Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Generátor reportů

2021

Bc. Lucia Harceková

Vedoucí práce: RNDr. Eduard Bartl, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Lucia Harceková
Název práce:	Generátor reportů
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2021
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	RNDr. Eduard Bartl, Ph.D.
Počet stran:	67
Přílohy:	1 CD/DVD
Jazyk práce:	slovenský

Bibliograhic info

Author:	Bc. Lucia Harceková
Title:	Generator of reports
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2021
Study field:	Computer Science, full-time form
Supervisor:	RNDr. Eduard Bartl, Ph.D.
Page count:	67
Supplements:	1 CD/DVD
Thesis language:	Slovak

Anotácia

Cieľom práce je navrhnúť nový prístup k tvorbe reportov u informačných systémov spoločnosti Oltis Group. Práca zahŕňa implementáciu navrhnutého riešenia ako samostatného reportovacieho nástroja v podobe webovej aplikácie. Pojednáva o pôvodnom riešení a o konkurenčných riešeniach. Text práce ďalej obsahuje návrh a technické detaily aplikácie, a užívateľskú dokumentáciu.

Synopsis

The thesis aim is to propose a new approach to creating reports for information systems of the company Oltis Group. The work includes the implementation of the proposed solution as a separate reporting tool in the form of a web application, discusses the original solution and competing solutions. The text of the thesis also contains the structure of the model and technical details of the application, and user documentation.

Kľúčové slová: generátor reportov; webová aplikácia, OGreport, správa

Keywords: generator of reports; web application, OGreport, report

Moje poďakovanie patrí spoločnosti OLTIS Group za umožnenie realizácie mojej diplomovej práce a RNDr. Eduardovi Bartlovi, Ph.D. Za príkladné vedenie pri jej vypracovaní.

Čestne vyhlasujem, že som celú prácu vrátane príloh vypracovala samostatne a za použitia iba zdrojov spomínaných v texte práce a uvedených v zozname literatúry.

dátum odovzdania práce podpis autora

Obsah

1 Úvod **10**

2 Cieľ **11**

3 Analýza existujúcich riešení **12**
3.1 Pôvodný OGreport 12
3.2 Komerčné riešenia 13
3.2.1 Google Data Studio 14
3.2.2 AutoGPS 14
3.3 Zhrnutie 14

4 Nové riešenie **15**
4.1 Aktéri systému 15
4.2 Vlastnosti aplikácie 16
4.2.1 Správca dopytov 16
4.2.2 Správca reportov 17
4.2.3 Správca užívateľov 18

5 Návrh aplikácie **19**
5.1 UML 19
5.1.1 Diagramy prípadov použitia 19
5.1.2 Diagramy tried 22
5.2 Dátový model 24

6 Použité technológie **26**
6.1 ASP.NET MVC 26
6.1.1 ASP.NET 26
6.1.2 MVC architektúra 26
6.1.3 ASP.NET MVC 5 27
6.2 Repository Pattern 28
6.3 C# 28
6.4 SQL a MS SQL 28
6.5 JavaScript 29
6.6 CSS a HTML 30
6.7 NuGet Package Manager 30

7 Štruktúra projektu **31**

8 Programátorská dokumentácia **34**
8.1 Spracovanie dátových modelov 34
8.1.1 Implementácia 34
8.1.2 Pridanie nového modelu 35
8.1.3 Výhody a nevýhody 35
8.2 Parametre 36

8.2.1 Značenie parametrov . . . 36
8.2.2 Implementácia . . . 36
8.2.3 Nepovinné parametre . . . 39
8.2.4 Vytváranie parametrov . . . 39
8.2.5 Kontrola parametrov . . . 40
8.2.6 Typy parametrov . . . 40
8.2.7 Výhody a nevýhody . . . 41
8.3 Spracovanie dopytov . . . 43
8.3.1 Tvar dopytu . . . 43
8.3.2 Kontrola dopytu . . . 43
8.3.3 Autorizácia dopytu . . . 45
8.3.4 Výhody a nevýhody . . . 45
8.4 Zostava . . . 47
8.4.1 Oprávnenie zostáv . . . 47
8.4.2 Formáty . . . 47
8.5 Spracovanie akcií . . . 48
8.6 Užívateľské rozhranie . . . 48

9 Užívateľská dokumentácia 49
9.1 Každý užívateľ . . . 49
9.1.1 Správa profilu . . . 49
9.2 Administrátor zostáv . . . 50
9.2.1 Prehľad zostáv . . . 50
9.2.2 Detaily zostavy . . . 51
9.2.3 Vytvorenie zostavy . . . 51
9.2.4 Generovanie zostavy . . . 52
9.2.5 Upravenie zostavy . . . 53
9.2.6 Zmena poradia zostavy . . . 53
9.2.7 Odstránenie zostavy . . . 54
9.3 Administrátor dopytov . . . 54
9.3.1 Prehľad dopytov . . . 54
9.3.2 Vytvorenie dopytu . . . 54
9.3.3 Editácia dopytu . . . 56
9.4 Administrátor užívateľov . . . 56
9.4.1 Prehľad užívateľov . . . 57
9.4.2 Vytvorenie užívateľa . . . 57

Záver 58

Conclusions 59

A Spustenie 60
A.1 Potrebné technológie . . . 60
A.2 Obnovenie databázy . . . 60
A.3 Sprevádzkovanie aplikácie . . . 60

B Testovanie **61**
B.1 Testovacie databázy 61
B.1.1 Restaurant 61
B.1.2 TrainTimetable 61
B.1.3 LWP_Kartoteka_Vyvoj 62
B.2 Príklady dopytov 62

C Obsah priloženého CD **66**

Literatúra **67**

Zoznam obrázkov

1 Príklad uloženia zostavy VlakAkce pôvodného generátoru na disku 13
2 Príklad exportovaného reportu VlakAkce v pôvodnom riešení . . . 13
3 UML, 1. diagram prípadov použitia 20
4 UML, 2. diagram prípadov použitia 20
5 UML, 3. diagram prípadov použitia 22
6 UML, 1. diagram tried 23
7 UML, 2. diagram tried 23
8 UML, 3. diagram tried 24
9 Schéma databázy Generator_of_reports 25
10 MVC pattern 27
11 ASP.NET MVC Repository pattern so servisnou vrstvou 28
12 Zobrazenie časti pripojeného modelu LWP_Kartoteka_Vyvoj . . 35
13 Tabuľka nastavovania oprávnení pre report alebo dopyt 45
14 Pohľad na užívateľský profil 50
15 Pohľad na vytvorenie zostavy 52
16 Pohľad na vytvorenie dopytu 55
17 Pohľad na editáciu dopytu 56
18 Pohľad na editáciu užívateľa 57
19 Schéma testovacej databázy Restaurant 61
20 Schéma testovacej databázy TrainTimeline 62

Zoznam tabuliek

1 Text prípadu použitia vytvorenia dopytu 21

Zoznam zdrojových kódov

1 Pridanie spojenia na dátový model 35
2 Zadefinovanie regexu na rozpoznanie parametrov 36
3 Ukážka prvej možnosti riešenia parametrov 37
4 Ukážka druhej možnosti riešenia parametrov 37
5 Možné implementácie parametru áno/nie 38
6 Implementácia rozhrania pre typ parametru 39
7 Ukážka nepovinných parametrov 40
8 Ukážka nahradzovania parametrov 42
9 Dopyt nepoužiteľný pre reportovanie 44
10 Príklady základných chýb a chybových hlášok 46
11 Príklady dopytov s parametrom typu pozitívne číslo 62
12 Príklady dopytov s parametrom obdobie pre dátum 63
13 Príklady dopytov s číselným parametrom pre podmienky 63
14 Príklady dopytov s textovým parametrom pre podmienky 64

15 Príklady dopytov pre obecný číselník nad modelom 64
16 Príklady dopytov pre logický parameter True/False 65
17 Príklady dopytov s obecným výberom 65
18 Príklad dopytu s viacerými typmi parametrov s ich nahradením . 65

1 Úvod

Reportovanie slúži na monitorovanie situácie v podniku, prehľadu o stave a odhaľovanie nedostatkov. Je to súčasť každého dátového systému či organizácie. Za týmto účelom existujú generátory reportov. Generátor použije informácie zo zdroja, spracuje ich podľa potreby a využije ich k tvorbe dokumentu.

Report zobrazuje aktuálny stav v dobe, keď bol vytvorený a obyčajne nepodáva bližšie vysvetlenie k údajom. Zobrazuje zhrnutie dát a interpretáciu vyžiadaných výsledkov. Preto je dôležité vedieť adresáta, ktorému je určený a mať zmysluplne vyfiltrované položky v ňom.

Primárny účel reportu je podať potrebnú informáciu čo najjednoduchšou formou. Takmer vo všetkých prípadoch sa jedná o čitateľné tabuľky dát. Reporty nemusia byť zrozumiteľné pre každého, dôležitá je osoba, respektíve publikum, ktorému je daný report určený. Reporty nie sú určené k propagácii, ani k prezentácii, na grafickej stránke reportu z praktického hladiska nezáleží, pokiaľ sú údaje v ňom zrozumiteľné. Preto sa nesmú mýliť a grafické spracovania, reprezentujúce dáta na marketingové účely a reklamy, ktoré slúžia na vizualizáciu už spracovaných dát a nie ich získanie.

Väčšina systémov, ktoré sa hlásia k reportovacím nástrojom v skutočnosti dáta iba vizualizujú. Nájsť vhodnú aplikáciu, ktorá by umožňovala zadávanie špecifických dopytov s komplikovanejšími metrikami pre bežného užívateľa a umožňovala by ich opätovné použitie, spracovala dáta podľa potreby a následne ich exportovala, je veľmi náročné. Mnohé firmy si preto vytvárajú vlastné reportovacie nástroje, priamo zamerané iba na ich vnútropodnikové použitie. Často im chýba univerzálnosť a sú určené iba užívateľom s vysokou znalosťou dopytovacieho jazyka a štruktúrou príslušného zdroju dát.

2 Cieľ

Zámer mojej práce je nájsť nový prístup k tvorbe reportov u informačných systémov (IS) spoločnosti OLTIS Group. A to vytvoriť generátor reportov, pomocou ktorého si môže reporty nad dopytmi vytvárať zákazník sám vo vlastnej réžii. Vyškolený užívateľ zákazníka si bude vytvárať dopyty s parametrami nad pripojeným dátovým modelom. Následne sa tieto dopyty môžu použiť k tvorbe požadovaných reportov. Navrhnuté riešenie bude implementované ako webová aplikácia, ktorá bude poskytovať správu dopytov a reportov.

Text práce oboznamuje čitateľa s navrhovaným modelom riešenia a podrobným popisom implementácie vybraných častí aplikácie, spolu s ich odôvodnením. Ďalej text zahŕňa súpis využitých technológií, užívateľskú príručku a popis aktuálneho riešenia.

3 Analýza existujúcich riešení

3.1 Pôvodný OGreport

Pôvodný systém generovania reportov nie je samostatná aplikácia alebo jeden spustiteľný súbor. Generátor pozostáva z viacerých priečinkov, každý reprezentuje samostatnú zostavu. Zostava pre jeden report obsahuje nasledovné súbory:

- .exe súbor pre generovanie zostavy,
- .sqm programátorom vytvorený SQL dopyt,
- .xls šablónu, obsahujúcu všetko formátovanie,
- pri niektorých súboroch je špeciálna koncovka reprezentujúca verziu upravenú na mieru pre zákazníka (napríklad _34 znamená id zákazníka 34).

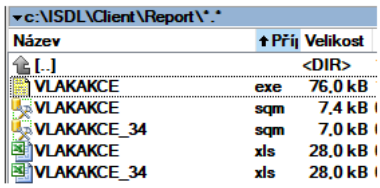
Pôvodné riešenie používa pre každú zostavu samostatnú kolekciu súborov. Každá kolekcia je vytvorená zaškoleným pracovníkom. Tento proces je popísaný v nasledujúcich krokoch.

1. Najprv sa vytvorí dopyt v samostatnom .sqm súbore, kde je miesto pre vloženie podmienok a značiek pre parametre. Možné typy parametrov sú obmedzené a systém neposkytuje žiaden spôsob na odladovanie chýb alebo malej nápovedy. Preto je zaužívané, že si programátor dopyt najskôr samostatne otestuje.
2. Nasleduje pripravenie .xls šablóny, respektíve v mnohých prípadoch jej skopírovanie, keďže okolo 98% reportov má rovnakú šablónu. V .xls je nutné dodržovať zavedené štandardy pre šablónu.
3. Na záver sa pripraví súbor .exe s príslušnými cestami k šablóne a dopytu a bližším popisom parametrov.

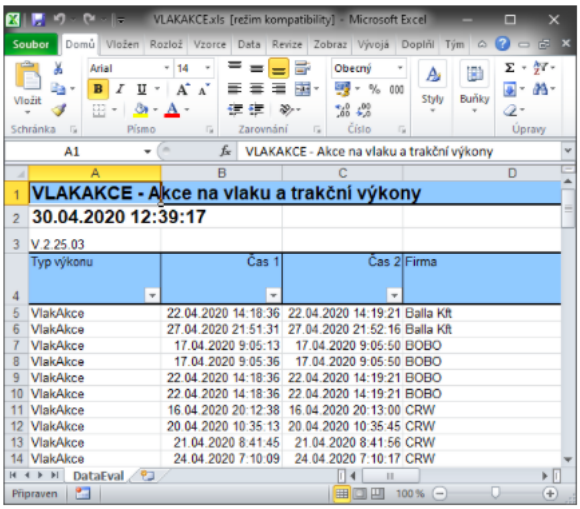
Po pripravení zostavy môže ľubovoľná osoba s prístupom k daným priečinkom generovať reporty. Priebeh generovania pozostáva z viacerých krokov. Po kliknutí na .exe súbor, sa otvorí dialógové okno pre zadanie parametrov. Následne na pozadí naprogramované .exe s podmienkami upraví SQL dopyt a potom ho vykoná. Výstupné dáta z .exe sú vložené do šablóny a výstup je exportovaný ako .xls formát.

Celkové riešenie je funkčné, ale s mnohými nedostatkami a obmedzeniami. Mnohé z nich nie je možné elegantne vyriešiť pri stave a zvolenej technológií aktuálneho riešenia. Za hlavný problém považujem nedostatok robustnosti a možné zavedenie reštrikcií nad dátami, rozšírenie parametrov a generovanie reportu s viacerými dopytmi. Negatívom je aj redundancia kódu a porušovanie princípu DRY[1]. Komunikácia s užívateľom a užívateľské rozhranie je ďalší nedostatok, keďže až na jedno modálne okno, sa jedná o konzolovú aplikáciu a užívateľovi nie je sprostredkovaná odozva systému.

[1]Don't repeat yourself, princíp na znižovanie nadbytočného a duplicitného kódu.



Obr. 1: Príklad uloženia zostavy VlakAkce pôvodného generátoru na disku



Obr. 2: Príklad exportovaného reportu VlakAkce v pôvodnom riešení

3.2 Komerčné riešenia

V mnohých prípadoch je reportovací nástroj súčasťou aplikácie (AutoGPS, Toggl, Clevork, . . .). Niekedy však aplikácia disponuje iba dátami, ktoré je nutné spracovať ďalej do prehľadného reportu. Z tohto účelu existujú generátory reportov ako samostatné systémy. Pri dispozícii dátami, ktoré vyžadujú spracovanie sa naskytujú nasledovné možnosti:

- Najväčšie množstvo reportovacích nástrojov na trhu je zamerané na webové stránky. Obsahom tejto práce je výskum reportov nad dátami. Z tohto dôvodu sa tejto kategórii venovať ďalej nebudem.

- Online generátory ponúkajú širokú škálu funkcionalít, predovšetkým čo sa týka vizualizácie pred exportom. Tieto riešenia však často obmedzujú užívateľov neznalých SQL, pokiaľ dáta už nie sú presne upravené. Väčšinou neumožňujú odkladanie šablón, spracovanie viacerých dopytov a aj s spresnením dopytu a zvolením metriky majú problém. Preto je lepšie najprv dáta spracovať cez dopyt napr. v MSSQL a až potom ich nahrať do generátoru pre vytvorenie reportu, čo predstavuje prácu navyše.

- Desktopové aplikácie ušité na mieru pre danú firmu. Výhodou je, že sú jednoduché na ovládanie a prispôsobené presným potrebám. Mnohé takéto

systémy majú problém s pridaním nového typu reportov. Pre rozšírenie a pridanie nového reportu je nutné u väčšiny pridanie nového modulu.

- Ďalej sú tu firmy, ktoré akceptujú dáta, spracujú ich a vrátia na základe požiadaviek hotový report.

Bližšie som sa rozhodla popísať dvoch zástupcov reportovacích systémov.

3.2.1 Google Data Studio

Google Data Studio zastrešuje enormné množstvo funkcionalít zadarmo a na voľnom trhu je mu ťažké konkurovať. S jeho pomocou môžete vytvoriť rôzne druhy marketingových správ alebo informačných panelov. Umožňuje načítanie dát z rôznych zdrojov ako je cloud, .csv súbor, ads, ... Nevýhoda je v upravovaní zvolených dát a nastavovaní metriky pri vytváraní tabuliek. Nedostačuje na spracovanie dát, jeho prednosť je vizualizácia. Výstup a grafické spracovania sú prepracované, napriek tomu je vhodnejšie využitie tohto reportovacieho zariadenia na marketingové účely, než priamo na reportovacie.

3.2.2 AutoGPS

AutoGPS som si vybrala ako zástupcu, kde reportovanie je dodatkom k inej primárnej funkcionalite systému. AutoGPS je elektronická kniha jázd, ktorá slúži na sledovanie osobných a nákladných automobilov.

Aplikácia umožňuje vytvorenie a uloženie reportovacej šablóny pre opakované použitie, kopírovanie a úpravu existujúcich šablón, zdieľanie vytvorených šablón, uloženie rozpracovaných šablón, export do viacerých formátov, ako sú tabuľky, grafy, PDF, XLS, CSV, TXT. Avšak nedovoľuje vlastné vytvorenie filtrov nad dátami, upravenie reportu a vloženie doplnkových dát do reportu. Preto užívateľ pracuje už s pripravenými reportmi, ktoré nemôže meniť.

3.3 Zhrnutie

Aktuálne riešenie spĺňa účel, avšak má viacero nevýhod. Na prvý pohľad k nim patrí užívateľské rozhranie v podobe preklikávania súborov a chýbajúca spätná väzba. Neposkytuje reštrikcie nad informáciami a ich rýchle zdieľanie v prípade úpravy. Možnosti budúceho rozšírenia systému sú značne obmedzené, či už v zmene formátu reportov, v pridaní typov parametrov či prípadných obmedzení nad dopytmi. Riešenie postráda redundanciu, pre každú novú zostavu iba s malou úpravou musia byť vytvorené minimálne tri nové súbory (takmer identické). O robustnosti riešenia sa nie je možné vyjadrovať.

Oproti tomu komerčné riešenia disponujú širokou mierou funkcionalít, ktorú je potrebné náležite zaplatiť. Jednou z najväčších nevýhod komerčných riešení je poskytnutie firemných dát tretej strane. Dáta musia byť pred odoslaním upravené. Ďalšie nedostatky spočívajú hlavne v zdĺhavej korešpondencii, odozve a celkovej závislosti na inej spoločnosti.

4 Nové riešenie

Prieskum existujúcich riešení mi umožnil stanoviť základné prvky, ktoré musí reportovací systém spĺňať. Tieto vlastnosti skĺbené spolu s požiadavkami zadávateľom práce tvoria množinu funkcií, ktoré nová aplikácia OGreport ponúka, dokonca je aplikácia rozšírená aj o mnohé ďalšie funkcie na uľahčenie práce.

Aplikácia je založená na vzájomnej koordinácii vývojárov dopytov a pracovníkov, ktorým sú výsledné dopyty poskytnuté iba na dosadenie parametrov v reporte. To znamená, že pre nich nie je dôležitý text alebo štruktúra dopytu. Majú k dispozícii iba popis k čomu dopyt slúži a na základe toho ho môžu použiť v reporte na získanie žiadaných dát.

Nasleduje popis celého procesu vzniku výsledného reportu od vytvorenia dopytov po výsledný report. Je nutné podotknúť, že vzniknuté dopyty tým, že sú nositeľmi nastaviteľných údajov v podobe parametrov, nájdu široké uplatnenie a môžu byť použité viackrát. Následne reporty so zvolenými dopytmi ostávajú uložené v systéme a po dosadení parametrov môžu byť generované znovu, ľubovoľný počet krát. Z toho vyplýva, že pre generovanie reportu budú užívateľovi stačiť posledné dva kroky.

1. Určený, vyškolený užívateľ (správca reportov) znalý dopytovacieho jazyka a štruktúry dát vytvorí dopyty.

2. Správca reportov vytvorí report, do ktorého si môže vybrať ľubovoľný počet z prístupných dopytov.

3. Pre generovanie reportu je nutné, aby bol report validný. Potom je možné v prehľade reportov si zvoliť možnosť generovať.

4. Pri generovaní užívateľ vyplní príslušné parametre podľa dopytov, stlačí tlačidlo “Generovať” a exportuje sa mu report.

Nové riešenie zmienený proces zastrešuje a snaží sa ho čo najviac zjednodušiť s ohľadom na dodržanie oprávnení. Z popisu je zrejmé, že užívatelia budú musieť byť rozdelení podľa prístupu k jednotlivým funkciám.

4.1 Aktéri systému

Momentálne aplikácia disponuje troma rolami **administrátor užívateľov**, **administrátor zostáv** a **administrátor dopytov**. Práve aktuálna rola užívateľa, mu určuje oprávnenia v rámci aplikácie. To znamená, určuje s akou množinou funkcií môže disponovať a k akým častiam aplikácie má prístup.

Zaujímavé je, že s výnimkou role sú užívatelia rozdelení aj podľa lokalít. Zámerne je pomenovanie lokalita a nie firma. Je to z dôvodu, že napríklad pobočka v Prahe má iný prístup k reportom a dopytom, ako pobočka v Olomouci. Alebo v rámci Olomouca v jednej firme môžu mať technici iný prístup k reportom a dopytom než administratívny pracovníci.

Používateľ môže mať v danej lokalite viac rolí, pričom užívateľ a role sú na sebe navzájom nezávislé. Pre užívateľa je vždy aktuálna jedna lokalita a k nej prislúchajúca rola, v ktorej je prihlásený. Pokiaľ má prístup k iným lokalitám alebo v danej lokalite má viac rolí, má možnosť sa medzi nimi prepínať. To znamená, že aplikácia používa k identifikácii identifikátor užívateľa, aktuálnej lokality a aktuálnej role.

4.2 Vlastnosti aplikácie

Na základe predchádzajúceho popisu, zadania práce a inšpirácie existujúcimi riešeniami bol sformulovaný súhrn vlastností, ktoré aplikácia spĺňa. Ako každá dobrá webová aplikácia, tak aj OGreport by mal byť jednoduchý na ovládanie, bezpečný, rýchly a poskytovať spätnú väzbu.

Ďalej aplikácia disponuje základnými užívateľskými funkciami, akými sú prihlásenie, odhlásenie, obnova hesla, úprava profilu, kontaktovanie podpory, náhľad akcií užívateľa a zmena jazyka. Pre riadenie prístupu sú to možnosti, zmena lokality a zmena role. Práve podľa aktuálnej role má užívateľ k dispozícii ďalšie funkcie.

4.2.1 Správca dopytov

Správca dopytov je jediný aktér systému, od ktorého sa očakáva odborná znalosť vytvárania dopytov a zaškolenie, na správne vykonávanie funkcií v systéme, obzvlášť pri zadávaní dopytov a parametrov do systému. Tomuto aktérovi sú k dispozícii nasledovné akcie.

- Prehľad pre daného užívateľa viditeľných dopytov v systéme.
- Filtrovanie dopytov na základe názvu a zoradenie zobrazených dopytov podľa názvu a modelu.
- Vytvorenie dopytu, s touto možnosťou je spätých veľa funkcií. Je to z dôvodu bezpečnosti a hlavne uľahčenia práce pri zadávaní dopytu. V tomto riešení nikto nepožaduje, aby si správca dopytov zapamätal štruktúru zdroja dát alebo vždy bezchybne zadal dopyt.
  - Zvolenie si modelu dát v dialógovom okne, nad ktorým bude dopyt vytvorený. V prípade, že užívateľ má prístup iba k jednému modelu, tak je zdroj dát automaticky načítaný a ušetrí sa čas pri výbere.
  - Náhľad na štruktúru zvoleného modelu dát (tabuľky, atribúty a typy).
  - Definovanie parametrov ak sa v dopyte nejaké vyskytujú. Určiť či je parameter povinný, zadať jeho predvolenú hodnotu, stručný popis a typ. Možnosť použiť jeden parameter na viacerých miestach v dopyte.
  - Nastavenie autorizácie nad dopytom.

  - Spustenie dopytu ešte pred uložením a zistiť či sa vyhodnotí s príslušnými parametrami.
  - Poskytnutie náhľadu na výstupné dáta dopytu, pričom sa použijú predvolené hodnoty parametrov. Ak sú zadané inak, sú parametre pri vyhodnocovaní označené ako nepovinné.
  - Zvolenie možnosti vygenerovania reportu nad dopytom. Je to report, ktorý obsahuje iba tento dopyt a základné nastavenia sú u tohto reportu rovnaké ako u dopytu.

- Editovanie existujúceho dopytu, upravenie základných vlastností, parametrov a autorizácie nad dopytom.
- Vymazanie dopytu s upozornením.
- Kopírovanie existujúceho dopytu na tvorbu nového.

4.2.2 Správca reportov

Správca reportov je bežný užívateľ, ktorý nepotrebuje predchádzajúce zaškolenie na intuitívne používanie aplikácie, keďže pre generovanie reportov a dopĺňanie parametrov nepotrebuje žiadnu odbornú znalosť. Systém ho naviguje krok po kroku a vždy mu poskytne hlášku, či už pri úspešnej činnosti, alebo chybne zadanom údaji. Množina jemu dostupných funkcií zahŕňa:

- Prehľad všetkých viditeľných zostáv pre daného užívateľa.
- Filtrovanie reportov na základe kľúčových informácií (názov, validita, popis) a zoradenie vyhľadaných reportov podľa nich.
- Vytvorenie novej zostavy, definovanie základných informácií, poradia, zvolenie si dopytov pre report a oprávnení.
- Editovanie zostavy.
- Odstránenie alebo zneplatnenie zostavy.
- Zmeniť poradie zostavy, pričom pri zmene poradia u jedného reportu, musia aj ovplyvnené reporty zmeniť poradie. Napríklad, ak report s poradím 5 by bol posunutý na 2. miesto, tak by reporty 2, 3 a 4 museli tiež zmeniť svoje poradie.
- Vygenerovanie valídného reportu.
  - Zvolenie si príslušného formátu, hlavný formát XLSX a k dispozícii sú aj PDF a CSV.
  - Vyplnenie parametrov pre report.

4.2.3 Správca užívateľov

Jedná sa o doplnok do systému, nie primárnu funkcionalitu. Predovšetkým z praktického hľadiska, aby mal niekto práva nad užívateľmi a pre každú zmenu sa nemusel kontaktovať správca databázy a zasahovať priamo do nej. Do budúcnosti je predpoklad, že užívateľov si bude aplikácia brať z administratívnej databázy, kde sú špeciálne softvéry na ich správu. Zatiaľ systém poskytuje nasledovné provizórne funkcie na správu používateľov.

- Vytvorenie užívateľa, nastavenie jeho základných vlastností, rolí a lokalít.
- Editácia užívateľa, možnosť zmenenia základných informácií spolu s rolami a lokalitami, až na heslo. Pre zmenu hesla má užívateľ možnosť Zabudnuté heslo. Žiaden iný užívateľ nemá právo mu jeho heslo zmeniť. Ak si nepamätá svoj email, môže mu administrátor užívateľov pomôcť nastavením nového emailu.
- Vyhľadanie užívateľa podľa základných údajov a zoradenie výsledkov podľa abecedy.
- Detail používateľa, ako náhľad na informácie konkrétneho užívateľa spolu s akciami, aké zmeny v systéme vykonal.

5 Návrh aplikácie

Podľa požiadaviek z predchádzajúcej kapitoly som vykonala analýzu a návrh systému pomocou diagramov štruktúry a správania v jazyku UML a zostrojila model databázy. Vzniknuté návrhy tvoria základ implementácie OGreportu.

5.1 UML

Zjednotený modelovací jazyk (unified modeling language, UML) je druh grafickej notácie podporovaný nezávislým meta-modelom, ktorý umožňuje popisovať a navrhovať softvérové systémy, konkrétne systémy budované využitím objektovo orientovanej metodiky [1].

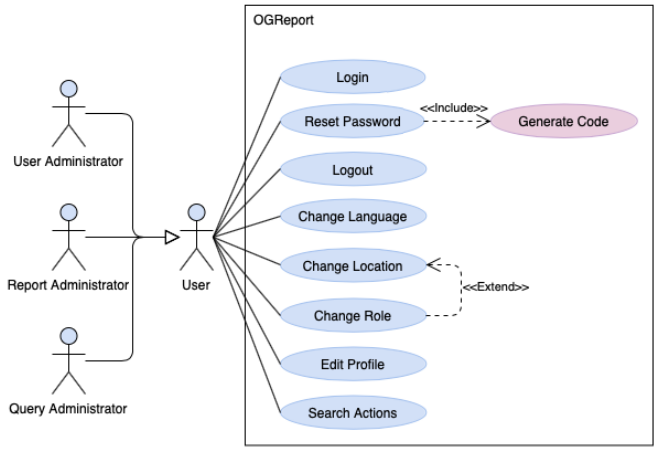
Ako jazyk vizuálneho modelovania sa UML vo veľkej miere spolieha na grafické konštrukcie. Schémy sú užitočné mnohými spôsobmi. Diagramy môžu zoskupiť príslušné údaje, čím sa minimalizuje čas strávený hľadaním požadovaných prvkov. Môžu zoskupovať aj informácie iba o jednom prvku. Diagramy podporujú veľké množstvo vnemov, ktoré sú pre človeka ľahko použiteľné. Stručne povedané, schematické znázornenia majú pozitívny vplyv na tri zložky spracovania ľudských informácií: vyhľadávanie, rozpoznávanie a odvodzovanie [2].

5.1.1 Diagramy prípadov použitia

Diagramy prípadov použitia zobrazujú správanie systému z pohľadu používateľa. Znázorňujú, čo užívateľ očakáva a akú sadu akcií bude systém poskytovať. Nepopisuje však bližšie informácie o tom, ako to má systém urobiť, alebo implementovať. Sú cenným nástrojom pre lepšie porozumenie funkčných požiadaviek na systém. Jedná sa o zachytenie kľúčových funkcií, aby bol rozumný počet prípadov použitia a zároveň rovnomerne pokrývali funkcionalitu systému.

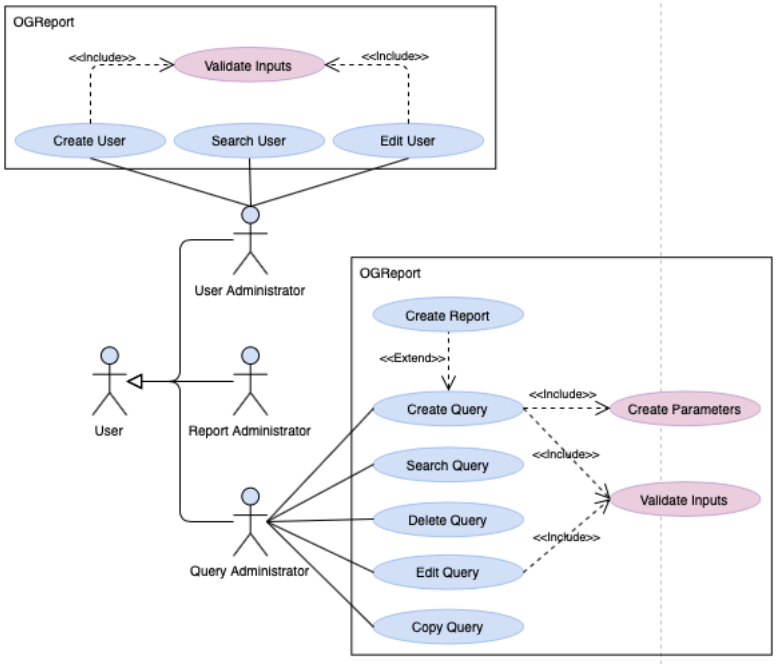
Tieto diagramy pozostávajú z prípadov použitia, reprezentujúcich sadu scenárov, ktoré sú zviazané spoločným užívateľom, nazývaným aktér. Aktér vyjadruje, v akom vzťahu je užívateľ so systémom a akú má rolu. Jedna osoba môže pôsobiť ako viac než jeden aktér. Ak jeden aktér zastrešuje všetky funkcie iného aktéra, tento dej sa označuje ako dedenie. Tento vzťah sa nazýva generalizácia a je znázornený prázdnou uzavretou šípkou smerom k predkovi. Užívateľ je spojený s prípadom použitia vzťahom nazývaným asociácia.

Na prvom diagrame sú znázornené základné vlastnosti systému, ktoré sú poskytnuté každému užívateľovi. S týmito prípadmi užitia je spojený užívateľ (User) asociáciou a ostatné role systému z neho dedia. Väzba ≪include≫ vyjadruje, že jeden prípad použitia zahŕňa druhý. Podľa prvého diagramu to znamená, že kód sa vygeneruje vždy pri zabudnutí hesla. Vzťah medzi akciami zmena lokality a zmena role sa nazýva ≪extend≫ a znamená, že prípad použitia rozširuje iný prípad, ale zároveň má sám o sebe význam. Z náčrtu to vyplýva, že užívateľ môže zmeniť rolu podľa potreby a pri zmene lokality sa automaticky poskytne zmena role, napríklad pokiaľ v druhej lokalite aktuálnu rolu nemá, musí sa mu automaticky zmeniť.



Obr. 3: UML, 1. diagram prípadov použitia

Ďalší diagram reprezentuje prípad použitia z pohľadu správcu dopytov a správcu užívateľov. U správcu užívateľov sú základné funkcie úpravy užívateľa spojené s patričnou validáciou. Komplikovanejšie a abstraktnejšie sú prípady použitia, čo sa týka správcu dopytov, keďže nemusí byť intuitívne zreteľné, ako má proces prebiehať. Lepšie povedané akú interakciu z pohľadu užívateľa zahŕňa. Z toho dôvodu uvediem textový popis prípadu použitia vytvorenia dopytu. Zaujímavá na tomto diagrame je väzba ≪extend≫ medzi vytvorením dopytu a vytvorením reportu. Je to špeciálna funkcia, aby správca dopytov mohol priamo nad jedným dopytom vytvoriť report, čím ušetrí prácu a čas správcovi reportov.

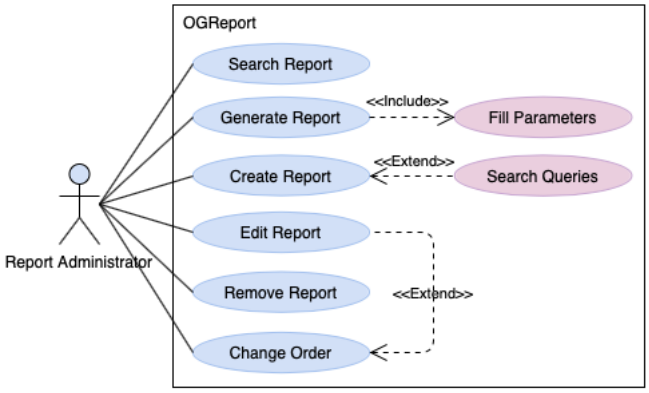


Obr. 4: UML, 2. diagram prípadov použitia

Tabuľka 1: Text prípadu použitia vytvorenia dopytu

Vytvorenie dopytu
Aktéri: Správca dopytov
Vstupné podmienky: Užívateľ musí byť autentifikovaný. K systému musí byť pripojený dátový model, nad ktorým je možné dopyt vytvoriť.
Výstupné podmienky: Dopyt bol úspešne vytvorený a evidovaný v systéme spolu s príslušnou autorizáciou a parametrami.
Hlavný úspešný scenár: 1. Užívateľ si z hlavného menu vyberie možnosť vytvoriť dopyt. 2. Zvolí si dátový model, nad ktorým chce pracovať. 3. Vyplní základné údaje o dopyte. 4. Zvolí si oprávnenia dopytu. 5. Vyplní text dopytu. 6. Klikne na tlačidlo "Vytvoriť" dopyt. 7. Dopyt bol pridaný do systému.
Alternatívne scenáre: 1) Užívateľ kopíroval dopyt. 1. Všetky údaje sa automaticky vyplnia podľa kopírovaného dopytu. 2 a) Užívateľ nezadal dátový model. 1. Systém upozorní na chybu. 2 b) Užívateľ má prístup iba k jednému dátovému modelu. 1. Dátový model sa automaticky vyplní. 3) Užívateľ chybne zadal údaje o dopyte. 1. Systém upozorní na chybu. 5) V prípade ak dopyt obsahuje parametre. 1. Užívateľ určí typ parametru. 2. Prípadne vyplní či je parameter povinný a či má predvolenú hodnotu. 6 a) Pred vytvorením má možnosť dopyt spustiť. 1. Systém vyhodnotí dopyt a poskytne výsledok dopytu. 6 b) Užívateľ nevyplnil dopyt. 1. Systém upozorní na chybu. 6 c) Užívateľ zadal chybný dopyt. 1. Systém mu poskytne spätnú väzbu o zistenej chybe.

Diagram na obrázku 5, reprezentuje pohľad na systém z role správcu reportov. Jeho hlavnou úlohou je nad dopytmi vytvárať reporty a tie následne generovať. Zmeniť poradie reportu môže samostatne, alebo počas editovania informácií o reporte. Za zmienku stojí rozdiel medzi zadaním parametrov u správcu dopytov a správcu reportov. Kým správca dopytov parametre definuje a určuje ako majú byť použité, správca reportov iba vyplní do parametrov hodnoty pri generovaní.



Obr. 5: UML, 3. diagram prípadov použitia

5.1.2 Diagramy tried

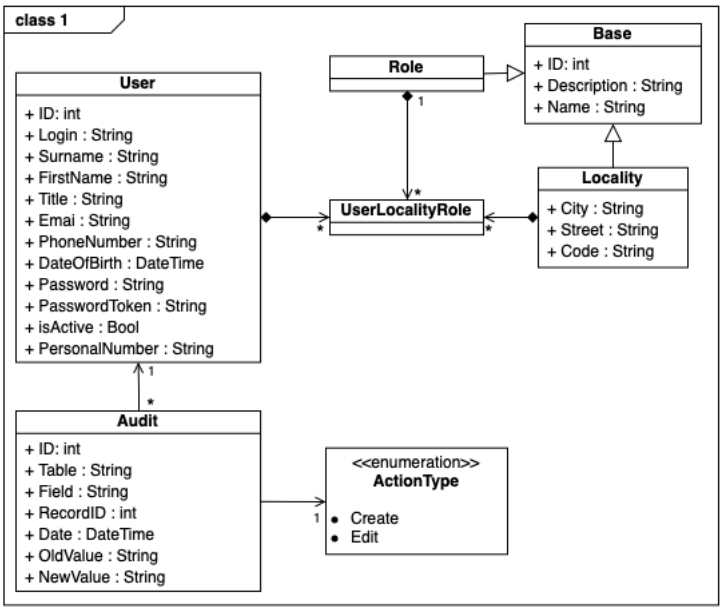
V minulej sekcii pri prípadoch použitia aktér interaguje so systémom. Údaje potrebné k tejto interakcii sa zachytávajú práve pomocou diagramov tried. Diagramy tried popisujú štruktúru systému, znázorňujú triedy a vzťahy medzi nimi. Môžu obsahovať aj balíky, objekty a rozhrania. Diagramy tried už priamo odrážajú ako bude implementácia systému vyzerať [3].

Trieda je logická skupina vlastností a metód, ktoré vystupujú v systéme celistvo a jednotne. V diagrame sa trieda značí obdĺžnikom, obsahuje názov triedy jej atribúty a metódy. Základné vzťahy medzi entitami sú nasledujúce:

- **asociácia** - všeobecný vzťah, kedy triedy navzájom volajú svoje metódy. V diagrame sa znázorňuje plnou čiarou.
- **generalizácia** - dedenie, prázdna šípka (trojuholník) je pri rodičovi. Značí, že podradená trieda dedí všetky atribúty, operácie a vzťahy od rodiča.
- **agregácia** - používa sa vo význame ako časť niečoho, kde jeden objekt je súčasťou iného, ale zároveň má sám význam. Grafickou reprezentáciou je prázdny kosoštvorec.
- **kompozícia** - oproti agregácii vyjadruje silnú väzbu, kde jedna trieda je závislá na životnom cykle druhej, grafickou reprezentáciou je plný kosoštvorec.

Prvý diagram tried zobrazuje štruktúru tried spojených s užívateľom. Lokalita aj rola majú spoločné atribúty ako identifikátor, meno a popis. UserLocalityRole je trieda silne závislá na User, Locality a Role. Každá inštancia UserLocalityRole je napojená iba na jednu lokalitu, rolu a jedného užívateľa. Užívateľ, rola alebo lokalita však môžu mať viac inštancií UserLocalityRole. Reprezentuje to, že užívateľ môže pôsobiť vo viacerých lokalitách a v rámci lokalít môže mať viac rolí.

Trieda Audit značí činnosť typu ActionType, ktorú užívateľ vykonal. Jednému užívateľovi pritom prináleží viac akcií, ktoré v rámci systému učinil.

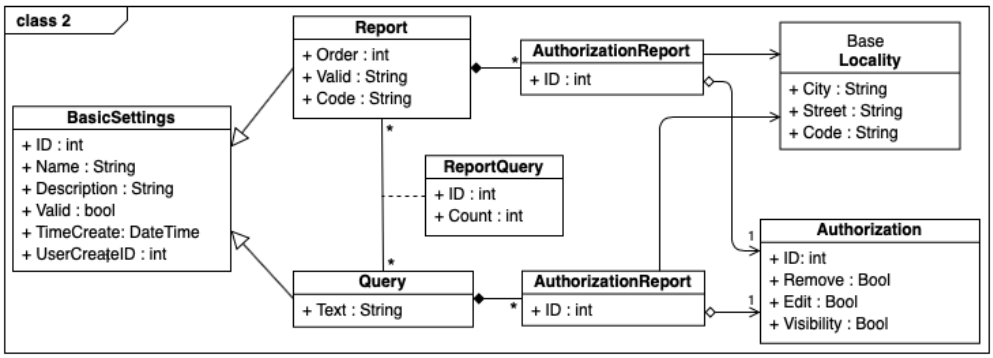


Obr. 6: UML, 1. diagram tried

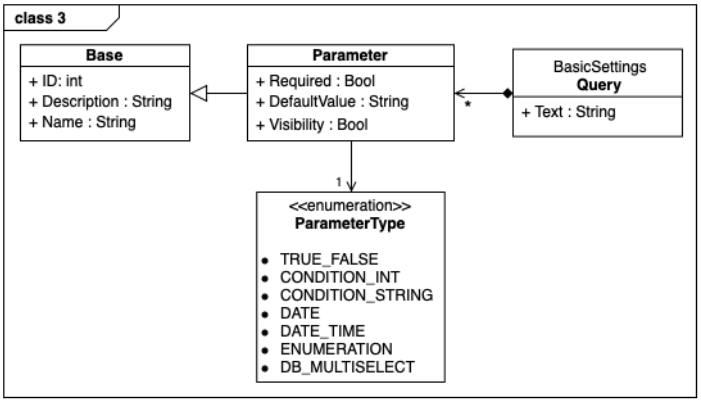
Nasledujúci obrázok reprezentuje príklad diagramu tried, ktorý je ovplyvnený zvolenou implementáciou. Mohlo by sa zdať, že triedy AuthorizationReport a AuthorizationQuery môžu byť nahradené jednou. Táto náhrada nie je možná, pretože ID od reportu a dopytu sú síce unikátne pre svoju triedu, môže sa ale stať, že jeden report bude mať rovnaké ID ako dopyt.

Medzi triedami Report a Query je asociačná trieda ReportQuery, ktorá obsahuje atribút Count, značiaci aký je počet jednej inštancie dopytu v reporte. Jeden report môže mať viac rovnakých dopytov, ale aby sa neopakovala väzba, iba sa nastaví Count.

Kompozícia medzi Query a AuthorizationQuery vyjadruje, že Query môže mať viac inštancií AuthorizationQuery, ale inštancia AuthorizationQuery smie byť súčasťou iba jedného dopytu. Rovnaké platí pre triedy AuthorizationReport a Report.



Obr. 7: UML, 2. diagram tried



Obr. 8: UML, 3. diagram tried

Posledný diagram je venovaný vzťahu parametrov a dopytov. Na diagrame je znázornené, že parameter nesie rovnaké vlastnosti ako Base a zároveň je rozšírený o ParameterType. Väzba medzi parametrom a dopytom znamená:

- inštancia parameter môže byť obsiahnutá iba v jednom dopyte,
- dopyt môže obsahovať viac parametrov,
- inštancia parametru môže vzniknúť, iba ak existuje dopyt,
- ak dopyt zanikne, zaniknú aj jeho parametre.

5.2 Dátový model

Dátový model definuje štruktúru dát v relačnej databáze a vzťahy medzi nimi. Namiesto tried sa základný prvok nazýva entita. Pre OGreport je dátový model riešený ako samostatná jednotka, ktorá sama bude organizovať konzistenciu svojich dát a integritu. Doménová integrita je zabezpečená obmedzeniami atribútov, stanovením dátového typu a obmedzeniami pre rozsah hodnôt. Entitná integrita je zabezpečená primárnym kľúčom ID v každej tabuľke a referenčnú integritu zabezpečujú cudzie kľúče. Schéma databázy obsahuje entity:

Audit – záznam súvisiaci s činnosťou užívateľa pri modifikovaní objektov,

Authorization – základné oprávnenia,

AuthorizationReport – práva na zostavu,

AuthorizationQuery – oprávnenia pre dopyty, podľa lokalít a rolí,

Locality – geografické alebo logické umiestnenie,

Parameter – parametre pre dopyty,

Query – informácie o dopyte,

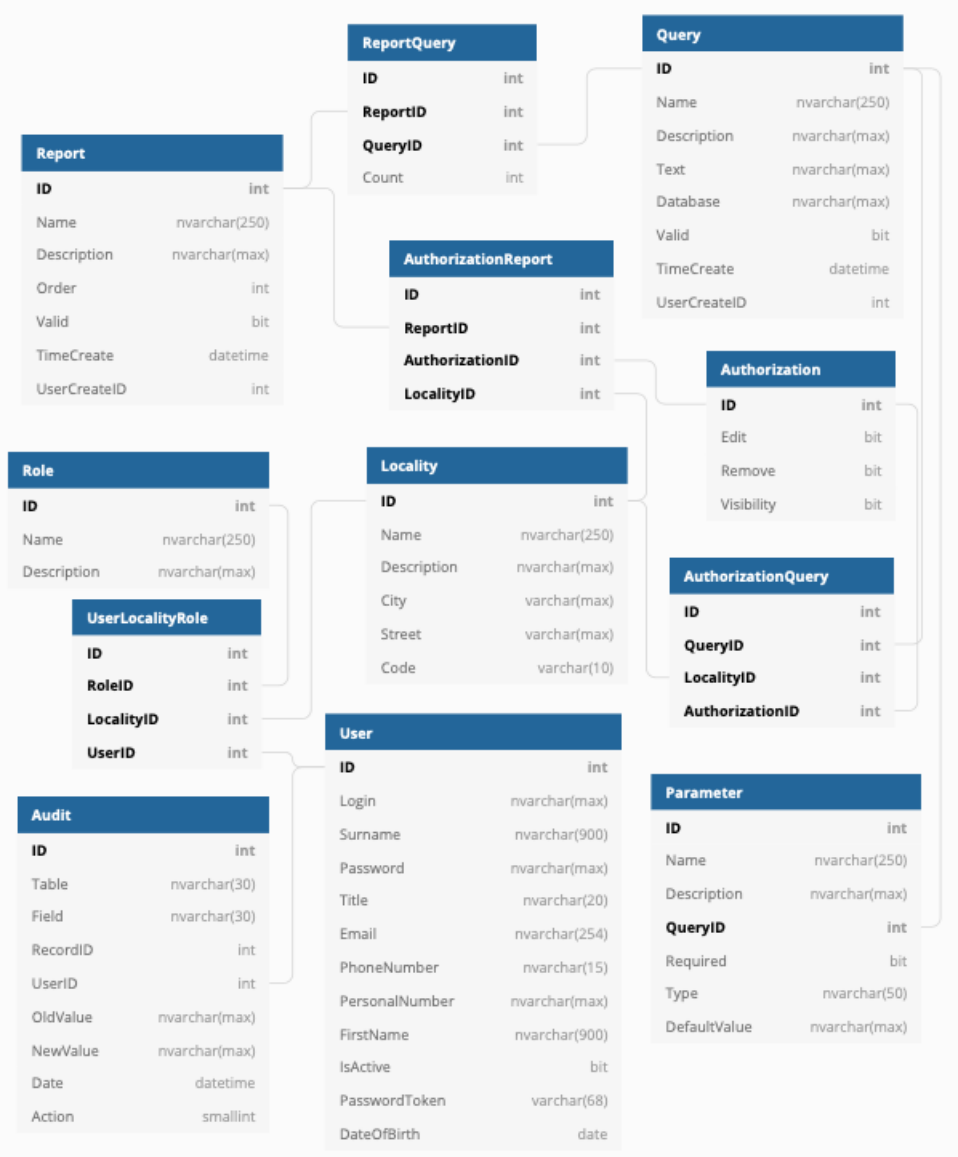
Report – obsahuje základné informácie o zostave,

ReportQuery – umiestnenie dopytov v reportoch a ich násobnosť,

Role – zoznam užívateľských rolí,

User – užívateľské údaje a dáta potrebné na identifikáciu užívateľa v systéme,

UserLocalityRole – evidencia, ktorý užívateľ môže vystupovať v akej roli pre určitú lokalitu.



Obr. 9: Schéma databázy Generator_of_reports

6 Použité technológie

Aplikácia bola vyvíjaná v C# .NET nad MVC architektúrou v Microsoft Visual Studiu 2019 s pripojenou databázou v MS SQL. Pre tvorbu efektov a validáciu na klientskej strane je primárne využitý JavaScript a od neho odvodené frameworky. V nasledujúcich sekciách sú tieto technológie stručne popísané spolu so zvyšnými využitými nástrojmi.

6.1 ASP.NET MVC

ASP.NET MVC je open-source framework od Microsoftu pre tvorbu webových aplikácií. Kombinuje techniky z agilného vývoja platformy ASP.NET a uplatňuje všeobecný vzor Model-View-Controller (MVC) [4].

6.1.1 ASP.NET

ASP.NET je server-side framework rozširujúci platformu .NET o nástroje a knižnice určené pre vytváranie dynamických webových stránok, aplikácií a služieb. Nasleduje niekoľko základných vecí, ktoré ASP.NET pridáva k platforme .NET.

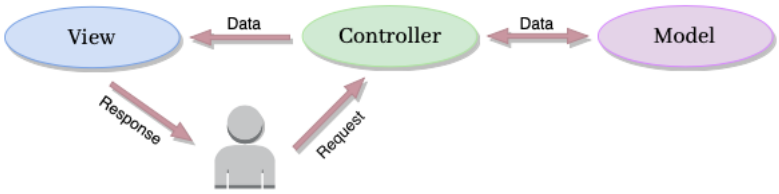
- Základný rámec pre spracovanie webových požiadaviek v C# alebo F#.
- Syntax Razor, na vytváranie dynamických webových stránok pomocou C#.
- Systém overovania obsahujúci knižnice, databázu a šablóny na spracovanie prihlásení, vrátane viacfaktorového overovania a externého overovania.
- Rozšírenia editora, ktoré poskytujú zvýraznenie syntaxe, dokončenie kódu a ďalšie funkcie špeciálne pre vývoj webových stránok [5].

6.1.2 MVC architektúra

Model-View-Controller je architektonický programátorský vzor. Predstavuje elegantné rozdelenie aplikácie a vytvorenie abstrakcie medzi užívateľským rozhraním, logikou a dátovým modelom.

- Model reprezentuje množinu dát, ktorú na frontende užívateľ vidí a pracuje s ňou. Tiež popisuje reštrikcie ako sa dáta môžu meniť a ako sa s nimi môže manipulovať. Na jeho popis sa používajú triedy v C#.
- View alebo pohľad definuje zobrazenie užívateľského rozhrania. Používa sa na zobrazenie modelu a dát prístupných užívateľovi. View pozostáva z HTML, CSS a špeciálnej syntaxe Razor[2].
- Controller alebo radič, odvodený od riadiacej logiky, spracúva komunikáciu medzi užívateľom a aplikáciou samotnou.

[2]Syntax ASP.NET používaná na vytváranie dynamických webových stránok v programovacích jazykoch C# a VB.NET.



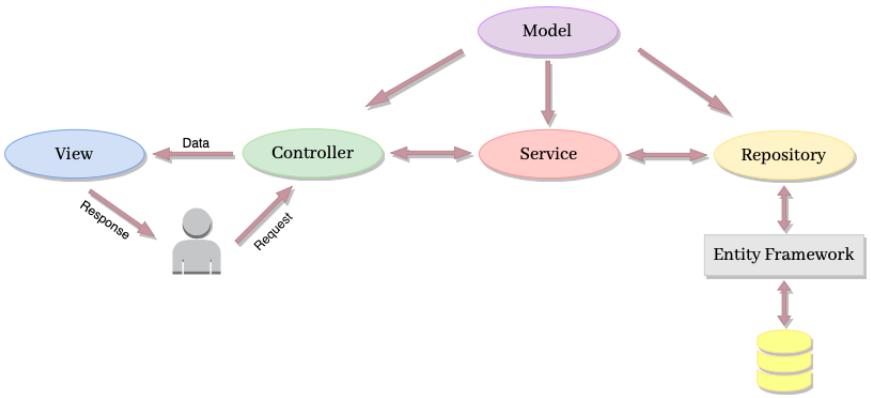
Obr. 10: MVC pattern

6.1.3 ASP.NET MVC 5

Prvá verzia MVC 1.0 nad .NET Frameworkom 3.5 vyšla v roku 2009 a priniesla dodnes základné vlastnosti, akými sú routing, auto binding, možnosť použitia HTML Helpers a Ajax Helpers. od .NET 3.5 bolo pridaných veľa nových funkcií. Verzia ASP.NET MVC 5 bola vydaná spoločne s Visual Studiom 2013. U ASP.NET MVC 5 boli prelomové vlastnosti:

- One ASP.NET alebo inak povedané "One size fits all". Väčšinou platí, že jeden typ projektu nemôže vyhovieť všetkým potrebám. V predchádzajúcich ASP.NET verziách sa bolo nutné hneď na začiatku rozhodnúť, aký typ projektu sa bude vyvíjať (MVC aplikácia, Web Forms aplikácia, ...). Neskoršie zmeny boli náročné až nemožné.
- Identita ASP.NET vo verzii MVC 5 zažila kompletné prerobenie autentifikačného systému a členstva. Tento nový spôsob dovoľuje ukladať prídavné údaje o užívateľoch ako atribúty do triedy reprezentujúceho užívateľa. Pridáva možnosť písať Unit testy nad autentifikáciou a zlepšiť tak testovateľnosť systému. Rozširuje špecifiká identity na možnosť použitia iných údajov, než role používateľa.
- Autentifikačné filtre povoľujú reštrikcie na prístup ku kontrolérom a akciám alebo inej logike. Tu je dôležité rozlišovať autentifikáciu (kto je užívateľ) a autorizáciu (čo overený užívateľ môže robiť). Nové autentifikačné filtre sú vykonané pred autorizačnými filtrami.
- ASP.NET Scaffolding je proces generovania kódu na základe modelu tried. Túto vlastnosť má ASP.NET od verzie jedna, ale bola limitovaná iba na MVC projekty.
- Využitie atribútov pri smerovaní je nová možnosť použitia anotácií na kontroléry alebo metódy.

Poslednou verziou vo vývoji je MVC 5.2. Pôvodne bola vydaná aj MVC 6 ako súčasť ASP.NET 5 a mala byť považovaná za ďalšiu verziu, ale kvôli veľkým zmenám priamo v základoch (nový projektový systém, nový konfiguračný systém, nahradenie System.Web odľahčeným HttpContext) bolo rozhodnuté, že je lepšie ju vydať ako nový framework a zmenili názov na ASP.NET Core [5].



Obr. 11: ASP.NET MVC Repository pattern so servisnou vrstvou

6.2 Repository Pattern

Slúži na vytvorenie abstraktnej vrstvy medzi business logikou aplikácie a priamym prístupom k dátam. Použitie repository pattern poskytuje výhody akými sú: menej redundantného kódu na prístup k dátam a centrálne spravovateľný kód na prístup k databáze. Hlavnou výhodou je izolácia logiky prístupu k dátam a business logiky. Ak v jednej z nich urobíte zmeny, nenaruší to chod aplikácie.

6.3 C#

C# je vysokoúrovňový, silne typový, imperatívny, deklaratívny, objektovo orientovaný jazyk. Microsoft založil C# na jazykoch C++ a Java a bol predstavený spolu s Visual Studio. Pomocou C# je možné vytvárať širokú škálu projektov, ktoré možno použiť na zostavenie kompletnej aplikácie klient-server. C# poskytuje kompletné prostredie pre vývoj aplikácií. C# spája silu C, objektovo orientované vlastnosti C++, grafické rozhranie VB a ako Java sa navyše kompiluje do bajtkódu [6].

6.4 SQL a MS SQL

Počítačový program slúžiaci na riadenie databázy sa nazýva systém správy databázy (DBMS). Microsoft SQL Server (MS SQL) je relačný DBMS. Tento systém je skonštruovaný pre základnú funkciu ukladania a načítania údajov podľa požiadaviek iných aplikácií.

Structured Query Language (SQL) je nástroj na interakciu s databázou. Zabezpečuje organizáciu, správu a načítanie údajov uložených v relačnej databáze. Na rozdiel od počítačových jazykov, ako C, C ++ alebo Java, je SQL databázový subjazyk, pozostávajúci z príkazov špecializovaných na úlohy správy databázy. Príkazy SQL je možné vložiť do iného jazyka, aby sa rozšíril na použitie prístupu k databáze. SQL sa taktiež líši od ostatných jazykov, tým že predovšetkým popisuje, čo užívateľ chce, aby počítač spravil, namiesto toho ako by to mal urobiť.

SQL je oveľa viac ako dopytovací nástroj, aj keď získavanie údajov je stále jednou z jeho najdôležitejších funkcií. SQL sa používa na riadenie všetkých funkcií, ktoré DBMS poskytuje svojim používateľom, vrátane nasledujúcich.

- Definícia údajov SQL umožňuje používateľovi definovať štruktúru a organizáciu uložených údajov a vzťahy medzi uloženými údajovými položkami.
- Načítanie údajov SQL umožňuje užívateľovi alebo aplikačnému programu načítať uložené údaje z databázy a používať ich.
- Manipulácia s údajmi SQL umožňuje používateľovi alebo aplikačnému programu aktualizovať databázu pridaním nových údajov, odstránením starých údajov a úpravou predtým uložených údajov.
- Riadenie prístupu SQL sa môže použiť na obmedzenie schopnosti používateľa načítať, pridávať a upravovať údaje, čím chráni uložené údaje pred neoprávneným prístupom.
- Zdieľanie údajov SQL sa používa na koordináciu zdieľania údajov súčasnými používateľmi, čím sa zabezpečí, že zmeny vykonané jedným používateľom, neúmyselne nevymažú zmeny vykonané takmer v rovnakom čase iným používateľom.
- Integrita údajov SQL definuje obmedzenia integrity v databáze a chráni ju pred poškodením v dôsledku nekonzistentných aktualizácií, alebo zlyhaní systému.

Pre načítanie údajov z databázy, sa zadá požiadavka v jazyku SQL. Systém DBMS spracuje požiadavku SQL, načíta požadované údaje a vráti ich. Tento proces vyžiadania údajov z databázy a prijímania výsledkov sa nazýva databázový dopyt (query), odtiaľ pochádza názov Structured Query Language [8].

6.5 JavaScript

JavaScript je programovací jazyk webu. Drvivá väčšina webových stránok používa JavaScript a všetky moderné webové prehliadače obsahujú interprety JavaScriptu, vďaka čomu je JavaScript najbežnejším programovacím jazykom v histórii. JavaScript je vysokoúrovňový univerzálny dynamický interpretovaný programovací jazyk bez atribútov, ktorý je vhodný pre objektovo orientované a funkčné štýly programovania. JavaScript odvodzuje svoju syntax z jazyka Java[3], svoje prvotriedne funkcie zo Scheme[4] a prototypové dedenie zo Self[5] [7].

[3]Objektovo orientovaný multiplatformový programovací jazyk s automatickou správou pamäte. Je považovaný za bezpečný, jednoduchý a rýchly jazyk, čím si vyslúžil vysokú popularitu.

[4]Jednoduchý funkcionálny jazyk, kde väčšina syntaxe je odvodená z primitívnejších foriem.

[5]Prototypový objektovo orientovaný programovací jazyk, virtuálny stroj a prostredie.

- jQuery je JavaScriptová knižnica, ktorej účelom je zjednodušiť použitie JavaScriptu na webových stránkach, respektíve prepojenie HTML a JavaScriptu. Poskytuje výber DOM elementov, vytváranie efektov a animácií, použitie AJAX, manipuláciu s CSS, spracovanie udalostí, a vývoj Ajax aplikácií.

6.6 CSS a HTML

K triáde technológií, ktoré sú nevyhnutné pre vývojárov internetových stránok patrí JavaScript, HTML a CSS. HTML je značkovací jazyk pre špecifikáciu obsahu webových stránok. CSS sú kaskádové štýly pre prezentáciu a formátovanie webových stránok a JavaScript pre špecifikáciu správania webových stránok.

6.7 NuGet Package Manager

Správca balíkov alebo balíkový systém poskytujúci kódy iných vývojárov, ako open-source. Niektoré využité balíky sú bližšie popísané v nasledujúcom texte. Ich celý zoznam aj s popisom je k nahliadnutiu priamo vo Visual Studiu cestou *Tools > NuGet Package Manager > Manage NuGet Packages for Solution > Insatelled*, alebo v riešení v súbore *packages.config*.

`EPPlus`
: EPPlus je knižnica na správu tabuliek Office Open XML. Umožňuje vytvárať, čítať a upravovať súbory programu Excel v ľubovoľnej aplikácii ASP.NET MVC bez akýchkoľvek závislostí na balíku Microsoft Office.

`Entity Framework`
: Predstavuje objektovo-relačný mapovací rámec pre .NET od Microsoftu. Umožňuje integrovanie aplikácie s relačnou databázou, poskytuje abstrakciu nad dátovým modelom, kde s dátami v databáze pracujeme ako s POCO objektmi .NET.

`jQuery UI`
: Je sada interakcií užívateľského rozhrania, efektov, widgetov a tém, postavená nad knižnicou jQuery. Zahrňuje prvky ako datepicker, dataTable, progressbar, selectmenu a ďalšie.

`Microsoft.AspNet.Razor`
: Balík obsahuje behové zostavy slúžiace na rýchle a čisté kombinovanie serverového kódu s HTML.

`toastr`
: JavaScriptová knižnica priamo závislá na jQuery (>= 1.6.3) pre oznámenia.

`Web Grease`
: Je sada nástrojov na optimalizáciu javascriptových, css súborov a obrázkov.

7 Štruktúra projektu

Štruktúra projektu je ovplyvnená zvolenou technológiou, dizajnovým vzorom pre abstrakciu medzi logickými časťami a navyše dodržuje sadu dizajnových princípov SOLID[6]. Nasleduje stručný popis jednotlivých súborov aplikácie. Cesta k nim je na priloženom CD *src > GeneratorOfReports > GeneratorOfReports*.

App_Start/
: Tento priečinok obsahuje niektoré kľúčové konfiguračné súbory, ktoré sa vykonajú pri spustení aplikácie a sú použiteľné pre celú aplikáciu.

  - BundleConfig.cs – používa sa na vytvorenie a registráciu javasriptových a CSS balíkov,
  - FilterConfig.cs – obsahuje globálne filtre pre všetky akcie a kontroléry v aplikácii,
  - RouteConfig.cs – slúži k zaregistrovaniu vzorov pre smerovacie cesty.

Attributes/
: Zahŕňa triedy používané ako validačné doplnky k anotáciám v modeloch a ku globálnym filtrom. Napríklad *JSTranslatorAttribute.cs* je filter nad javascriptovými súbormi, ktorý slúži na preloženie výrazov, pretože MVC NET pre skripty viacjazyčnosť pomocou *Resources* nepodporuje.

Authentication/
: Obsahuje základné triedy určené k overeniu identity a zaobaleniu informácií o overenom užívateľovi. Tieto triedy sú nevyhnutné pre správne fungovanie autentifikácie na základe rolí. Patria sem konštanty pre role, serializačný model, triedy pre členstvo a identitu.

Content/
: Obsahuje statické súbory, akými sú obrázky, ikony a kaskádové štýly.

ContextManagement/
: Obsahuje obyčajné objekty reprezentujúce kontexty databázy a výčtovú triedu zastrešujúcu výpis sprístupnených kontextov a metód pre ne. Bližší účel týchto tried je popísaný v kapitole venovanej spracovaniu kontextov.

Controllers/
: Kontroléry slúžia na spracovanie URL requestov a na ich základe poskytujú spätnú väzbu. Každému kontroléru prináleží priečinok pohľadov s rovnakým názvom. Ďalej platí, že každý radič zastrešuje operácie na spracovanie požiadavky týkajúcej sa jeho účelu. Napríklad *ReportController.cs* odpovedá na požiadavky týkajúce sa reportov, ich vytvorenie, mazanie, výpis a podobne. Výnimkou je *BaseController.cs*.

[6]Je sada princípov návrhu pri vývoji objektovo orientovaného softvéru. SOLID je skratka odvodená od: Single Responsibility Principle, Open/Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion.

- BaseController.cs – Dedia z neho všetky kontroléry. V konštruktéri inicializuje hodnotu *UnitOfWork*. Tá je dôležitá, keďže vďaka nej vzniká následne kaskádové prepojenie do nižších vrstiev aplikácie.

Models/

Obsahuje triedy reprezentujúce jednotlivé objekty namapované z databázy a business objekty. Súbor *.edmx* definuje koncepčný model, uloženie modelu a mapovanie medzi modelmi. Entity Framework Designer umožňuje grafickú vizualizáciu modelov a väzieb medzi nimi. *GORModel.edmx* je model pre databázu aplikáciu OGreport.

- UnitOfWork.cs – Obal nad repozitármi umožňujúci vytváranie transakcií medzi rôznymi typmi entít. Je to ďalšia časť abstrakcie medzi obchodnou logikou a prístupom k dátam. Týmto spôsobom implementácie pracujú všetky repozitáre nad jedným kontextom. Hlavnou výhodou je konzistencia dát a pohľad na vykonané operácie ako na jednu transakciu. Napríklad, ak repozitáre pre report a dopyt nie sú zastrešené UnitOfWork, tak si každý vytvorí a udržiava vlastnú inštanciu kontextu. Obe inštancie majú vlastný zoznam zmien záznamov. Ak by pri operácii vytvorenia dopytu so single reportom jedna z inštancií kontextu zlyhala a druhá úspešne dokončila, malo by to za následok databázu v nekonzistentnom stave.

ParameterManagement/

Priečinok pre správu parametrov v dopyte. Obsahuje zoznam tried pre typy parametrov. Podrobne je tomu venovaná kapitola týkajúca sa parametrov.

ReportGenerator/

Obsahuje formátory a k nim príslušné triedy na spracovanie dát do výslednej podoby reportu.

- Generator.cs – Univerzálna statická trieda nad všetkými typmi prípustných formátov reportu. Táto trieda umožňuje jednoduché pridanie nového typu reportu. Pridanie formátu by spočívalo vo vytvorení triedy (napr. TXT) s kódom na generovanie, a do metódy Generate v triede Generator.cs sa doplní prípad pri zvolení si formátu .txt.

Repositories/

Vrstva zaisťujúca abstraktnú bariéru medzi dátovou (DAL, data access layer) a business logikou (BAL, business access layer) pre vykonávanie CRUD operácií.

- Repository.cs – Trieda na implementáciu rozhrania *IRepository.cs*. Obsahuje generické databázové operácie pre akúkoľvek entitu domény. Dedia z neho všetky repozitáre.

Resources/
Obsahuje XML súbory s príponou .resx reprezentujúcich jazyky respektíve príslušné preklady slov aplikácie.

Scripts/
Tu sa nachádzajú javascriptové knižnice a skripty (.js).

Services/
Servisné triedy reprezentujúce servisnú vrstvu, tá sprostredkováva komunikáciu medzi kontrolérmi a repozitármi. Servisná vrstva zodpovedná za komplexnú logiku, často obsahuje mnoho validačných metód a formátorov, aby čo najviac odtienila kontroléry a posunula už spracované dáta na DAL.

Utilities/
Statické triedy poskytujúce kolekciu pomocných funkcií, používaných v rôznych častiach aplikácie na doplnenie funkcionality, manipuláciu s dátami a zjednodušenie činnosti. Sú to triedy *FormattingUtility.cs*, *LocalizationUtility.cs*, *MailUtility.cs*, *NotificationUtility.cs* a *TypeUtility.cs*.

- TypeUtility.cs - Výčtová trieda používaná pre prácu s typmi namiesto výčtových typov. Zlepšuje to redundantnosť kódu a nahradzuje príkazy riadenia toku, kontrolujúce výčtové hodnoty.

ViewModels/
ViewModels sú tu umiestnené triedy pre modely pohľadu, predstavujú údaje, ktoré sa ukážu na zobrazení.

Views/
Obsahuje súbory HTML s razor syntaxou (.cshtml) zodpovedné za výzor výslednej stránky po vygenerovaní. Pre každý radič je samostatný priečinok a v priečinku *shared* sa nachádzajú čiastočné pohľady[7].

Global.asax
Pomocou *Global.asax* môže programátor písať kód spustiteľný na úrovni aplikácie. Napr. pri udalostiach Application_start, Application_error, ...

Packages.config
Súbor na správu NuGet balíkov. Zaznamenáva aké balíky a ich verzie boli v aplikácii nainštalované.

Web.config
Obsahuje konfiguráciu aplikácie.

[7]Partial view je značkovací súbor Razor, svoj výstup vykresľuje v rámci výstupu iného značkovacieho súboru.

8 Programátorská dokumentácia

Pri vytváraní aplikácie som sa snažila poučiť z chýb existujúceho riešenia a implementovať systém jednoduchý na ovládanie, zároveň s vysokou mierou funkcionality pre uľahčenie práce a rýchlou odozvou. Z programátorského hľadiska bol môj prvý cieľ, aby vzniknutá aplikácia bola ľahká na udržateľnosť a úpravu. Je to hlavne z dôvodu, pretože sa jedná o klientský systém a často sa stáva, že časom sa na danú aplikáciu zmenia nároky a bude nutnosť doplnenia funkcií.

8.1 Spracovanie dátových modelov

V tejto sekcii sa zaoberám pripojením dátového modelu IS, aby sa nad ním mohli vytvárať dopyty na výber dátových položiek pre generovanie reportu, pričom nás zaujíma fyzický dátový model, reprezentujúci najnižšiu úroveň dát, ktorá popisuje konkrétnu reprezentáciu dát v databáze (entity sú tabuľky, atribúty sú stĺpce v tabuľkách).

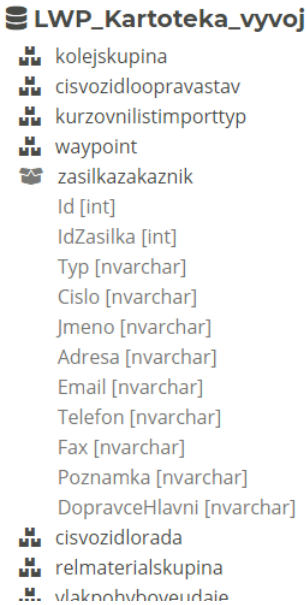
Za fyzický dátový model vhodný pre aplikáciu OGreport sa považuje množina štruktúrovaných dát ako kolekcia záznamov, kde je možné pomocou dopytovacieho jazyka získavať potrebné údaje.

8.1.1 Implementácia

Pre pripojenie dátového modelu som zvažovala dve cesty. Prvá možnosť bola využiť zmapovanie databázových tabuliek do kontextu pomocou Entity Framework. Výhodou je, že by sa s pripojeným modelom pracovalo rovnako ako s kontextovou triedou. Nevýhodou tohto riešenia je významné zväčšenie veľkosti aplikácie už pri stredne veľkých databázach. Ďalej nutnosť, aktualizovať zmapovaný model vždy pri zmene štruktúry originálnej databázy. To je časovo náročné, pokiaľ nie je známa konkrétna entita, kde nastala zmena. Ďalšou, jednou z najväčších nevýhod je, že by bola zmapovaná databáza súčasťou riešenia aplikácie, tak by v prípade výpadku jednej databázy alebo nekorektnosti modelu (originálny zdroj by bol zmenený napr. premenovanie stĺpca, zmena dátového typu), prestala ísť celá aplikácia.

Iné riešenie, najčastejšie používané pre získavanie dát zo vzdialených zdrojov, je pripojenie pomocou pripojovacieho reťazca v reálnom čase behu aplikácie na vyžiadanie. Výhody sú nasledovné: je možné, aby dáta a štruktúra zdroja boli zmenené a aplikácia by fungovala ďalej bez nutnej aktualizácie. Jednoduché a rýchle pridanie a odobranie zdroja modelu dát. Nevýhoda je, že pripájanie trvá dlhšiu dobu ako u prvej možnosti.

Prvé riešenie by bolo výhodné, keby som chcela dané dáta zo zdroja aj upravovať, ale pre získavanie dát je to nadbytočné. Z tohto dôvodu je implementácia realizovaná druhou cestou, kde je zoznam prípustných modelov evidovaný pomocou výčtovej triedy a každý pripojený model je osobitne definovaný ako inštancia triedy *ContextType.cs*. To umožňuje v budúcnosti zaviesť reštrikcie nad jednotlivými modelmi.



Obr. 12: Zobrazenie časti pripojeného modelu LWP_Kartoteka_Vyvoj

8.1.2 Pridanie nového modelu

Pre pridanie nového dátového modelu do riešenia, nad ktorým by aplikácia mohla pracovať postačuje do triedy *ContextType.cs* pridať správne definované pripojenie connectionString[8]. *ContextType.cs* dedí z *ContextWrapper.cs*, kde sú všetky funkcie pre prácu nad modelom v rámci OGreport aplikácie.

```
1  public static readonly ContextType NEW_MODEL = new ContextType(name
       , connectionString);
```

Zdrojový kód 1: Pridanie spojenia na dátový model

8.1.3 Výhody a nevýhody

Hlavnou výhodou implementácie je jednoduché pridanie dátového modelu. Tým, že model nie je inou cestou viazaný k systému a je nezávislý na systéme, umožňuje to jeho úpravu bez aktualizácie aplikácie. Pracuje sa iba s modelom, ktorý je používaný k dopytu, nemusia byť otvorené iné spojenia. Načítajú sa iba potrebné dáta. Najdlhšie trvá operácia pripojenia sa k zdroji dát a doba vyhodnotenia dopytu trvá rovnako dlho ako v systéme riadení báze dát, záleží aj na dopyte. Použitie výčtovej triedy nad definíciami tried dátových modelov umožňuje prístup k rozdielnym modelom bez opakovaného porovnávania a v rámci triedy aplikovať dopyty s konkrétnym spojením pre získanie dát.

[8]Reťazec špecifikujúci údaje o zdroji dát.

8.2 Parametre

Pred rozborom tvaru dopytov spracovateľných OGreportom, sa zameriam na vysvetlenie, čo sú to parametre pre dopyt, ktoré užívateľ zadáva. Ako by sa mali správne zadefinovať, spracovávať a ako ich zadať spolu s predvolenou hodnotou.

Pre nájdenie vhodného spôsobu ako spracovať parametre je nutné si položiť pár základných otázok. Ako vyriešiť požiadavku na úpravu, keď je parameter nepovinný, a ako má systém reagovať, keď sa parameter skladá z viacerých častí, ako je tomu napríklad u času od-do. Ďalej si stanoviť, či sa jeden parameter môže vyskytnúť viackrát v dopyte alebo nie. Ako vôbec program rozpozná, že sa jedná o parameter. Ako presne kontrolovať parametre, aby kontrola bola čo najpresnejšia a zároveň dostatočne rýchla.

8.2.1 Značenie parametrov

Pre rozpoznanie označenia parametru v texte dopytu je použitý znak '@'. Za možný parameter sa považuje každé slovo v textovom reťazci začínajúce zavináčom. Takéto zadefinovanie je bežná prax, ktorá sa využíva naprieč rôznymi systémami.

```
1 const parameterRegex = /(@\w+)/g;
```

Zdrojový kód 2: Zadefinovanie regexu na rozpoznanie parametrov

Rozpoznávanie značiek parametrov je veľmi efektívne vo vyhodnocovaní a rýchlom overovaní na vstupy. Má to však zabraňovať užívateľovi použiť v komentári alebo v pomenovaní rezervovaný znak @? Aby táto definícia nebola obmedzujúca pre užívateľa, je pridané začiarkavacie políčko, kde si užívateľ môže odkliknúť či sa jedná alebo nejedná o parameter. Túto možnosť je vidieť v ukážke zdrojového kódu 12, kde @TrainIdStationFrom a @TrainIdStationTo nie sú parametre pre aplikáciu, ale iba pomocné premenné pri dotaze.

8.2.2 Implementácia

Najdôležitejšie je si stanoviť ako budú parametre použité. Na základe otázok v úvode a zhromaždenia príkladov, kde je možné parametre použiť, ponúkli sa nasledujúce možnosti ako spracovať parametre.

1. Najjednoduchšie by bolo, čo jedna hodnota, to jeden parameter. Toto riešenie je ľahké na programovanie, ale zároveň nevyhovujúce. Komplikovalo by to zadefinovanie zložitejších parametrov, akým je napríklad parameter pre čas od-do znázornený na príklade tretieho kódu. Konkrétne tento parameter sa dá použiť zadefinovaním dvoch parametrov. Ale ako by sa pri tomto riešili nepovinné parametre? Pravdaže, dá sa s tým náročne vysporiadať preložením celého dopytu a zistením, ktorú časť podmienky textu treba vynechať.

2. Vhodnejšie by bolo nájsť riešenie, ktoré umožní zadefinovať ľubovoľnú štruktúru parametru. Druhá možnosť zahŕňa náhradu parametra za celé podmienky. Ako je uvedené v ukážke kódu štyri. Namiesto @@1 zostava vloží celú podmienku, napríklad "BETWEEN d '2020-12-20' AND d '2020-12-21'". Pri tomto riešení je otázka ako by sa vstup overoval a aká by musela byť znalosť bežného používateľa, aby s niečím takým mohol pracovať. Odpoveď na prvú otázku je, že transformáciu by uskutočňoval program, nie užívateľ. Pri konkrétne zvolenom príklade, by to znamenalo, že užívateľ zadá dva dátumy, program overí či sa jedná o dátumy, a podmienku vloženú do programu sformuluje sám. Znalosť užívateľa je teda kladená na úroveň byť schopný zadať dva dátumy. Toto riešenie uľahčuje zadefinovanie nepovinných parametrov. Týmto spôsobom sa môže do parametra podsunúť ľubovoľná štruktúra.

```
1 SELECT TOP(30)
2 VlakAkce.casPozadovany,
3 VlakAkce.casZalozeni
4 FROM VlakAkce
5 JOIN Vlak ON VlakAkce.idVlak = Vlak.idVlak
6 WHERE Vlak.realizaceDatum IS NOT NULL
7 AND Vlak.denJizdy >= @@01
8 AND Vlak.denJizdy < @@02
```

Zdrojový kód 3: Ukážka prvej možnosti riešenia parametrov

```
1 SELECT TOP(30)
2 VlakAkce.casPozadovany,
3 VlakAkce.casZalozeni
4 FROM VlakAkce
5 JOIN Vlak ON VlakAkce.idVlak = Vlak.idVlak
6 WHERE Vlak.realizaceDatum IS NOT NULL
7 AND Vlak.denJizdy @@01
```

Zdrojový kód 4: Ukážka druhej možnosti riešenia parametrov

Na základe zistených informácií a porovnaní výhod a nevýhod pri jednotlivých riešeniach, som sa rozhodla implementovať druhé riešenie. Hlavným rozhodovacím faktorom bola univerzálnosť na štruktúre, rýchlosť na overovanie a jednoduchšie na správu, a prístup k nepovinným parametrom.

To by bol spôsob použitia parametru v dopyte. Ostáva vyriešiť ako implementovať typy parametrov, aby systém rozlišoval jednotlivé typy na vstupe a podľa toho ich pripravil a vložil do textu dopytu. Pravdepodobne každý typ parametru bude vyžadovať inú validáciu, iný pohľad a aj iný zobrazený model pre vyplnenie informácií.

Keďže sa jedná o zoznam vopred daných možností parametrov, intuitívne to vedie k použitiu výčtových typov. Toto riešenie by bolo obmedzujúce kvôli upresneniu typov iba číselnými hodnotami. Ďalej validácia, prekladanie, rozšírenie a ďalšie spracovanie by bolo značne komplikované. Z toho dôvodu *ParameterTypes.cs* rovnako ako *ContextType.cs* využíva dedenie z *TypeUtility.cs*.

Toto riešenie sa dá ďalej rozvíjať, podľa toho ako bude konkrétny typ zadefinovaný. Či všetky informácie o ňom budú uvedené v danej výčtovej triede alebo nad typom parametru bude ešte obal v podobe triedy s funkciami pre prácu s daným typom. Pre porovnanie oboch možností slúži ukážka kódu číslo päť.

```
1  // the first possible implementation
2  public static readonly ParameterType DATE =
3  new ParameterType(
4    "DATE",
5    Translation.Parameter_Type_Date,
6    ParameterValidators.ValidDate
7    new DateParameterTypeViewModel(),
8    "Forms/ParameterTypes/_DateParameterType"
9    // ...
10   );
11
12 // the second option
13 public static readonly ParameterTypes DATE = new ParameterTypes(
       Translation.Parameter_Type_Date, new DateParameterType());
```

Zdrojový kód 5: Možné implementácie parametru áno/nie

Je zrejmá elegancia druhého riešenia a prehľadnejšia orientácia v projekte. Toto riešenie umožňuje univerzálny systém pre správu parametrov, ľahkú udržateľnosť a pridanie nového typu, bez kaskádovej úpravy kódu. Ďalej možnosť nastaviť tvar každého typu parametru pre zobrazenie a uviesť bližšiu špecifikáciu.

Trieda *ParameterType.cs* je rodičom všetkých parametrov v aplikácii. Táto trieda implementuje rozhranie *IParameterType.cs*, ktoré obsahuje hlavičky metód používané pre typy parametrov. K základným metódam, ktoré by mala potom implementovať zdedená trieda podľa svojho typu, je validácia, zisťovanie či údaje v modeli sú vyplnené, spracovanie modelu parametru, spôsob nahradzovania parametru v dotaze, keď je zadaná hodnota a naopak, keď má byť parameter ignorovaný. Dôležité je aj uviesť pohľad na zobrazovanie a jeho model.

Treba ešte brať v úvahu prípad, keď na zadefinovanie typu parametru sú potrebné iné údaje ako pri jeho použití. Napríklad pri obecnom výbere nad modelom dát pri definovaní potrebujeme uviesť identifikátor tabuľky a dva stĺpce nad modelom. Užívateľ pri generovaní reportu vidí iba rozbalovací zoznam s hodnotami. Preto sú uvedené aj metódy s príponou -default, ktoré v rodičovi *ParameterType.cs* predvolene vyvolajú základné metódy, napríklad metóda IsValid-Default volá IsValid. Je na odvodenej triede, či prepíše zdedené členy alebo určí iné chovanie typu parametru pri jeho vytváraní a jeho použití.

```
1  interface IParameterType
2 {
3  // Base value
4  bool IsValid(string model);
5  bool IsEmpty(string model);
6  ParameterTypeViewModel GetModel();
7  string ProcessParameter(string text,string name,string model);
8  string DisappearParameter(string text, string name);
9  // Default value
10 bool IsEmptyDefaultValue(string model);
11 bool IsValidDefault(string model);
12 ParameterTypeViewModel GetDefaultModel(string model);
13 string GetDefaultView();
14 string DefaultParameter(string text,string name,string model);
15 string DisplayDefault(string model);
16 }
```

Zdrojový kód 6: Implementácia rozhrania pre typ parametru

8.2.3 Nepovinné parametre

Nepovinný parameter užívateľ nemusí vyplniť, v tom prípade bude v texte dopytu ignorovaný. Ignorácia parametru je používaná na popis stavu, kedy sa podmienka, v ktorej je uvedený vždy vyhodnotí na pravdu. Preto je dôležité pre správne vyhodnotenie celého dopytu, aby správca dopytov správne zadal jeho text. Odporúča sa používať prednostne spojku AND pri nepovinných parametroch. Pretože ak niekto uvedie podmienky v dopyte a medzi nimi spojku OR bez zátvoriek, a nepovinný parameter sa ignoruje, budú sa ignorovať aj filtre za ním. Pretože podmienka s nepovinným parametrom bude pravdivý výraz a z logiky vieme, že spojka OR sa vyhodnocuje kým nenarazí na pravdivý výraz.

8.2.4 Vytváranie parametrov

Formulár na nový parameter sa automaticky načíta v tabuľke pod dopytom, ak užívateľ do textu dopytu vloží znak @ spolu s textovým reťazcom. Pri parametri má možnosť zaškrtnúť, že sa nejedná o parameter v zmysle tejto kapitoly. Automatické načítanie formulárov s názvom parametru je výhodné z praktického dôvodu. Užívateľa si vďaka tomu nemusí pri dlhých dopytoch strážiť názvy parametrov.

Názov parametru sa zmení podľa textu za @. Pri zaznamenaní zmeny v texte dopytu sa načíta zoznam parametrov vo formulári a pomocou regexu sa zistia parametre v texte dopytu. Zo zoznamu nových parametrov sa odstránia duplikáty. Potom sa skontroluje, či sa počet parametrov zmenil. Ak sa zmenil, buď je formulár pre parameter zmazaný alebo sa pridá nový. Ak sa počet parametrov nezmenil, porovnajú sa zoznami parametrov, a nájde sa prvý zmenený parameter. Na jeho základe sa aktualizuje názov parametru. Tento proces sa opakuje, dokiaľ zoznamy parametrov v texte dopytu a vo formulári nie sú rovnaké.

```
1  -- zle zadaný nepovinný parameter
2  SELECT *
3  FROM [Order] AS o
4  WHERE o.Time @time OR o.TableNo = 1
5
6  SELECT *
7  FROM [Order] AS o
8  WHERE o.Time < CURRENT_TIMESTAMP OR 1=1 OR o.TableNo = 1
9
10 -- správne zadaný nepovinný parameter
11 SELECT *
12 FROM [Order] AS o
13 WHERE (o.Time @time) AND o.TableNo = 1
14
15 SELECT *
16 FROM [Order] AS o
17 WHERE (o.Time < CURRENT_TIMESTAMP OR 1=1) AND o.TableNo = 1
```

Zdrojový kód 7: Ukážka nepovinných parametrov

8.2.5 Kontrola parametrov

Pre rýchlu odozvu na chyby v stupných poliach je validácia najprv uskutočnená na strane klienta. Kontroluje sa, či si užívateľ zvolil typ parametra. Každý typ parametra má vlastný model v pohľade. Umožňuje to mať pre každý parameter špecifikovanú validáciu cez anotácie pre overenie na strane užívateľa. Pokiaľ by niekto manipuloval so vstupnými dátami sa validátory nachádzajú aj na strane serveru z dôvodu celkovej bezpečnosti.

8.2.6 Typy parametrov

Nasleduje zhrnutie podporovaných typov parametrov OGreportu. Je k ním uvedené ich využitie a u niektorým ich spracovanie pri vyhodnocovaní dopytu.

Celé kladné číslo
: Základný typ parametra, reprezentujúci prirodzené číslo. Jeho uplatnenie je napríklad v nastavení počtu vybratých položiek. Tento parameter má nastavenú hodnotu ignorovania na 9999. Je to z dôvodu využívania parametra typu podmienka pri výberoch v klauzulách. Tento parameter je hlavne na korigovanie počtu dát na výstupe. Preto je nezmysel, aby toto číslo bolo väčšie.

Obdobie (od - do) pre dátum
: Parameter pre časový úsek od-do, vo výhľade ponúka datepicker a formátovanie na správny tvar dátumu. Pri tomto type sú buď vyplnené obe položky, alebo ani jedna. Užívateľ vidí dátum vo formáte podporovaným systémom. Pri dosadzovaní sa dátum premení na tvar {d 'yyyy-MM-dd'}.

Obdobie (od - do) pre dátum a čas
: Parameter pre dátum a čas funguje ako predchádzajúci typ iba s časom, môže byť použitý na rovnakých miestach. Využité formátovanie je {ts 'yyyy-MM-dd HH:mm:ss'}.

Textový parameter pre podmienky
: Pre podmienky s textovým reťazcom obsahuje operátor podmienok pre =, <>, <=, >=, <, >, obsahuje a neobsahuje. Obsahuje sa formátuje pomocou reťazca LIKE '%{0}%'.

Celočíselný parameter pre podmienky
: Je možné s ním zadefinovať číselné podmienky, kde operátor je zo sady =, <>, <=, >=, < a > a číslo je celé (. . . ,−3, −2, −1, 0, 1, 2, 3, . . .).

Logický parameter True/False
: SQL Server nemá boolovský dátový typ. Najbližšia aproximácia je bit, ale to je číselný typ, nie boolovský. Preto aj nahrádzanie za logický parameter je v skutočnosti nahradenie za hodnoty, ktoré sú buď pravdivé ($2 > 1$) alebo nepravdivé ($1 > 2$). Predvolená hodnota je ako u booleanu nepravda.

Výber obecný pro multiselect
: Užívateľ zadáva hodnoty, ktoré sa dosadia do zoznamu s uzavretými zátvorkami. Napríklad vstupná hodnota 'hrnček šálka' sa spracuje na '('hrnček', 'šálka')'.

Výber obecný pre číselník multiselect nad modelom
: Typ reprezentujúci zdrojový číselník z databáze. Má dva pohľady, jeden pri vytváraní parametra a druhý pri jeho použití. Pri vytváraní je dôležité zadať tabuľku, nad ktorou sa vytvorí číselník a identifikátory stĺpcov pre zobrazený text a hodnotu, za ktorú sa pri spracovaní zobrazený text nahrádza. Tieto stĺpce môžu byť rovnaké napríklad Name a Name, alebo rozdielne ako Id a Name. Pri generovaní si užívateľ môže vybrať viac hodnôt z číselníka. Uplatnenie má ako predchádzajúci typ, kde si užívateľ nemusí pamätať možnosti aké môže zadať, pretože mu ich aplikácia ponúkne.

8.2.7 Výhody a nevýhody

Po všetkých detailoch o parametroch je čas na stručnú rekapituláciu výhod a nevýhod zvoleného riešenia. Riešenie poskytuje voľnosť v tvare parametra a umožňuje zadefinovanie nepovinných parametrov. Toto riešenie sa ukázalo byť vhodné a univerzálne, dokonca aj pri vývoji, keď bolo potrebné doplniť ďalšie neplánované parametre, tak zavedenej štruktúre správy parametrov to nerobilo problém. Nevýhoda na tomto riešení je, že správca dopytov si musí byť vedomý ako správne parametre použiť, hlavne čo sa týka optimálnych parametrov a výčtov. Ďalej, s narastajúcim počtom parametrov narastá počet tried v projekte. To je cena za abstrakciu a s tým spojenou aj nezávislosť vrstiev, ľahko udržateľnú aplikáciu, jednoduchú úpravu a bezproblémové možnosti rozšírenie.

```
/*
Parametre
@count_1, @count_2 - celé čísla
@ingredients - číselník multiselect nad modelom
@time - obdobie od-do
@meal_name - textová podmienka
*/

-- pred nahradením parametrov
WITH searched_ingredient AS (
  SELECT TOP(@count_1) * FROM [Ingredient]
  WHERE [Ingredient].Name IN @ingredients
)
SELECT TOP(@count_2)
[Meal].ID,
[Meal].Name,
[searched_ingredient].Name AS 'Ingredient',
[MealIngredient].Quantity
FROM [Meal]
LEFT JOIN [MealIngredient] ON [Meal].ID = [MealIngredient].MealID
INNER JOIN [searched_ingredient] ON [searched_ingredient].ID = [
    MealIngredient].IngredientID
INNER JOIN [OrderMeal] ON [OrderMeal].MealID = [Meal].ID
INNER JOIN [Order] ON [OrderMeal].OrderID = [Order].ID
WHERE [Order].Time @time AND [Meal].Name @meal_name
ORDER BY [Meal].Name ASC

-- po nahradení parametrov
WITH searched_ingredient AS (
  SELECT TOP(99) * FROM [Ingredient]
  WHERE [Ingredient].Name IN ('Pepper', 'Salt')
)
SELECT TOP(4)
[Meal].ID,
[Meal].Name,
[searched_ingredient].Name AS 'Ingredient',
[MealIngredient].Quantity
FROM [Meal]
LEFT JOIN [MealIngredient] ON [Meal].ID = [MealIngredient].MealID
INNER JOIN [searched_ingredient] ON [searched_ingredient].ID = [
    MealIngredient].IngredientID
INNER JOIN [OrderMeal] ON [OrderMeal].MealID = [Meal].ID
INNER JOIN [Order] ON [OrderMeal].OrderID = [Order].ID
WHERE [Order].Time BETWEEN {d '2021-01-01'} AND {d '2021-02-01'} AND
[Meal].Name LIKE '%sa%'
ORDER BY [Meal].Name ASC
```

Zdrojový kód 8: Ukážka nahradzovania parametrov

8.3 Spracovanie dopytov

Keď sa v tejto práci vyjadrujem o dopyte (query), myslím tým objekt s názvom, popisom, autorizáciou, parametrami a textom obsahujúcim žiadosť na databázu na získanie vybraných dát. Administrátor dopytov, ktorý definuje zostavu (vytvára novú), je znalý dotazovacieho jazyka SQL a ako dané dopyty vytvárať.

K zvoleniu modelu a nastaveniu parametrov sa pridáva vloženie textu dopytu. Otázkou je, či systém bude podporovať všetky texty dopytu, ktoré užívateľ zadá alebo to bude nejakým spôsobom obmedzovať. Ďalej, či poskytne pri chybe nejakú odozvu, prípadne či systém ukáže výsledok dopytu. Všetky pripomienky sú spracované v nasledujúcom texte.

8.3.1 Tvar dopytu

Tvar dopytu je v tvare SQL príkazu pre výber sady výsledkov z databázy spojeným so syntaktickými značkami pre parametre. Kľúčový je príkaz SELECT a pomocné príkazy, napr. DECLARE. Nie sú povolené príkazy na zmenu štruktúry modelu alebo inej úpravy dát. Ďalej, rezervované slová SQL servera musia byť v hranatých zátvorkách, inak neprejdú v interaktívnom režime.

8.3.2 Kontrola dopytu

Je vhodné, aby pred uložením do databázy bol dopyt overený. Prípadne, aby bol administrátor dopytov informovaný o chybne zadanom dopyte. Inak by chybný (škodlivý) dopyt bol zistený až pri generovaní reportu. Práve kvôli chybám a odladeniu dopytu vývojári väčšinou používajú systém riadenia na báze dát, kde zadajú dopyt a ten následne skopírujú do generátora reportov.

Zámer bol, aby aplikácia sama zabezpečovala kontrolu dopytu a nepovolovala ani uložiť, ani vyhodnotiť dopyt, ktorý by mohol viesť ku zmene dát alebo štruktúry modelu a aby poskytla spätnú väzbu kde nastala chyba.

Najprv je potrebné si stanoviť, aké dopyty sú prijateľné programom. Dáta sú v bežných MS SQL tabuľkách v databázy, do ktorej má administrátor dopytov a administrátor zostáv prístup iba na čítanie povolených údajov. Potrebujeme dopyty na výber dát z databázy príkazom SELECT. Plus povolený môže byť aj SQL komentár (jednoriadkový – *komentár* a viacriadkový */* komentár */*), pre orientáciu v zložitých query.

Kľúčový prvok u každého systému je rýchlosť, preto zvolená kontrola musí prebehnúť čo najrýchlejšie a zároveň presne. Dopyt sa kontroluje vždy pred uložením do databázy a pri zvolení si možnosti ukázania výsledku dopytu. Tvar dopytu sa začne kontrolovať potom ako parametre v ňom boli spracované, nahradené za podobu vyhodnocovacej hodnoty. Kontrola prejde nasledujúcimi fázami.

1. Odstránenie komentárov zo vstupného textu.

2. Kontrola na úpravu štruktúry modelu, hľadajú sa kľúčové slová akými sú napríklad CREATE TABLE, DROP TABLE, . . . , zistí sa, či sú použité ako

príkazy alebo v pomenovaniach. Ak vstup obsahuje nepovolené príkazy na zmenu štruktúry, vráti sa chybová hláška, že užívateľ má právo iba na vyťahovanie dát.

3. Následne sa skontroluje cez transakciu korektnosť tvaru dopytu na syntaktické a sémantické chyby. Ak sa objaví problém vráti sa hlásenie s popisom chyby. Napríklad ak namiesto SELECT užívateľ uvedie SLECT.

4. Posledná kontrola je, či by sa uvedeným príkazom by sa nejakým spôsobom zmenili dáta v modeli. Ak áno, vráti sa chybová hláška rovnaká ako v prvom bode.

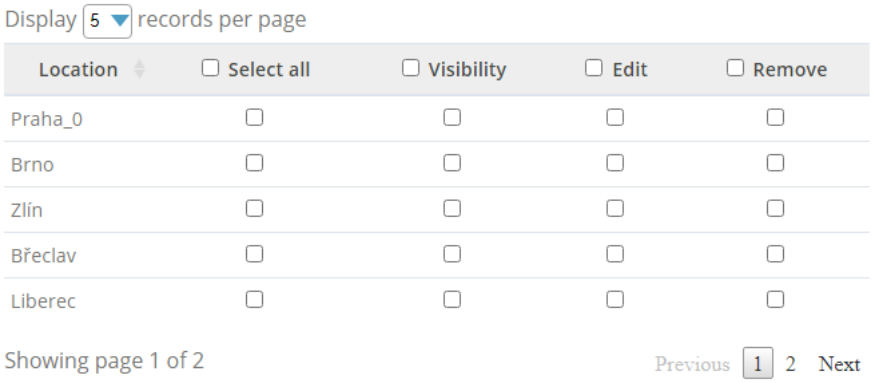
5. Ak dopyt prejde všetkými podmienkami, vráti sa TRUE a aplikácia prejde k vyhodnocovaniu dopytu. Ak by vstupný dopyt jeden z predošlých bodov nespĺňal, systém nesmie povoliť užívateľovi pokračovať vo vyhodnocovaní dopytu.

Príklady jednoduchých dopytov, aké by nemali byť akceptovateľné spolu s príslušným varovaním, sú uvedené na konci kapitoly. Zaujímavý (zo zdrojového kódu desať) je posledný prípad. Aplikácia priamo nezakazuje vloženie viacerých príkazov typu SELECT do textu jedného dopytu. V každom prípade vyhodnotí a vráti výsledok prvého z nich, v prípade, že sú tieto príkazy oddelené osobitne. Výnimku tvorí, ak je jeden príkaz súčasťou druhého a dávajú spoločný skombinovaný výsledok.

Ešte by som chcela upozorniť, že časovač vyhodnocovania povoľuje dopyty, ktoré sa uskutočnia do 5 minút. 5 minút je naozaj dlhá doba pre rozumné dopyty, ktoré sa používajú do reportov. Dokonca, kapacita jedného dátového listu v Exceli je 1 048 576 riadkov, preto sa predpokladá, že dopyty budú upravené podľa tohto obmedzenia. Je tam časová rezerva, pre prípad ak by overovanie trvalo dlhšie alebo systém bol extra zaneprázdnený. Obecne vyhodnocovanie trvá vždy rádovo niekoľko milisekúnd pre dobre definované dopyty. Pokiaľ sa teda systém dlho načítava a na konci poskytne prázdny výsledok, alebo chybovú hlášku typu čas vypršal, je na úlohe administrátora dopytov, aby si štruktúru dopytu upravil. Taký dopyt je napríklad, v ukážke kódu deväť. Jeho vyhodnocovanie aj v MSSQL trvá dlhšie ako 20 minút, a jeho výsledok v databáze zákazníka tvorí viac ako 80 411 788 položiek.

```
1 SELECT
2 [Vlak].Cislo AS 'Vlak',
3 [StanicaCielova].Nazev AS 'Cielova stanica'
4 FROM [Stanice] AS StanicaCielova
5 JOIN [Vlak] ON [Vlak].IdStaniceCilova = StanicaCielova.Id
```

Zdrojový kód 9: Dopyt nepoužiteľný pre reportovanie



Obr. 13: Tabuľka nastavovania oprávnení pre report alebo dopyt

8.3.3 Autorizácia dopytu

Pri vytváraní alebo úprave dopytu sa ponúkne tabuľka nastavovaní oprávnení pre všetky lokality. Z dôvodu aby dáta ostali konzistentné a v databáze neležali skryté objekty (nikto na ne nemá oprávnenia), platia pri nastavovaní oprávnení určité podmienky. Rovnaké platia aj pri nastavovaní oprávnení pri reporte.

- Pokiaľ je možné dopyt odstrániť alebo upravovať, je pre danú lokalitu viditeľný.
- To, že je dopyt viditeľný neznamená, že s ním môže užívateľ manipulovať a meniť jeho nastavenia.
- Pri nastavovaní oprávnení je v tabuľke vynechaná možnosť nastavenia pre aktuálnu lokalitu užívateľa. Je to z dôvodu, aby zámerne neodobral práva na editovanie všetkým lokalitám a z objektu sa nestal neviditeľný objekt v databáze z užívateľského pohľadu. Čiže vždy má aspoň jedna lokalita prístup na editovanie k dopytu.

8.3.4 Výhody a nevýhody

Text dopytu nie je žiadnym spôsobom obmedzovaný na použitie syntaxe inak než v klasickom dotazovacom jazyku SQL. Navyše užívateľ môže používať aj vyhradenú syntax pre parametre s @, ak zaklikne, že sa nejedná o parameter. Kontrola prebehne rýchlo. Pri jej neúspešnom priebehu je užívateľovi poskytnutá odozva, kde v texte dopytu nastal problém. Všetky dotazy nad modelom sú v transakciách, žiadnym spôsobom sa nenaruší tvar dát v zdroji. Veľkou výhodou je poskytnutie nadhľadu na výstupné dáta dotazu, aké budú neskôr exportované v reporte s použitím vzniknutého dopytu. Nevýhodou je, že pri vyhodnocovaní dopytu je nutné sa dvakrát pripojiť pomocou pripojovacieho reťazca k zdroju, raz na overovanie dopytu a raz na získanie dát.

```
1  -- error type: Unauthorized query
2  -- error text: You would interfere with a given database. You are
       granted rights for obtaining information, not to change data.
3  ALTER TABLE [Adresa]
4  ADD Email varchar(255);
5
6  -- error type: Invalid syntax
7  -- error text: Incorrect syntax near the keyword 'from'.
8  SELECT top(5) FROM [ogvTreeECVZakladni];
9
10 -- error type: Invalid syntax
11 -- error text: Invalid column name 'IdStaniceCilo'.
12 -- vyberie prvých 20 vlakov, ktorých cielová stanica má stat
       nastavený na false
13 SELECT top(20) [Vlak].Id, [Vlak].Kod, [Vlak].Nazev, [Vlak].Cislo
14 FROM [Vlak]
15 WHERE [Vlak].IdStaniceCilo IN
16 (
17  SELECT [Stanice].Id
18  FROM [Stanice]
19  INNER JOIN [Stat]
20  ON [Stanice].IdStat = [Stat].Id
21  WHERE [Stat].Platny = 'False'
22 )
23
24 -- error type: Invalid syntax
25 -- error text: Invalid object name 'Route'.
26 -- vypis pre~useky zacinajucu a cielovu stanicu
27 SELECT TOP(999) Route.ID AS TrainID, StationFrom.Name, StationFrom.
       ID AS StationFromID, StationTo.Name, StationTo.ID AS StationToID
28 FROM Route
29 JOIN Station AS StationFrom ON StationFrom.ID = Route.StartStationID
30 JOIN Station AS StationTo ON StationTo.ID = Route.
       DestinationStationID
31 ORDER BY Route.ID ASC
32
33 -- error type: Unauthorized query
34 -- error text: You would interfere with a given database. You are
       granted rights for obtaining information, not to change data.
35 SELECT * FROM [Adresa]; DROP Table [Adresa]
```

Zdrojový kód 10: Príklady základných chýb a chybových hlášok

8.4 Zostava

Zostava reprezentuje skupinu dopytov s vlastným oprávnením, pomenovaním a poradím. Využíva sa na zoskupenie dopytov, ktoré majú byť spoločne vygenerované do jedného výsledného súboru.

8.4.1 Oprávnenie zostáv

Nastavovanie oprávnení je rovnaké ako u nastavovaní oprávnení u dopytov. Zaujímavý je rozdiel medzi nastavovaním oprávnení u správcu dopytov a správcu reportov. Pri vytvorení dopytu sa automaticky stane dopyt viditeľný pre správcov dopytov podľa vyplnených lokalít v tabuľke. Správcovia reportov v rovnakej lokalite ako autorizovaný správcovia dopytov môžu nad vzniknutým dopytom vytvárať report a pre report určiť osobitne oprávnenia na editáciu, mazanie a viditeľnosť. To znamená, že keď správcovia reportov v lokalite X majú prístup k reportu vytvorenom v lokalite Y, neznamená to ihneď, že správci dopytov môžu dopyty obsiahnutých v reporte v lokalite X upravovať.

8.4.2 Formáty

Primárny formát stanovený klientom je .xslx. Avšak po vyskúšaní si iných systémov so širokou ponukou výstupných formátov som sa rozhodla aplikáciu navrhnúť tak, aby v budúcnosti bolo jednoduché aplikáciu rozšíriť o ďalšie formáty. Kvôli časovým možnostiam sú implementované tri základné formáty pre export. Všetky sú podporované na mnohých zariadeniach a umožňujú svoj export do ďalďalších formátov.

`.xlsx`
: XLSX je predvolený formát pri vytváraní tabuľky pomocou moderných verzií programu Excel, je založený na formáte XML. Súbory vo formáte XLSX sú organizované do tabuliek. Tento formát ponúka možnosti zabezpečenia, vytváranie formulí, vkladanie grafov, a ďalšie. Výstupný dokument obsahuje dva listy, jeden s naformátovanými dátami a jeden výhradne s údajmi bez zarovnania, či inej úpravy.

  Editácia výsledného .xlsx súboru nie je k dispozícii pred stiahnutím. Užívateľ má možnosť si editovať stiahnutý súbor, kde sa mu v tabuľkovom editore poskytne viac možností ako by zastrešovala knižnica ASP.NET MVC. Navyše to nie je primárny účel aplikácie.

  Generovanie Excelu v porovnaní s konkurenčnými riešeniami prebehne veľmi rýchlo. Rádovo o niekoľko minút pri dopytoch, kde už je query viac ako 20. Výsledný .xlsx súbor sa aj s veľkým množstvo dopytom vytvorí za pár sekúnd, najdlhšie trvá získavanie dát.

`.pdf`
: Dokumenty typu .pdf sú nezávislé od softvéru a operačného systému, na ktorom boli vytvorené. Tento formát je vhodnejší na korešpondenciu než XLSX.

Taktiež pre tlač nie je potrebné ďalšie nastavovanie a dokument bude mať rovnaký vzhľad z ľubovoľnej tlačiarne ako originál v počítači. Generovanie pdf trvá najdlhšie.

.csv

Jednoduchý formát súborov určený na ukladanie tabuľkových dát. CSV neobsahuje žiadne špeciálne formátovanie, sú to iba hodnoty oddelené čiarkami. Vzhľadom k tomu, že sú dáta uložené v .csv ako obyčajný text, ľahko sa môžu importovať do tabuľky alebo inej úložnej databázy. Tento typ formátu sa využíva na exportovanie vysoko objemných údajov.

Zvolila som si ho ako doplnkový formát z dôvodu, ak by nastala potreba mať výstupné dáta vo vizuálne neprezentovateľnej forme (na prezentácie, grafy a diagramy), alebo ich ďalšieho využitia. V tomto prípade by bolo ich získavanie z predošlých formátov namáhavejšie ako z .csv.

8.5 Spracovanie akcií

V niektorých situáciách musíme zabezpečiť, aby boli dôležité akcie vykonané v systéme zaznamenané. Aplikácia OGreport sa nezameriava priamo na informácie typu: Kto sa kedy prihlásil? Z akého zariadenia sa užívateľ prihlásil? Pre aplikáciu je dôležité zaznamenávať činnosť o zmenách aké užívateľ vykonal v systéme. Tieto záznamy majú význam pri spätnom hľadaní chýb v reportoch (spätná kontrolovateľnosť). Výhody tohto riešenia sú, že sa neovplyvňuje skutočná tabuľka a neukladajú sa žiadne nadbytočné informácie.

Riešené je to spôsobom, že v servisnej vrstve v *Servise.cs*, od ktorej dedia všetky ostatné servisné triedy, je inštancia AuditRepository z *UnitOfWork.cs*. V metódach pre vytvorenie a úpravu sa pre zvolený generický repozitár zavolá metóda, a aj príslušná metóda z *AuditRepository.cs* na uloženie údajov o zmene, ktorá nastala.

8.6 Užívateľské rozhranie

Pre uľahčenie práce som zvažovala pre počiatočný štýl použitie knižnice Bootstrap[9]. Z nasledujúcich dôvodov som toto riešenie prehodnotila. Knižnica je pre túto aplikáciu zbytočne rozsiahla, aplikácia OGreport bude obsahovať iba pár stránok, na ktorých bude prevažne použité rovnaké formátovanie, preto by som väčšinu komponent z bootstrapu nevyužila. Pre prispôsobenie štýlu by bol kvôli korektnosti a nemenenia existujúcej knižnice nutný ďalší .css súbor a bohužiaľ bootstrapová knižnica má na mnohých miestach !important, obchádzanie k preferovanému štýlu by bolo zdĺhavé.

Výsledné rozhranie je výsledkom aj ďalších balíkov, ktoré majú vlastné štýly napr. *jqueri-ui.css*. Tieto štýly boli doplnené sadou nástrojov pre písmo a ikonami Font Awesome.

[9]Bezplatný CSS rámec zameraný na responzívny front-endový webový vývoj.

9 Užívateľská dokumentácia

Táto kapitola je venovaná dokumentácii používania aplikáciu a popisu, ako fungujú jednotlivé funkcie poskytnuté rôznym roliam. Užívateľov, okrem rolí delíme na prihlásených a neprihlásených. Každopádne využívať hlavné vlastnosti aplikácie môžu iba autorizovaný užívatelia.

9.1 Každý užívateľ

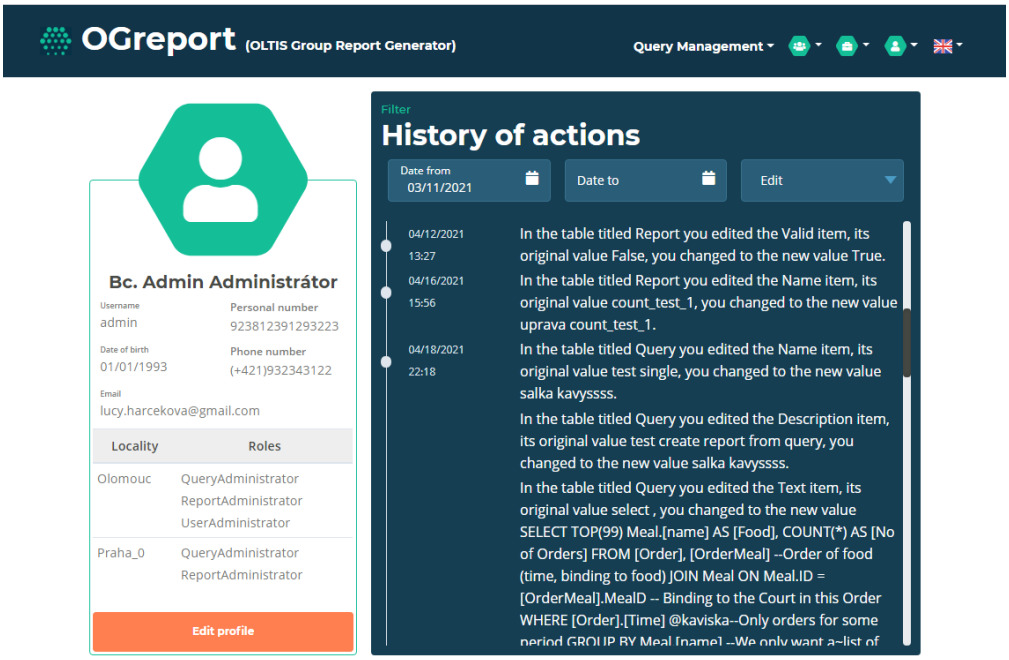
Každý užívateľ, nezávisle na roly, pod ktorou momentálne vystupuje v aplikácii, má prístup k základným funkciám, ktorými sú nasledujúce.

- Zmena jazyka – momentálne sú k dispozícii tri jazykové variácie aplikácie, medzi ktorými môže užívateľ prepínať za pomoci ikoniek v podobe vlajok krajín v pravom hornom rohu.
- Prihlásenie
- Odhlásenie
- Technická podpora
- Zabudnuté heslo / užívateľské meno – užívateľ vyplní emailovú adresu, ktorá je vedená v systéme pre jeho užívateľský účet. Na e-mail mu príde prihlasovacie meno a odkaz pre obnovu hesla.
- Zmena lokality – táto možnosť je označená ikonkou skupiny, zobrazí sa iba ak je užívateľ autorizovaný vo viacerých lokalitách.
- Zmena roly – v rámci pracoviska sa užívateľ môže prepnúť medzi jemu prístupnými rolami v danej lokalite. Užívateľ môže mať rôzne role na rozličných pracoviskách. V prípade ak má iba jednu, tak sa táto možnosť nezobrazí. Zmena roly je označená ikonkou kufríka v pravej časti horného menu.

9.1.1 Správa profilu

K správe profilu prejde užívateľ zakliknutím pod ikonkou postavy v hlavnom menu v rozbaľovacej ponuke možnosť “Profil”. Na stránke profilu sú zobrazené informácie o užívateľovi vedené v systéme. K dispozícii je aj zoznam akcií, ktoré užívateľ vykonal v rámci aplikácie, týkajúce sa pridávania a úpravy objektov. Zoznam akcií je možné filtrovať podľa dátumu od-do a typu činnosti (úprava alebo vytvorenie).

V tejto časti môže užívateľ editovať základné informácie o ňom, akými sú meno, priezvisko, email a podobne v súlade so základnými validáciami (tvar emailovej adresy, maximálny počet znakov pre telefónne číslo je pätnásť, atď.). Naopak upravovať jemu pridelené role, lokality, aktívny stav a identifikačné číslo nemôže. Toto oprávnenie má iba administrátor užívateľov.



Obr. 14: Pohľad na užívateľský profil

9.2 Administrátor zostáv

Má na starosti radenie dopytov do reportov, a ich následné generovanie. Nemusí byť špeciálne zaškolený na ovládanie jemu pridelených funkcií, pretože sa mu nezobrazujú zdrojové texty dopytov. Správca reportov sa riadi názvom a popisom dopytov pri určovaní aké dáta dopyt získava zo zdroja. V nasledujúcom texte je ako synonymum pre report používané slovo zostava a naopak.

9.2.1 Prehľad zostáv

Prehľad zostáv obsahuje zoznam zostáv, ktoré má administrátor reportov právo vidieť. Reporty sú usporiadané v tabuľke podľa vlastného poradia. V tabuľke je možné reporty zoradiť podľa jednotlivých stĺpcov a nad tabuľkou je formulár pre vyhľadávanie zostáv. V pravom stĺpci tabuľky môžu byť pri jednotlivých reportoch zobrazené nasledujúce ikonky:

- odpadkový kôš – ikonka pre odstránenie reportu,
- papier s perom – odkaz na editáciu reportu,
- oko – náhľad na detaily vybranej zostavy,
- dve šípky od seba – ikonka pre otvorenie modálneho okna pre zmenu poradia (viac v časti zmena poradia zostavy),
- ikonka pre sťahovanie – odkaz na prechod na pohľad generovania reportu,

- oranžový varovný trojuholník – každý report by mal obsahovať viac ako jeden dopyt. Môže sa stať, že report nebude obsahovať žiadne dopyty, pokiaľ boli vymazané. V tom prípade sa zobrazí výrazný trojuholník ako upozornenie na report bez dopytov.

9.2.2 Detaily zostavy

Pre otvorenie modálneho okna s detailami vybraného reportu je nutné v tabuľke so zostavami na riadku, kde sa daný report nachádza kliknúť na ikonu pre náhľad v tvare oka.

K zobrazeným informáciám v modálnom okne patrí základný popis reportu, akými sú názov, validita, popis, dátum vytvorenia. Ďalej v ľavom stĺpci je zobrazená tabuľka s oprávneniami reportu a na pravo sa nachádza stručný popis dopytov, ktoré report zahŕňa.

9.2.3 Vytvorenie zostavy

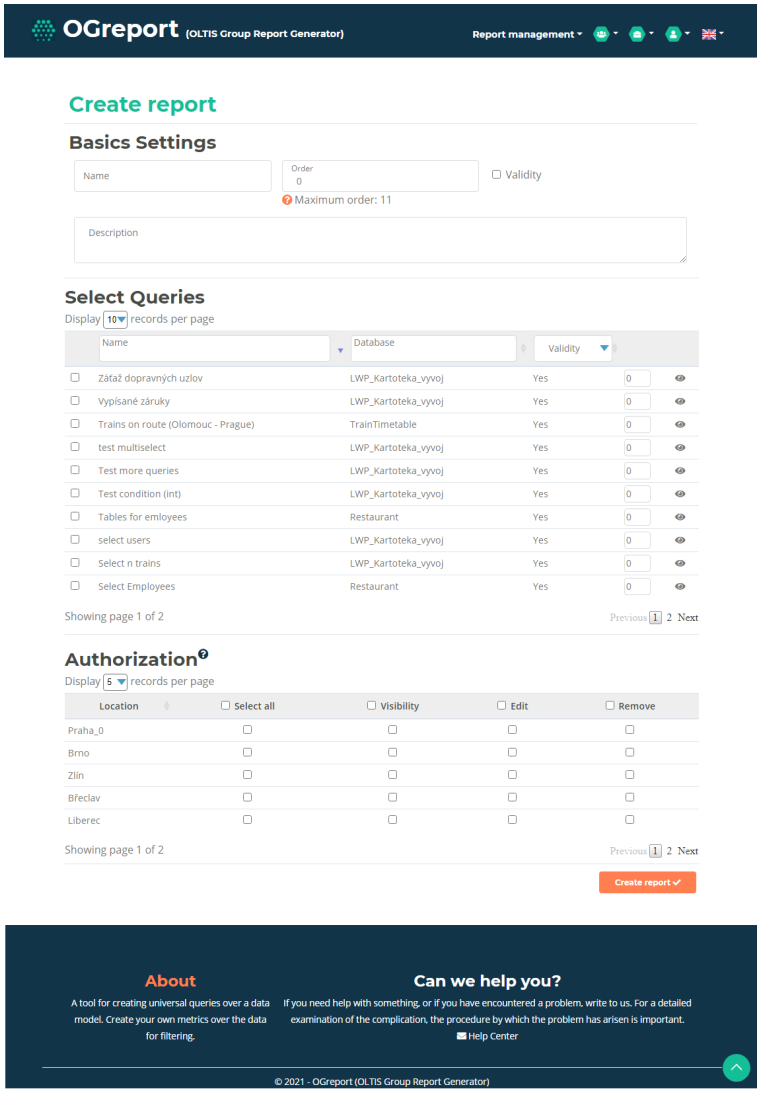
Na pohľad vytvorenia reportu sa užívateľ preklikne tlačidlom "Create report" umiestneným v hornej časti pohľadu "Prehľad zostáv" alebo kliknutím na možnosť "Create report" v rozbaľovacom menu pod správou reportov.

V hornej časti sa nachádza formulár pre základné nastavenia zostavy. Povinné sú polia pre názov a poradie. Zvolené poradie môže ovplyvniť poradia existujúcich zostáv. Ako sa tieto poradia menia je popísané v sekcii zmena poradia zostavy.

Výber dopytov z tabuľky je ďalším krokom pri tvorbe zostavy. Dopyt do zostavy si užívateľ zvolí zaškrtnutím štvorca v ľavom stĺpci, pri vybratí dopytu sa automaticky jeho počet výskytov v reporte nastaví na jedna. V tabuľke dopytov je možné vyhľadávať podľa názvu, validity a modelu. Takisto je možné dáta v tabuľke zoradiť podľa príslušných stĺpcov. Na každom riadku v tabuľke s dopytmi je ikonka pre zobrazenie detailov o dopyte. Povinné je si zvoliť aspoň jeden dopyt, ktorý bude výsledná zostava obsahovať. Výhodou OGreport je možnosť kombinovať viaceré dopyty a generovať tak komplexnejšie reporty. Aplikácia dokonca umožňuje užívateľovi zadať počet koľkokrát sa má jeden dopyt pri generovaní reportu aplikovať, pričom pokiaľ je počet jedného dopytu v reporte väčší ako päť, zobrazí sa pred uložením potvrdzovacie dialógové okno, kde užívateľ potvrdí svoju voľbu alebo ju zruší s tým, že sa jedná iba o preklep.

Nasleduje nastavenie oprávnení pre zostavu. Pre urýchlenie práce je v tabuľke oprávnení stĺpec pre výber všetkých položiek a na každom riadku je takisto táto možnosť. Aktuálna lokalita bude mať automaticky nastavenú viditeľnosť, možnosť úpravy a vymazania pre vzniknutý report.

Po vyplnení všetkých údajov, nastavení autorizácie a vybraní dopytov sa report vytvorí kliknutím na tlačidlo "Create report" v spodnej časti stránky. Pri chybne vyplnených údajoch sa zobrazia chybové hlášky, v opačnom prípade sa vytvorí report, načíta sa opäť pohľad pre tvorbu zostavy bez vyplnených údajov a zobrazí sa zelená notifikácia.



Obr. 15: Pohľad na vytvorenie zostavy

9.2.4 Generovanie zostavy

Pre generovanie konkrétnej zostavy musí užívateľ v tabuľke reportov kliknúť na ikonu stiahnutia. Generovať je možné iba validný report, to znamená, že on sám je validný a sú validné aj všetky jeho dopyty. Pokiaľ by sa užívateľ pokúsil dostať ku generovaniu reportu napríklad zmenou url, aplikácia mu to nepovolí. Užívateľovi sa zobrazí upozornenie, že zvolený report nie je valídny alebo neexistuje, ak report nie je pre lokalitu užívateľa viditeľný.

V pohľade generovania zostavy sú v hornej časti vypísané stručné informácie o reporte. Nasleduje formulár s viacerými krokmi, kde je možné sa vrátiť späť v ľubovoľnom kroku, pričom pôvodné hodnoty zostanú uložené. Posunúť sa na ďalší krok formulára sa užívateľ môže iba v prípade správne vyplneného aktuálneho kroku.

Postup pri vygenerovaní reportu je nasledovný:

1. Prvý krok formulára obsahuje základné nastavenia pre vygenerovanie reportu. Zatiaľ k nim patrí iba nastavenie formátu výsledného súboru. XLSX je predvolený formát, keďže je to primárny formát používaný na generovanie reportov aplikácie OGreport. Po zvolení si formátu užívateľ prejde na ďalšiu stranu tlačidlom "Next >".

2. V druhom kroku formulára užívateľ vyplní hodnoty, ktoré sa majú dosadiť za parametre v dopytoch. Hodnotu parametru nemusí vypĺňať, ak parameter nie je povinný. V opačnom prípade je si nutné zvoliť buď použitie predvolenej hodnoty alebo zadať hodnoty ručne do formuláru. Pri použití predvolenej hodnoty, ak táto hodnota neexistuje, čiže pri vytváraní dopytu ju správca dopytov nezadal, je na to užívateľ upozornený. Rovnako je upozornený aj na chybne zadané hodnoty vstupných poliach pri rôznych typoch parametrov.

3. Tlačidlom "Generate report" v spodnej časti strany druhého kroku sa vyvolá stiahnutie výsledného súboru vo zvolenom formáte.

Vo výslednom reporte formátu PDF alebo XLSX je v hornej časti umiestnený čas, kedy bol report vygenerovaný a krstné meno s priezviskom užívateľa, ktorý report vygeneroval.

9.2.5 Upravenie zostavy

K upraveniu zostavy sa užívateľ dostane kliknutím ikonky s papierom a perom v tabuľke so zostavami alebo z modálneho okna detailov zostavy kliknutím na tlačidlo "Edit". Upraviť zostavu môže iba v tom prípade, pokiaľ je k takejto funkcii oprávnený.

V modálnom okne pre upravenie zostavy je možná zmena všetkých nastavení u reportu od základných akými sú názov a popis, po práva reportu a dopyty v reporte. Všetky zmeny musia byť validné rovnako ako pri tvorbe reportu.

9.2.6 Zmena poradia zostavy

Poradie reportu je možné zmeniť troma spôsobmi. Prvý postup je priamo pri editácii v modálnom okne nastavením nového poradia zostavy. Druhým je potiahnutie reportu v tabuľke zostáv, kde svoju voľbu musí užívateľ potvrdiť v dialógovom okne. Posledný spôsob je kliknutie na ikonku pre zmenu poradia zostavy. Následne sa otvorí modálne okno pre nastavenie nového poradia.

Nastavenie nového poradia pre zvolený report môže ovplyvniť poradie ostatných reportov. Poradie vybraného reportu sa nastaví na zvolenú pozíciu, podľa toho sa upraví poradie ostatných správ. Napríklad, ak bol report na pozícii 5 a presunie sa na pozíciu 1. Zostavy 1, 2, 3, 4 posunú svoje poradie o jedna hore na 2, 3, 4, 5. Ak by sa report z pozície 5 posunul na 7. miesto, poradie zostáv 6, 7 za zníži o jedna.

9.2.7 Odstránenie zostavy

Zostavu môže užívateľ vymazať, pokiaľ je v lokalite s oprávnením na jej vymazanie. Kliknutím na ikonku odpadkového koša v tabuľke reportov sa pre report na riadku, kde bola ikona stlačená vyvolá akcia pre jeho zmazanie. Tú je potrebné dodatočne potvrdiť, kvôli bezpečnosti pri nechcenom prekliknutí.

9.3 Administrátor dopytov

Administrátor dopytov je užívateľ, ktorý je zaškolený a pozná dopytovací jazyk SQL. Má na starosti správu dopytov v aplikácii OGreport.

9.3.1 Prehľad dopytov

Po kliknutí v hlavnom menu na položku "Správa dopytov" sa zobrazí prehľad dopytov. V ľavej časti je tabuľka s dopytmi viditeľnými v aktuálnej lokalite užívateľa, v ktorej je možné dopyty vyhľadávať a zoraďovať podľa názvu a modelu.

Na pravo je miesto pre zobrazenie detailov pre vybraný dopyt. Pre zvolenie dopytu, ktorého podrobné informácie sa majú zobraziť, je nutné v tabuľke dopytov kliknúť na riadok, kde sa požadovaný dopyt nachádza.

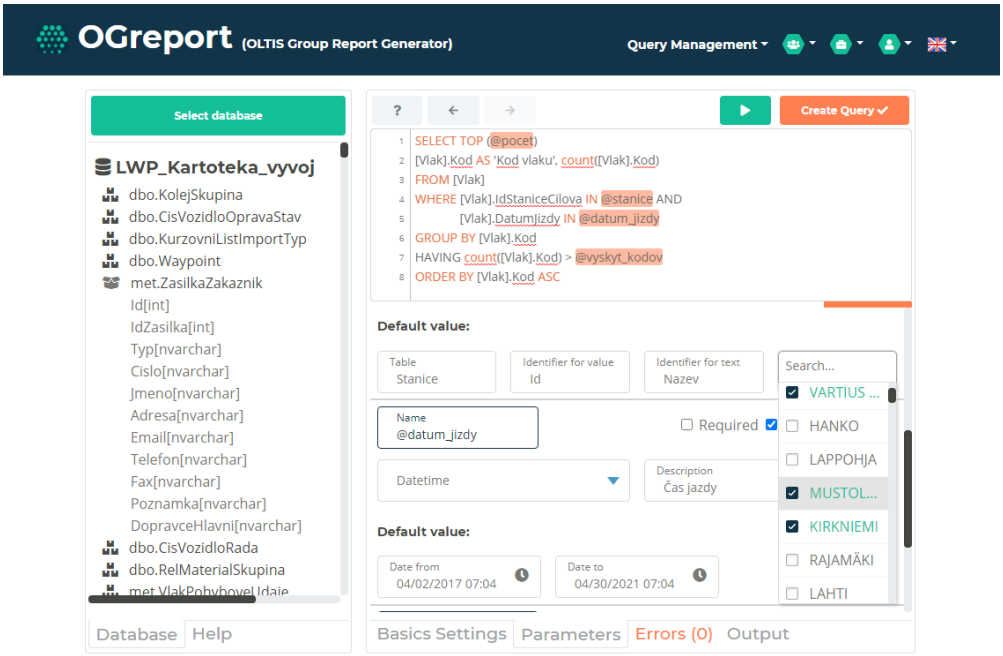
Pri dopytoch sú v tabuľke v pravom stĺpci možnosti pre mazanie, editáciu a kopírovanie označené špeciálnymi ikonami. Jednotlivé ikony sú zobrazené na riadkoch s dopytmi, pre ktoré sú v danej lokalite oprávnenia.

- dva papiere – ikonka pre kopírovanie dopytu. Po kliknutí na ňu sa otvorí pohľad pre vytvorenie dopytu s vyplnenými údajmi na základe skopírovaného dopytu.
- odpadkový kôš – ikonka pre mazanie dopytu, túto akciu je nutné po zvolení potvrdiť v dialógovom okne.
- papier s perom – odkaz na editáciu dopytu.

9.3.2 Vytvorenie dopytu

Počíta sa s tým, že administrátor zostáv je zaškolený užívateľ so znalosťami SQL a tvorby dopytov, každopádne sa aplikácia snaží byť čo najviac odolná voči nesprávnemu zaobchádzaniu. Okrem nepovolenia chybných vstupov má na ľavej strane vypísanú štruktúru dátového modelu, s ktorým sa pracuje, náhľad na správny tvar zadávaných parametrov a v spodnej časti stránky má celý postup ako správne vytvoriť dopyt.

Model, nad ktorým bude report vyhodnotený, si užívateľ vyberie kliknutím na tlačidlo "Select datatabase" v ľavej časti. Pokiaľ je k dispozícii iba jeden model, bude už automaticky načítaný po zobrazení stránky pre ušetrenie času. Štruktúra modelu na ľavej strane sa skladá z názvu modelu a zoznamu mien tabuliek. Po kliknutí na názov tabuľky sa pod jej menom zobrazia jej stĺpce s príslušnými dátovými typmi.



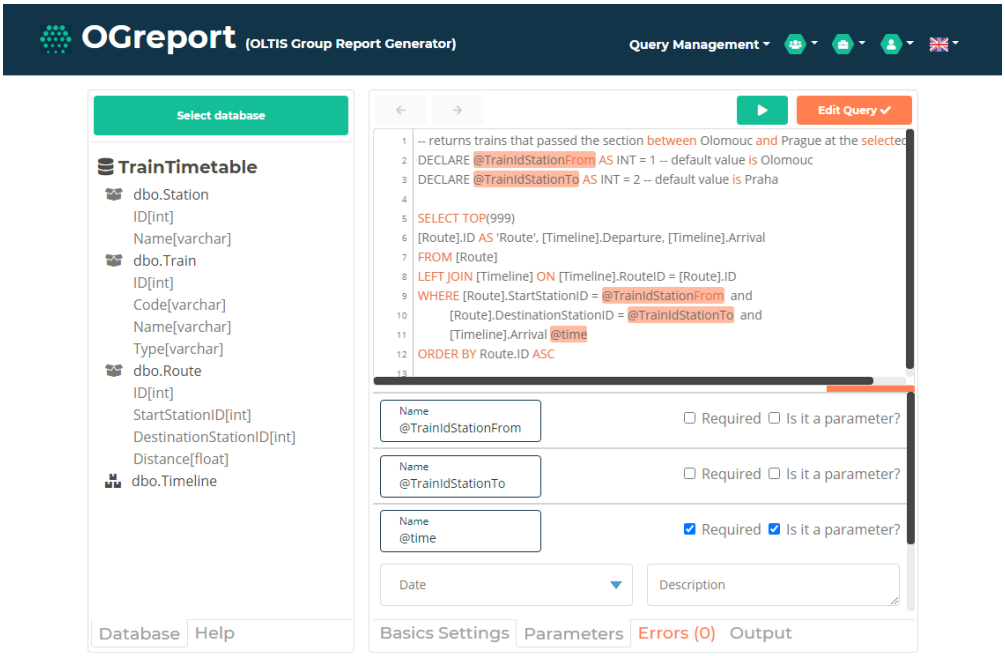
Obr. 16: Pohľad na vytvorenie dopytu

Pravá strana je rozdelená na dve časti, časť pre vkladanie textu dopytu a časť pre nastavovacie a informačné záložky. Medzi týmito úsekmi je horizontálna oranžová čiara. Pri jej ťahaní je možné zmeniť veľkosť rozdelenia týchto častí.

Nad časťou pre vloženie textu dopytu, sú šípky doprava a doľava pre riadenie zmien vykonaných v texte dopytu. Pri zadávaní textu dopytu sú zvýraznené slová v dopyte obsahujúce vyhradenú syntax pre parametre a farebne sú odlíšené časti kľúčových slov jazyka SQL.

Pokiaľ dopyt obsahuje špeciálny syntax pre parametre, tak sa v záložke "Parameters" načítajú formuláre pre parametre. Ak sa jedná iba o rovnaký syntax a nejedná sa o parameter, ktorý sa má v systéme ďalej spracovať, užívateľ nezaškrtne políčko "IsParameter". Pre zmenu názvu parametra je nutné upravovať jeho pomenovanie priamo v texte dopytu. Vo formulári pre parameter, je povinné si zvoliť aký typ má parameter. Podľa typu sa načítajú vstupné polia pre predvolenú hodnotu, ktorá však nie je povinná.

Užívateľ má k dispozícii funkciu pre zobrazenie dát, ktoré napísaný dopyt vytiahne zo zdroja. Táto možnosť, je označená zeleným tlačidlom v pravo nad textom dopytu. V texte dopytu sa pri testovacom vyhodnotení parametre ignorujú, pokiaľ nemajú vyplnenú predvolenú hodnotu. Ak ju majú určenú, tak sa do výrazu pri testovaní správnosti dosadí za parameter. Pred samotným vyhodnotením dotazu sa skontroluje model, dopyt a prípadné paramatre. Pokiaľ pri overení korektnosti jedného z nich nastala chyba, aplikácia v záložke "Errors" zobrazí chybové hlášky odpovedajúce problémom. V prípade, že kontrola prebehla úspešne, sa v záložke "Output" zobrazí výsledok dopytu v tabuľke, v ktorej je možné daný výsledok filtrovať alebo zoradiť podľa jednotlivých stĺpcov.



Obr. 17: Pohľad na editáciu dopytu

Špeciálnu možnosť vytvorenia reportu nad dopytom je možné si zvoliť zaškrtnutím políčka “Create Report” v záložke základných nastavení. V tom prípade bude nad vzniknutým dopytom vytvorená zostava s rovnakým názvom, popisom, validitou a autorizáciou, s maximálnym možným poradím, obsahujúca iba jeden dopyt.

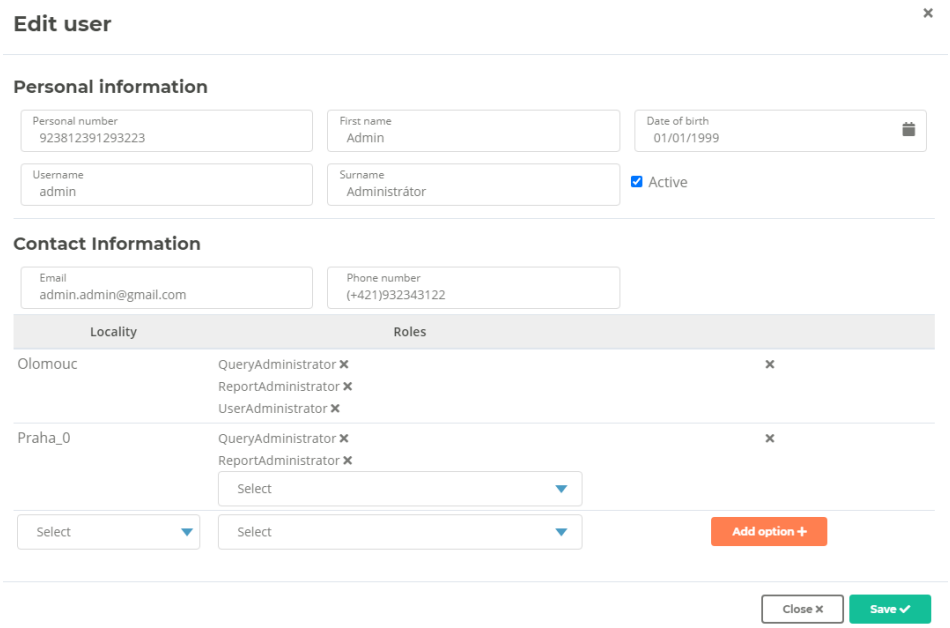
Po zvolení si modelu, vyplnení základných informácii, vpísaní textu dopytu a prípadnom nastavení parametrov sa dopyt vytvorí kliknutím na tlačidlo “Create Query”. Ak by bol čo i len v jednej z vyplnených častí problém pri validácii, dopyt sa nevytvorí a v záložke “Errors” sa upozorní na existujúce chyby.

9.3.3 Editácia dopytu

Užívateľ sa k editovaniu dopytu dostane kliknutím na ikonku editovania v tabuľke dopytov. Pohľad na editovanie je veľmi podobný pohľadu pre vytvorenie dopytu, ovládanie je identické ako pri vytváraní dopytu. Jediný rozdiel je, že užívateľ si nemôže zvoliť možnosť pre vytvorenie reportu nad upraveným dopytom. Všetky funkcie ostávajú rovnaké.

9.4 Administrátor užívateľov

Administrátor užívateľov je doplnková rola, má na starosti správu užívateľov. Ich vytvorenie, editáciu a k dispozícii má aj ich prehľad. Všetky spomenuté funkcie sa priamo nachádzajú alebo sa k ním užívateľ preklikne cez pohľad “User Management”, ktorého odkaz je v hlavnom menu.



Obr. 18: Pohľad na editáciu užívateľa

9.4.1 Prehľad užívateľov

Prehľad užívateľov obsahuje tabuľku užívateľov, v ktorej je možné zobrazené položky zoradiť podľa jednotlivých stĺpcov. Nad tabuľkou je umiestnený formulár pre vyhľadávanie na základe všetkých údajov uvedených o užívateľovi v systéme. Pri vpísaní textu do formulára sa automaticky aktualizujú vyfiltrovaní užívatelia v tabuľke. Pravý stĺpec tabuľky obsahuje ikonky pre:

- detail užívateľa – po kliknutí na ňu sa zobrazia základné informácie o zvolenom užívateľovi a jeho akcie, ktoré učinil,
- úpravu užívateľa – slúži na zobrazenie modálneho okna pre úpravu užívateľa.

9.4.2 Vytvorenie užívateľa

Pri kliknutí na tlačidlo “Create User +” sa zobrazí modálne okno s príslušným formulárom. V ňom je nutné vyplniť povinné položky akými sú email, prihlasovacie meno, krstné meno, priezvisko a identifikačné číslo. Zaujímavé je, že správca užívateľov nenastavuje heslo pre vytvoreného užívateľa. Heslo si nastaví vzniknutý užívateľ sám, po obdržaní emailu pre nastavenie hesla.

Záver

Výsledná aplikácia poskytuje nástroj pre správu reportov a dopytov, je plne funkčná a spĺňa zadanie diplomovej práce od spoločnosti OLTIS Group. Prekonáva existujúce riešenie predovšetkým tým, že umožňuje vyškolenému užívatelovi povereného zákazníkom vytvárať a spravovať reporty samostatne a dopĺňa ďalšie funkcie. Vďaka jej univerzálnosti môže byť distribuovaná a prispôsobená viacerým lokalitám.

Mojím cieľom bolo vytvoriť silný základ, na ktorom je možné ďalej stavať a dopĺňať prvky. Zvolená architektúra a návrh správy kontextov s parametrami umožňuje vhodnú validáciu nad rôznymi typmi parametrov, jednoduchú rozšíriteľnosť, pridanie formátov pre export a stanovenie oprávnení.

Vzniknutá webová aplikácia je nenáročná na ovládanie a orientáciu, čo je nevyhnutný predpokladom pre používanie zákazníkom, narozdiel od komerčne pojatých produktov. K základným vlastnostiam systému patrí definovanie SQL dopytu nad dátovým modelom so syntaktickými značkami pre parametre a vytvorenie reportu. Aplikácia navyše poskytuje možnosti generovania niekoľkých dopytov v jednom reporte, uloženie predvolených hodnôt pre parametre, nastavenie validity dopytov a reportov, použitie nepovinných parametrov v dopytoch, kopírovanie dopytov, atď.

Aplikácia sa bude v budúcnosti naďalej vyvíjať na základe spätnej väzby od zákazníka. Pre vylepšenie navrhujem implementáciu nástrojov pre úpravu vzhľadu exportov, prípadné doplnenie ďalších formátov reportov.

Conclusions

The resulting application provides a tool for managing reports and queries, is fully functional, and meets the assignment of a diploma thesis from the company OLTIS Group. In many ways, it outperforms the existing solution and complements other features. Thanks to its versatility, it can be distributed and adapted to several locations.

My goal was to create a strong foundation on which it is possible to further build and add elements. The chosen architecture and design of context management with parameters allows suitable validation over various types of parameters, easy extensibility, the addition of formats for export, and determination of authorizations.

The resulting web application is easy to use and navigate, which is a necessary prerequisite for customers usage, unlike usual commercially designed products. The basic features of the system include defining an SQL query over a data model with syntax tags for parameters and creating a report. In addition, the application provides the ability to generate multiple queries in a single report, save default values for parameters, set the validity of queries and reports, use optional parameters in queries, copy queries, etc.

In the future, the application will, of course, continue to evolve based on customer feedback. To improve, I propose the implementation of tools for modifying the appearance of exports, possibly adding other report formats.

A Spustenie

A.1 Potrebné technológie

`Visual Studio`
: Integrované vývojové prostredie (IDE) pre spustenie a testovanie projektu. Dostupné z www.visualstudio.microsoft.com/cs/downloads/. Vhodné verzie od VS 2013. Stačí verzia pre vývojárov dostupná zadarmo.

`MS SQL a SQL Server`
: Databázový systém pre správu a ukladanie dát. Zadarmo k stiahnutiu na www.microsoft.com/en-us/sql-server/sql-server-downloads, vhodná ľubovoľná verzia z dostupných.

`Windows`
: Operačný systém (OS), na ktorom bola aplikácia vyvíjaná. Je možné použiť aj iný OS. Napríklad, pre MacOS je VS k dispozícii a MS SQL za použitia Dockeru je možnosť sprevádzkovania, prípadne pre získanie vhodného OS použite virtuálny stroj slúžiaci na emuláciu počítačového systému.

A.2 Obnovenie databázy

1. V MS SQL sa pripojíte k lokálnemu SQL serveru.
2. Pravým tlačidlom kliknete na *Databases* > *Restore Database*.
3. V dialógovom okne vyplníte ako zdroj zariadenie a kliknete na tlačidlo s tromi bodkami na zadanie cesty.
4. Otvorí sa dialógové okno, v ňom si vyberiete možnosť Add a následne zadáte cestu k súboru z priloženého CD.
5. Automaticky sa vyplní názov databázy, ktorá sa má obnoviť.
6. Pre obnovu databázy následne stlačíte tlačidlo OK.

Tento proces je nutné vykonať pre zálohu *GeneratorOfReports.bak* v priečinku */data* na priloženom CD. Nájdete tam aj zálohy testovacích modelov *LWP_Kartoteka_vyvoj.bak*, *TrainTimetable.bak* a *Restaurant.bak*, kde podľa svojej voľby obnovte aj testovacie databázy.

A.3 Sprevádzkovanie aplikácie

V súbore `Web.config` je nutné nastaviť connection string pre pridanú databázu *GeneratorOfReports*. Spojenie k testovacím databázam nastavíte v *src* > *GeneratorOfReports* > *GeneratorOfReports* > *ContextManagement* > *ContextType.cs*, kde pôvodné pripojovacie reťazce nahradíte novými. Connection string nájdete v MS SQL, pri pravom kliknutí na názov databázy a otvorením Details.

B Testovanie

Po obnovení databáz podľa prílohy Obnovenie databázy a nastavení potrebných údajov je možné aplikáciu spustiť s ukážkovými dátami. Prihlasovacie údaje k testovaciemu účtu so všetkými prístupnými rolami v dvoch lokalitách sú **meno admin** a **heslo admin**. Na vytvorenie vlastného účtu prejdite z role administrátor užívateľov na možnosť vytvoriť užívateľa. Tam vyplníte povinné položky a zadáte príslušné lokality s rolami, nezabudnite prosím vyplniť správne emailovú adresu, na tú vám bude doručený odkaz s nastavením hesla do aplikácie.

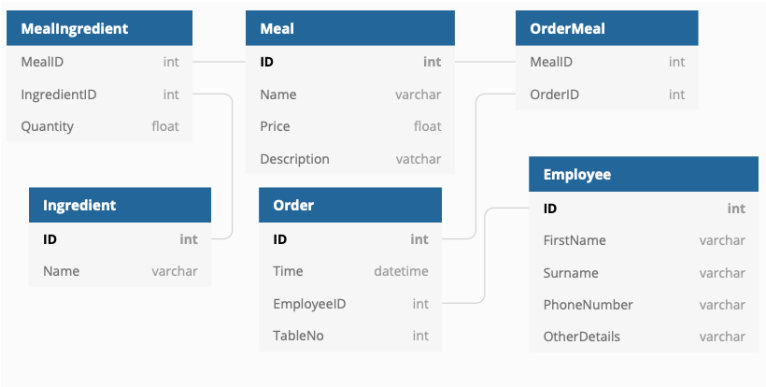
B.1 Testovacie databázy

Popis modelov testovacích databáz, ktorých záloha je na priloženom CD. Rada by som podotkla, že prvé dve databázy majú jednoduchú štruktúru, obsahujú iba niekoľko tabuliek a slúžia výhradne na testovanie, nie reálne použitie pre nejaký systém. Posledná databáza obsahuje viacej údajov a jej model je rozsiahlejší.

B.1.1 Restaurant

Schéma databázy pre reštauračný systém. Vďaka tomu, že schéma nie je zložitá, je rýchla na orientáciu a intuitívne pochopenie.

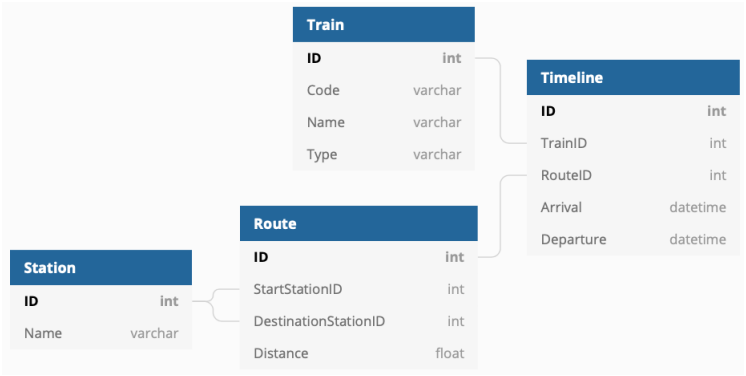
Vhodná je na dopyty typu: Počet objednávok za určitý čas pre stôl, ktorý obsluhovali vybraní zamestnanci. Zoznam jedál obsahujúci danú surovinu určitý počet krát. A ďalšie.



Obr. 19: Schéma testovacej databázy Restaurant

B.1.2 TrainTimetable

Testovacia databáza TrainTimetable obsahuje nasledovné tabuľky: `Route` (trasa medzi dvoma stanicami a ich vzdialenosť), `Station`, `Timeline` (rozvrh spoja, kedy je vlak na akej trase), `Train`.



Obr. 20: Schéma testovacej databázy TrainTimeline

B.1.3 LWP_Kartoteka_Vyvoj

Rozsiahla databáza s 594 tabuľkami a enormným počtom testovacích dát. Pri tomto zdroji dát je dobré sa najskôr zoznámiť so štruktúrou, pretože je ľahké napísať dopyt, ktorý sa bude kvôli veľkému množstvu dát vyhodnocovať dlho. Moje doporučenie je si najskôr vyskúšať dopyty na predchádzajúcich dvoch podstatne primitívnejších modeloch.

B.2 Príklady dopytov

Sekcia obsahuje sadu testovacích dopytov pre znázornenie syntaxe parametrov. V komentároch nad dotazom je uvedený model, typ parametru a možná predvolená hodnota, ktorou bude parameter nahradený. Predvolenú hodnotu formátuje program, užívateľ vidí box, napríklad s multiselectom, kde si vyberie hodnoty.

```
1  -- LWP_Kartoteka_Vyvoj (pocetVlakov: POSITIVE_INT, 50)
2  -- select first n number of trains
3  SELECT TOP(@pocetVlakov) [Vlak].Id,
4  [Vlak].Kod AS 'Kod vlaku',
5  [Vlak].Nazev AS 'Nazov vlaku'
6  FROM [Vlak]
7  ORDER BY [Vlak].Kod DESC
8
9  -- LWP_Kartoteka_Vyvoj (records_number: POSITIVE_INT, 50; code_used:
       POSITIVE_INT, 50)
10 -- top n records where code is used more than code_used times
11 SELECT TOP (@records_number)
12 [Vlak].Kod AS 'Kod vlaku', COUNT([Vlak].Kod) AS 'pocet vyskytov'
13 FROM [Vlak]
14 GROUP BY [Vlak].Kod
15 HAVING count([Vlak].Kod) > @code_used
16 ORDER BY count([Vlak].Kod) DESC;
```

Zdrojový kód 11: Príklady dopytov s parametrom typu pozitívne číslo

```
1  -- LWP_Kartoteka_Vyvoj (time: DATE, BETWEEN {d '2015-02-21'} AND {d
       '2015-12-21'} )
2  -- the first 99 warranties for the selected period
3  SELECT TOP(99)
4  [VozidloZaruka].ID,
5  [VozidloZaruka].Cyklus,
6  [Vozidlo].Kod,
7  [Vozidlo].PocetNaprav
8  FROM [VozidloZaruka]
9  JOIN [Vozidlo] ON [Vozidlo].ID = [VozidloZaruka].IdVozidlo
10 WHERE [VozidloZaruka].[CasVytvoreni] @time
11
12 -- TrainTimetable (time: DATE, BETWEEN {d '2021-02-21'} AND {d
       '2021-04-14'} )
13 -- returns trains that passed the section between Olomouc and Prague
        at the selected time
14 DECLARE @TrainIdStationFrom AS INT = 1 -- default value is Olomouc
15 DECLARE @TrainIdStationTo AS INT = 2 -- default value is Praha
16 SELECT TOP(999)
17 [Route].ID AS 'Route', [Timeline].Departure, [Timeline].Arrival
18 FROM [Route]
19 LEFT JOIN [Timeline] ON [Timeline].RouteID = [Route].ID
20 WHERE [Route].StartStationID = @TrainIdStationFrom AND
21 [Route].DestinationStationID = @TrainIdStationTo AND
22 [Timeline].Arrival @time
23 ORDER BY Route.ID ASC
```

Zdrojový kód 12: Príklady dopytov s parametrom obdobie pre dátum

```
1  -- LWP_Kartoteka_Vyvoj (pocetVlakov: CONDITION_INT, > 3)
2  -- Returns the names of stations that are transport nodes for the
       pocetVlakov trains.
3  SELECT TOP (20) [Stanice].Id AS 'Identifikator',
4  [Stanice].Nazev,
5  COUNT(*) AS 'Pocet vlakov prejde'
6  FROM [Vlak]
7  INNER JOIN [VlakTrasa] ON [VlakTrasa].IdVlak = [Vlak].Id AND
8          [Vlak].IdStaniceCilova = [VlakTrasa].IdStanice
9  INNER JOIN [Stanice] ON [Stanice].Id = [VlakTrasa].IdStanice
10 GROUP BY [Stanice].Nazev, [Stanice].Id
11 HAVING COUNT(*) @pocetVlakov
12
13 -- Restaurant (conditon: CONDITION_INT, >= 100)
14 -- return all meals where the price meets the condition
15 SELECT *
16 FROM [Meal]
17 WHERE [Meal].price @conditon
```

Zdrojový kód 13: Príklady dopytov s číselným parametrom pre podmienky

```
-- Restaurant (condition: CONDITION_STRING, = 'Pepper')
SELECT *
FROM [Ingredient]
WHERE [Ingredient].Name @condition;

-- LWP_Kartoteka_Vyvoj (condition: CONDITION_STRING, LIKE '\%tis\%')
-- Find all companies that contains particular part of string.
SELECT TOP (50)
[Id],
[Nazev],
[Mesto],
[Stav]
FROM [Firma]
WHERE [Firma].Nazev @condition

-- Restaurant (condition: CONDITION_STRING, >= 'v')
-- Select employees sorted alphabetically from __ position.
SELECT TOP (100)
[FirstName],
[Surname],
[PhoneNumber]
FROM [Employee]
WHERE [Surname] @condition
ORDER BY [Surname] ASC
```

Zdrojový kód 14: Príklady dopytov s textovým parametrom pre podmienky

```
-- LWP_Kartoteka_Vyvoj (vyberCielovychStanic : DB_MULTISELECT,
    ('114','117'); vyberVlakov : DB_MULTISELECT, ('7398', '5092',
    '3227')
SELECT
[Vlak].Cislo AS 'Vlak',
[StanicaCielova].Nazev AS 'Cielova stanica'
FROM [Stanice] AS StanicaCielova
JOIN [Vlak] ON [Vlak] .IdStaniceCilova IN @vyberCielovychStanic
where [Vlak].Id IN @vyberVlakov

-- Restaurant (employee_names: DB_MULTISELECT, ('Zeman','Novak','
    Varga'))
-- Select all information about all tables served by names staff.
SELECT *
FROM [Employee]
LEFT JOIN [Order] ON [Employee].ID = [Order]. EmployeeID
WHERE [Employee].ID IN @employee_names
```

Zdrojový kód 15: Príklady dopytov pre obecný číselník nad modelom

```
-- LWP_Kartoteka_Vyvoj (switch_code: CONDITION_INT, 2>1)
-- Return codes to ecv according to user preference.
SELECT [ECV].Id,
CASE WHEN [ECV].Kod IS NULL THEN 'Bez kodu'
ELSE [ECV].Kod
END AS 'Primarny kod' ,
IIF(@switch_code, [ECV].InventarniKod, [ECV].VyrobniKod) AS '
   Uschovny kod'
FROM [ECV]
ORDER BY [ECV].Id ASC;
```

Zdrojový kód 16: Príklady dopytov pre logický parameter True/False

```
-- LWP_Kartoteka_Vyvoj (stav: ENUMERATION, ( '0', '1' ))
-- Select top 9 users where @stav is in (condition).
SELECT TOP 9 *
FROM Uzivatel
WHERE Uzivatel.Stav IN @stav
ORDER BY id ASC
```

Zdrojový kód 17: Príklady dopytov s obecným výberom

```
-- LWP_Kartoteka_Vyvoj (pocet: POSITIVE_INT, 50; datum_jizdy;
   vyskyt_kodov)
-- Listing of the first n codes with occurrence, for selected
   stations for the selected period.
SELECT TOP (@pocet)
[Vlak].Kod AS 'Kod vlaku',
 COUNT([Vlak].Kod)
FROM [Vlak]
WHERE [Vlak].IdStaniceCilova IN @stanice AND
[Vlak].DatumJizdy @datum_jizdy
GROUP BY [Vlak].Kod
HAVING count([Vlak].Kod) > @vyskyt_kodov
ORDER BY [Vlak].Kod ASC

SELECT TOP (10)
[Vlak].Kod AS 'Kod vlaku',
 COUNT([Vlak].Kod)
FROM [Vlak]
WHERE [Vlak].IdStaniceCilova IN ('114','126','129','140','148','151'
   ,'152') AND
  [Vlak].DatumJizdy BETWEEN {ts '2017-04-02 07:04:00'} AND {ts '
     2021-04-30 07:04:00'}
GROUP BY [Vlak].Kod
HAVING count([Vlak].Kod) > 20
ORDER BY [Vlak].Kod ASC
```

Zdrojový kód 18: Príklad dopytu s viacerými typmi parametrov s ich nahradením

C Obsah priloženého CD

doc/
: Text práce vo formáte PDF bol vytvorený s použitím záväzného štýlu KI PRF UP v Olomouci pre záverečé práce, vrátane všetkých príloh pre bezproblémové vygenerovanie PDF dokumentu textu.

src/
: Kompletné zdrojové texty webovej aplikáce OGreport so všetkými potrebnými zdrojovými textami, knižnicami a ďalšími súbormi potrebnými pre bezproblémové vytvorenie spustiteľných verzií programov.

readme.txt
: Inštrukcie pre inštaláciu a spustenie aplikácie, vrátane všetkých požiadaviek na bezproblémovú prevádzku a webová adresa, na ktorej je aplikácia nasadená pre účel testovania, spolu s názvami účtov a príslušnými heslami.

data/
: Zálohy hlavnej databázy aplikácie a doplnkových testovacích databáz.

Literatúra

[1] FOWLER, Martin. Destilované UML. Grada Publishing, A.S., 2009 [cit. 2021-03-24], str. 23. ISBN: 978-80-247-2062-3. EAN: 9788024720623.

[2] LARKIN, Jill; SIMON Herbert. Why a Diagram is (Sometimes) Worth Ten Thousand Words?, 1987 [cit. 2021-04-24], str. 19. Cognitive Science.

[3] KNEZEK, Marián. Kedy servírujeme diagram tried?, [cit. 2021-04-24]. Dostupné z: https://www.like-it.sk/sk/n/39/uml-kedy-servirujeme-diagram-tried.html.

[4] GALLOWAY, Jon; MATSON, David; WILSON, Brad; ALLEN, Scott. Professional ASP.NET MVC 5, 1st Edition. WROX PRESS 2014, John Wiley & Sons Inc, [cit. 2021-03-13]. ISBN-13: 978-1118794753. ISBN-10: 1118794753.

[5] MICROSOFT. What is ASP.NET? [online]. 2021-01-01. [cit. 2021-03-13]. Dostupné z: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet.

[6] GEETANJALI, Arora ; BALASUBRAMANIAM, Aiaswamy; NITIN, Pandey. C# Professional Projects. Library of Congress Catalog Card Number: 2001096998. Premier Press, Inc.All 2002. [cit. 2021-03-13]. ISBN: 1-931841-30-6.

[7] FLANAGAN, David. JavaScript: the Definitive Guide, Sixth Edition. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2011 [cit. 2021-03-14]. ISBN: 978-0-596-80552-4.

[8] WEINBERG, Paul; GROFF, James; OPPEL, Andrew. SQL the Complete Reference, Third Edition. the McGraw-Hill Companies, 2010 [cit. 2021-03-14]. ISBN: 978-0-07-159256-7.