ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

ROMAN PECHO

Systém pre podporu rozhodovania v prepravnej spoločnosti

Vedúci práce: Ing. Marek Kvet, PhD.

Registračné číslo: 54/2017

Žilina, 2018

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

ŠTUDIJNÝ ODBOR: INFORMATIKA

ROMAN PECHO

Systém pre podporu rozhodovania v prepravnej spoločnosti

Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

Katedra informatiky

Žilina, 2018

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY BAKALÁRSKEJ PRÁCE.

Študijný odbor : Informatika

Meno a priezvisko
Roman Pecho

Osobné číslo
555965

Názov práce v slovenskom aj anglickom jazyku

Systém pre podporu rozhodovania v prepravnej spoločnosti

Decision support system for a parcel service

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ bakalárskej práce:
Cieľom bakalárskej práce je vytvoriť interaktívny informačný systém prepravnej spoločnosti.

Obsah:
Obsah práce možno zhrnúť nasledovne:
1. Analýza existujúcich softvérových riešení
2. Zber požiadaviek na informačný systém
3. Návrh údajových štruktúr a algoritmov pre riadenie logistiky a iných procesov prepravnej spoločnosti
4. Implementácia navrhnutého riešenia s podporou grafického rozhrania na mapových podkladoch
5. Dokumentácia formou UML

Meno a pracovisko vedúceho BP: Ing. Marek Kvet, PhD., KI, ŽU
Meno a pracovisko tutora BP:

vedúci katedry
(dátum a podpis)

Zadanie zaregistrované dňa 16. 10. 2017 pod číslom 54/2017 podpis _____

Čestné vyhlásenie

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí. Všetky zdroje použité v bakalárskej práci som uviedol v súlade s predpismi.

Súhlasím so zverejnením práce a jej výsledkov.

_____

Meno Priezvisko

V Žiline, dňa 20.4.2018.

Poďakovanie

Touto cestou by som sa chcel poďakovať vedúcemu bakalárskej práce Ing. Marekovi Kvetovi , PhD. za odbornú pomoc, pripomienky a usmerňovanie pri tvorbe práce.

ABSTRAKT V ŠTÁTNOM JAZYKU

PECHO, Roman : *Systém pre podporu rozhodovania v prepravnej spoločnosti.* [Bakalárska práca] – Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra informatiky – Vedúci: Ing. Marek Kvet, PhD. – stupeň odbornej kvalifikácie: Bakalár v odbore Informatika. Žilina: FRI ŽU v Žiline, 2018. – Počet strán 47 strán.

Cieľom bakalárskej práce je vytvoriť interaktívny informačný systém pre prepravnú spoločnosť. Informačný systém poskytuje grafický systém v ktorom je možné riadiť sklad v prepravnej spoločnosti s možnosťami , ako prijímať objednávky na sklad , zobrazovať tovary ktoré sa aktuálne nachádzajú na sklade atď. Ďalej informačný systém vie vygenerovať optimálnu trasu pre jednotlivé vozidlá a rozloženie tovarov do vozidiel.

Kľúčové slová: prepravná spoločnosť, mapy, problém okružných jázd, skladové hospodárstvo

ABSTRAKT V CUDZOM JAZYKU

PECHO, Roman : *Decision support system for a parcel service.* [Bachelor's thesis] – University of Žilina, Faculty of Management Science and Infromatics, Department of informatics – Supervisor: Ing. Marek Kvet, PhD. – Bachelor's degree in informatics. Žilina: FRI ŽU in Žilina, 2018. – Number of pages 47 pages.

Aim of the work is to create interactive information system for parcel service. Information system provides graphical system in which is possible to manage stock in parcel service with possibilities such as, accept orders for stock, show up goods which are currently in the stock etc. Information system can generate optimal route for each vehicles and arrangement goods in the vehicles.

Key words: parcel service, maps, vehicle routing problem, warehousing

Obsah

Zoznam obrázkov

Úvod

V dnešnej modernej a uponáhľanej dobe sa ľudia snažia využiť svoj čas čo najefektívnejšie. Využívajú na to aj internet a informačné technológie. Ak majú doma počítač a pripojenie k internetu, menia aj spôsob nakupovanie tovaru. Nakupovanie cez internet je jednoduchou výsadou moderného človeka. Na slovenskom internete sa objavuje čoraz viac nových elektronických obchodov a virtuálnych obchodných domov. Ich sortiment je rôznorodý, ovplyvnený dopytom zákazníkov. Z prieskumov vyplýva, že počet nakupovaných položiek prostredníctvom internetu sa neustále zvyšuje. Výhodou takéhoto nákupu je, že nákup si môžu užiť z pohodlia svojho domova, mimo bežných otváracích hodín. Zákazník sa do predajne môže prihlásiť sedem dní v týždni, v ktorúkoľvek hodinu. Môže si v pokoji svojej obývačky vybrať tovar a prostredníctvom banky cez účet zaplatiť. Tovar si môže vytriediť podľa množstva kategórií, preto má jednoduchšiu úlohu pri rozhodovaní medzi niekoľkými tovarmi naraz. Internetové obchody poskytujú možnosť prečítať si hodnotenie tovaru inými kupujúcimi a porovnať ich s podobnými produktami. Platbu zákazník realizuje buď pri dodávke tovaru alebo prevodom zo svojho konta v banke. Vyspelejší predajcovia sú priamo napojení na niektorú banku, ktorá zabezpečí bezpečnú finančnú transakciu prostredníctvom Internetu. Navyše bonusom takéhoto nákupu je aj to, že nákup si môžu nechať doručiť priamo domov a presne vtedy, kedy im to vyhovuje. Na dovoz takého tovaru sa využívajú prepravné spoločnosti. Prepravné spoločnosti sa v súčasnosti tešia veľkej obľube. Logistika je proces plánovania, uskutočnenia a kontroly postupov na úspornú a efektívnu prepravu a uskladnenie tovaru, do ktorej zahŕňame aj služby a informácie spojené s miestom pôvodu do miesta spotreby, aby sa plne vyhovelo požiadavkám zákazníka. Zahŕňa zároveň aj prichádzajúce, odchádzajúce, vonkajšie a vnútorné pohyby. Výstupná logistika je proces, ktorý súvisí so skladovaním a pohybom konečného výrobku a ktorý je spojený s informáciami z konca výrobnej linky až ku koncovému užívateľovi. Prepravné spoločnosti tak ponúkajú prepravné a logistický výhodné riešenia pre svojich klientov, v zhode s ich požiadavkami. Poskytujú im spoľahlivé a nákladovo efektívne riešenia prepravy tovaru. Zákazníci si na prepravných spoločnostiach cenia ich vysokú kvalitu služieb za výhodnú cenu. Tie využívajú výber najlepšieho variantu prepravnej trasy, tovar prevážajú vozidlami s alternatívnym pohonom a majú energeticky úsporné sklady. Myslia teda na dopady na životné prostredie aj pri doprave a skladovaní tovaru.

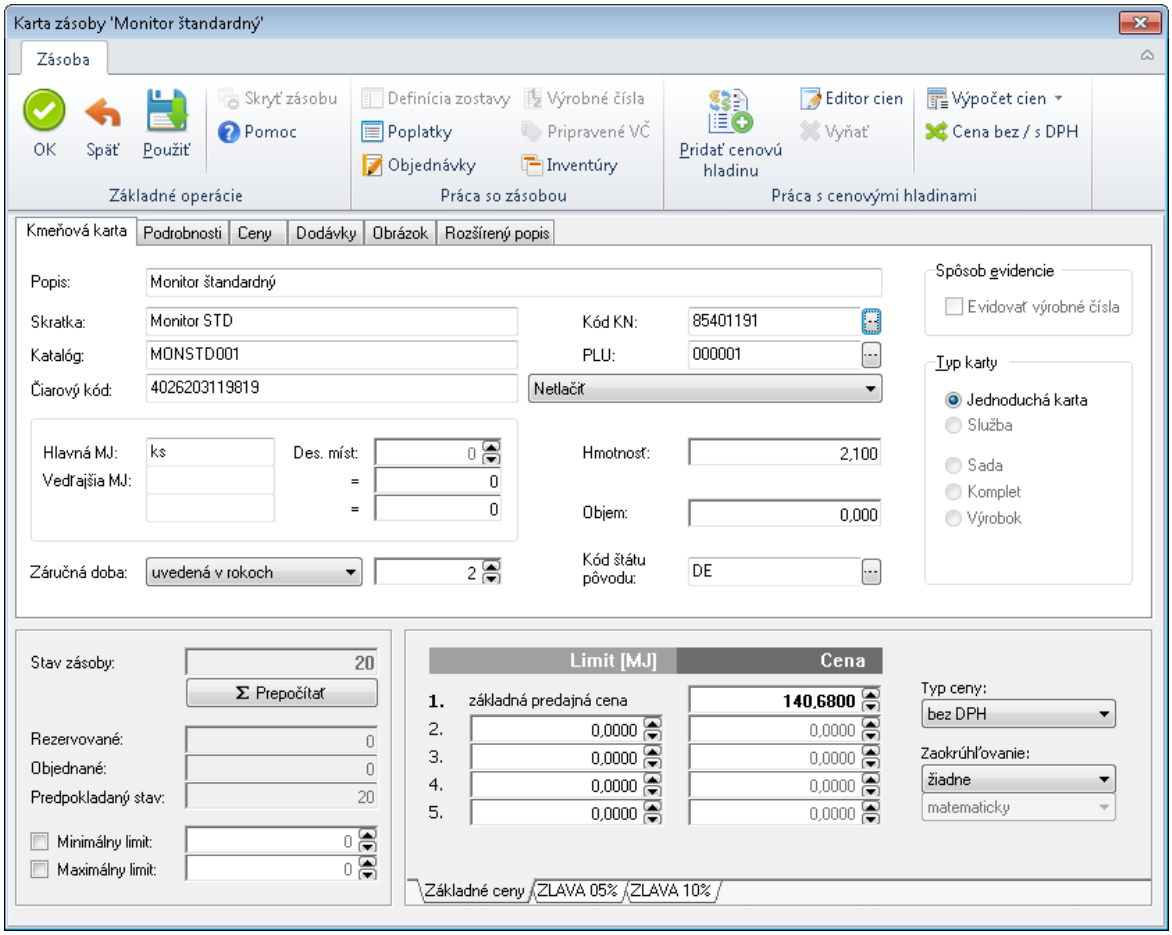
Výhodou pre klienta je aj skutočnosť, že prepravné spoločnosti dokážu previezť zásielky najrôznejšieho druhu a veľkosti. Vďaka modernému spracovaniu množstva dát, dokáže presne dopravná spoločnosť určiť, presný stav zásielky. Od prijatia objednávky do systému, cez jej ďalšie spracovanie až po doručenie konečnému zákazníkovi. Hlavnou činnosťou týchto spoločností je skladovanie a distribúcia tovaru. Tovar sa skladuje v žiadúcej kvalite a dodáva sa v určenom čase v súlade s prianiami a požiadavkami zákazníka. Tento stanovený cieľ dosahujú pravidelnou a systematickou kontrolou jednotlivých operačných oblastí. Prepravné spoločnosti tvoria významnú úlohu v zlepšovaní životov zákazníkov. Ako vieme, ziskový obchod a prosperita sú bezprostredne prepojené. Prosperujúci a globálny obchod je úzko spojený s logistikou, čo znamená, že prepravná spoločnosť prináša spoločnosti skutočné hodnoty.

Táto bakalárska práca sa zaoberá problematikou skladového hospodárstva, ako pridávanie, prijímanie objednávok, prehľad položiek na sklade atď. a taktiež zvolenie optimálnej trasy , pre rozvoz prijatých tovarov, tak aby náklady na rozvoz boli čo najmenšie a bolo rozvezených čo najviac tovarov s najvyššou prioritou. Nachádza sa v nej Vehicle Routing problém konkrétne Capacitated Vehicle Routing Problem. Aplikácia je vytvorená v jazyku C#.

1 Dostupné riešenia

1.1 Program Money S3

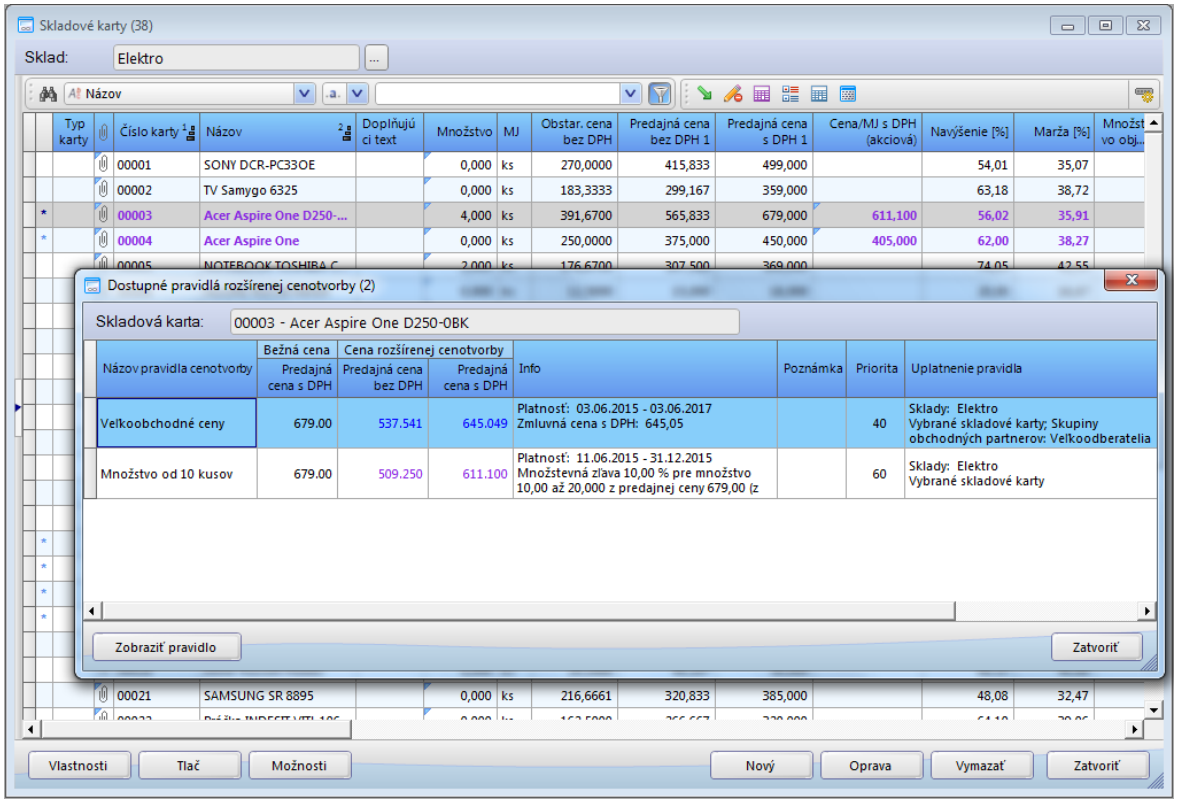
Program Money S3 je účtovný program , ktorý umožňuje vedenie neobmedzeného množstva skladov. Prehliadať , tlačiť a upravovať zoznam všetkých vytvorených dokladov. Umožňuje vykonávať hromadnú zmenu cien celej vybranej skupiny zásob atď.



Obrázok 1. Program Money S3

1.2 Program Oberon

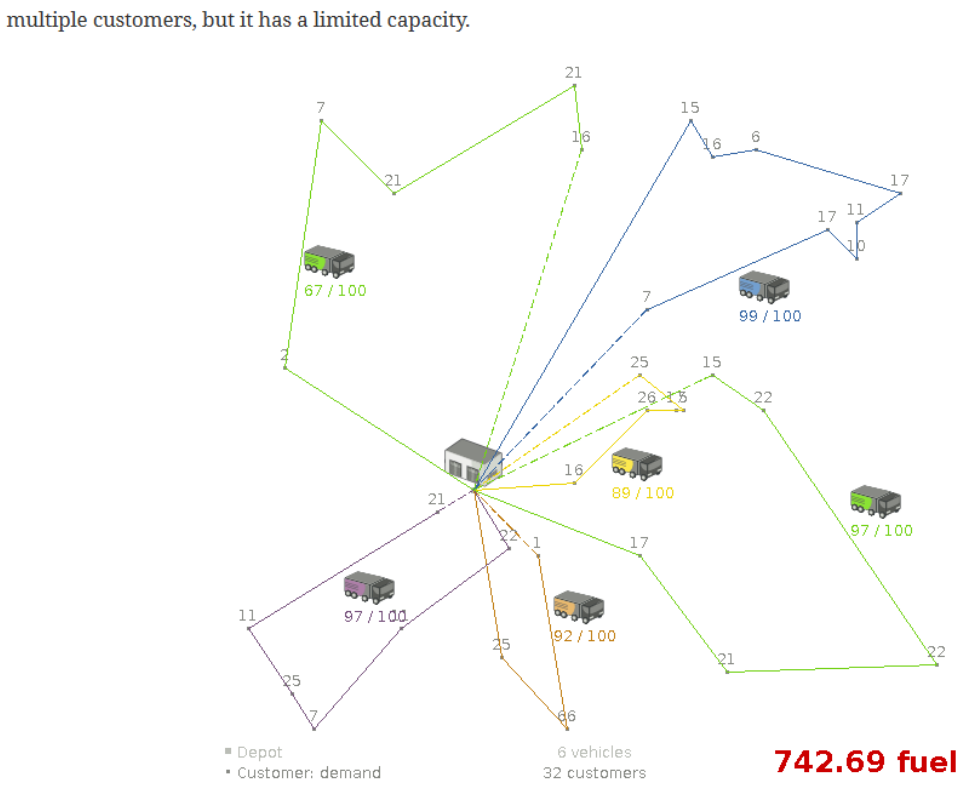
Je komplexný účtovný systém, umožňuje mať nekonečný počet skladov, ktoré môžu mať medzi sebou väzby alebo sú nezávislé, automatickú tvorbu objednávok podľa evidovaných prijatých objednávok a aktuálneho stavu zásob, výpočet predajných cien atď.



Obrázok 2. Program Oberon

1.3 Program OptaPlanner

Je to Open Source solver na optimalizovanie Vehicle Routing Problému , Traveling Salesman problému a podobných problémov. Využíva heuristiky a metheuristky na riešenie problému ako Tabu Search , Simulated Annealing a Late Acceptance



Obrázok 3. Program OptaPlanner

1.4 Zhodnotenie dostupných riešení

Jednotlivé riešenie buď riešia problematiku skladového hospodárstva , alebo optimalizáciu trás pre jednotlivé vozidlá. Nenašiel som riešenie , ktoré by sa zaoberalo obidvoma problémami súčasne. To je dôvod prečo som sa rozhodol vytvoriť softvér , ktorý by bol univerzálnym nástrojom pre firmy a nebolo by potrebné zakúpiť dva softvéry a stačil by iba jeden

2 Analýza aplikácie

Hlavným cieľom projektu je vytvoriť softvér v ktorom bude možné robiť základné funkcie skladového hospodárstva a následné vygenerovanie optimálnych trás pre jednotlivé vozidlá a ich zobrazenie na mapovej podpore.

2.1 Analýza databázy

Všetky dáta o jednotlivých tovaroch, objednávkach , vozidlách atď. by mali byť uložené v databáze, pretože databáza poskytuje rýchle vyhľadávanie a nie je nutné mať uložené dáta lokálne, taktiež môže byť aplikácia spustená na rôznych zariadeniach a dáta budú konzistentné pre každú spustenú aplikáciu.

2.2 Analýza algoritmu pre optimalizáciu trás

Optimalizačná úloha pri tomto probléme je veľmi náročná. Ide vlastne o Vehicle Routing Problem. Vehicle Routing Problem je súhrnný názov sady problémov, kde pomocou flotily vozidiel musíme obslúžiť zákazníkov a ich požiadavky. Cieľom je vytvoriť optimálne trasy pre flotilu vozidiel, ktoré vychádzajú z depa, tak aby náklady na rozvoz boli minimálne a všetky požiadavky zákazníkov boli uspokojené. Základná definícia problému je veľmi všeobecná a úloha môže nadobúdať rôzne typy obmedzujúcich, alebo doplňujúcich podmienok. Práca sa zaoberá konkrétnym typom danej úlohy a to je Capacitated Vehicle Routing Problem , ktorý má obmedzujúcu podmienku a to kapacitu jednotlivých vozidiel.

2.2.1 Clarkeova-Wrightova metóda

Ide o jednu z najznámejších heuristických metód, ktorá sa zaoberá riešením VRP. Zverejnili ju autori G. Clarke a J. W. Wright v roku 1964. Postup spočíva v každej iterácii metódy sú podľa istého kritéria vybraté dve možné trasy (V_0 - V_i - V_0) a (V_0 - V_j -V_0) a tieto trasy sú spojene do tzv. združenej trasy (V_0 - V_i – V_j - V_0). Dve trasy môžu byť združené len ak vzniká trasa, ktorá bude vyhovovať podmienkam prípustnosti riešenia (1) a (2), čo znamená, že súčet záťaže združených tras nesmie prekročiť kapacitu vozidla K. Výhodnosť, alebo nevýhodnosť združenia dvoch trás je určená úsporou, ktorá ich združením vznikne. Túto úsporu meriame tzv. výhodnostným koeficientom z_{ij} podľa vzťahu $z_{ij} = (d_{0i} + d_{0j} - d_{ij})$. Kde d_{0i} , d_{0j} a d_{ij} označujú dĺžky hrán (V_0 ,V_i) (V_0 ,V_j) a (V_i , V_j). Hodnota z_{ij} potom vyjadruje rozdiel medzi súčtom dĺžok tras (V_0 - V_i - V_0) a (V_0 - V_j -V_0) a dĺžku zlúčenej trasy (V_0 - V_i

– V_j - V_0). Metóda združuje v každej iterácii postupu tie dva uzle, ktoré majú najväčší výhodnostný koeficient z_{ij}, pokiaľ je možné s ohľadom na prípustnosť toto združenie spraviť.

2.3 Analýza funkčných požiadaviek

Jednou s hlavných funkcii aplikácie by mal byť prehľad o jednotlivých tovaroch , vozidlách objednávkach a podobne. To by malo byť prehľadné a malo by byť možné zobrazovať dané položky podľa určitých kritérií ako sú napríklad tovary, ktoré sú prijaté na sklad , alebo tovary ktoré sú už doručené a podobne. Používateľ by mal byť schopný si zobraziť tovary ktoré sa aktuálne nachádzajú v konkrétnom vozidle, alebo tovary ktoré patria ku konkrétnej objednávke.

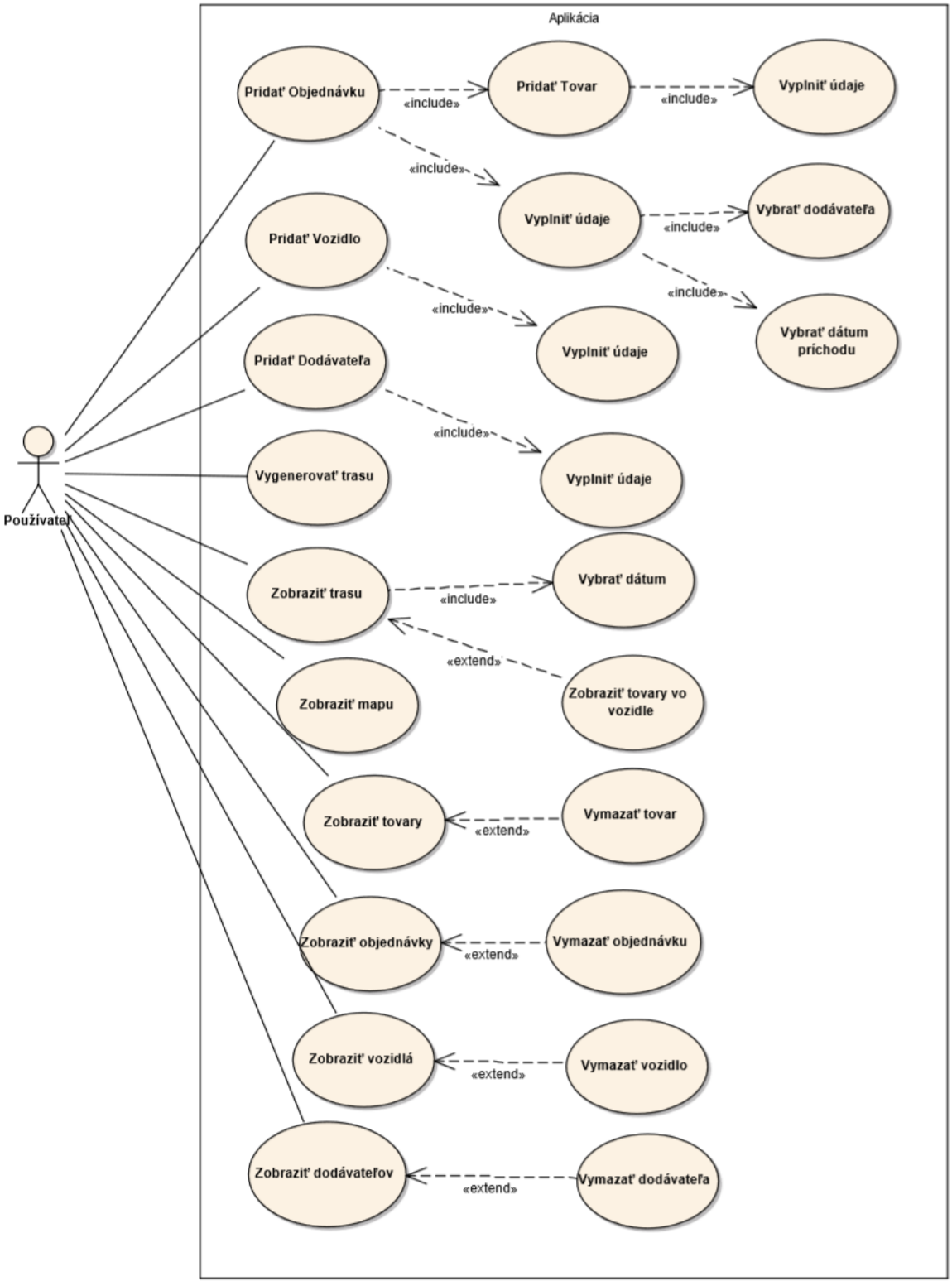
Ďalšou funkciou by malo byť pridanie nového tovaru. Tovar musí byť priradený k niektorej objednávke, nebude možné pridať tovar už do vytvorenej objednávky. Tovar bude možné pridávať iba do novej objednávky , ktorá ešte nie je vytvorená. Tovar sa nebude dať pridať dokým nebudú vyplnené všetky atribúty.

Aplikácia by taktiež mala byť schopná pridávať nové objednávky. Ako náhle bude tovar pridaný do objednávky , nebude možné meniť jeho atribúty a ani ho odstrániť z objednávky. Atribúty ako plánovaný dátum príchodu na sklad a dodávateľ sa tovaru priradí z objednávky. Ak sa zmení dátum alebo dodávateľ, automaticky sa zmenia tieto atribúty aj pre všetky tovary , ktoré budú vytvorené pre danú objednávku.

Nesmie chýbať pridávanie vozidiel a dodávateľov.

Jednou s hlavných funkcii by mala byť vytvorenie trás pre dané vozidlá. Trasa sa bude vytvárať z tovarov ktoré sú aktuálne prijaté na sklade. Tovary sa budú načítavať z databázy, pokiaľ bude objem dovtedy načítaných tovarov menší ako prepravný objem všetkých dostupných vozidiel. Po načítaní tovarov sa začnú získavať potrebné informácie pre vytvorenie trás. Po načítaní potrebných informácii sa spustí algoritmus pre vytvorenie optimálnych trás. Ak sa nebude dať vytvoriť trasy pre všetky načítané tovary, trasy sa vytvoria iba pre tie tovary , pre ktoré to bude možné. Po úspešnom vytvorení trás sa trasy zobrazia na mape a zobrazí sa okno s vozidlami. Bude možné si prezerať tovary, ktoré sa nachádzajú v danom vozidle. Pre tovary bude určené poradie v ktorom majú byť doručené. Ďalej bude možné označiť tovary ako doručené.

Bude možné si rozkríknuť požadovanú objednávku a označiť tovary, ktoré bude chcieť používateľ pridať na sklad. Akonáhle bude tovar pridaný do skladu, nebude možné tovar označiť ako nedoručený.

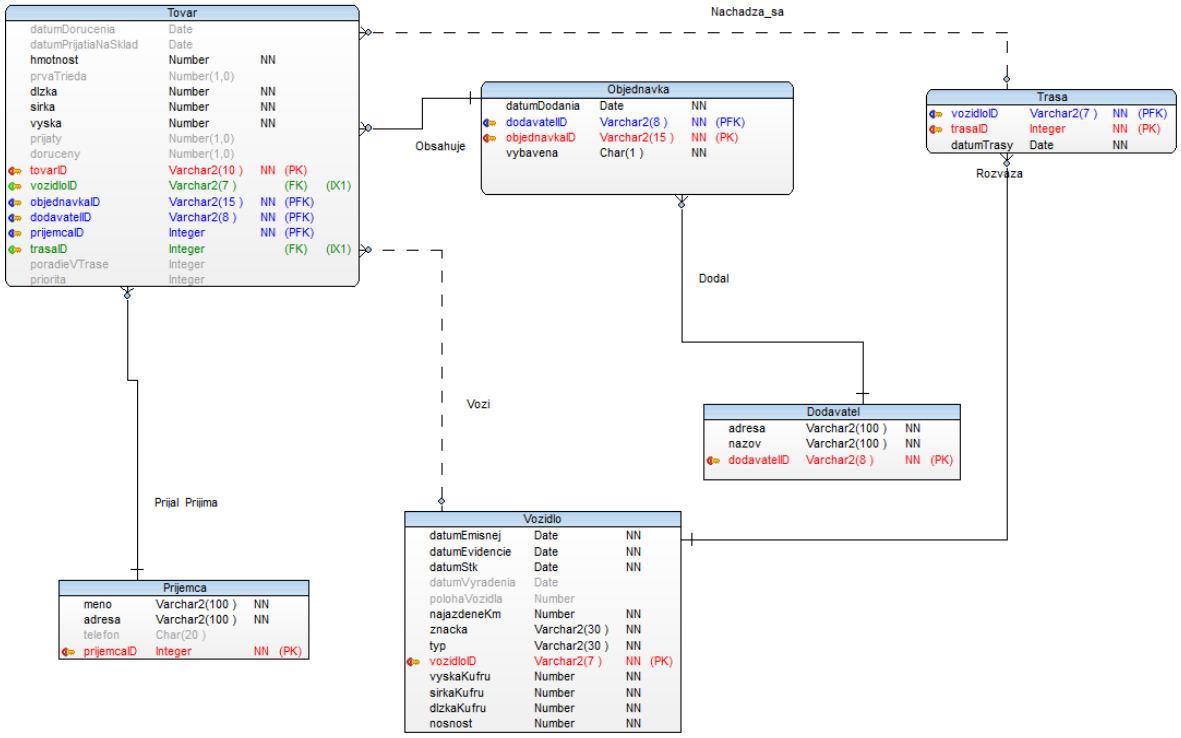


Obrázok 4. Diagram prípadov použitia

3 Návrh databázy

Databáza je implementovaná v systéme Oracle. Oracle bol zvolený kvôli jednoduchému prístupu a možnosti ľahkej podpory. Databáza je implementovaná na momentálne najnovšej verzii Oracle Databázy a to je verzia 12c.

3.1 Dátový model databázy



Obrázok 5. Dátový model databázy

3.2 Popis entít

3.2.1 Entita Dodavatel

Je to entita , ktorá predstavuje daných registrovaných dodávateľov, od ktorý daná firma prijíma objednávky. Nemôže byť vytvorená objednávka bez toho , aby sa dodávateľ nenachádzal v databáze. Ako primárny kľúč je **dodavatelID** , čo predstavuje IČO daného dodávateľa.

3.2.2 Entita Objednavka

Táto entita predstavuje jednotlivé objednávky , ktoré boli registrované do systému. Ako primárny kľúč je **objednavkaID** , ktorá predstavuje jedinečné 15 miestne číslo, číslo objednávky. Obsahuje cudzí kľúč **dodavatelID** , ktorý označuje dodávateľ, ktorý požaduje prepravu danej objednávky. Atribút **vybavena** predstavuje stav objednávky , ktorý môže byť buď **V**, **N**, **C**, kde **V** predstavuje objednávku , ktorá je už vybavená, to znamená všetky tovary , ktoré mali byť prijaté na sklad z danej objednávky už boli prijaté. **N** predstavuje objednávku , ktorá nemá ešte ani jeden tovar prijatý na sklad. **C** predstavuje objednávku, ktorá má iba niektoré tovary prijaté na sklad , ale nie všetky , čiže objednávka je iba čiastočne vybavená.

3.2.3 Entita Prijemca

Predstavuje jednotlivých príjemcov , ktorým má byť daný tovar doručený. Ako primárny kľúč má **prijemcaID**, ktorý predstavuje číslo príjemcu v databáze. Keďže nevieme o príjemcoch rodné číslo, alebo IČO ako sa jedná o nejakú firmu, alebo nejaký identifikátor, ktorý je u každého príjemcu jedinečný , musíme mať ako primárny kľúč iba číslo, ktoré sa inkriminuje pri každom pridaní nového príjemcu. O to sa stará trigger `AutoIncrementPrijemca` , ktorý zo sekvencie `prijemcaID_sequence`, zoberie číslo a priradí ho danému príjemcovi pri jeho pridávaní do databázy.

3.2.4 Entita Vozidlo

Nasledovná entita predstavuje jednotlivé registrované vozidla danej spoločnosti. Ako primárny kľúč je použité **vozidloID**, ktoré je vlastne štátna poznávacia značka pre dané vozidlo. Obsahuje základné informácie o vozidle ako šírka, výska a dĺžka kufru, najazdné kilometre, dátum evidencie atď.

3.2.5 Entita Tovar

Entita Tovar reprezentuje jednotlivé tovary v databáze. Obsahuje jednotlivé informácie o danom tovare. Primárny kľúč je **tovarID** , ktorý reprezentuje 10 miestne číslo, čo predstavuje číslo tovaru. Obsahuje primárne cudzie kľúče **vozidloID** a **trasaID** , ktoré určujú v ktorom vozidle a v ktorej trase bol daný tovar rozvážaný. Taktiež obsahuje cudzie kľúče a to **objednavkaID**, **dodavatelID**, **prijemcaID.** Tie identifikujú do ktorej objednávky patrí daný tovar, ktorý dodávateľ daný tovar dodal a ku ktorému príjemcovi sa daný tovar ma doviesť. Atribút **poradieVTrase** určuje , aké poradie v trase mal daný tovar. Atribút **priorita** určuje prioritu tovaru , priorita je vlastne číslo, ktoré je upravené vždy pri spustení

aplikácie a generuje sa ako **priorita** = (CURRENT_DATE - **datumPrijatiaNaSKlad**), kde **datumPrijatiaNaSKlad** je dátum kedy bol tovar prijatý do skladu a ak tovar je označený ako prvá trieda, to znamená **prvaTrieda** = **1** tak sa priráta hodnota **100000**. Čim vyššie číslo tak tým vyššia priorita.

3.2.6 Entita Trasa

Je to jednoduchá entita , ktorá iba spája entitu **Tovar** a **Vozidlo**, kde určuje , ktoré vozidlo rozvážalo aký tovar a v ktorý dátum. Ako primárny kľúč je použitý **trasaID**, ktorý rovnako ako pri entite **Prijemca**, je len číslo , ktoré je inkrementované vždy, keď sa pridá nová trasa. O to sa stará trigger `priTrasy` , ktorý číslo čerpá zo sekvencie `trasaID_sequence`

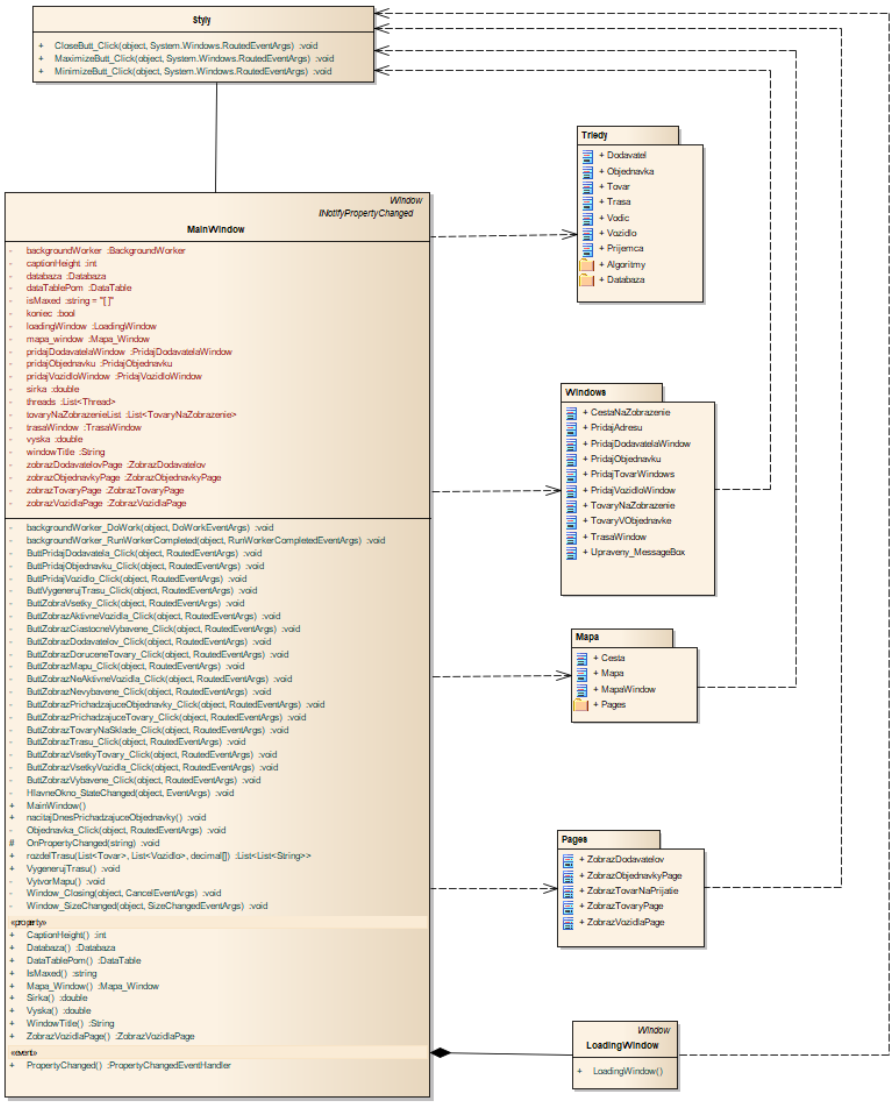
4 Logická vrstva

Keďže aplikácia bude programovaná v jazyku **C#** ako **WPF** projekt, niektoré triedy budú okná ktoré sa budú skladať z funkčnej časti , ktorá je písaná v jazyku **C#** a vizuálnej časti , ktorá je písaná v jazyku **XAML**. Nie všetky triedy budú pôsobiť ako okná , niektoré triedy vystupujú ako klasické triedy v jazyku **C#**.

4.1 Diagramy Tried

Samotná aplikácia je veľmi rozsiahla a bolo veľmi neprehľadné , keby bol vytvorený iba jeden diagram tried, preto sú vytvorené diagramy podľa namespace-ov v aplikácii.

4.1.1 Diagram tried hlavnej aplikácie



Obrázok 6 Diagram tried hlavnej aplikácie

Diagram obsahuje triedy **MainWindow**, **Styly** a **LoadingWindow** a následne ukazuje vzťahy medzi jednotlivými zložkami programu ako sú **Mapa**, **Pages**, **Windows** a **Triedy**, čo je označenie namespace-ov aplikácie.

4.1.1.1 Trieda MainWindow

Slúži ako hlavné vlákno na ovládanie aplikácie. Implementuje interface `InotifyPropertyChanged`, ktorý slúži na prepojenie medzi **XAML** a **C#** a zapríčiňuje zmenu určitých hodnôt objektov, ktoré sa nachádzajú v **XAML** a menia sa dynamicky z kódu. V konštruktory sa inicializujú hlavné komponenty pre zobrazenie samotného okna. Keďže je to hlavné vlákno a okno aplikácie, tak vlastní všetky zdieľané inštancie tried. Vytvára sa v ňom inštancia triedy **Databaza** , ktorá zabezpečuje spojenie s databázou. Vytvárajú sa v ňom aj ďalšie okná a stránky potrebné pre plynulý chod aplikácie. Po vytvorení inštancie triedy **Databaza** sa zavolá metóda `pripoj()` , ktorá slúži na pripojenie k databáze. V konštruktory sa taktiež inicializuje **BackGroundWorker** , ktorý slúži na zobrazenie animácie načítavania a informačných titulkov pri inicializácii aplikácie. Po úspešnom pripojení k databáze sa načítajú objednávky, ktorých plánovaný dátum príchodu sa zhoduje s aktuálnym dátum v zariadení pomocou metódy `nacitajDnesPrichadzajuceObjednavky()`. Vytvorí sa nové vlákno, ktoré obsahuje inštanciu triedy Mapa.

4.1.1.1.1 Metóda VygenerujTrasu()

Táto metóda slúži na vygenerovanie optimálnych trás pre jednotlivé vozidlá. Najprv sa načítajú vozidlá z databázy , ktoré nemajú žiadnu polohu, číže **polohaVozidla** v databáze je **null**, to znamená, že vozidlo nie je na žiadnej už vytvorenej trase a dá sa použiť na rozvoz. Následne sa pomocou príkazu:

```
select tovarID, AdresaPrijemcu ,sirka, vyska, dlzka, hmotnost, Decode(Prijaty, null, 'NIE', 'ÁNO') Prijaty, Decode(Doruceny, null, 'NIE', 'ÁNO') Doruceny, Decode(PrvaTrieda, 0, 'NIE', 1, 'ÁNO') PrvaTrieda, MenoPrijemcu, NazovDodavatela, PlanovanyDatum from(select t.*, prijemca.Meno as MenoPrijemcu, dodavatel.NAZOV as NazovDodavatela, datumDorucenia, DatumPrijatiaNaSklad, objednavka.DATUMDODANIA as PlanovanyDatum, prijemca.ADRESA as AdresaPrijemcu, sum(t.sirka * t.vyska * t.dlzka) over(order by( (priorita / t.sirka * t.vyska * t.dlzka)) desc) as credit_sum from tovar t join prijemca on (t.prijemcaID = prijemca.prijemcaID) join dodavatel on (t.dodavatelID = dodavatel.dodavatelID) join objednavka on (t.objednavkaID = objednavka.objednavkaID) where prijaty = 1 and doruceny is null) where credit_sum <= (select SUM(VYSKAKUFRU * SIRKAKUFRU * DLZKAKUFRU) from vozidlo where polohavozidla is null)  order by ((priorita / sirka * vyska * dlzka)) desc
```

Kde sa vyberajú tovary podľa priority, ktorá je tvorená odčítania dnešného dátumu od dátum kedy bol tovar prijatý na sklad, pokiaľ objem nesčítaných tovarov je menší ako objem všetkých voľných vozidiel. Následne sa vytvorí vytriedený zoznam miest do ktorých je treba jednotlivé tovary rozviesť, aby sa nenachádzalo žiadne mesto viac krát a vytvorí sa pole požiadaviek pre jednotlivé mestá, čo sú vlastne objemy jednotlivých vozidiel. Po získaní nasledovných dát sa spustí metóda `rozdelTrasu(vytriedenyZoznam, vozidlaList, poziadavky)`,kde **vytriedeny-Zoznam** je vytriedený zoznam jednotlivých miest, **vozidlaList** je zoznam vozidiel, ktorý je potrebný pre hodnoty objemov kufrov jednotlivých vozidiel a **poziadavky** je pole požiadaviek pre jednotlivé vozidla. Metóda nasledovne vráti optimálne trasy pre vozidlá. Po úspešnom vytvorení trás sa trasy nahrajú do databázy pomocou príkazu

`INSERT INTO trasa (trasaID, vozidloID, datumTrasy) VALUES (:trasaID, :vozidloID, :datumTrasy)` , kde trasaID sa vytvára automaticky pomocou triggru **AutoIncremetTrasa** , ktorý sa nachádza v databáze tak sa vloží iba číslo 1, vozidloID je pridelené postupne z počas cyklu z **vozidlaList**. Algoritmus počíta s tým , že všetky vozidla majú rovnako veľké objeme kufrového priestoru, keďže spoločnosti nakupujú rovnaký typ vozidla. **DatumTrasy** je uložený ako aktuálny dátum, kedy bola trasa vygenerovaná. Po vložení trás do databázy sa aktualizujú vozidla a ich poloha sa nastaví na 1 , aby sa stali aktívne. Následne sa aktualizujú jednotlivé tovary , ktoré boli vybraté na rozvoz. Nie všetky tovary, ktoré boli vybraté pomocou vyššie uvedeného príkazu select, sa mohli zmestiť do vozidiel, po optimalizovaní, tak sa aktualizujú iba tie, ktoré boli skutočne vybraté. Nakoniec sa zobrazia jednotlivé trasy a vozidla s jednotlivými balíkmi v okne `TrasaWindow(this, DateTime.Today)`, kde `this` je hlavné okno aplikácie a **DateTime.Today**, je aktuálny dátum. Celá metóda je volaná pomocou tlačítká, ktoré je obsluhované metódou `ButtVygenerujTrasu_Click(object sender, RoutedEventArgs e)` a metóda sa spúšťa pre plynulosť a urýchlenie procesu vygenerovania trás na novom vlákne.

4.1.1.1.2 Metóda RozdelTrasu()

Nasledovná metóda slúži na vytvorenie optimálnych trás pre jednotlivé vozidla, kde sa volá Clark Wrightova metóda z triedy **`Clarke_Wright`**. Metóda slúži aj na zobrazenie stavu procesu vytvárania pomocou objektu **`processBar`** , ktorý je umiestnený v triede **`LoadingPage`.** Preberá tri parametre a to sú **`vytriedenyZoznam`**, ktorý slúži ako zoznam vstupných miest pre algoritmus. **`VozidlaList`**, je zoznam vozidiel , ktoré sú dostupné na rozvoz a **`poziadavky`** , je pole požiadaviek pre jednotlivé lokácie do ktorý majú byť rozvezené tovary.

4.1.1.1.3 **Ostatné metódy**

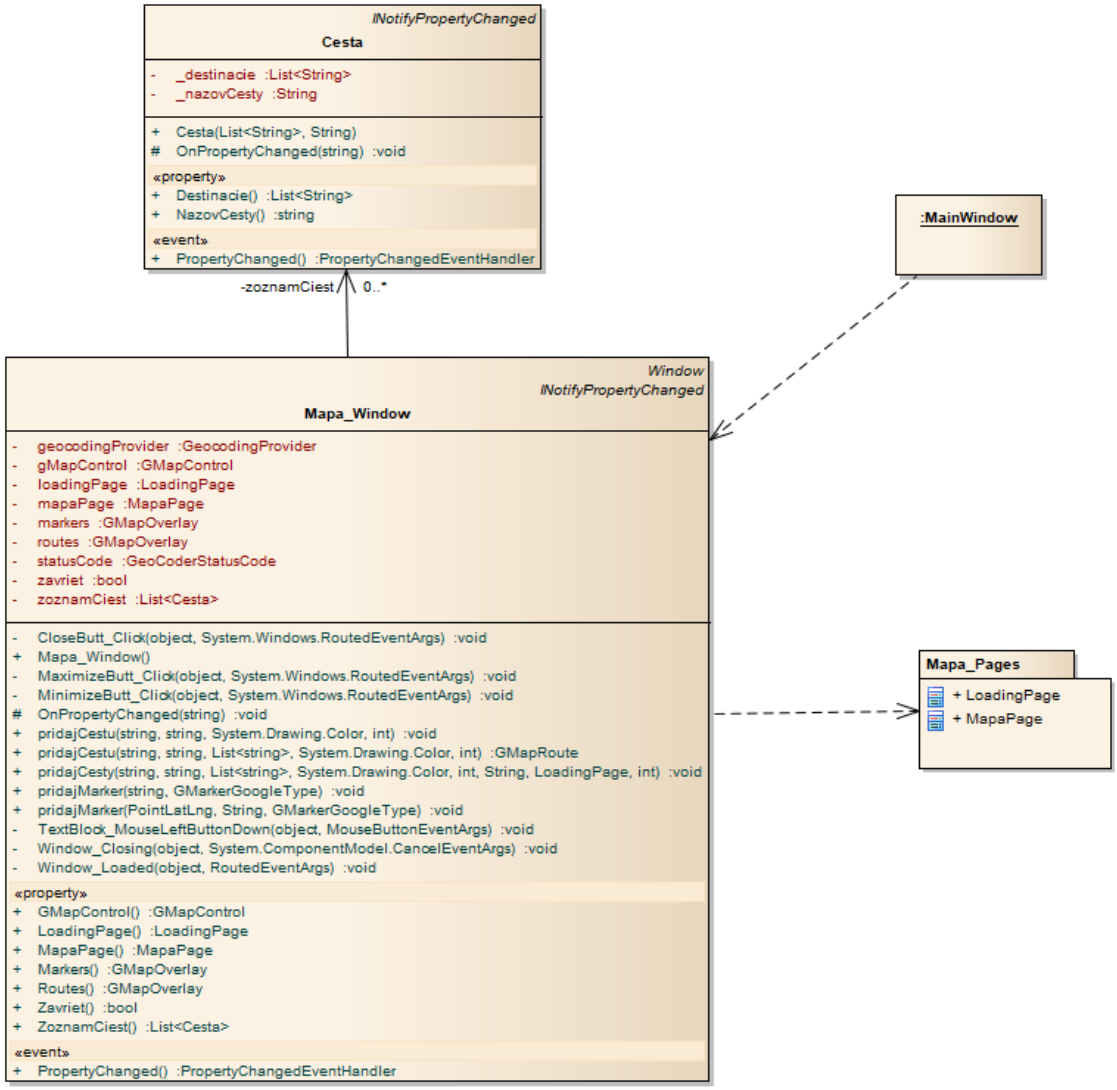
Ostatné metódy slúžia predovšetkým pre zobrazovania jednotlivých položiek ako sú vozidlá , tovary , objednávky a dodávatelia. Niektoré metódy obsahujú špeciálne príkazy select, kde sa vyberajú položky podľa určitých kritérií ako sú napríklad prijaté tovary, vybavené objednávky a podobne. Niektoré metódy slúžia pre vytvorenie okien na pridávanie položiek.

4.1.1.2 **Trieda Styly**

Táto trieda slúži pre celú aplikáciu ako **ResourceDictionary** jednotlivých štýlov. Je v nej definované veľké množstvo štýlov, ktoré nahrádzajú klasický dizajn jednotlivých prvkov, ktoré sú používané v aplikácii. Jeden zo štýlov je napríklad `WindowStyle`, ktorý určuje štýl pre všetky okná aplikácie. Obsahuje aj riadiace tlačítká pre minimalizovanie, maximalizovanie a vypnutie daného okna. Preto daná trieda obsahuje tieto 3 metódy , inač slúži čisto ako **ResourceDictionary** pre štýly. Keďže tlačítko maximalizovať sa mení podľa toho či je okno maximalizované, alebo nie tak každé okno si musí definovať ikonku tohto tlačítka samé, pretože `property content` pre dané tlačítko , ktorý slúži na zobrazovanie , čo je v danom tlačítku napísané sa dynamicky mení , ako je spomínané vyššie , tak nie je možné ho z danej triedy obsluhovať, keďže to je **ResourceDictionary** pre štýly. Štýl `LoadingIndicatorRingStyleKey,` ktorý je použitý pri načítavaní aplikácie využíva NuGet **LoadingIndicators.WPF** , ale jemne upravený. Je na ňom zmenená farba a veľkosť guličiek, ktoré sa zobrazujú ako animácia načítavania.

4.1.2 Diagram tried Mapa

Tento diagram tried zobrazuje namespace **Mapa**, v ktorom sa nachádzajú triedy **Mapa_Window** a **Cesta**. Tento namespace taktiež vlastní podnamespace a to je **Mapa_Pages** a následne je v ňom zobrazený vzťah ku hlavnej triede aplikácie a to je **MainWindow**.



Obrázok 7 Diagram tried Mapa

4.1.2.1 Trieda Mapa_Window

Táto trieda, je hlavná trieda , ktorá riadi zobrazovanie jednotlivých trás na mape a samotnú mapu a taktiež markery , ktoré sa na mape nachádzajú. Trasy a označenia pre jednotlivé adresy sa zobrazujú na princípe vrstiev.

4.1.2.1.1 Metóda pridajCesty()

Metóda slúži na zobrazenie jednotlivých trás na mapu, ako parametre sú počiatočná adresa, konečná adresa, zoznam všetkých adries , ktoré sa majú zobraziť na mape, farba trasy, šírky trasy , **LoadingPage** , ktorá slúži na zobrazenie stavu načítavania a **pocetCiest**, ktorý slúži len na výpočet percent pre zobrazenie načítavania, označuje vlastne počet trás. Metóda následne volá metódu `pridajCestu(startAdresa, endAdresa, color, strokeWidth)`, kde sa generuje cesta z jednej adresy do ďalšej v poradí, pre poslednú adresu platí , že sa generuje pre prvú a tým vznikne okružná trasa.

4.1.2.1.2 Metóda pridajCestu()

Nasledovná metóda, môže preberať dve rôzne skupiny a nasledovný odstavec sa bude venovať pre parametre `pridajCestu(string startAdresa, string endAdresa, System.Drawing.Color color, int strokeWidth)`

Táto variácia metódy slúži na zobrazenie trasy medzi dvomi bodmi na mape, aby sa táto operácia mohla uskutočniť, je treba získať geografické súradnice pre jednotlivé adresy. Pre získavanie týchto súradníc je použitý NuGet **gmaps-api-net** , ktorý slúži ako API na podporuje **Google APIs**. Pre plnú funkcionalitu je treba získať od firmy Google API Key , ktorý je nasledovne vložený do vyššie spomínaného NuGetu. Keďže Google svoje služby ponúka obmedzene a môže nastať situácia , že sa prečerpá limit, ktorý je nimi stanovený sa skúsi vykonať získanie geografických lokácii pomocou ďalšieho NuGetu a to **GMap.Net.Windows**, ktorý ponúka neobmedzené služby , ale nie vždy vie vyhľadať danú adresu. Tento NuGet ponúka rôznych providerov pre tieto služby a keďže Google v tomto prípade zlyhal, je použitý **OpenStreetMaps**. Ak sa úspešne **získaju** geografické súradnice vykreslí sa trasa medzi nimi pomocou **OpenStreetMaps** a vykreslia sa taktiež body, ktoré označujú dané adresy.

Parametre `pridajCestu(string startAdresa, string endAdresa, List<string> listAdries, System.Drawing.Color color, int strokeWidth)`

Táto variácia metódy slúži podobne ako metóda `pridajCesty()` , ale s tým , že nepotrebuje získavať adresy postupne a ani nepotrebuje už správne zoradené adresy. Pomocou NuGetu **GMap.Net.Windows** je možnosť získať optimálnu trasu a to pomocou Google providera , ktorý je vstavaný v danom NuGete, ale táto trasa sa dá vygenerovať iba do 8 adries. Ak trasa obsahuje vyše ako 8 adries , nie je nasledovná služba dostupná.

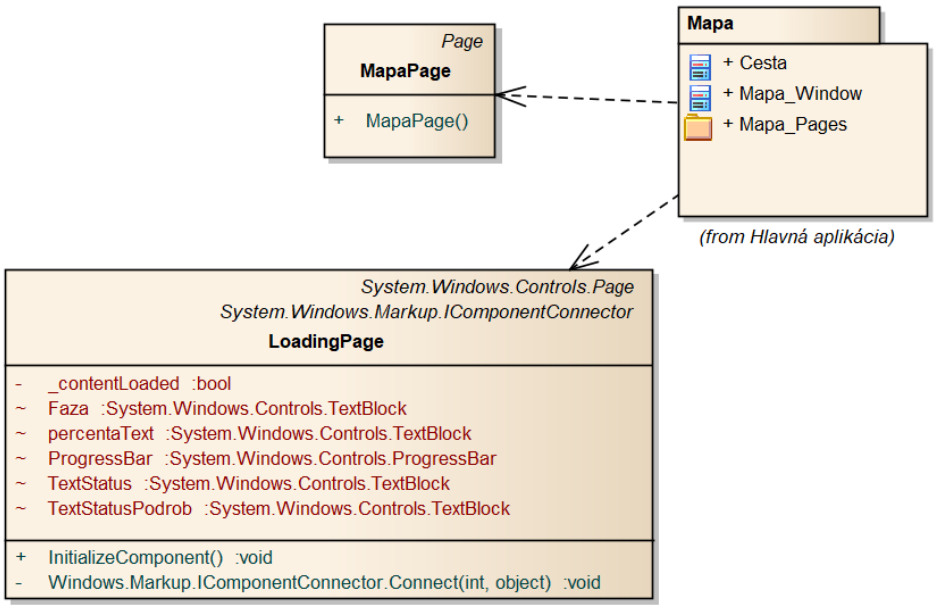
4.1.2.1.3 Ostatné metódy

Zbytok metód slúži buď a riadenie jednotlivých tlačidiel, alebo pridávanie označení pre trasy a podobne

4.1.2.2 Trieda Cesta

Táto trieda slúži pre zobrazenie jednotlivých adries pre každú vygenerovanú trasu. Je to veľmi jednoduchá , ktorá slúži viac menej ako trieda pre jednotlivé trasy.

4.1.3 Diagram tried Mapa_Pages



Obrázok 8 Diagram tried Mapa_Pages

Nasledovný diagram je diagram tried pre podnamespace **Mapa_Pages**, ktorého hlavný namespace je **Mapa**. Sú na ňom zobrazené dve triedy, ktoré reprezentujú jednotlivé stránky , ktoré trieda **MapaWindow** používa a to sú **LoadingPage** a **MapaPage**.

4.1.3.1 LoadingPage

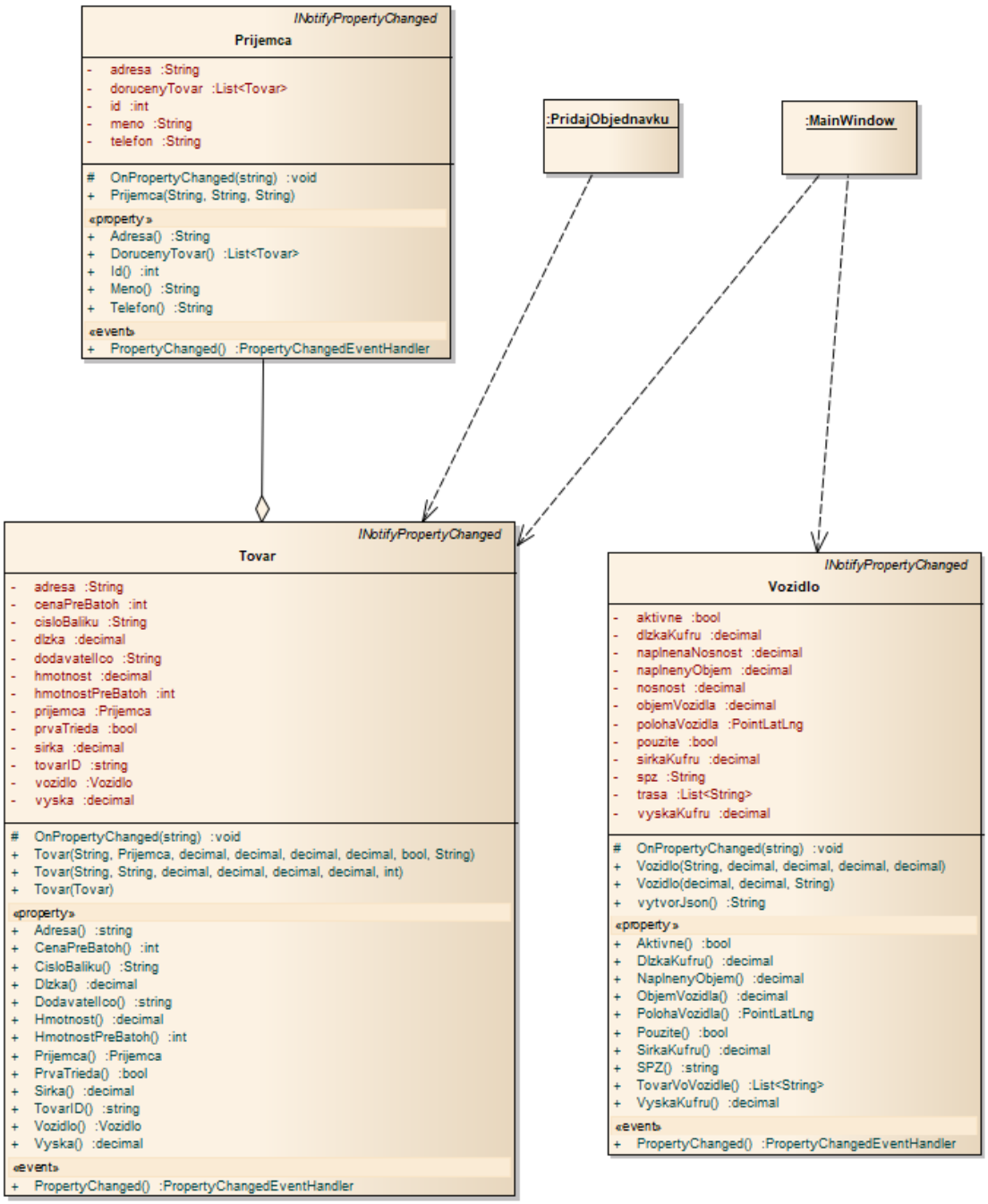
Nasledovná trieda slúži ako stránka , ktorá zobrazuje stav pri načítavaní jednotlivých trás pri ich zobrazovaní a vykresľovaní na mape.

4.1.3.2 MapaPage

Táto napohľad jednoduchá trieda slúži pre zobrazenie samotnej mapy. Obsahuje iba prvok **MainGrid**, ktorý ale na diagrame nie je vidieť, keďže je to prvok, ktorý sa nachádza v jej grafickej stránke a to **MapaPage.xaml**. Mapa sa zobrazuje pomocou NuGetu **GMap.Net.WindowsForms**, ktorý je, ale na podporu Windows **Form** a nie na podporu WPF aplikácie. Preto je potrebné spraviť **host** medzi **WPF** a **Windows Form** a nasledovní

prvok z **Windows Form**, **GMap.Control**, čo je vlastne samotná mapa sa premietne do prvku z WPF a to je **MainGrid**.

4.1.4 Diagram tried Triedy



Obrázok 9 Diagram triedy Triedy

Nasledovný diagram zobrazuje triedy, ktoré slúžia viac menej iba ako pomocné triedy. Keďže celá aplikácia ukladá a načítava dáta z databázy preto nie je treba vytvárať objekty jednotlivých prvkov, ktoré sa aplikácii nachádzajú. Nasledovné triedy slúžia iba ako

pomocné triedy, ktorých inštancie sa vytvárajú iba pri pridávaní nových prvkov do databázy, alebo ako pomocné triedy pri vykonávaní optimalizačného algoritmu pre trasy.

4.1.4.1 **Tovar**

Nasledovná tried slúži iba ako pomocná trieda pri pridávaní novej objednávky a pri vykonávaní optimalizačného algoritmu. Pri pridávaní novej objednávky sa inštancia vytvára preto, aby používateľ bol schopný vidieť aké tovary do objednávky pridal kým nie je objednávka ešte pridaná do databázy.

Inštancia tejto triedy sa taktiež vytvára pri načítavaní tovarov, ktoré majú byť rozvezené, pri požiadavke na vygenerovanie nových optimálnych trás. Kvôli optimalizácii a šetrenie požiadaviek na databázu sa tovary, ktoré sme získali z databázy sa uložia ako inštancie tejto triedy a používajú sa ďalej v algoritme.

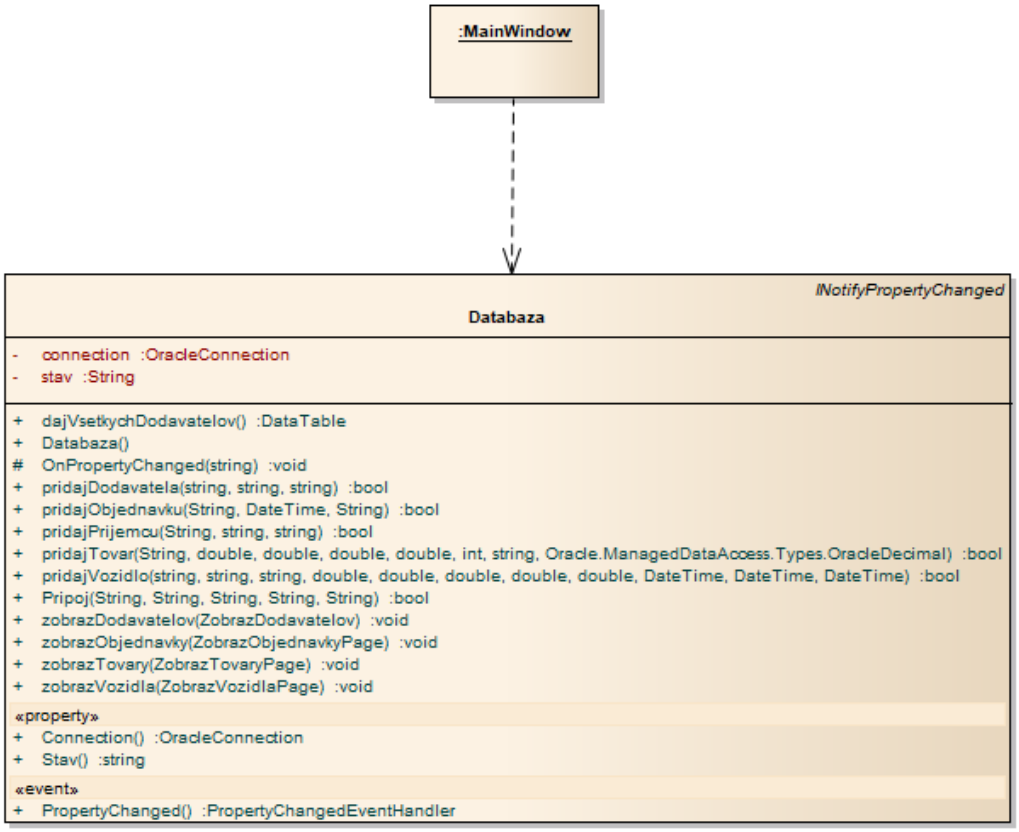
4.1.4.2 **Prijemca**

Nasledovná trieda slúži čisto iba pri pridávaní nového tovaru do objednávky a pre uloženie informácii o príjemcovi daného tovaru. Je potrebná iba , keď sa pridáva nová objednávka.

4.1.4.3 **Vozidlo**

Táto trieda slúži pri optimalizačnom algoritme, preto nemá žiadne prepojenie medzi tovarom , keďže to vôbec nie je potrebné. Nasledovná trieda sa podobne ako tovar vytvára pri požiadavke na vygenerovanie nových trás pre optimalizáciu a šetrenie požiadaviek na server.

4.1.5 Diagram tried Databaza



Obrázok 10 Diagram tried Databaza

Je to veľmi jednoduchý diagram tried, ktorý obsahuje iba jednu triedu a to je **Databaza**. Je to vlastne diagram podnamespace-u Triedy.

4.1.5.1 **Trieda Databaza**

Nasledovná trieda slúži na pripojenie , ktoré je uložené v premennej **Connection** a udržiavania pripojenia na databázu. Je vytvorená preto, aby pri každom spojení a každom požiadavku sa nemuselo vytvárať nové pripojenie, tak sa vytvorí inštancia tejto triedy. Spomínaná inštancia triedy je vytvorená v triede **MainWindow**, ktoré drží túto inštanciu a zdieľa ju pre všetky ostatné triedy , ktoré ju potrebujú.

Táto trieda obsahuje niekoľko jednoduchých metód, ktoré buď obsluhujú príkazy na zobrazenie všetkých záznamov niektorej tabuľky v databáze, alebo pridanie nového záznamu do niektorej tabuľky.

4.1.5.1.1 **Metóda pripoj()**

Nasledovná metóda slúži na pripojenie ku danej databáze. Obsahuje niekoľko parametrov a to sú **hostname**, **port**, **serviceName** , **user** a **heslo**. **Hostname** pre danú

databázu, port je port potrebný na pripojenie ku danej databáze, **serviceName** je service name pre danú databázu, **user** je používateľský login pre účet pod ktorým sa prihlasuje aplikácia na databázu a **heslo** je heslo pre daný účet.

4.1.6 Diagram tried Algoritmy

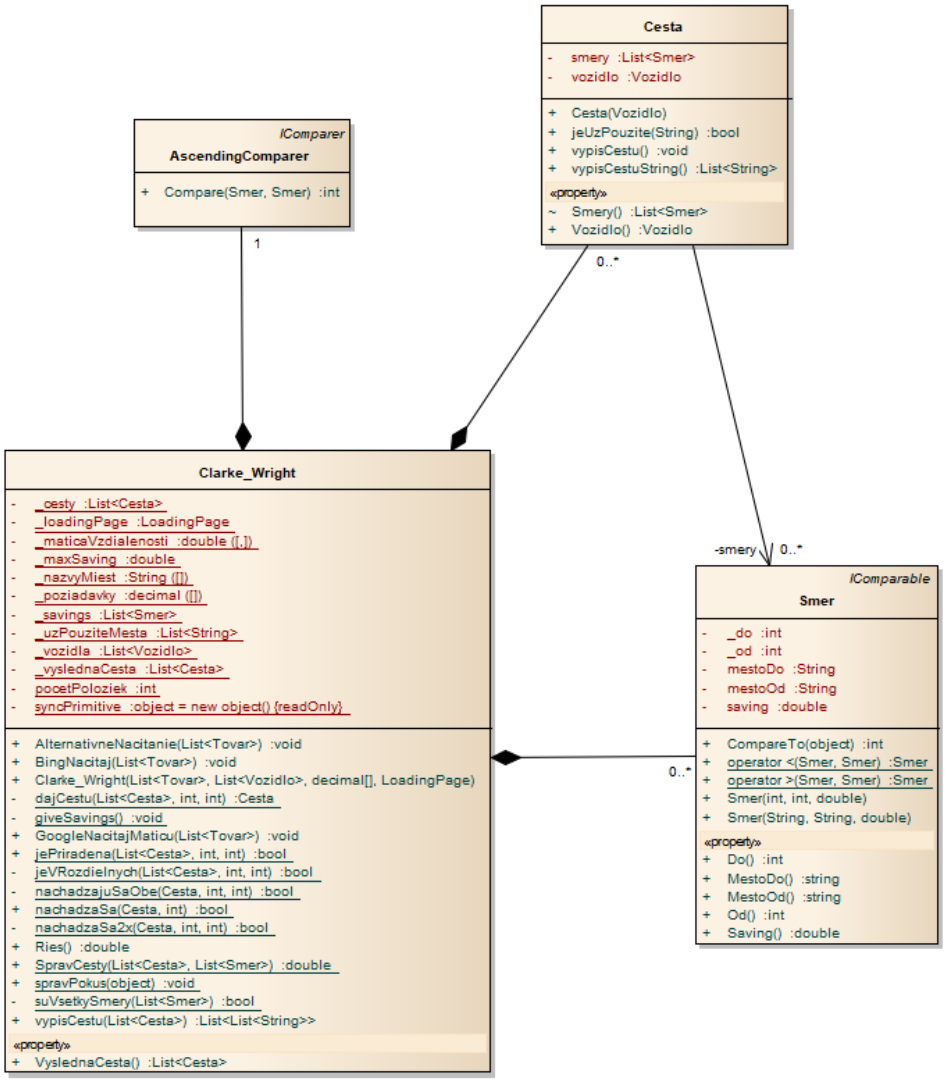


Diagram tried Algoritmy, zobrazuje triedy **Clarke_Wright** , **Cesta** , **Smer** a **AscendingComparer**. Zobrazuje vzťahy medzi jednotlivými triedami, ktoré všetky slúžia pre vykonávania algoritmu , ktorý sa zaoberá vytváraním optimálnych trás.

4.1.6.1 **Trieda Clarke_Wright**

Táto trieda obsahuje hlavný algoritmus a ostatné triedy sú iba pomocné triedy. Algoritmus , ktorý vytvára optimálna trasy je Clarke Wrightov algoritmus , ktorý je popísaný vyššie v tejto práci. Metóda , ktorá vykonáva samotný algoritmus sa nazýva `SpravCesty()`, ale ta volaná z metódy `Ries()`, ktorá spracováva výsledky zo spomínaného algoritmu.

Samotný algoritmus potrebuje pre svoje vykonávanie , niekoľko pripravených dát , nato aby sa mohol vykonať. V prvom rade , je treba získať maticu vzdialeností medzi jednotlivými miestami, následne je potrebné z tejto matice získať tzv. **savings** a ak máme tieto dve veci pripravené algoritmus sa môže začať vykonávať a výsledne trasy sú uložené ako triedy **Cesta**, ktoré obsahujú zoznam inštancii **Smer**.

Pre získavanie matice je potrebný veľký počet požiadaviek na server. Maticu však nemusíme vyplniť úplne, keďže vzdialenosti medzi jednotlivými trasami by mali byť rovnaké nehladno na smer, či ideme z bodu A do bodu B , alebo z bodu B do bodu A, vzdialenosť by mala byť rovnaká. Preto je potrebný $\sum_{i=1}^{n} i = \binom{n+1}{2} = \frac{n(n+1)}{2}$, to znamená napríklad ak by sme potrebovali vytvoriť trasy, pre 10 adries potrebujeme 45 požiadaviek. Samotná požiadavka trvá istý čas a vzhľadom nato, že pri získavaní týchto požiadaviek je istý limit z hľadiska poskytovateľov , či už na počet požiadaviek za sekundu, alebo celkový počet požiadaviek, ktorý je možný za 24h. Našťastie, niektorí poskytovatelia umožňujú službu, ktorá vracia rovno maticu týchto vzdialeností a nie je potrebné si žiadať vzdialenosti jednotlivo. Táto služba veľmi urýchľuje získavanie potrebných dát. Lenže aj táto služba ma svoje obmedzenie. Existuje 3 metódy , ktoré získavajú tieto potrebné dáta do matice a to sú `BingNacitaj()`, `GoogleNacitajMaticu()` a `AlternativneNacitanie()`. Sú vytvorené 3 metódy v rámci toho, že `BingNacitaj()` a `GoogleNacitajMaticu()` využívajú požiadavky od spoločnosti Bing a Google , ktoré pre využívanie svojich služieb potrebujú API kľúče. Môže nastať situácia , kedy tieto kľúče prestanú byť platné a preto existuje metóda `AlternativneNacitanie()`, ktorá tieto kľúče nepotrebuje. Využíva sa hlavne metóda `BingNacitaj(tovary, _poziadavky)` , ktorá využíva služby od spoločnosti Bing. Pre využívanie týchto spoločnosti je pridaný NuGet s názvom **BingMapsRESTToolkit,** ktorý podporuje niektoré služby spoločnosti Bing s tým, že si pridáte vlastný API kľúč a môžete využívať ponúkane služby. Služba , ktorá je využitá sa nazýva **Distance Matrix API.** Tá vracia požadovanú maticu , ale len do veľkosti 25x25, ak je matica väčšia, čiže potrebuje viac ako 25 adries , služba nie je dostupná. Algoritmus pre získavanie je naprogramovaný tak, ak je počet adries menší, alebo rovný ako 25 , spraví sa iba jedna požiadavka, ale ak je viac ako 25 vypočíta sa potrebný počet požiadaviek a následne sa vytvorí matica z viacerých požiadaviek do jednej. Táto metóda preberá jeden parameter a to je tovary a kde tovary sú tovary, ktoré majú byť rozvezené a obsahujú svoje adresy Metóda `GoogleNacitajMaticu(tovary),` slúži ako náhradná metóda, pretože má limit iba 8 adries , ale stále je optimálnejšia ako načítavať vzdialenosti po jednej a metóda

`AlternativneNacitanie(tovary)` je úplne posledná , lebo vyžaduje načítanie vzdialenosti po jednej, je využitá, iba v krajnom prípade, kedy zlyhali obe spomenuté možnosti vyššie. Obe alternatívne metódy preberajú rovnaký parameter ako metóda `BingNacitaj()`.

Ďalej je potrebné vytvoriť zoznam **saving**-ov, ktorý je určený výpočtom z matice vzdialeností. Jeden **saving** sa vypočíta podľa vzťahu **saving** = ($d_{0i} + d_{0j} - d_{ij}$), kde indexy pri jednotlivý zložkách určujú pozíciu v matici. Každá z tých hodnôt je vytvorená ako trieda **Smer,** ktorá vlastní vzdialenosť medzi dvomi adresami. Prvá adresa je zdrojová, z ktorej vychádzame a druhá adresa je destilačná, do ktorej smerujeme. Po získaní týchto hodnôt je potrebné ich zoradiť podľa najväčšieho a nato slúži trieda **AscendingComparer,** keďže vzdialenosti sú ako triedy je potrebné vytvoriť vlastný komparátor.

4.1.6.1.1 Metóda Ries()

Táto metóda sa zaoberá spracovaním výsledkov z metódy `SpravCesty()`, ktorá obsahuje samotný **Clarke Wrightov** algoritmus. Ak máme všetky dáta pripravené metóda sa môže začať vykonávať. Metóda najprv zavolá metódu Smer s tým , že zozname saving-ov, ktorý metóda preberie ako parameter je úplne plný, to znamená , že obsahuje všetky kombinácie inštancii tried Smer. Keďže **Clarke Wrightov** algoritmus pracuje ako Greedy algoritmus , nemusí vrátiť optimálne riešenie, preto je potrebné niektoré inštancie triedy **Smer** odstrániť zo zoznamu a tým môže vzniknúť optimálnejšie riešenie. Je to vlastne heuristická nadstavba **Clarke Wrightovej** metódy, ktorá sa nazýva **Holmes and Parker**.

Pseudo kód algoritmu:

```
for (int i = 0; i < celkovy pocet saving; i++) {
        odstraneny = _savings[i]; // uloží sa prvok, ktorý sa odstráni
            _savings.RemoveAt(i);
                for (int y = 0; y < celkovy pocet saving ; y++) {
                    _cesty.Clear(); // odstránia sa cesty, ktoré boli získane
                                        v minulom cykle
                    if (suVsetkySmery(_savings)) {  // skontroluje sa prípustnosť
                                                        riešenia
                        double saving = SpravCesty(_cesty, _savings);
                        if (saving > maxSaving){ // skontroluje sa
```

```
                                                    optimálnejšie riešenie
                                    maxSaving = saving;
                                    _vyslednaCesta=newList<Cesta>(_cesty)
                                    }
                            }
                            _savings.RemoveAt(i)
                    }
            _savings.Add(odstraneny);
            _savings.Sort(); // pre zachovalosť správneho poradia
            }
```

Algoritmus obsahuje dva cykly, jeden cyklus sa odstráni jeden **Smer** , ďalší cyklus následne skontroluje, či sa dá vytvoriť prípustne riešenie, ak áno zavolá sa metóda `SpravCesty()` ,získa sa z nej výsledok a ak je výsledne riešenie optimálnejšie ako predchádzajúce riešenie, uloží sa ako doposiaľ najlepšie riešenie a odstráni sa ďalší **Smer**. Tento vnútorný cyklus pokračuje kým nie je počet odstránených rovný celkovému počtu všetkých **saving**-ov. Po jeho skončení , vonkajší cyklus pridá predtým odstránený **Smer** a ďalšom cykle sa odstráni ďalší **Smer** a postupuje sa do vnútorného. Celkový algoritmus pracuje , kým sa vo vonkajšom cykle neodstráni , každý Smer jeden krát.

Algoritmus nezaručuje najoptimálnejšie riešenie, keďže ide o heuri-stický prístup.

4.1.6.1.2 Metóda SpravCesty()

Nasledovná metóda vykonáva samotný Clarke Wrightov algoritmus. Preberá dva parametre a to sú **cesty**, **savings,** kde cesty je prázdne pole ciest, ktoré napĺňa a **savings** je pole hodnôt **saving**. Metóda obsahuje dva zoznamy saving-ov a to zoznam **saving**-ov, ktoré už boli pridané a zoznam doposiaľ nepridaných **saving**-ov. Obsahuje taktiež zoznam vozidiel, ktorý je ale ako statická premenná triedy, tým pádom ma k nemu metóda prístup.

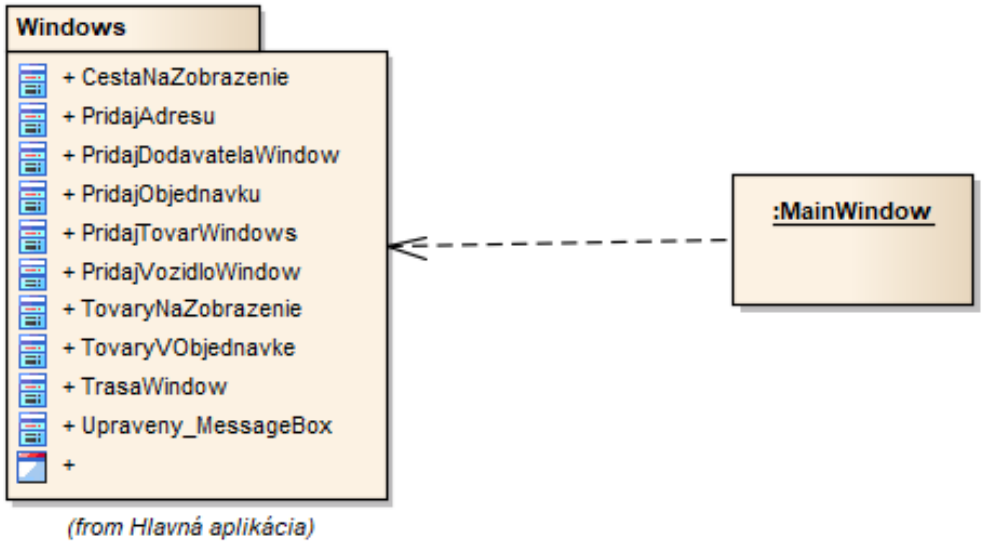
Najprv sa pridá prvý tovar, ktorý je prípustný do prvého vozidla , to znamená , že tovar sa zmestí do kufru vozidla.

Vždy po pridaní nejakého tovaru sa tovar odstráni zo zoznamu doposiaľ nepridaných tovarov a pridá sa do pridaných a výši sa pomocná premenná **pridane**, ktorá obsahuje počet,

doposiaľ pridaných vozidiel. Aktualizuje sa naplnený objem vozidla a vozidlo sa nastaví ako aktívne Aktualizuje sa pomocná premenná **saving** , ktorá určuje, ako výsledná trasa je optimálna. Čim väčšia hodnota **saving**, tým je trasa viac optimálna. A **Smer** sa pridá do inštancie **Cesta**. Ak ide o pridanie prvého tovaru, vytvorí sa nová **Cesta** a tá sa následne pridá do zoznamu **cesty**.

Následne sa vykonáva vonkajší cyklus, ktorý sa vykonáva pokiaľ , **pridane** sa nerovná počtu adries – 1, alebo premenná **nedaSa** nie je **true** . Premenná **nedaSa** sa nastaví na **true** v prípade, že sa algoritmus nedopracoval k výsledku, to znamená, že tovary boli do vozidiel rozložené tak , že sa nedá vytvoriť prípustné riešenie. To sa zistí tak, že vnútorný cyklus prejde celý a nepridá sa žiadny tovar do žiadneho vozidla. Vnútorný cyklus prechádza všetky doposiaľ nepridané **Smery** a najprv zisťuje , či sa **Smer** zo smeru nachádza nejaká , ktorá je už priradená do nejakej **Cesty**, ak áno zisti sa do ktorej **Cesty** je priradený a snaží sa smer priradiť do tej **Cesty** ,ale najprv musí skontrolovať, či náhodou adresy sa už nenachádzajú v rôznych **Cestách** a ak nie tak sa skúsi pridať. Ak sa nenachádza žiadna adresa v nejakej **Ceste**, vytvorí sa nová **Cesta** a to **Smer** sa do nej priradí.

4.1.7 Diagram tried Windows



Obrázok 11 Diagram tried Windows

Nasledovný diagram zobrazuje , že všetky triedy ktoré sú vytvorené pod namespac-ov Windows , sú používané hlavnou triedou MainWindow. V tomto namespace-i je veľmi veľa tried a bolo by neprehľadné ich zobrazovať detailne, preto je diagram zobrazený

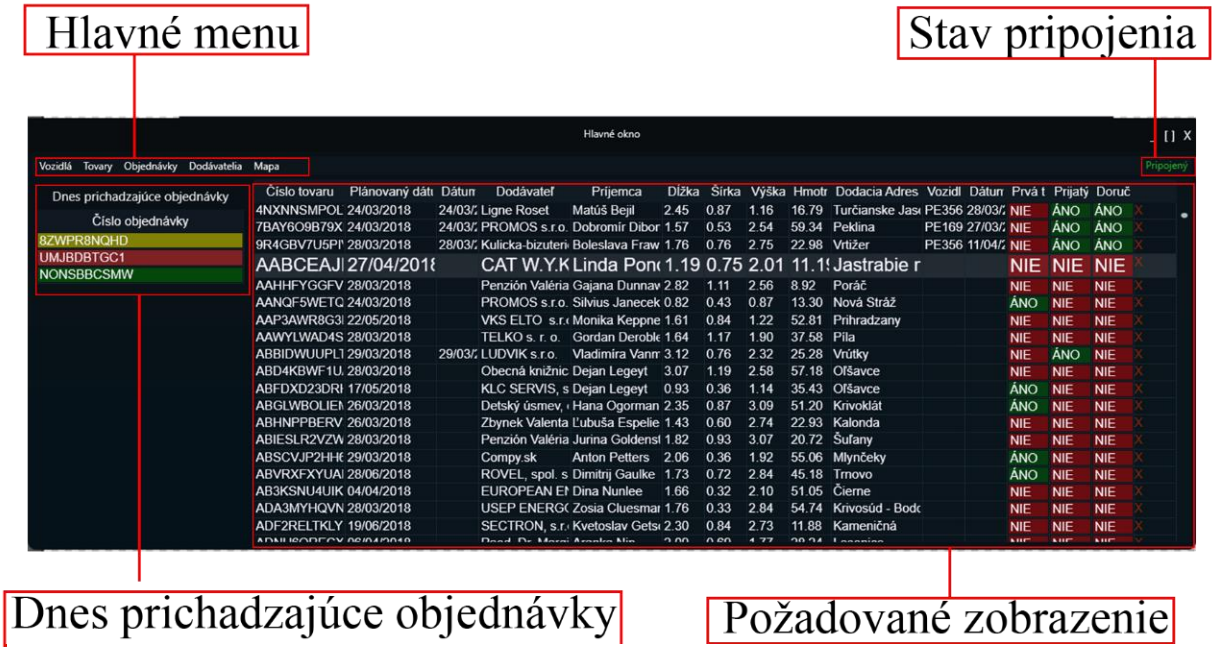
nasledovným spôsobom. Väčšina tried sú jednoduché okná, ktoré slúžia na pridávanie prvok do databázy, alebo zobrazujú tovary pri konkrétnych situáciách, ako napríklad pri zobrazení tovarov, ktoré sa nachádzajú vo vozidle alebo v objednávke a podobne. Trieda **Upraveny_MessageBox** je vlastne obyčajný **MessageBox**, len s tým, že má definovaný vlastný dizajn.

4.1.7.1 **Trieda TrasaWindow**

Táto trieda je slúži pre zobrazovanie trasy a vozidiel , ktoré by mali danú trasu absolvovať. Je možné taktiež zobrazovať tovary, ktoré sa nachádzajú v jednotlivých vozidlách. Trasa je vyberaná podľa dátumu , kedy bola vytvorená respektíve ku ktorému dátumu patrí. Ak v jeden deň bolo vytvorených viac trás, zobrazia sa niektoré vozidlá viac krát, podľa toho do koľkých trás patria pre vybratý dátum. Rovnaké vozidlá sa budú zobrať pod tým istým vozidloID, čiže pod štátnou poznávacou značkou , ale ako dve osobité vozidlá, to znamená, že ak napríklad vozidlo bolo použité 2 krát v ten istý dátum do dvoch rôznych trás, bude sa jeho štátna poznávacia značka zobrazovať dva krát. Je možné v tomto okne meniť dátum pre ktorý chce používateľ zobraziť trasy a trasy sa budú meniť na mape a budú sa meniť aj vozidlá a ich tovary podľa požadovaného dátumu.

5 Implementácia

5.1 MainWindow



Obrázok 12 MainWindow

Na obrázku **12 MainWindow** je vidieť implementáciu hlavného okna aplikácie, po zobrazení všetkých tovarov, ktoré sú evidované v databáze.

5.1.1 Stav pripojenia

Môže nadobúdať dve hodnoty a to stav **Pripojený** a **Nepripojený**. Indikujú, či je aplikácia pripojená ku databáze, alebo nie.

5.1.2 Hlavné menu

Hlavné menu slúži ako hlavné menu v ktorom je možné vyberať položky na zobrazenie, pridávať položky, zobraziť mapu a vygenerovať trasu a podobne. Slúži pre obsluhu celej aplikácie. Je implementované ako **Menu** v jazyku **XAML**.

5.1.3 Dnes prichádzajúce objednávky

Následný blok slúži pre zobrazenie objednávok, ktoré by mali doraziť na prevzatie do skladu. Jednotlivé farby indikujú stav objednávky, v ktorom sa objednávka nachádza. **Žltá** farba znamená , že objednávka je v stave **Čiastočne vybavená** , čiže nie všetky tovary

z objednávky sú prijaté do skladu. **Zelená** farba znamená, že celá objednávka je prijatá do skladu a je v stave **Vybavená**. **Červená** farba znamená, že ani jeden tovar z objednávky nebol prijatý do skladu , takže objednávka je v stave **Nevybavená**. Jednotlivé tovary sa dajú prijať pomocou okna **TovaryNaZobrazenie** , ktoré sa zobrazí po dvojkliku na objednávku

5.1.3.1 **Okno TovaryNaZobrazenie**

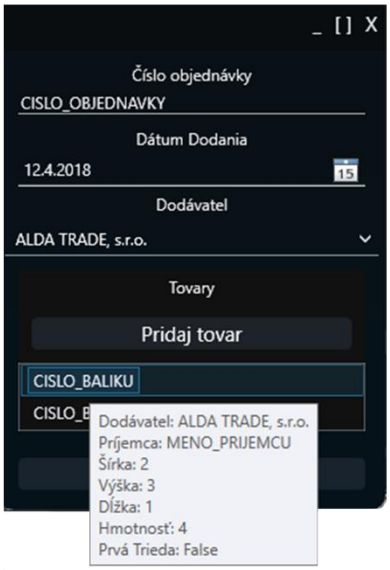
Obrázok 13 Okno TovaryNaZobrazenie

Následne okno zobrazuje tovary pre objednávku. Dajú sa v ňom tovary prijať do skladu pomocou tlačidla **Prijať tovary na sklad**, ktoré nastaví každému tovaru , ktorý má zaškrtnutý stĺpec **Prijať** atribút prijatý na 1 a pomocou príkazu update sa v databáze táto hodnota zmení. Následne sa zmení farba stĺpca Prijatý a stav tovaru na Áno, čo znamená , že tovar bol úspešne prijatý do skladu. Ak sa tovar príjme na sklad , možnosť zaškrknúť tovar zmizne.

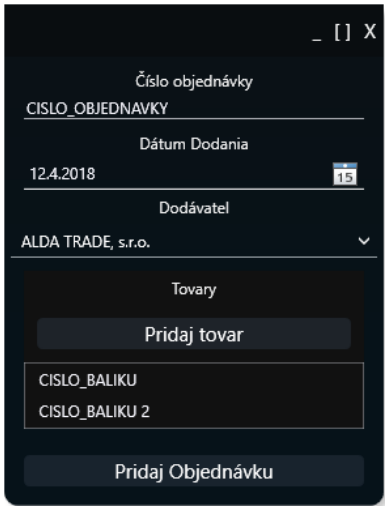
5.1.4 Požadované zobrazenie

V nasledujúcom bloku okna sa zobrazujú položky, ktoré chce užívateľ zobraziť. Každé zobrazenie sa zobrazuje v objekte `DataGrid`, ktoré dáta načítava z databázy kde výsledok z príkazu select sa uloží do `DataTable`, ktorý slúži ako zdroj pre `DataGrid`. Jednotlivé položky majú vlastný `DataGrid`, ktorý je uložený v **Page**-ov, ktoré sa menia podľa toho , ktorú položku chceme zobraziť. Jednotlivé **Page**-e, ktoré sú uložené pod namespace-om **Pages**, sa zobrazujú v objekte `Frame`.

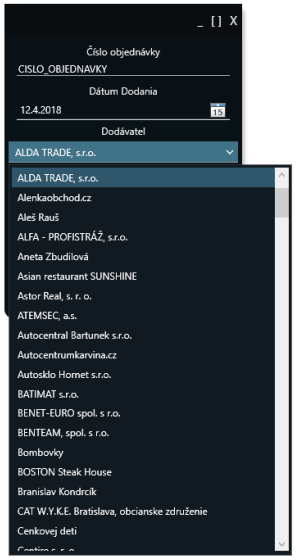
5.2 Okno PridajObjednavku



Obrázok 14 Okno PridajObjednavku 1.



Obrázok 15 Okno Pridaj Objednávku 2.



Obrázok 16 Okno PridajObjednavku 3.

Nasledovne okno slúži na pridávanie objednávok do databázy. **Tovary** , ktoré boli pridané do objednávky sa zobrazujú na čiernom pozadí a zobrazujú sa ako čísla jednotlivých tovarov. Keď sa nadíde myšou nad niektorí tovar, zobrazia sa o ňom informácie, ako je vidieť na **obrázku 15**. Objednávka sa dá pridať, iba ak sú všetky informácie o objednávke vyplnené a je pridaný aspoň jeden tovar do objednávky. Pridanie tovaru do objednávky sa

vykonáva pomocou tlačidla **Pridaj Tovar**, po stlačení tlačidla sa zobrazí okno **PridajTovarWindows**. Jednotlivý dodávatelia, **obrázok 16**, sa vyberajú z objektu `ComboBox` , ktorého zdroj je `DataTable`. Pri zmene dodávateľa alebo dátumu, sa automaticky zmení dodávateľ a dátum pre všetky tovary, ktoré sú pridané do objednávky.

5.2.1 Okno PridajTovar

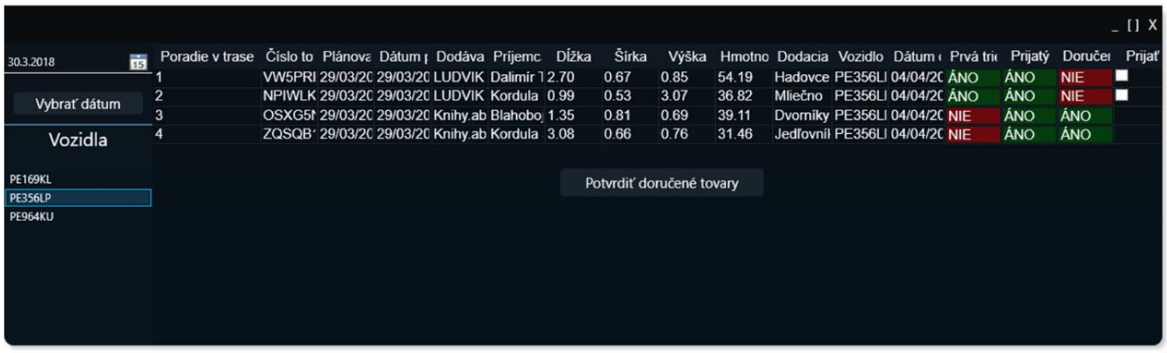
Obrázok 17 Okno PridajTovar

Nasledovné okno slúži na pridávanie tovaru do objednávky. Dodávateľ sa automaticky načíta z objednávky. Pri vytvorení tovaru sa vždy vytvorí nový príjemca. Po stlačení na tlačidla **Pridaj Tovar** sa tovar pridá do objednávky.

5.3 Ostatné okná na pridávanie

Ostatné okná na pridávanie položiek fungujú rovnako ako okno **PridajTovar**, len s tým , že ostatné položky sa rovno pridajú do databázy, kde **PridajTovar**, pridáva iba do objednávky.

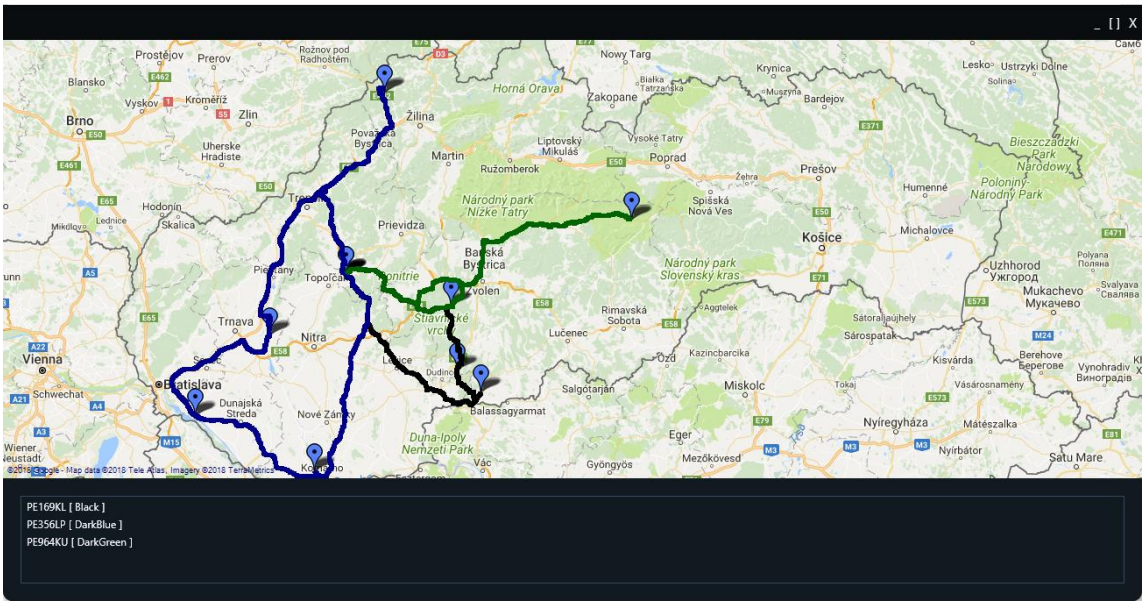
5.4 Okno TrasaWindow



Obrázok 18 Okno TrasaWindow

Toto okno slúži , ako hlavné okno pre doručovanie tovarov. Po stlačení tlačidla **Potvrdiť doručené tovary**, sa všetky tovary, ktoré majú zaškrtnuté Prijať, označia ako doručené. Každý tovar ma stĺpec **Poradie v trase**, ktorý označuje poradie tovaru v ktorom sa má doručiť. V ľavom hornom rohu je dátum, ktorý je dátum kedy bola trasa vygenerovaná a uložená databáza. Tento dátum sa dá meniť, podľa toho , pre ktorý dátum chceme zobraziť trasu. Zobrazí sa celý kalendár , nie len dátumy, ktoré sú platné.

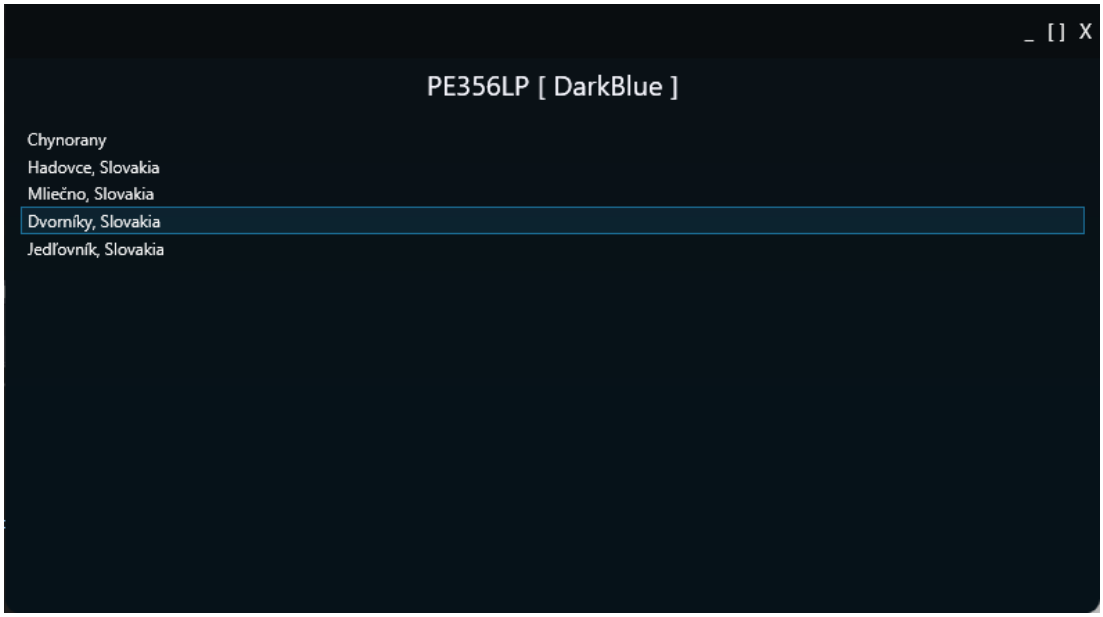
5.5 Okno MapaWindow



Obrázok 19 Okno MapaWindow

Nasledujúce okno zobrazuje mapu s optimálnymi vygenerovanými trasami, ktoré sa načítajú podľa dátumu, ktorý sa vyberá v okne **TrasaWindow**. Pod mapou sa zobrazujú štátne poznávacie značky pre vozidlá, pre ktoré platia tieto trasy. Každá štátna poznávacia značka , patrí pre jedno vozidlo a v hranatej zátvorke je názov farby, pre ktorú nasledujúca trasa patrí. Po dvojkliku na štátnu poznávaciu značku sa zobrazí okno **CestaNaZobrazenie**.

5.5.1.1 Okno CestaNaZobrazenie



Obrázok 20 Okno CestaNaZobrazenie

Toto okno sa zobrazí po dvojkliku na štátnu poznávaciu značku v okne **MapaWindow**. Nachádzajú sa v ňom zoradené adresy , do ktorý má dané vozidlo tovary doručiť. Prvá adresa je adresa **depa** v tomto prípade ide o adresu **Chynorany**. Samozrejme trasa je navrhnutá tak, aby sa vozidlo vrátilo do **depa**, len táto adresa sa už nezobrazuje, ale je posledná v zozname.

Záver

Cieľom tejto práce bolo vytvoriť aplikáciu ktorá bude podporovať základne prvky skladového hospodárstva a bude umožňovať vygenerovať optimálne trasy pre vozidlá.

Výsledná aplikácia poskytuje užívateľom, všeobecný prehľad o jednotlivých tovaroch, objednávkach a ostatných prvkoch, ktoré sú potrebné sa funkcionalitu dopravnej spoločnosti. Je možné pridávať tovary a nové objednávky, prijímať tovary na sklad a celkovo pracovať tak , aby spoločnosť bola schopná plne fungovať. Navyše aplikácia umožňuje vygenerovať optimálne trasy, pre rozvoz tovarov, ktoré sa nachádzajú na sklade.

Aplikácia je programovaná v jazyku C# ako WPF projekt a to umožňovalo jednoducho vytvoriť grafický dizajn aplikácie pre lepší používateľský dojem.

Aplikácia je založená na databáze , ktorá umožňuje rýchle získavanie dát a taktiež zaručuje konzistenciu v dátach. Je možné používať aplikáciu na dvoch rôznych zariadeniach a obaja užívatelia budú mať rovnaké dáta.

Na vytvorenie optimálnych trás pre vozidla je použitý algoritmus Clarke Wright , ktorý je popísaný v kapitole 2. Výsledne trasy je možné si zobraziť na interaktívnej mape, ktorá jednotlivé trasy zobrazí a rozdelí podľa farieb. Dá sa zobraziť aj poradie jednotlivých adries pre všetky vozidlá, ktoré boli vybraté na rozvoz.

Možnosti ďalšieho rozšírenia

Aplikácia by mohla použitá aj reálnou dopravnou spoločnosťou, len predtým by potrebovala ďalšie rozšírenia pre plnú funkcionalitu a to sú napríklad:

- Zobrazovanie vozidiel na trase podľa polohy
- Ďalšie potrebné funkcie pre plnohodnotné skladového hospodárstvo
- Generovanie faktúr a prijímacích dokumentov a iných
- Plnohodnotnú podporu zo strany služieb na získavanie geografických lokácii
- Podpora viacerých skladov
- Možnosť generovať optimálne trasy pre viacero skladov
- Webstránka pre prijímanie objednávok

Zoznam použitej literatúry

[1] TUZAR, Antonín, Petr MAXA a Vladimír SVOBODA. *Teorie dopravy*. Praha: Vydavatelství ČVUT, 1997. ISBN 80-01-01637-4.

[2] CLARKE, G; WRIGHT, J W. : *Scheduling of Vehicles from Central Depo to Number of Delivery Points*, Operations research 12,1964 , strana 568-581 ISSN 0030-364X

[3] JANÁČEK, Jaroslav. *Optimalizace na dopravních sítích*. Žilina: Žilinská univerzita, 2002. ISBN 80-8070-031-1.

[4] MATIAŠKO, Karol; VAJSOVÁ, Monika; ZÁBOVSKÝ, Michal: *Databázové systémy a technológie.* 1.vyd. Bratislava: STU 2009. ISBN 9788022730358

[5] MATIAŠKO, Karol; VAJSOVÁ, Monika; ZÁBOVSKÝ, Michal: *Základy databázových systémov.* 1.vyd. Bratislava: EDIS 2008. ISBN 9788080708207

[6] Nathan, A : *Windows Presentation Foundation Unleashed*. Pearson Education, 2006. ISBN 9780132715621

[7] Microsoft, BingMapsRESTToolkit version 1.1.4 , 10.3.2018, , [online] [cit.12.4.2018] dostupné na: https://github.com/DotNetKit/DotNetKit.Wpf.AutoCompleteComboBox

[8] Microsoft, EntityFramework version 6.2.0, 10.26.2017, dostupné na: https://github.com/aspnet/EntityFramework6/wiki

[9] Jurgen De Leon Rodriguez, GMap.NET.Windows version 1.8.5, 4.4.2018, , [online] [cit.12.4.2018] dostupné na: https://github.com/judero01col/GMap.NET

[10] radioman @ FLAT EARTH, GMap.NET.WindowsForms version 1.7.5, 2.12.2016, , [online] [cit.12.4.2018] dostupné na: https://archive.codeplex.com/?p=greatmaps

[11] Eric Newton,Richard Thombs, gmpas-api-net version 0.22.0, 27.12.2017, , [online] [cit.12.4.2018] dostupné na: https://github.com/ericnewton76/gmaps-api-net

[12] 100GPing100, LoadingIndicatiors.WPF version 0.0.1, 13.6.2015, , [online] [cit.12.4.2018] dostupné na: https://github.com/100GPing100/LoadingIndicators.WPF

[13] Gareth Evans, Microsoft.Windows.Shell version 3.0.1 18.5.2012

[14] James Newton-King, Newtonsoft.Json version 11.0.2, 24.3.2018, , [online] [cit.12.4.2018] dostupné na: https://www.newtonsoft.com/json

[15] Oracle, Oracle.ManagedDataAccess version 12.2.1100, 31.5.2017

[16] SQLite Development Team, System.Data.SQLite version 1.0.108, 3.2.2018, , [online] [cit.12.4.2018] dostupné na: https://system.data.sqlite.org

[17] SQLite Development Team, System.Data.SQLite.Core version 1.0.108, 3.2.2018, , [online] [cit.12.4.2018] dostupné na: https://system.data.sqlite.org

[18] SQLite Development Team, System.Data.SQLite.EF6 version 1.0.108, 3.2.2018, , [online] [cit.12.4.2018] dostupné na: https://system.data.sqlite.org

[19] SQLite Development Team, System.Data.SQLite.Linq version 1.0.108, 3.2.2018, , [online] [cit.12.4.2018] dostupné na: https://system.data.sqlite.org

[20] Money S3, [online] [cit.12.4.2018] dostupné na: http://www.money.sk/money-s3/vlastnosti-systemu/skladove-hospodarstvo-objednavky/

[21] Oberon, , [online] [cit.12.4.2018] dostupné na: https://exalogic.sk/

[22] OptaPlanner, , [online] [cit.12.4.2018] dostupné na: https://docs.optaplanner.org/7.6.0.Final/optaplanner-docs/html_single/index.html#vehicleRouting

Zoznam príloh

Prílohy

Príloha A: CD

Príloha obsahuje:

- Práca vo formáte PDF
- Zdrojový kód aplikácie