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

**SZAKDOLGOZAT**

|  |  |  |
| --- | --- | --- |
| **OE-NIK**  **2014** | Hallgató neve:  Hallgató törzskönyvi száma: | **Varga Viktor**  **T-002321/FI12904/N** |

Óbudai Egyetem

Neumann János Informatikai Kar

Alkalmazott Informatikai Intézet

SZAKDOLGOZAT

FELADATLAP

Hallgató neve: **Varga Viktor**

Törzskönyvi száma: T-002321/FI12904/N

A dolgozat címe:

**Vendéglátás hatékonyságának növelése mobil alkalmazással**

**Support of Catering by Mobile Application**

Intézményi konzulens: Dr. Kutor László

Külső konzulens:

Beadási határidő: 2015. január 7.

A záróvizsga tárgyai: Számítógép architektúrák

Mobil informatika

**A feladat**

Tervezzen és valósítson meg egy olyan mobil alkalmazást ami Android platformú mobil készülékkel segíti a vendéglátás hatékonyságának növelését!

A rendszert kliens-szerver architektúrában valósítsa meg.

A vendéglátó személyzet (a dolgozók és a tulajdonos) munkáját több szolgáltatással segítse. Kezelje az asztalokhoz rendelt rendelések felvételét és fizetésnél a számlaadást.

Tartsa nyilván az étel és ital készleteket és kezelje a bevételezést.

A kliens oldal asztali és mobil változatban is készüljön el. Különös figyelmet fordítson az egyszerű kezelhetőségre!

**A dolgozatnak tartalmaznia kell**:

* a feladat pontos leírását,
* a meglévő, hasonló célú rendszerek elemzését,
* a megvalósítandó rendszer részletese specifikációját,
* a feladat megoldásához alkalmazható módszerek és technológiák bemutatását,
* a választott megoldások indoklását,
* a tervezés és megvalósítás részletes leírását,
* a tesztelés szempontjainak tervét és eredményeit,
* a továbbfejlesztési lehetőségek vizsgálatát.

Ph.

……....……………….

Dr. Galántai Aurél

mb. intézetigazgató

A szakdolgozat elévülésének határideje: **2017. január 7.**

(OE TVSz 32.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

|  |  |
| --- | --- |
| ……………….. | ..……………………. |
| külső konzulens | intézményi konzulens |



**Neumann János Informatikai Kar**

**ÓBUDAI EGYETEM**

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 20... ..........................................

|  |  |
| --- | --- |
|  | hallgató aláírása |

[1. BEVEZETÉS 6](#_Toc402115010)

[1.1. Feladat ismertetése 6](#_Toc402115011)

[1.2. A rendszer részei 6](#_Toc402115012)

[1.3. Követelmények 6](#_Toc402115013)

[1.4. A probléma fontossága, felvezetése 7](#_Toc402115014)

[1.5. Fejlesztés során használt eszközök 7](#_Toc402115015)

[2. IRODALOMKUTATÁS 8](#_Toc402115016)

[2.1. Hasonló alkalmazások 8](#_Toc402115017)

[2.1.1. Multi Store [2] 8](#_Toc402115018)

[2.1.2. eSystem [4] 10](#_Toc402115019)

[2.1.3. Összegzés 14](#_Toc402115020)

[2.2. Környezetkutatás 14](#_Toc402115021)

[2.2.1. Operációs rendszer 14](#_Toc402115022)

[2.2.2. Terheltség 15](#_Toc402115023)

[2.2.3. Adminisztratív adatok tárolása 16](#_Toc402115024)

[2.3. Felhasználható technológiák 16](#_Toc402115025)

[2.3.1. Szerver oldal 16](#_Toc402115026)

[2.3.1.1. Apache Tomcat [6] és Glassfish [7] 16](#_Toc402115027)

[2.3.1.2. Internet Information Services [8] [9] 17](#_Toc402115028)

[2.3.1.3. Microsoft ASP.NET Web API 17](#_Toc402115029)

[2.3.1.4. Microsoft Entity Framework [18] 18](#_Toc402115030)

[2.3.2. Kliens oldal 19](#_Toc402115031)

[2.3.2.1. Microsoft Windows Presentation Foundation [19] 19](#_Toc402115032)

[2.3.2.2. HTML5 és SPA 20](#_Toc402115033)

[2.3.2.3. Google AngularJS [24] [25] 21](#_Toc402115034)

[2.3.3. Összegzés 21](#_Toc402115035)

[3. IRODALOMJEGYZÉK 22](#_Toc402115036)

1. BEVEZETÉS
   1. Feladat ismertetése

A vendéglátásban számos olyan esemény van, amely a számítástechnika segítségével leegyszerűsíthető, felgyorsítható. Egy éttermet alapul véve az asztalfoglalások kezelése, a rendelések felvétele és kiszolgálása, a fizetés lebonyolítása is számottevően gyorsítható és pontosítható, ha felhasználjuk az informatikát, és szoftveresítjük a feladatokat. Mindez nem csak a fentebb felsorolt eseményeknél nyújt segítséget, mert az ezzel kapcsolatos és ezekre épülő egyéb tevékenységeket – mint például a számottevő adminisztrációs feladatokat – is elősegíti.

A szakdolgozat célja egy több programból álló szoftver rendszer létrehozása, amely segítséget nyújt e feladatokban.

* 1. A rendszer részei

A rendszer három elméleti részre osztható fel:

* Vendéglátói rész / asztali kliens

Feladata a napi nyitás, napi zárás, rendelések felvitele, módosítása, törlése, kifizetése, a blokknyomtató kezelése.

Ez a rész általában a pultnál található terminál(ok)on fut.

* Menedzselés

Az alkalmazás segítségével a főnökök számára adminisztrációs és kimutatási funkciók nyújtása, mint például készletmozgások, forgalom, alapanyagfogyás, bevételezés, stb. kezelése.

* Androidos kliens

A rendelésekkel kapcsolatos funkciók ellátása – hasonlóan a vendéglátói részhez – csak mobileszközön.

* 1. Követelmények

A szoftvernek tudnia kell szinte az összes olyan folyamatot, amivel le lehet fedni egy étterem, bisztró, kocsma, stb. feladatait. Ilyen feladatok közé sorolhatók az asztalok-, az asztalfoglalások-, a rendelések-, a fizetések- és a raktárkészletek-, a termékek illetve a hozzájuk kapcsolódó receptek-kezelése, a különböző kimutatások, napi zárások, bevételezések, és ezekhez kapcsolódóan még számottevő apró feladat.

Ezen feladatokból mindre kell biztosítani asztali alkalmazást, és jelentős részükre mobil alkalmazást is, mint például a rendelések felvételére, és a fizetésekre.

Mivel ezek a szoftverek – mind az asztali és a mobil is – adott célközönségnek készülnek, ezért nem kell figyelmet fordítani a multiplatformitásra, hiszen azon helyek, ahol jelenleg is található számítógép, kizárólag Windows operációs rendszert használnak, még a pultban lévő terminálok esetén is. A mobileszközök terén kifejezetten az Androidos eszközöket célzom meg, mivel ezek lefedik a mobil piac 80%-át [1], és jelentősen olcsóbban beszerezhetőek, mint más operációs rendszerrel rendelkező társaik.

* 1. A probléma fontossága, felvezetése

Azért választottam ezt a projektet, mivel nem sok ilyen rendszer van, és azok is számos szükséges funkciót nem tartalmaznak vagy nem megfelelő formában. Például csak a mobil eszköz használatának lehetőségét nézve sok ilyen szoftvernek nincs is mobileszközös felülete, vagy ha van, akkor csak Windows Mobile platformra készül, amelyhez megvásároltatják a mobileszközt is csillagászati áron. Hiányosság, hogy nincsenek megfelelően megtervezve a felületek, ezáltal nem kifejezetten kényelmes a használatuk. Természetesen vannak jól megtervezettek is, de ezek ár/érték arányhoz viszonyítva elérhetetlen áron kerülnek forgalmazásra. Ezen okokból, sok helyen még mindig a „jól bevált” papíros, fél-számítógépesített megoldást választják inkább.

* 1. Fejlesztés során használt eszközök

A fejlesztéshez használt eszközöket jelentősen befolyásolja a két platform: a Windows és az Android. Így az ezekhez a legjobban használható programokat részesítem előnyben. A Windows-os alkalmazás elkészítéséhez a Microsoft Visual Studio 2013 programot, az Android-os alkalmazáshoz pedig az Eclipse vagy az Android Studio fejlesztőkörnyezetet szeretném használni.

A témakör meghatároz egy eszközt, amelyet fel kell használnom a fejlesztések folyamán, mégpedig egy blokknyomtatót, amellyel a nyugtákat lehet elkészíteni.

1. IRODALOMKUTATÁS

Az irodalomkutatás célja a hasonló témakörrel foglalkozó szoftverek felkutatása, és elemzése, annak vizsgálata, hogy miben különböznek a saját magam által tervezettől, van-e értelme belekezdeni a saját implementációmba, vagy fel tudom-e használni valamely más nyílt rendszerek részeit.

E fejezetben írom le a megvizsgált rendszereket, és keresek bennük olyan kiemelkedő pontokat, amelyek alátámasztják, hogy érdemes belevágni ebbe a fejlesztésbe.

* 1. Hasonló alkalmazások

Meglehetősen nehéz piackutatást végezni ebben a témában. Az összes ilyen szoftver fizetős, ezáltal nehezen lehet hozzájutni a szoftverekhez, illetve azok demo verziójához. Szakirodalom sem áll olyan szinten rendelkezésre, mint sok más program esetében, így saját használat közbeni tapasztalatokat, vendéglátásban dolgozók segítségét kérve, illetve a nehezen megszerezhető demo verziók és/vagy videók elemzése alapján végeztem el a piackutatást.

Két cég szoftverét vizsgáltam meg és állítom szembe az általam tervezett programmal. A mobilkliensük vizsgálata során kiderült, hogy az asztali kliensek is rendelkeznek hiányosságokkal, így azokat is megvizsgáltam.

* + 1. Multi Store [2]

**Mobil kliens:**

A többi vendéglátói rendszerekhez hasonlóan, ez is PDA-ra van kifejlesztve, amely egy kihalóban lévő technológia. Emiatt kifejezetten drágán árusítják a mobil klienshez szükséges eszközöket, ezáltal a vásárló egy szinte semmi másra nem használható hardverhez jut, amely a mai legolcsóbb mobiltelefonok teljesítményéhez sem ér fel. Emellett ezek a PDA-k még rezisztív kijelzővel rendelkeznek, amelyek technológiájukból fakadóan kifejezetten pontatlanok kézzel való érintéskor, így csak valamilyen eszköz, például a hozzá tartozó stylus (ceruza) segítségével lehet egy megfelelő pontosságot elérni. Így az esetek többségében a dolgozóknak szükséges egy tollat hordozniuk magukkal, amelyet minden egyes alkalommal elő kell venniük, és eltenniük. A másik problémája ennek a Multi Store-os mobilkliensnek, hogy a felhasználó azonosítás csak a szoftver indításakor szükséges, így ha a felhasználók tranzakcióit be akarjuk azonosítani, akkor minden felhasználó váltásnál újra kell, hogy indítsuk a szoftvert – amely a PDA teljesítményéből adódóan is – egy hosszadalmas folyamat.

Ezzel szemben én egy Android alapú mobil klienst tervezek készíteni. Ezáltal a megrendelőnek nem is feltétlen kell eszközt vásárolnia, hiszen bármilyen Androidos eszközre fel lehet tenni a szoftvert, így jelentős költségeket takarítva meg. Azonban ha mégis vásárlásra kerül a sor, akkor sem jelentős kiadás, ráadásul bármilyen más funkcióra felhasználható az eszköz. Lehetséges így táblagépek használata is, amely sokkal nagyobb kijelző felületet biztosít a pincéreknek. Az androidos szoftvert – ugyanúgy, mint az asztalit – több felhasználósra tervezem, így az egyszerű és gyors felhasználóváltás a legfontosabb szempont itt is.

**Asztali kliens:**

A Multi Store egy alapvetően érintőképernyős, asztali számítógépekre kidolgozott szoftver, amelyet a Kamex nevű cég értékesít Magyarországon. Külsőségeit tekintve egyszerű ikonokat használnak, amik esetenként alig felismerhetőek, ezért könnyen bele tud bonyolódni a felhasználó. A rendelésben szereplő tételeknek megtervezett rész minimális méretű, ami nagyban megnehezíti az érintő képernyőn történő használatát, és problémássá teheti az oda-vissza keresést a tételeknél.



. ábra A Multi Store kezdőképernyője [3]

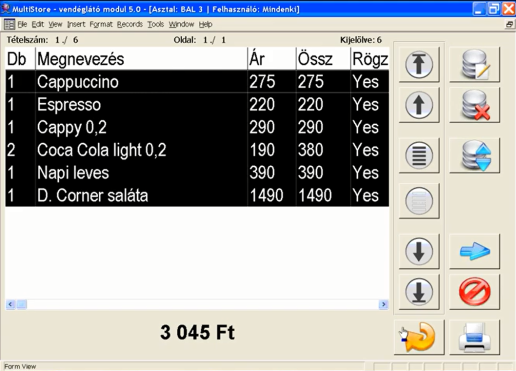
A programnak egy fő képernyője van, ami nem jeleníti meg elkülönítve a vendéglátó egység egyes területeit – min például terasz, galéria - minden ömlesztve szerepel a programban, nincs mód különálló egységeket létrehozni. Ezáltal egy nagyobb vendéglátó egység esetében igen zsúfolttá válik a kezdőképernyőnk, ami az adott vendéglátó egységben dolgozó személyzet munkáját akadályozni, lassítani tudja.

Esetemben ezzel ellentétben, lehetőséget biztosítok különálló helyiségek létrehozására, ezen belül elrendezésre, ezáltal az adott részen, csak az ott ténylegesen elhelyezkedő asztalokat és székeket látjuk, nem pedig a vendéglátóegység összes férőhelyét ömlesztve.

A Multi Store program az asztalokra és székekre felvett rendeléseket nem jeleníti meg vizuálisan, ami azt jelenti, hogy az asztal, ahol már van felvett rendelés, ugyanúgy jelenik meg a képernyőn, mint az, ahol még nincs. Tehát, ha a pultos rápillant a fő képernyőre, nem látja egyértelműen mennyi a foglalt és szabad asztalok száma.

Ezt a problémát az asztalok színének, esetleg megjelenítési képének változtatásával tervezem kiküszöbölni. Ennek célja, hogy minél jobban elkülöníthetőek legyenek, akár csak egy odapillantással a foglalt és a szabad férőhelyek és láthatóvá váljon, hogy mely asztalok azok, amelyeknél a fizettetés zajlik. Ez a változtatás nagyban hozzájárulhat a hatékonyabb munkavégzéshez, az új vendégek fogadásához.

Az adminfelület, - ahol a vendéglátó egység lelkét ténylegesen irányítjuk, ellenőrizzük - ebben a szoftverben is a fő képernyőre nyílik. Ez akadályozhatja a munkafolyamatok gyorsaságát, mivel vissza kell lépni, esetleg kilépni az adminfelületből, ha rendelést szeretnénk felvenni. Ezzel a félkész munka elveszhet, a rendelést pedig késéssel lehet leadni a konyha, illetve a pult felé.

A végső lépésnél, a fizetésnél sok gombot látunk, szebbnél szebb piktogrammokal ellátva, amik valóban tetszetősek, mindössze annyi velük a probléma, hogy ha a felhasználó nem jártas a szoftverben, akkor nem fogja tudni azonnal kezelni, mivel a gombok funkcióját a program használatba vétele előtt meg kell tanulnia, mivel nem magától érhetőek.

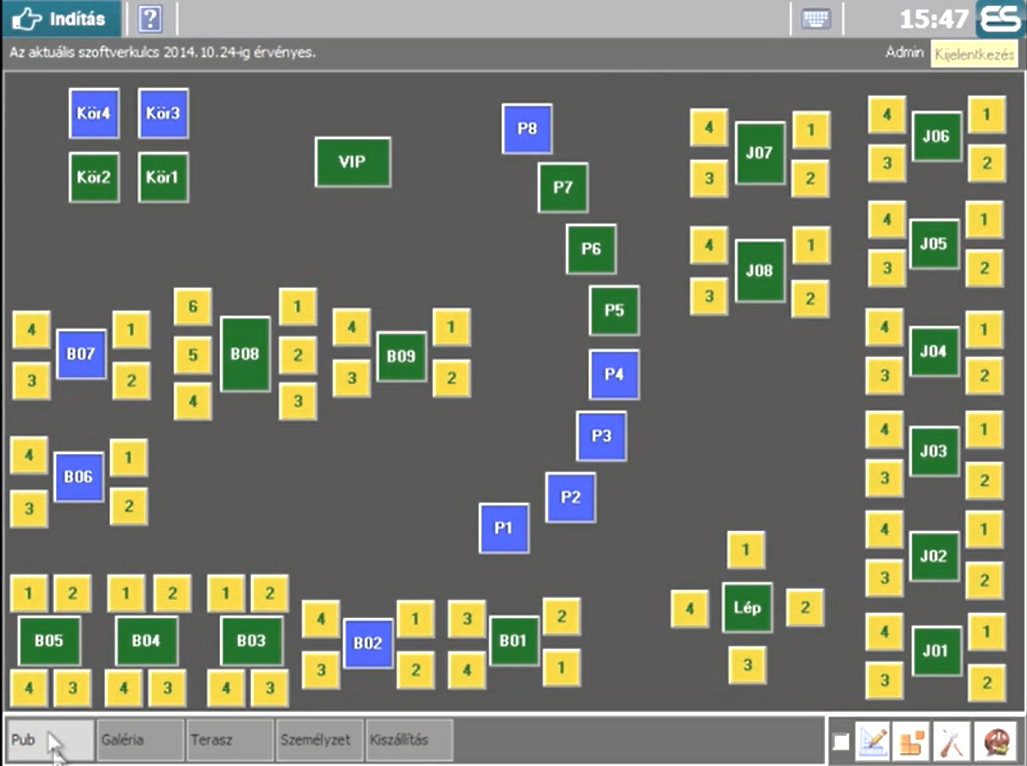
2. ábra Nem egyértelmű gombok [3]

Ezzel szemben egy olyan letisztult kezelőfelületet tervezek létrehozni, ami a szoftver minden részében, így fizetésnél is egyértelmű gombokat alkalmaz, hogy könnyen kezelhető legyen és a szoftvert nem ismerők számára is azonnal alkalmazhatóvá, érthetővé váljon.

* + 1. eSystem [4]

Az eSystem, egy 2001 óta a piacon lévő szoftver. Mobilalkalmazása a weboldalán többször meg van említve, viszont hosszadalmas kutatás után sem találtam semmi olyan információt, amely alapján el tudtam volna kezdeni az elemzését.

Így az asztali klienst kezdtem el vizsgálni, amely kifejezetten jól működik, minden funkcióval rendelkezik, ami egy vendéglátó egység akadálymentes működéséhez elengedhetetlen, ezért szeretnék egy ehhez hasonló, és ha lehet még jobb, letisztultabb, modernebb programot létrehozni.



. ábra E-System kezdőképernyő [5]

Az asztalok megjelenítése itt is igen egyszerű; kisebb nagyobb kockákkal történik (asztal, szék). Itt már az asztalokra felvezetett rendelések és egyes munkafolyamati fázisok vizuálisan is meg vannak jelenítve. A kis és nagy kockák színei változnak, ami áttekinthető és értelemszerű, viszont a program grafikai háttere igen szegényes.

Ez a program – hasonlóan az enyémhez – már elkülönít különálló helyiségeket (pl. terasz, galéria), ami az előbb bemutatott Multi Storenál nem volt meg.

**Rendelésfelvétel, felhasználók**

A rendelés felvétele szépen meg van jelenítve, a tételek kilistázva egyik oldalon, az üres mező, ahová hozzáadjuk a rendelést a másik oldalon, a termékek azonban nincsenek kategóriákba rendezve, hanem ömlesztve szerepelnek. Kereső mező van ugyan, de így is nehézkes és hosszadalmas lehet az egyes tételek megtalálása.

A program ugyanolyan munkafolyamatokkal dolgozik, mint az összes többi a rendelésfelvétel-, és a háttérműveletek terén is (árubeszerzés, raktárkezelés).

Különféle felhasználói fiókokat különít el különböző jogosultságokkal. Tartalmaz olyan fiókokat, amelyek használatával a dolgozók csak az alap munkafolyamatokat tudják végrehajtani, szerkeszthetik a rendeléseket. Megkülönböztet még teljes, illetve rész jogokkal felruházható főnöki és üzletvezetői fiókokat. Egy hiányossága van a felhasználó fiókkezelésekkel kapcsolatban, hogy nincs semmilyen időkorlát, ami az éppen bejelentkezett fiókot egy idő után kilépteti. A ki-be jelentkezés pedig munka közben igen időigényes és hajlamos lehet emiatt átsiklani rajta a dolgozó. Ez több problémát is felvet, mivel a bejelentkezve maradt fiók azonosítójával bonyolódik le minden művelet és így nem lehet nyomon követni az esetleges hibák, hiányok okát illetve felelősét. Ez is lényeges probléma, de ha egy teljes jogkörű fiók marad bejelentkezve, az visszaélésre ad lehetőséget a dolgozók számára, mivel így számukra is elérhetővé válik az adminfelület.

**Adminfelület**

Az adminfelület ebben a programban már komolyabban fel van építve, mint a Multi Storeban, de itt is a fő ablakra nyílik rá, és ha rendelést kell felvenni, akkor ugyanúgy be kell zárni az admin felületet, hogy elérhetővé váljon a normál kezelő felület a dolgozó számára. Rengeteg opciót találunk itt, viszont egyes munkafolyamatok során akár 3-4 felugró ablakon keresztül vezet az út az elérni kívánt opcióhoz.

Ugyanitt érhetők el a kimutatások és a statisztikák is, amelyek nagyon fontosak a vezetőség számára, mivel ezek segítségével tudják megállapítani, hogy az étterem arculatán, étlapján mi az, ami változtatásra szorul. Ezek a kimutatások egyesével nézhetők át, nem tudunk több tételt egyszerre átnézni az egymásra nyíló ablakok miatt.

Az étlapon szereplő ételek receptúráit is be tudjuk állítani specifikusan az adott ételre, italra, viszont itt is 3-4 felugró ablakon keresztül tudunk hozzáférni magához a receptúrához.

**Raktárak kezelése**

A raktárak kezelése az egyik legfontosabb, hanem a legfontosabb feladata egy ilyen szoftvernek. A pontosan dokumentált raktárkészlet rengeteg időt takarít meg a dolgozóknak. Ez ebben a programban kicsit túl van bonyolítva. El lehet különíteni több raktárt is az adott helységen belül, ami abban az esetben célszerű, ha van több terminálunk, és mindegyik a hozzá rendelt raktárkészletből vonja a fogyást. Ez a standolásnál komoly eltéréseket eredményezhet.

Abban az esetben például, ha rossz helyre bevételezik az árut, vagy jó helyre bevételezték ugyan, de közben a raktárból áthelyezték a pult készletébe, a mozzanat gépen történő feltüntetése nélkül, – ami könnyedén előfordulhat időhiány miatt egy forgalmas étteremben – akkor máris többlet keletkezik a pult raktárkészletében, a raktáréban pedig hiány. Ez ismételten visszaélésre adhat lehetőséget, mivel a gyakori eseteknél – és a rendszer raktárkezelési hiányosságaiból adódóan gyakori esetekről beszélhetünk – nem lesz alaposan kivizsgálva a hiány oka. Ebből adódóan az adminisztrációban keletkezett hiányokból előbb-utóbb valós, nem csak álhiányok keletkeznek.

**Saját tervek**

Ezekre az alapvető hiányosságokra a következő megoldásokat szeretném alkalmazni:

kinézetnél a kis kockákat felváltanám ízlésesen megtervezett asztal, illetve szék ikonokkal.

A munkafolyamatokat sem csak színekkel szeretném jelezni, hanem ezeknek az ikonoknak egyértelmű, jól látható változtatásával.

Ezeken túl még egy lényeges változtatást tervezek eszközölni az eddigi programokhoz képest, amelyekben nincs mód az étterem stílusához illő megjelenítési módokat beállítani. A program ugyanúgy néz ki egy kocsmában, és egy elegáns étteremben. Ennek kiküszöbölésére szeretnék egy olyan megoldást alkalmazni, hogy a program által megjelenített színeket illetve asztal és szék ikonokat be lehessen egyedileg, a vendéglátó egység arculatával megegyezőre állítani, kicsit interaktívabbá téve így az amúgy egyhangú programokat. Amellett, hogy az imént említett program színekkel való jelzésének átláthatóságát megőrizném, ezekkel, a program egészét tekintve apró változtatásokkal szeretném emelni a megjelenítés színvonalát.

Mindezek mellé egy éjszakai üzemmódot is tervezek a programba – mind az asztali, mind a mobil programok esetén, – ezáltal változó fényviszonyok mellett is jól olvasható felületeket kapok. Ez hasznos lehet, hiszen nappal jobb a világos felületen sötét betűket, sötétben pedig sötét háttéren világos betűket olvasni.

A rendelés felvételénél a tételeket fő- és alkategóriákba rendezve, áttekinthetőbbé tenném, így – mivel kategórián belül átlagosan 20-30 tétel szerepel – azt már könnyebb átlátni, mint ömlesztve 200 tételt. Itt is meglenne a keresés lehetősége, ezáltal az akár percekig tartó keresés másodpercekre redukálódik, ami összességében nagyon sokat számit egy vendéglátó egység mindennapjaiban.

A fiókkezeléssel kapcsolatos problémát, az említett időkorlátos kiléptetés bevezetésével valósítanám meg, ami esetében az időkorlát beállítás jogát meghagynám a programot használó vendéglátó egység személyzetének, igaz egy alapérték is lesz, amit sorozatos tesztek alapján állapítok majd meg.

A felhasználói fiókoknál nem statikus jogkörökkel rendelkező felhasználókat tervezek létrehozni – mint a legtöbb esetben a három fő csoport: főnöki, aki minden joggal bír a programon belül; egy üzletvezetői, aki tud bevételezni; és egy dolgozói – hanem olyan lehetőséget biztosítani, amely segítségével minden felhasználó egyedi jogkörrel rendelkezhet. Ez lehetővé teszi, hogy míg az egyik dolgozó csak az alap munkafolyamatokat érheti el, addig a másik akár bevételezni is tudjon a vezetőség távollétében, anélkül hogy bármilyen más adminisztratívabb jellegű funkciót elérne.

Az admin felületet teljesen más elképzelés alapján, de ugyanazon az elven készíteném el, mivel lényegében ugyanazt kell, hogy tudja, mint az összes ilyen témával foglalkozó program.

Még egy fontos változtatás, hogy nem az alap szoftver fő képernyőjére nyílik rá az admin felület, hanem egy különálló programként. Ezt a felületet is minél letisztultabbra és minél egyszerűbben használhatóra tervezem, de minden olyan funkcióval felruházom, amivel az elődjei bírnak. Itt is igyekszem plusz dolgokkal bővíteni: az egyik ilyen, hogy egy adott statisztikához nem 3-4 felugró ablakon keresztül jutunk el, mint eddig, hanem a letisztultabb menünek köszönhetően egyből a felületen ki tudjuk választani, amire kíváncsiak vagyunk. Emellett még egy interaktív grafikonnal is ellátni, ahova tetszőleges tételeket lehet ráhúzni, így a kiválasztott tételek egymás közt egyszerűen összehasonlíthatóak grafikusan, ránézésre pillanatok alatt. Ezeket a rendszer automatikusan különböző színekkel jeleníti meg, így a grafikonra ráhúzott tételek egyértelműen megkülönböztethetőek. A grafikon X tengelyén jelenne meg az időtartomány, amelyet a legutóbb ráhúzott termék előzetesen beállított időtartománya szabna meg. Ennek a grafikonnak a segítségével egyszerre több tétel fogyását vagy beszerzésének gyakoriságát is vizsgálni tudjuk.

A raktárak kezelésénél a több raktárkészlet helyett, egy fő raktárkészletet hoznék létre. Viszont ezt a raktárkészletet virtuálisan felosztanám az egyes terminálokra, így ha például van három helyünk: pult, bár, önkiszolgáló rész, akkor ezekre külön-külön – meg persze egyben is – lehet lekérdezni a termékek fogyását. Így az esetleges adminisztrációs és tényleges hibák, hiányok sikeresen elkerülhetőek, mivel nem kell a raktárkészletek között átviteleket generálni, de mégis lekérdezhető hogy melyik helyen milyen fogyások történtek.

* + 1. Összegzés

A fenti kutatások kimutatták, hogy sem megfelelő asztali kliens, sem megfelelő mobilkliens nem áll rendelkezésre. Úgy látszik, hogy van lehetőség – és igény is – egy sokkal jobb ilyen rendszer létrehozására, amely segítségével a vendéglátói ipari szoftvereknek nem csak a számát gyarapítanám, hanem egy magasabb színvonalra is emelném mind funkcionalitásban, mind megjelenésben.

* 1. Környezetkutatás

A megfelelő technológia kiválasztásához elengedhetetlen ismernünk azt a környezetet is, amelyen a programunkat használni fogják. Számos olyan tényező van, amely befolyásolhatja a döntésünket. Esetemben – például egy éttermet megvizsgálva - az alábbi kérdések merülnek fel:

* ha van már számítógép az üzletben, akkor azon milyen operációs rendszer fut,
* egy nagy forgalmú étterem esetén, mennyire lehet leterhelve a rendszer,
* hol tároljuk az adminisztratív adatokat?
  + 1. Operációs rendszer

Eddigi vizsgálódásaim és tapasztalataim alapján azokban az üzletekben, ahol már van valamilyen szintű számítástechnika bevezetve az üzletmenetbe – legyen az akár csak a főnök irodájában lévő számítógép – azokon szinte kizárólag csak Microsoft Windows operációs rendszer fut. Úgy tűnik – mint a hétköznapi életben is –, hogy a szabad felhasználású (például Unix) rendszerek háttérbe szorulnak. Ezen okokból az asztali programomat elegendő ezen egy operációs rendszerre kiterjeszteni, ezért bőven elég, ha a Microsoft valamilyen programnyelvét használom – szemben más multiplatformos nyelvekkel, mint például a Java. Így számos olyan funkciót is elérhetek, amelyek egy multiplatformos fejlesztés esetén nem egyszerűen, vagy egyáltalán nem érhetőek el. A fejlesztés is sokkal gyorsabb és egyszerűbb lesz, mivel a Microsoft kifejezetten jó támogatást ad a szoftvereihez, és a fejlesztéshez is.

A mobil kliens pedig szintén megfelel, ha csak Android operációs rendszert futtató eszközökön fut, mivel ez a jelenleg piacon lévő mobilkészülékek jelentős részét lefedi és ezek a készülékek egyre olcsóbban szerezhetőek be. Igaz itt is, hogy egy adott platform nyelvén készítve egy szoftvert, jóval több funkcionalitást tudunk beletenni – mobileszközön ilyen szembetűnő funkció például a gesztusok, szenzorok, stb. kezelése.

Összegezve: az asztali kliensnek valamilyen Microsoft specifikus nyelvet érdemes választani, a mobil kliensnek pedig az Android nyelvét, amely a Java. Érdemes megjegyezni, hogy az Androidos Java nem ugyanaz, mint az Oracle Java. Mindkettő valóban a Java programnyelv nyelvszerkezeteit használja, de egy Androidra készített alkalmazás nem fut Oracle-s futtatókörnyezetben – és fordítva is igaz.

* + 1. Terheltség

Jelentősen sokat számít a szerver típusának kiválasztásában az adott szerverre nehezedő teher:

* a kérések/kiszolgálások száma (darab/perc),
* mennyisége - milyen jellegű adatmennyiség mozgás van (KiloBájt/másodperc),
* kapcsolat típusa - alkalmi kapcsolatok vagy egyszer kiépített és fenntartott.

Esetemben egy nagyobb étterem esetén, még mindig csak maximum 5-15 mobilkliens, és legfeljebb 10 asztali kliens lehet állományban. Ez a kapcsolatok számát tekintve nem jelent nagy gondot.

Az adatmennyiség sem számottevő tényező, főleg amennyiben egy-két trükköt bevezetünk. Alapvetően szöveges adatok lennének forgalomban, és abból is jelentősen kevés, hiszen például egy 10 fős asztal rendelésekor is, csak pár tíz darab azonosítót és értéket kell elküldeni a szerverhez. Az elérhető menük, termékek listáját pedig elegendő csak egyszer letölteni a kliensekre, utána már csak karban kell tartani.

Ezekből az következik, hogy az adatok mennyisége eléggé csekély, így ez a komponens sem követel meg semmilyen pluszt a szerver részéről.

A kapcsolat típusának esetünkben elegendő lehet alkalmi kapcsolatokat létesíteni, mivel nem szükséges a folyamatos kapcsolatfenntartás. Hiszen ezek a kommunikációk a szerver és kliens között nem olyanok, mint mondjuk egy online videó közvetítés, ahol folyamatos kapcsolat kell. Ezzel a választással jelentős erőforrásokat szabadítunk fel mind a szerveren, mind a klienseken.

* + 1. Adminisztratív adatok tárolása

Egy önálló kis éttermet vizsgálva, bőven megfelelhet nekik, ha az adataikat ott az üzletükben tároljuk. Egy étteremlánc esetén viszont már felmerülhet olyan kérés, hogy az összes hozzájuk tartozó étteremnek a forgalmát egy helyen láthassa az igazgató. Ez persze még mindig nem feltételezi azt, hogy tárolni is egy helyen kell, de valamilyen másolatnak kell, hogy legyen. Mindenestre az biztos, hogy a folyó tevékenységek adatait az adott üzletben kell tárolni, az ott lévő szerveren. Folyó tevékenységeknek számítanak mindazon események, amelyek a fizetés előtt lebonyolódnak. Tehát az asztalfoglalás, a rendelések, stb..

A fizetés után lévő adatok viszont már nem módosíthatóak, és mindenféleképpen le kell, hogy legyenek tárolva, nem tűnhetnek el. Nem mellesleg ezek azok az adatok, amelyek egy igazgató számára értékes információkat tartalmaznak. Így jó ötletnek tűnhet, ha ezeket az adatokat a létrejöttük után azonnal feltöltjük valamilyen biztonságos szerverre. Ez azonban nem biztos, hogy a legmegfelelőbb megoldás, hiszen ez jelentős forgalmat generál az internetszolgáltatónk irányába, sőt van esély rá, hogy nincs is internet elérhetőség. Lehetséges, hogy sokkal jobban járunk, ha azt a helyi szervert használjuk fel, amit a folyótevékenységekkor is használtunk, hiszen így nem kell az internet felé küldeni az adatokat.

Jelen implementációmban az utóbbi megoldás mellett döntöttem, mivel ez egyszerűbben megvalósítható, és az esetek többségében nincs szükség az internetes elérésre, de természetes továbbfejlesztési lehetőségként továbbra is fenn áll az internetes szinkronizáció lehetősége.

* 1. Felhasználható technológiák
     1. Szerver oldal
        1. Apache Tomcat [6] és Glassfish [7]

Igaz, az előzőekben már említést tettem, hogy valamilyen Windows alapú technológiát érdemes használni a szerver oldalon, de a teljesség igénye miatt megvizsgáltam milyen lehetőségek vannak a Java területein is. E kettő gyakran használt alkalmazásszerver vizsgálatát végeztem el.

Mindkettő egy több csomagból álló keretrendszer, amely segítségével servleteket tudunk létrehozni és hosztolni. Ezen servletek fejlesztéséhez a Java és JSP nyelv ismerete szükséges. Mivel mindkettő Oracle Java alapú, ezért jól működnek mindenféle operációs rendszeren, amely számos esetben előnyt jelent. Sok keretrendszer található meg hozzá – például a LifeRay – melyekkel egyszerűbb a webalkalmazások készítése.

Számomra előnyösebb funkciókat nem találtam bennük.

* + - 1. Internet Information Services [8] [9]

Az IIS egy webszerver, amely a Microsoft Windows Server operációs rendszerekhez készült. Ennek ellenére elérhető normál operációs rendszereken is – mint a Windows XP / 7 / 8 – mindössze csak be kell kapcsolni, vagy – korlátozottabb típusú operációs rendszereken – feltelepíteni. Segítségével hosztolhatunk webalkalmazásokat, amelyek mellé számos menedzselési funkciót is kapunk, mint például:

* biztonsági funkciók,
* dinamikus gyorsítótárazás és adattömörítés,
* távoli adminisztráció,
* stb.

Mivel ez egy Microsoft termék, így a náluk megszokott módon jól megfogalmazott, elérhető dokumentációkat, mintapéldákat nyújtanak felénk. Így jóval egyszerűbb a fellépő problémáinkra megoldást találni.

Az előbb említett előnyök miatt, és mivel számomra elegendő is csak a Windows alapú használat, ezért az IIS használata mellett döntöttem. Persze ez így kifejezetten nem igaz, mivel a szerverek kutatásai közben figyelmet fordítottam a rajtuk futtatandó webalkalmazások készítési módjára, használhatóságára is. Így e döntésbe az alábbi technikák megtalálása is jócskán beleszólt.

* + - 1. Microsoft ASP.NET Web API

Microsoft ASP [10]:

Az ASP-t (Active Server Pages) a Microsoft kezdte el fejleszteni 1996-ban, azonban a köztudatban már csak a 2000-ben megjelent ASP 3.0-ás verzió terjedt el, ami azóta sem változott. A programozása VisualBasicScript segítségével történik, azonban használhatóak külső nyelvek is, mint például a PerlScript – az adott szkriptmotor telepítésével. Feladata, hogy az adott szkriptmotor előállítson egy olyan tartalmat, amelyet a szerver – ezesetben az IIS – továbbjuttat a klienseknek. Ez a tartalom lehet egy HTML alapú weboldal, vagy egy adatokat tartalmazó XML fájl is.

Microsoft ASP.NET [11]:

Ez a továbbfejlesztett verziója az ASP-nek. 2002-ben jelent meg. A .NET használata miatt, C# nyelven is lehet programozni, és az összes olyan funkció elérhető, mint amelyet a .NET Framework nyújt, így gördülékenyen, és gyorsan lehet vele weboldalakat készíteni. Egyik ilyen funkció, hogy a vezérlő elemeket például rá lehet helyezni grafikusan a weboldalra, mint sima alkalmazás készítéskor. Ez a Web Forms [12] néven terjedt el – hasonlítva a Windows Forms-ra – mivel funkcionalitásban is hasonlóan működik.

MVC [13]:

Az MVC egy mozaikszó, amely a Model-View-Controller (Modell-Nézet-Vezérlő) szavakból áll össze. Ez egy fejlesztési minta, amelyet számos helyen felhasználnak – nem csak a Microsoft – ezáltal egyszerűbben átlátható programokat lehet készíteni. Lényege, hogy az adott feladatot végző kódok egy helyen legyenek. Így például az adatbázis, vagy az adatokat szolgáló kódok a modellekben. Egy asztali alkalmazás esetén vagy egy HTML oldal esetén a megjelenítésért és HTML oldal generálásáért – tehát a felhasználóhoz eljutó felületért – , a nézetbe tartozó kódok legyenek a felelősek. A kettőt összekötő kódrészletek pedig a vezérlő csoportba tartozzanak, amelyek a felhasználói interakciókat hivatottak lekezelni.

Mint azt már említettem, ez csak egy minta, tehát a felosztás lehet szűkebb, és tágabb is. A fontos az, hogy egyértelműen legyen meghatározva hogy melyik csoportba tartozó kódok, milyen alapvető funkciókért felelnek. Ilyen MVC mintát használ az ASP.NET is.

Microsoft ASP.NET Web API [14] [15]:

Egy olyan összetett keretrendszer, amely segítségével egyszerűen lehet HTTP szolgáltatásokat létrehozni. A .NET Framework felhasználásával tudunk készíteni alkalmazásokat, amelyek megfelelnek a REST [16] [17] elveknek. A REST igazából egy már régóta élő fogalom, csak mostanában került be a köztudatba. A lényege az, hogy vannak bizonyos szabályok, amelyeknek ha megfelel a kliens és a szerver, akkor az adott kommunikáció REST-nek nevezhető. A legfontosabb szabályai:

* a kliens-szerver architektúra el van különülve
* állapotmentesek a kérések.

Ha a kommunikációnk teljesíti a REST követelményeit, akkor rendelkezhet az alábbi tulajdonságokkal, mint: skálázhatóság, hordozhatóság, megbízhatóság, egyszerűség stb..

A mai mindennapi életben használt HTTP és SOAP is REST-es.

* + - 1. Microsoft Entity Framework [18]

Az Entity Framework, egy objektum-relációs leképző keretrendszer. Használata a .NET keretrendszerhez lett kitalálva. Legfőképpen az adatorientált szoftverek fejlesztését segíti elő. Lényegében egy olyan köztes réteget kapunk az adatbázis és az alkalmazás közé, amely segítségével kihagyhatóak azon lépések, amelyeknél közvetlen az adatbázissal beszélgetnénk. Így például nem kell bonyolult lekérdezéseket, tábla összekapcsolásokat gyártanunk.

Mindezt az EF úgy nyújtja felénk, mint egy Objektum Orientált osztályt, így például az adatbázisban található táblák, leképződnek egy-egy ilyen osztállyá, a benne található sorok pedig példányokká. Ugyanígy a táblák közötti kapcsolatokat is leképzi, tehát ha van A és B táblánk, arra képez egy A és B osztályt. Így a B táblában lévő sorok, az adott A osztályomban e B táblát reprezentáló B osztály típusú adattagként jelenik meg.

Így például A osztály valamely példányának adattagját megváltoztatva, majd mentést hívva, az EF ezt leképezi egy SQL paranccsá, így az adatbázisban rögtön elmentődik a változás anélkül, hogy egyetlen egy szónyi SQL parancsot mi kiadnánk.

E keretrendszer segítségével jóval egyszerűbben lehet az adatokat kezelni, szinte ugyanúgy, mintha az egészet Objektum Orientált osztályhierarchia segítségével készítettük volna el, ugyanakkor megkapjuk az adatbázis rendszerek használatának minden funkcionalitását.

Első lépésként három lehetőségünk van: ha nincs meglévő adatbázisunk, akkor lehet először osztályok létrehozásával létrehozni az adatbázist, vagy lehet az adatbázist létrehozni, amelyből majd legenerálódnak az osztályok. Ezeket hívják CodeFirst, és EF Designer-es tervezésnek. A harmadik lehetőség, pedig ha már van tervezett adatbázisunk, és annak a mintájára szeretnénk legeneráltatni az osztályokat (ez is az EF Designer segítségével történik).

* + 1. Kliens oldal
       1. Microsoft Windows Presentation Foundation [19]

A WPF egy grafikai alrendszer, amely a Windows alapú alkalmazások megjelenítéséért felelős. A .NET Framework 3.0-ban jelent meg tehát 2006 óta érhető el. Célja ugyanaz, mint a Windows Forms-nak, csak ezt másképp éri el. A legfontosabb elve, hogy elkülönítse a megjelenési réteget, a logikai/vezérlő rétegtől. Használatához egy XML nyelvből kialakult XAML nyelv ismerete szükséges, amellyel hasonlóan tudjuk felépíteni az oldal szerkezetét, mint egy HTML oldal esetén – persze itt jóval több lehetőségünk van az XAML-nyelvvel.

Amíg a Windows Forms-nál a logikai rétegben tudtuk a felületi elemek különböző megjelenítésért felelős tulajdonságait beállítani – minthogy jelenjen meg az adott gomb vagy sem –, addig ezeket a beállításokat az XAML nyelvben tudjuk elvégezni, így még jobban betartva az MVC mintát. Nem mellesleg számos olyan funkciót vezettek be, amely a Windows Forms-nál ilyen közvetlen módon nem volt elérhető, mint például a különböző elemek effektezései: torzítás, nyújtás, forgatás, stb..

Van lehetőségünk adatkötésekre is, ami arra szolgál, hogy mondjuk egy gomb felirata, egy belső változótól függjön. Így ha a belső változót bármikor megváltoztatja valami a logikai rétegen belül, akkor az automatikusan megváltozik a megjelenítési rétegen is. Így nem kell nekünk kézzel módosítani a gomb tulajdonságát, hiszen ezt az XAML-ben definiált adatkötés elvégzi automatikusan helyettünk.

Számos ilyen plusz funkció került bele a WPF-be, amely így egy teljes értékű program létrehozását teszi lehetővé, amely megfelel a mai elvárásoknak is.

Azonban a HTML5 megjelenésével, és egyre szélesebb körű támogatásával kezd visszaszorulni a WPF használata.

* + - 1. HTML5 és SPA

HTML5 [20] [21] [22]:

A HTML4 [20] egy kifejezetten jó nyelv, de sajnos az idő múlásával – mint minden – ez is elavult. Hogy megfelelő tartalmat tudjon nyújtani a felhasználóinak, különböző beépülőket kell használni, amelyek így egyéb telepítést igényelnek, és vannak olyan eszközök ahol erre nincsen lehetőség.

Vegyük például a videók megjelenítését: lehet az Adobe Flash Playert használni, lehet a Silverlightot, de a HTML4 nyelv nem nyújt arra lehetőséget, hogy ő maga jelenítse meg a tartalmat. Így minden eszközre ahol videót szeretnénk nézni, valamilyen külső programot kell feltelepíteni, amelyek így plusz védelmi réseket nyithatnak a számítógépünkön. Ráadásul, vannak olyan rendszerek, amelyekre például megszűnt az Adobe Flash Player támogatása, így hivatalos megoldással nem tudunk Flash tartalmakat megjeleníteni – ez történt az Android 4.1 megjelenésével [21] is.

De ez a fajta elavultság igaz a megjelenésre is, így csak egyszerű stílusokkal tudunk felépíteni egy weboldalt, illetve trükközéssel tudunk más stíluselemeket használni. Erre nagyon jó példa a lekerekítés. Ha egy egyszerű – dinamikusan változó méretű – téglalapot lekerekített sarkokkal szeretnénk ábrázolni, ahhoz a HTML4 esetén különféle trükközésekkel, képek készítésével és bepozícionálásával tudjuk megvalósítani. Ez a HTML5 esetén egyetlen egy CSS tulajdonság megadásával elérhető.

Fontos azonban megjegyezni, hogy a HTML5 nem egy nyelvből tevődik össze. Ez egy olyan „programcsoport” amely kifejezetten jól együttműködve, tudja elérni mindazt, amire hivatott: egy egységes tartalommegjelenítést böngésző-, tartalom-, eszköz- és felbontás függetlenül.

Mindezt a megújított HTML4-el, a CSS3-al, és a továbbfejlesztett JavaScript-el tudja elérni. Olyan funkciók lettek elérhetőek, mint például:

* teljes mértékű CSS3 támogatás (új effektusok)
* beépített videó/hang megjelenítés
* 2D/3D-s grafika (Canvas – rajzfelület)
* kliensoldali adattárolók/adatbázisok
* …

2004-ben kezdődött el a HTML5 fejlesztése, és még a mai napig is tart. Jelenleg úgy gondolják, hogy 2016-ra lesz kiforrott, ennek ellenére már nagyon sokan kezdenek áttérni erre a technológiára.

SPA [22] [23]:

A Single Page Application, egy egy oldalból álló webalkalmazás. Célja olyan weboldalak létrehozása, amelyek inkább hasonlítanak normál asztali alkalmazásokra, mint weboldalakra. Ez főleg a Cloud (felhő) szolgáltatások megjelenésével jelent meg, hiszen így elegendő egy SPA-t készíteni, amelyet a felhőbe feltéve, egy bárhonnan, bármilyen platformról elérhető – akár komplett – alkalmazást kapunk.

Ezen SPA-k lelke a HTML5. Vannak direkt erre szakosodott keretrendszerek, amelyek segítségével jóval egyszerűbben, és jóval átláthatóbb webalkalmazást tudunk készíteni.

Összességében talán az egyik legjobb ilyen keretrendszer az AngularJS.

* + - 1. Google AngularJS [24] [25]

A Google fejleszti már 2009 óta, és összességében talán az egyik legjobb választás egy SPA elkészítésére. Számos más ilyen keretrendszer is létezik – BackBone.js [26], Ember.js [27] vagy Knockout.js [28] – de az e mögött álló támogatás, folyamatos fejlesztések, dokumentációk, felhasználók száma miatt érdemesebb ezt választani.

Összességében egy olyan JavaScript csomagról beszélhetünk, amely segítségével kliens oldalon tudunk nem csak megjelenítésbeli módosításokat végezni, de egy komplett MVC rendszert ki tudunk építeni. Mindezt számos eszközzel segíti, és számos funkciót nyújt:

* adatkötések Nézet és Modell között,
* egyszerű szerverkommunikáció – nincs oldalfrissítés,
* újrafelhasználható komponensek,
* lokalizáció,
* …
  + 1. Összegzés

A szerver típusának kiválasztásában is sokat segített az ASP.NET és az Entity Framework megtalálása, mivel ezek segítségével jóval könnyebben és gyorsabban tudom majd elkészíteni a programomat. Az ASP.NET Web API segítségével tervezem megvalósítani a két kliens kommunikációját a szerverrel, és kiszolgálni az adminisztrációs felületet. A szerveren pedig Entity Framework segítségével tárolni az adatokat.

A mobil klienst az Android nyújtotta lehetőségekkel készítem el, míg az asztali klienst pedig a WPF segítségével. Hogy az adminisztrációs felület minél jobban átlátható legyen, ezért számos interakciót kell beleépíteni, amelyre a legmegfelelőbb technológia a HTML SPA használata. Így ráadásul – mivel lehet is rá igény –, az adminisztrációs felületet meg is lehet nyitni az internet felé, így a vezetők bárhonnan tudják kezelni a rendszert egy egyszerű böngésző használatával. Ezen belül az AngularJS keretrendszert tervezem használni, mivel jól dokumentált és kiforrott rendszer.

1. IRODALOMJEGYZÉK

[1] Gartner: 55 Percent of Overall Mobile Phone Sales in Third Quarter of 2013  
(http://www.gartner.com/newsroom/id/2623415), utoljára megtekintve: 2014-04-10.

[2] Kamex  
(http://www.kamex.hu/), utoljára megtekintve: 2014-04-10.

[3] MultiStore videó  
(http://www.youtube.com/watch?v=-U4QuDGfArI), utoljára megtekintve: 2014-04-10.

[4] eSystem Pincér és Vendéglátó program  
(http://www.vendeglatoprogram.com/), utoljára megtekintve: 2014-04-18.

[5] Videók - eSystem  
(http://www.vendeglatoprogram.com/videok), utoljára megtekintve: 2014-04-18.

[6] Apache Tomcat - Welcome!  
(http://tomcat.apache.org/), utoljára megtekintve: 2014-05-02.

[7] GlassFish Server  
(https://glassfish.java.net/documentation.html), utoljára megtekintve: 2014-05-02.

[8] Internet Information Services  
(http://www.iis.net/), utoljára megtekintve: 2014-05-04.

[9] Internet Information Services - Wikipedia  
(http://en.wikipedia.org/wiki/Internet\_Information\_Services), utoljára megtekintve: 2014-05-04.

[10] Active Server Pages - Wikipedia  
(http://en.wikipedia.org/wiki/Active\_Server\_Pages), utoljára megtekintve: 2014-05-04.

[11] ASP.NET  
(http://www.asp.net/get-started), utoljára megtekintve: 2014-05-04.

[12] Microsoft Web Forms  
(http://msdn.microsoft.com/en-us/library/ms973868.aspx#introwebforms\_topic1), utoljára megtekintve: 2014-05-04.

[13] ASP.NET MVC  
(http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview), utoljára megtekintve: 2014-05-04.

[14] ASP.NET Web API  
(http://www.asp.net/web-api), utoljára megtekintve: 2014-05-04.

[15] ASP.NET Web API  
(http://msdn.microsoft.com/en-us/library/hh833994%28v=vs.108%29.aspx), utoljára megtekintve: 2014-05-04.

[16] REST  
(http://stackoverflow.com/questions/671118/what-exactly-is-restful-programming), utoljára megtekintve: 2014-05-04.

[17] REST - Wikipedia  
(http://hu.wikipedia.org/wiki/REST), utoljára megtekintve: 2014-05-04.

[18] Entity Framework  
(http://msdn.microsoft.com/en-us/data/ee712907), utoljára megtekintve: 2014-05-04.

[19] Microsoft Windows Presentation Foundation  
(http://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx), utoljára megtekintve: 2014-05-04.

[20] HTML4  
(http://www.w3.org/TR/html401/), utoljára megtekintve: 2014-05-04.

[21] Adobe Flash megszűnése Androidon  
(http://www.telefonguru.hu/hir/Ma\_megszunik\_a\_Flash!\_2012-8-15), utoljára megtekintve: 2014-05-07.

[22] SPA .NET-ben  
(http://msdn.microsoft.com/en-us/magazine/dn605877.aspx), utoljára megtekintve: 2014-05-07.

[23] SPA  
(http://en.wikipedia.org/wiki/Single-page\_application), utoljára megtekintve: 2014-05-07.

[24] AngularJS  
(https://angularjs.org/), utoljára megtekintve: 2014-05-07.

[25] AngularJS - Wikipedia  
(http://en.wikipedia.org/wiki/AngularJS), utoljára megtekintve: 2014-05-07.

[26] BackBoneJS  
(http://backbonejs.org/), utoljára megtekintve: 2014-05-07.

[27] EmberJS  
(http://backbonejs.org/), utoljára megtekintve: 2014-05-08.

[28] KnockoutJS  
(http://knockoutjs.com/), utoljára megtekintve: 2014-05-08.

[29] HTML5 - W3.org  
(http://www.w3.org/TR/html5/), utoljára megtekintve: 2014-05-07.

[30] HTML5 - W3Schools  
(http://www.w3schools.com/html/html5\_intro.asp), utoljára megtekintve: 2014-05-07.

[31] HTML5 - Wikipedia  
(http://en.wikipedia.org/wiki/HTML5), utoljára megtekintve: 2014-05-07.