

VŠB – Technická univerzita Ostrava  
Fakulta elektrotechniky a informatiky  
Katedra informatiky

# **Absolvování individuální odborné praxe**

## **Individual Professional Practice in the Company**

## Zadání bakalářské práce

Student:

**Tomáš Chalupa**

Studijní program:

B2647 Informační a komunikační technologie

Studijní obor:

2612R025 Informatika a výpočetní technika

Téma:

**Absolvování individuální odborné praxe  
Individual Professional Practice in the Company**

Jazyk vypracování:

čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: AVE Soft s.r.o.
2. Struktura závěrečné zprávy:
  - a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
  - b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
  - c) Zvolený postup řešení zadaných úkolů.
  - d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
  - e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
  - f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.


Vedoucí bakalářské práce: **doc. Ing. Radim Bača, Ph.D.**

Konzultant bakalářské práce: Ing. Martin Prokeš

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2020



  
doc. Ing. Jan Platoš, Ph.D.  
vedoucí katedry

  
prof. Ing. Pavel Brandštetter, CSc.  
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární  
prameny a publikace, ze kterých jsem čerpal.

V Ostravě 15. května 2020

.....

Rád bych na tomto místě poděkoval všem mým kolegům v práci, mému vedoucímu práce, mému konzultantovi ve firmě a všem ostatním, kteří mi s prací pomohli.

## **Abstrakt**

Tato práce popisuje průběh mé bakalářské práce formou individuální odborné praxe ve firmě AVE Soft s.r.o.. Ve firmě jsem působil jako C# .NET DEVELOPER. Mým úkolem bylo vytvořit pomocnou aplikaci k již existujícímu informačnímu systému Evolio, která by zvládla pracovat s velkým množstvím dat.

**Klíčová slova:** Bakalářská praxe; Informační systém; C#; WPF

## **Abstract**

This paper describes my bachelor's thesis by form of individual apprenticeship in company AVE Soft s.r.o.. I worked in the company as C# .NET DEVELOPER. My task was to create a helper application for the already existing Evolio information system. This helper application would be able to work with a large amount of data.

**Keywords:** Bachelor thesis; Information system; C#; WPF

# Obsah

Seznam použitých zkratk a symbolů	7
Seznam obrázků	8
Seznam tabulek	9
Seznam výpisů zdrojového kódu	10
<b>1 Úvod</b>	<b>11</b>
<b>2 O firmě</b>	<b>12</b>
2.1 Startupy . . . . .	12
2.2 Firemní software . . . . .	13
<b>3 Moje úkoly</b>	<b>14</b>
3.1 Použité technologie . . . . .	14
3.2 Evolio Power Reporting . . . . .	15
<b>4 Využité a scházející znalosti a dovednosti</b>	<b>21</b>
4.1 Využité znalosti . . . . .	21
4.2 Scházející znalosti . . . . .	21
<b>5 Závěr</b>	<b>22</b>
<b>Literatura</b>	<b>23</b>

## Seznam použitých zkratek a symbolů

WPF	– Windows Presentation Foundation
HTML	– Hyper Text Markup Language
VB.NET	– Visual Basic .NET
ASP	– Active Server Pages
API	– Application Programming Interface
REST	– Representational State Transfer
SQL	– Structured Query Language
JSON	– JavaScript Object Notation
MVVM	– Model–view–viewmodel
XML	– Extensible Markup Language
XAML	– Extensible Application Markup Language
CDS	– Custom Data Source

## Seznam obrázků

1	Takto vypadá jeden z Filtrů v aplikaci Evolio Power Reporting. . . . .	15
2	Diagram aplikace Evolio Power Reporting a EDat. . . . .	16
3	ER diagram tabulek filtrů. . . . .	18
4	Ukázka editace filtru na záložce SQL dotaz. . . . .	20
5	Tohle je ukázka nastavení CSD 1. typu. . . . .	20



## Seznam tabulek

1	Porovnání serializačních formátů. . . . .	17
---	---	----

## Seznam výpisů zdrojového kódu

1	Serializace dat v controlleru. . . . .	16
2	Deserializace dat v Evolio Power Reporting. . . . .	17
3	MarkupExtension příkad . . . . .	18

# 1 Úvod

Svou bakalářskou práci jsem vykonával jako odbornou praxi u firmy AVE Soft s.r.o.. Praxi jsem preferoval oproti klasické bakalářské práci už od začátku. A AVE Soft s.r.o. měl zajímavou nabídku, hlavní pro mne bylo, že bych mohl uplatnit své zkušenosti s programováním v jazyce C#, ve kterém mě navíc pracovat baví.

V této práci popíši některé problémy a jak jsem je řešil. Například jak jsem streamoval data ze serveru do mé aplikace, něco o filtrech a akcích, zmíním problémy, na které jsem narazil a napíši, co mi je pomohlo vyřešit, dále pak co jsem se musel ještě doučit a nakonec zhodnotím, jak moje praxe dopadla.

## 2 O firmě

AVE Soft s.r.o.[1] je softwarová společnost založena roku 1997 vyvíjející informační systémy pro exekutory a právní kanceláře, kromě toho také vytváří software na míru. Mezi jejich zákazníky patří české pobočky prodejců automobilů Opel a životní pojišťovna Wustenrot, D.A.S. - Pojišťovna pro právní výdaje. AVE Soft s.r.o.[2] získala řadu ocenění, včetně druhého místa v soutěži české mobilní aplikace 2012 nebo finalisty Microsoft Industry Awards 2007 a 2012.

### 2.1 Startupy

Vedle klasický produktů v AveSoftu vznikly projekty, které svým přesahem a zaměřením vyžadovaly odlišný přístup. Proto je AVE Soft s.r.o. vytvořil jako plnohodnotné startupy.

#### 2.1.1 EXDRAZBY.CZ

V roce 2010 byl změněn zákon, který umožnil provádět dražby nemovitostí online. Byla to skvělá příležitost pro rozjezd nového projektu exdrazby.cz.

Po osmi měsících vývoje měly na trhu ostrou verzi dražebního portálu, který se během prvního roku stal největším na trhu, svou pozici v dalších letech ještě posílil. Klíčovým faktorem úspěchu byla sada nadstandardních služeb rozšiřující základní službu provedení aukce.

Za prvních pět let provozu byly prostřednictvím exdrazby.cz vydraženy nemovitosti v hodnotě více než 8 miliard korun.

#### 2.1.2 PRESENTIGO

Hlavní orientací tohoto startupu pod křídly AVE Soft s.r.o. je trh USA, z tohoto důvodu v Silicon Valley strávil téměř rok. Dnes se zákazníci nacházejí převážně v USA a ČR. Společnost získala investici ve výši 500 tisíc dolarů a našla zákazníky jako např. E.ON, O2, UPC, Škoda Auto apod.

Presentigo slouží pro zvýšení efektivity prodeje obchodním týmům. Firmám přináší dva základní benefity:

- Digitalizace obchodního procesu. Presentigo sbírá data z terénu, které pak využívají zaměstnanci marketingového oddělení i obchodní manažeři. Kromě toho dochází ke zjednodušení práce obchodníků.
- Na klíč jsou vytvářeny prezentace, které zákazníky zaujmou a zároveň pomohou dobře vysvětlit přínosy nabízeného produktu nebo služby.

## **2.2 Firemní software**

### **2.2.1 Evolio 8**

Evolio 8 je desktopová aplikace napsaná v C# WPF s jádrem z předchozí verze napsané v VB.NET. Jedná se o informační systém navržený pro použití v advokacii a příbuzných oborech.

### **2.2.2 Evolio**

Tento přímý nástupce Evolio 8 je webová aplikace napsaná technologií TypeScript, backend pak v C# ASP.NET Core MVC.

### **2.2.3 EData**

EData je webové REST API napsané v jazyce C# a postavené na technologii ASP.NET Core MVC. Slouží především k přístupu do databázi pro Evolio a Evolio Power Reporting.

### **2.2.4 EData Client**

EData Client je knihovna napsaná na ulehčení práce s API EData. Například přihlášení do něj nebo získání tabulky z databáze.

## 3 Moje úkoly

Mojí hlavní náplní praxe byl vývoj aplikace Evolio Power Reporting, sekundárně pak také oprava chyb aplikace Evolio 8.

### 3.1 Použité technologie

#### 3.1.1 MessagePack

MessagePack[3] je velmi rychlý binární serializační formát. Naštěstí už byl naimplementován pro C#, nebylo tedy nutné, abych tak učinil sám. Bohužel přetrvával problém, že jsem neměl k dispozici žádnou třídu do které bych data mohl serializovat, takže jsem je musel data mít v paměti jako pole polí objektů, kde každé pole objektů byl jeden řádek. Teoreticky bych si mohl za běhu vytvořit třídu a serializovat data do ní, ale pak bych k datům musel přistupovat přes reflexion API, což by vyústilo v problémy s výkonem. Zvolená implementace MessagePacku[4] dokonce podporovala kompresi pomocí LZ4. LZ4[5] je velmi rychlý bezztrátový kompresní algoritmus, který ještě nadále zrychlil přenos zmenšením přenášených dat.

#### 3.1.2 WPF

WPF[6] bylo vybráno hlavně kvůli tomu, že firma už vlastnila několik komponentů, které byly použity v předchozí aplikaci.

WPF podporuje návrhový vzor MVVM[7], který stejně jako další MVX vzory odděluje data od prezentační vrstvy, čehož je docíleno pomocí bindingu. Což je mechanismus, který synchronizuje data, která jsou ve View a ViewModel. To znamená, že pokud uživatel zadá něco do text boxu, tak se to propíše i do property, která je na daný TextBox navázána. Pokud ViewModel implementuje rozhraní „INotifyPropertyChanged“, bude tento proces fungovat i opačným směrem.

Pro to, aby binding fungoval, se nad každým View musí vyplnit property DataSource, což je náš ViewModel, a pro zachování separace mezi daty a vizuálem je toto děláno prostřednictvím vkládání závislostí. Tam si na začátku programu specifikuji k jakému View vytvořit jaký ViewModel a při vytvoření je spáruji. Pokud chce jakákoliv část programu komunikovat s tímto View, musí tak učinit přes spárovaný ViewModel.

View jsou napsána v značkovacím jazyce XAML[8], jenž je založený na XML. Kde jednotlivé tagy jsou reprezentace tříd v C# (např. „<sys:Int32>“ kde „sys“ je namespace „System“, je třída „Int32“ ). XAML umožňuje editaci za běhu programu, což bylo velmi užitečné při stylování.

Nejsložitější částí byly styly, hlavně kvůli používání komponent od třetích stran. Ty sice většinou podporovaly nějakou customizaci, ale ne vždy dostatečně. To spolu s mou neznalostí WPF/XAML zapříčinilo pro mě vysokou časovou náročnost.

## 3.2 Evolio Power Reporting

Evolio Power Reporting je desktopová aplikace, kterou jsem měl za úkol vytvořit. Tato část aplikace Evolio 8 (v ní pod názvem „Filtr“ více v podkapitole 3.2.6) by jako webová aplikace fungovala hůře. Filtry jsou uživatelem napsané SQL dotazy. Pro optimalizaci exekučního plánu jsou uloženy jako procedury. A nad daty které procedury vrací lze provádět hromadné operace (3.2.7 Akce), ať už s pomocí SQL nebo jiné interní systémy.

Vzhledem k tomu, že filtry mohou vrátit velké množství dat – až v řádech milionů záznamů, není webová aplikace pro jejich implementaci vhodná. Na obrázku 1 lze vidět filtr otevřený v aplikaci Evolio Power Reporting, nalevo lze vidět nastavení parametrů pro tento filtr a také 3 akce které lze provést nad tímto filtrem.

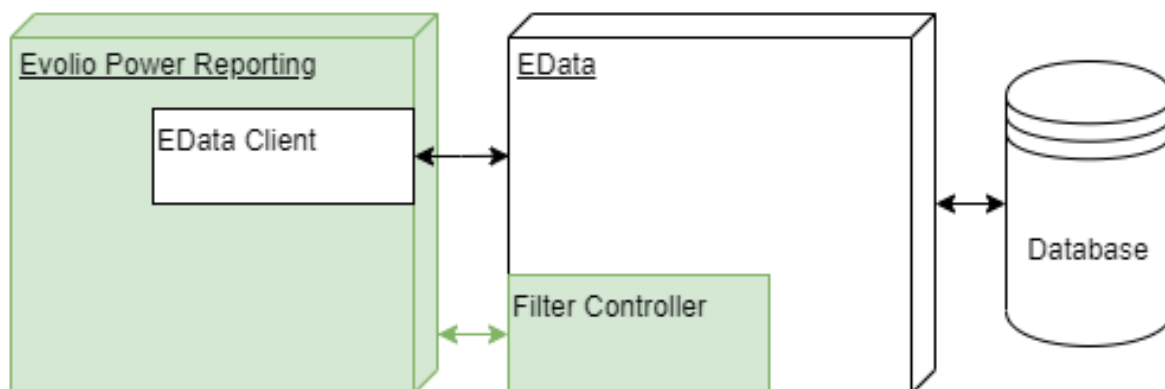
#	Akce	Případ/Vec	Dlužník	Klient	Zahájeno	Ukončeno	Stav	Uplatnění	Zůstatek	Měna	Vyřizuje	Zodpovídá	Zadal	Zadáno
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														

Obrázek 1: Takto vypadá jeden z Filtrů v aplikaci Evolio Power Reporting.

Poté na obrázku 2 lze vidět jak probíhá komunikace mezi Evolio Power Reporting, EData a databází.

### 3.2.1 Demo aplikace

Prvním krokem při vývoji této aplikace bylo zjistit, zda vůbec lze vytvořit dostatečně rychlou aplikaci, která by zvládla stáhnout a zobrazit milion záznamů stejně rychle nebo rychleji než předchozí aplikace, a zároveň je začala zobrazovat ihned poté, co je obdržela ze serveru – bez potřeby stáhnout je před zobrazením všechny, jako tomu bylo u předchozí aplikace Evolio 8. Taktéž tato demo aplikace mi sloužila jako seznámení s WPF, se kterým jsem do té doby nepracoval.



Obrázek 2: Diagram aplikace Evolio Power Reporting a EDat.

### 3.2.2 Filter Controller

K databázi jsem měl přístup jen přes EData, takže můj první úkol byl napsat controller, který by poskytoval data pro daný filter. Předat data z databáze controlleru bylo poměrně jednoduché – použil jsem SqlDataReader, který mi umožnil číst jednotlivé řádky a ihned je streamovat na klienta. Nakonec jsem ještě začal posílat data v malých skupinkách. Tato metoda byla rychlejší a pokud skupinky byly dostatečně malé, tak na klientovi ani nebylo z pohledu uživatele poznat, že načítání probíhá průběžně. Ve výpisu 1 je vidět zjednodušený kód s použitím formátu MessagePack.

---

```

var tempList = new object[ItemsPerBatch] [];
using (var data = cmd.ExecuteReader()){
    while (data.Read()){
        object[] obj = new object[data.FieldCount];
        data.GetValues(obj);
        tempList[count++] = obj;
        if (count >= ItemsPerBatch){
            LZ4MessagePackSerializer.Serialize(Response.Body, tempList);
            count = 0;
        }
    }
    LZ4MessagePackSerializer.Serialize(Response.Body, tempList.Take(count));
    LZ4MessagePackSerializer.Serialize(Response.Body, null);
}

```

---

Výpis 1: Serializace dat v controlleru.



### 3.2.3 Serializace

Jedno z prvních rozhodnutí, které jsem musel udělat, bylo v jakém formátu data přenášet (z EDat do mé aplikace). Plain text, nejjednodušší možnost, měla nevýhodu, že během přenosu nezachovala datový typ. Je sice možné parsovat string zpátky do jednotlivých typů, ale není to nejrychlejší – naopak je to velmi pomalé, a to natolik, že reálně tato varianta nešla použít a bylo potřeba zvolit jinou možnost.

Jako další možnosti se nabízel JSON nebo serializace do binárního kódu, při zkoumání těchto variant jsem narazil na MessagePack. A po trochu testování jsem zjistil že doopravdy je mojí nejrychlejší možností tak jsem se rozhodl ho použít.

Serializace	Rychlost <sup>1</sup> [s]	Velikost[MB]
MessagePack	114,26 (1) <sup>2</sup>	84,69 (1) <sup>2</sup>
MessagePack s LZ4	81,88 (0,7166) <sup>2</sup>	4,02 (0,0475) <sup>2</sup>
JSON	119,32 (1,0443) <sup>2</sup>	354,13 (4,1812) <sup>2</sup>

Tabulka 1: Porovnání serializačních formátů.

### 3.2.4 Deserializace

Na druhé straně v Evolio Power Reporting je nutné data deserializovat. Zde se vyskytl první problém, kde deserializace byla prováděna v asynchronní metodě, aby uživatelské rozhraní zůstalo responzní. To znamenalo, že tento kód běžel na jiném vlákně – jenže WPF nedovoluje úpravu prezenční vrstvy z jiného než hlavního vlákna (na které běží Dispatcher). Proto jsem musel použít Dispatcher.Invoke, aby Dispatcher pustil můj kód na správném vlákně.

---

```
using (var stream = await response.Content.ReadAsStreamAsync()){
    while (true){
        object[] [] batch = null;
        batch = LZ4MessagePackSerializer.Deserialize<object[] []>(stream, true);
        if (batch is null) { break; }
        else{
            Application.Current.Dispatcher.Invoke(Data.AddRange(batch));
        }
    }
}
```

---

Výpis 2: Deserializace dat v Evolio Power Reporting.

---

<sup>2</sup>Normalizované k MessagePack

<sup>1</sup>Čas je zprůměrován mezi 10 běhy (každý běh 1 000 000 záznamů)

### 3.2.5 Lokalizace

Poněvadž aplikace Evolio 8 byla používána nejen v České republice, lokalizace byla velmi důležitá. Překlady jsou uloženy v databázi, takže po přihlášení uživatele jsou staženy a použity v uživatelském prostředí. Také je důležité mít přístup k lokalizaci jak z C#, tak z XAML. Pro C# stačí statická metoda, ale pro XAML je potřeba napsat speciální třídu, v mém případě třída „Localization“, která bude dědit z „MarkupExtension“. To mi dovolí ji použít jako ve výpisu 3.

```
<Label Content="{Shared:Localization Key='PŘEKLADOVÝ.KLÍČ', Default='Defaultní text'}"/>
```

Výpis 3: MarkupExtension příklad

Kde „Shared“ je namespace ve kterém se nachází daná třída, „Localization“ je MarkupExtension třída, „Key“ je property na třídě „Localization“, která obsahuje překladový klíč a property „Default“ obsahuje výchozí text, který se zobrazí pokud překlad není dostupný.

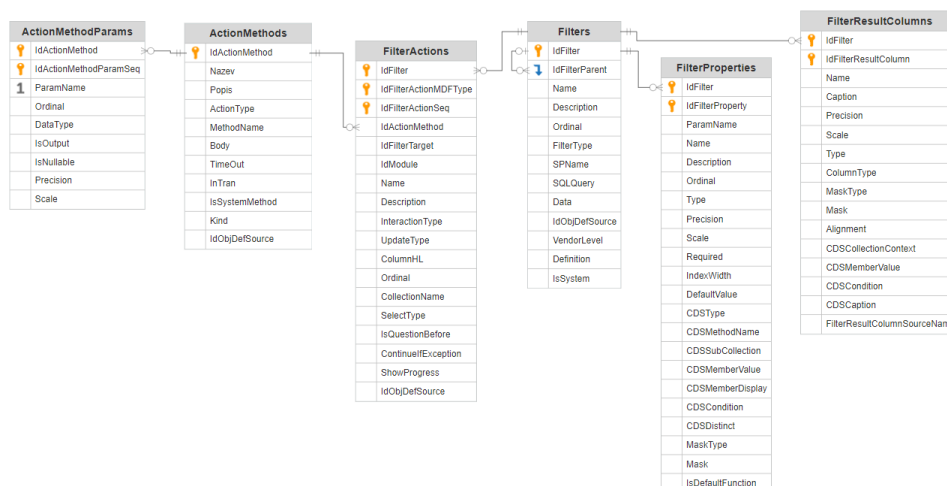
### 3.2.6 Filtry

Filter je rozdělen do několika tabulek, tabulka „Filters“ obsahuje jméno, SQL dotaz a jméno procedury spolu s dalšími nepovinnými daty.

Dále tabulka „FilterProperties“ obsahuje data pro jednotlivé property filtru údaje jako jsou jméno, datový typ, přesnost, velikost, zda je parametr povinný a také data pro CDS.

Tabulka „FilterResultColumns“ obsahuje data o sloupcích, které filtr vrátí. Ta jsou generována při ukládání nebo editaci, pomocí systémové SQL funkce *sys.sp\_describe\_first\_result\_set*, která bez toho, aniž by proceduru spustila, vrátí sloupce, které by procedura vrátila. Taktéž se zde ukládají data k zobrazení sloupců.

A nakonec tabulka „FilterActions“ obsahuje data k spárovaným akcím.



Obrázek 3: ER diagram tabulek filtrů.

### 3.2.7 Akce

Jsou 3 druhy akcí: uložené procedury, EService a Assembly. Uložené procedury se dají přidávat a editovat přímo z této aplikace, jsou to uložené procedury. EService jsou microservices založené nad technologií RabbitMQ[9]. Nakonec, pokud zákazník potřebuje nějakou funkci (např. uložit soubor na disk), která nelze uspokojit předchozíma dvěma druhy, použije se Assembly. Parametry pro tyto akce mohou pocházet ze 3 různých zdrojů: buď je to konstanta a vždy se použije stejná hodnota, nebo se hodnota může vzít z parametrů filtru, nebo z označeného sloupce (pokud chci poslat hodnotu ze sloupce "UserID", tak jednoduše specifikuji sloupec).

### 3.2.8 Editor filtrů

Další součástí této aplikace byl editor filtrů, který dovolil uživatelům přímo v aplikaci vytvářet a editovat filtry. Jelikož se edituje více než jedna tabulka najednou, tak toto musí být provedeno v transakci, aby se uložila buď všechna data, nebo žádná. Při uložení se také vytváří také procedura daného filtru, pokud uživatel napsal dotaz špatně, dojde v tomto bodě k selhání a vrátí se chybová hláška s uvedeným problémem; filtr se neuloží. To znamená, že není potřeba kontrolovat SQL kód na klientovi, což by jinak bylo nejspíše uděláno přes vhodnou knihovnu. Pokud se procedura úspěšně vytvoří, zavolá se na ni již zmíněná systémová SQL funkce *sys.sp\_describe\_first\_result\_set* a poté je filtr konečně uložen. Na obrázku 4 lze vědět jak editor vypadá v aplikaci Evolio Power Reporting.

### 3.2.9 CDS

CDS (custom data source) je velmi podobné typu Enum v C#, kde je hodnota – v případě Enumu číslo – a text, který vidí uživatel. CSD má dva typy, ten první je z tabulky, což znamená, že uživatel vybere dva sloupce z jedné tabulky, kde jeden bude text a druhý bude hodnota. Taktéž lze přidat podmínku pro výběr jen některých hodnot.

Pro uživatele by CSD z obrázku 5 vypadal jako combobox jmen zaměstnanců, kteří mají PlatceDPH nastaveno na false, proceduře by se ale poslalo CisloZamnestnance místo jména.

Druhý typ CSD je předem připravená procedura, pro složitější dotazy (např. když je nutno použít join) je potřeba vytvořit proceduru, která bude vracet páry textu a hodnot.

Všechny subjekty ✕

Základní údaje
SQL dotaz
Parametry
Akce
Sloupce
Oprávnění

```

1  select
2      s.IdSubjekt,
3      Subjekt = s.zaraditjako,
4      [Typ] = (Select Nazev from CSubjektyTypySubjektuVw where TypSubjektu = s.typosubjektu),
5      [Typ osoby] = (Select Nazev from CSubjektyTypyOsob where TypOsoby = s.typosoby),
6      Narozen = s.narozen,
7      [Datum úmrtí] = s.datumumrti,
8      [Rodné číslo] = s.rodne cislo,
9      IČ = s.ico,
10     Ulice = sa.Ulice,
11     CP = sa.CP,
12     CO = sa.CO,
13     Město = sa.Obec
14 from subjekty s left join SubjektyAdresy sa
15     on s.IdSubjekt = sa.IdSubjekt and sa.TrvalaAdresa = 1
16 order by s.zaraditjako

```

Naposledy upravil AveAdmin dne 10/18/2019 9:38:19 AM

ODSTRANIT
ZRUŠIT
ULOŽIT
ULOŽIT A ZAVŘÍT

Obrázek 4: Ukázka editace filtru na záložce SQL dotaz.

☐ Nic
☒ Kolekce
☐ Uložená Procedura

Kolekce	Zamestnanci
Hodnota	CisloZamestnance
Text	Jmeno
Podmínka	PlatceDPH = false
Distinct	<input type="checkbox"/>

Obrázek 5: Tohle je ukázka nastavení CSD 1. typu.

## **4 Využité a scházející znalosti a dovednosti**

### **4.1 Využité znalosti**

Při odborné praxi jsem využil několik znalostí získaných při studiu. Práci s C# a ASP.NET pomohly předměty „Architektura technologie .NET“ a „Programovací jazyky II“. S databází mi pomohly předměty „Úvod do databázových systémů“ a „Databázové a informační systémy“. A nejednou mi pomohla moje znalost verzovacího systému GIT.

### **4.2 Scházející znalosti**

Naopak když jsem začínal s praxí, postrádal jsem zkušenosti s WPF, jež jsem velmi rychle získal spolu s prací s XAML soubory. Taktéž jsem se dozvěděl mnoho o asynchronní programování. Dozvěděl jsem se, jak se vydávají a spravují aplikace. Dále jsem nikdy předtím nepracoval v týmu – práce ve více lidech byla také moje první (a velmi cenná) zkušenost.

## 5 Závěr

V této práci jsem popsal, jak jsem řešil některé z problémů, které nastaly při této odborné praxi a moje řešení. Během praxe jsem převážně pracoval na Evolio Power Reporting, ale naučil jsem se spoustu dovedností a zjistil jsem také jak probíhá práce v týmu. Aplikace je teď vydaná a u většiny zákazníků nahradila předchozí verzi (Evolio 8).

## Literatura

1. AVE SOFT S.R.O. *AVE Soft s.r.o.: Přehled / LinkedIn* [online]. 2001 [cit. 2018-09-02]. Dostupné z: <https://www.linkedin.com/company/ave-soft-s-r-o-/>.
2. AVE SOFT S.R.O. *O nás - AVE Soft.* [online]. 2001 [cit. 2018-09-02]. Dostupné z: <https://avesoft.cz/o-nas/>.
3. FURUHASHI, Sadayuki. *MessagePack: It's like JSON. but fast and small.* [online]. 2008 [cit. 2018-09-30]. Dostupné z: <https://msgpack.org/>.
4. KAWAI, Yoshifumi. *MessagePack: Extremely Fast MessagePack Serializer for Csharp* [online]. 2017 [cit. 2018-09-30]. Dostupné z: <https://github.com/neuecc/MessagePack-CSharp/>.
5. KRAJEWSKI, Milosz. *lz4: Extremely Fast Compression algorithm.* [online]. 2017 [cit. 2018-09-30]. Dostupné z: <https://github.com/lz4/lz4>.
6. *Windows Presentation Foundation*. San Francisco (CA): Wikimedia Foundation, 2001-2020. Dostupné také z: [https://en.wikipedia.org/wiki/Windows\\_Presentation\\_Foundation](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation).
7. *Model-view-viewmodel*. San Francisco (CA): Wikimedia Foundation, 2001-2020. Dostupné také z: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel>.
8. *Extensible Application Markup Language*. San Francisco (CA): Wikimedia Foundation, 2001-2020. Dostupné také z: [https://en.wikipedia.org/wiki/Extensible\\_Application\\_Markup\\_Language](https://en.wikipedia.org/wiki/Extensible_Application_Markup_Language).
9. *RabbitMQ*. San Francisco (CA): Pivotal Software, 2007-2020. Dostupné také z: <https://www.rabbitmq.com/>.