VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Informační systém pro správu účtů databázových systémů

IS for Account Management of Database Systems

2019 Martin Gaier

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Gaier**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Informační systém pro správu účtů databázových systémů
IS for Account Management of Database Systems

Jazyk vypracování: čeština

Zásady pro vypracování:

Katedra informatiky poskytuje výuku šesti předmětů v oblasti databázových systémů. Kromě toho je každoročně vypsáno množství bakalářských, diplomových a dizertačních prací v této oblasti. Pro tyto účely má katedra k dispozici několik serverů, na kterých jsou nainstalované databázové systémy. V akademickém roce 2014/2015 byl v rámci diplomové práce vytvořen webový systém nazvaný DB Manager, který spravuje uživatelské účty databázových systémů. DB Manager má nedostatky, které byly objeveny až po nasazení do ostrého provozu, cílem práce je odstranění těchto nedostatků.

Hlavní úkoly práce jsou:
1. Seznámit se se stávajícím webovým portálem a prací s jednotlivými databázovými systémy.
2. Provést analýzu používaných funkcí a navrhnout novou datovou vrstvu.
3. Navrhnout a s využitím nových technologií naimplementovat novou prezentační vrstvu systému.
4. Doladit nedostatky jednotlivých podsystémů (např. hromadné vytváření účtů, logování, atd.).
5. Vytvořit online nápovědu.
6. Nasadit novou verzi prezentační a datové vrstvy do ostrého provozu.
7. Srovnat původní i novou verzi webového portálu.

Seznam doporučené odborné literatury:

[1] Adam Freeman. Pro ASP.NET MVC 5 (Expert's Voice in ASP.Net) 5th ed. Edition.
[2] Jay A. Kreibich. Using Sqlite (1st ed.). O'Reilly Media, Inc., 2010.

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

Vedoucí bakalářské práce: **Ing. Peter Chovanec, Ph.D.**

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

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 30.dubna 2019 ..... Gaier

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2019 ..... Gaier .....

Rád bych poděkoval *Ing. Petru Chovancovi, Ph.D*, *Ing. Martinu Zwierzynovi* a *Bc. Davidu Taborovi* za odborné konzultace při tvorbě této bakalářské práce.

Abstrakt

Motivem, pro vznik této práce bylo odstranění nedostatků dosavadního systému pro administraci databázových účtů katedry informatiky a jeho modernizace. Mezi tyto problémy patřilo přesunutí systémové databáze z SQLite na Microsoft SQL Server, úprava vzhledu webového rozhraní, změna systému zaznamenávání událostí a přesun na technologii .NET Core. Byla vytvořena nová datová vrstva, pomocí frameworku EF Core, založená na původní databázi. Práce s propojenými databázovými systémy je nyní sjednocena do jednotlivých modulů, které se načítají za běhu, což umožňuje jejich změny bez nutnosti odstavení systému. Nové webové rozhraní je inspirováno původním, ale je kompletně předěláno pomocí technologií MVC, Bootstrap, JQuery a DataTables. Celý systém je plně lokalizován pro češtinu a angličtinu. Přihlašování je řešeno kontrolou přihlašovacích údajů pomocí školního adresářového serveru LDAP.

Klíčová slova: Informační systém, uživatelské databáze, databázový systém, přihlašování LDAP, technologie ASP.NET Core, Microsoft SQL Server, Entity Framework Core, MVC, Bootstrap, JQuery, DataTables. lokalizace, zaznamenávání událostí, e-mailové notifikace, synchronizace účtů, asynchronní operace, dynamické načítání knihoven

Abstract

The motive for the creation of this work was to eliminate the shortcomings of the existing system for the administration of database accounts of the Department of Informatics and its modernization. These issues included moving the system database from SQLite to Microsoft SQL Server, adjusting the appearance of the web interface, changing the event logging system, and migrating to .Net Core. A new data layer was created, using the EF Core framework based on the original database. Working with linked database systems is now consolidated into individual modules that load during runtime, allowing them to be changed without shutting down the system. The new web interface is inspired by the original, but is completely redesigned with MVC, Bootstrap, JQuery, and DataTables. The whole system is fully localized for Czech and English. Logging in is handled by checking login data using the LDAP school directory server.

Key Words: Information system, user databases, database system, LDAP login, ASP.NET Core technology, Microsoft SQL Server, Entity Framework Core, MVC, Bootstrap, JQuery, DataTables. localization, event logging, email notification, account synchronization, asynchronous operation, dynamic library loading

Obsah

Seznam použitých zkratek a symbolů

MVC	–	Model View Controller
LDAP	–	Lightweight Directory Access Protocol
SQL	–	Structured Query Language
IIS Express	–	Internet Information Service Express server
Server IIS	–	Server Internet Information Services
RAM	–	Random Access Memory
RWM-RAM	–	Read Write Memory - Random Access Memory
HTML	–	Hypertext Markup Language
CSS	–	Cascading Style Sheets
JS	–	JavaScript

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Databázové systémy patří mezi nejrozšířenější metodu ukládaní a sdílení informací. Většina systémů, které pracují s klientskými údaji, je využívá. Může se jednat o uložení údajů formuláře nebo zaznamenávání teploty místnosti, jednoduše všude tam, kde se informace (neboli data) přenáší mezi zařízeními. V širším slova smyslu je databáze hardwarové médium a softwarové prostředky, které umožňuji manipulaci s daty a přístup k nim. V dnešní době se databáze stará o ukládání a výpis dat i o operace s daty. Například třídění, shlukování a výpočty. Dále se moderní databáze starají i o zabezpečení dat, čím se dostáváme k problému se spravováním přístupu k jednotlivým databázím. Jedná-li se o malý systém s několika uživateli, není problém jej spravovat manuálně přímo z databáze. Pokud ale systém obsahuje mnoho uživatelů nebo i více databázových systému, stává se jeho správa až neúnosnou.

S tímto problémem se potýkají i správci databázových systémů na Katedře Informatiky, Fakulty elektrotechniky a informatiky, kde každý semestr vzniká několik stovek účtu, na pěti různých databázových systémech. Tyto účty jsou používané pro výuku mnoha předmětů, jako *Úvod do databázových systémů*, *Databázové a informační systémy I a II*, *Databázové systémy*, *Administrace databázových systémů*, *Informační systémy a datové sklady*, *Algoritmy vykonávání dotazů* či *Fyzická implementace databázových systémů*. Každý rok se také vytváří několik účtu pro bakalářské, diplomové nebo dizertační práce, zabývajících se databázovou problematikou, které také využívají některé z databázových účtů. Z toho důvodu byl vymyšlen informační systém pro správu databázových účtů *DB Manager*, který správu databázových účtu zjednodušoval pomocí webového rozhraní napojeného na jednotlivé databáze.

Informační systém *DB Manager* byl vytvořen (Ing. Martinem Zwierzynou) jako bakalářská práce v roce 2013, a poté rozšířen jako diplomová práce v roce 2015. Systém byl nasazen, a po vyřešení nedostatků využíván katedrou informatiky. Informační systém zvládal vytvářet účty jak samostatně, tak i hromadně, spravoval žádosti o vytváření či změny účtu, hesla pro přihlašování, notifikace pomocí e-mailu, a později také synchronizaci. Bohužel se nepodařilo vyřešit veškeré nedostatky a systém byl příliš rozsáhlý pro jejich opravu. Většina použitých technologií již byla překonána, webové technologie pokročily a spolu s úpravou webu *DB Edu*, byly důvodem pro předělání systému.

Cílem této práce je vytvořit nový systém, který umožní přesun systému ze stávající souborové SQLite databáze, na server s Microsoft SQL Databází. Vyřešení problémů se zaznamenáváním chyb a událostí systému při spouštění databázových procedur. Zjednodušit práci s jednotlivými databázemi a vytvořit nové webové rozhraní s použitím dnešních technologií, které bude ladit s designem webu *DB Edu*.

V práci jsou na začátku popsány popsány jednotlivé databázové systémy, jejich verze a vnitřní struktura. Zaměřujeme se pouze na použité systémy jako *Microsoft SQL Server*, *MySQL*, *PostgreSQL*, *Oracle* a *IBM DB2*. Dále je popsaná úprava datové vrstvy a procedur, pro nové systémy. Poté je popsán původním informačním systémem, jeho datová vrstva a webová administrace.

Další částí je nový informační systém, fungování datové vrstvy, nové webové rozhraní a jeho součásti. Nakonec je uvedeno nasazení a spouštění systému.

2 Databázové systémy

V této části se podíváme na jednotlivé databázové systémy a způsoby připojení. Na katedře informatiky aktuálně funguje pět databázových systémů, které je potřeba spravovat. Ke všem systémům se připojuje pomocí technologie *ADO.NET* [10].

2.1 Microsoft SQL Server

Microsoft SQL Server [11][14] je sada technologií a nástrojů umožňujících vytvářet inteligentní klíčové aplikace. Jedná se o škálovatelnou hybridní databázovou platformu, která obsahuje technologie pro zpracování a ukládaní dat, pokročilé zabezpečení nebo analytické funkce. Podporuje rozšířitelnost na několik systémových platforem, například Windows a Linux, a také na ostatní technologie jako jsou kontejnery Docker nebo cloudová úložiště. Server nabízí velkou škálovatelnost, dostupnost a výkon. Uložená data jsou velmi dobře chráněna [15] a rychle přístupná. Verze 2017, oproti starším verzím, podporuje například práci s JSON formátem, Graph data nebo dočasné tabulky. Pro správu se využívá program SQL Server Data Tools [9].

Obrázek 1: Logo Microsoft SQL Server 2017

```
1 Data Source={host},{port};Initial Catalog={database};User ID={login};Password={password};
2 Pooling=False;Min Pool Size=100;Max Pool Size=100000;MultipleActiveResultSets=True
```

Kód 1: Připojovací řetězec SQL Serveru

2.1.1 Edice

Microsoft SQL Server je rozdělen na několik edicí, podle potřebných funkcí a použití. Pro vývoj byla použita *Enterprise edice Microsoft SQL 2017 Serveru verze 14.0.2002.14*. [12]

- **Enterprise** - Tato edice je určena pro nasazení ve velkém produkčním prostředí. Nabízí škálování, vylepšené ukládání do paměti, lepší bezpečnost, vysokou dostupnost a rychlost. Je cílena na velké firmy s velkými objemy dat. Nemá omezení na počet jader procesoru ani na velikost paměti. [13]
- **Standart** - Určena pro středně velké produkční prostředí. Nabízí škálovaní, vylepšené ukládání do paměti, základní analýzu a hlášení. Je cílena pro normální firmy s běžnými

objemy dat. Má omezení na 24 jader procesoru a až 128 GB operační paměti. Je možné ji změnit na *Enterprise edici* bez nutnosti měnit kód. [13]

- **Express** - Edice pro malé produkční prostředí. Je dostupná zdarma pro objem dat až 10 GB. Obsahuje nástroje pro vývoj a správu včetně možnosti záloh a obnovení do cloudu Microsoft Azure. [12] [13]
- **Developer** - Určena pro vývojové prostředí. Je dostupná zdarma pro vytváření a testování aplikací. Obsahuje všechny funkce *Enterprise edice*. [13]

2.1.2 Členění databáze

Každá instance databáze obsahuje seznam uživatelských účtů, které se mohou připojit pomocí loginu a hesla. Každý z účtů může obsahovat několik databází, ke kterým může přistupovat. Databáze dále obsahuje tabulky, zobrazení, triggery, funkce, procedury, diagramy a další součásti. Přístup je limitován pomocí oprávnění účtu.

2.2 MySQL

MySQL [22] patří mezi nejznámější open source databázové servery. Je vlastněná společností Oracle. Tyto databáze lze najít na většině webových serveru, díky její lehké instalaci, výkonu a snadné administraci. Databáze je schopná pracovat jak s SQL tak také s NoSQL a ukládat dokumenty. Je dostupná na více jak 20 platformách včetně Windows, Linuxu, Unixu a Macu. Pro správu se využívá program MySQL Workbench [21].

Obrázek 2: Logo MySQL

```
server={host};port={port};database={database};user id={login};password={password};
allowuservariables=True;minpoolsize=10;maxpoolsize=1000;pooling=False
```

Kód 2: Připojovací řetězec MySQL serveru

2.2.1 Edice

MySQL obsahuje několik edicí, rozdělených podle ceny, podpory, funkcí či systémů. Pro vývoj byla použita *Community edice MySQL serveru verze 8.0.12*. [22]

- **Community** - Tato edice je zdarma a obsahuje veškeré funkce pro práci s databází. Mezi tyto funkce patří spouštění SQL a NoSQL skriptů, ukládání dokumentu, transakce, replikace, routování, partitioning, procedury, triggery, zobrazení, výkonnostní a informační schémata, konektory a program MySQL Workbench. [21]
- **Standart** - *Standardní edice* vychází aktuálně na 2 000 USD ročně, a obsahuje databázový server, podporu ze strany společnosti Oracle, konektory pro připojení, replikaci, program MySQL Workbench a certifikaci na některé ze systémů Oracle. [22]
- **Enterprise** - Tato edice obsahuje vše co *Standardní edice* a vychází aktuálně na 5 000 USD ročně. Obsahuje navíc ukládání dokumentů, routování, partitioning a nástroje pro monitorování, zálohování, zabezpečení a další. Je také certifikován na veškeré systémy Oracle. [22]
- **Cluster CGE** - Jedná se o *Enterprise edici* s rozšířeními pro správu clusteru a Geo-Replikaci. Vychází aktuálně na 10 000 USD ročně. [22]
- **Classic** - Určena pro embedded aplikace, kde je vyžadováno časté čtení. Jedná se o velice výkonnou verzi, s minimální administrací. Dá se lehce rozšířit na vyšší edice. [20]

2.2.2 Členění databáze

Každá instance databáze obsahuje seznam uživatelských účtů, které se mohou připojit pomocí loginu a hesla. Každý z účtů může obsahovat několik schémat, ke kterým může přistupovat. Schéma dále obsahuje tabulky, zobrazení, funkce a procedury. Přístup je limitován pomocí oprávnění účtu.

2.3 PostgreSQL

PostgreSQL [24] je další open source databázový server vyvíjený skupinou PostgreSQL Global Development Group. Obsahuje nejvíce implementovaných SQL vlastností, 160 ze 179. Je vhodný pro ukládání a zpracování komplikovaných dat. Databáze je plně zdarma a neobsahuje žádné dílčí edice, liší se pouze sestavením pro spouštěný systém. Podporuje hned několik systémů jako FreeBSD, OpenBSD, Linux, MacOS, Solaris a Windows. Pro správu se používá program pgAdmin3 LTS. Pro vývoj byl použit *PostgreSQL server verze 10.5*. [25]

Obrázek 3: Logo PostgreSQL

```
Host={host};Port={port};Database={database};Username={login};Password={password};Minimum Pool
↪  Size=10;Maximum Pool Size=1000;Pooling=False
```

Kód 3: Připojovací řetězec PostgreSQL serveru

2.3.1 Členění databáze

Každá instance databáze obsahuje seznam rolí pro přihlášení, které se mohou připojit pomocí loginu a hesla. Každá z rolí může přistupovat k několika databázím. Databáze obsahuje katalogy, rozšíření a schémata. Každé ze schémat poté obsahuje funkce, sekvence, tabulky, zobrazení a triggery. Přístup je limitován pomocí oprávnění.

2.4 IBM DB2

Databázové systémy IBM DB2 [2], vyvíjené společností IBM, se zaměřují na efektivitu, jednoduchost a spolehlivost. Systém je vhodný pro komplexní systémy. Poslední verze se zaměřili na zlepšení zabezpečení, spojení, zjednodušení instalace a zlepšení procesu aktualizace. Pro správu systému se používá program Data Studio Client. [3]

Obrázek 4: Logo IBM DB2

```
Database={database};User ID={login};Password={password};Server={host}:{port};
Max Pool Size=1000;Min Pool Size=10;Pooling=False
```

Kód 4: Připojovací řetězec IBM DB2

2.4.1 Edice

Databáze IBM DB2 obsahuje několik edic s různými omezeními a vlastnostmi. Pro vývoj, byla použita verze *IBM DB2 Express-C 11.1.3.3 (Community)*. [1]

- **DB2 Workgroup Server Edition** - Tato edice neobsahuje řazení tablek do sloupců, funkce DB2 Connect, Q-replikace s dalšími servery, rozdělení na podoblasti a adaptivní kompresi. [1]

- **DB2 Enterprise Server Edition** - Je velice podobná edici *DB2 Workgroup Server Edition*. [1]
- **DB2 Advanced Workgroup Server Edition** - Tato edice obsahuje vše až na součást *Data Server Manager Base*. [1]
- **DB2 Advanced Enterprise Server Edition** - Je velice podobná edici *DB2 Advanced Workgroup Server Edition*. [1]
- **DB2 Direct Advanced Edition** - Je velice podobná edici *DB2 Advanced Workgroup Server Edition*, ale neobsahuje funkce pro zálohování a obnovu. [1]
- **DB2 Direct Standard Edition** - Je velice podobná edici *DB2 Workgroup Server Edition*, ale neobsahuje funkce pro zálohování a obnovu. [1]
- **DB2 Express C Edition** - Je velice podobná edici *DB2 Advanced Enterprise Server Edition*. Je určena pro vývoj aplikací a neprodukční prostředí. [1]
- **DB2 Developer Edition** - Je velice podobná edici *DB2 Advanced Enterprise Server Edition*. Je určena pro vývoj aplikací a neprodukční prostředí. [1]

2.4.2 Členění databáze

Instance databáze obsahuje seznam uživatelů, skupin a rolí. Uživatel se přihlásí pomocí loginu a hesla. Dále obsahuje seznam schémat, ke kterým se uživatelé pomocí oprávnění připojí. Každému schématu jsou globálně přiřazené tabulky, zobrazení, indexy, funkce, procedury a další součásti.

2.5 Oracle Database

Databázový systém Oracle patří mezi nejrozsáhlejší z výše zmíněných. Zaměřuje se na zpracování a ukládaní dat, zabezpečení a rozšiřitelnost. Je optimalizována pro spouštění na systémech Oracle.

Obrázek 5: Logo Oracle Database 18c

```
User Id={login};Password={password}; Data Source=(DESCRIPTION=
(ADDRESS=(PROTOCOL=TCP)(HOST={host})(PORT={port}))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME={ServiceName})));
DBA Privilege=SYSDBA;Min Pool Size=10; Connection Lifetime = 120;
Connection Timeout = 60; Incr Pool Size = 5; Decr Pool Size = 2;
Max Pool Size = 1000; Pooling=False;
```

Kód 5: Připojovací řetězec databáze Oracle

2.5.1 Edice

Databáze Oracle obsahuje několik edic které se liší svým technickým použitím. Pro vývoj, byla použita verze *Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 - Production*. [23]

- **Oracle Database** - Klasická databáze pro produkční použití.
- **Oracle Database In-Memory** - Obohacení klasické databáze o data uložena přímo v paměti. Umístění uložení dat, lze určit pomocí dotazu SQL nebo automaticky systémem. [19]
- **Oracle Database Express** - Databáze určená pro tvorbu a testování aplikací. Je vhodná pro vývojové prostředí.

2.5.2 Členění databáze

Databáze obsahuje seznam uživatelů. Každý uživatel má svoje tabulky, procedury, funkce, zobrazení a další součásti. Přistup do systému je omezen oprávněními uživatele.

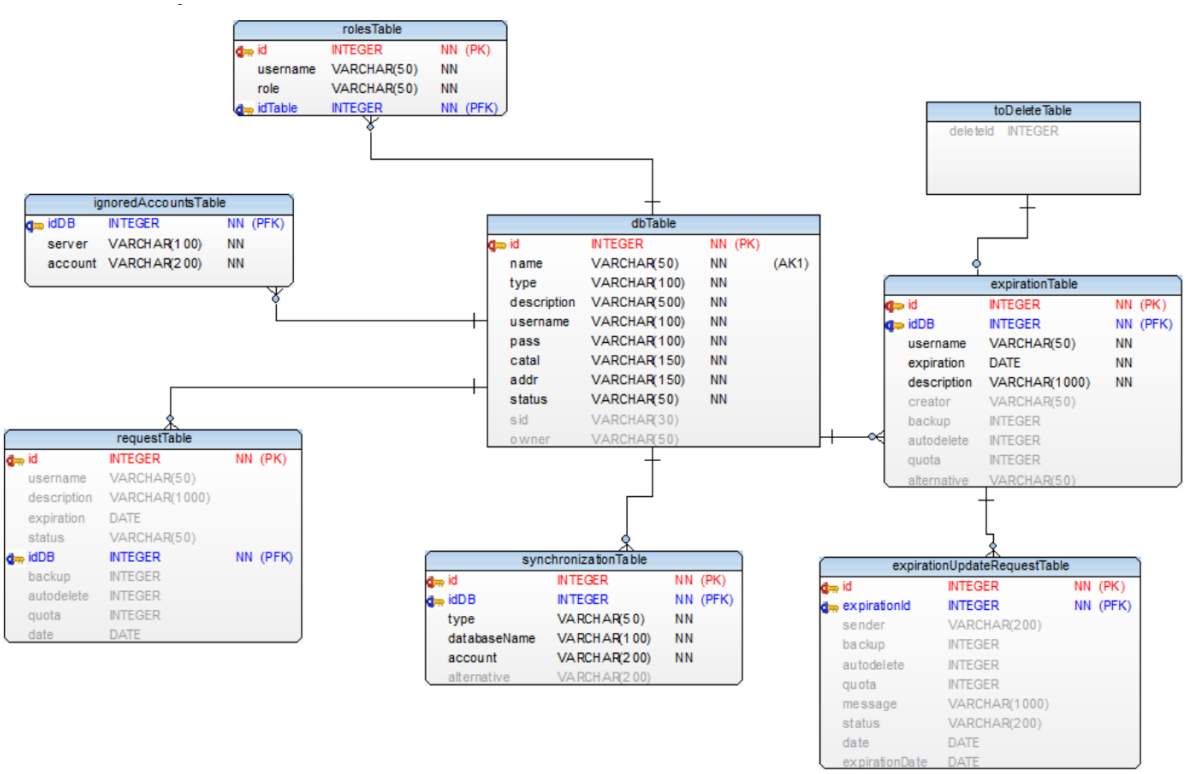
3 Datová vrstva

Datová vrstva se stará o práci s databázemi, spouštění procedur a předávání dat do vyšších vrstev.

3.1 Původní

3.1.1 Databáze

Původní databáze byla vytvořena pro databázový systém SQLite a umístěna lokálně na disku ve složce projektu. Její schéma je níže na obrázku 6. Finální verze byla navržena jako nadstavba nad původní databází z bakalářské práce [4]. Databáze obsahovala 8 tabulek, které jsou uvedeny v tabulce 1. [5]



Obrázek 6: Původní schéma databáze

Název	Účel
dbTable	informace o spravovaných systémech
expirationTable	databázové účty náležící systémům
requestTable	žádosti o vytvoření databázového účtu
expirationUpdateRequestTable	žádosti o změnu parametru databázového učtu
toDeleteTable	databázové účty připravené ke smazání
synchronizationTable	konflikty databázových účtů mezi spravovaným systémem a interní databází
ignoredAccountsTable	ignorované konfliktní databázové účty
rolesTable	tabulka uživatelů a rolí v systému

Tabulka 1: Seznam tabulek původní databáze

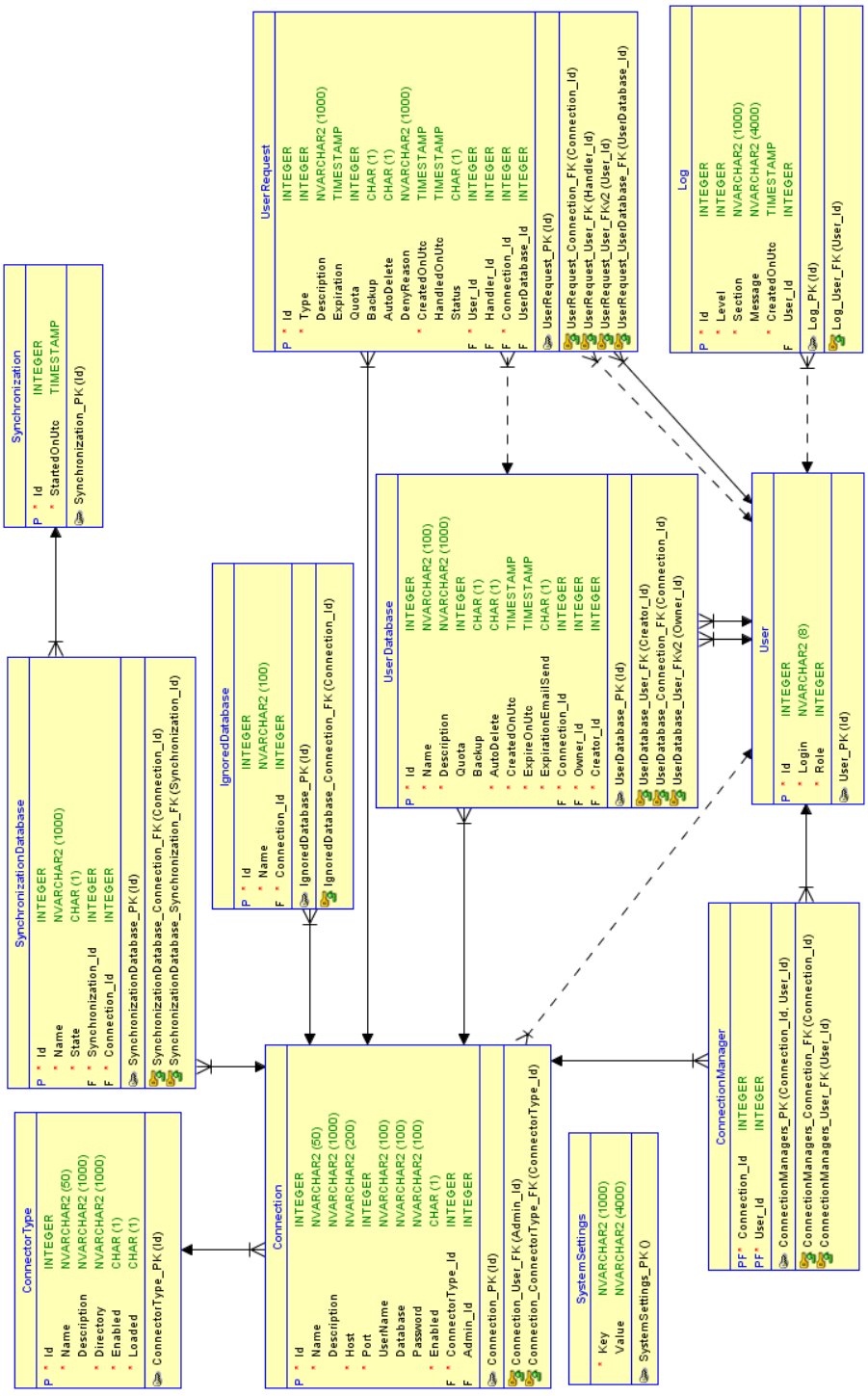
3.1.2 Připojování a spouštění úloh

V původním systému byly vytvořeny třídy v projektu *Connections*, které implementovali rozhraní *IConnections*. Toto rozhraní můžete vidět v kódu 6 v přílohách, na straně 55. Implementace obsahovaly vytvořené připojení a veškeré příkazy, které se spouštěly na daném spojení. Z metod se vracely již zkonvertované data. Projekt *Connections* se načetl při spuštění aplikace. Poté se vytvořila instance spojení, a spustila daná metoda, jak je ukázáno v kódu 7 na straně 56.

3.2 Nová

3.2.1 Databáze

Nová databáze je založena na původní databázi, ale upravena pro potřeby Microsoft SQL Serveru. Důvodem pro tuto úpravu byl přesun původní databáze na databázový systém SQL Server. Při úpravě byly sjednoceny tabulky pro správu požadavků, odstraněna tabulka *toDeleteTable*, rozdělena tabulka *synchronizationTable* a upraveny některé vazby. Databáze je vytvořena pomocí frameworku Entity Framework Core [17], který dokáže vytvářet migrace z vytvořených tříd v jazyce C# a ty následně převést na SQL příkazy pro databázi. Dále je framework schopen vkládání, úprav, mazání, hledání, opožděného načítání entit a kontroly stavu databáze vůči novým verzím. Relační model je na obrázku 7 níže.



Obrázek 7: Relační model databáze

3.2.2 Tabulky pro správu spojení

Ukládají informace o jednotlivých databázových konektorech a spojeních.

ConnectorType [IConnectorType]				
Účel	Definice typu konektoru (načtený ze souboru) v databázi			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Name	String (50)	Ano		Zobrazovaný název
Description	String	Ne	Null	Zobrazovaný popis
Type	String	Ano		Umístění složky konektoru
Enabled	Bool	Ano	False	Povolení použití konektoru
Loaded	Bool	Ano		Stav načtení konektoru

Tabulka 2: Tabulka ConnectorType

Connection [IConnectorFactory]				
Účel	Definice spojení pro konektor			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Name	String (50)	Ano		Zobrazovaný název
Description	String	Ne	Null	Zobrazovaný popis
UserName	String(100)	Ne	Null	Přihlašovací jméno
Password	String(100)	Ne	Null	Přihlašovací heslo
Database	String(100)	Ne	Null	Název databáze
Host	String(200)	Ano		Url databáze
Port	INT	Ano		Port připojení
Enabled	Bool	Ano	False	Povolení použití konektoru
AdminId	INT, FK	Ne	Null	Správce databáze
ConnectorTypeId	INT, FK	Ano		Typ konektoru

Tabulka 3: Tabulka Connection

3.2.3 Tabulka nastavení systému

Ukládá model nastavení. Každá *property* třídy nastavení je vedena jako záznam v tabulce 4, kde její název je *Key* a její hodnota *Value*.

SystemSettings				
Účel	Nastavení systému (Klíč hodnota, Serializovatelné)			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Key	String, PK	Ano	Auto Increment	Klíč hodnoty
Value	String	Ne	Null	Hodnota

Tabulka 4: Tabulka SystemSettings

3.2.4 Uživatelské tabulky

Ukládají informace o uživatelích, jejich účtech a požadavcích.

User				
Účel	Uživatelský účet v systému			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Login	String (50)	Ano		Login účtu
Role	Enum	Ano	Student	Role účtu

Tabulka 5: Tabulka User

UserDatabase				
Účel	Nastavení databáze uživatele			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Name	String (100)	Ano		Zobrazovaný název
Description	String	Ano		Zobrazovaný popis
Quota	INT	Ne	Null	Nastavená kvóta
Backup	Bool	Ne	Null	Zálohování
AutoDelete	Bool	Ano	True	Automatické smazání
CreatedOnUtc	DateTime	Ano	GetDate()	Datum vytvoření účtu
ExpireOnUtc	DateTime	Ano		Datum expirace účtu
ExpirationEmailSend	Bool	Ano	False	Informace o odeslání emailu
CreatorId	INT, FK	Ano	Null	Uživatel, který účet vytvořil
ConnectionId	INT, FK	Ano		Databáze, které náleží
OwnerId	INT, FK	Ano		Uživatel, kterému náleží

Tabulka 6: Tabulka UserDatabase

UserRequest				
Účel	Požadavek, o vytvoření/změnu/smazání databáze			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Name	String (100)	Ano		Zobrazovaný název
Description	String	Ano		Zobrazovaný popis
Quota	INT	Ne	Null	Nastavená kvóta
Backup	Bool	Ne	Null	Zálohování
AutoDelete	Bool	Ano	True	Automatické smazání
CreatedOnUtc	DateTime	Ano	GetDate()	Datum vytvoření účtu
HandledOnUtc	DateTime	Ne	Null	Datum schválení/zamítnutí požadavku
Expiration	DateTime	Ano		Datum expirace účtu
DenyReason	String	Ne	Null	Důvod zamítnutí
HandlerId	INT, FK	Ne	Null	Uživatel, který požadavek vyřizoval
UserDatabaseId	INT, FK	Ne		Účet databáze, kterého se požadavek týká
ConnectionId	INT, FK	Ano		Databáze, které náleží
UserId	INT, FK	Ano		Uživatel, kterému náleží
Status	Bool	Ne	Null	Stav požadavku

Tabulka 7: Tabulka UserRequest

3.2.5 Tabulky pro synchronizaci

Obsahují informace o databázích, které se neschodují se systémem, pro dané databázové spojení. Administrátor, ve webovém rozhraní, rozhoduje o jejich vyřešení.

IgnoredDatabase				
Účel	Ignorované databáze při synchronizaci			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Name	String (100)	Ano		Název databáze
ConnectionId	INT, FK	Ano		Spojení na databázi

Tabulka 8: Tabulka IgnoredDatabase

Synchronization				
Účel	Čas spuštění globální synchronizace			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
StartedOnUtc	DateTime	Ano	GetDate()	Datum spuštění synchronizace

Tabulka 9: Tabulka Synchronization

SynchronizationDatabase				
Účel	Databáze jež se nachází v konfliktu			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Name	String (1000)	Ano		Název databáze
State	Enum	Ano		Stav synchronizace
ConnectionId	INT, FK	Ano		Spojení na databázi
SynchronizationId	INT, FK	Ano		Synchronizace

Tabulka 10: Tabulka SynchronizationDatabase

3.2.6 Tabulka událostí

Zaznamenává události systému do databáze pro pozdější zobrazení.

Log				
Účel	Zaznamenávání událostí systému			
Klíč	**Typ**	**Vyžadováno**	**Výchozí hodnota**	**Popis**
Id	INT, PK	Ano	Auto Increment	Primární klíč
Level	Enum	Ano		Úroveň záznamu
Section	String (1000)	Ano		Sekce, kde k události došlo
Message	String	Ne	Null	Zpráva o události
CreatedOnUtc	DateTime	Ano	GetDate()	Datum vytvoření
UserId	INT, FK	Ne	Null	Přihlášený uživatel

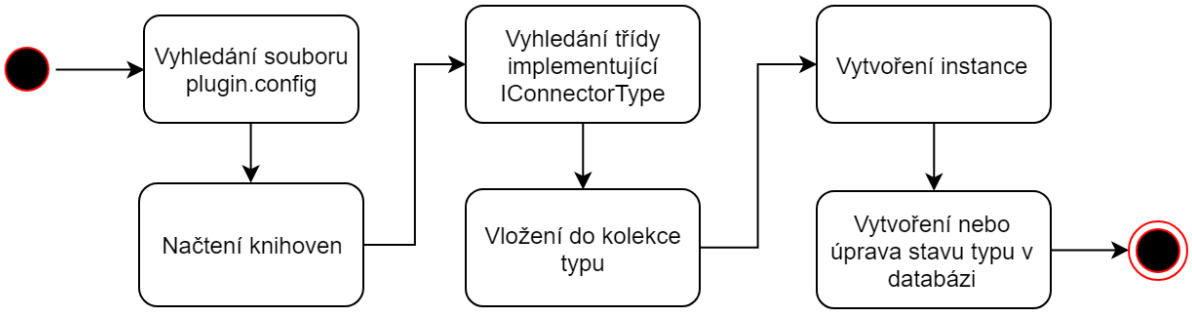
Tabulka 11: Tabulka Log

3.2.7 Připojování a spouštění úloh

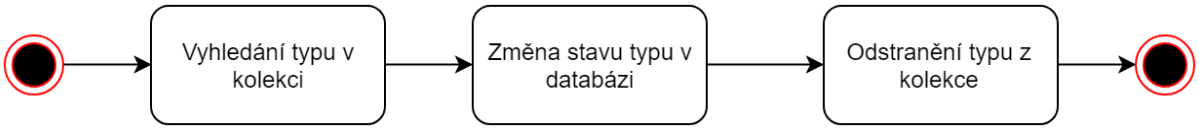
Při spuštění systému nebo na vyžádání, se po přesunutí knihoven do dočasné složky, načtou z dočasné složky konektory, které implementují rozhraní *IConnectorType*. Rozhraní je ukázáno v kódu 8 na straně 56. Načítání je řešeno pomocí balíčku *DotNetCorePlugins* [18], pro který se ve složce vyhledá soubor *plugin.config*. Ten obsahuje informace pro načtení knihovny konektoru. Pokud je konektor úspěšně načten, je zaregistrován do systému a je možné jej použít pro databázi. Knihovny jsou přesunuty do dočasné složky, aby bylo možné bezpečně přepisovat původní umístěni, jelikož nelze kontrolovat zamykání souborů načtených knihoven. Princip načítání na obrázku 8 níže.

Konektor lze také odebrat, při této operaci dojde k jeho odstranění z kolekcí a změnění jeho stavu v databázi, jakmile dojde k odebrání veškerých referencí, dojde k uvolnění knihovny. Princip odebrání konektoru na obrázku 9 níže.

Opětovné načtení je řešeno odebráním a opětovným načtením konektoru.

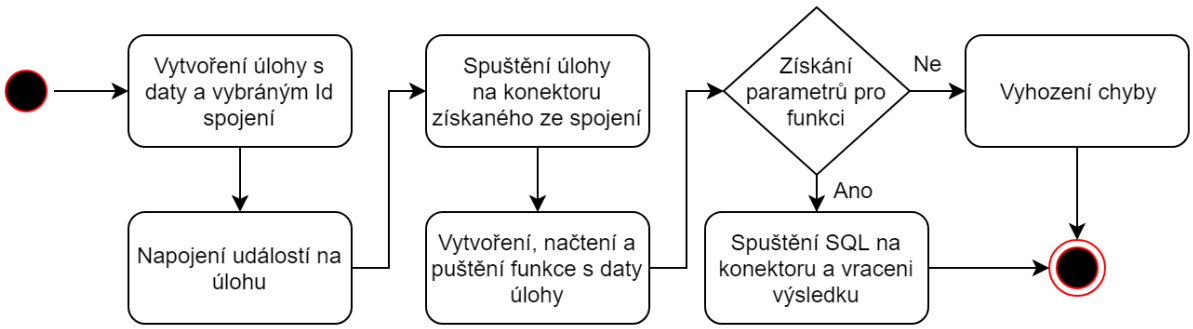


Obrázek 8: Princip načítání konektoru



Obrázek 9: Princip odebrání konektoru

Pro vytvoření úlohy stačí připravit data, vybrat funkci (třída implementující *IFunction*, kód 14 na straně 60), která se má spustit. Následně pomocí zvoleného spojení, lze spustit úlohu. Daná funkce si automaticky načte správný příkaz z původní složky konektoru. Následně se spustí na databázi, data se ve třídě zkonvertují a vrátí se výsledek operace. Použití je zobrazeno v kódu 9 na straně 57 a schéma níže na obrázku 10.



Obrázek 10: Princip spuštění úlohy na databázi

3.3 Původní funkce

Seznam původních SQL funkcí a procedur je možné vidět v tabulce 12.

Typ	Název	Popis
Procedura	CreateLogin	Vytvoření účtu, katalogu a nastavení kvóty
Procedura	DeleteLogin	Smazání katalogu a uživatelského účtu
Procedura	SetQuota	Změná kvóty
SQL dotaz	GetQuota	Získání aktuální kvóty
SQL dotaz	GrantConnect	Povolení připojení k databázi
SQL dotaz	DenyConnect	Zakázání připojení k databázi
SQL dotaz	GetLogins	Získání uživatelských účtů
SQL dotaz	GetLoginsPermissions	Získání povolených účtů
SQL dotaz	CheckLogin	Ověření existence účtu
SQL dotaz	CheckPermission	Ověření povolení připojení účtu
SQL dotaz	CheckLock	Ověření uzamčení účtu
SQL dotaz	UnlockAccount	Odemčení účtu
SQL dotaz	PasswordReset	Změna hesla účtu
SQL dotaz	CreateProcedure	Vytvoření procedury
SQL dotaz	CheckProcedures	Ověření existence procedur

Tabulka 12: Seznam původních používaných funkcí

3.4 Úpravy

Většina funkcí byla upravena na nový systém, pomocí úpravy a sjednocení parametrů. Některé přebytečné funkce nebyly použity. Byla vytvořena funkce pro synchronizaci, která vrací list všech uživatelů, databází a stav oprávnění pro připojení. Dále byly vytvořeny jednoduché funkce pro vytvoření databáze. Vytváření procedur je řešeno v samotném kódu, ve třídě *BaseFunction* (kód 10), jako součást základní implementace rozhraní *IFunction* (kód 14), která spustí přiložený kód SQL na databázi, pokud procedura neexistuje. Veškeré funkce jsou uloženy ve složce *Functions* daného konektoru. Funkce musí mít přesné pojmenování, jinak se nenačtou a nespustí. Seznam aktivních funkcí a jejich parametry lze najít v souboru *Functions/Functions.json* (kód 12) ve složce konektoru. Tento soubor definuje jaké parametry se do dané funkce pošlou a zda je povolena. Pokud je funkce zakázaná nebo není uvedena, nebude povolena a nebude možné pro ni získat parametry, tento stav je indikován chybou *NotAllowedException* (kód 13). Příklad funkce v kódu 11 na straně 59. Seznam změn ve funkcích a procedurách je v tabulce 13.

Typ	**Název**	**Změny**	**Popis**
Procedura	CreateLogin	Upraveny parametry	Vytvoření účtu, katalogu a nastavení kvóty
Procedura	DeleteLogin	Upraveny parametry	Smazání katalogu a uživatelského účtu
Procedura	SetQuota	Upraveny parametry	Změna kvóty
SQL dotaz	GrantConnect	Upraveny parametry	Povolení připojení k databázi
SQL dotaz	DenyConnect	Upraveny parametry	Zakázání připojení k databázi
SQL dotaz	PasswordReset	Upraveny parametry	Změna hesla účtu
SQL dotaz	CheckProcedures	Upraveny parametry	Ověření existence procedur
SQL dotaz	CreateProcedure	Implementováno v kódu	Vytvoření procedury
SQL dotaz	GetSyncData	Spojení několika funkcí	Vrátí seznam uživatelů a katalogů z databáze
SQL dotaz	Backup	Nová funkce	Spustí zálohu databáze
SQL dotaz	CreateDatabase	Nová funkce	Vytvoření katalogu
SQL dotaz	DropDatabase	Nová funkce	Smazání katalogu
SQL dotaz	GetQuota	Není použita	Získání aktuální kvóty
SQL dotaz	GetLogins	Není použita	Získání uživatelských účtů
SQL dotaz	GetLoginsPermissions	Není použita	Získání povolených účtů
SQL dotaz	CheckLogin	Není použita	Ověření existence účtu
SQL dotaz	CheckPermission	Není použita	Ověření povolení připojení účtu
SQL dotaz	CheckLock	Není použita	Ověření uzamčení účtu
SQL dotaz	UnlockAccount	Není použita	Odemčení účtu

Tabulka 13: Seznam změn ve funkcích

4 Prezentační a Servisní vrstva

Prezentační vrstva se stará o zobrazení dat uživateli, a jejich získávání od uživatele. Tuto roli zastupuje webové rozhraní informačního systému. Servisní vrstva (Business layer) se stará o zpracování dat, komunikaci s datovou vrstvou a předávaní dat prezentační vrstvě. [6]

4.1 Původní systém

Původní systém byl vytvořen pomocí technologie *ASP.NET Web Forms* [8]. Tato technologie umožňovala vytvářet dynamické weby pomocí komponent, které se vkládaly do webu pomocí návrháře, obsaženého v programu Visual Studio [16]. Každá stránka také obsahovala vlastní třídu, která se starala o předávání dat a zpracování událostí. Web byl rozdělen do 17 stránek. Nevýhodou těchto webů byla větší velikost stánky, díky řídícím prvkům v HTML. V dnešní době je tato technologie již zastaralá a nahrazena technologií *Razor pages*.

4.2 Členění řešení nového systému

Řešení je rozděleno do 17 projektů, jejich seznam je v tabulce 14. Projekty končící na Core, obsahují definice rozhraní, bázové implementace rozhraní a pomocné typy a třídy. Ostatní projekty jsou jejich implementací. Každý konektor má také svoji implementaci pro specifickou databázi. Výstup z projektu konektoru je po sestavení umístěn do složky projektu webu, odkud je poté načten.

Název	**Popis**
DatabaseManagerContext	Obsahuje implementace funkcí, databázový kontext pro EF Core a definice databázových entit
DatabaseCore	Obsahuje vše potřebné pro tvorbu konektoru
DataMapper	Zařizuje mapování mezi SQL a objekty
DataStore	Obsahuje implementaci cache pro datovou vrstvu
DataStoreCore	Rozhraní cache datové vrstvy
MainDatabaseCore	Rozhraní a základní implementace pro databázový kontext
Utilities	Obsahuje rozšíření, například zamykání nebo konverze textu
DB2Sql	Implementace konektoru pro IBM DB2
MsSql	Implementace konektoru pro Microsoft SQL Server
MySql	Implementace konektoru pro MySQL
OracleSql	Implementace konektoru pro databázi Oracle
PostgreSql	Implementace konektoru pro PostgreSQL
ConnectorManager	Implementace správy úloh a konektorů
ManagerCore	Rozhraní pro správce konektorů
Services	Implementace služeb webu
ServicesCore	Základní třídy a služby pro web
DBMan	Samotná webová aplikace

Tabulka 14: Seznam projektů a jejich účel

4.3 Nová Prezentační vrstva

Nový systém je založen na technologii *ASP.NET Core MVC* [7]. Tato technologie a návrhový vzor zajišťuje rozdělení webu na zobrazení, model a kontrolér. Kontrolér zajišťuje předání a konvertování dat do modelu, obsahuje akce, kde každá z nich řeší jednu úlohu a vrací výsledek v podobě odpovědi. Převod adresy na je řešen pomocí rout, které směřují na nalezenou akci v kontroléru. Model obsahuje sjednocená data, jejich pravidla a detaily. Zobrazení přijímá model a vytváří pomocí něj HTML stránku, detaily v modelu jsou také použity pro vytváření formulářů a dalších částí stránky. Zobrazení jsou psaná v jazyku Razor, který kombinuje syntaxe C#, HTML, CSS a JS. Zobrazení se také dělí na plná, která obsahují layout stránky, a částečná, která obsahují pouze samotnou stránku.

Seznam kontroléru a jejich účel:

- **AccountController** - uživatelské účty (obrázek 12)
- **ConnectionController** - databáze a nastavení spojení (obrázek 13)
- **ConnectorController** - konektory a jejich nastavení
- **HomeController** - úvodní stránka a přihlašování (obrázek 11)

- **LogController** - výpis událostí systému (obrázky 15 a 16)
- **RequestController** - uživatelské požadavky
- **SettingsController** - nastavení systému
- **SynchronizationController** - výsledky a akce synchronizace systému
- **TaskController** - dlouhodobé úlohy systému (obrázek 14)

4.3.1 Nápověda

Každá stránka s tabulkou obsahuje popis, co je na ní zobrazeno a seznam možností, co na ní lze dělat. Uživatelsky viditelné stránky obsahují také krátký popis fungovaní dané části. Formuláře obsahují informace o daném formuláři a informace o procesu, co nastane po jeho odeslání. Veškerá tlačítka mají svůj titulek.

4.4 Nová servisní vrstva

V původním projektu byla veškerá logika obsažena v kódu stránek, docházelo zde k vytvoření konektoru na databázi, spuštění funkce i konverzi dat. Nově byla struktura rozdělena na několik vrstev, které se starají o svoje části operace. Mezi tyto vrstvy patří datová vrstva, která komunikuje s databází, servisní vrstva, která se stará o přípravu operací a spouštění funkcí na datové vrstvě. Prezentační vrstva se poté stará o zobrazení dat a jejich sběr od uživatele.

Servisní vrstva obsahuje funkce pro spouštění funkcí na databázích, správu paměti, souboru a zabezpečení. Dále obsahuje funkce pro hledání, výpis, vytváření, úpravu a mazání entit z hlavní databáze.

Seznam služeb:

- **CacheManager** - ukládání dat do paměti programu
- **TemporaryDirectoryService** - správa dočasných souborů a složek
- **FileService** - načítání a ukládání souborů
- **ConnectorTypeService** - správa konektoru
- **ConnectionService** - správa databází
- **ConnectorManagerService** - spouštění úloh a registrace konektoru
- **TaskService** - tvorba a spouštění komplexních úloh
- **EncryptionService** - šifrování a dešifrování textu
- **PasswordGeneratorService** - generování hesel pro databáze

- **LogService** - zaznamenání událostí
- **LogServiceProvider** - vytváří LogService pro systémové účely
- **MailService** - odesílání emailu
- **UserDatabaseService** - správa uživatelských databází v hlavní databázi
- **UserRequestService** - správa požadavků
- **LdapService** - ověření uživatele pomocí serveru LDAP a získání informací o uživateli
- **UserService** - správa uživatelů a přihlašování
- **WorkContext** - aktuální hodnoty pro požadavek
- **SystemSettingService** - načítání a ukládání systémových nastavení

4.4.1 Notifikace emailem

Pro oznámeni změn v systému uživatelům, jsou použity emailové zprávy. Byl použit původní emailový klient, který byl upraven pro použití vzoru zpráv ze souboru. Vzory zpráv se přesunuty do souboru *Emails.json*, ze kterého jsou načteny. Email je vytvořen vybráním vzoru a dodáním parametrů pro jeho vyplnění. Pomocí třídy *DbEduTemplate* je vytvořena výsledná zpráva. Následně se zpráva odešle vybraným uživatelům. Třída *DbEduTemplate* se upravila pro použití jako singleton, který je definován v nastavení služby.

4.4.2 Synchronizace

Jelikož databázové systémy je možné spravovat i manuálně, mohou nastat případy, kdy se stav serveru nebude shodovat se záznamy v systému. Pro tyto účely je využita synchronizace, která probíhá jednou denně ve vláknu na pozadí webu, a ukládá změny mezi databázemi a záznamy do tabulky 10. Pro získání seznamu účtů a databází ze spravovaných systémů, byla vytvořena SQL funkce, která vrací tabulku ve formátu *Login*, *Database*, *Connect*. *Login* udává uživatelský účet, *Database* je databáze, schéma nebo katalog patřící účtu a *Connect* udává stav oprávnění účtu. Výsledek se porovná se záznamy v hlavní databázi. Pokud jsou nalezeny nové změny, je tato skutečnost oznámena správci systému mailem. Ten může pomocí webového rozhraní vyřešit jednotlivé problémy, nebo označit dané účty za ignorované při synchronizací.

4.4.3 Zálohování

Před každou synchronizací je spouštěno zálohovaní databázových účtů, které jej mají povoleno. Vytvoří se úloha pro dané spojení s názvem účtu a funkcí *Backup*, která spustí zálohovací skript na daném databázovém systému pro daný účet. Jednotlivé zálohy jsou uloženy na daném serveru.

4.4.4 Kontrola připojení

Jelikož servery mohou běžet na různých systémech, je třeba ověřit jejich funkčnost. Toto je v pravidelných intervalech zajišťováno vláknem v pozadí webu, který otestuje veškeré databáze a uloží si výsledky do paměti pro další zpracování. Testování probíhá několikrát za sebou s časovým odstupem. Nejprve se otestuje vytvoření spojení a poté spuštění úlohy na daném serveru. Stačí, aby se spojení jednou povedlo, a je celý test prohlášen za úspěšný. Pokud se však žádné spojení nenaváže, je odeslán mail správci serveru, informující o jeho výpadku.

4.4.5 Mazání prošlých účtů

Pokud databázovému účtu vyprší platnost, je ten den nebo nejbližší další, při spuštění synchronizace odeslán mail o vypršení platnosti vlastníkovi účtu. Informace o odeslání emailu, se uloží do hlavní databáze. Pokud si uživatel účet neprodlouží je při příští synchronizaci, s odstupem až několika dní podle nastavení, účet smazán a je odeslán mail vlastníkovi, informující o smazání účtu. Mazání je spuštěno před samotnou synchronizací.

4.5 Konfigurace webu

Konfigurace webu, registrace služeb a nastavení zpracování požadavku je umístěno v souboru *Startup.cs* v projektu webu. V metodě *ConfigureServices* se registrují služby a nastavení. V metodě *Configure* je poté nastaveno zpracování požadavku a spouštění úloh při spuštění, jako načtení konektorů.

Služby lze registrovat třemi způsoby:

- **Singleton** - služba je vytvořena pouze jednou při startu aplikace
- **Scoped** - služba se vytváří pro každý požadavek zvlášť
- **Transistent** - služba je vytvořena pokaždé znova

5 Závěr

Cílem tohoto projektu bylo přenést původní systém správy databází, na novou, modernější verzi. Bylo třeba vytvořit novou databázi pro *Microsoft SQL Server* a upravit webové rozhraní na nový design a technologie. Nakonec bylo nutné vše otestovat a nasadit na server.

Prvním krokem bylo vytvoření nových konektorů a sjednocení práce s databázemi. Tento krok trval déle, než se předpokládalo, ale značně zjednodušil další implementaci. Sjednocení práce umožňuje použití stejné části kódu, pro práci s různými databázovými systémy, jednoduché spouštění a vytváření funkcí. Díky dynamickému načítání je možné celou implementaci některého z konektorů vyměnit či odebrat. Umístěním SQL kódu do souborů je možné změnit jejich obsah bez nutnosti kompilovat kód či jinak zasahovat do systému. Pokud dojde ke změně v souborech funkcí, je nutné vymazat paměť konektoru ve webovém rozhraní.

Dalším krokem bylo vytvoření nové databáze pro *SQL Server*. Databáze byla založena na původní verzi, ale upravena pro potřeby nového systému. Její implementace byla řešena pomocí frameworku *EF Core*, který zjednodušil její vytváření, úpravy i práci. Byly vytvořeny třídy entit v kódu, které se poté převedly na migraci, což je C# kód pro úpravu databáze. Nakonec se migrace spustily a vytvořila se databázová struktura.

Následovalo vytvoření designu, nejprve vznikl jednoduchý HTML web s designem. Tento web se poté odeslal grafikovi, který jej doladil. Jakmile byl design hotový, byl implementován do samotné webové aplikace. Byly vytvořeny ostatní stránky webu a pokračovalo se implementací tlačítek, zobrazení a dialogů. Celý web byl také lokalizovaný do češtiny a angličtiny, pomocí zdrojových souborů. Jazyk se vybíral podle adresy url dané stránky.

Po dokončení zobrazení se vytvořily služby pro práci s databází a funkce potřebné pro rozhraní webu. Jakmile byl web v provozu, dodávali se do něho další potřebné stránky.

Vytvořily se hromadné úlohy a funkce pro jejich vytváření a správu. Tyto úlohy jsou přiřazeny uživateli a jsou uloženy v paměti, odkud nejsou smazány.

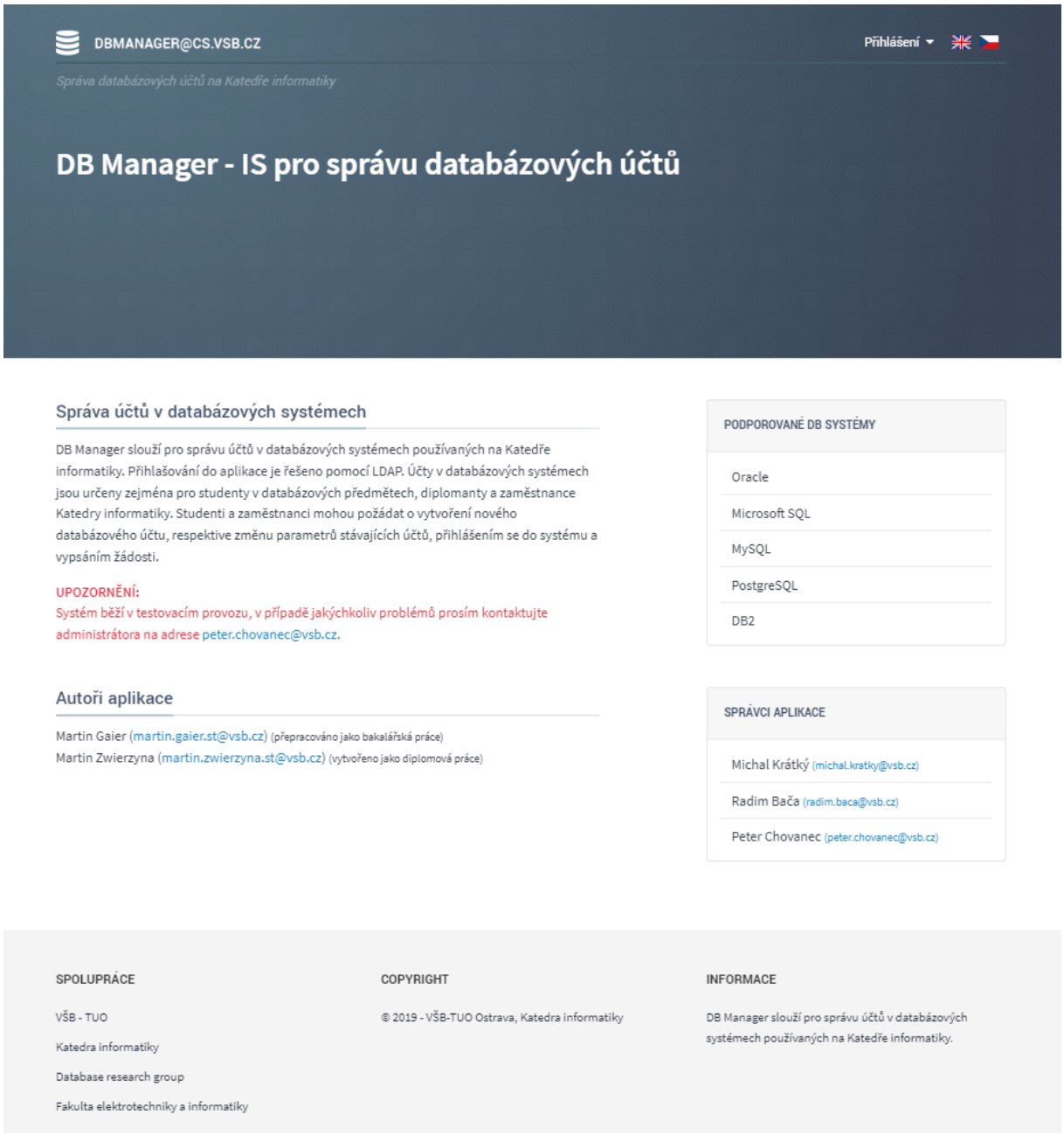
Další části byla implementace emailového klienta, který posílal zprávy o změnách uživatelům. Byl použit původní emailový klient s úpravami. Jednou z hlavních úprav je přesunutí vzoru zpráv do souboru *Emails.json*, ze kterého jsou načteny a pomocí třídy *DbEduTemplate* je vytvořena výsledná zpráva. Třída *DbEduTemplate* byla také upravena pro použití jako singleton.

Nakonec byly přidány úlohy, které se spouští v daných intervalech, jako je synchronizace nebo mazání účtů.

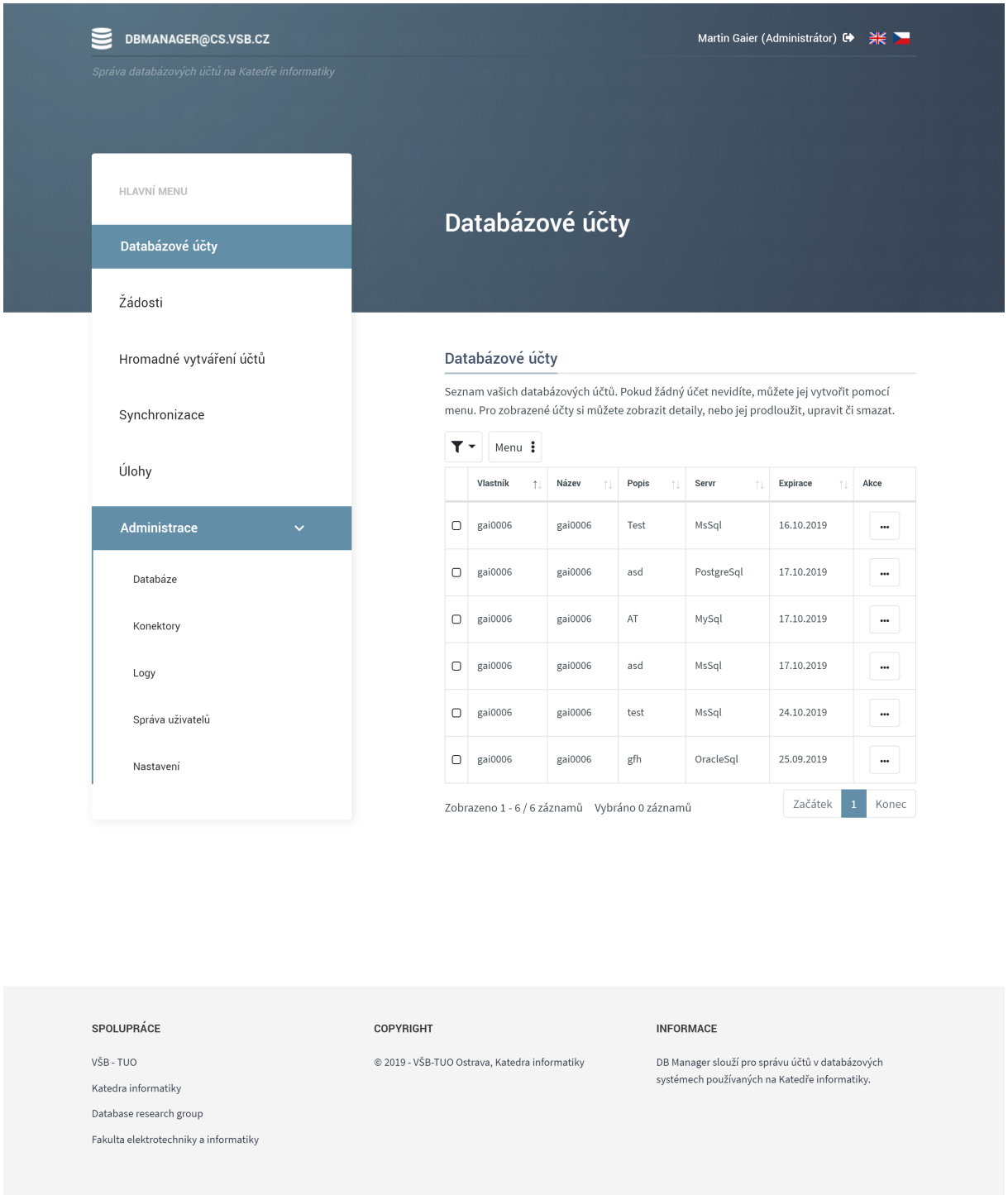
Web se otestoval při nasazení na *server IIS*, kde byly vyřešeny drobné konfigurační problémy, které se objevily.

Díky struktuře webu je možné jej lehce rozšiřovat a upravovat. Lze jednoduše vytvářet funkce a konektory, měnit stávající, upravovat emaily či lokalizace, přidávat entity do databáze nebo vytvářet služby. Většinu implementací, lze jednoduše zaměnit pomocí použitých rozhraní.

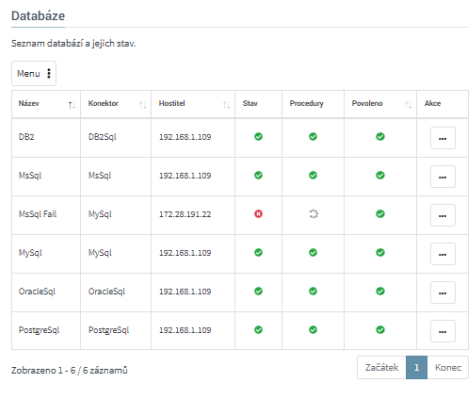
A Obrázky



Obrázek 11: Přihlašovací a úvodní stránka

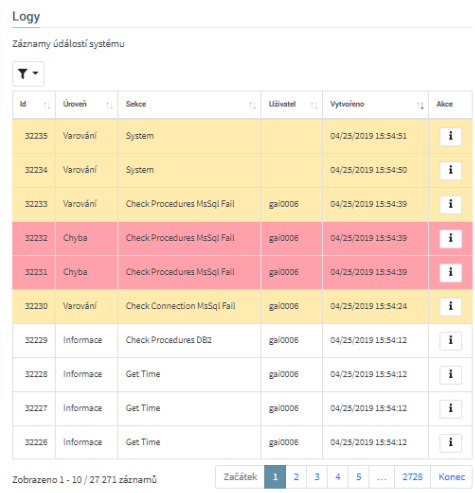


Obrázek 12: Stránka uživatelských účtů

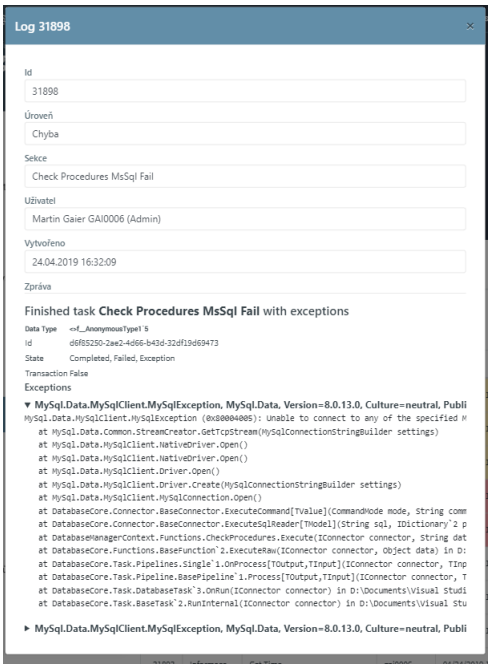


Obrázek 13: Stránka databází

Obrázek 14: Probíhající dlouhodobá úloha



Obrázek 15: Stránka událostí



Obrázek 16: Detail zobrazení události

B Části kódu

```
namespace Connections
{
    public interface IConnections
    {
        bool Connect(bool writeToLog = true);
        bool CreateLogin(String login, String alternative, String pass, int quota);
        bool DeleteLogin(String login, String alternative);
        bool SetQuota(String login, int quota);
        int GetQuota(String login);
        bool GrantConnect(String login);
        bool DenyConnect(String login);
        List<String> GetLogins();
        List<Logins> GetLoginsPermissions(String dbname);
        bool CheckLogin(String login);
        bool CheckPermission(String login);
        bool CheckLock(String login);
        bool UnlockAccount(String login);
        bool PasswordReset(String login, String pass);
        bool CreateProcedure(String sql);
        bool CheckProcedures();
        bool Destroy();
    }
}
```

Kód 6: Rozhraní IConnections původního projektu

```
 List<Request> requestToDelete = new List<Request>();
foreach (Request r in request) {
    conn = (IConnections)Activator.CreateInstance(Type.GetType("Connections." + r.dbtype +
     ↪  ",Connections"), getConnectionStringFromList(r.dbname));
    if (conn.Connect() == true) {
        if (conn.CheckLogin(r.username) == true) {
            mainConn.Connect();
            mainConn.UpdateRequest(true, r.id, Page.User.Identity.Name);
            mainConn.Destroy();
            requestToDelete.Add(r);
        }
        conn.Destroy();
    }
}
```

Kód 7: Použití rozhraní IConnections v původním projektu

```
namespace DatabaseCore.Type
{
    public interface IConnectorType : IDisposable
    {
        string Directory { get; }
        IConnectorType Configure();
        IGeneratorProvider GeneratorProvider { get; }
        IConnectionBuilder ConnectionBuilder { get; }
        IFileCacheManager FileCacheManager { get; }
        IMemoryCacheManager FunctionCacheManager { get; }
        IDictionary<string, IConnectorFactory> Factories { get; }
        FunctionConfigurationsCollection FunctionConfigurations { get; }
        IConnectorFactory GetFactory<TKey>(string connectionString, TKey generatorProviderKey =
         ↪  default(TKey));
        bool IsFunctionAllowed(IFunction function);
        bool IsFunctionAllowed(System.Type function);
        string[] GetFunctionParams(IFunction function);
        string[] GetFunctionParams(System.Type function);
        IConnectorType ClearCaches();
    }
}
```

Kód 8: Rozhraní IConnectorType

```
 // Prepare data
var data = new DatabaseUser() {
    Database = request.User.Login,
    UserName = request.User.Login
};
 // Update Quota
try {
    data.Quota = request.Quota;
    var task = new DatabaseTask<Single<SetQuota>, int, DatabaseUser>(data);
    await _connectionService.RunTask(await _connectionService.BindTask(task, $"Set Quota
     ↪  {request.User.Login}", db.ConnectionId, false));
}
catch (NotAllowedFunctionException ex) {
    // Quota not supported
    _logService.Log(LogLevel.Information, $"Set Quota {request.User.Login}",
        "Setting user account quota not supported on system", $"User {request.User.Login}",
        request, ex);
}
catch (Exception ex) {
    if (request.Type == UserRequestType.Create) {
        request.UserDatabaseId = 0;
        _userDatabaseService.Delete(db);
    }
    request.Status = null;
    _logService.Log(LogLevel.Error, "Handle Request", "Setting user account quota failed",
     ↪  $"Failed user {request.User.Login}", request, ex);
}
```

Kód 9: Spuštění úlohy na daném spojení

```
namespace DatabaseCore.Functions
{
    public abstract class BaseFunction<TOutput, TInput> : Lockable, IFunction<TOutput, TInput>
    {
        protected bool ReloadFile { get; set; }
        protected bool FileByType { get; }
        public string FileName { get; }
        public bool IsFunction { get; }
        public string SqlCommand { get; protected set; }

        public virtual IFunction Load(IConnector connector)
        {
            if (!string.IsNullOrWhiteSpace(FileName))
                if (IsFunction && ReloadFile)
                    CreateFunction(connector);
                else
                    LoadSql(connector);
            return this;
        }

        protected virtual void LoadSql(IConnector connector)
        {
            if (string.IsNullOrWhiteSpace(SqlCommand) || ReloadFile)
                lock (GetLock(nameof(SqlCommand)))
                {
                    SqlCommand = connector.FileCacheManager.Get($"{Path.Combine("Functions",
                    ↪  FileName)}.sql", ReloadFile);
                    lock (GetLock(nameof(ReloadFile)))
                        ReloadFile = false;
                }
        }

        protected virtual void CreateFunction(IConnector connector)
        {
            var sql = connector.FileCacheManager.Get($"{Path.Combine("Functions",
            ↪  FileName)}.sql", ReloadFile);
            if (!string.IsNullOrWhiteSpace(sql))
                connector.ExecuteSql(sql, null);
        }
    }
}
```

Kód 10: Část základní implementace rozhraní IFunction třídou BaseFunction

```cs
namespace DatabaseManagerContext.Functions
{
    /// Creates login, database and sets quota. Works only if neither exists.
    public class CreateUser : BaseFunction<int, DatabaseUser>
    {
        // FileName, FunctionName
        public CreateUser() : base("CreateUser", "CreateUser") {}

        public override int Execute(IConnector connector, DatabaseUser data)
        {
            // Check if allowed
            var attributes = connector.Factory.Type.GetFunctionParams(this);
            if (attributes == null)
                throw new NotAllowedFunctionException(this, connector);

            // Create user
            return connector.ExecuteFunction(SqlCommand,
             ↪  data.ToDictionary(AccessProtection.Private, MapMode.Both, attributes));
        }
    }
}
```

Kód 11: Funkce pro vytváření uživatele

```json
{
  "Functions": [
    {
      "Allowed": true,
      "Name": "CreateUser",
      "Attributes": [
        "UserName",
        "Password",
        "Quota",
        "Database"
      ]
    }
  ]
}
```

Kód 12: Nastavení funkcí konektoru

```
namespace DatabaseCore.Functions
{
    public class NotAllowedFunctionException : Exception
    {
        public NotAllowedFunctionException(){}
        public NotAllowedFunctionException(string message) : base(message){}
        public NotAllowedFunctionException(string message, Exception innerException) :
        ↪  base(message, innerException){}
        protected NotAllowedFunctionException(SerializationInfo info, StreamingContext context)
        ↪  : base(info, context){}
        public NotAllowedFunctionException(IFunction function, IConnector connector, string
        ↪  message = "Function '{0}' is not allowed on connector '{1}'!") :
           this(string.Format(message, function.GetType().Name, connector.GetType().Name)){}
    }
}
```

Kód 13: Definice chyby NotAllowedException, která je vyhozena pokud funkce není povolena

```
namespace DatabaseCore.Functions
{
    public interface IFunction {
        string FileName { get; }
        bool IsFunction { get; }
        string SqlCommand { get; }
        IFunction Load(IConnector connector);
        IFunction Reload();
        object ExecuteRaw(IConnector connector, object data);
    }
    public interface IFunction<out TOutput, in TInput> : IFunction {
        TOutput Execute(IConnector connector, TInput data);
    }
}
```

Kód 14: Rozhraní IFunction

Literatura

[1] IBM. Funkce ve vydáních produktu Db2 a nabídkách produktu Db2. https://www.ibm.com/support/knowledgecenter/cs/SSEPGG_11.1.0/com.ibm.db2.luw.licensing.doc/doc/r0053238.html. [cit. 2019-04-22].

[2] IBM. Hlavní rysy produktů Db2 verze 11.1. https://www.ibm.com/support/knowledgecenter/cs/SSEPGG_11.1.0/com.ibm.db2.luw.wn.doc/doc/c0060311.html. [cit. 2019-04-22].

[3] IBM. IBM Developer : Download: IBM Data Studio. https://www.ibm.com/developerworks/downloads/im/data/index.html. [cit. 2019-04-22].

[4] Ing. Martin Zwierzyna ZWI0009. IS administrace účtů na katedrálních databázových systémech. https://dspace.vsb.cz/bitstream/handle/10084/98993/ZWI0009_FEI_B2647_2612R025_2013.pdf?sequence=1&isAllowed=y, 5 2013. [cit. 2019-04-23].

[5] Ing. Martin Zwierzyna ZWI0009. Informační systém pro správu databázových systémů. https://dspace.vsb.cz/bitstream/handle/10084/108606/ZWI0009_FEI_N2647_2612T025_2015.pdf?sequence=1&isAllowed=y, 5 2015. [cit. 2019-04-23].

[6] Jiří Tonar. Jak uplatnit principy třívrstvé architektury v rámci web integračního projektu | Webová integrace. http://www.web-integration.info/cs/blog/jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu/. [cit. 2019-04-23].

[7] Microsoft Corporation. ASP.NET MVC Pattern | .NET. https://dotnet.microsoft.com/apps/aspnet/mvc. [cit. 2019-04-24].

[8] Microsoft Corporation. ASP.NET Web Forms | .NET. https://dotnet.microsoft.com/apps/aspnet/web-forms. [cit. 2019-04-24].

[9] Microsoft Corporation. Nástroje pro vývojáře SQL | Microsoft. https://www.microsoft.com/cs-cz/sql-server/developer-tools. [cit. 2019-04-22].

[10] Microsoft Corporation. Přehled ADO.NET | Microsoft Docs. https://docs.microsoft.com/cs-cz/dotnet/framework/data/adonet/ado-net-overview. [cit. 2019-04-22].

[11] Microsoft Corporation. SQL Server 2016 | Microsoft. https://www.microsoft.com/cs-cz/sql-server/sql-server-2016. [cit. 2019-04-22].

[12] Microsoft Corporation. SQL Server 2017 Editions | Microsoft. https://www.microsoft.com/en-us/sql-server/sql-server-2017-editions#CP_StickyNav_1. [cit. 2019-04-22].

[13] Microsoft Corporation. SQL Server 2017 Editions datasheet_09192017. http://download.microsoft.com/download/A/7/0/A7049F57-D68B-49A2-941F-322D364342BC/SQL_Server_2017_Editions_Datasheet.pdf. [cit. 2019-04-22].

[14] Microsoft Corporation. SQL Server 2017 v systémech Windows a Linux | Microsoft. https://www.microsoft.com/cs-cz/sql-server/sql-server-2017. [cit. 2019-04-22].

[15] Microsoft Corporation. SQL Server 2017 v systémech Windows a Linux | Microsoft. https://www.microsoft.com/cs-cz/sql-server/sql-server-2017#ft1. [cit. 2019-04-22].

[16] Microsoft Corporation. Visual Studio IDE, Code Editor, Azure DevOps a App Center - Visual Studio. https://visualstudio.microsoft.com/cs/. [cit. 2019-04-24].

[17] Microsoft Corporation. Přehled – EF Core | Microsoft Docs. https://docs.microsoft.com/cs-cz/ef/core/, 10 2016. [cit. 2019-04-23].

[18] Nate McMaster. natemcmaster/DotNetCorePlugins: .NET Core library for loading assemblies as a plugin. https://github.com/natemcmaster/DotNetCorePlugins. [cit. 2019-04-23].

[19] Oracle. In-Memory Database | Oracle. https://www.oracle.com/database/technologies/in-memory.html. [cit. 2019-04-22].

[20] Oracle. MySQL :: MySQL Classic Edition. https://www.mysql.com/products/classic/. [cit. 2019-04-22].

[21] Oracle. MySQL :: MySQL Community Edition. https://www.mysql.com/products/community/. [cit. 2019-04-22].

[22] Oracle. MySQL :: MySQL Editions. https://www.mysql.com/products/. [cit. 2019-04-22].

[23] Oracle. Oracle Database 18c - Get Started. https://docs.oracle.com/en/database/oracle/oracle-database/18/index.html. [cit. 2019-04-22].

[24] The PostgreSQL Global Development Group. postgresql: About.

[25] The PostgreSQL Global Development Group. PostgreSQL: Downloads. https://www.postgresql.org/download/. [cit. 2019-04-22].