**Tartalomjegyzék**

[Bevezetés 3](#_Toc467444979)

[1. Feladat leírása 4](#_Toc467444980)

[2. Hasonló alkalmazások megismerése 5](#_Toc467444981)

[2.1. GSMtasks 5](#_Toc467444982)

[2.2. Főbb jellemzői 6](#_Toc467444983)

[2.2.1. Fleet Managment (FM) 7](#_Toc467444984)

[2.2.2. Field service management (FSM) 7](#_Toc467444985)

[2.3. Viamente 8](#_Toc467444986)

[2.3.1. Főbb jelemzői 8](#_Toc467444987)

[3. Kiindulási alapötlet 9](#_Toc467444988)

[3.1. Tervezés 10](#_Toc467444989)

[4. Használt technológiák 14](#_Toc467444990)

[4.1. Git: 14](#_Toc467444991)

[4.1.1. Miért a Git? 14](#_Toc467444992)

[4.2. MySQL 15](#_Toc467444993)

[4.2.1. Miért a MySQL? 15](#_Toc467444994)

[4.3. C# - Microsoft Visual Studio 2015 15](#_Toc467444995)

[4.3.1. Microsoft Visual Studio 16](#_Toc467444996)

[4.3.2. Miért C# és Microsoft Visual Studio? 16](#_Toc467444997)

[4.4. Trello 16](#_Toc467444998)

[4.4.1. Mire jó a Trello? 17](#_Toc467444999)

[4.4.2. Miért a Trello? 17](#_Toc467445000)

[5. Vehicle Routing Problem (VRP) ismertetése 18](#_Toc467445001)

[5.1. Általános leírás 19](#_Toc467445002)

[5.2. VRP modell 19](#_Toc467445003)

[5.3. Definíció (legrövidebb út) 22](#_Toc467445004)

[5.4. VRP változatai 22](#_Toc467445005)

[5.5. Költség megtakarítás 24](#_Toc467445006)

[6. C#-ban történő megvalósítás 25](#_Toc467445007)

[7. Jövőbeni tervek 27](#_Toc467445008)

[Összegzés 28](#_Toc467445009)

[Irodalomjegyzék 29](#_Toc467445010)

[Ábrajegyzék 31](#_Toc467445011)

# Bevezetés

A mai világban fontos szerepet tölt be, hogy a cégek az ügyfeleiket oly módon szolgálják ki, hogy a rendelkezésre álló munkaerőt és eszközparkot optimálisan használják ki, a lehető legtöbb feladat teljesítése mellett. Legyen szó, futárszolgálatról, étteremről, akik házhozszállítást is vállalnak, vagy bármely más, szállítással, fuvarozással, probléma elhárítással foglalkozó cégről.

A vállalatok sok időt és – ami még fontosabb- pénzt tudnak megtakarítani azzal, ha feladataik jól ütemezik, a teljesítésük útvonalát pedig optimálisan választják meg. Ezzel a témával foglalkozik a **Vehicle Routing Problem (VRP**) is, amelyről számos információt és megközelítést találhatunk mind az irodalomban, mind az interneten. Magyarul jármű útválasztási problémát jelent, ha szó szerint fordítjuk le, azonban én továbbiakban is az angol kifejezést használnám, melynek fő oka, hogy leginkább idegen nyelvű források találhatók a témához.

Maga **a VRP** egy kombinatorikus optimalizálási és egészértékű programozási probléma. **Arra a kérdésre keresi a választ**, hogyha adottak a kiszolgálásra váró ügyfelek, feladatok, amelyekhez mindenképpen el kell jutnunk, mely vagy **melyek azok az optimális útvonalak**, amiken keresztül haladva eljuthatunk hozzájuk és idő valamint **költségek tekintetében** számunkra a **legkedvezőbb eredményt foglyuk kapni**?

Az első fejezetben a feladatmeghatározásról lesz szó, ahol részletesen kifejtem elvégzendő feladatatom és a hozzákapcsolódó követelményeket.

A második fejezetben a hasonló alkalmazások utáni kutatásom és tapasztalataim fejtem ki. Majd ismertetem ezeket a szoftvereket és a tulajdonságaikat.

A harmadik fejezetben kiindulási alapötletem fogom felvázolni.

A negyedik fejezetben a használt technológiákat ismertetem, amelyeket a feladatom megvalósítása közben használtam.

Az ötödik fejezetben a Vehicle Routing Problem (VRP) részletes ismertetése következik. Kifejtve pontosan mit is takar ez a fogalom, milyen fajtái vannak, hol találkozhatunk vele és miért olyan hasznos ez?

Majd mindezeket követi a hatodik fejezet, ahol a C#-ban megvalósított változat rövid ismertetése következik.

Végezetül pedig a jövőbeli továbbfejlesztési lehetőségekről és tervektől lesz szó.

# Feladat leírása

Egy cég számára fontos, hogy a rendelkezésre álló munkaerőt és eszközparkot optimálisan használják ki, a lehető legtöbb feladat teljesítése mellett. Ennek a támogatásához kell egy olyan alkalmazást elkészíteni, ahol különböző vagyonelemek, például villanyvezetékek és oszlopok karbantartására lehet csapatokat vezényelni.

A program nyilvántartja a feladatokat, valamint az egyes csapatokhoz tartozó embereket, eszközöket. Egy csapat összetétele előre meghatározott, azonban a különböző elvégzendő munkák más-más eszközöket és eltérő képesítésű szakembereket igényelnek.

Ki kell választanunk egy csapatot, a szoftver pedig segítséget nyújt az ütemezőnek azon feladatok kiválasztásában, amelyet az adott csapat teljesíteni tud. Ezt követően útvonal optimalizálási algoritmust alkalmazva meghatározza, hogy azokat milyen sorrendben teljesítsék. Különböző számítások segítségével pedig megállapítja a távolságot és becslést ad a munka várható befejezi idejéről.

# Hasonló alkalmazások megismerése

Egy szoftver megtervezése és elkészítése előtt elengedhetetlen lépés, hogy utánajárjunk milyen hasonló, azonos vagy közel azonos feladatokat ellátó szoftveralkalmazások érhetőek el a piacon.

Ennek tudatában felmérhetjük a versenytársaink által nyújtott programok szolgáltatásait, ezek hiányosságait, vagy épp erősségeit. Tanulmányozhatjuk, hogy egy adott problémát, az ő applikációjuk milyen módon közelített meg. Ötleteket meríthetünk, és a kutatásaink figyelembe vételével megtervezhetjük a saját programunk.

Az én témám a „Mobil munkaerőmű menedzsmentet támogató keretrendszer megvalósítása”. Így a kutakodásom a *mobil munkaerő menedzsment* és a *munkaerő menedzsment,* valamint *az útvonal optimalizálás (VRP témaköréhez tartozik)* témaköröket átfogó szoftverek köré összpontosult.

Számos ilyen alkalmazás érhető el jelenleg a világhálón, azonban kevés az, ami az általam megvalósítandó feladattal -vagy ahhoz hasonlóval- foglalkozik. Számomra egy olyan program szükség, ahol számon tarthatjuk a faladataink, csapataink, azon belül is az emberek, eszközök és járművek listáját. Csapatokat feladatokkal társíthatunk, mindezt úgy, hogy a rendelkezésre álló munkaerőt és eszközparkot optimálisan használjuk ki, a lehető legtöbb megbízatás teljesítése mellett. A program elvégzi az útvonal megtervezését és a szükséges számítások elvégzését, mint például az utazási időt, becsült munkaidő, megtett távolság stb.

Az első igazán ígéretes szoftver, ami hasonló feladatokat lát el az a GSMtasks.

## GSMtasks

Ez egy webes felületen elérhető mobil munkaerő menedzsmentet támogató fizetős alkalmazás. Rendelkezik telefonos alkalmazással is mind a két nagy platformra, így IOS-re és Androidra is. Három területet ölel át az alkalmazás, [Fleet Management](https://www.gsmtasks.com/fleet-management-software/) (Flottakezelő)[, Field Service Management](https://www.gsmtasks.com/field-service-management-software/) (Területi Szolgáltatás Menedzsment), [Route Optimization](https://www.gsmtasks.com/route-optimization-software/) (Útvonal optimalizálás).

A webes felületen tudjuk kezelni feladatainkat, illetve embereinket. Lehetőségünk van személyzetünk bővítésére, feladatok felvételére, valamint a kiválasztott feladatok alkalmazotthoz rendelésére.

## Főbb jellemzői

Teljes áttekintést biztosít a feladatokról, hogy melyek azok, amik még teljesítésre várnak, melyek, amik éppen folyamatban vannak, illetve azok amelyek már teljesítve lettek. A webes felületen, térkép segítségével látható, a járművezető aktuális pozíciója, valamint nyomon követhető, hogy mely feladatok vannak még hátra, és melyek lettek már elvégezve.

Statisztikával rendelkezik arról, hogy meddig vannak úton az egyes sofőrök, mekkora távolságot kell megtenniük és ebből mennyi van még hátra. Becslést ad arról, hogy mennyi ideig tart a feladatok elvégzése.

Minden dokumentumot, szállítási iratot fel lehet tölteni online az adott feladathoz, így digitálisan kezelhetővé válnak, ezzel segítve azt, hogy minél kevesebb papírt használjanak fel.

Valós idejű frissítéssel rendelkezik a szoftver. Aktuális állapottal rendelkezik az egyes feladatuk státuszáról és mindig naprakész információt biztosit a felhasználónak, aki jelen esetben a diszpécser és a sofőr.

Útvonal optimalizálással rendelkezik, tehát a teljesítendő feladatok kiválasztását követően a GSMtasks a leghatékonyabb útvonalat tervezi meg.

Talán a legpraktikusabb funkciója az alkalmazásnak, hogy elérhető telefonos applikáció formájában is, így a sofőr a mobilján, a saját felhasználójával bejelentkezve láthatja a mai napon elvégzendő feladatok listáját és az ezekhez fűződő fontosabb információkat (pl. helyszín), mindezt valós időben.

Az eddigi funkciók mellett könnyű integrációt biztosít a jelenleg használt alkalmazásokhoz.

A GSMtasks alkalmazása használható ételkihordással foglalkozó éttermek, futárszolgálatok és költöztető, valamint egyéb szállítással foglalkozó cégek számára a feladataik ütemezésében, optimális útvonal megtervezésében.

Itt szeretnék kicsit kitérni a fent említett **Fleet Managment (FM)** és **Field Service Managment (FSM)** fogalmakra, helyre tenni ezek jelentéseit, mire is szolgálnak ezek és miért hasznosak.  
Az útvonal optimalizálásról a Vehicle Routing Problem tárgyalásakor szó fog esni, így ezzel most nem szeretnék foglalkozni.

### Fleet Managment (FM)

A Fleet Managment magyar nevén flottakezelés, mint a nevében is szerepel, a járműparkunk menedzselésére szolgál. A jármű fogalma jelen esetben a motoroktól a hasznosgépjárműveket és a teherautókon át, a repülőgépekre, hajókra és minden egyéb kereskedelmi járműre kiterjed.

A Fleet Managment hatásköre igen nagy, ide tartozik a gépjármű-finanszírozás, gépjármű karbantartás, gépjármű telematika -járművek számítógép vezérlésű irányítástechnikája- (követés és diagnosztika), sebesség és üzemanyag kezelés.

A flottakezelés egy olyan funkció, amely a lehetővé teszi a szállítással foglalkozó vállalatok számára, hogy a beruházásaikat csökkentsék (felesleges jármű vásárlás). Segítségével javítani lehet a hatékonyságot, a termelékenységet és csökkenteni a szállítási és személyzeti költségeket.

### Field service management (FSM)

Egy vállalat különböző szolgáltatásokat nyújt az ügyfelei számára egy adott térségen belül.

Az FSM-et számos iparágban használják az erőforrások kezelésére:

* A távközlés iparban, amikor a technikusok kábelt telepítenek vagy telefon vonalat hoznak létre lakóházak, üzleti létesítmények között
* Az egészségügyben a mobil ápolók, akik otthoni gondozást vállalnak
* Közművesek, mérnökök, technikusok körében, akik karbantartási és megelőző javítási munkákat végeznek.
* stb.

A második hasznosnak vélt szoftver, ami ezzel a témakörrel foglalkozik a Viamente.

## Viamente

Egy útvonal és ütemezés optimalizálására szolgáló online alkalmazás. Használható flotta kezeléshez, mobil munkaerő menedzsmenthez. Segítségével jelentős költségeket takaríthatunk meg, hiszen az optimálisan megtervezett útvonalak minimalizálják az üzemanyag költséget, valamint nincs felesleges jármű felhasználás sem. Automatikus ütemezéssel rendelkezik, amely egy idővonalon számolja a becsült érkezési időt, szemlélteti az egyes célállomások közti utazásiidőt, valamint beiktatja az ebédszüneteket is.

### Főbb jelemzői

Letisztult kezelő felülettel rendelkezik, ahol kiválaszthatjuk a sofőrt, hozzáadhatjuk az elvégzendő feladatokat. Majd ez követően egy gombnyomásra megtervezi az optimális útvonalat, figyelembe véve a jármű kapacitását és a munkaidőt. A térképen pedig láthatóak a feladatok és az optimális útvonal. A képernyő alján az idővonalsávon megjelenik az automatikus ütemezés, ahol akár „fogd és vidd” módszerrel átrakhatunk egy tetszőleges feladatot egy másik sofőrhöz. A megtervezett ütemterv után beérkező feladatokat is eltudjuk látni, csak hozzá kell adni az egyik sofőrhöz, majd valós időben újratervezi a szoftver az útvonalat és átszámolja az utazási időt.

A különböző feladatokat feltölthetjük táblázat formájában (Excel, CSV) vagy az adatbázisunkhoz csatlakoztatva is lekérdezhetjük azokat.

Miután megterveztünk egy útvonaltervet és véglegesítettük azt, megoszthatjuk ezt a sofőrökkel, akikhez az adott feladatcsoport tartozik. Lehetőség van elküldeni az útvonalat email-ben, amit közvetlen meg tud nyitni az okoskészülékével, vagy továbbíthatjuk az útvonalat tartalmazó fájlt, amelyet a telefon navigációs szoftverébe betöltve beállításra kerülnek a célállomások, de akár ki is nyomtathatjuk a célállomások listáját, illetve a térképet.

A mobil applikáció megléte egy hatalmas pozitív tényező. Az alkalmazáson keresztül bármikor értesíthetjük a sofőrt a fennálló forgalmi akadályokról, a késést elkerülvén újratervezhetjük az útvonalat, majd ezt a módosított változatott elküldhetjük a gépjárművezető számára. Valamint a telefonos alkalmazás megoldást jelent az olyan kisebb vállalkozások számára is, akik nem engedhetik meg maguknak, hogy minden gépkocsiba GPS-t szereljenek.

Felhasználó bázisuk igen nagy. Túlnyomórészt Amerikában, Angliában és Olaszországban használják a cégek a Viamente alkalmazását útvonal optimalizálásra. Azonban a világ többi országában is előszeretettel veszik igénybe ezt a szoftvert, köztük Magyarországon is. Számos különböző feladatokat ellátó cég alkalmazza a Viamente által nyújtott szolgáltatásokat. Használják takarítást, kiszállítást, karbantartást és sok másmunkakört ellátó cégek.

# Kiindulási alapötlet

A hasonló alkalmazások tanulmányozása után levontam a következtetéseimet. A pozitív és negatív tulajdonságokat összegyűjtöttem.

A feladat megvalósítása előtti kutatások rendkívül fontosak voltak és a tapasztaltak nagyban hozzájárultak a saját szoftverem megtervezésének megkönnyítéséhez is. Jó kiindulási alapötleteket adtak, láthattam miben kell javítanom, milyen fontos szempontokat kell betartanom a tervezés és megvalósítás során.

A kutatásom során fellelt alkalmazások webes felületet használtak. Szép és mutatósak voltak, funkciójukat ellátták, azonban első találkozás alkalmával kissé ijesztően komplex és átláthatatlan érzést adtak a felhasználónak. Ez természetesen elmúlik, ahogy egyre több időt tölt el vele az ember, de úgy éreztem az én szoftveremnek letisztultabbnak és egy átlag felhasználó szempontjából könnyebben kezelhetőnek kell lennie. Hiszen, azok számára, akik egy ilyen feladatot ellátó alkalmazást használnak, sokkal fontosabb a kezelhetőség és letisztultság, mint a káprázatos kinézet. Emellett én a webes felület helyett egy különálló szoftvert képzeltem el, amely természetesen grafikus felülettel rendelkezik az átláthatóság és könnyed kezelhetőség érdekében. Valamint, igaz ez csak saját meglátásom, de én sokkal inkább bízok egy külön szoftverben, mint egy weblapban, főleg, ha az ilyen fontos számításokat lát el, amely jelentős költségeket és időt tud megspórolni a vállalkozások számára.

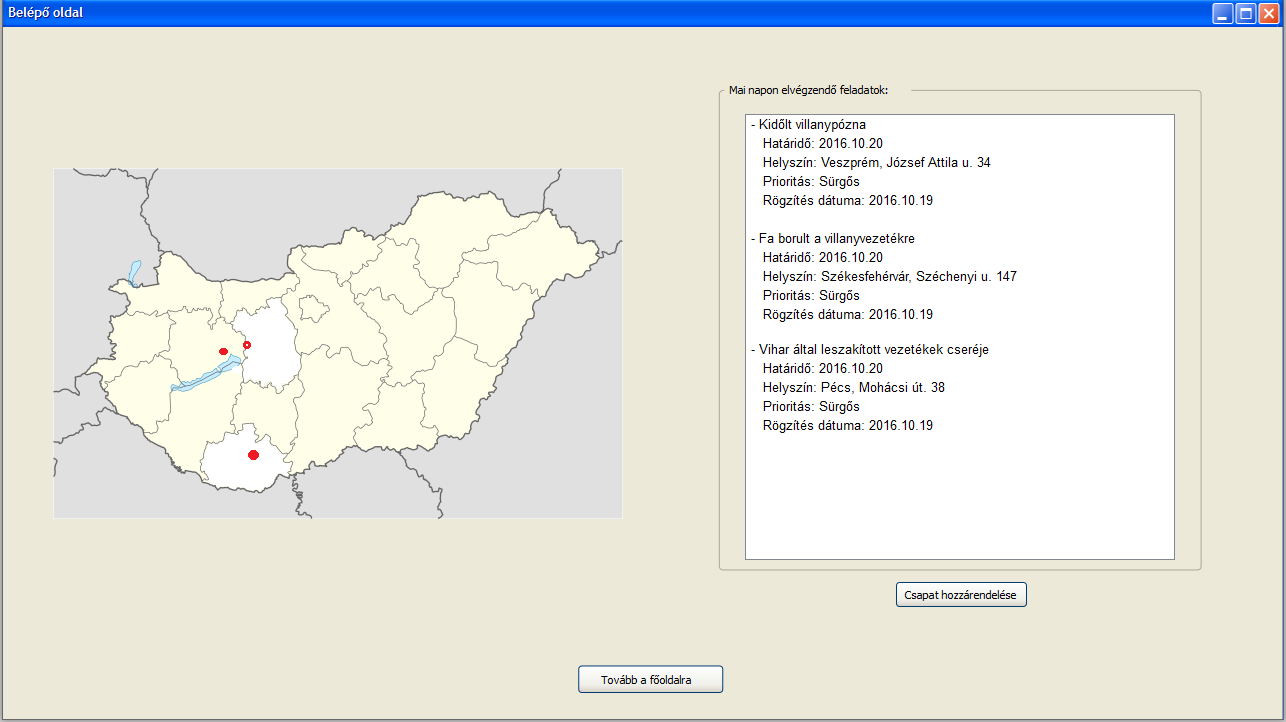
Egy másik fontos szempont, hogy a feladatok kiosztása, csapatokhoz rendelése átlátható és egyszerű legyen. A szoftver végezzen el miden számítás, segítse a felhasználót az optimális feladat kiosztás elvégzésében. Tervezze meg automatikusan az útvonalat, de ha szükséges lehessen változtatni rajta. Ne engedje, hogy egy csapathoz túl sok feladatot adjunk. Vegye figyelembe a csapatok képességeit, hogy ne történhessen meg, hogy olyan elvégzendő munkát rendeljünk hozzá egy brigádhoz, amelyet nem képesek elvégezni, vagy idő, vagy eszköz, esetleg szaktudás hiányának következtében. Számolja az utazási időt, a várhatóbefejezési időt valamint a megtett távolságot.

Legyen lehetőség új feladatok felvételében. Emellett az általam talált alkalmazások esetében nem volt naptár, amelyen láthatóak a feladatok és azok határideje. Így ezt a saját alkalmazásomban mindenképp implementálni szeretném, mivel egy hasznos funkciónak tartom, amely segít átlátni a határidőket.

## Tervezés

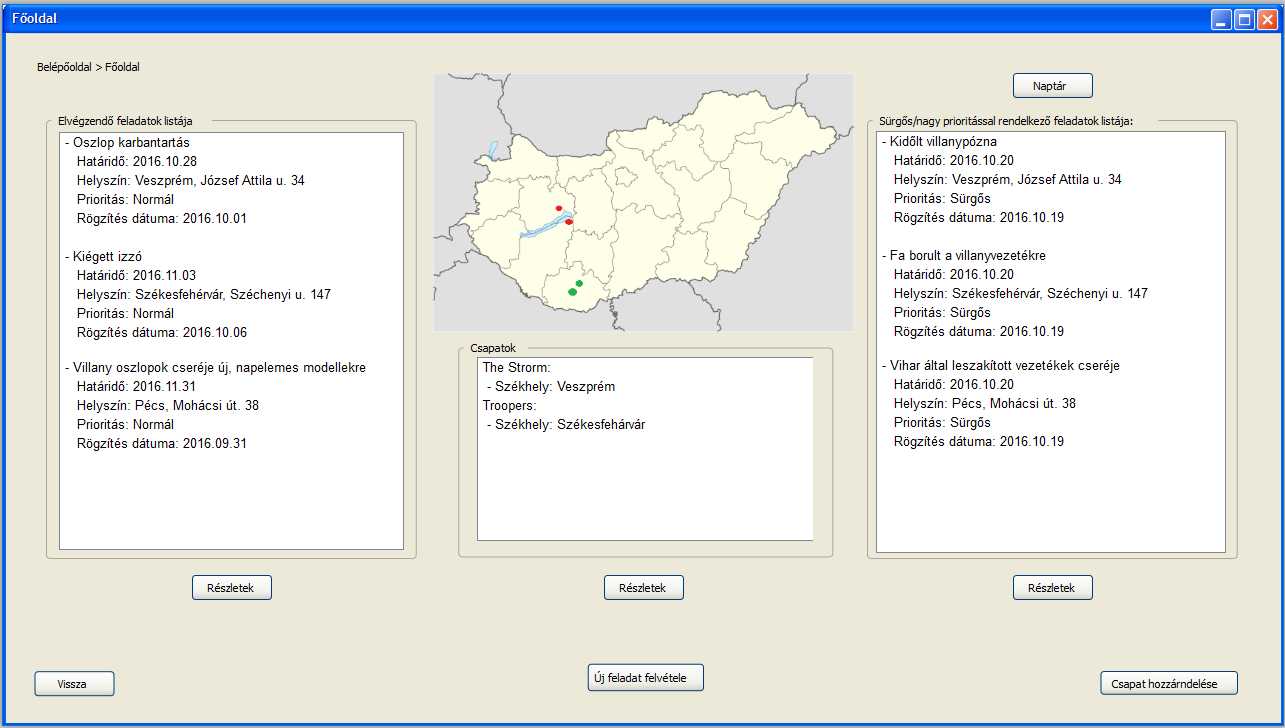
A tervezési feladataim a képernyőtervek elkészítésével kezdtem. A kiindulási alapötletek felhasználásával megterveztem a szükséges képernyőket, űrlapokat. A programom 4 fő lapból áll, ezeken találhatóak a fontosabb információk, és funkciók, amelyek ellátják a szükséges feladatot.

A program indulásakor egy „Belépő oldal” fogadja a felhasználót (1.ábra). Itt a nevével ellentétben nem a bejelentkezésre kerül sor. Ezen az oldalon lesznek kilistázva a mai határidővel rendelkező feladatok, valamint a tervek szerint egy térkép lesz elhelyezve, amin a feladatok helye jelenik majd meg. Akár erről az oldalról lehetőség nyílik majd feladatokat rendelni az egyes csapatokhoz.



3.1. ábra: A Belépő oldal képernyő terve

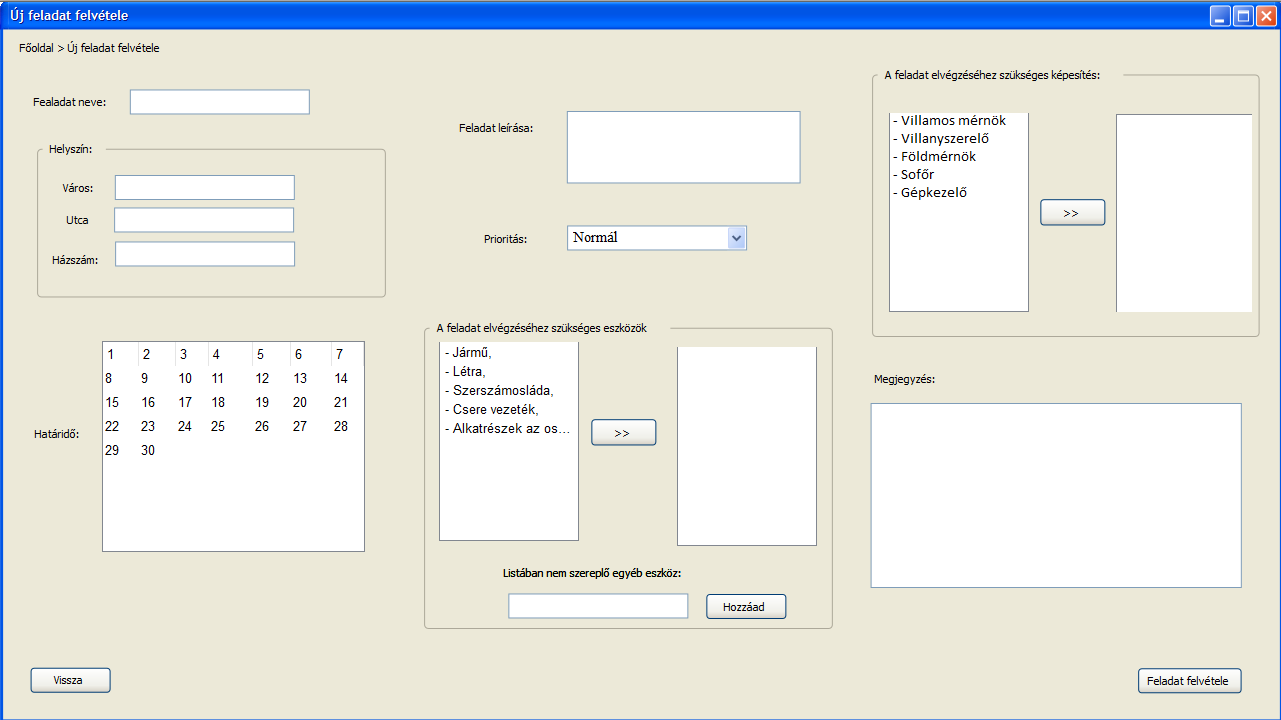
E képernyőt elhagyva juthatunk a „Főoldalra”. Itt láthatóak a feladatok listája, külön-külön bontva a sürgős, és normál prioritással rendelkező feladatokat. Láthatóak ezen kívül a csapatok listája is, valamint, hogy az egyes csapatoknak milyen városba helyezkedik el a telephelyük. Erről az oldalról eljuthatunk egy olyan oldalra, ahol az új feladatokat tudjuk felvenni az adatbázisba. Továbbá innen a főoldalról juthatunk majd el a naptárhoz is, ahol minden feladat határideje szerint lesz majd látható. A tervek szerint a „Főoldalon” is elérhető lesz egy térkép, amin a feladatok pontos helye lesz megtalálható.



3.2. ábra: Főoldal képernyőterve

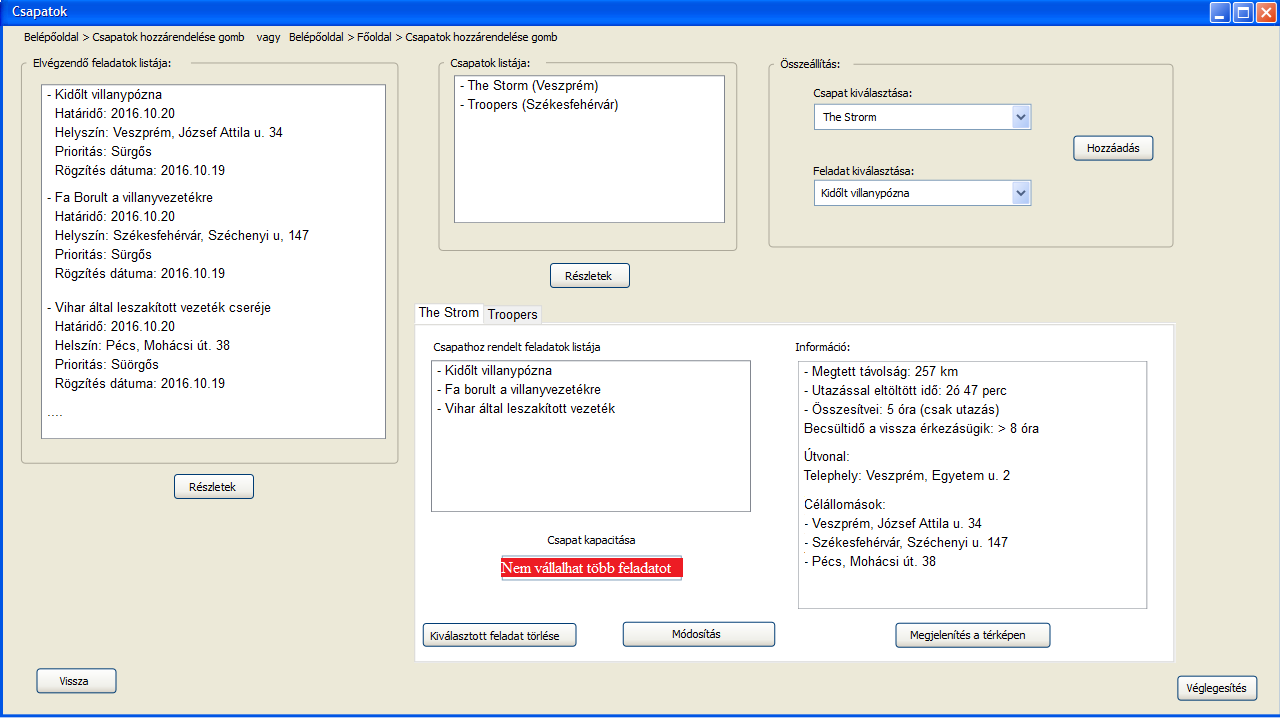
Mindemellett ezen a képernyőn lehetőség van a feladatok vagy a csapatok esetében egy részletesebb áttekintésére is. Ekkor egy új oldalra lépünk át, ahol a kiválasztott feladatról, csapatról minden rendelkezésre álló információt megtudhatunk.

Az alábbi képen annak az oldalnak a képernyő terve látható, ahol új feladatokat rögzíthetünk az adatbázisban:

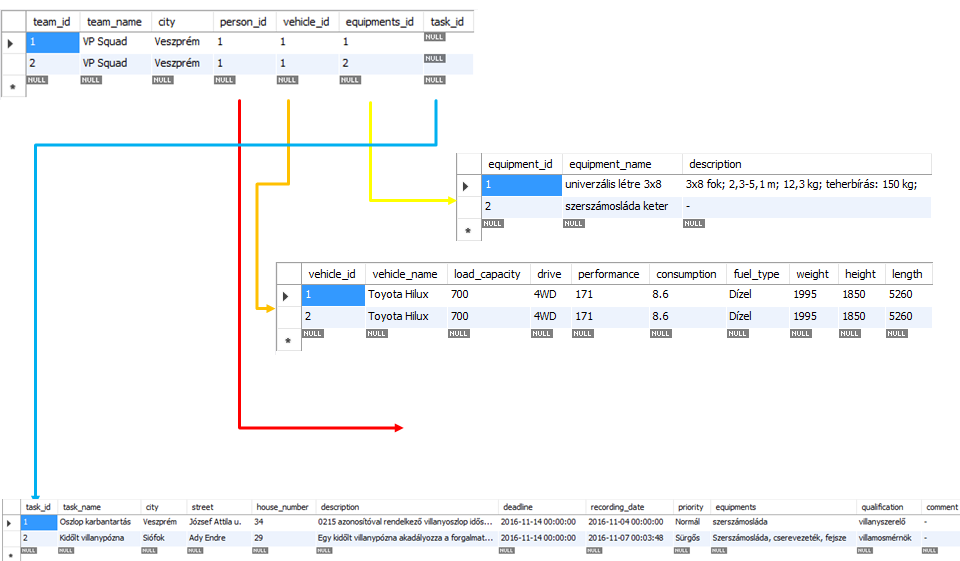
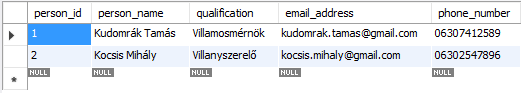


3.3. ábra: Új feladat felvétele oldal képernyőterve

Végezetül a szoftver szempontjából a legfontosabb képernyő, a „Csapatok hozzárendelése”. Mondhatni ez a „szuper oldal”, ahol minden számítás és háttérfolyamat elvégződik. Láthatóak a feladatok és a csapatok listája ezen a képernyőn is, ezzel segítve a felhasználót az áttekintésben. Azonban mikor a „Csapat kiválasztása” combobox-ban kiválasztjuk a csapatot, amelyikhez feladatot szeretnénk rendelni, akkor a feladatok listájában csak azon feladatok jelennének meg, amelyeket az adott csapat képes elvégezni. Valamint csak ezek közül tudnánk választani a „Feladat kiválasztása” combobox-ban. Így elkerülhetjük azt, hogy olyan feladatot rendeljünk egy csapathoz, amelyet nem képes elvégezi. Miután hozzárendeltünk egy feladatot egy csapathoz, az megjelenik a „Csapathoz rendelt feladatok listájában”. Minden egyéb információ pedig a mellette lévő box-ban jelenik meg (távolság, utazási idő, célállomások, stb). Látható, hogy az adott csapathoz társíthatunk e még feladatot, vagy már nem képes többet elvégezni. Lehetőség van egykiválasztott feladatot eltávolítani a csapattól, módosíthatjuk az útvonalat, esetleg megnézhetjük a térképen a megtervezett optimális útvonaltervet. Ez az oldal képezi a programunk magját, a legtöbb számítás itt végződik el, így ez a képernyő a legkomplexebb mind közül.



3.4. ábra: Csapatok hozzárendelése oldal képernyőterve

A képernyőtervek elkészítését követően pedig megterveztem az adatbázist, amelyből a program az adatokat fogja kinyerni. Létrehoztam 5 táblát. Ebben tárolom külön-külön a csapatok adatait, az eszközöket, járműveket, embereket valamint a feladatokat. Az ezek közötti kapcsolatot és magát a táblákat az alábbi kép illusztrálja:

3.5. ábra: Az adatbázis felépítése

Az egyestáblák id mezői idegen kulcsok, segítségükkel lehet a táblákat összekapcsolni. A tábla egyes oszlopai vehetnek fel null értéket, azonban a fontos adatokat tárolő mezők nem.

# Használt technológiák

## Git:

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver, amely a sebességre helyezi a hangsúlyt. A Gitet eredetileg Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez.

Segítségével fájljaink, dokumentumaink különböző verzióit tudjuk kezelni és tárolni. A Git használata során kiadott „commit” parancs az kiválasztott könyvtárról csinál magának egy helyi adatbázist a .git nevű könyvtárba. Ezekkel az adatbázisokkal:

* nyomon tudjuk követni, hogy mikor hogyan változott a könyvtárunk,
* visszaállíthatjuk bármelyik korábbi (commit-olt) állapotát a könyvtárnak,
* képes szinkronizálni egy másik gépen levő hasonló könyvtárral, közben intelligensen átvezeti a változásokat, illetve jelez, ha problémába ütközött.

A központi szerverrel történő hálózati kommunikáció helyett a lokális számítógépen hajtódnak végre a parancsok, így a fejlesztés offline megy végbe a workflow megváltoztatása nélkül.

Mivel minden egyes fejlesztő lényegében teljes biztonsági másolattal rendelkezik az egész projektről, ezért a szerver meghibásodásának, a tároló megsérülésének vagy bármilyen bekövetkező adatvesztésnek a kockázata sokkal kisebb, mint a központosított rendszerek által támasztott pont-hozzáférés esetében.

### Miért a Git?

Nagy objektumok hatékony használata, gyorsasága és egyszerűsége miatt választottam én is a Git-et mint verziókövetőt. Kezelése könnyen és gyorsan elsajátítható. Nem szükséges rengeteg, nehéz bash parancsot megtanulni, hiszen akár már 4 parancs ismeretében és ezek helyes használatával tudjuk használni Git-et. Sőt a SmartGit program használatával akár ezt mind egy grafikus felületen keresztül is elvégezhetjük, így még bash parancsokra sincs szükségünk.

Manapság még olyan nagyobb projektek is a Git verziókövetést használják, mint: [Linux-rendszermag](https://hu.wikipedia.org/wiki/Linux-rendszermag), Qt, [VLC media player](https://hu.wikipedia.org/wiki/VLC_media_player), és az android platform, hogy csak az ismertebb említsem.

## MySQL

A MySQL egy többfelhasználós, többszálú, SQL-alapú relációs adatbázis-kezelőszerver. A világ egyik legnépszerűbb, nyílt forráskódú adatbázis szervere. Több platformon, párhuzamosan zajlik a fejlesztése. Olcsó, hatékony alternatívát jelent mind a programfejlesztők, mind a rendszergazdák számára.

A termék mindenki számára elérhető verziói 100%-ig a GPL (Gnu General Public License) licence alá esnek, ami annyit jelent, hogy ingyenesen használhatók és terjeszthetők. Nincs szükség licencekre a kliensek telepítésénél. Ez lehet az oka a sikerének is, mivel többször volt a MySQL az év adatbázisszervere (jelenleg [2. helyen áll](http://db-engines.com/en/ranking) az Oracle után és a Microsoft SQL Server előtt), de emellett még rengeteg sikert elkönyvelhet magánkan.

A MySQL gyakorlatilag bármilyen 32 bites (vagy nagyobb) operációs rendszeren használható: Windows, Unix, Linux, FreeBSD, Solaris, SCO, OS2, MacOS, HP-UX, AIX, stb. A megfelelő verziókat pedig a MySQL [hivatalos oldaláról](http://www.mysql.com) tölthetőek le. A különböző Linux disztribúciók beépítve tartalmazzák, még letölteni sem kell (legfeljebb újabb verzióra való frissítést).

Egyedi illesztőfelületekkel az adatbázis-kezelő elérhető C, C++, C#, Delphi, Java, Lisp, Perl, PHP és még sok más nyelvből.

A MySQL kezelhető parancssori eszközökkel(konzol segítségével), de létezik grafikus felületű adminisztrációs eszköz is (MySQL Workbench).

### Miért a MySQL?

A MySQL egy gyors, megbízható, könnyen használható és nagy teljesítményű relációs adatbázis-kezelő szerver. Legnagyobb előnye, amiért ezt választottam a szakdolgozatomhoz, hogy nyíltforráskódú, ingyenesen használható, valamint bármilyen operációs rendszeren alkalmazható. Egyedi illesztőfelületekkel elérhető az adatbázis a manapság használt programozási nyelvekből, így C#-ból is, amiben én is dolgozom.

## C# - Microsoft Visual Studio 2015

A C# (szí-sarp, neve az angol ’see-sharp’ szavakból ered, mely ’láss élesen’-t jelent magyarul) programozási nyelv a Microsoft által a .NET keretrendszer részeként kifejlesztett objektumorientált programozási nyelv. Nyílt szabványú, vagyis bárki készíthet C# fordítóprogramot. A fejlesztésénél szempont volt, hogy a meglévő programozási nyelvekből kiindulva, olyan nyelvet hozzanak létre, amely a többi nyelv pozitív tulajdonságait megtartja, a rosszakat pedig kijavítja, vagy elhagyja. Ennek következtében nagyon áttekinthető szintaktikája van, egyszerű, gyorsan tanulható.

Alapja a C nyelv szintaktikája, így sok programozó számára ismerős. Merített más, széles körben elterjedt és ismert nyelvekből, mint C++, Java, Delphi. Tisztán OOP nyelv, tehát Objektum Orientált programozást támogat. Ez azt jelenti, hogy minden függvényt és eljárást valamely osztályban kell elhelyezni (metódus).

### Microsoft Visual Studio

A Visual Studio a Microsoft több programozási nyelvet tartalmazó fejlesztőkörnyezete, amely az évek során egyre több új programnyelvvel bővült. Jelenleg a F#, C++, C# és Visual Basic programozási nyelveket, valamint az XML-t támogatja. A csomag része még a MASM (Microsoft Macro Assembler) is, ami részleges assembly támogatást biztosít.

### Miért C# és Microsoft Visual Studio?

A szakdolgozatomhoz elkészítendő program grafikus felületet igényel. Manapság szinte az összes program grafikus felületet használ a könnyebb és átláthatóbb kezelés érdekében.  
Az eddigi tapasztalataim és tanulmányaim során arra a megállapításra jutottam, hogy ha grafikus felületről van szó, akkor a C# és a Microsoft Visual Studio az, ami a leginkább a programozó keze alá dolgozik. Egyszerűen és gyorsan állíthatóak össze a grafikus felületek, valamint a C# nyelv is könnyedén megtanulható és rengeteg segédlet található hozzá az interneten. Azonban, ami talán az egyik legfontosabb a projektem esetében, hogy a MySQL adatbázis is egyszerűen elérhető a Visual Studioból.

## Trello

A Trello egy online projekt és feladatmenedzser program. Segít átlátni minden projektet, akár munkahelyen, akár otthon. Legyen az egy csapat irányítása, egy remek forgatókönyv megírása vagy egy egyszerű bevásárlólista összeállítása, a Trello kiváló segítség, hogy elvégezzük a dolgainkat és rendszerezettek maradjunk. Előnye, hogy online, így internet hozzáféréssel bárhonnan elérhető, akár számítógépről, akár mobilról, alkalmazás segítségével.

Egy adott projektre, feladatra táblát hozhatunk létre. A táblákhoz listákat rendelhetünk. Ezekhez a listához pedig kártyákat adhatunk hozzá, amin szerepel az elvégzendő feladat, felelős személyek, akikhez a feladat tartozik, csatolmányok, címkék és határidők. Egy kártyát a projekt előre haladtával mozgathatjuk egyik listáról a másikra, míg el nem készülünk a feladattal, amikor archiválhatjuk a kártyát.

### Mire jó a Trello?

* Táblákat készíthetünk, ami segítségével rendszerezhetjük, amin éppen dolgozunk.
* Használható egyedül vagy munkatársakkal, barátokkal, családtagokkal közösen.
* Elkészíthetjük különböző projektek munkamenetét.
* Kártyákhoz ellenőrzőlistát, „Teendők” listát adhatunk hozzá.
* Magunkhoz vagy a munkatársainkhoz rendelhetünk feladatokat.
* Kommentálhatjuk az egyes feladatokat.
* Fájlokat csatolhatunk a Google Drive-ból és Dropboxból.
* Fényképeket és videókat tölthetünk fel.

### Miért a Trello?

A Trello segítségével egyszerűen és átláthatóan tudjuk a menedzselni a projektünk. Egy letisztult és látványos online felületen, amit telefonról is elérhetünk. A kártyák listák közötti mozgatásával könnyedén számontarthatjuk, hogy hogy állunk az adott feladat elvégzésében.   
Címkéket használhatunk arra, hogy lássuk egy feladat mennyire fontos. Határidőt állíthatunk be, amelyet ha megközelítünk, vagy túlépünk figyelmeztet minket. Nagy segítséget nyújt, mind a szakdolgozatom, mind a hétköznapi életben az elvégzendő feladataim nyilvántartásában és az elvégzésük ütemezésében.

# Vehicle Routing Problem (VRP) ismertetése

Magyarul *jármű útválasztási problémát* jelent, ha szó szerint fordítjuk le.  
Azonban az irodalomban és az interneten leginkább angolul, Vehicle Routing Problem néven találkozhatunk vele.

A VRP egy kombinatorikus optimalizálási és egészértékű programozási probléma, ami az útvonalak optimális megtervezésével foglalkozik, ha adott egy járműparkunk és a kiszolgálni kívánt ügyfélcsoport.

Egy általános szóbeli meghatározása a VRP családjának a következő:

**Adott:** A szállítási kérelmek halmaza és a járműpark.

**A cél pedig**, hogy megtaláljuk a megoldást a következőre:

Határozzuk meg a járművek optimális útvonalát, hogy minden (vagy a lehető legtöbb) szállítási kérelmet teljesítsük a rendelkezésre álló gépkocsipark figyelembe vételével, a költségek minimalizálása mellett. Meghatározva azt, hogy melyik jármű, melyik feladatot, milyen sorrendben hajtsa végre.

Az optimális megoldás meghatározása NP-nehéz probléma. Ez azt jelenti, hogy az optimálisan megoldható problémák mérete korlátozott.

VRP-t elsőként George Dantzig és John Ramser jegyezte papírra 1959-ben. Az első algoritmus megközelítését írták le, amelyet benzin szállítás optimális útvonalának kiválasztására alkalmaztak.

## Általános leírás

A VRP-t leginkább a szállító cégek alkalmazzák. Használatára akkor kerül sor, mikor bizonyos **árucikkeket** szállítanak ki az **ügyfelek** részére egy vagy több **raktárból**. Meghatározott számú **szállító jármű** és az azokat vezető **sofőr** áll rendelkezésre, akik az **úthálózaton** viszik el a rakományt a célállomásokhoz.

Egy egyszerű példa segítségével szeretném elmagyarázni a fent leírtakat:

*Adott egy étterem, amely a városban az ügyfeleinek kocsival szállítja házhoz az ételt.   
Az ügyfelek halmaza, akikhez el kell jutni 1....n-ig terjedhetnek. Hozzájuk kell kézbesíteni a rendelést a korlátozott jármű flottánk segítségével. A járműveknek van egy maximális kapacitásuk, esetünkben, hogy maximálisan mennyi ételt szállíthatnak, és fogyasztásuk, hogy mennyi üzemanyagot fogyasztanak el a kiszállításhoz szükséges út során. Ezek a gépjárművek* *egy központi telephelyről (étterem) indulnak és ide is kell visszaérkezniük.   
A probléma, amelyre a megoldást keressük, hogy hogyan, milyen útvonalon tudjuk az összes ügyfelünket kiszolgálni időben, optimális erőforrás kihasználás mellett.*

Feladatunk pedig nem más, mint meghatározni ezen útvonalak halmazát, oly módon, hogy minden ügyfélhez eljussunk, az üzemeltetési korlátozások figyelembe vétele mellett, mindezt úgy, hogy a szállítási költségek minimálisak legyenek.

Érdemes megjegyezni, hogy az optimális útvonal kiválasztásánál fontos, hogy egy pontot csak egyszer érinthetünk.

## VRP modell

A VRP legáltalánosabb változata a Capacited Vehicle Routing Problem. Ezentúl VRP alatt ezt fogom érteni. A VRP modellje az alábbi paraméterekkel rendelkezik.

n az ügyfelek száma

K jelöli a kapacitását az egyes járműveknek

di jelöle az i ügyfél keresletét (ez azonos mértékegység mint a jármű kapacitása)

cij pedig az utazás költsége az i ügyféltől a j ügyfélig

Minden paraméter egy nem negatív egésznek tekinthető. Adott egy jármű flotta korlátozott kapacitással (K) és egy központi raktár (telephely). A telephely rendelkezik a 0. indexel, a kiszállításokat pedig 1-től n-ig indexeljük. A problémát pedig az jelenti, hogy miként tudjuk meghatározni az optimális útvonalat, ha a járműveink a telephelyről indulnak és ide is kell visszaérkezniük az út végén. Magát az útvonalat úgy kell megválasztanunk, hogy lehetőleg az összes ügyfélhez eljussunk, de csak pontosan egyszer érintve őket. Fontos szempont azonban, hogy az útvonal során a kliensek igényeinek összege nem haladhatja meg az járművek kapacitását.   
A cél pedig, hogy minimalizáljuk a teljes utazási költséget. Azonban ezt nem csak kiadás szempontjából közelíthetjük meg, lehetőség van több szempontból optimalizálni az útvonalakat, ez lehet távolság, vagy kevesebb gépkocsi felhasználás vagy egyéb szempont is.

Továbbiakban a költségminimalizálással fogunk foglalkozni, ez lesz az útvonal optimalizálásánál a fő szempont. Ennek oka egyszerű, ha a kiadásaink nézőpontjából igyekszünk kedvező útvonalat meghatározni, akkor az valószínűleg távolság szempontjából megfelelőnek mondható eredményt fogunk kapni. Hiszen a költségek akkor a minimálisak, ha az utazás során felhasznált üzemanyag is a lehető legkevesebb, ami csak akkor lehetséges, ha a megtett távolság és az útvonal is ideálisan van megválasztva.

A matematikai modellje szemléltethető egy gráf (N, A) segítségével. A gráfunk esetén az Njelenti az ügyfeleink (C) halmazát, ami 1-től n-ig terjedhet. Az őket összekötő (A)-val jelöljük, ez jelképezi a kapcsolatot az egyes csomópontok között. Magyarán szólva ezek az utak. Egyes ügyfelek közt több útvonalon is el tudunk jutni, de a feladat, hogy ezek közül a legkedvezőbbet találjuk meg.

Minden *i* és *j* pontpár (ügyfél) között létezik egy ív , amelynek költsége leírható Cij segítségével,melyet a **legrövidebb út** költségeként definiálunk *i* és *j* között. Feltételezzük, hogy az egyes költségek szimmetrikusak, tehát , valamint .

A gépjárműveinket - amelyeknek halmazát V-vel jelöljük - azonosnak tekintjük. A járművek kapacitása K, valamint di–vel jelöljük az i ügyfél igényét.

Ezeken felül létezik még egy döntési változó :

(5.1)

Matematikai képlete:

(5.2)

Azzal a kikötéssel, hogy:

(5.3)

(5.4)

(5.5)

(5.6)

(5.7)

A 5.3 –as egyenlet értelmében, minden ügyfélhez pontosan egy jármű rendelhetünk. Tehát ez biztosítja azt, hogy minden ügyfelet csak egyszer érintünk az útvonal megtervezése során. Pontosan egy ív léphet be, és pontosan egy ív hagyhatja el minden csúcsponthoz tartozó ügyfelet. A 5.4-es egyenletben a kapacitási korlátok vannak feltüntetve. Mely előírja, hogy az útvonalat úgy kell megtervezni, az ügyfelek igényeinek az összege minden útvonalon legfeljebb a jármű kapacitása vagy annál kisebb legyen. Tehát egy útvonalon minden ügyfelet kitudjunk szolgálni úgy, hogy a jármű teherbírását ne haladjuk meg.

A 5.5- és a 5.6-os egyenlet az áramlási korlátokat írja le. Ennek értelmében egy jármű pontosan egyser hagyhatja el a központi raktárat, valamint amennyi jármű elhagyja a telephelyet, ugyanannyinak kell visszaérkeznie is.

Létezik egy egyszerűbb változata is a modellnek, ahol a járművek száma konstans. Azonban, ha a járművek száma módosítható, mint itt, akkor a lehető legalacsonyabb költségeket érhetjük el.

Találkozhatunk egy másik korláttal is, amely a járművek számát aszerint határozza meg, hogy legalább mennyi szükséges belőlük ahhoz, hogy minden ügyfelet kitudjunk szolgálni a gépjárművek véges kapacitásának figyelembe vétele mellett.

## Definíció (legrövidebb út)

A legrövidebb út fogalma alatt a gráfelméletben egy minimális hosszúságú utat értünk egy gráf két különböző 𝑢 és 𝑣 csúcsa között. Abban az esetben, ha a gráfunk éleihez nem tartoznak súlyok, akkor ez egyet jelent egy olyan úttal a két csúcs, és között, amelyben a legkevesebb él szerepel. Ha a gráf élei súlyokkal vannak ellátva, akkor olyan útról beszélünk, amelynek élein szereplő súlyok összege minimális. Tehát ha adott egy 𝐺 = (𝑉, 𝐸) gráf a 𝑘(𝑓), 𝑓 ∈ 𝐸 élsúlyokkal, akkor

(5.8)

## VRP változatai

Számos variációja és ágazata van a Vehicle Routing Problem-nek. Közülük szeretném bemutatni a fontosabb, gyakrabban használt változatait, amelyek a mindennapokban is jelentős szerepet töltenek be a szállítással foglalkozó cégek körében:

* **Vehicle Routing Problem with Pickup and Delivery (VRPPD):**

Magyarul: *Jármű útválasztási probléma felvétellel és kézbesítéssel*.

Számos árut kell bizonyos felvevő helyekről, a kijelölt lerakati helyekre szállítanunk. Mindemelett a cél az, hogy megtaláljuk azt az optimális útvonalat, amelyen a járműveink gazdaságosan elérhetik ezeket a felvevő és lerakó állomásokat.

* **Vehicle Routing Problem with LIFO:**

**LIFO jelentése: Last In First Out**, rengeteg helyen használatos kifejezés, ami annyit tesz, hogy, amit utoljára betettünk azt vesszük ki elsőként. Az informatika világában legtöbbször a stack-ek adattárolása esetében találkozhatunk vele. Esetünkben ez annyit tesz, hogy azt a tételt szállítjuk ki elsőként a megadott szállítási helyre, amelyet legutoljára vettünk fel.

Maga a módszer hasonló, mint a VRPPD , kivéve, hogy a fent említett LIFO korlátozást figyelembe kell venni a szállításnál, és a gépjárművek bepakolásánál.   
Ez a rendszer csökkenti a be és kirakodások számát a szállítási helyeken, mert nem kell ideiglenesen eltávolítani azokat az elemeket, amik csak később kerülnek kiszállításra, hiszen az a szállítmány lesz legelöl, amit éppen ki kell szállítanunk.  
Ezzel a módszerrel a ki és bepakolásokkal elvesztegetett időt tudjuk megspórolni.

* **Vehicle Routing Problem with Time Windows (VRPTW):**

Jelentése: *Időablakos Jármű útválasztási probléma*  
A szállítási címek időablakkal rendelkeznek, amelyen belül a kiszállításoknak meg kell történniük.

* **Capacitated Vehicle Routing Problem: CVRP or CVRPTW:**

A járművek korlátozott teherbírással rendelkeznek. Így korlátozott a rakományoknak a száma, amelyeket képesek elszállítani.

Ezzel a megoldással limitálni lehet, hogy egy kiszolgáló jármű mennyi címhez tud eljutni, ezzel pedig az utazási időt lehet korlátozni. E megoldás akkor célszerű, ha a kiszolgálni kívánt ügyfeleink száma nem haladja meg a jármű kapacitását egy adott útvonalon. Valamint a klasszikus VRP nem veszi figyelembe az útvonalak meghatározásánál a járművek kapacitását, így olyan útvonalak keletkezhetnek, amelyek optimálisak, azonban a gépjárműveink kapacitását meghaladják. Az útvonalak megtervezésénél pedig fokozottan oda kell figyelni erre a tényre.

* **Vehicle Routing Problem with Multiple Trips (VRPMT):**

Ebben az esetben a járművek több mint egy útvonalat is bejárhatnak.   
Erre az esetre azért van szükségünk, mert a VRP alapértelmezetten nem veszi figyelembe az útvonalak meghatározásánál, hogy hány autóval rendelkezünk.   
Ennek következményeként az útvonalak száma meghaladhatja a járműink számát, így egy autóval több útvonalat is be kell járni annak érdekében, hogy minden ügyfél kiszolgálásra kerüljön. Azonban oda kell figyelni, hogy az egyes járművek ne lépjenek át egy meghatározott időtartamot, a munkaidőt. Valamint a gépkocsik kapacitása sem lépheti át az előre meghatározott értéket.

* **Open Vehicle Routing Problem (OVRP):**

Ilyen esetben nem előírás az, hogy a járművek visszatérjenek a kiindulási helyre az útvonal végén.

## Költség megtakarítás

A VRP-nek számos felhasználása van az iparban, előszeretettel alkalmazzák a költségek csökkentése érdekében. Így fontos kérdés, hogy mennyit lehet megspórolni ennek az alkalmazásával.

Valójában a számítógépes optimalizálási programok nagyságrendileg 5% megtakarítást jelenthetnek a cégeknek. Egy termék költségének pedig általában jelentősrészét, hozzávetőlegesen 10%-át teszi ki a szállítás, sőt az Európai GDP 10%-át a közlekedési ágazat adja.

# C#-ban történő megvalósítás

Mint már azt a használt technológiák fejezetében említettem a feladat megvalósítását C#-ban kezdtem el.

A kezdeti fő kihívást az jelentette, hogy a programomból el tudjam érni az adatbázisban tárolt adataimat majd ezeket kezelni tudjam. Létrehoztam az alap osztályokat az emberek, eszközök, csapatok, járművek és feladatok tárolására. Valamint egy osztályt, ami az adatbázisból történő beolvasásért felel. Ezen az osztályon keresztül férünk hozzá az adatbázishoz. A beolvasott adatokat kezelését a hozzátartozó osztály biztosítja, majd ezeket az osztályokat tároljuk el egy listába. Minden osztályhoz (eszköz, ember, jármű, feladat, csapat) tartozik egy lista, amely ezeket az osztályokat tárolja, így a beolvasás után a programunkban már könnyedén hozzá tudunk férni az adatokhoz a Gettereken keresztül. Ezzel biztosítva, hogy a tárolt adatokat csak lekérdezni tudjuk, módosítani nem, legfeljebb ott ahol ez szükséges.

Az adatbázis elérése, könnyebbnek bizonyult, mint amire számítottam, pár helyes sor megadásával már el is érhetjük a kívánt adatokat.

dbinfo = @"server=localhost;userid=root;password=jelszo;database=goeon\_db";

Egy string változóban adjuk meg az adatbázis elréséhez szükséges adatokat. Címét, ami jelen esetben localhost, mivel saját gépen vanaz adabázis. A felhasználó nevet (userid), a jelszót (password), valamint az adatbázis nevét (database) amiben a táblák helyezkednek el.

Ezt követően két változót kell definiálni, egyet ami a csatlakozásért felel, egyet pedig ami a beolvasásért.

private MySqlConnection conn;

private MySqlDataReader rdr;

Kezdetekben ezeket null paraméterrel látjuk el.

conn = new MySqlConnection(dbinfo);

//csatlakozás az adatbázishoz

conn.Open();

//összeköttetés létesítése

string stm = "SELECT \* FROM task\_table";

//az adatbázisunkon végrehajtandó utasítást tárolja

MySqlCommand cmd = new MySqlCommand(stm, conn);

rdr = cmd.ExecuteReader();

//utasítás végrehajtása

while (rdr.Read())

{

//beolvasás  
 //Az adatokat az rdr.GetString(oszlopszám) segítségével //tudjuk kinyerni  
//Nem csak stringet tudunk beolvasni, lehet intigert vagy

//egyéb más szokványos változót

}

Miután képes voltam eltárolni és kezelni az adatbázisból elért adatokat létrehoztam, 3 grafikus felületet, amelyeket a „Kiindulási alapötlet” fejezetben már ismertettem. Elkészítettem a főoldalt, az új feladat felvétele és a csapatok hozzárendelése képernyőket. Ezek működéseit implementáltam. Nem minden funkció került megvalsításra idő hiánya következtében, főként a csapatok hozzárendelése képernyő tekintetében igaz ez.

A szoftver képes az adatbázisból beolvasni az adatokat, majd kilistázni a csapatokat és a feladatokat külön-külön prioritás szerint. Feltudunk venni új feladatokat az adatbázisba. Hozzátudunk rendelni feladatokat a csapatokhoz. Igaz itt sok funkció még nem került implementálásra, de ez az elkövetkezendő időben pótlásra kerül.

# Jövőbeni tervek

A jövőre nézve a kitűzött cél minden hiányzó funkció, képernyő megvalósítása.

* A hiányzó grafikus felületek megvalósítása (Naptár, Belépő oldal).
* Az útvonal optimalizálás, távolság és idő számítás, VRP algoritmus beépítés
* Szoftver mentse az aktuális állapotát, a hozzárendeléseket és módosításokat
* Tökéletesítés, tesztelés

**Végső cél:** Egy beépített térkép implementálása, amelyen láthatóak az elvégzendő feladatok, továbbá a megtervezett optimális útvonal.

# Összegzés

A féléves kutatásom aköré összpontosult, hogy miként lehetne megvalósítani a feladat leírásban megfogalmazott elvárásokat. Mindezt a lehető leghatékonyabb módon. Egy olyan szoftveralkalmazást elkészíteni, mely segít a feladatok csapatokhoz rendelésében, figyelembe véve a csapatok kapacitását és képességeit.

Megismertem a hasonló feladatokat ellátó alkalmazásokat, amik elérhetőek jelenleg a piacon. A tapasztaltak alapján kiindulási alapötletet állítottam fel, majd megterveztem a saját szoftverem és annak funkcióit.

Ezt követően ki kellett választanom, hogy milyen technológiákat használjak fel a feladat megvalósításához. Információt kellett gyűjteni, hogy az egyes területeken, mint például az adatbáziskezelő rendszerek és program nyelvek közül melyek azok, amik a legpraktikusabbak. Majd megismertem a használatukat.

Ezek után az irdalom áttekintése került előtérbe. Részletesen körül jártam a Vehicle Routing Problem (VRP) témakörét, amely szorosan kapcsolódik az én feladatomhoz is. Megismertem mit jelent, mivel foglalkozik, hol és hogyan használják. Megtapasztalhattam, hogy ez milyen fontos szerepet tölt be a mai világban. Hiszen minden fuvarozással, szállítással foglalkozó cég arra törekszik, hogy optimális útvonalat válasszon meg a kiszolgálás során, amivel jelentős összegeket és időt képesek megtakarítani.

Hozzáláttam a saját szoftverem elkészítéséhez C#-ban. A megvalósítás objektum orientált alapelveket követ. Megalkottam a program vázát, amely képes az adatbázisból beolvasott adatok kezelésére.

Végül pedig ismertettem jövőbeni terveimet. A kitűzött cél, hogy minden hiányzó grafikus felület és funkció megvalósításra kerüljön. A program képes legyen az optimális útvonal megtervezésére, a szükséges számítások elvégzésére. A végső cél pedig egy beépített térkép implementálása az alkalmazásomba.

# Irodalomjegyzék

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, 2001
2. Toth P, Vigo D (2001) The Vehicle Routing Problem. SIAM, Philadelphia
3. Geir Hasle, Knut-Andreas Lie, Ewald Quak,; Kloster, O (2007). Geometric modelling, numerical simulation, and optimization applied mathematics at SINTEF
4. Paulo Toth, Daniele Vigo: Vehicle Routing: Problems, Methods, and Application, Second Edition
5. Wikipedia. (2016. Szeptember 20.) Letöltés dátuma: 2016. november 20, forrás: <https://en.wikipedia.org/wiki/Vehicle_routing_problem>
6. GSMtasks (2016. október 19.) Letöltés dátuma: 2016. november 20, forrás: <https://www.gsmtasks.com/>
7. Wikipedia (2016. november 01.) Letöltés dátuma: 2016. november 20, forrás: <https://en.wikipedia.org/wiki/Fleet_management>
8. Wikipedia (2016. november 01.) Letöltés dátuma: 2016. november 20, forrás: <https://en.wikipedia.org/wiki/Field_service_management>
9. Viamente (2016. október 19.) Letöltés dátuma: 2016. november 20, forrás: <https://www.viamente.com/>
10. Wikipedia (2016. október 23.) Letöltés dátuma: 2016. november 20, forrás: <https://hu.wikipedia.org/wiki/Git>
11. University of Bristol (2016. október 23.) Letöltés dátuma: 2016. november 20, forrás: <https://people.maths.bris.ac.uk/~mb13434/git/miagit.html>
12. Hugyák Tamás jegyzet (2016. október 23.) Letöltés dátuma: 2016. november 20, forrás: <http://desoft.hu/oktatas/git/tartalom>
13. Wikipedia (2016. október 24.) Letöltés dátuma: 2016. november 20, forrás: <https://hu.wikipedia.org/wiki/MySQL>
14. (2016. október 24.) Letöltés dátuma: 2016. november 20, forrás: <http://softwareonline.animare.hu/cikk.aspx?id=3216>
15. Áslaug Sóley Bjarnadóttir: Solving the Vehicle Routing Problem with Genetic Algorithms (2004). (2016. november 02.) Letöltés dátuma: 2016. november 20, forrás: <http://etd.dtu.dk/thesis/154736/imm3183.pdf>
16. Wikipedia (2016. október 25.) Letöltés dátuma: 2016. november 20, forrás: <https://hu.wikipedia.org/wiki/C_Sharp>
17. Wikipedia (2016. október 25.) Letöltés dátuma: 2016. november 20, forrás: <https://hu.wikipedia.org/wiki/Microsoft_Visual_Studio>
18. Sulinet (2016. október 25.) Letöltés dátuma: 2016 november 20, forrás: <http://tudasbazis.sulinet.hu/hu/informatika/informatika/informatika-9-12-evfolyam/alapok-a-nyelv-tortenete-jellemzoi-a-kornyezet-hasznalata/c-programozasi-nyelv>
19. Google play (2016. október 25.) Letöltés dátuma: 2016 november 20, forrás: <https://play.google.com/store/apps/details?id=com.trello&hl=hu>

# Ábrajegyzék

[3.1. ábra: A Belépő oldal képernyő terve 10](#_Toc467439148)

[3.2. ábra: Főoldal képernyőterve 11](#_Toc467439149)

[3.3. ábra: Új feladat felvétele oldal képernyőterve 11](#_Toc467439150)

[3.4. ábra: Csapatok hozzárendelése oldal képernyőterve 12](#_Toc467439151)

[3.5. ábra: Az adatbázis felépítése 13](file:///E:\SZAKDOLGOZAT\Dokumentumok\Udvardi%20Dávid%20-%20Szakdolgozat%20féléves%20beszámoló%20v0.docx#_Toc467439152)