtam tárolt eljárásokat, melyeket az alkalmazásban hívok meg, és megnéztem, hogy lehet jogosultságokat létrehozni, és azokhoz engedélyeket társítani. Ez összefügg a tárolt eljárásokkal, mert például be lehet állítani, hogy bizonyos felhasználó csak azokon keresztül tudja az adatbázist lekérdezni.

Csináltam egy egyszerű weboldalt is ASP.NET használatával, melyben kérdéseket lehet írni és keresni közöttük, valamint lehet regisztrálni és belépni.

Ezek után elkezdtem az API-kkal foglalkozni. Az API-k kezelésére kipróbáltam a Swagger oldalt, valamint az Insomnia és Postman alkalmazásokat is. Mivel az API-k Json formátumban vannak, ezért azok használatát is megtanultam, mint például a szerializálás és deszerializálás. Megnéztem, hogy kell C#-ban lekérni API-t és felhasználni RestSharp segítségével.

**Ötödik hét:**

**Órák száma: 19, 2022.04.04-2022.04.07**

Megkaptam az MPL API technikai leírását és kijegyzeteltem a fontosabb lépéseket az általános feladatok megvalósításához, mint például: szállítmány létrehozása, szállítmány címiratának lekérdezése, ragszám szerinti törlése, vagy, hogy egy megrendelést hogy lehet több csomagban feladni. Ezekhez tokent kell kérni először egy http POST kéréssel, majd a token 3600 másodpercig érvényes. Ezután lehet elküldeni a kívánt kérést.

Elkezdtem kidolgozni ezekhez a kérésekhez egy alkalmazást, mely az adatok kitöltése után elvégzi mindezt, esetleg adatbázisba menti a visszakapott adatokat. Mivel még kulcsokkal nem rendelkeztem az API használatához, de a JSON sémákat láttam technikai leírásból, így elkezdtem megcsinálni a szállítmány létrehozásának POST kérésének a vázát. Osztályokat hoztam létre, ezeket egymásba ágyaztam, és végül szerializáltam az egészet JSON formátumba, így visszakaptam a leírásban látott JSON-t.

**Hatodik hét:**

**Órák száma: 8, 2022.04.11**

Ezen a héten, betegség miatt csak egy napot voltam. Ez alatt megcsináltam a teljes grafikus felületet egy szállítmány feladásához, mely a beírt adatokat összegyűjti, példányosítja velük az osztályokat, és az egészet Json formátumba alakítja. A jövőben képes lesz el is küldeni.

**Hetedik hét:**

**Órák száma: 17, 2022.04.19-2022.04.21**

Ezúttal megkaptam az MPL API-hoz tartozó felhasználónevet, és jelszót. Ezek segítségével tokent kértem postman-ben, majd a token, és még pár hiányzó adat segítségével szállítmány kérés üzenetet és küldtem az API-nak és megkaptam a válaszban a címke bájtkódját. Ezt pdf-be alakítva ki lehet nyomtatni majd a leendő csomagokra matricaként.

Ugyanezeket megcsináltam C#-ban is és beépítettem az alkalmazásomba. Így ha kitöltöttük az adatokat, és rányomunk a szállítmány létrehozása gombra, akkor először tokent kér, majd a token felhasználásával leadja a kérést. Minden kéréshez egyedi Guid azonosítót generál, mely szükséges a leírás szerint. Kibővítettem az alkalmazást, hogy a válaszban kapott címke bájtkódot átalakítsa és lementse pdf-be.

**Nyolcadik hét:**

**Órák száma: 24, 2022.04.25-2022.04.28**

Kipróbáltam az MPL API többi releváns részét is. A szállítmány létrehozása volt a legfontosabb, ezt valósítottam meg C#-ban is az előző hetekben, mert ha ez meg van, akkor a többi könnyen megvalósítható ezt sablonként használva. Ezúttal pedig kipróbáltam a szállítmány címiratának lekérdezését, ragszám szerinti törlését, új csomagok felvételét meglevő nem együtt kézbesítendő szállítmányba, valamint szállítmány lezárását. Az éles rendszerben ezek lesznek használatban.

Ezek után tökéletesítgettem az alkalmazásom GUI részét, főként az input mezők validációját ellenőriztem, és parciális osztályokkal segítettem a láthatóságot.

Megtaláltam a csomagfelvevő pontok listáját a Posta oldalán. Ezek egy XML fájlban voltak kilistázva. Írtam egy programot, ami kiszedi pontokat, aszerint hogy Csomagpont, Postapont, Vagy Postán maradó csomag. Az így kapott 3 listával töltöttem fel a GUI-m egyik Combobox-át, ami aszerint változik, hogy egy másik mezőben mi van kiválasztva. Ezt binding list-el valósítottam meg.

**Kilencedik hét:**

**Órák száma: 15, 2022.05.02-2022.05.04**

Átnéztem a Git használatát, és kipróbáltam, Git bash, Git Sourcetree és Kdiff nevű program használatával, Githubon és Gitlabon is.

Csináltam egy adatbázist az itt használt GLS adatbázist mintázva, melybe a programom immáron feltölti az adatokat minden egyes szállítmány feladás után. Be lett kötve a címke nyomtatóm is, úgyhogy következő lépésnek azt tervezem, hogy a PDF-be lementett címkét ki is nyomtassa szállítmány feladás után.

Felújítottam a programom kinézetét Telerik Winforms segítségével. Ezt a külsős kiegészítőt Nuget Package-ek közül kellett telepíteni, és kaptam bejelentkezési adatok a használatához, mivel fizetős csomag. Következő lépésnek táblát teszek bele, melyet lazy loading-gal fogok feltölteni, így sok adat esetén sem kell sokat várni a betöltésre, mivel részletekben fog az lezajlani, például tízesével.

**Tízedik hét:**

**Órák száma: 27, 2022.05.09-2022.05.12**

Megkaptam a hozzáférést a backend kódhoz, így elkezdhettem beintegrálni a megvalósított MPL api megoldásomat, hogy élesben is működjön, de először csak az éles rendszer tesztrendszerén. Nagyjából megértettem a kód ezen részének működését, és logikáját, és sikerült beillesztenem a saját megvalósításom.

Azzal kezdtem, hogy kerestem már korábban megvalósított mintát, hogy azt vegyem példának. Szerencsére kettő ilyen is akadt, egyik már használatban van, a másik pedig később lesz beiktatva. Az előbbit vettem alapul, de az én módszeremmel ötvözve a végeredmény eltér ettől. Erre azért is volt szükség, mert az MPL esetében nem rendelkeztem olyan előre elkészített könyvtárakkal, melyekkel például könnyedén szerializálhatom az osztályokat az adatbázisba lementett kódból.

Csináltam egy model mappát, amiben azok az osztályok szerepelnek, amikből szerializálással lesz az api hívás törzse. Egy Manager osztályban csinálom meg a szerializálást, egy Communication osztályban pedig REST kapcsolatot létesítek valamint tokent kérek minden, hogy minden api hívásnál legyen érvényes token. Elkészültem az integráció nagy részével, a következő héten lejjebbi szintekre megyek az mvc modelben ahol az adatbázis kapcsolatok létesítése zajlik.

**Tizenegyedik hét:**

**Órák száma: 20, 2022.05.18-2022.05.20**

Befejeztem az integrációt, olyan értelemben, hogy minden szükséges metódust megvalósítottam minden szinten, de még nem működik minden.

Leteszteltem a programot, úgy hogy az oldal staging részén rendeltem egy könyvet, majd lefuttattam a kódot és számláztam a könyvet, amit rendeltem. Ehhez telepítenem kellett az MPL címiratkitöltő alkalmazását, és futtatnom miközben a számlázás zajlik.

Ez esetben egy címkét kellett volna nyomtatni a programnak, ami nem történt meg így folytatnom kell a hibajavítást. Breakpointokat helyeztem el a kódban bizonyos metódusoknál és figyelem, hogy mikor mi romlik el, ami akadályozza a működést.

Felvettem egy új oszlopot az staging adatbázisba, az MPL táblájában, ami a címkét fogja tárolni. Valamint nagyobb problémákba ütköztem az api korlátoltsága miatt, ami valószínűleg frontend változtatást fog maga után vonni.

**Tizenkettedik hét:**

**Órák száma: 16, 2022.05.24-2022.05.26**

Működik a program. Ha rendelek egy könyvet majd számlázom, akkor az MPL api-val is kommunikál, és kinyomja a fuvarlevelet a megfelelő adatokkal. Valamint lementődnek az adatok az adatbázisba, az MPL táblájába a címkével együtt.

Van még egy kis munka vele, még nem tökéletes minden, de a lényeg meg van, és le lehet tesztelni. Kikommenteztem az MPL korábbi megvalósítását, és így is működik minden. Már nincs szükség az MPL címiratkitöltő alkalmazásra a használatához.

Hogy minden adat megfelelően szerepeljen a címkén, a nehézséget az okozta, hogy a könyv súlyát is fel kell rajta tűntetni, vagyis el kell küldeni az api-nak. Ugyanis ezt nem tároljuk el az adatbázisban, pedig egyszerűbb lenne onnan lekérni, hanem számlázás közben kell megadni, és runtime kell lekérnem és „lejjebb” továbbítanom a view rétegről.

**Tizenharmadik hét:**

**Órák száma: 25, 2022.06.01-2022.06.03**

Megvalósítottam a sztornózást is, így ha sztornóznak egy csomagot, az api szinten is törlődik, és visszakerül, a még nem számlázott tételek közé. Az MPL táblára felvettem még két oszlopot - korábban a labelt - most egy olyat, amivel össze lehet kapcsolni a számlázott tételek táblájával, valamint egy törlés dátuma oszlopot, mely akkor kap értéket, ha sztornóztuk a tételt. A számla id oszlop problémát okozott a visszáru kezelésnél, mert nem volt egyértelmű már a korábbi sql lekérdezésekben hogy melyik id oszlopra vonatkozik a where feltétel. Ezt egy view táblába kellett javítani, valamint egyes sql lekérdezésekben.

Ezek után Martin kollégámmal teszteltük, hogy minden működik e, a hibákat javítottam, és jelenleg minden úgy működik, ahogy kell. Vár még egy komolyabb tesztelésre, és ha azon is átmegy, akkor mehet ki élesbe.

Mivel módosítottam az adatbázist, ezért a backend kódban létre kellett hoznom egy sql file-t, amiben a módosítás script-jét beleírtam. Erre azért van szükség, hogy ha esetleg valami történne az adatbázissal, ezeket a scripteket sorban lefuttatva visszakapjuk az eredetit. Emellett logoltam is benne hogy mit módosítottam.

Elkészítettem a jegyzék zárásához a megfelelő metódusokat. Két félét: az egyiket egy idő intervallummal kell meghívni, és azokat a szállítmányokat fogja lezárni amik a két intervallum között vannak. A másik egy szállítmányt kap, és csak azt az egyet fogja lezárni. Valószínűleg az előbbi kerül majd felhasználásra.

Ezen a héten először otthon is töltöttem pár órát, melyet a Jaspersoft nevű program megismerésével töltöttem, mely egy statisztika készítésre alkalmas alkalmazás. Csináltam benne kördiagramot, és sql lekérdezést, ugyanis lehet benne adatbázishoz kapcsolódni, és az így kapott adatokról statisztikát készíteni.

Végezetül bővítettem a C# tudásom a null-coalescing operátorral, mely összehasonlítást végez: ha a bal oldalán lévő operandus nem null, akkor azt adja vissza, egyébként a jobb oldalit. Jelölése ??, de újabb C# verziókban a ??= is megjelent. Fontos tudni, hogy ha a bal oldali operandus nem null, akkor a jobb oldali kifejezés már ki sem fog értékelődni.

**Tizennegyedik hét:**

**Órák száma: 28, 2022.06.08-2022.06.10**

Elvégeztem pár simítást, dokumentáltam a metódusaimat, ahogy azt az elődeimtől is láttam, végeztem egy „fa rázást” amit annyit jelent, hogy a nem használt importokat kitöröltem. Továbbá a szükséges adatokat beírtam, amik még hiányoztak a képletből, például az extraszolgáltatások, és a cég bankszámlaszáma. Az adatbázis kezelő metódusokat átszerveztem egy külön osztályba, ahol csak api-hoz tartozóak vannak.

Elkészítettem az api finalize metódusát, ami lezárja a kapott szállítmányokat, és így kerülnek majd tényleges feladásra. Ehhez egy külön gombot is elhelyeztem, a többi futárszolgálat kezelésének mintájára. Ezzel elkészültem első fő feladatommal, az MPL api bevezetésével.

Következő feladatom az „MPL Házhozszállítás” szállítási mód bevezetése lesz, melyhez frontend változtatás is kelleni fog majd, hogy ki tudjuk rendelésnél választani a megfelelő szállítási módot. A backend részével végeztem is, nagyjából annyi volt a dolgom, hogy felvettem az új szállítási módot az adatbázisba, és amikor számlázásnál nem található a szállítási mód (isnull) akkor házhozszállítást írok a helyére az api-ban. Ezen felül beállítottam, hogy minden ugyanúgy működjön, mint a postapont esetében, hiszen ugyanazt az api-t használja.

Elkezdtünk egy tréninget a Jaspersoft használatáról, eddig körülbelül 6 órát foglalkoztunk vele, ez idő alatt számos dolgot megtanultunk, melyekre majd később szükség lesz belsős statisztikák készítése céljából. Első lépés volt a szerverhez és az adatbázishoz való csatlakozás. Ezekből lehet lekérdezni sql vagy plsql segítségével, majd a lekérdezett adatokat szabadon felhasználhatjuk a statisztikák készítéshez. Ezeken felül létrehozhatunk változókat, vagy paramétereket. Utóbbi arra való, hogy a statisztika lefuttatása előtt megkérdezi, mi legyen a paraméter, és ez alapján alkalmaz megszorításokat.

**Tizenötödik hét:**

**Órák száma: 24, 2022.06.13-2022.06.15**

Találtam egy nem kezelt hibát a belső rendszerben, melyről feladatot vettem fel. A feladat végül hozzám került kiosztásra, és meg is oldottam. A hiba azzal volt, hogy a számlázó felületen, ha egy szűrt keresés nem ad eredményt és így próbáljuk a „semmit” sztornózni, vagy nyomtatni akkor nem kezelt kivételbe botlunk. Ezt úgy kezeltem, hogy megnézem azt is, hogy a kiválasztott sor null e.

Felvettem az MPL házhozszállítás árait az erre kitalált táblába, valamint van egy szótár tábla is ahova szintén fel kellett venni, mint szállítási metódus. A rendelés tábla kapott egy megjegyzés oszlopot, melybe házhozszállítás esetén, az oldalon megadott „megjegyzés a futárnak” kerül. Ezt számlázás előtt a belső rendszerben is módosítani lehet, és frissíti az adatbázisban. Erre csináltam egy új panelt két beviteli mezővel. Az egyiken a telefonszámot a másikon a megjegyzést lehet módosítani, mert erre szükség lehet, tapasztalatok alapján. Mindkét mező default text-jét az adatbázisból kérdezi le. Megvalósult az is, hogy ez a panel csak akkor elérhető, ha házhozszállításos rendelés van abban a pillanatban kiválasztva, egyébként ki van szürkítve.

Végül a tesztelő kollégával nagyjából leteszteltük a programot, és miután ezen átment, és a felmerülő hibákat javítottam, beadtam egy merge requestet. Az MPL-nek pedig el lett küldve pár példa rendelés lezárással, hogy ellenőrizzék, hogy minden megfelelő e.

**Tizenhatodik hét:**

**Órák száma: 12, 2022.06.23-2022.06.24**

kuponkezeles martin

hangfire, azure

**Tizenhetedik hét:**

**Órák száma: 5, 2022.06.28**