Széchenyi István Egyetem

Informatika Tanszék

CASE-technológia

[NGB_IN027_1]

Összegző dokumentáció a

Éttermi rendelésfelvevő és számlázó szoftver

című feladat fejlesztésének megvalósításáról

Témát kidolgzó csoport kódja: XX

Készítették:

_____

_____

_____

Tartalomjegyzék

1. Problémadefiniálás, helyzetelemzés

1.1 A projektműködés rendje

1.1.1 A csoport működése

A csoportunk 3 főből áll. A tagok: _____. Csapatunk minden tagja aktívan részt vesz projektünk munkálataiban.

1.1.1.1 Kommunikáció/Dokumentálás

A gyorsabb kommunikáció érdekében létrehoztunk egy *Facebook* csoportot. Csoportunk heti rendszerességgel tart megbeszélést, hogy személyesen megvitassuk az elkövetkezendő feladatokat. Egyénenként összegyűjtjük a felmerülő lehetőségeket a következő lépéssel kapcsolatban. Ezután egy előre megbeszélt helyen és időpontban egyeztetjük gondolatainkat. A meeting-en közösen hozzuk létre a dokumentációnkat, melyet *Microsoft Word*-ben rögzítünk.

1.1.1.2 Cél

Elsődleges célunk egy olyan éttermi alkalmazás létrehozása, amely egy helyen tartalmazza az étlapon megtalálható összes ételt. A pincérek számára gyorsabb, pontosabb rendelésfelvételt biztosít és ezáltal a szakácsok sem tévednek.

1.2 Üzleti igényspecifikáció

1.2.1 Probléma definiálása:

Fejlesztőcsapatunk olyan problémával szembesült, amelyre szeretnénk minél előbbi megoldást javasolni.

Problémaként felmerült, hogy (egy jól menő étteremben is) a pincér figyelmetlenné válik a sok rendelés felvétel miatt, így összekeveri vagy akár felre érti a kívánt ételeket, amely a fogyasztó vendégek számára elég kellemetlen lehet. Előfordulhat, hogy kevés a pincér, így lassúvá válik a kiszolgálás. Ez szintén nem jó fényt vet az étteremre. A sok vendég miatt a pincérek megpróbálják gyorsan összeírni a rendelést ám ez olvashatatlanná válik, ezáltal megnehezítik, lassítják a szakácsok munkáját. Így az is megesik, hogy nem megfelelő ételt tálalnak fel a fogyasztók számára.

A következőkben különböző elemzési technikákkal (SWOT, Ishikawa, Pareto stb.) feltárjuk az észlelt problémát, ezzel elősegítve a megoldás megtalálását.

1.2.2 SWOT-analízis

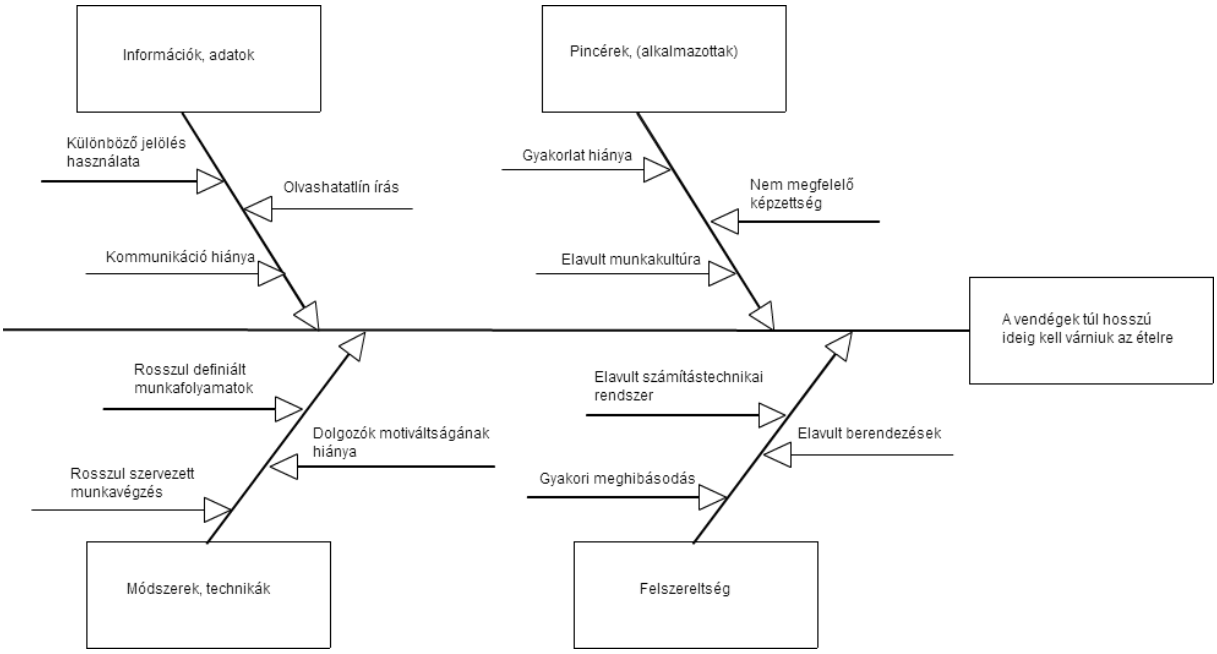
SWOT-elemzés	Segítik a célok elérését	Gátolják a célok elérését
Belső tényezők (szervezeti jellemzők)	Erősségek • Kialakult törzsvendégi kör • Kellemes környezet • Hazai ételek • Vendégek pozitív véleménye	Gyengeségek • Lassú a kiszolgálás • Nehézkes kommunikáció • Elavult munkakultúra • Rosszul szervezett munkavégzés • Számlázás
Külső tényezők (környezeti jellemzők)	Lehetőségek • Új vendégek megnyerése • Informatikai fejlesztés • Reklámozás • Terjeszkedés • Nagyobb profit realizálása	Fenyegetettségek • Új versenytársak megjelenése • Vásárlói igények és ízlések változása • Kedvezőtlen demográfiai változások • A verseny fokozódó nyomása • Kiadások növekedése

1.2.3 Pareto elemzés

Probléma azonosítása	Előfordulási gyakoriság	Relatív gyakoriság	Halmozott gyakoriság
1. Különböző jelölés hiánya	16	0,123077	0,123077
2. Olvashatatlan írás	15	0,115385	0,238462
3. Kommunikáció hiánya	14	0,107692	0,346154
4. Gyakorlat hiánya	13	0,1	0,446154
5. Nem megfelelő képzettség	11	0,084615	0,530769
6. Elavult munkakultúra	10	0,076923	0,607692
7. Rosszul definiált munkafolyamatok	9	0,069231	0,753846
8. Rosszul szervezett munkavégzés	8	0,061538	0,815385
9. Dolgozók motiváltságának hiánya	5	0,038462	0,907692
10. Elavult számítástechnikai rendszer	4	0,030769	0,938462
11. Elavult berendezések	3	0,023077	0,992308
12. Gyakori meghibásodás	1	0,007692	1

1.2.4 Ishikawa-diagram

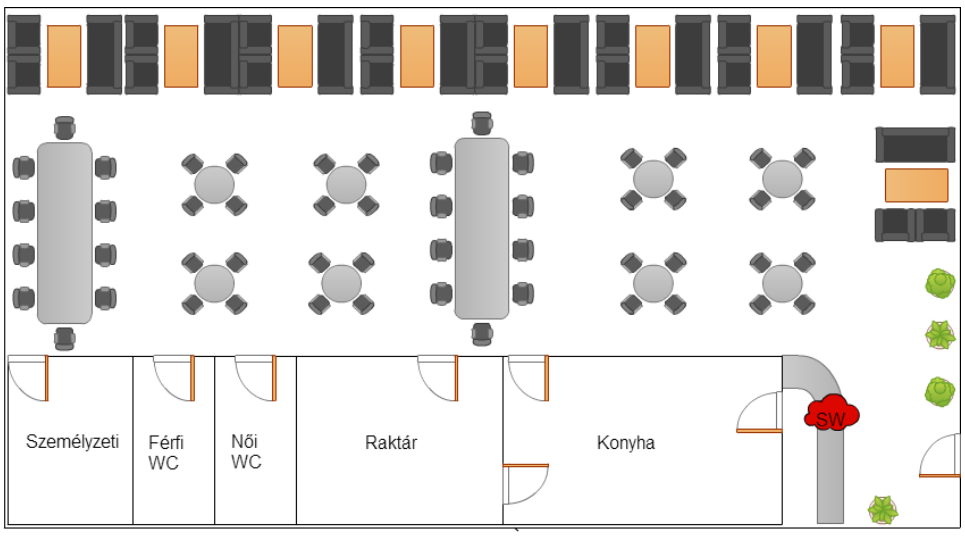
Az Ishikawa elemzési módszer (fishbone, halszálka diagram) az ok-okozati összefüggések megkeresésére és rögzítésére szolgál. A diagram készítésekor a probléma tartalmára, lényegére tudunk koncentrálni, meghatározni magát a problémát, majd számba venni a lehetséges okokat. A diagramon a gerinc jobb végéhez egy téglalapot illesztve megjelöljük a vizsgált problémát, a gerinchez kapcsolódó ágak végén szintén egy téglalapban az ok-kategóriákat, majd az egyes ágak szálkáihoz hozzárendeljük a problémát okozó tényezőket.



A feltárt probléma Ishikawa-diagramja

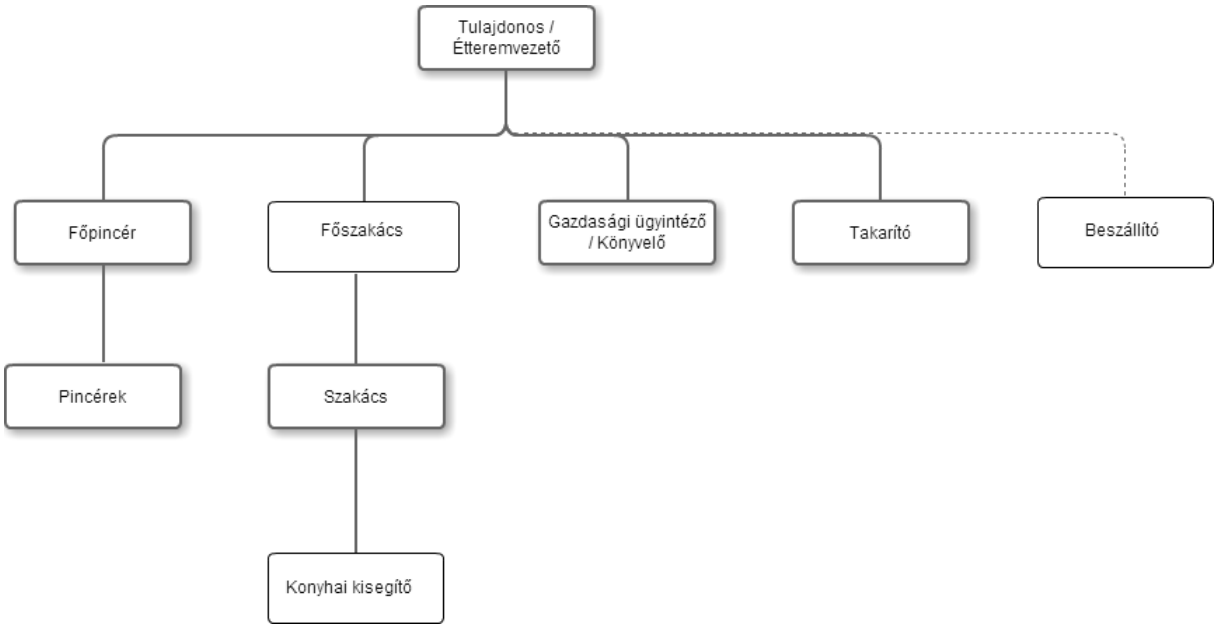
1.2.5 A horizontális struktúra

A horizontális struktúra a szervezetek különböző egységeinek a területi elhelyezkedését ábrázolja. Ilyen az alábbiakban ábrázolt étterem alaprajza is.



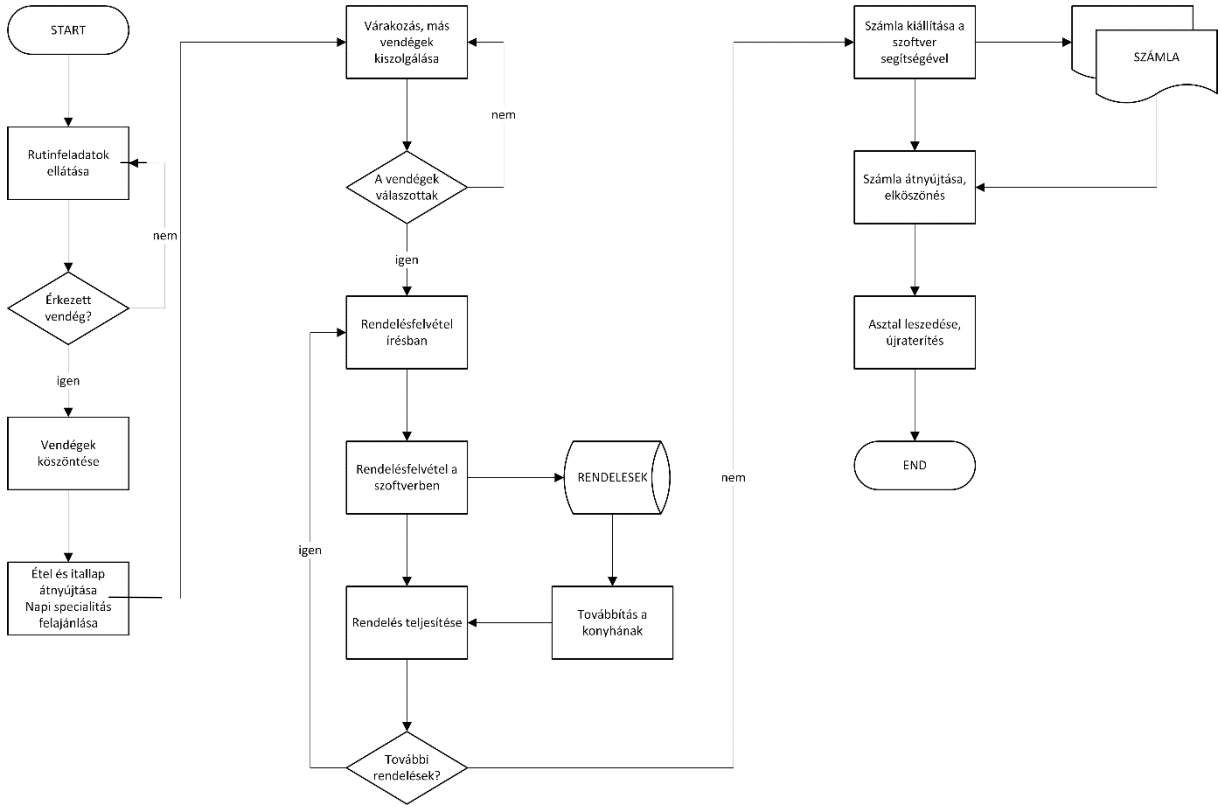
1.2.6 A vertikális struktúra

A vertikális struktúra a szervezet különböző irányítási szintjeit és a szervezeti egységek egymáshoz való viszonyát ábrázolja. Az alábbiakban a Resto Étterem szervezeti felépítése látható.



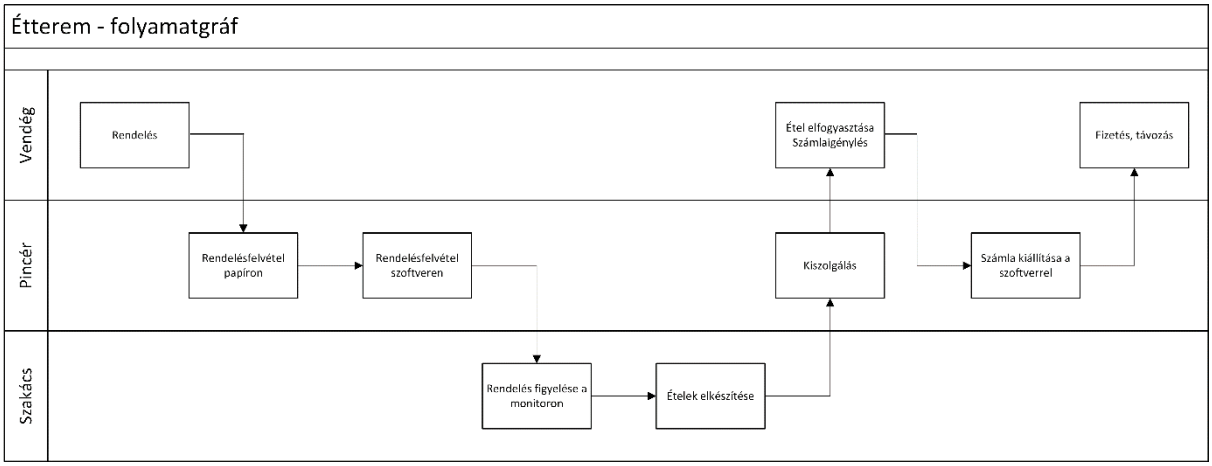
1.2.7 Folyamatábra, folyamatgráf

A folyamatábra kiválóan szemlélteti egy adott tevékenységsorozat elemeit. A feladatok egymásutániságát mutatja be, az egyes feladatok végrehajtásához szükséges bementekkel és a kimenetekkel. A folyamtok menetének irányát nyilakkal jelöljük, amik csak lefelé vagy jobbra mutathatnak, kivéve a visszacsatolást jelző felfelé mutató nyilakat. Az adott probléma feltárásában fontos szerepet játszik a fejlesztendő szoftver leendő használóinak tevékenységi körének elemzése. Esetünkben a szoftver elsődleges felhasználói az étterem pincérei lesznek, így az ő tevékenységüket elemezzük. Egy étterem pincérének tevékenységét az alábbi ANSI szabvány szimbólumaival jelölt folyamatábrán láthatjuk:



A rendelésfelvétel folyamatábrája

A folyamatgráf (Process Graph) az egyes szervezeti egységek által végzett feladatokat és azok végrehajtásának sorrendjét szemlélteti. A folyamatgráf szemléletes módszer az adott szervezet folyamatainak az elemzésére. Esetünkben nem szervezeti egységek lesznek a folyamatok végrehajtói, hanem az étterem dolgozói, és az étterembe érkező vendégek. Az étterem folyamatgráfja a következő:



2. Megoldási javaslatok a problémára

A Resto Étteremben felmerülő problémák, hiányosságok ismeretében az alábbiakban feltárjuk a felmerülő lehetőségeket. Megvizsgáljuk milyen alternatívák állnak rendelkezésre jelenleg a szoftverpiacon, számunkra ezek mennyire megfelelőek. A vizsgálat során végiggondoljuk, hogy a fejlesztési feladat hogyan valósítható meg. A döntést a rendelkezésre álló lehetőségeknek az ismeretében hozzuk meg.

2.1 I. MozaiX Software Studio – Éttermi rendszer

Ez a rendszer egy új megközelítéssel készült, melynek végeredményeként kisebb személyzettel lehet a vendégeket kiszolgálni. A vendég az étterembe történő belépésekor kap egy bankkártya méretű azonosító kártyát, amelyet majd a fizetéskor ad csak vissza. Ez a kártya csak azonosításra szolgál, a fogyasztási adatok a rendszer adatbázisában tárolódnak.

Az alkalmazás arra épül, hogy a vendégek az asztalba beépített érintőképernyős terminálon – azonosítás után – informálódni tudnak az ételekről-italokról, majd egy egyszerű, áttekinthető felületen meg is tudják adni rendeléseiket. A megrendelések azonnal megjelennek a pultnál és a konyhában elhelyezett képernyőn. A pincér csak ezután viszi ki a rendelést.

A vendég az asztali terminálon nyomon követheti számláját és különböző üzeneteket küldhet a pincérnek, ami neki a PDA-ján jelennek meg hangjelzés kíséretében. Fizetéskor így a pincér könnyedén leolvassa az együtt fizető vendégek kártyáit a PDA-ján, majd kinyomtatja a hordozható nyomtatóján a blokkot (számla készítésére is van lehetőség).

A rendszer naplózza az egyes események időpontjait ezáltal értékes információkat kapunk, a vendégek szokásaikról és az alkalmazottak reakcióidejéről. A háttér rendszer fő feladata a törzsadatok karbantartása, lekérdezések biztosítása, készletnyilvántartás.

Az asztalba beépített érintőképernyős terminálon, a Vendég asztal program segítségével, kártyás azonosítás után a vendég: megrendelést vehet fel, tájékozódhat az ételekről, illetve az italokról, értékelheti azokat, valamint kérheti a pincért.

Ehhez a típusú éttermi rendszerhez tartozik még egy Pult program is, amely feladata az asztaloktól beérkező megrendelések megjelenítése és a kiadás jóváhagyása. Program érintőképernyőn is megjeleníthető.

2.2 II. Kézi pincér terminál

Ez egy olyan megoldást jelentene, melynek során minden pincér számára biztosítunk 1-1 saját PDA berendezést. Így biztosítva a mobilitást, mellyel a hibázás lehetőségét egészen kicsire minimalizáljuk. Mivel a PDA összeköttetésben áll a konyhával, így a rendszer képes az azonnali kommunikációra, melynek hatalmas előnyei közé sorolható, hogy így a pincér azonnal tájékoztatni tudja a vendéget, ha valami éppen elfogyott, továbbá a kiszolgáló a konyha és a vendégtér közötti rohangálást megspórolva több időt tud a vendéggel foglalkozni. Ennek következtében a vendégek is elégedettebbek – megrendelt ételeket gyorsabban szervírozzák, a számla helyben, a vendég asztalánál készül, stb. A kiszolgálás gördülékenyebbé válik, ezáltal az asztalok gyorsabban forognak és a bevétel is megnő.

Továbbá a rendszer előnyei közé sorolható, hogy nagy hatótávolságú, pontos, megbízható. Viszont minden pincérnek rendelkeznie kell 1-1 PDA-val illetve az arra telepített szoftverrel.

2.3 III. RESTO szoftver

SZOLGÁLTATÁSOK:

- A rendelések felvétele:
  - ételek/italok adatai
  - asztalok adatai
  - pincérek azonosítása/adatai
- Lekérdezések, statisztikák készítése:
  - legkedveltebb ételek
  - legkedveltebb italok
- Számla, blokk kitöltése és nyomtatása.

RÁFORDÍTÁSOK:

- Minimális hardver igény: Intel ATOM D525, Intel D2700 vagy Intel N2600/N2800 processzor
- 512 MB RAM
- 16 GB tárhely a tárolt adatoknak.
- 1 db POS terminál
- 1 db nyomtató

- 1 db monitor

MINIMÁLIS SZOFTVER IGÉNY:

- Microsoft Windows 7 vagy későbbi operációs rendszer
- Képkezelő program
- Total Commander

1 MUNKAÁLLOMÁS lesz elérhető, itt lesz elhelyezve a szoftver, a konyhai monitornak csak a rendelés kijelzésében van szerepe.

BERUHÁZÁSI KÖLTSÉGEK:

- 1 db új munkaállomás 17” síkmonitorral: 200.000 Ft
- 1 db síkmonitor: 30.000 Ft
- 1 db mátrixnyomtató: 40.000 Ft
- hálózat kiépítése: 50.000 Ft
- szoftverfejlesztés: 150.000 Ft
- betanítási, programozási és egyéb költségek: 100.000 Ft
- beruházási költségek összesen: 570.000 Ft

ÜZEMELTETÉSI KÖLTSÉGEK:

- energia, papír, alkatrészek, javítási költség, amortizáció a becslések szerint havonta: 20.000 Ft
- bérkiegészítés járulékokkal havi: 30.000 Ft
- összesen havi: 50.000 Ft

ELŐNYÖK:

- gyors hozzáférés az adatokhoz
- biztonságos adattárolás
- kisebb hibalehetőséggel pontos statisztikák, lekérdezések, listák készíthetőek
- egy, fő munkaállomás van

HÁTRÁNYOK:

- adatrögzítéshez csak 1 db gép áll rendelkezésre, így egyidejűleg ezt a feladatot csak egyetlen személy végezheti, ami lassítja a feldolgozást
- nagyobb fájlok mentésére csak a szerveren van lehetőség

Ez a változat akkor javasolt, ha a másik kettő változat egyikét sem tudja az étterem finanszírozni.

2.4 Az alternatívák kiértékelése, döntés a fejlesztésről

A megfelelő döntés meghozatalához összevetettük a specifikációt, illetve a rendelkezésünkre álló ismeretek alapján a megoldási javaslatokat. Esetünkben egy régi, kis családi vállalkozásról van szó, amely nem rendelkezik nagy vagyonnal, jelentős anyagi háttérrel. A RESTO étterem eddigiek során a rendelések felvételét, a konyhába juttatását, a számlázást elektronikus eszköz nélkül, papír formában kézi írással oldották meg. Jelenleg szeretnék az éttermet modernizálni, számítógépes alapokra áthelyezni az eddig jól megszokott munkamenetet. A RESTO igényei a modernizálással kapcsolatban, hogy a rendelések rögzítése, a számlázás számítógéppel megoldott legyen. Illetve a számítógépre felvitt rendelések a konyhában egy monitoron jelenjenek meg, így kikerülve a pincérek olvashatatlan írását. Ezen felül plusz funkcióként szerepeljen, hogy mely ételek a legkedveltebbek. A rendszer kifejlesztésére 1.000.000 Ft áll az étterem rendelkezésére. Ezen információk ismeretében került értékelésre a felmerült négy megoldási alternatíva.

I. MozaiX Software Studio – Éttermi rendszer

Ezen rendszer esetében szükségünk lesz minden asztalhoz egy beépíthető érintőkijelzős terminálra, minden pincérnek egy PDA-ra, és megközelítőleg annyi azonosító kártyára amennyi az étterem vendég befogadóképessége.

Ráfordítások:

- 15 db érintőkijelzős terminál: 3.000.000 Ft
- 4 db PDA: 140.000 Ft
- 60 db azonosító kártya: 120.000 Ft
- szoftver: 300.000 Ft
- betanítás: 100.000 Ft
- **Összesen: 3.660.000 Ft**

Kiértékelés:

A MozaiX Software Studio által nyújtott lehetőség sajnos nem felel meg a tulajdonos elképzeléseinek, ez kizáró ok lehet. Tovább vizsgálva viszont egyértelműen megállapítható,

hogy ennek a megoldásnak a megközelítő végösszege több, mint 3-szorosa a vállalkozás számára rendelkezésre álló összeg nagyságánál. Ezért ez a megoldás NEM merülhet fel, mint végleges változat.

II. Kézi pincér terminál

Ezen megoldás esetében szükségünk lesz minden pincér számára egy PDA-ra, kézi pincér terminálra. Ezen felül a szoftverre és legalább egy munkaállomásra.

Ráfordítások:

- 4 db PDA: 140.000 Ft
- 1 db munkaállomás: 300.000 Ft
- szoftver fejlesztés: 200.000 Ft
- betanítás: 90.000 Ft
- **Összesen: 730.000 Ft**

Kiértékelés:

Az Kézi pincér terminál megoldás sajnos nem felel meg a megrendelő elképzeléseinek. Hiszen ebben az esetben minden pincérnek szüksége van egy kézi PDA-ra és nem egy munkaállomáson folyik a rendelések rögzítése, illetve a blokkolás. Így ez a megoldás nem felel meg a specifikációban rögzítetteknek. Tovább vizsgálva azonban láthatjuk, hogy megközelítő végösszegben ez jóval közelebb áll a tulajdonos meglévő keretösszegéhez, mint a MozaiX Software. Végső soron azonban mivel nem felel meg az elképzeléseknek, ez NEM merülhet fel, mint végleges megoldás.

III. Saját fejlesztésű szoftver

Ebben az esetben szükségünk lesz egy munkaállomásra, magára a szoftverre, ezen felül még egy monitorra és egy nyomtatóra.

Ráfordítások:

- 1 db új munkaállomás 21”-os, SAW érintőképernyős technológiával: 150.000 Ft
- 1 db síkmonitor: 30.000 Ft
- 1 db mátrixnyomtató: 40.000 Ft
- hálózat kiépítése: 50.000 Ft
- szoftverfejlesztés: 150.000 Ft

- betanítás: 60.000 Ft
- egyéb: 40.000 Ft
- **Összesen: 520.000 Ft**

Kiértékelés:

A Saját fejlesztésű szoftver direkt úgy kerül kialakításra, ahogy az a specifikációban rögzítve van. Így teljes mértékben megfelel a megrendelő elképzeléseinek. A költségeket vizsgálva, a megközelítő végösszegből azt láthatjuk, hogy ez belefér a tulajdonos által meghatározott keretösszeg nagyságába, még akkor is, ha esetleg a fejlesztés során felmerülnek kisebb, előre be nem kalkulált költségek. A költségeket és a kialakítás megvizsgálása után arra a következtetésre jutunk, hogy ez MEGFELEL, mint végleges megoldás.

2.4.1 Döntés

A végleges döntés meghozatala előtt alaposan körbejártuk a felmerülő lehetőségeket. Megvizsgáltuk a piacon már fellelhető éttermi szoftvereket, így lett választható alternatíva a MozaiX Software. Ezek mellett felvetettük egy saját fejlesztésű szoftver tervét, amely teljes mértékben a megrendelő igényeihez igazodik, illetve egy modernebb változatot is számba vettük, melynek során PDA használatára helyeznénk a hangsúlyt.

Ez a három megoldás áll rendelkezésünkre, hogy kiválasszuk a legalkalmasabbat a megrendelő számára. Először külön-külön megvizsgáltuk mind a 3 javaslatot. Megvizsgáltuk a funkcióikat, a tudásukat, a szükséges megközelítő ráfordítást. Egy-egy rövidebb kiértékelés feljebb található minden egyes szoftverről, illetve az, hogy ez megfelelő lehet-e mint végleges változat vagy nem. Ezek a kisebb leírások megkönnyítették a döntésünket.

Az I. és a II. változat nagymértékben távol áll az elképzelésektől, hiszen mind a kettő rendszerben a lényeg, hogy a pincérek PDA-n kommunikálnak. Ezen felül az I. változatban még szükség lenne az asztalok számával megegyező mennyiségű, asztalba beépíthető terminálra, illetve annyi azonosító kártyára, amennyi vendéget el tudnak látni az étteremben. Ráfordított költségkeret szempontjából az I. változat jóval meghaladja a megrendelő által rendelkezésre álló keretösszeget, így ez biztosan NEM megfelelő. Míg a II. változat megközelítő keretösszege igaz megfelel a megrendelő költségvetési tervének, de az elképzelésben a PDA-é lenne a főszerep, így az nem megfelelő az étterem tulajdonos elképzeléseinek, ezáltal sajnos ez a megoldás szintén NEM megfelelő.

Utolsó lehetőségként vettük számba és vizsgáltuk meg egy saját fejlesztésű szoftver lehetőségét melyet a III. pont alatt részleteztünk. Ez teljes mértékben a megrendelő igényei szerint, a specifikáció alapján került kidolgozásra. Ebbe a változatba csak azok a funkciók és eszközök kerültek bele, amelyek a megbízó elképzeléseit teljes mértékben kielégítik. Így nem kerültek bele felesleges plusz funkciók, illetve nincs szükség több elektronikai eszköz beszerzésére, mint ami feltétlenül szükséges. Megvizsgáltuk a ráfordított költségeket is, melyből egyértelműen kiderült, hogy a modernizálásra szánt 1.000.000 Ft-os költségkeret elegendő a fejlesztésre még akkor is, ha a fejlesztés során előjön néhány apróbb, előre be nem tervezett, váratlan költség.

Ezek ismeretében meghoztuk a döntést, miszerint a RESTO étterem modernizálására a SAJÁT FEJLESZTÉSŰ szoftvert választjuk, azt valósítjuk meg.

2.5 Business Case – Megtérülés-kimutatás

2.5.1 Emberi ráfordítás mérése:

Egy projekt ráfordításában ez az egyik legfontosabb mérőszám. Egy adott időszak alatt a résztvevők átlagos létszáma alapján határozzuk meg úgy, hogy a létszámot szorozzuk az időtartammal. Tehát 3 ember 3 hónapig dolgozik:

3E * 3H=9EH

2.6 ROI – Megtérülési ráta:

Egy informatikai beruházás esetében is célszerű vizsgálni a beruházás hozamát, azaz, hogy milyen megtérülést sikerül a projekten elérni. Ezt a megtérülési mutatót szokás ROI (Return on Investment) mutatónak nevezni. A mutató kiszámításakor a beruházási értéket vetjük össze egy meghatározott periódusban származó realizált haszonnal:

$$ROI = \frac{Realizált\ haszon}{Beruházási\ érték}$$

	0	1	2	3	Összesen
Hardver	-220000				-220000
Szoftver	-150000				-150000
Bevétel növekedés		100000	100000	130000	315000
Költségcsökkenés		80000	100000	115000	295000
Működési kockázat mérséklése		90000	90000	90000	270000
Összes pénzáramlás	-370000	270000	290000	335000	510000

$$ROI = \frac{510000}{370000} = 1.3783 = 137.83\%$$

Tehát esetünket tekintve megközelítőleg akár 38%-os növekedés érhető el, a meghatározott periódusban.

3. A fejlesztési projekt

A projekt egy adott probléma megoldásához szükséges tevékenységek összessége. A projekteket bizonyos célok elérése érdekében hajtjuk végre. Meg kell határoznunk a projekt kezdési és befejezési dátumát, az egyes tevékenységekhez szükséges időt, és erőforrásokat. Minden projekt adott célra irányul, és adott a projekt megvalósításának ideje, a költségvetés és az erőforrások.

A fejlesztési projekt az egész szoftverfejlesztési életciklust átfogja. A projekt indítása előtt fontos, hogy a célok egyértelműen tisztázottak legyenek, és a lehetséges kockázatokat is azonosítani kell. Előre meg kell tervezni az elvégzendő feladatokat, mérföldköveket és a fejlesztéshez szükséges erőforrásokat. Ezek alapján el kell készíteni a projekt ütemtervét, amely a későbbiekben változhat is, igazodva a fejlesztés menetéhez. A fejlesztői munkában bármilyen szinten résztvevő személyek csoportját projekt team-nek nevezzük. Minél bonyolultabb a fejlesztési projekt, annál nagyobb a team létszáma. A projekt céljainak meghatározása és a felhasználóval való folyamatos egyeztetés rendkívül fontosak a fejlesztés sikerének szempontjából. A fejlesztési projektet sikeresnek mondhatjuk, ha a következő kritériumoknak megfelel:

- elkészül a meghatározott határidőre
- az előzetesen megtervezett költségkereten belül megvalósul
- az előzetesen specifikált funkciók többsége teljesül és megfelel a minőségi elvárásoknak

Hogy ezek a kritériumok teljesüljenek körültekintő elemzési, tervezési, szervezési és menedzselési tevékenységre van szükség, hiszen csak ezen feladatok elvégzésével érhető el a projekt sikere.

3.1 A projekt szereplői

A projekt szereplői azok a személyek, akik bármilyen szinten érdekeltek a projektben, illetve részt vesznek benne. Vegyük sorra az éttermi szoftver fejlesztési projektjének szereplőit és a feladataikat!

Az éttermi szoftverünk fejlesztésének szereplői a következők:

- projektmenedzser
- elemző
- tervező (adatbázis-tervező)
- designer
- programozók
- tesztelők

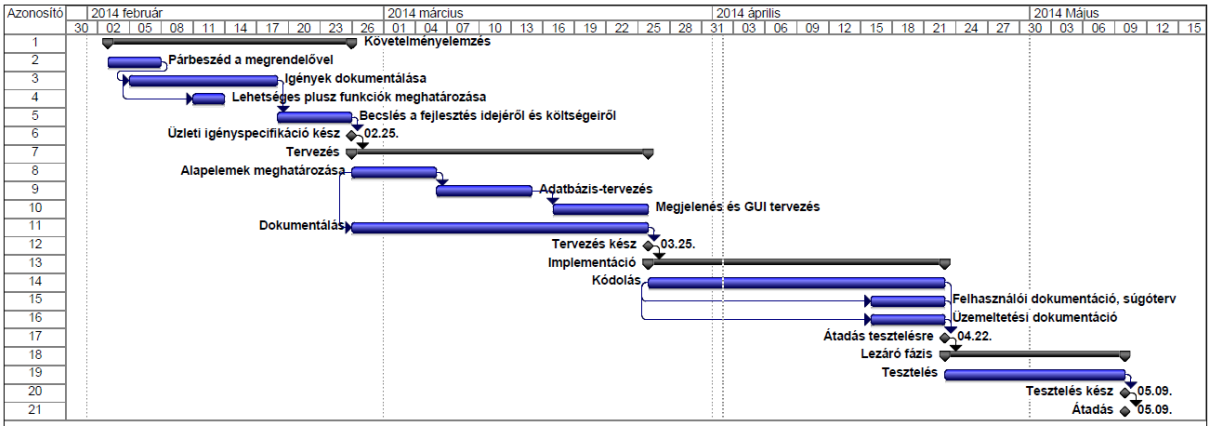
3.2 A fejlesztési ütemterv

A fejlesztési projekt folyamatának megtervezése az összes projekthez szükséges tevékenység meghatározásából és azok függőségi kapcsolatainak megtervezéséből áll. Fontos az egyes mérföldkövek meghatározása is. Szükség van az egyes tevékenységek végrehajtására szánt időnek a megtervezése, ez a projekt időbeli ütemezése. A tevékenységeket és a tőlük függő mérföldköveket az úgynevezett ütemezési táblázatban foglaljuk össze.

3.3 A Gantt-diagram

A Gantt-diagram a projekt időbeli tervezéséhez használatos ábrázolási technika, melyet Henry Gantt fejlesztett ki. A Gantt-diagramban az egyes tevékenységek soronként egymás alatt helyezkednek el, a táblázat fejlécében pedig az időt jelöljük. A tevékenységek egymás utáni sorrendben helyezkednek el aszerint, hogy mikor van a kezdési dátumuk. A tevékenységek mellé ábrázolt sáv megmutatja, hogy mikor kell elkezdeni és befejezni azokat. A Gantt-diagram egy átlátható, könnyen megérthető ábrázolási technika. A projekttervező szoftverek képesek a Gantt-diagram tevékenységeihez kapcsolatokat is rendelni.

A RESTO éttermi rendelésfelvevő szoftver fejlesztési projektjének Gantt-diagramja Microsoft Project 2007 programban megvalósítva a következő ábrán látható:



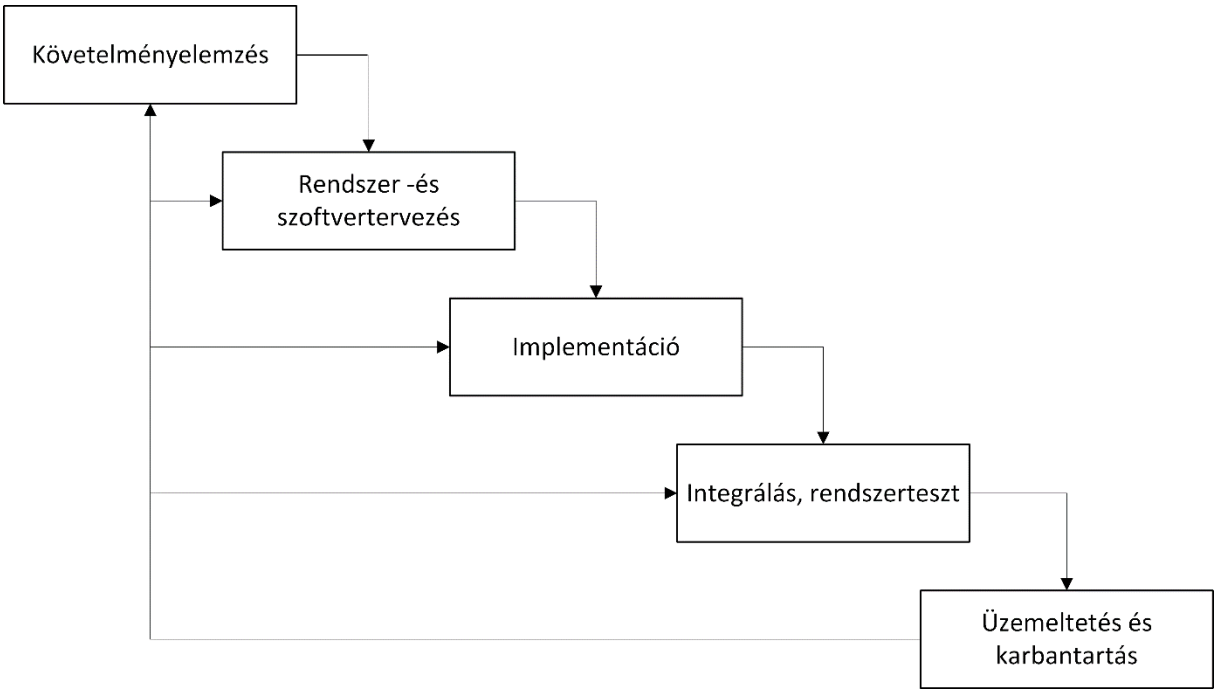
4. A fejlesztési életciklus

A fejlesztés megkezdése előtt törekednünk kell arra, hogy részletesen áttekintsük a fejlesztési igény megjelenésétől az elkészített szoftver üzemeltetéséig terjedő folyamat minden egyes feladatát. Mindegyik fejlesztési folyamatban elhatárolható egy kiindulási szakasz, egy főfolyamat, amelynek részei az elemzési, a tervezési és kivitelezési szakaszok, az üzemeltetés és a karbantartás, végül pedig a szoftver használaton kívülre kerülése.

4.1 Az életciklus modell

Az életciklus modell a fejlesztési folyamatot fázisokra bontja, és ezek egymás utáni végrehajtását írja elő. Az életciklus-modellek a fejlesztési folyamat meghatározott lépéssorozatát adják meg, így az egész folyamatot egy keretbe foglalják. A különböző szoftverek, különböző fejlesztési modelleket igényelnek, és célszerű a fejlesztési folyamathoz leginkább illő modellt választani. Az életciklus modelleknek több változata is van, azonban a fő hangsúly mindegyik esetében a fejlesztési fázisok elemekre bontásán van.

Éttermi szoftverünk fejlesztéséhez az úgynevezett visszacsatolásos vízesésmodellt választjuk.



A visszacsatolásos vízesésmodell

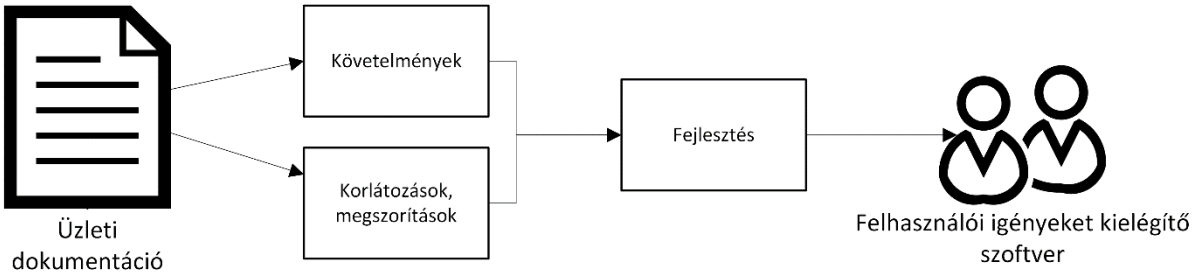
4.2 A vízesés modell

A vízesés modell az első publikált szoftverfejlesztési modell. A vízesés modell lényege, hogy a fázisok egymás után követik egymást, a fázisok végrehajtásával elért mérföldkövek azok lezárására szolgálnak, és minden fázis csak akkor kezdhető meg, ha az előző fázis rendben befejeződött. Ez a modelltípus akkor javasolt, ha a fejlesztők pontosan tudják, hogy mi az elérendő cél és képesek definiálni az egyes elvégzendő feladatokat. A vízesés modell hatékonyabb változata az úgynevezett visszacsatolásos vízesés modell, amely megengedi a korábbi lépésekhez való visszalépést. Ezzel a megoldással elkerülhetők az egész fejlesztési életcikluson végigvitt hibák, hiszen lehetőség van azok korrigálására. A vízesésmodell nem alkalmas nagyvonalúan meghatározott, nehezebben átgondolható fejlesztési feladatok megoldásához, hiszen a modell szerint nincs lehetőség a kísérletezésre, és a korábbi hiányosságok állandó pótlására.

5. Követelményspecifikáció

A követelményspecifikáció során meg kell határoznunk az elérendő célt, a szoftver által nyújtott szolgáltatásokat, és a felmerülő korlátokat. A cél természetesen nem lehet más, mint a követelményeknek, a korlátozásoknak és a megszorításoknak való megfelelés. Ebben a fázisban kerül sor a megrendelő és a fejlesztők közötti párbeszédre. A fázis végén meg kell határoznunk a pontos követelményspecifikációt.

Az elérendő cél egyértelműen egy működőképes éttermi szoftver létrehozása. A követelményspecifikációnak tartalmaznia kell a szoftverrel szemben támasztott funkcionális és nem funkcionális követelményeket, melyek meghatározzák annak használhatóságát. A szoftverrel szemben támasztott követelményeket a megrendelő és a fejlesztők közötti párbeszéd alapján határozhatjuk meg. Ha ennél a folyamatnál kellő alapossággal járnak el, akkor az a későbbiekre nézve sok idő –és költségmegtakarítást jelenthet. A fejlesztők nagy feladata ilyenkor, hogy pontosan megértsék a megrendelő igényeit, és megértessék, hogy milyen lehetőségek állnak rendelkezésre a megvalósításához.



A szoftverfejlesztés összetevői

5.1 Áttekintés

A fejlesztési folyamat a követelmények részletes meghatározásával kezdődik meg. A követelmények specifikálásakor a rendszer struktúráját leíró PIM (Platform Independent Model) megtervezését tűzzük ki első célunknak. A fejlesztendő szoftver egy étterem rendelésfelvételét és számlázását megkönnyítő rendszer.

5.1.1 Felhasználói követelmények

Ezek a követelmények a rendszertől elvárt igényeket írják le a felhasználó szemszögéből. A leírás elkészíthető köznyelven, szemléletes diagramokkal, és a megértést segítő táblázatokkal. A szoftver leendő felhasználói – esetünkben a pincérek – legfőképp a rendszer általuk használt felhasználói felületét tudják értelmezni, valamint képesek meghatározni azt az üzleti környezetet, amelyben a szoftver működni fog. A fejlesztők feladata az, hogy a rendszert pontosan modellezzék a felhasználóktól kapott információk alapján.

5.1.2 Funkcionális követelmények

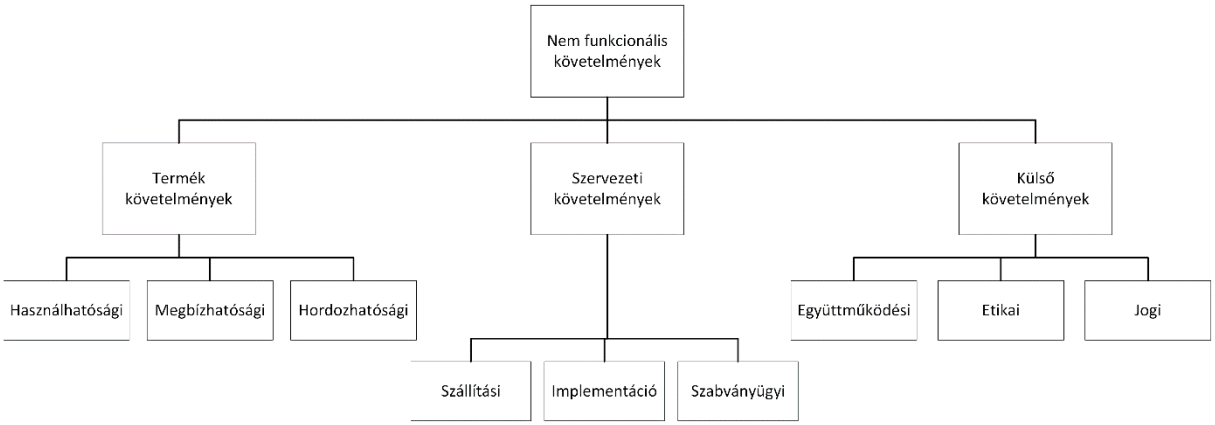
A funkcionális követelmények a rendszer szolgáltatásait írják le a felhasználó szemszögéből. A funkcionális követelmények a rendszer külső hatásokra történő viselkedését is leírják. Ezek a követelmények nagyban függnek a szoftver leendő felhasználóitól, és attól, hogy milyen helyszínen történik majd a rendszer használata. A szoftver funkcióit az UML modellezés

használatakor az úgynevezett use case-ek segítségével modellezzük, ennek folyamatát később részletesen tárgyaljuk. A fejlesztendő szoftver funkcionális követelményei a következők:

- a szoftvernek működőképesnek kell lennie az étterem nyitvatartása alatt
- a rendszer rendelkezzen egy karbantartott adatbázissal
- a szoftver képes legyen több párhuzamos rendelés kezelésére is
- a pincérek képesek legyenek az adatbázisban tárolt ételek felvételére
- a szoftver pontosan összegezze a rendelt tételeket
- a szakácsok egy monitoron figyelhessék a rendeléseket
- az aktuális rendelés zárása után a szoftver készítsen számlát

5.1.3 Nem funkcionális követelmények

A nem funkcionális követelmények körébe a rendszer funkcióira vonatkozó előírások és működési feltételek tartoznak. Ide tartozik például a megbízhatóság és a válaszidő, illetve a fejlesztői környezet meghatározása is. Ezek a feltételek gyakran fontosabbak, mint a funkcionális követelmények, így nagy figyelmet kell rájuk szentelni. Ezek a követelmények három nagy csoportba sorolhatók. A főbb nem funkcionális követelmények a következő ábrán láthatók:



A nem funkcionális követelmények

5.1.3.1 Termék követelmények

A termék követelmények a szoftver viselkedési módját határozzák meg. Ezek a követelménytípusok a későbbiekben részletesen kifejtésre kerülnek a fejlesztendő szoftverre vonatkozóan.

5.1.3.2 Szervezeti követelmények

A szervezet stratégiájából, illetve működési módjából adódó követelmények tartoznak ebbe a csoportba. A fejlesztendő szoftverünk esetében fennáll egy átadási határidő (2014.04.15.).

Ezenkívül adott az átadás helye is, amely ez esetben az étterem pontos címe. A szoftver felhasználói felületének megtervezése a WinDev 15 nevű CASE fejlesztőeszközzel készül el.

5.1.3.3 Külső követelmények

Ide tartoznak a szoftver és annak fejlesztésére ható külső tényezők, más rendszerekkel való együttműködés, illetve a jogi szabályozások.

5.1.4 Használhatóság

Ebben a pontban a nem funkcionális, vagyis az alkalmazás használhatóságára vonatkozó követelményeket kell meghatározni. A betanulási idő minimalizálása érdekében a felület egyszerűségére kell törekedni, valamint célszerű egy részletes felhasználói dokumentációt készíteni. A gördülékeny munkavégzés érdekében fontos, hogy minél átláthatóbb felhasználói felületet tervezzünk. A felület esetében fontos elvárás a konzisztencia, tehát hogy hasonló műveleteket minden ablakon hasonlóan lehessen elvégezni. A felhasználói felület tervezésének folyamatát a későbbiekben részletezzük. A szoftver egy érintőképernyős POS terminálon lesz használható, különleges billentyűkombinációk nem kerülnek beépítésre.

5.1.5 Megbízhatóság

A rendszer megbízhatósága számos tényezőtől függ. Alapvetően elmondható, hogy a rendszer nem működtethető áramforrás nélkül, azonban például szünetmentes tápegységekkel felkészülhetünk az áramkimaradásokra, hogy a rendszert ilyen helyzetekben se kelljen nélkülözni.

5.1.6 Teljesítmény

A szoftver teljesítménye hardver –és szoftverfüggő egyaránt. Fontos az igényeknek megfelelő hardver megválasztása, és a skálázhatóság, tehát hogy a szoftver a későbbiekben bővíthető legyen az igényeknek megfelelően. A rendszer adatbázisának kapacitását is az igényekhez kell igazítanunk.. Mindenképpen fel kell készülnünk tehát, a teljesítmény növelésére, hiszen ha a megnövekedett igények mellett már nem tudjuk a szolgáltatást elfogadhatóan üzemeltetni, az a felhasználók elégedetlenségéhez vezethet.

5.1.7 Támogatottság

A szoftver támogatottságával szemben támasztott követelmények az üzembe helyezést követő karbantartási feladatok elvégzésénél nyújtanak segítséget. A szoftvert elsősorban annak fejlesztői tartják karban, nekik ismerniük kell a program készítése során használt szabványokat, és technikákat. A karbantartás során is szükség lehet a tervezési is kivitelezési munkákban

alkalmazott adatbázis-tervezőkre, grafikusokra, és programozókra. A karbantartók munkáját különböző segédprogramokkal támogathatjuk.

5.2 Az UML nyelv

Mielőtt részletesen tárgyalnánk a use case modell kialakítási folyamatát, szükséges néhány szót említenünk az annak alapjául szolgáló UML szabványról.

Az UML (Unified Modeling Language) egy általános célú modellező nyelv, amely a szoftver-rendszerek alaposabb kidolgozásának folyamatát segíti. A nyelvet Grady Booch, Ivar Jacobson és James Rumbaugh alkották meg 1997-ben. Azóta a nyelv széles körben elterjedt, alapvető tervezési és dokumentációs eszközzé vált. A nyelv vizuális formában teszi lehetővé a szoftver modellezését, amely alapján elvégezhető a szoftver implementálása és tesztelése is. A szabvány legfrissebb verziója a 2.5-ös, amely 2012 októberében jelent meg.

5.3 A use case modell megtervezése

A követelmények elemzése során az egyik legfontosabb folyamat a rendszer működését szemléltető use case modell megtervezése. A korábban meghatározott funkcionális követelményeket egy-egy use case segítségével írjuk le. Az egyes use case-ek végigkísérik az egész rendszerfejlesztési folyamatot, hiszen a use case modellt a későbbiekben felhasználjuk az implementációs és a tesztelési fázisokban is. A use case modell alkalmas a rendszer viselkedésének külső szemszögből történő leírására. A modell három alapelemet tartalmaz:

- **use case-ek:** a szoftver funkciói
- **aktorok:** a rendszer külső szereplői, akik a rendszerrel kapcsolatban állnak
- **kapcsolatok:** az aktorok és a use case-ek közötti viszonyokat szemlélteti

Első lépésben célszerű a rendszerrel kapcsolatban álló aktorokat feltárni, majd hozzájuk rendelni az egyes use case-eket (funkciókat).

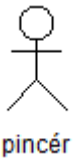
Éttermi szoftverünk modelljének tervezésekor kialakítunk egy általánosabb és egy a rendszerre specifikált use case modellt is. Az általános modellünkben szerepeltetjük az étterem összes dolgozótípusát és az általuk végzett alapvető feladatokat, a speciális modellen pedig kizárólag a szoftverrel kapcsolatban álló akotorkat és use case-eket tűntetjük fel részletezve a szoftver funkcióit.

5.3.1 Az aktorok

A szoftver megrendelőjével és leendő felhasználóival folytatott beszélgetések alapján meghatározhatók a rendszer érdekeltjei. Ezek az érdekeltek lesznek a use case modellünkben az aktorok. Az aktor tulajdonképpen egy szerep, amit az adott szereplő végrehajt a rendszerrel való kapcsolata során. Az aktorok a rendszerrel feladatokat hajtanak végre, illetve igénybe veszik annak szolgáltatásait. Az aktrok lehetnek személyek, gépek, és külső rendszerek is. Fontos sajátosság, hogy a rendszerben nem lehet két azonos szerepkört betöltő aktor. Az aktoroknak két fő típusa van:

- **kezdeményező**: a funkciók végrehajtásának kezdeményezői
- **résztvevő**: a funkció végrehajtásában csak részt vesz

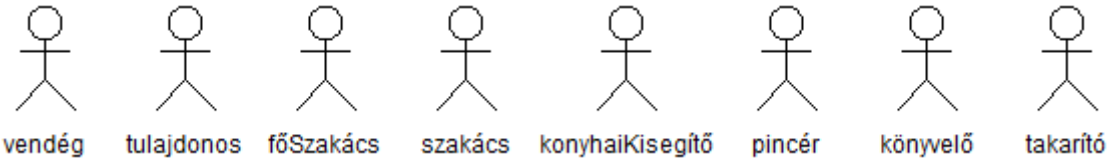
Az aktor szimbóluma egy pálcikaember.



Az aktor UML jelölése

5.3.1.1 Az aktorok megtalálása

Az aktorokat legegyszerűbben úgy találhatjuk meg, hogy a felhasználói igények dokumentációjából kikeressük a főneveket. Ebben a folyamatban célszerű arra koncentrálni, hogy ki, mire használja a rendszert, illetve a rendszerünk kommunikál-e más rendszerekkel. Az étterem általános – az étterem működését bemutató – modelljének aktorai a következők:



A szoftverre koncentráló use case modellünkben, csak a rendszerrel valamilyen szinten kapcsolatban álló aktorokat tűntetjük fel, tehát például nem ábrázoljuk a takarító által végzett feladatokat, hiszen a takarító semmilyen módon nem érintkezik a szoftver-rendszerrel.

5.3.2 A use case-ek

A use case-ek a rendszertől elvárt funkciókat, illetve szolgáltatásokat írják le. Egy use case azt határozza meg, hogy a felhasználó mit kíván végrehajtani a szoftverrel, ugyanakkor a megvalósítás mikéntjére nem tér ki. A követelményelemzés során meghatározott use case-eket

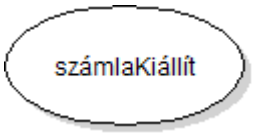
úgynevezett fekete doboz use case-eknek nevezzük, ami azt jelenti, hogy a cél csak a rendszer viselkedésének leírása külső szemszögből. A use case-eket mindig a rendszer szereplői kezdeményezik. A szoftver-rendszerre vonatkozóan a use case-eknek három típusát különböztethetjük meg, ezek a következők:

- **architekturálisan fontos use case-ek:** a rendszer fő szolgáltatásai
- **egyéb use case-ek:** kevésbé fontosak a rendszer működését tekintve
- **rendszeridegen use case-ek:** korábban meghatározott, de a végső modellből elhagyott use case-ek

5.3.2.1 A use case-ek megalálása

A use case-ek feltárásának alapvető módja a megrendelővel folytatott beszélgetések, illetve interjúk. Ezenkívül hasznos lehet az úgynevezett brainstorming (ötletbörze) módszer alkalmazása is, amely különösen ajánlott összetett problémák megoldására. A use case-ek feltárhatók az úgynevezett favágó módszerrel is, melynek során a követelmények dokumentumából egyszerűen kiválogatjuk az igéket.

A use case szimbóluma az ellipszis, amely alá, vagy amelybe a use case nevét írjuk. A név azonosítja a use case rendszerben végrehajtott funkcióját.



A use case UML szimbóluma

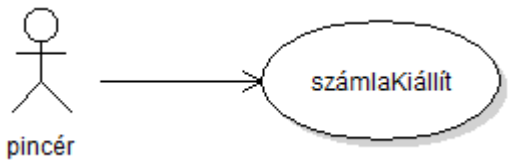
5.3.3 Forgatókönyvek

Minden egyes use case-hez készítenünk kell egy részletes leírást, amelyben a felhasználó szemszögéből írjuk le a rendszerrel való kommunikáció lépéseit. A végrehajtásnak lehetnek alternatív és különleges esetei is, ezeket szintén rögzítenünk kell. A forgatókönyv, más néven szcenárió a use case egy konkrét instanciája, amely a use case lépésenkénti lefutását írja le. Egy use case-hez több forgatókönyv is készülhet, hiszen annak több alternatív működése is létezhet. A szcenáriók készítésére szigorú szabályok nem léteznek, egyszerűen szöveges formában megfogalmazzuk azok tartalmát. A forgatókönyv tartalma vázlatosan a következő:

- feladat értelmezése
- alternatív útvonalak
- a kapcsolódó szereplők meghatározása

5.3.4 A kapcsolatok

A kapcsolat alapértelmezésben az aktorok és a use case-ek között értelmezett kezdeményező viszonyt jelenti, amelyet irányított vonallal jelölünk. Az aktorok és a use case-ek közötti kapcsolatok modellezésére a use case diagramot használjuk. Egy use case megvalósításában több aktor is részt vehet, a résztvevőket irányítás nélküli vonallal jelöljük.



A kezdeményező kapcsolat jelölése

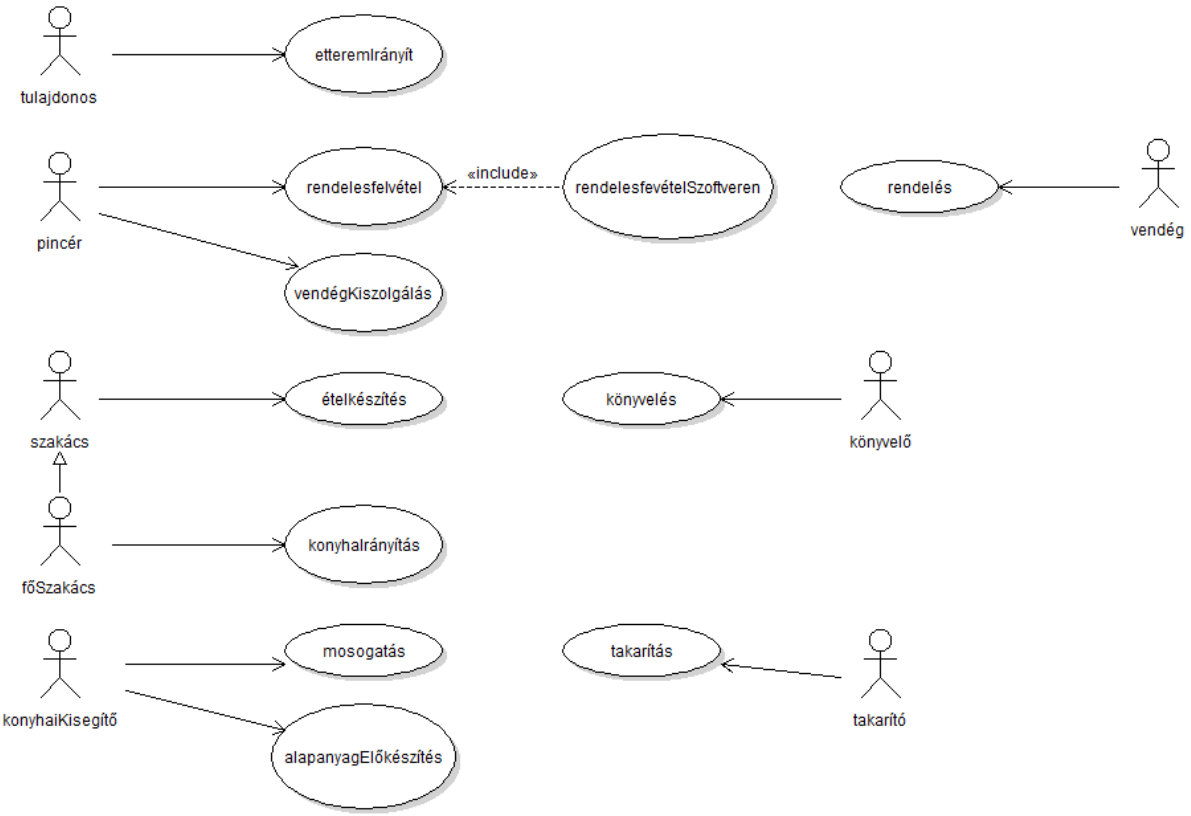
Ezen kívül léteznek a diagramban léteznek még speciális kapcsolatok, ezek a következők:

- **öröklődési kapcsolat:** Két aktor közötti kapcsolat, ha egy use case megvalósításakor több szereplő is betölti ugyanazt a szerepet. A leszármazott minden olyan use case-zel kapcsolatban áll, amelyikkel az őse. Öröklődési kapcsolat értelmezhető use case-ek között is.
- **tartalmazás:** Olyan use case-ek, melyek egy adott use case végrahajtása során minden alkalommal azonos módon hajtódnak végre. Az alap és a tartalmazott use case-t szaggatott vonal köti össze, a nyíl az alaptól a tartalmazott felé mutat. A kapcsolatot az <<***include***>> sztereotípiával jelöljük.
- **kiterjesztés:** Olyan use case-ek esetén, amikor a vezérlés bizonyos feltételek hatására egy másik use case-nek adodik át, tehát az alap use case bővített változata fut le. Az alap és a kiterjesztett use case-t szaggatott vonal köti össze, a nyíl az alap felé mutat. Ezt a kapcsolatot az <<***extend***>> sztereotípiával jelöljük.

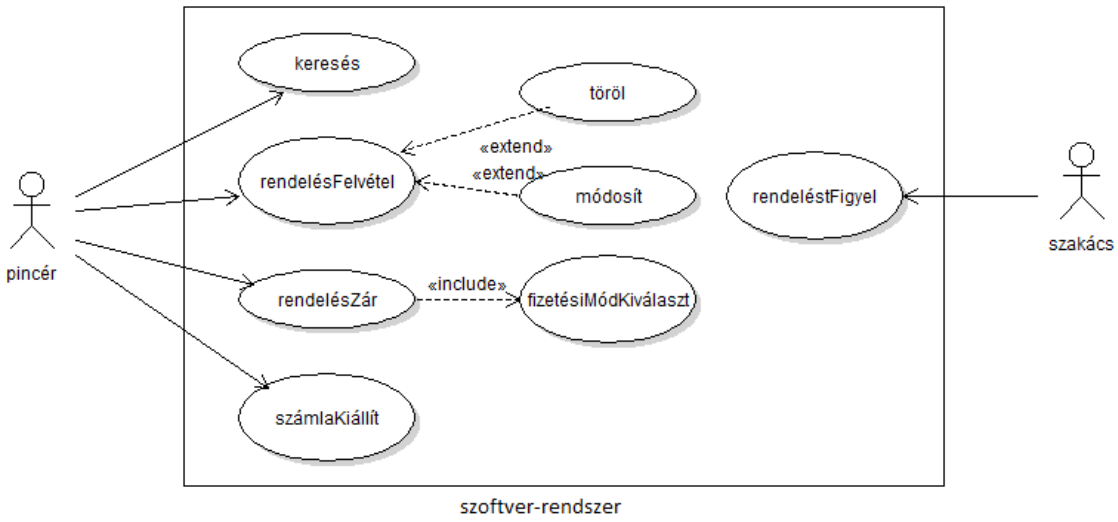
5.3.5 A fejlesztendő szoftver use case modelljei

Ahogy a fentiekben már leírásra került, fejlesztendő szoftverünkhöz két use case-t készítettünk, egy általánosat az étterem működéséről, valamint egy speciálisat, amely csak a szoftver-rendszerre és annak funkcióira koncentrál.

Az általános éttermi modell a következő ábrán látható:



A fejlesztendő szoftverre specializált use case modell a következő képen látható:



5.3.6 Forgatókönyvek a működés leírására

Mivel mi a pincérek munkáját elősegítő szoftvert szeretnénk létrehozni, ezért a fejlesztendő szoftver szempontjából a többi munkatárs illetve munkaterület, amely nem kapcsolódik közvetlen a felszolgálók munkájához, esetünkben hanyagolható. Így a fejlesztendő szoftverre specializált use case modell forgatókönyve kerül rögzítésre az alábbiakban.

AKTOROK:

- pincér: a szoftvert elsődlegesen használó személy, aki elvégzi a rendelésekkel kapcsolatod teendőket, illetve kiszolgálja a vendégeket
- szakács: az a személy, aki a leadott rendelések alapján elkészíti az ételeket

USE CASE-EK:

- keresés: a pincér keres a lehetséges ételek és italok között
    - Prekondíció (előfeltétel): legyenek a rendszerben/ az étterem étlapján ételek és italok
    - Postkondíció (következmény): a pincér könnyedén hozzá tudja adni a tételt a listához
    - Szokásos működés: a pincér kikeresi az ételek/ italok közül azt, amelyiket a vendég szeretne
    - Alternatív esetek: -
    - Kivételes esetek: nem működik a terminál
- rendelésFelvétel: rendelés felvétele
    - Prekondíció (előfeltétel): a vendég kiválasztotta a számára megfelelő ételt/italt
    - Postkondíció(következmény): a rendelés rögzítésre kerül, továbbítódik a szakácshoz elkészítésre
    - Szokásos működés: rendelés rögzítése, továbbítsa
    - Alternatív esetek: -
    - Kivételes esetek: nem működik a terminál
- rendelésZár: a meglévő rendelések zárása, összegzése
    - Prekondíció (előfeltétel): vannak rendelések

    - Postkondíció (következmény): a rendelés lezáródik, adott számlához újabb tétel nem adható hozzá
    - Szokásos működés: a pincér lezárja a rendelést, mikor a vendég távozni készül, hogy kiállíthassa a számlát
    - Alternatív esetek: -
    - Kivételes esetek: nem működik a terminál
- számlaKiállít:
    - Prekondíció (előfeltétel): legyen lezárt rendelés
    - Postkondíció (következmény): a vendég megkapja a számlát és a terminál készen áll újabb műveletek végrehatására
    - Szokásos működés: rendelés lezárása után a számlát a vendég rendelkezésére bocsátják
    - Alternatív esetek: csatlakozási hiba, kifogy a papír
    - Kivételes esetek: nem működik a nyomtató
- töröl: nem kívánt tétel törlése a listáról
    - Prekondíció (előfeltétel): legyen rögzített tétel
    - Postkondíció(következmény): adott tétel törlésre kerül a rendeléseket tartalmazó listából
    - Szokásos működés: törlésre került a pincér által tévesen felvitt tétel a listáról
    - Alternatív esetek: -
    - Kivételes esetek: nem működik a terminál
- módosít: felvitt tételek módosítása
    - Prekondíció (előfeltétel): legyen felvitt tétel/rendelés a rendszerben
    - Postkondíció (következmény): az igény szerint megváltozott rendelés az igényeknek megfelelően módosul a listában
    - Szokásos működés: a pincér a tévesen bevitt tételt átírja/módosítja
    - Alternatív esetek: -
    - Kivételes esetek: nem működik a terminál
- fieztésiMódKiválaszt: fizetési mód választásának lehetősége

- Prekondíció (előfeltétel): a pincér a rendelés lezárásával felméri a fizetési módot
- Postkondíció (következmény): a pincér tudja, hogy kézpénzzel vagy kártyával fog a vendég fizetni
- Szokásos működés: a pincér tájékozódik arról, hogy a vendég milyen módon kíván fizetni
- Alternatív esetek: nem működik a kártya leolvasó, nem működik a kézpénzes kassza
- Kivételes esetek: nem működik a terminál

- rendeléstFigyel: a szakács figyelemmel követi a pincér által rögzített rendeléseket
  - Prekondíció (előfeltétel): legyen a pincér által rögzített rendelés
  - Postkondíció (következmény): a szakács a meglévő rendelés alapján elkészíti a vendég által kívánt tételt
  - Szokásos működés: a szakács egy monitor segítségével nyomon követi a pincér által felvett rendelést és elkészíti azt
  - Alternatív esetek: nem működik a szakács számára beüzemelt monitor
  - Kivételes esetek: nem működik a terminál

6. Az objektummodell

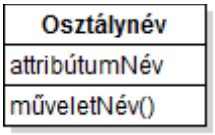
Az üzleti modellezés során készül el az úgynevezett osztálymodell, amely fontos szerepet játszik a modellezett szervezet és a feltárt probléma megértésében. Az osztálymodellezés során feltárt objektumokat, és a köztük lévő kapcsolatokat az üzleti objektum-modellben (Business Object Model) ábrázoljuk.

6.1 Objektum, osztály

Az objektumokkal a valós világ elemeit modellezzük, például személyeket, fizikai tárgyakat, berendezéseket. Az objektumoknak vannak tulajdonságaik, valamint jellemezhetők állapotukkal és viselkedésükkel. Az objektumok állapota más objektumok hatására változhat meg. Az aktuális állapotot az objektum éppen felvett tulajdonságértékei és kapcsolatai határozzák meg. Az objektum viselkedése az általa végrehajtott feladatsorozat, amit más objektumok hatására végez.

6.1.1 Az osztály

Az objektumokat a modellben az osztály elemmel ábrázoljuk. Az osztályok azonos attribútumokkal, metódusokkal és kapcsolatokkal rendelkező objektumok csoportját jelentik. Az osztály konkrét példánya (instanciája) az objektum. Az UML-ben az osztály egy három részre osztott téglalappal modellezzük. Az osztály nevét a felső részben adjuk meg, az attribútumokat a középsőben, az osztály műveleteit pedig az alsó részben adjuk meg.



Az osztály UML szimbóluma

Az osztály nevének megválasztásakor figyelnünk kell, hogy az egyedien jellemezze az adott osztályt. Az osztály nevét nagybetűvel kezdjük, ha több szóból áll, célszerű az egyes szavak kezdőbetűit nagybetűvel írni.

6.1.1.1 Az attribútumok

Az objektumok tulajdonságai az úgynevezett attribútumok. Az attribútum megadásának formája a következő:

[láthatóság] név [: típus][=kezdeti érték] [{jelleg}]

A láthatóság megadja, hogy az adott objektum tulajdonságai mennyire „láthatók” a külvilag számára. Az UML nyelv három láthatósági szintet definiál, ezek a következők:

- \+ public: minden osztály által elérhető
- \- private: csak saját objektumon belül látható
- # protected: csak saját objektumból és az utódokból látszik

6.1.1.2 A műveletek

A művelet olyan tevékenység, amelyet az osztály végrehajt, megvalósítása a metódus. Egy osztály minden objektuma azonos metódusokkal rendelkezik. A műveletek megadásának általános formája a következő:

[láthatóság] név [(paraméter lista)] [: visszatérési érték típusa] [{jelleg}]

6.1.1.3 Az asszociáció

Az osztályok együttműködnek egymással, ezért kapcsolatot kell létesíteniük egymással. Az asszociáció az osztályok közti kapcsolat absztrakciója. Az osztályok közti kapcsolatot az őket

összekötő vonal jelöli. Az asszociációhoz név rendelhető, melyet a vonal fölé írunk. Megadható az osztályok kapcsolatának foka is, amely azt jelenti, hogy az egyik osztály elemei közül a másik osztályból mennyi tartozhat. A kapcsolat bejárhatóságának irányát nyilakkal jelölhetjük, amely akár a vonal mindkét végén lehet.

6.1.1.4 Az öröklődés

Az öröklődés lehetővé teszi, hogy egy modellelem sajátjaként kezelje a nála általánosabb modellelem attribútumait és műveleteit. Az utód osztály örökli az általánosabb ős tulajdonságait. Az öröklődés jele egy üres háromszöggel végződő nyíl, amelynek csúcsa az ősosztálynál van.

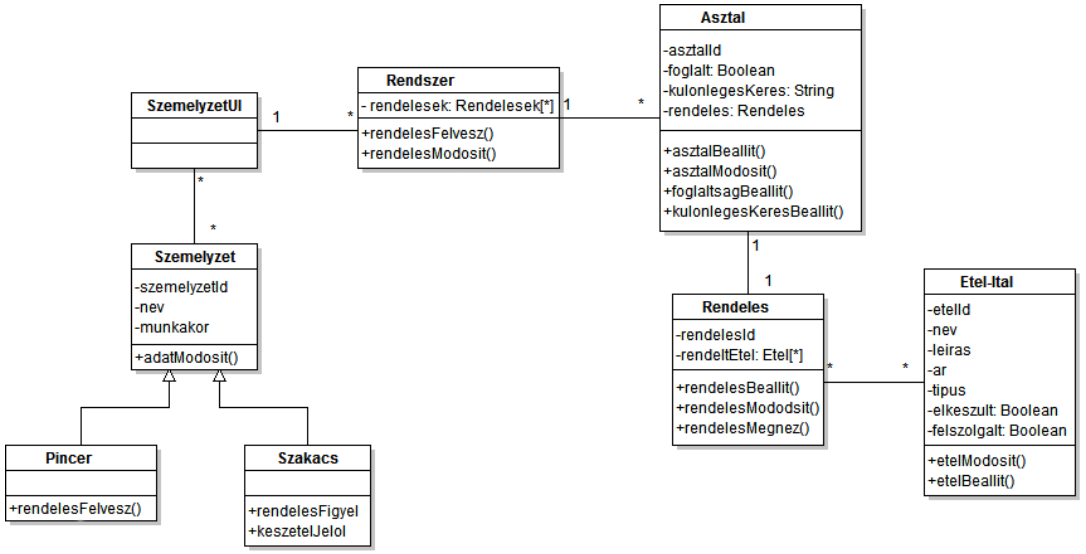
6.1.1.5 Aggregáció és kompozíció

Az aggregáció tulajdonképpen egy gyenge rész-egész viszony. Az egész osztály és a részosztályok elkülönülnek egymástól. A kapcsolat akkor aggregációs, ha a részosztály az egész nélkül nem értelmes.

A kompozíció az aggregáció erősebb változata. A tároló objektum fizikálisan tartalmazza a részobjektumot, tehát együtt keletkeznek és szűnnek meg.

Az UML az aggregációt egy üres, a kompozíciót egy tömör rombuszban végződő vonallal jelöli. A rombuszok az egész osztály felől vannak.

6.2 Az osztálydiagram



Az éttermi szoftver osztálydiagramja

6.3 A CRC kártyák

A CRC kártyák (Class-Responsbility-Collaboration Card) egy rendszer szereplőinek felelősségeit írják le. Az kártyák objektumorientált rendszerfejlesztés esetében a korábban definiált osztályok felelősségeit tartalmazzák. A CRC kártya azt határozza meg, hogy az egyes feladatok elvégzéséért melyik osztály felel. A kártya fejlécében az adott osztály szerepel, az alatta lévő két oszlopban pedig az egyes felelősségek és a hozzájuk tartozó felelősök.

Példaként az éttermi rendszerünk „Rendszer” osztályának CRC kártyáját a következő ábrán láthatjuk:

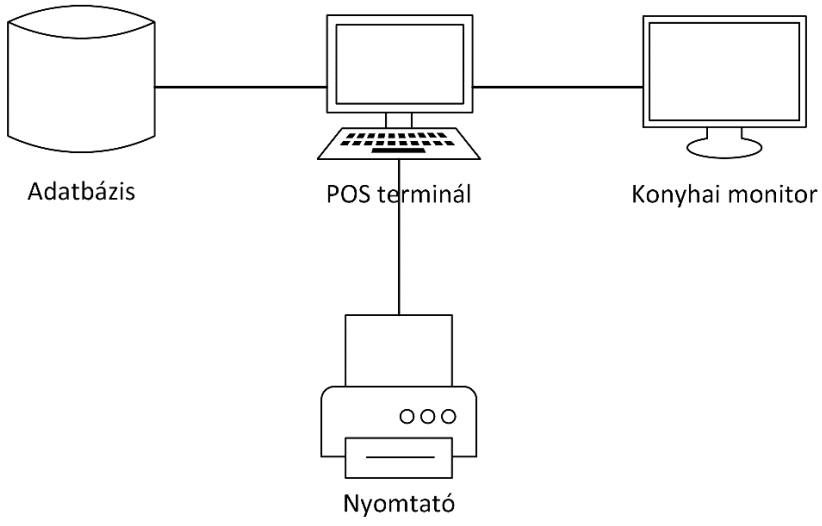
Rendszer	
Rendelések rögzítése	SzemelyetUI
Rendelések módosítása	SzemelyzetUI

A Rendszer osztály CRC kártyája

7. A fizikai leképezés feladatai

A fizikai leképezés feladata a fogalmi modell számítógéprendszerre történő leképezése, így a platformfüggő modell kialakítása. Ebben a fázisban meghatározzuk a szükséges számítógépes környezetet, amellyel a feladatok hatékonyan végezhetők. Ez a leképezési folyamat a számítógép konfiguráció tervét, az adatbázistervet és a programspecifikációt tartalmazza.

7.1 A számítógéprendszer terve



A RESTO éttermi rendszer hardver-achitektúrája

A rendszer hardverkövetelményeit a megoldási alternatívák kiértékelésekor már leírtuk.

7.2 Adatbázis-tervezés

Az informatikai rendszerek adattárolási és feldolgozási folyamataiban kiemelkedő szerepet játszanak az adatbázisok, ezért különös hangsúlyt kell helyezni az adatbázis megtervezésére. Az adatbázisokkal szemben támasztott alapvető követelmények a következők:

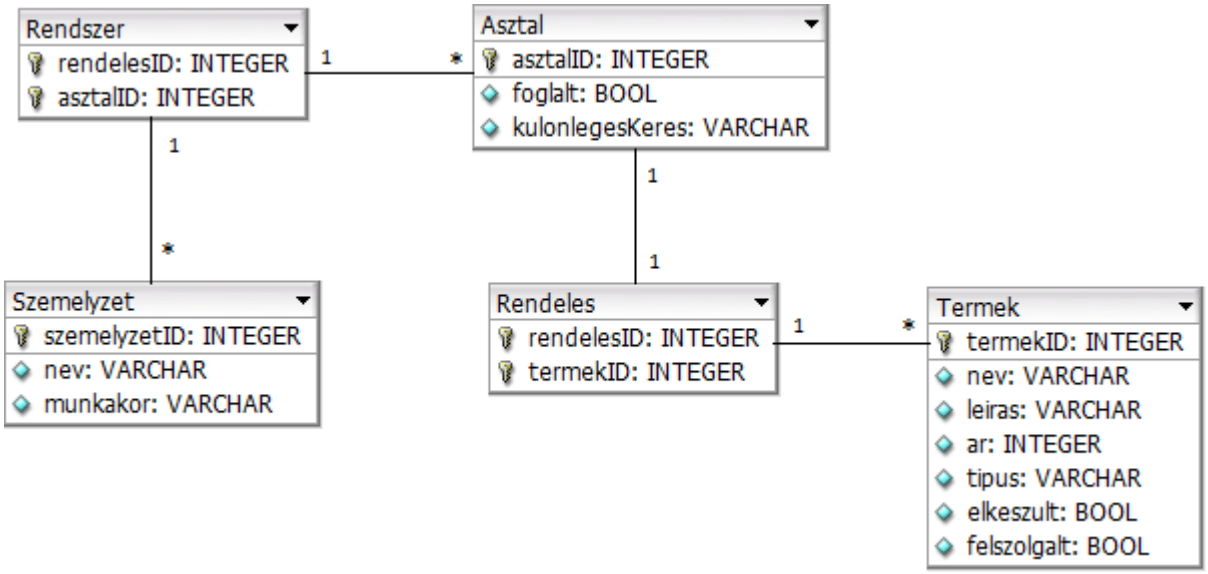
- **logikai adatfüggetlenség:** az adatmodell megváltozása nem befolyásolja a felhasználói programok módosítását
- **fizikai adatfüggetlenség:** az adatok fizikai helyének megváltozása nem befolyásolja a felhasználói programokat

Az számítógépen lévő adatok kezelését, tárolását, és feldolgozását egy erre a célra kialakított adatbáziskezelő-szoftver végzi.

7.2.1 Az adatbázis-tervezés feladatai

1. Adatszerkezetek specifikálása
2. Tárolókapacitás-igény számítása
3. Az elhelyezési modell rögzítése
4. Az adatintegritás megvalósítási tervének kidolgozása
5. Az adatbiztonság megtervezése

Az éttermi szoftverünk adatbázisterve a következő:



Az éttermi szoftver adatbázisterve

A RESTO szoftver adatbázisa a következő adatokat tartalmazza majd:

- a személyzet adatai
- az étterem asztalait
- az aktuális rendeléseket
- az étteremben kapható összes terméket (ételek, italok, stb.)

7.3 A felhasználói felület (GUI) terve

A WinDev által biztosított programot használjuk a felületek elkészítésére. Szoftverünk egyszerű, letisztult, amit a képernyő érintésével tudunk navigálni. Minden egyes képernyő tervén megjelenik a RESTO étterem logója.

Első felületünk a bejelentkező ablak. Itt a pincér felhasználónevével és jelszavával a bejelentkezés gombra kattintva belép és azonosítja magát.

A következő felület, ahol feltűnik a „Kérjük válasszon egy asztalt” felirat. A képernyőn megjelenik mind a 19 asztal. A pincér kiválasztja azt az asztalt, amelyik rendelését éppen felveszi. A kiválasztás után a bal alsó sarokban található „Vissza” gombbal továbblép az utána levő felületre, ahol a fő menüpontok mellett megjelenik a számlázás is. A képernyő jobb oldalán kiválaszthatja a fő menüpontok közül a számára megfelelőt. A menüpontok a következők: Italok, Főételek, Saláták, Levesek, Desszertek.

Például a „Levesek” menüpontra kattintva előugrik a következő ablak, ahol a leves fajták találhatók. Ezek közül az érintőképernyőn könnyen kiválaszthatja a kívánt ételt. A jobb alsó sarokban található vissza gombbal visszaérkezik az előző felülethez, ahol a kiválasztott termék megjelenik a számlázási táblázatban a hozzá tartozó árral együtt. A bal felső sarokban megjelenik az eddig kiválasztott termékek összege forintban értendően. A jobb felső sarokban a rendelés felvételével és a számlázás végezetével a piros betűs „Kijelentkezés” gombra kattintva a pincér kijelentkezik, ami után kinyomtatja a rendszer a számlát.

8. Tesztelés

8.1 Definíciók:

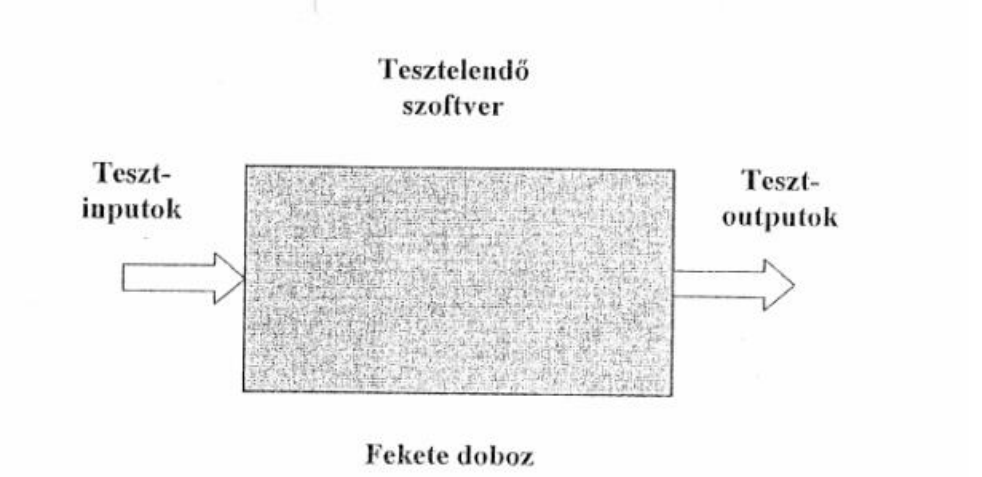
Bemeneti adat: Olyan számítógépes adat, amely egy tetszőleges szoftver működését vonja maga után, felhasználói szinten.

Kimeneti adat: Olyan számítógépes adat, amely egy tetszőleges szoftver működése során jelenik meg, a működtetés eredményeként, felhasznált szinten.

Tesztelés: A szoftver bemeneti adatokkal való sorozatos ellátása és kimeneti válaszadatok megfigyelése.

8.2 Funkcióteszt:

Funkcionális tesztelésről beszélünk akkor, amikor a programot egyedül csak a külső viselkedésében, az előírt funkciónak teljesítésben vizsgáljuk azáltal, hogy a bemeneti adatokra kapott kimeneti válaszokat figyeljük meg, a futtatható tárgykód felhasználásával. Ebben a megközelítésben egyáltalán nem vesszük figyelembe a forráskódot, míg a program megírási nyelvét sem, vagyis a programot olyan egységnek tekintjük, amibe nem „látunk bele”. Emiatt szokás ezt a megoldást fekete doboz, vagy vasdoboz tesztelésnek is nevezni (angolban black-box testing, iron-box testing).

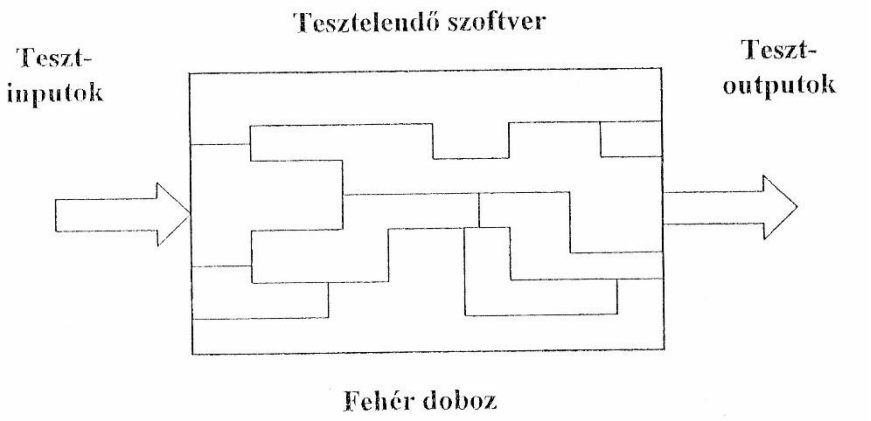


Fekete doboz

A funkció teszt során minden egyes funkciót külön-külön megvizsgálunk, hogy működésük megfelelő-e. Működésük úgy valósul-e meg, hogy az adott funkciónak eleget tegyen. Ezek a funkciók az alábbiak: rendelés felvétel. törlés, módosítás, rendelés zárás, fizetési mód kiválasztása, számla kiállítása, rendelés figyelése.

8.3 Strukturális tesztelés:

~ről akkor beszélünk, amikor a szoftver forráskódjának felhasználásával a belső struktúra, a belső működés végigkövetésére irányul az ellenőrzés. Ebben a megközelítésben a tesztelési adatok megtervezése során az a cél, hogy a forráskód utasításainak és elágazásainak minél alaposabb végigjárását tudjuk elérni. Itt a programot olyan egységnek tekintjük amibe „belelátunk”. Emiatt szokás fehér doboz, vagy üveg doboz tesztelésnek nevezni.



Fehér doboz

A szoftver leprogramozásának végeztével strukturális tesztelés alá vetjük az elkészült programot. Ekkor megvizsgáljuk, hogy a programozás során akadt-e olyan probléma, amely miatt a program nem az elvártaknak megfelelően működik-e.

8.4 Terheléses tesztelés:

~ről akkor beszélünk, amikor az egész elkészült programot egy adott időben meghatározott számú szakember teszteli. Így megvizsgálva a szoftver teherbírását.

Az elkészült szoftvert egy előre kijelölt időpontban legalább 6 szakember (mivel az étteremben 4 pincér dolgozik jelenleg) fogja egyszerre használni, így megvizsgálva a program teherbírását. A tesztelés során kiderül a program reakcióidejének gyorsasága, és teherbírása. Amennyiben a vizsgálat során 6 szakemberrel gördülékenyem működik a program, akkor szinte biztos, hogy élesben 4 pincérrel is megfelelően fog működni.

9. Telepítés és üzemeltetés

9.1 Telepítés:

Az elkészült és letesztelt szoftver telepítése teljesen automatizált, továbbá a telepítéssel járó feladatok elvégzése a szoftvert fejlesztő munkatársak feladata közé tartozik.

A telepítés USB-meghajtóról történik. Mivel ez egy érintő kijelzős terminál a telepítés során sem szükséges billentyűzet illetve egér hozzácsatlakoztatása a munkaállomáshoz. Miután a terminál érzékelte az UBS-eszközt, előugrik az „Automatikus lejátszás” ablak mely alatt ki kell választanunk a „Mappa megnyitása a fájlok megtekintéséhez” lehetőséget. Ezt követően két fájl jelenik meg az ablakban: RESTOsw.exe és a kod.txt. A telepítés elindításához „kattintsunk” (érintsük meg) a RESTOsw.exe fájlra, majd amikor elindul kérni fogja az ellenőrző kódot. A kod.txt fájl megnyitása után másoljuk ki a benne található kódot az erre alkalmas ablakba. Az „Oké” gomb lenyomása után elindul a telepítés, a folyamat automatizált, ezt követően nincs vele teendő, csak ki kell várni, hogy a szoftver telepítse magát. Amikor befejezte a telepítést, szükség van a rendszer újraindítására. Ezt követően a rendszer már üzemkész.

9.2 Üzemeltetés:

Az üzemeltetés folyamatában a mindenkori cél, hogy rendszerünk mindenkor helyesen működve álljon a felhasználók rendelkezésére, az üzemeltetőnek kötelessége a felhasználó érdekeit figyelemben tartva az észlelt hibák mielőbbi javítása.

- A rendszer helyesen működik, ha:
- azonosítás nélkül a rendszer nem használható
- az azonosítást követően a pincérek használhatják a szoftver szolgáltatásait: rendelés felvétel, módosítás, törlés, számla zárása
- karbantartott adatbázissal rendelkezik

Az üzemeltetési feladatok elvégzésére a **rendszergazdát** (esetünkben Pálfy János) jelöltük ki, aki köteles a hiba fennállásának bejelentésétől számított 3 órán belül javítani a rendszerben fellépő hibát.

A rendszergazdának kiemelten ügyelni kell a következőkre:

- A rendszernek elérhetőnek kell lennie a nap 24 órájában.
- Az azonosítási funkció működőképességére
- Adatbázis karbantartására

FOKOZOTT FIGYELMET KELL FORDÍTANI:

- **Biztonságosságra**: A rendszer helyesen vagy hibásan működik, de nem veszélyeztet emberi életet, nem okoz anyagi vagy környezeti kárt és nem befolyásolja károsan más rendszerek működését.
- **Rendelkezésre állásra**: A rendszer helyesen működik egy t időpontban, vagyis rendelkezésre áll. Jelenleg az atw ingyenesen hozzáférhető tárhelyét használjuk. A weboldal rendelkezésre állása az atw szolgáltatótól függ.
- **Karbantarthatóságra**: A meghibásodott rendszer újra működőképessé tehető t időtartam alatt.

10. A fejlesztés implementálása WinDev-ben

A szoftver felületeinek háttere fehér, betűszíne fekete színű. A betűtípus AR JULIAN, míg a címeket kivéve (amik tizennyolcas nagyságúak) a betűméret tízes nagyságú.

10.1 A bejelentkező felület:

Kiválasztottunk két sávot, amit egymás alá helyeztünk el. A felső sáv „Felhasználónév” címet kapta, míg az alsó a „Jelszó” címet. A sávok alá helyeztünk egy gombot, mely a „Bejelentkezés” címet kapta. A „Felhasználónév” sávjában a pincér felhasználónevét, a „Jelszó” sávjában pedig a jelszavát kéri a rendszer. A „Bejelentkezés” gombra kattintva belép a program.

A jobb felső sarokba a kép beillesztésével került éttermünk lógója. A jobb alsó sarokba pedig a dátumot helyeztük el. Ha a pincér véletlen hibás felhasználó nevet vagy jelszót ír be, felugrik egy ablak a következő felirattal: „Hibás felhasználónevet vagy jelszót adott meg. Próbálkozzon újra!” Ezt új ablak létrehozásával, szöveg begépelésével hoztuk létre.

10.2 Az asztalok kiválasztásának felülete:

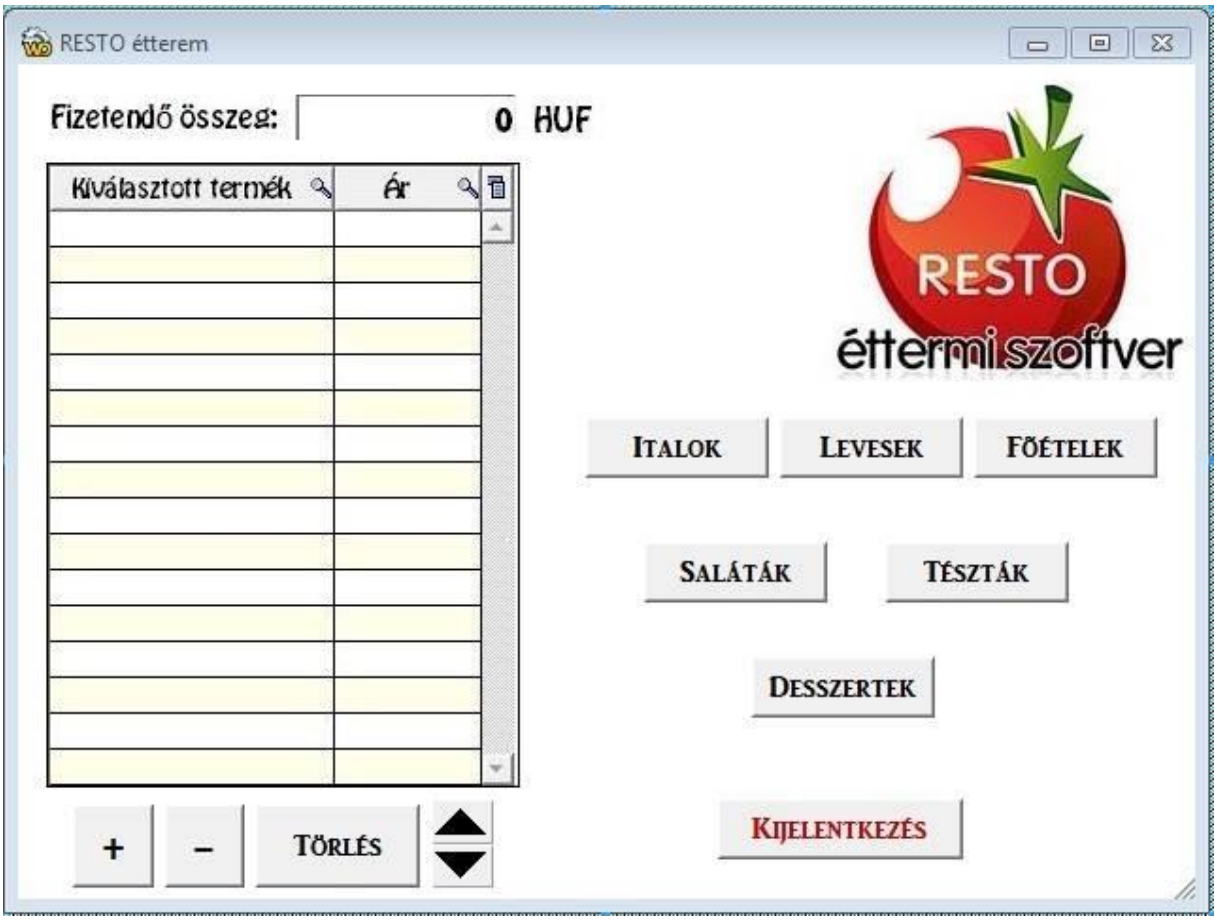
Középen megjelenik a „Kérjük válasszon egy asztalt” felirat, mely szöveg begépelésével jött létre. Alatta tizenkilenc egyforma gombot helyeztünk el esztétikusan a képernyőn. Ezeken a gombokon az asztalok számozása látható. Erre rákattintva kiválaszthatjuk a kívánt asztalt. A bal felső sarokba beillesztettük az étterem lógóját. A bal alsó sarokban levő „Vissza” gombbal pedig továbblépünk a következő felületre.


RESTO étterem
RESTO
éttermi szoftver
HIBÁS FELHASZNÁLÓNEVET VAGY JELSZÓT ADOTT MEG!
PRÓBÁLJA MEG ÚJRA!


RESTO étterem
RESTO
éttermi szoftver
Kérjük válasszon egy asztalt!
1. Asztal
2. Asztal
3. Asztal
4. Asztal
5. Asztal
6. Asztal
7. Asztal
8. Asztal
9. Asztal
10. Asztal
11. Asztal
12. Asztal
13. Asztal
14. Asztal
15. Asztal
16. Asztal
VISSZA
17. Asztal
18. Asztal
19. Asztal

10.3 A fő menüpontok kiválasztásának és a számlázásnak a felülete:

Jobb oldalon találhatók a fő menüpontok, amelyeket gombokkal jelenítettünk meg. Ezekre rálépve megjelennek az adott menüpontok étlapjai.



Példánkban a „Levesek” étlapját láthatjuk. A fő menüpontok közül kiválasztjuk a „Levesek” gombot, rákattintva előugrik egy ablak, melyen megjelenik a kívánt felsorolás. Középen látható a „Levesek” cím. Alatta találhatók a gombokkal megjelenített ételek. Ezekre kattintva felkerülnek az említett számlázási listánkra. A jobb alsó sarokban látható „Vissza” gombbal visszaléphetünk (ha már befejeztük az adott étlapon a kívánt ételek felvételét) a fő menüpontok és a számlázási felületre.

Az előző felületen (a fő menüpontok kiválasztásának és a számlázásnak a felületén) a jobb felső sarokban látható éttermünk lógója. A baloldalon egy 2 hasábos számlázási táblázatot helyeztünk el. Az első hasábban a kiválasztott termékek jelennek meg, míg a második hasábban az adott termékek árai. Alattuk a „+” és „-” gombokkal a termékek darabszámait növelhetjük. A fel és le nyilakkal lépkedhetünk a kiválasztott ételek között. A törlés gombbal pedig a nem kívánt ételeket kitörölhetjük. Ezeket is gombok elhelyezésével oldottuk meg. A végén megjelenik a bal felső sarokban a végösszeg forintban értendően. Ezt egy WinDev-ben elérhető opcióval tettük láthatóvá. A fő menüpontok alatt piros színű „Kijelentkezés” gombot helyeztünk el, melyre rákattintva a program kijelentkezik és kinyomtatja a számlát.

Irodalomjegyzék

[1] Dr. Sziray József: Bevezetés a szoftver-technológiába, Logsoft, Budapest, 2009.

[2] Dr. Sziray József, Kovács Katalin: Az UML nyelv használata, Logsoft, Budapest, 2009.

[3] Raffai Mária: Információrendszerek fejlesztése és menedzselése, Novadat Kiadó, Győr, 2003.

[4] Raffai Mária: IRT5 Információrendszer-tervezés, Alexander Alapítvány, Győr, 2007.