Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Práce s MySql v prostředí .NET

David Voska

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

David Voska

Informatika

Název práce

Práce s MySQL v prostředí .NET

Název anglicky

Using MySQL in .NET

Cíle práce

Práce je zaměřena na problematiku využití databáze MySQL v prostředí Microsoft .NET Framework. Hlavním cílem je vytvořit aplikaci umožňující práci s databázemi uloženými na MySQL serveru. Dílčím cílem je pak popsat práci s MySQL v .NET a související technologie a postupy.

Metodika

Metodika bakalářské práce je založená na studiu a analýze odborných informačních zdrojů. Na základě těchto poznatků a uživatelských požadavků bude zvoleno odpovídající technické řešení, proveden návrh a implementace aplikace pro práci s MySQL databází. Postup implementace bude popsán a výsledná aplikace bude otestována, zhodnocena a budou navrženy případné další možnosti jejího rozvoje.

Doporučený rozsah práce

35-40 stran

Klíčová slova

NET framework,C sharp,MySQL,ADO.NET,SQL dotaz,System.Data

Doporučené zdroje informací

Microsoft. MSDN [on-line]. 2015 [10.7.2015]. Dostupné z www: <http://msdn.microsoft.com/cs-CZ/>

MySQL Reference Manual [on-line]. 2015 [10.7.2015]. Dostupné z www: <http://dev.mysql.com/doc/refman/>

PETZOLD, Charles. Programování Microsoft Windows v jazyce C#. Praha: SoftPress, c2003. ISBN 80-86497-54-2.

ROBINSON, Simon. C#: programujeme profesionálně. Vyd. 1. Brno: Computer Press, 2003. ISBN 80-251-0085-5.

WELLING, Luke a Laura THOMSON. MySQL: průvodce základy databázového systému. Vyd. 1. Brno: CP Books, 2005. ISBN 80-251-0671-3.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 29. 02. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Práce s MySql v prostředí .NET", jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 3.3.2016 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Brožkovi Ph.D. za užitečné rady při tvorbě bakalářské práce.

Práce s MySql v prostředí .NET

Working with MySql in .NET environment

Souhrn

Teoretická část se zabývá popisem jazyka C Sharp, zejména .NET knihoven pro práci s jazykem SQL v prostředí MySql. Rovněž je zde věnován prostor jazyku SQL, který je v této práci také popsán.

Praktická část se skládá z kódu, využívající konektoru od Oracle, zprostředkující přístup do MySql a komponent jazyka C# zajišťující vizualizaci dat a jejich editaci.

Summary

Theoretical part deals with describing C Sharp .NET language, especially of libraries for working with SQL language in MySql environment. There is a space devoted to SQL language, which is described as well in this work.

Practical part is composed from code which use connector from Oracle, which provides access to MySql and components of C# language, which provides visualisation of data and it’s editing.

Klíčová slova:

.NET Framework, C Sharp, MySql, ADO .NET, SQL dotaz, System.Data

Keywords:

.NET Framework, C Sharp, MySql, ADO .NET, SQL query, System.Data

Obsah

1 Úvod

Většina podnikových aplikací má za úkol zobrazovat data z datových úložišť a poskytovat tak uživateli informace, které jsou důležité pro chod firmy a maximalizaci zisku, například přehled skladových zásob apod. Na softwarové úrovni dnes existuje mnoho řešení, jakým způsobem tato data spravovat a získávat z nich potřebné výstupy. Velké firmy jako například Oracle nabízí řešení v podobě MySql serveru jako úložiště dat a Microsoft se svým programovacím jazykem C# na platformě .NET, zase umožňuje vyvinout aplikaci, která požadovaná data zobrazí.

Autor práce se věnuje databázovému řešení od firmy Oracle, neboť se domnívá, že je nejrozšířenější. Používají ho sociální sítě jako Facebook, Twitter, YouTube a mnoho dalších významných webových hostingů. Jedná se o open source a pro nekomerční účely je zdarma.

Jazyk C# nabízí přehlednou syntaxi, dokumentaci a snaží se maximálně zjednodušit běžné programovací postupy, proto byl vybrán pro realizaci aplikace vizualizující data z MySql.

Obsluha databáze MySql pomocí jazyka C Sharp by měla být bezproblémová, jelikož existuje pro platformu .NET MySql connector, který zajistí spolupráci s databází. Na otázku, zdali tomu tak skutečně je, by měla odpovědět tato práce.

2 Cíl práce a metodika

Cílem práce je demonstrovat možnosti jazyka C Sharp pro práci s MySql databází a z programátorského hlediska zhodnotit náročnost celého řešení.

Teoretická část práce má za úkol popsat nástroje použité k realizaci aplikace za pomoci analýzy odborných informačních zdrojů, kterými jsou odborná literatura a internetové zdroje. Jako nástroj k realizaci aplikace, bude použito Visual Studio 2012, kde bude napsán kód aplikace, dále pak MySql server 5.6, ve kterém budou uloženy databázové tabulky. Tabulky budou vytvořeny prostřednictvím vizuálního rozhraní, jenž poskytne program MySql Workbench verze 6.3.

V praktické části práce je cílem sestavit aplikaci za pomocí knihoven jazyka C Sharp a popsat strukturu jejího kódu. Aplikace se bude umět připojit k libovolné instanci, databázi a konkrétní tabulce MySql serveru. Tuto tabulku bude umět prohlížet a editovat.

Na základě znalostí z teoretické části dojde k sestavení, otestování aplikace a syntézou poznatků při její tvorbě bude následně stanoven závěr.

3 Přehled řešené problematiky

3.1 .Net Framework

Platformě nezávislé prostředí, které poskytuje programovacím jazykům z kolekce .NET rozhraní pro komunikaci s operačním systémem a služby pro jejich obsluhu.
Každý zdrojový kód je pomocí kompilátoru příslušného jazyka přeložen do jazyka **MSIL**. Tento jazyk obsahuje sadu instrukcí nezávislých na procesoru.

3.1.1. Běhové prostředí (Common Language Runtime)

Slouží ke spuštění aplikace prostřednictvím kompilátoru **JIT**(Just in time), který kompiluje jazyk MSIL do strojového kódu pro daný procesor na vyžádání podle toho, která část aplikace je spuštěna.

3.1.2. Garbage collector

Stará se o správu paměti spuštěné aplikace, zajišťuje její alokaci a uvolnění. (Microsoft, 2015)

3.1.3. Komplety (Assembly)

Obsahují kód MSIL a ostatní soubory nutné ke spuštění aplikace. (Robinson, 2003)

3.1.4. Bázové třídy

Jsou společné pro všechny jazyky na bázi .NET, obsahují například definici datových typů. (Nash, 2010)

3.1.5. Jmenné prostory (Namespace)

Slouží k oddělení jmen tříd v jednotlivých projektech. Framework má jmenný prostor System. (Robinson, 2003)

3.2 Jazyk C Sharp

Je objektový, typově bezpečný programovací jazyk, provozovaný na platformě Framework, pomocí kterého můžeme vytvářet oknové, konzolové, nebo webové aplikace. Struktura aplikace je tvořena třídami, které slouží jako šablony pro objekty z reálného světa, které mohou obsahovat metody a data. Datové typy jsou dvojího druhu: hodnotové a referenční. (Nagel, 2009)

3.2.1. Hodnotové typy

Obsahují hodnotu přímo a jsou ukládány na místo v paměti nazývané „Stack". (Nagel, 2009)

3.2.2. Referenční typy

Nesou pouze odkaz na hodnotu a jsou ukládány v paměti nazývané „Heap". (Nagel, 2009).

Jako každý objektový jazyk i C Sharp implementuje základní vlastnosti objektových jazyků, kterými jsou Dědění, Polymorfismus a Zapouzdření. (Archer, 2001)

3.3 MySql

Je relační databázový stroj, který komunikuje s databází pomocí jazyka SQL. Jako rozhraní pro komunikaci je možné použít příkazový řádek, nebo grafické rozhraní MySql Workbench. (MySQL, 2015)

Podporuje Ansi SQL a ODBC SQL standard, přesto mají některé rozdíly oproti standardnímu SQL jako například odlišné chování příkazů select, update, v syntaxi komentářů, nebo v chování cizích klíčů. (MySQL, 2015) (Welling, a další, 2005)

3.4 Speciální syntaktické konstrukce v jazyce C#

3.4.1. Exception, try, catch

Ošetření chyb a výjimek v kódu je řešeno pomocí bloků try, catch ,třídy Exception a od ní odvozených tříd tak, že pokud se v kódu uvnitř bloku try vyskytne chyba, která je zabalena do třídy Exception, nebo odvozených tříd a vyhozena příkazem throw, tak tato chyba je následně zachycena v bloku catch, kde je možno definovat konkrétní typ třídy, který má být zachycen. Následně proběhne její programátorsky definované zpracování, nebo její předání do nadřazeného bloku try příkazem throw.
Pokud se chyba dostane až na místo, kde už nad sebou nemá žádný blok try, je automaticky ukončen běh celého programu a zobrazeno chybové hlášení. (Nagel, 2009)

3.4.2. Vlastnosti (Property)

Vlastnosti jsou veřejně přístupné prvky sloužící převážně pro čtení, či zápis do privátních proměnných. Obsahují speciální metodu get, která se využívá pro čtení a set pro zápis. Mohou fungovat i samostatně, aniž by byly svázány s privátní proměnnou. (Bayer, 2007)

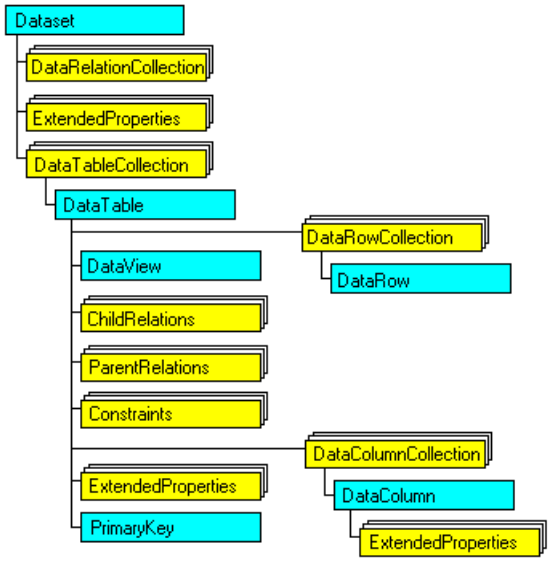
3.4.3. LambdaExpression

Jedná se o formu zápisu funkce, která se liší od standardního zápisu tím, že je anonymní a je možné ji psát přímo na místo vstupního argumentu. Obecný zápis je ve tvaru, kdy se na levé straně nacházejí vstupní parametry, následuje znak => a na pravé straně se zapisuje blok kódu, který se má vykonat. (Microsoft, 2015)

3.5 Obecné třídy pro práci s databází

Pro přístup k datům se v .NET Frameworku používá množina knihoven nazvaná ADO .NET. Třídy těchto knihoven najdeme ve jmenném prostoru System.data, obsahují rozhraní zprostředkovatele dat, pro obsluhu SQL databáze od Microsoft a univerzální OLEDB. Tyto rozhraní pracují přímo s příkazy SQL, které vykonají nad databází a jejich výsledky zanesou do příslušných objektů. (Nagel, 2009)

Třídou DataSet, je možno namodelovat strukturu databáze tak, jak je uložena v databázovém stroji. DataSet dovoluje pracovat s daty bez nutnosti být neustále připojen k databázi, změny se v ní provedou, až ve vhodný okamžik. Obsahuje obvykle několik tabulek třídy DataTable. Relace mezi tabulkami zajišťuje třída DataRelation a o omezení hodnot v tabulkách se stará třída Constraint. (Robinson, 2003)



Obrázek č. 1 Architektura objektu Dataset

Zdroj: (Microsoft, 2015)

3.6 Třídy pro práci s MySql

MySql provider není součástí ADO .NET, a proto musíme využít externí knihovny tzv. „Connectors“ jako zprostředkovatele pro přístup k databázi MySql. Nachází se ve jmenném prostoru MySql.data. Obsahuje několik důležitých tříd pro obsluhu databáze a plně spolupracuje s obecnými třídami v ADO .NET. (Schwartz, 2009)

3.6.1. MySqlConnection

Slouží pro spojení s databází MySql. Obsahuje dvě základní metody open a close, které otevírají a zavírají přístup k databázi. (MySQL, 2015)

3.6.2. MySqlCommand

Zajišťuje základní operace nad databází, mezi jeho nejdůležitější metody patří:

- **ExecuteReader**-používá se na vyhledávání v databázi.
- **ExecuteNonQuery**-slouží ke vkládání a mazání dat z databáze.
- **ExecuteScalar**-používá se k vyhledávání dat stejně jako ExecuteReader, ale s rozdílem, že vrací pouze jednu hodnotu. (MySQL, 2015)

3.6.3. MySqlCommandBuilder

Automaticky vytváří SQL příkazy INSERT, UPDATE a DELETE.

3.6.4. MySqlDataAdapter

Slouží jako prostředník mezi databází a třídou DataSet, ve které aktualizuje data z databáze. Obsahuje dvě základní metody Fill a Update, které se starají o synchronizaci dat mezi DataSetem a databází.

- **Fill-** přidává do kolekce DataRow záznamy z databáze, nebo je pouze aktualizuje.
- **Update-** zajišťuje synchronizaci záznamů z MySql databáze do DataTable na základě flagů výčtového typu RowState každého záznamu třídy DataRow v DataTable. Pokud je záznam označen flagem Deleted je volán DeleteCommand příslušného MySqlDataAdapteru a proveden SQL příkaz, který je v něm uložen**.**

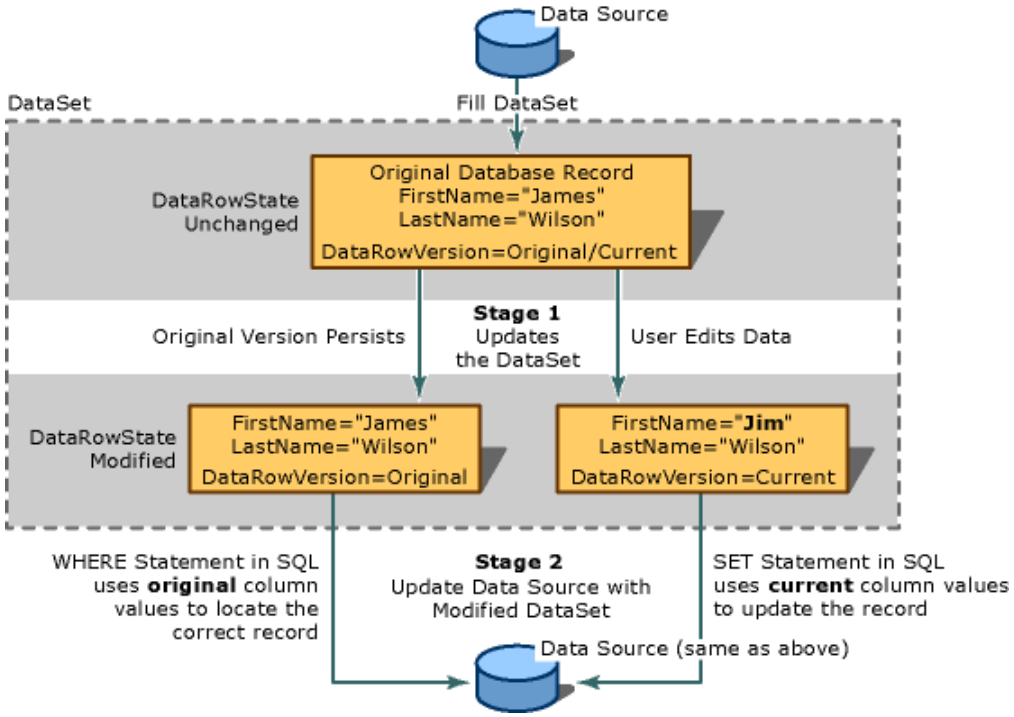
  V situaci, kdy je záznam označen flagem Modified**,** je volán UpdateCommand. (Microsoft, 2015) (MySQL, 2015)

RowState	Popis
Added	Záznam byl přidán do DataRowCollection a metoda AcceptChanges nebyla dosud volána.
Deleted	Záznam byl smazán metodou Delete třídy DataRow.
Detached	Záznam byl vytvořen, ale nepatří dosud do žádné DataRowCollection.
Modified	Záznam byl modifikován, ale metoda AcceptChanges nebyla dosud volána.
Unchanged	Záznam se nezměnil od posledního volání metody AcceptChanges.

Tabulka č. 1 Výčtový typ RowState

Zdroj: (Microsoft, 2015)

Rovněž obsahuje metodu FillSchema, která z databáze zjistí její strukturu, poté podle toho nastaví jednotlivé sloupce v DataTable, především primární klíče a jejich omezení. Pro ostatní sloupce pak povolení nulové hodnoty, povolení pouze pro čtení, unikátní hodnotu, délku a autoinkrementaci. (MySQL, 2015) (Microsoft, 2015)



Obrázek č. 2 Synchronizační proces DataSet< >Databáze

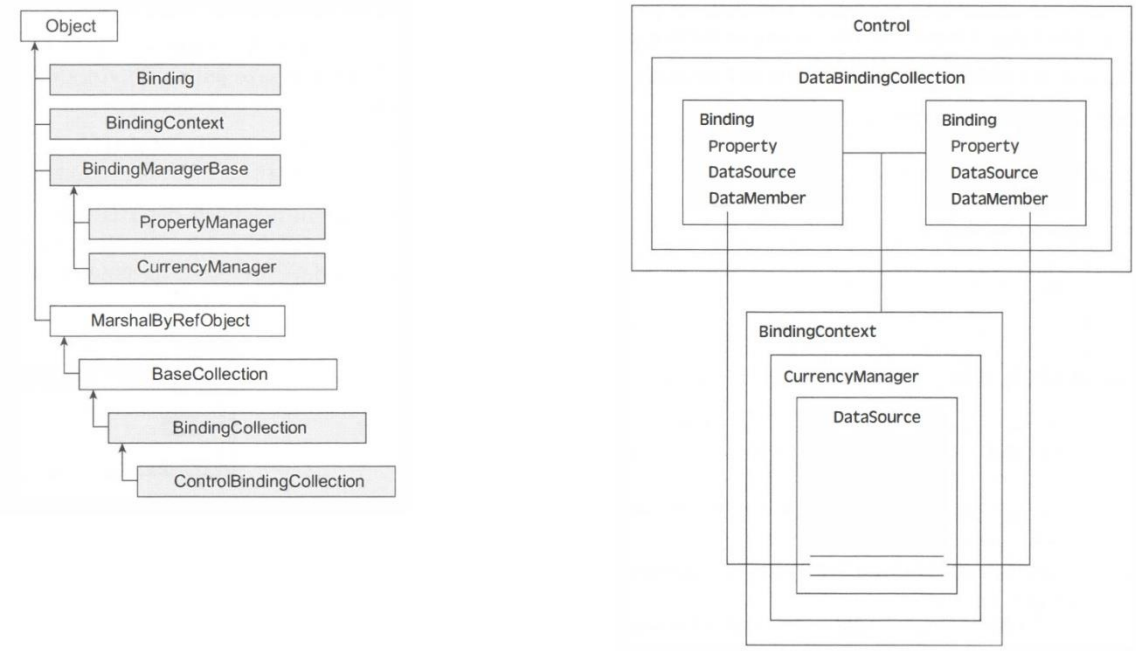
Zdroj: (Microsoft, 2015)

3.7 Třídy pro zobrazení dat v prostředí .NET

Pro práci byly zvoleny zobrazovací komponenty ze jmenného prostoru Windows.Forms, jenž obsahují komponentu Form, která tvoří základní stavební kámen pro uživatelský interface a slouží jako nosný prvek pro další komponenty. Nachází se zde i komponenta DatagridView, která dokáže znázornit data v tabulkovém zobrazení a spolupracuje s třídou DataSet. Jako pomocná komponenta bude použita třída SplitContainer, která dělí svůj vnitřní prostor na dvě části a bude nápomocná při umisťování DatagridView a Button komponent. (Nagel, 2009)

3.7.1. Datagrid a vázání dat

Data se s Datagridem propojují skrze Binding. Informace o zdroji dat se ukládá do objektu BindingContext, jenž je společný pro všechny prvky formuláře. Informace o tom, jaký zdroj dat a konkrétní člen zdroje dat bude zobrazen v prvku, jsou uloženy v objektu Binding, který se nachází přímo ve třídě DatagridView. (Nagel, 2009)



Obrázek č. 3 Architektura binding

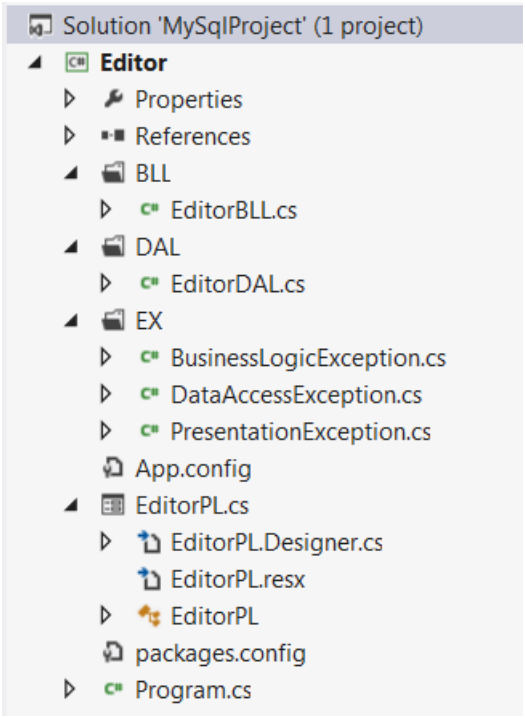
Zdroj: (Robinson, 2003)

4 Vlastní řešení

Aplikace je koncipována jako třívrstvá:

- **Datová vrstva (Data Access Layer)-** nachází se v adresář DAL. Obsahuje kód, který komunikuje s MySql databází.
- **Aplikační vrstva (Business Logic Layer)** -nachází se v adresáři BLL. Obsahuje kód, který zajišťuje logiku aplikace, zpracování dat, komunikuje se sousedními vrstvami.
- **Prezentační vrstva (Presentation Layer)-** je umístěna v souboru EditorPL.cs. Zajišťuje zobrazování obsahu uživateli.

Každá z vrstev má svojí vlastní třídu pro obsluhu výjimek. Aplikace je funkčně postavena tak, že ze statické třídy Program je spuštěna v novém vlákně Prezentační vrstva, která následně spustí ve svém vlákně Aplikační vrstvu a ta Datovou vrstvu. Vrstvy vytvoří potřebný interface mezi aplikací a MySql databází. Uživatel má následně k dispozici přes grafické rozhraní přehled o všech tabulkách vybrané databáze a následně po zvolení konkrétní tabulky z tohoto výčtu i samotnou vizualizaci dat z této tabulky. Pokud má tabulka primární klíč, je možné v záznamech provádět editaci a přidávat nové. Dále slouží k ovládání databáze pár tlačítek pro načtení a uložení záznamů v MySql databázi. Příkazy, které vykonávají tlačítka, jsou předávány prostřednictvím Aplikační vrstvy do Datové vrstvy. Jako nástroj pro vývoj je použito Visual Studio ve verzi 2012 pod knihovnou Framework 4.5. Samotná aplikace je umístěna ve jmenném prostoru MySqlProject a skládá se z několika souborů.



Obrázek č. 4 Struktura projektu

4.1 App.config

Obsahuje uživatelsky čitelné údaje, které slouží pro přístup do MySql databáze.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <clear />
    <add name="local" providerName="MySql"
         connectionString="server=localhost;user=root;database=test;port=3306;password=1234;" />
  </connectionStrings>
</configuration>
```

Kód č. 1 App.config

4.2 Program.cs

Obsahuje statickou třídu program a jedinou statickou metodu Main.

```
using System;
using System.Windows.Forms;
namespace MySqlProject
{
    static class Program
    {

        /// <summary> ...
        [STAThread]
        static void Main()...
    }

}
```

Kód č. 2 Struktura Třídy Program

4.2.1. Main

Je to první metoda, která se spustí po startu aplikace. Spouští grafické rozhraní v novém vlákně metodou Run, kde je jako parametr předána instance třídy EditorPL.

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new EditorPL());
}
```

Kód č. 3 Metoda Main

4.3 Data Access Layer

Obsahuje třídu EditorDAL, která reprezentuje Datovou vrstvu.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Configuration;
using MySql.Data;
using MySql.Data.MySqlClient;
using System.Data;
using MySqlProject.BLL;
using MySqlProject.EX;
namespace MySqlProject.DAL
{
    class EditorDAL
    {
        private DataSet _ds = null;
        private MySqlDataAdapter _da = null;
        private string _table = null;
        public string Table { get { return _table; } }
        public DataSet DataSet { get { return _ds; } }


        public EditorDAL()...
        private string GetConnectionString()...
        private MySqlDataAdapter GetAdapter()...
        private DataSet GetFillDataSet(MySqlDataAdapter adapter)...
        public void SelectTable(string tableName)...
        public void UpdateDatabase()...
        public void RefreshData()...
        public DataTable GetSqlTables()...
    }
}
```

Kód č. 4 Struktura třídy EditorDAL,vrstva DAL

Proměnné, které jsou určeny pouze pro potřeby této vrstvy a nesou označení private. Dalším prvkem jsou Vlastnosti, které slouží pouze pro čtení a předávání dat do vyšší vrstvy.

Table vrací název tabulky, kterou chceme v MySql prohlížet, či editovat.
Dataset vrací objekt DataSet, jenž bude použit jako prostředník mezi databází a GUI.

4.3.1. GetConnectionString

Vrací objekt ConnectionStringSettings, obsahující údaje nutné pro přístup k MySql databázovému stroji. Údaje jsou uloženy v souboru App.config, z něhož data získává třída ConfigurationManager (Drayton, a další, 2003)

```
private  string GetConnectionString()
{
    string conString = "";
    ConnectionStringSettings connectionStringSetting =
        ConfigurationManager.ConnectionStrings["local"];
    conString = connectionStringSetting.ConnectionString;
    return conString;
}
```

Kód č. 5 Metoda GetConnectionString, vrstva DAL

4.3.2. GetAdapter

Nastaví spojení s databází a vrací objekt typu MySqlDataAdapter. Nejprve je vytvořena instance třídy MySqlConnection. Ta je uložena do proměnné connection jako parametr konstruktoru, je vložena proměnná connectionString, která je naplněna metodou GetConnectionString a určuje tak cestu k databázi. Dále se vytvoří instance třídy MySqlDataAdapter. Uloží se do proměnné adapter. Třída je inicializovaná konstruktorem s dvěma parametry. Prvním je SQL příkaz projekce, uložený v proměnné select a sloužící k definování dat, jenž mají být uložena v DataTable pro případné zobrazení v Prezentační vrstvě. V této práci se používá SQL příkaz vybírající všechny sloupce z dané tabulky, uložené ve vlastnosti Table. Druhý parametr je proměnná connection, následně je vrácen objekt adapter. (Welling, a další, 2005) (Bayer, 2007)

```
private MySqlDataAdapter GetAdapter()
{
    MySqlDataAdapter adapter = null;
    string connectionString = GetConnectionString();
    MySqlConnection connection = new MySqlConnection(connectionString);
    string select = "select * FROM " + Table;
    adapter = new MySqlDataAdapter(select, connection);
    return adapter;
}
```

Kód č. 6 Metoda GetAdapter, vrstva DAL

4.3.3. GetFillDataSet

Vrací objekt DataSet, jehož DataTable předtím naplní daty z MySql databáze, jenž budou prezentovány uživateli. Vytvoří příkazy jazyka SQL pro obsluhu databáze a nastaví sloupce v DataTable tak, aby korespondovaly s vlastnostmi sloupců v MySql databázi. Vstupním parametrem metody je instance typu MySqlDataAdapter, na které závisí instance tříd pracujících s databází. První je vytvořena instance třídy DataSet a uložena do proměnné dataSet. Instance třídy MySqlCommandBuilder se vytváří pro zajištění obsluhy databáze, jako je mazání, vkládání a editace záznamu. Uloží se do proměnné cb. Konstruktorem třídy je proměnná typu MySqlDataAdapter, tu reprezentuje proměnná adapter. Na základě SQL příkazu SELECT, který je uložen v instanci MySqlDataAdapteru, jsou vygenerovány příkazy INSERT, UPDATE a DELETE. (MySQL, 2015)

Aby bylo zabráněno vkládání nesprávných dat uživatelem, je potřeba nastavit omezení (constraint) na sloupce DataTable. To je zajištěno metodou FillSchema. Tato metoda nastaví vlastnosti sloupců v DataTable podle MySql databáze a to zejména primární klíče, jenž uloží do vlastnosti PrimaryKey řešené tabulky DataTable. PrimaryKey je pole instancí tříd typu DataColumn, a z těch je složena celá DataTable. Jako první parametr je vložena instance třídy DataSet, obsahující tabulku se sloupci, které chceme nastavit. Následuje určení toho, jakým způsobem mají být vlastnosti sloupců aplikovány. Je předán výčtový typ SchemaType.Source ,který stanoví, že se veškeré předchozí vlastnosti, přepíší těmi novými a jako poslední je vložen název tabulky, ve které budou změny aplikovány. Data jsou načtena z databáze pomocí metody Fill proměnné dataSet, jíž jsou předány parametry. První parametr definuje DataSet, který se má naplnit daty a druhý parametr určuje jméno tabulky DataTable v DataSetu, na kterou budou data mapována.

Jakmile jsou data z databáze uložena v DataTable, je nutno zjistit, zdali se zde nenachází sloupec s primárním autoinkrementačním klíčem. Tato informace je nezbytná z důvodu, že komponenta DataGridView v editačním módu potřebuje znát jedinečný primární klíč, který identifikuje každý řádek tabulky v DataTable pro zajištění korektní manipulace s daty, zejména jejich mazání a editaci. Problém nastává v případě, že máme tabulku v MySql s autoinkrementačním primárním klíčem a při vkládání záznamu přes komponentu DataGridView dojde v MySql tabulce k vytvoření záznamu spolu s primárním klíčem. Tento klíč se nepřenese do DataGridView, respektive do DataTable,

protože se pracuje s odpojenou datovou sadou, a ta nereflektuje změny v MySql databázi, pokud ji k tomu nedáme příkaz, nebo nezajistíme při vkládání nového záznamu návratovou hodnotu v podobě vygenerovaného primárního klíče.

Detekci přítomnosti primárního autoinkrementačního klíče v DataTable zajišťuje metoda FirstOrDefault, vracející objekt typu DataColumn, pokud je klíč nalezen, nebo null pokud v řešené tabulce není přítomen alespoň jeden primární klíč, který má zapnutou autoinkrementaci. Jako parametr metody je pomocí LambdaExpression vložena funkce, která pro každý objekt typu DataColumn zjišťuje, zdali jeho vlastnost AutoIncrement nese hodnotu true, která značí, že se jedná o klíč vygenerovaný automaticky. Pokud proměnná autoIncrementPK neobsahuje hodnotu null, znamená to, že existuje v řešené MySql tabulce primární klíč s autoinkrementací, který zajistí, že každý vložený řádek do MySql databáze bude mít unikátní číselnou hodnotu, navýšenou o defaultní hodnotu, definovanou při zakládání tabulky, o což se stará engine MySql databáze.

Pokud je autoinkrementační klíč v tabulce přítomen, je nutné nastavit MySqlAdapter a sloupec v DataTable tak, aby DataGridView a DataTable dostávaly korektní záznamy o hodnotách primárního klíče vytvořeného MySql databázovým strojem. Zajišťuje ho blok kódu, který udělá kopii MySqlCommandu INSERT metodou MySqlCommandBuilderu GetInsertCommand.Clone, uloží ji do proměnné insertCmd a modifikuje ji tak, že upraví vlastnost CommandText proměnné insertCmd přidáním SQL příkazu SELECT last_insert_id(), který se provede po vygenerovaném příkazu pro vkládání nových záznamů. Takový příkaz vrací poslední automaticky vygenerovanou číselnou hodnotu, v tomto případě primárního klíče. Aby se hodnota zapsala do správného sloupce v DataTable, je potřeba navíc přidat za zmiňovaný příkaz alias se jménem sloupce, který koresponduje s názvem primárního klíče v DataTable. To zajistí příkaz AS a název sloupce primárního klíče, který je předán z proměnné autoIncrementPK prostřednictvím vlastnosti ColumnName. Vlastnost UpdatedRowSource proměnné insertCMD slouží k nastavení toho, jak má být mapována data pro aktualizaci, v tomto případě hodnoty primárního klíče, která jsou vrácena z databáze po přidání záznamu do MySql tabulky na DataTable. Vzhledem k použitému SQL příkazu last_insert_id(), byl zvolen typ UpdateRowSource.FirstReturnedRecord, jenž předpokládá návrat jednoho záznamu. Modifikovaná třída MySqlCommand je uložena do vlastnosti InsertCommand proměnné adapter. Aktuálně je nutné zajistit, aby uživatel nemohl zapisovat do sloupce, kde se

nachází autoinkrementační klíč, protože tato hodnota je zajištěna MySql databázovým strojem. Toho je docíleno nastavením vlastnosti ReadOnly třídy DataColumn na true. Vrácen je kompletně inicializovaný DataSet. (Microsoft, 2015) (Nagel, 2009) (MySQL, 2015) (Schwartz, 2009)

```
private DataSet GetFillDataSet(MySqlDataAdapter adapter)
{
    DataSet dataSet = new DataSet();
    MySqlCommandBuilder cb = new MySqlCommandBuilder(adapter);
    adapter.FillSchema(dataSet, SchemaType.Source,Table);
    adapter.Fill(dataSet, Table);
    DataColumn autoIncrementPK = null;
    autoIncrementPK = dataSet.Tables[Table].
        PrimaryKey.FirstOrDefault(x => x.AutoIncrement == true);
    if (autoIncrementPK != null)
    {
        MySqlCommand insertCmd = cb.GetInsertCommand().Clone();
        insertCmd.CommandText = insertCmd.CommandText +
            ";SELECT last_insert_id() AS "
            + autoIncrementPK.ColumnName;
        insertCmd.UpdatedRowSource = UpdateRowSource.FirstReturnedRecord;
        adapter.InsertCommand = insertCmd;
        autoIncrementPK.ReadOnly = true;
    }
    return dataSet;
}
```

Kód č. 7 Metoda GetFillDataSet, vrstva DAL

4.3.4. SelectTable

Slouží k nastavení tabulky, s kterou bude aplikace dále pracovat. Metoda má vstupní parametr, který nese jméno tabulky.

Nastaví proměnnou _table na jméno tabulky. Pokud se v bloku vyskytne chyba, je vytvořena výjimka DataAccessException a spolu s původní výjimkou vyhozena na vyšší úroveň toku aplikace.

```
public void SelectTable(string tableName)
{
    try
    {
        _table = tableName;

    }
    catch (Exception ex)
    {
        DataAccessException dataEx = new DataAccessException
            ("Toto je chyba v Datové vrstvě,vzniklá v metodě SelectTable()", ex);
        throw dataEx;
    }

}
```

kód č. 8 Metoda SelectTable, vrstva DAL

4.3.5. UpdateDatabase

Tato metoda zajišťuje synchronizaci nových, nebo změněných záznamů v DataTable s MySql databází.

```
public void UpdateDatabase()
{
    try
    {
        _da.Update(DataSet, Table);
    }
    catch (Exception ex)
    {
        DataAccessException dataEx = new DataAccessException
            ("Toto je chyba v Datové vrstvě,vzniklá v metodě UpdateDatabase()", ex);
        throw dataEx;
    }
}
```

Kód č. 9 Metoda UpdateDatabase, vrstva DAL

4.3.6. RefreshData

Zajišťuje načtení dat z MySql databáze do DataTable.

První krok je získání MySqlDataAdapteru, který zajistí metoda GetAdapter. Tato metoda inicializuje proměnnou _da. Proměnná _da je využita jako vstupní parametr metody GetFillDataSet, která naplní DataTable, umístěnou v DataSetu daty z MySql (Schwartz, 2009) (Archer, 2001)

```
public void RefreshData()
{
    try
    {
        _da = GetAdapter();
        _ds = GetFillDataSet(_da);
    }
    catch (Exception ex)
    {
        DataAccessException dataEx = new DataAccessException
            ("Toto je chyba v Datové vrstvě,vzniklá v metodě RefreshData()", ex);
        throw dataEx;
    }
}
```

Kód č. 10 Metoda RefreshData, vrstva DAL

4.3.7. GetSqlTables

Metoda vrací seznam tabulek databáze MySql. Je vytvořen objekt DataTable, který je pouze inicializován. Nyní je zapotřebí připravit spojení s MySql databází. Spojení je zajištěno přes třídu MySqlConnection. Spojení je třeba otevřít metodou Open. K získání informací o tabulkách je využito metody GetSchema s parametrem TABLES, která zajistí načtení parametrů o tabulkách v databázi, která je dána aktuálním databázovým připojením. Poslední část kódu vrací DataTable s daty o tabulkách.

ColumnName	DataType	Description
table_catalog	String	Catalog of the table.
table_schema	String	Schema that contains the table.
table_name	String	Table name.
table_type	String	Type of table. Can be VIEW or BASE TABLE.

Tabulka č. 2 Seznam sloupců, vrácených metodou GetSchema("TABLES")

Zdroj: (Microsoft, 2015)

```
public DataTable GetSqlTables()
{
    DataTable sqlTable=new DataTable();
    string connectionString = GetConnectionString();
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();
    sqlTable = connection.GetSchema("TABLES");
    return sqlTable;
}
```

Kód č. 11 Metoda GetSqlTables, vrstva DAL

4.4 Business Logic Layer

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using MySql.Data;
using MySql.Data.MySqlClient;
using System.Data;
using MySqlProject.DAL;
using MySqlProject.EX;
namespace MySqlProject.BLL
{
    class EditorBLL
    {
        EditorDAL dataLogic=null;
        public EditorBLL()...
        public DataTable GetSqlTables()...
        public DataTable GetDataTable(string tableName)...
        public int GetPrimaryKeysCount()...
        public void RefresDataTable()...
        public void UpdateDatabase()...
    }
}
```

Kód č. 12 Struktura třídy EditorBLL, vrstva BLL

Obsahuje třídu EditorBLL, která reprezentuje Aplikační vrstvu. Proměnná dataLogic reprezentuje přístup k Datové vrstvě.

Kód metod nacházejících se v bloku try, se chová tím způsobem, že pokud se v bloku vyskytne chyba, je vytvořena výjimka BusinessAccessException a spolu s původní výjimkou vyhozena na vyšší úroveň toku aplikace. (Microsoft, 2015)

4.4.1. EditorBLL

Představuje konstruktor vrstvy, který v bloku try inicializuje proměnnou, potřebnou pro komunikaci s Datovou vrstvou.

```
public EditorBLL()
{
    try
    {
        dataLogic = new EditorDAL();
    }
    catch (Exception ex)
    {
        BusinessLogicException bussinesEx = new BusinessLogicException
    ("Toto je chyba v Bussines vrstvě,vzniklá v konstruktoru EditorBLL()", ex);
        throw bussinesEx;
    }
}
```

Kód č. 13 Konstruktor EditorBLL

4.4.2. GetDataTable

Vrací instanci třídy DataTable, naplněnou daty MySql tabulky prostřednictvím instance Datové vrstvy reprezentované proměnnou dataLogic. Vstupní parametr je jméno tabulky, s kterou bude aplikace dále pracovat.

Nejprve je použita metoda z Datové logiky SelectTable nastavující jméno tabulky, dostupné prostřednictvím vlastnosti Table. Dalším krokem je spuštění metody RefreshData, zajišťující načtení dat z MySql do DataTable v Datové logice. Jakmile je DataTable naplněn, vybereme jej z DataSetu a následně vrátíme.

```
public DataTable GetDataTable(string tableName)
{
    try
    {
        dataLogic.SelectTable(tableName);
        dataLogic.RefreshData();
        return dataLogic.DataSet.Tables[dataLogic.Table];
    }
    catch (Exception ex)
    {
        BusinessLogicException bussinesEx = new BusinessLogicException
            ("Toto je chyba v Bussines vrstvě,vzniklá v metodě GetDataTable()",ex);
        throw bussinesEx;
    }
}
```

Kód č. 14 Metoda GetDataTable, vrstva BLL

4.4.3. GetPrimaryKeysCount

Vrací číslo reprezentující počet primárních klíčů v řešené tabulce DataTable. Využívá se metody Count vlastnosti PrimaryKey řešené tabulky, která počítá počet instancí tříd DataColumn.

```
public int GetPrimaryKeysCount()
{
    try
    {
        return dataLogic.DataSet.Tables[dataLogic.Table].PrimaryKey.Count();
    }
    catch (Exception ex)
    {
        BusinessLogicException bussinesEx = new BusinessLogicException
            ("Toto je chyba v Bussines vrstvě,vzniklá v metodě  GetPrimaryKeysCount()", ex);
        throw bussinesEx;
    }
}
```

Kód č. 15 Metoda GetPrimaryKeysCount, vrstva BLL

4.4.4. RefreshDataTable

Spouští z Datové vrstvy metodu RefreshData.

```
public void RefresDataTable()
{
    try
    {
        dataLogic.RefreshData();
    }
    catch (Exception ex)
    {
        BusinessLogicException bussinesEx = new BusinessLogicException
            ("Toto je chyba v Bussines vrstvě,vzniklá v metodě RefresDataTable()", ex);
        throw bussinesEx;
    }
}
```

Kód č. 16 Metoda RefreshDataTable, vrstva BLL

4.4.5. UpdateDatabase

Spouští z Datové vrstvy metodu UpdateDatabase.

```
public void UpdateDatabase()
{
    try
    {
        dataLogic.UpdateDatabase();
    }
    catch (Exception ex)
    {
        BusinessLogicException bussinesEx = new BusinessLogicException
            ("Toto je chyba v Bussines vrstvě,vzniklá v metodě UpdateDatabase()", ex);
        throw bussinesEx;
    }
}
```

Kód č. 17 Metoda UpdateDatabase, vrstva BLL

4.5 Presentation Layer

Nachází se v parciální třídě EditorPL a dědí z třídy Form.

Dědění způsobí, že se třída stává základním nosným prvkem uživatelského rozhraní.První část parciální třídy se nachází v souboru EditorPL.cs a obsahuje programátorem definovaný kód, který se nachází v bloku try. Chová se tak, že pokud se v bloku vyskytne chyba, je vytvořena výjimka PresentationException, zformátována metodou FormatException a zobrazena ve formě Messageboxu, který je vyvolán pomocí metody Show statické třídy MessageBox uživateli. Uživatelské rozhraní je sestaveno z horizontálního a vertikálního SplitContaineru, dvou GridViews a dvou tlačítek. Ve vertikálním SplitContaineru se nachází nalevo GridViewTables sloužící k zobrazení tabulek. Buňky tohoto GridView reagují na levý klik myší tak, že se spustí kód, který zobrazí data té tabulky, jejíž jméno je v buňce obsaženo. V databázi a napravo GridViewData zobrazující samotná data vybrané tabulky.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using MySqlProject.BLL;
using MySqlProject.EX;
namespace MySqlProject
{
    public partial class EditorPL : Form
    {
        EditorBLL bussinesLogic = null;
        public EditorPL()...
        private string FormatException(Exception ex)...
        private void buttonSave_Click(object sender, EventArgs e)...
        private void buttonLoad_Click(object sender, EventArgs e)...
        private void GridViewTables_CellClick(object sender, DataGridViewCellEventArgs e)...
    }
}
```

Kód č. 18 Třída EditorPL, vrstva PL

4.5.1. EditorPL

Konstruktor zajišťuje inicializaci veškerých komponent formuláře pomocí metody InitializeComponent(), nacházející se v druhé části parciální třídy EditorPL. Vytváří instanci třídy EditorBLL, jenž zajišťuje přístup k metodám Aplikační vrstvy. Jakmile je vytvořena instance třídy Aplikační vrstvy, inicializují se objekty grafického rozhraní. Po tomto kroku, je zapotřebí nastavit GridViewTables tak, aby sám negeneroval sloupce z DataTable, protože metoda GetSqlTables vrací DataTable s více sloupci, než je zapotřebí a tyto sloupce by se pak zobrazily v grafickém rozhraní. Pomocí vlastnosti DataPropertyName nabindujeme příslušný sloupec v GridView na sloupec v DataTable. Přidáme vytvořený sloupec do kolekce sloupců Gridu a tím bude následně viditelný v grafickém rozhraní. (Drayton, a další, 2003)

```
public EditorPL()
{
    try
    {
        bussinesLogic = new EditorBLL();
        InitializeComponent();
        this.GridViewTables.AutoGenerateColumns = false;
        this.GridViewTables.DataSource = bussinesLogic.GetSqlTables();
        DataGridViewColumn tablesColumn = new DataGridViewTextBoxColumn();
        tablesColumn.DataPropertyName="TABLE_NAME";
        tablesColumn.HeaderText="Tabulky";
        this.GridViewTables.Columns.Add(tablesColumn);
    }
    catch (Exception ex)
    {
        PresentationException uiEx = new PresentationException
            ("Toto je chyba v Prezentační vrstvě,vzniklá v konstruktoru PresentationTier()", ex);
        MessageBox.Show(FormatException(uiEx));
    }
}
```

Kód č. 19 Konstruktor EditorPL, vrstva PL

4.5.2. FormatException

Slouží k zobrazení chybových hlášení z jednotlivých vrstev, jak byly postupně předávány výše v toku programu. Vrací chybové hlášení třídou String. Jako vstupní parametr je požadována instance třídy Exception, která je pak zpracována pomocí rekurze tím způsobem, že se nejprve ověří, zdali je třída Exception inicializovaná. Pokud ano, tak se do proměnné message aktuální instance rekurze uloží z proměnné Message třídy Exception, která byla předána parametrem, chybové hlášení řešené vrstvy. Na konec chybového hlášení, se vloží znak nového řádku, což zajistí, že chybové hlášení bude přehledně zobrazeno uživateli. Spustí se další instance FormatException, které se předá vlastnost InnerException z řešené výjimky, která může obsahovat další vnořenou výjimku, jenž se týká řešené vrstvy, nebo se může jednat o výjimku z předchozí vrstvy. Tento proces se opakuje, dokud řešená výjimka neobsahuje již žádnou další vnořenou výjimku. Volání rekurze se poté ukončí a nastane sbírání nashromážděných dat v rekurzivním stromu. Vrátí se chybové hlášení počáteční výjimky vyvolávající všechny ostatní, to se předá předchozí instanci rekurze FormatException, která si ho připojí ke svému chybovému hlášení a pokračuje se takto až ke kmeni rekurze, jenž vrací kompletní chybovou zprávu. (Archer, 2001)

```
private string FormatException(Exception ex)
{
    string message = "";
    if (ex != null)
    {
        message = ex.Message + "\n";
        message += FormatException(ex.InnerException);
    }
    return message;
}
```

Kód č. 20 Metoda FormatException, vrstva PL

4.5.3. ButtonSave_Click

Metoda je eventHandlerem pro tlačítko, jehož proměnná má název buttonSave. Zabezpečuje zpracování události Click, která nastane po stisknutí tlačítka buttonSave. Metoda buttonSave provede spuštění metody z Aplikační vrstvy UpdateDatabase, ukládající do databáze MySql změny provedené v DataTable, způsobené uživatelem v DataGridView. Pokud uživatel modifikoval záznam na DatagridView, je nastaven databindingem RowState flag v DataTable na hodnotu Deleted, jestliže změnil záznam, pak je nastaven Modified a pokud přidával záznam, tak je nastaven na Added. (Nash, 2010)

```
private void buttonSave_Click(object sender, EventArgs e)
{

    try
    {

        bussinesLogic.UpdateDatabase();
    }
    catch (Exception ex)
    {
        PresentationException uiEx = new PresentationException
            ("Toto je chyba v Prezentační vrstvě,vzniklá v metodě buttonSave_Click", ex);
        MessageBox.Show(uiEx.Message);
    }

}
```

Kód č. 21 Metoda buttonSave_Click, vrstva PL

4.5.4. ButtonLoad_Click

Metoda je eventHandlerem pro tlačítko, jehož proměnná má název buttonLoad. Zpracovává události Click nastávající po stisknutí tlačítka buttonLoad. Tato metoda provede spuštění metody z Aplikační vrstvy RefreshDataTable, jenž zahájí synchronizaci databáze a DataGridu, čímž se uživateli zobrazí aktuální data z databáze.

```
private void buttonLoad_Click(object sender, EventArgs e)
{
    try
    {
        bussinesLogic.RefresDataTable();
    }
    catch (Exception ex)
    {
        PresentationException uiEx = new PresentationException
            ("Toto je chyba v Prezentační vrstvě,vzniklá v metodě buttonLoad_Click", ex);
        MessageBox.Show(uiEx.Message);
    }
}
```

Kód č. 22 Metoda buttonLoad_Click,vrstva PL

4.5.5. GridViewTables_CellClick

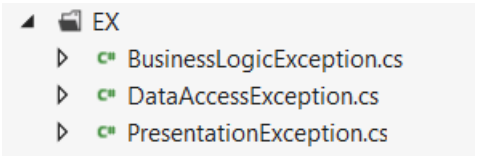
Metoda zpracovává událost, způsobenou kliknutím levým tlačítkem myši na buňku příslušného řádku v GridView, který má na starosti zobrazování a výběr tabulky z MySql databáze Z argumentu, předaného této události je vybrán textový údaj, který reprezentuje jméno vybírané MySql tabulky. Jméno tabulky je následně předáno metodě GetDataTable, která vrací požadovanou DataTable s daty. Tento zdroj dat, je předán DataGridu, čímž se zajistí nabindování dat mezi DataTable a Gridem. Nyní je třeba nastavit, zdali má být komponenta DataGridView v módu pouze pro čtení, protože může nastat situace, že nebude k dispozici žádný sloupec v DataTable plnící funkci primárního klíče a nebylo by tak možno určit, který záznam má být upravován. O tom, jestli bude DataGridView pouze v módu pro čtení, rozhoduje počet primárních klíčů v řešeném DataTable, tuto informaci podává metoda Aplikační vrstvy GetPrimaryKeys.Count. Pokud je počet klíčů menší, nebo roven 0, DatagridView je nastavena pomocí vlastnosti ReadOnly na hodnotu true.

```
private void GridViewTables_CellClick(object sender, DataGridViewCellEventArgs e)
{
   string value=GridViewTables.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString();
     this.GridViewData.DataSource = bussinesLogic.GetDataTable(value);
     if (bussinesLogic.GetPrimaryKeysCount() <= 0)
     this.GridViewData.ReadOnly = true;
     else this.GridViewData.ReadOnly = false;
}
```

Kód č. 23 Metoda GridViewTables_CellClick, vrstva PL

4.6 Exceptions

Třídy výjimek se nachází ve jmenném prostoru MySqlProject.EX a zajišťují identifikaci výjimky v příslušné vrstvě. Všechny výjimky mají jako předka třídu Exception, což umožňuje využívat vlastností klasické výjimky, jako je vyhazování o úroveň výše (throw), zachytávání (catch), apod. Jejich struktura se skládá z bezparametrického konstruktoru spouštějícího bázový konstruktor třídy Exception. Další konstruktor umožňuje parametrem vkládat chybovou zprávu, která se objeví jako hlášení v MessageBoxu uživateli, pokud se dostane do Prezentační vrstvy. Poslední z konstruktorů rozšiřuje kromě chybové zprávy i možnost vložit vnitřní výjimku (innerException) a umožňuje přenést výjimku vzniklou například v jiné vrstvě dále (Microsoft, 2015) (Virius, 2012).



Obrázek č. 5 Třídy výjimek

4.6.1. BusinessLogicException

```
class BusinessLogicException : Exception
{
    public BusinessLogicException():  base()
    {
    }
    public BusinessLogicException(string message)
        : base(message)
    {
    }
    public BusinessLogicException(string message, System.Exception inner)
        : base(message,inner)
    {
    }
}
```

Kód č. 24 Třída BusinessLogicException

4.6.2. DataAccessException

```
class DataAccessException : Exception
{
    public DataAccessException():  base()
    {
    }
    public DataAccessException(string message)
        : base(message)
    {
    }
    public DataAccessException(string message, System.Exception inner)
        : base(message,inner)
    {
    }
}
```

Kód č. 25 Třída DataAccessException

4.6.3. PresentationException

```
class PresentationException : Exception
{
            public PresentationException():  base()
    {
    }
    public PresentationException(string message)
        : base(message)
    {
    }
    public PresentationException(string message, System.Exception inner)
        : base(message,inner)
    {
    }
}
```

Kód č. 26 Třída PresentationException

4.7 Testování aplikace

K testování byl použit MySql server 5.6, v němž byly vytvořeny následující tabulky simulující možné scénáře v reálném prostředí.

```
CREATE TABLE `nopkey` (
  `idNoPkey` int(11) DEFAULT NULL,
  `NoPkeycol` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Obrázek č. 6 MySql tabulka bez primárního klíče 'nopkey'

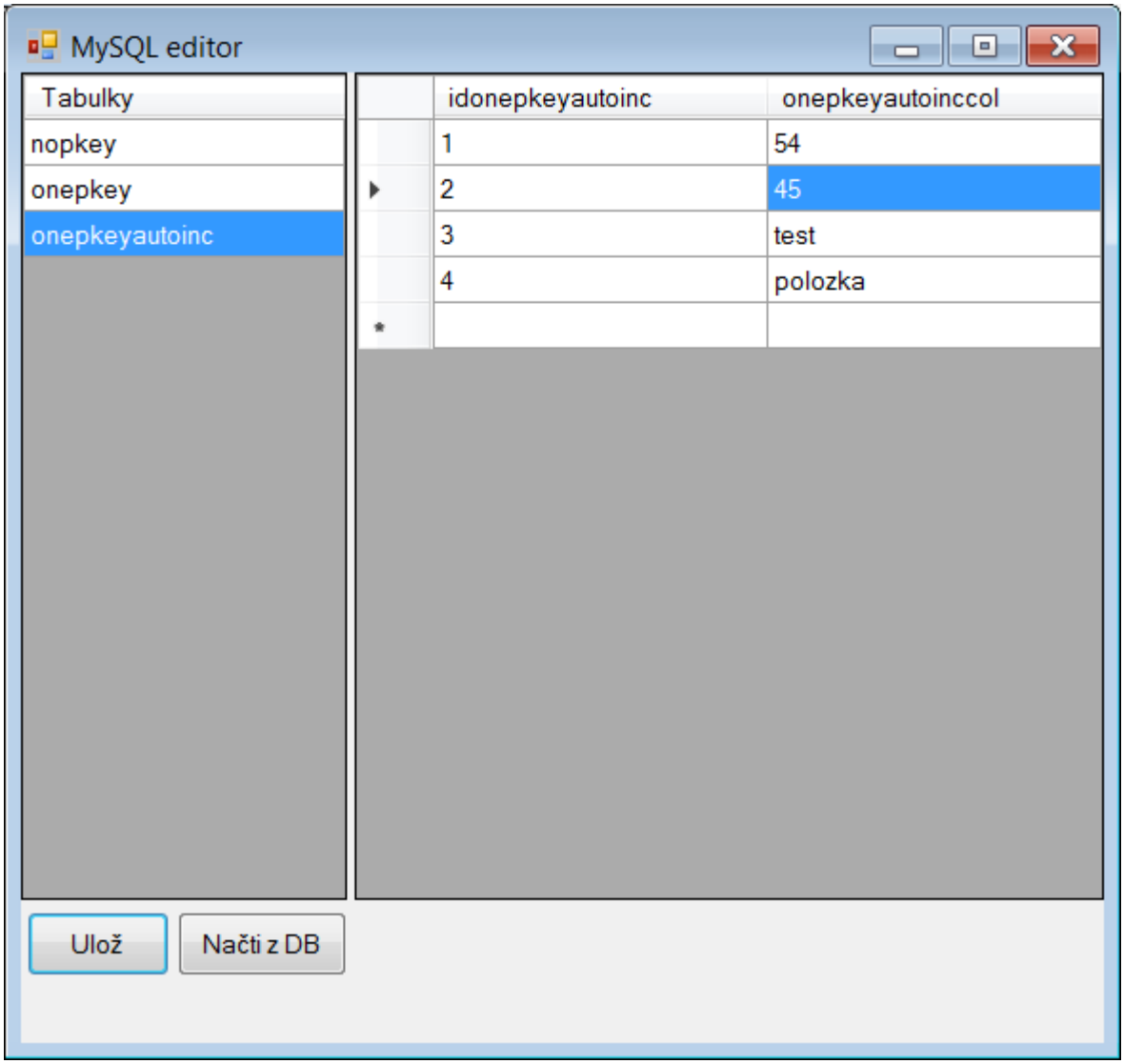
```
CREATE TABLE `onepkey` (
  `idonepkey` int(11) NOT NULL,
  `onepkeycol` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idonepkey`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Obrázek č. 7 MySql tabulka s jedním primárním klíčem 'onepkey'

```
CREATE TABLE `onepkeyautoinc` (
  `idonepkeyautoinc` int(11) NOT NULL AUTO_INCREMENT,
  `onepkeyautoinccol` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idonepkeyautoinc`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Obrázek č. 8 MySql tabulka s jedním primárním klíčem a autoinkrementací 'onepkeyautoinc'

V prvním případě došlo po načtení dat k přepnutí tabulky do čtecího módu. V druhém případě byla tabulka v editačním módu s možností vkládat, jak záznam primárního klíče, tak varcharová data do posledního sloupce. V třetím případě byla tabulka v editačním módu se zákazem vkládat primární klíč. Ve všech situacích se aplikace chovala korektně.



Obrázek č. 9 Spuštěná aplikace se zvolenou tabulkou 'onepkeyautoinc'

5 Závěr

Cílem práce bylo sestavit aplikaci na bázi jazyka C Sharp, jenž bude schopna prohlížet i editovat libovolnou tabulku v MySql databázi a posoudit z programátorského hlediska náročnost sestavení aplikace.

V teoretické části byla popsána vývojová platforma. Net Framework, její základní součásti a mechanismus fungování. Autor se rovněž zmínil o jazyku C#, jeho charakteristických vlastnostech a syntaktických strukturách. Byla popsána knihovna konektoru MySql s třídami, které byly v práci použity.

V praktické části byl popsán kód jazyka C#, který byl rozdělen na několik kapitol, které reflektují třívrstvou strukturu aplikace. V kapitole App.config byl popsán soubor s nastavením aplikace. V kapitole Program.cs byl popsán vstupní bod programu. V kapitole Data Access Layer byla popsána Datová vrstva, sloužící pro přístup do databáze. V kapitole Bussiness Logic Layer byla popsána vrstva zajišťující zprostředkování komunikace mezi vrstvami. V reálné aplikaci by se zde prováděli výpočty s daty, ale v tomto případě jde převážně o předávání řízení mezi vrstvami. V kapitole Presentation Layer bylo popsáno grafické rozhraní aplikace, které není úplně dokonalé, nicméně pro účely této práce vyhovující. V kapitole Exceptions byl popsán mechanismus výjimek, pro indikaci chyb v aplikaci.

Byla sestavena aplikace za pomoci Visual studia 2012. Byly vytvořeny tři tabulky pomocí aplikace WorkBench s různými scénáři, které by mohly nastat v reálné situaci a otestovány na aplikaci.

Ve všech případech se aplikace chovala dle předpokladu autora. Během programování aplikace se autor nesetkal se závažnými překážkami ztěžujícími vývoj aplikace. Na základě těchto skutečností je možno konstatovat, že propojení MySql se C Sharpem je řešení doporučitelné pro svoji časově nenáročnou a bezproblémovou implementaci. Autor si je vědom toho, že tato aplikace je pouze demonstrativní a ukazuje základní techniky, které by se dali využít v reálné aplikaci využívající přístup do databáze. Aplikace by se mohla upravit například na plnohodnotný nástroj pro správu MySql databáze, kde by se dalo pracovat se strukturami databáze komplexněji, například vytvářet nové databáze, nebo tabulky.

Cíl práce uvedený v počátku byl naplněn tak, jak se autor ve své prvotní tezi domníval.

6 Citovaná literatura

Archer, Tom. 2001. *Myslíme v jazyku C#.* Vyd. 1. Praha : Grada, 2001. str. 307. ISBN 80-247-0301-7.

Bayer, Jürgen. 2007. *C# 2005: velká kniha řešení.* Vyd. 1. Brno : Computer Press, 2007. str. 813. ISBN 978-80-251-1620-3.

Drayton, Peter, Neward, Ted a Ben , Albahari. 2003. *C# v kostce: pohotová referenční příručka.* Vyd. 1. Praha : Grada, 2003. str. 764. ISBN 80-247-0443-9.

Microsoft. 2015. MSDN. [Online] 2015. Dostupné z www:<http://msdn.microsoft.com/cs-CZ/>.

MySQL. 2015. Reference Manual. [Online] 2015. Dostupné z www:<http://dev.mysql.com/doc/refman/>.

Nagel, Christian. 2009. *C# 2008: programujeme profesionálně.* Vyd. 1. Brno : Computer Press, 2009. str. 1126. Programujeme profesionálně. ISBN 97880251240171-2.

Nash, Trey. 2010. *C# 2010: rychlý průvodce novinkami a nejlepšími postupy.* Vyd. 1. Brno : Computer Press, 2010. str. 624. ISBN 978-80-251-3034-6.

Robinson, Simon. 2003. *C#: programujeme profesionálně.* Vyd. 1. Brno : Computer Press, 2003. str. 1130. Programujeme profesionálně. ISBN 80-251-0085-5.

Schwartz, Baron. 2009. *MySQL profesionálně: optimalizace pro vysoký výkon.* Vyd. 1. Brno : Zoner Press, 2009. str. 712. ISBN 978-80-7413-035-9.

Virius, Miroslav. 2012. *C# 2010: hotová řešení.* Vyd. 1. Brno : Computer Press, 2012. str. 424. ISBN 978-80-251-3730-7.

Welling, Luke a Thompson, Laura. 2005. *MySQL: průvodce základy databázového systému.* Vyd. 1. Brno : CP Books, 2005. str. 255. ISBN 80-251-0671-3.

7 Seznam obrázků

8 Seznam tabulek