VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

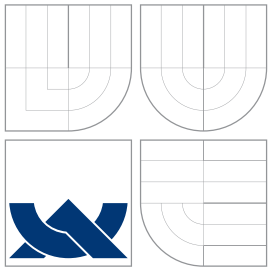
SYSTÉM PRO EVIDENCI SPOTŘEBY ENERGIÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

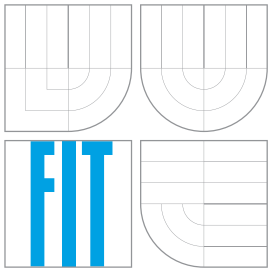
AUTOR PRÁCE
AUTHOR

MICHAL KOVÁČIK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO EVIDENCI SPOTŘEBY ENERGIÍ

ENERGY CONSUMPTION MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL KOVÁČIK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LADISLAV RUTTKAY

BRNO 2010

Abstrakt

Cílem bakalářské práce je implementovat webovou aplikaci pro sledování a evidenci spotřeby energií v domácnostech. Systém dovolí užívateli evidovat a sledovat spotřeby energií v jeho domácnosti. Systém rovněž srovná a navrhne užívateli řešení od jiných dodávatelů, pokud je to u konkrétní energie možné. Programovacím jazykem implementace je C# s použitím technologie Microsoft .NET a návrhovou metodologií objektově orientovaného programování.

Abstract

The goal of the bachelor's thesis is an implementation of web application for monitoring and evidence of energy consumptions in households. System allows user to store and monitor energy consumptions in his household. System also compares and nominates solutions from other providers, if it is possible with concrete energy. Programming language of implementation is C# with the use of Microsoft's .NET technology and object-oriented programming metodology.

Klíčová slova

C#, ASP.NET, SQL, XSLT, OOP, webová aplikace, spotřeba energií, systém pro evidenci

Keywords

C#, ASP.NET, SQL, XSLT, OOP, web application, energy consumption, management system

Citace

Michal Kováčik: Systém pro evidenci spotřeby energií, bakalářská práce, Brno, FIT VUT v Brně, 2010

Systém pro evidenci spotřeby energií

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Ladislava Ruttkaya. V práci som uviedol všetky publikácie a pramene, z ktorých som čerpal informácie.

.....
Michal Kováčik
18. května 2010

Poděkování

V prvom rade by som rád poďakoval Ing. Ladislavovi Ruttkayovi za pomoc a ochotu pri vypracovávaní tejto práce. Taktiež moje srdečné ďakujem patrí priateľke Petre a mojej rodine za podporu počas celého štúdia na FIT VUT v Brne.

Obsah

Kapitola 1

Úvod

V tejto dokumentácii sa nachádzajú všetky informácie týkajúce sa implementácie systému pre evidenciu spotreby energií v domácnosti. Pred popisom samotnej implementácie sú popisované použité technológie a prostriedky využité pri návrhu, a taktiež implementácii predchádza aj samotný návrh systému.

Motiváciou k vytvoreniu tejto aplikácie je systém dodávateľov energií v Českej republike, konkrétne sa jedná o systém distribútorov a dodávateľov elektriny a plynu, ktorý umožňuje zákazníkom vybrať si dodávateľa energie podľa vlastného uváženia a ponuky dodávateľa. Táto koncepcia však so sebou prináša pre bežného zákazníka zložitejšie, ba priam niekedy až chaotické vypočítavanie vlastných nákladov a výdavkov na spotrebu. Keď sa k tomu ešte pridá rozdelenie poplatkov do viacerých odberových skupín a podľa rôznych odberových parametrov u distribútorov a dodávateľov, človek nezaoberajúci sa problematikou môže mať vážne problémy zorientovať sa na tomto poli. Aplikácia by preto mala zjednodušiť a spríjemniť zákazníkom spravovanie ich výdavkov za energie.

Z momentálne dostupných riešení je možné využiť kalkulátory spotreby energie na stránkach väčšiny dodávateľov energií (napríklad kalkulátor cien plynu spoločnosti E.ON[1]). Vo väčšine prípadov sa jedná o malú pomôcku pre výpočet ceny, ktorú zaplatíte za ročnú spotrebu, po zadaní celkovej ročnej spotreby a ostatných potrebných údajov. Niektoré z týchto riešení ponúkajú dokonca aj porovnanie cien, ktoré by ste zaplatili u iných dodávateľov, no bohužiaľ ponuka dodávateľov pre porovnanie ani zďaleka nebýva kompletná. Kalkulátory sú síce jednoduché a rýchle, pričom tiež poskytujú aktuálne ceny, no taktiež v nich nie je možné hodnoty o spotrebe nijako ukladať či archivovať, čo sa javí ako najväčšia nevýhoda.

[1]Dostupný na adrese: http://www.eon.cz/php/cs/natural_gas_calculator/ (14.5.2010)

Tieto riešenia teda v konečnom dôsledku stále zostanú jednorázovými a vhodnými len, ak zákazník potrebuje rýchlo vypočítať orientačné ceny a nemá k dispozícii nič vhodnejšie. Kalkulátor dostupný na stránkach Energetického regulačného úradu Českej republiky[2] je na tom o niečo lepšie, možnosti jednoduchých kalkulátorov v podaní dodávateľov energií obohacuje o zadávanie údajov spotreby za jednotlivé mesiace v roku, a taktiež o kompletné porovnanie riešení iných dodávateľov. Ani táto pomôcka však nedovoľuje archiváciu, a tak už raz zadávané údaje o spotrebe nie sú nijako znovupoužiteľné. Pre vodu a výhrevné teplo sú kalkulátory veľmi zriedkavé a ide väčšinou o súkromné aplikácie. Cieľom tohto systému je preto zjednotiť správu a monitorovanie spotrieb pre viac druhov energií, ponúknuť ešte väčšiu voľnosť pri zadávaní údajov do systému a umožniť archiváciu údajov o spotrebe. Systém by teda mal byť komplexným riešením vhodným pre dlhodobé osobné použitie.

Analýzou riešenia, a tým aj implementačnými prostriedkami a technológiami, sa zaoberá hneď druhá kapitola. Obsahuje taktiež všeobecné informácie k objektovo orientovanému programovaniu a metodológii, popis využitých implementačných jazykov a popis systému dodávania energií v Českej republike, ktorý je hlavnou motiváciou tejto bakalárskej práce. Nasledujúca kapitola popisuje návrh riešenia v podobe diagramov a riešených problémov pri implementácii. Vybraté sú problémy, ktoré sú pre funkčnosť kľúčové. Štvrtá kapitola sa venuje samotnej implementácií systému, spolu s jeho grafickým rozhraním a možnosťami. V závere je zhodnotené splnenie cieľov, ktoré boli pre tento projekt stanovené.

Hlavným cieľom systému je jednoducho a interaktívne pomôcť užívateľom spravovať výdaje za energie vo svojej domácnosti. Dôraz sa kladie na jednoduchosť ovládania systému a jednoduchosť, a čo najväčšiu možnú transparentnosť práce s údajmi potrebnými pre výpočet výdajov. Taktiež by mal interaktívny systém ponúkať porovnania s ostatnými riešeniami od iných dodávateľov energií a pokrývať čo najväčší rozsah sledovaných položiek. Grafická reprezentácia v podobe grafov spotreby by mala užívateľovi monitorovanie ešte viac sprehľadniť a spríjemniť. Hlavná je prehľadná koncepcia grafického rozhrania, v ktorej budú jednotlivé sledované položky prehľadne oddelené a rozloženie nesmie byť mätúce alebo chaotické. Systém sa bude orientovať na odberovú skupinu domácností.

[2]Dostupný na adrese: http://kalkulator.eru.cz/ (14.5.2010)

Kapitola 2

Analýza riešenia

Táto kapitola sa venuje popisu princípov, metodológií a implementačných prostriedkov použitých pri vytváraní aplikácie. Na začiatku kapitoly je popísaná a rozobratá problematika systému distribútorov a dodávateľov v Českej republike pre jednotlivé energie.

2.1 Systém distribútorov a dodávateľov v Českej republike

Odberatelia energií v Českej republike majú možnosť si u niektorých energií zvoliť podľa vlastného výberu dodávateľa danej energie. Jedná sa konkrétne o elektrinu a plyn. Pri vode a výhrevnom teple je cena určená pre jednotlivé okresy, prípadne malé územia rôzna, no vyhýba sa zložitým vypočtovým postupom.

U elektriny a plynu sa systém skladá z distribútorov energie a dodávateľov energie. Distribútori pôsobia na určitom území republiky a ceny za distribúciu energie sú regulované Energetickým regulačným úradom Českej republiky a nemajú ich tak distribútori pevne vo svojich rukách. Tieto ceny sa pre rôznych distribútorov v rôznych oblastiach líšia, čo vedie k značnému zneprehľadneniu. Každopádne zákazník nemá možnosť vyberať si distribútora energie, distribútor je neovplyvniteľný vzhľadom na miesto distribúcie, pri výpočte sa preto použije cenník regulovaných poplatkov podľa oblasti disribúcie, v ktorej je odberateľom. Časť výdavkov vypočítaná z tarifov distribúcie tvorí tzv. regulovanú časť celkovej ceny.

Dodávatelia môžu byť rôzni v každej oblasti distribúcie. Cenu za dodávku jednotky energie spolu s cenou mesačného poplatku za dodanie a ďalšími menšími poplatkami si určuje dodávateľ samostatne. Dodávateľ môže mať dokonca rôzne ceny pre jednotlivé oblasti distribúcie. Pri tejto položke má však zákazník vo vlastných rukách to, pre ktorého

dodávateľa sa rozhodne, napríklad na základe dodávateľových cien. Regulované a taktiež aj ceny určené dodávateľmi sa samozrejme môžu časom meniť a pre výpočty výdavkov je vždy nutné použiť cenník aktuálny v danom období. Pre minulé poplatky pred určitou zmenou či úpravou cien je preto nutné použiť ceny, ktoré boli aktuálne v danom období. Teda je niekedy tiež nutné pracovať s viacerými verziami cenníkov súčasne.

Zákazník si môže meniť dodávateľa energie každých 6 mesiacov. Kategorizácia, druhy poplatkov a výpočet cien sa u elektriny a plynu líšia, a preto sa na ne pozrieme bližšie.

2.1.1 Kategorizácia a výpočet ceny odberov elektriny

Každý zákazník je pre svoj odber zaradený do odberovej kategórie distribučnou sadzbou, a je taktiež kategorizovaný typom ističa pre odber elektriny. Cena za elektrinu bola pri prechode na novú koncepciu platieb rozdelená na regulované platby za dopravu elektriny elektrickou sieťou do vašej domácnosti (poplatky za použitie energetickej siete) a platby za vlastnú odobratú energiu (tzv. „silovú elektrinu“). Výška regulovaných platieb je každoročne stanovená rozhodnutím Energetického regulačného úradu Českej republiky, neregulovaná časť je určovaná dodávateľom podľa situácie na trhu s elektrinou. Uvedené platby sa ďalej delia na menšie zložky.

Cena silovej elektriny zahŕňa:

- pevná cena za mesiac – jej výška je určená podľa produktovej rady dodávateľa,
- cena za odobratú jednotku elektriny, ktorá sa v niektorých kategóriách odberu delí na cenu v nízkom (NT) a vysokom (VT) tarife.

Vysoký a nízky tarif sa používa u takzvaných dvojtarifových produktov. Nízky tarif je zvýhodnená cena platná po určitú dobu dňa. Elektrina spotrebovaná v dobe platnosti nízkeho tarifu je účtovaná nižšou cenou. U jednotarifových kategórií existuje len jedna cena za odobratú energiu.

Cena za distribúciu zahŕňa:

- mesačný poplatok za príkon podľa prúdovej hodnoty hlavného ističa pred elektromerom – platí sa v stálej mesačnej výške, bez ohľadu na odobratú energiu,
- cena za dopravenú jednotku elektriny – v závislosti od kategórie sa môže opäť deliť na cenu vo vysokom a nízkom tarife.

Cena za dopravenú jednotku elektriny zahŕňa cenu systémových služieb, cenu na podporu výkupu elektriny a cenu za činnosť zúčtovania operátora trhu s elektrinou.

Daň z elektriny:

Od roku 2008 tvorí súčasť ceny elektriny tiež spotrebná daň z elektriny. Jedná sa o ekologickú daň vyplývajúcu zo záväzkov voči Európskej únií. Sadzba dane je pre všetkých jednotná a jej výška je 28,30 Kč/MWh.

Výpočet ceny za spotrebovanú elektrinu:

- stále platby – počet mesiacov x (mesačná platba za príkon podľa prúdovej hodnoty hlavného ističa pred elektromerom + pevná mesačná platba za silovú elektrinu),
- platba za spotrebu elektriny vo vysokom tarife – spotreba v MWh elektriny za obdobie vo VT x (cena za distribúciu 1MWh vo VT + cena systémových služieb + cena na podporu výkupu elektriny + cena za činnosť zúčtovania OTE + cena za 1MWh vo VT silovej elektriny),
- platba za spotrebu elektriny v nízkom tarife – spotreba v MWh elektriny za obdobie v NT x (cena za distribúciu 1MWh v NT + cena systémových služieb + cena na podporu výkupu elektriny + cena za činnosť zúčtovania OTE + cena za 1MWh v NT silovej elektriny).

Celková platba sa vypočíta sčítaním troch vyššie uvedených zložiek. Pred započítaním DPH k cene platby za spotrebu elektriny vo VT a NT je k cene pripočítaná daň z elektriny za každú spotrebovanú jednotku elektriny (za každý 1MWh). [1]

2.1.2 Kategorizácia a výpočet ceny odberov plynu

Ceny pre zákazníka sú kategorizované podľa výšky jeho ročného odberu plynu. Rovnako ako pri elektrine je celková cena rozdelená na regulovanú zložku za distribúciu plynu do odberového miesta určenú Energetickým regulačným úradom Českej republiky a na neregulovanú zložku dodávky plynu, ktorá je určená konkrétnym dodávateľom a aktuálnym stavom trhu s plynom. Spomenuté dve zložky ceny sa skladajú z niekoľkých menších zložiek.

V minulosti sa používalo meranie spotreby plynu v jednotke m^3, po novom sa prešlo na používanie jednotky MWh, rovnako ako pri elektrine. Medzi týmito dvomi jednotkami

je prepočet jednoduchý, hodnota odberu v jednotke m^3 sa vynásobí koeficientom prepočtu a koeficientom spalného tepla a dostaneme hodnotu v MWh. Od meraní spotreby v m^3 sa upustilo hlavne kvôli závislosti prepočtu na vedľajších vplyvoch ako napr. nadmorská výška, koeficienty sú prepočítavané a určované približne každoročne. Pri rôznych podmienkach a vplyvoch môže mať rovnaká hodnota v m^3 v rôznych miestach odberu inú hodnotu po prepočítaní do MWh.

Pri ročnom odbere nad 63 MWh sa používa namiesto stáleho mesačného poplatku pevná ročná cena za dennú rezervovanú pevnú distribučnú kapacitu.

Cena distribúcie zahŕňa:

- stály mesačný poplatok za pristavenú kapacitu – výška poplatku je určená podľa výšky ročnej spotreby,
- pevná cena za odobratý plyn – cena za každú odobratú jednotku plynu.

V každom regióne odberu má väčšinou každý jeden distribútor rôzne regulované ceny.

Cena dodávky zahŕňa:

- stály mesačný poplatok za dodávku plynu – výška poplatku určená podľa výšky ročnej spotreby,
- jednotková hraničná komoditná cena – cena za každú odobratú jednotku plynu.

Ceny dodávky plynu sa skladajú z menších zložiek ako pevná cena prepravy plynu, pevná cena štruktúrovanej dodávky plynu, cena za odobratý zemný plyn a cena za služby obchodu.

Výpočet ceny za spotrebovaný plyn:

- stále platby – počet mesiacov x (stály mesačný poplatok za pristavenú kapacitu + stály mesačný poplatok za dodávku plynu),
- cena za spotrebovaný plyn – spotreba MWh plynu za obdobie x (pevná cena za odobratý plyn + jednotková hraničná komoditná cena).

K cene za spotrebovaný plyn sa pripočítava tiež pevná cena za služby operátora trhu za každú odobratú jednotku (1MWh). Celková platba sa vypočíta sčítaním dvoch vyššie uvedených zložiek.

2.2 Objektovo orientované programovanie

Objektovo orientované programovanie (OOP) je logickým pokračovaním štruktúrovaného a modulárneho programovania. Pri tomto druhu programovania sa používajú tzv. **objekty**, ktoré môžu tiež existovať ako samostatné programové entity. Na objektový typ sa môžeme pozerať ako na abstraktný vzor pre vytváranie „podobných“ objektov, ktoré dedia vlastnosti predka. Objektový typ je vlastne modul postavený do role dátového typu. Objekty obsahujú vlastné atribúty a metódy, pomocou ktorých vykonávajú činnosti potrebné pre prácu s objektom. [10]

OOP stojí na troch základných princípoch:

- **Zapuzdrenie** – nový dátový typ – triedu, definujeme ako množinu dát spolu s operáciami nad nimi,
- **Dedičnosť** – od dátového typu môžeme odvodiť nový, ktorý prevezme tie vlastnosti, ktoré explicitne nezmeníme,
- **Polymorfizmus** – s inštanciou typu odvodeného môžeme zaobchádzať rovnakým spôsobom ako s inštanciou typu pôvodného (potomok môže vždy zastúpiť predka).

Dedičnosť uľahčuje používanie kódu, ktorý už je napísaný a nie je nutné ho znovu implementovať pre každý objektový typ s obdobnými vlastnosťami. S polymorfizmom tak umožňuje pracovať s množinami objektov, ktoré sú inštanciami typov odvodenými od spoločného predka jednotne. Základné vlastnosti spomínané vyššie taktiež dovoľujú skrývanie informácií a implementácií k obmedzeniu externej viditeľnosti implementácie, využívanie identity objektov k práci s každým objektom ako so samostatnou softvérovou entitou. Správy sú prostriedkom pre komunikáciu objektov ako je vyžiadanie metód atď.

2.3 Unified Modeling Language (UML)

Unified Modeling Language (UML) je grafický jazyk používaný v softvérovom inžinierstve. Uplatňuje sa pri špecifikácii, vizualizácii, dokumentovaní a navrhovaní počas vývoja programového systému. Štandardným spôsobom je v ňom možné zapisovať návrhy vrátane konceptuálnych modelov s ich prvkami ako systémové funkcie a business procesy, ale aj

konkrétne prvky ako schémy databázy, programové komponenty a samotný programovací jazyk.

K vyvíjaným systémom a ich návrhu, popisu a analýze podporuje objektovo orientovaný prístup. Štandard UML definuje štandardizačná skupina **Object Management Group** (OMG). Na systém sa pozerá pomocou rôznych pohľadov, pričom v konkrétnom pohľade sa zameriava na určitý aspekt a ostatné ignoruje. Na jeden systém môže teda existovať viacero pohľadov pre rôzne projekcie, ktoré sú modelované modelmi používanými v UML. Pri štrukturálnom pohľade je popisovaná vrstva medzi objektami a triedami a ich komunikácia. Pohľad správania popisuje vzájomné pôsobenie a ovplyvňovanie systémových komponentov, zatiaľ čo dátový pohľad sa zameriava na ich stavy a väzby. Zapúzdrením systémových častí a ich použitím okolím systému za zaoberá pohľad rozhrania. [9]

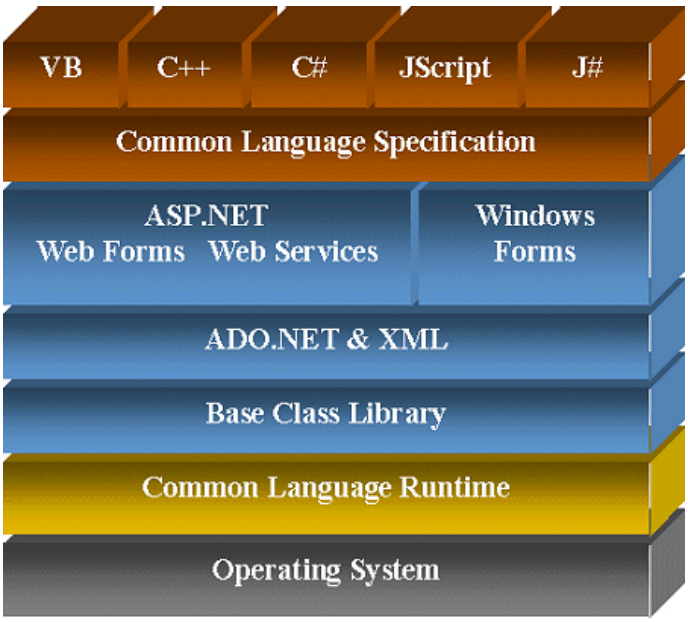
Najpoužívanejšie časti štandardu UML sú diagramy. V UML 2.0 sú to nasledovné: [5]

- **štruktúrálne diagramy**
  - diagram tried (class diagram)
  - diagram komponentov (component diagram)
  - diagram zloženej štruktúry (composite structure diagram)
  - diagram nasadenia (deployment diagram)
  - diagram balíčkov (package diagram)
  - diagram objektov (object diagram)
- **diagramy správania**
  - diagram aktivít (activity diagram)
  - diagram prípadov užitia (use-case diagram)
  - stavový diagram (state machine diagram)
  - **diagramy interakcie**
    - sekvenčný diagram (sequence diagram)
    - diagram komunikácie (communication diagram)
    - diagram prehľadu interakcií (interaction overview diagram)
    - diagram časovania (timing diagram)

2.4 Implementačný jazyk

2.4.1 .NET Framework

Technológia .NET, ktorá vznikla, a je aktívne podporovaná v spoločnosti Microsoft, je spoločný názov pre súbor softvérových technológií tvoriacich celú platformu. Základom je štandardizovaná špecifikácia jadra – **Common Language Infrastructure**. Hlavnou komponentou je **Microsoft .NET Framework**, prostredie pre beh aplikácií spolu s knižnicami a spúšťacím rozhraním. Microsoft svoju technológiu takisto podporil vývojovým prostredím **Microsoft Visual Studio .NET**. Microsoft .NET Framework je určená pre osobné počítače s operačným systémom Microsoft Windows, takisto však táto platforma existuje pre vreckové počítače a embedded zariadenia. Na všetkých platformách zdieľa .NET rovnaké základné princípy, čo činí prechod medzi nimi jednoduchší. V čase písania tejto práce vyšla verzia 4.0 spolu s novými verziami jazykov C# (4.0) a VB.NET (10.0) a vývojovým prostredím Visual Studio 2010. Vzhľadom na vydanie verzie 4.0 po začatí písania tejto práce je v nej využitá posledná verzia .NET 3.5 vydaná v roku 2007, s verziami jazykov C# (3.5) a VB.NET (9.0) a vývojovým prostredím Visual Studio 2008.[2]



Obrázok 2.1: .NET Framework. Obrázok prevzatý z [4].

2.4.2 Jazyk C#

Jazyk C# je objektovo orientovaný programovací jazyk spoločnosti Microsoft v rámci politiky jednoduchosti programovania, ktorú spoločnosť presadzuje spolu s technológiami .NET. Jazyk C# je často označovaný za takzvaný Java-like jazyk, pretože ako základ pre jeho vytvorenie si Microsoft okrem jazyka C++ zobral takisto jazyk Java a jeho princípy. Hlavným cieľom bolo využiť silu programovania v C++ spolu s typom programovania využívaným napr. vo Visual Basic, Delphi – tzv. rýchle programovanie (**Rapid Application Development**).

Jazyk C# bol vytváraný s cieľom vytvorenia jednoduchého a moderného jazyka pre všeobecné použitie. Jeho zameranie je široké, od aplikácií pre hostované, ako aj embedded systémy s ohľadom na škálovateľnosť systémov používajúcich operačné systémy po zariadenia určené na špecializovanú prácu. Navrhnutý bol s cieľom, aby umožňoval využitie všetkých vlastností vrstvy CLI (Common Language Runtime), ktorá pod ním leží. Väčšina typov zavedených v tomto jazyku korešponduje s hodnotovými typmi implementovanými v CLI Framework, narozdiel od jazykov s vlastnou syntaxou, ktoré využívajú len podmnožinu CLI. Jazyk takisto obsahuje tzv. **Garbage collector**, známy hlavne z programovacieho jazyka Java, ktorý sa stará o automatické uvoľňovanie alokovanej pamäte. Narozdiel od C++ je v C# dovolená len jednoduchá dedičnosť. [11]

2.4.3 ASP.NET

Súčasťou .NET Framework je taktiež ASP.NET. Je zameraná pre tvorbu webových aplikácií a služieb, čím sa stala nástupcom technológie ASP – Active Server Pages. Vďaka CLR (**Common Language Runtime**), ktorý je zdieľaný všetkými aplikáciami postavenými na .NET Framework, je možné aplikáciu implementovať v rôznych jazykoch podporujúcich CLR a riešenia medzi nimi zdieľať. Aplikácie na báze ASP.NET sú predkompilované, čím sa stávajú rýchlejšími od bežného parsovania pri každom prístupe.

Prostredie predstavuje jednoduchý prechod od tvorby aplikácií pre Windows k webovým aplikáciám tým, že využíva práve objekty a ovládacie prvky, ktoré sú obdobné. Rovnako je však možné používať bežné prvky HTML, ktoré však, narozdiel od serverových prvkov s rôznymi pokročilými možnosťami, ovládaním a generovaním udalostí, ponúkajú len základnú funkčnosť. Jazykom pre implementáciu kódu vykonávaného na serveri webovej stránky je C# alebo Visual Basic .NET. Komponenty napísané v jednom z týchto jazykov

je dokonca možné spoločne používať v rámci jedného zostavenia.

Každá ASP.NET stránka má niekoľko fáz životného cyklu, ktoré predchádzajú jej zobrazeniu a je možné ich taktiež využiť pri ich programovaní. Narozdiel od ASP, kde sa kód stránky vykonával od prvého do posledného riadku stránky, v ASP.NET je všetko inak a jedná sa už o plnohodnotné objektové programovanie. [7]

2.5 Kaskádové štýly

Kaskádové štýly alebo tiež CSS (Cascading Style Sheets) sú všeobecným rozšírením (X)HTML. CSS umožňujú oddeliť vizuálne a formátovacie prvky internetových dokumentov od štruktúry. Používaním CSS sa získava prehľadný kód, v ktorom je možné časti vzhľadu zhodujúce sa s nejakým typom alebo vizuálnou triedou meniť centralizovane. Takisto zmenšuje dátovú veľkosť internetového dokumentu a zaisťuje rovnaké vykresľovanie v rôznych prehliadačoch. [12]

2.6 Structured Query Language (SQL)

Structured Query Language (SQL) je v súčasnosti najpoužívanejší jazyk svojho druhu v relačných systémoch riadenia báz dát. Jazyk SQL prešiel za dobu svojej existencie množstvom verzií a modifikácií a v súčasnosti už podporuje moderné technológie s veľkým uplatnením pri práci s dátami. SQL má nasledovné základné syntaktické konštrukcie:

- **manipulácia dát (DML)** – príkazy používané na manipuláciu s dátami *INSERT*, *DELETE*, *UPDATE*, a taktiež príkaz pre vytváranie dotazov a vyberanie potrebných dát pomocou nich *SELECT* spolu s jeho klauzulami na vyšpecifikovanie cieľových dát,
- **definícia dát (DDL)** – príkazy pre vytváranie databázových objektov, ich upravovanie a rušenie (*CREATE*, *ALTER*, *DROP*),
- **riadenie dát (DCL)** – napr. príkazy na modifikovanie prístupových práv užívateľov k objektom v databáze (*GRANT*, *REVOKE*).

Taktiež sa používajú príkazy na riadenie transakcií. V súčasnej dobe sa jazyk SQL vyskytuje v rôznych formách, verziách a s rôznymi rošíreniami. V tejto práci je použitá im-

plementácia relačného databázového serveru od spoločnosti Microsoft vo verzii **Microsoft SQL Server 2008**. [13]

2.7 Transformácie XSL

XSLT (**eXtensible Stylesheet Language Transformations**) je deklaratívny jazyk založený na XML, používaný pre transformácie XML dokumentov na iné XML dokumenty. Pôvodný dokument pritom nie je zmenený, transformácia sa využije pri jeho zobrazení. Väčšinou sa jedná o transformácie do XML s inou štruktúrou, HTML alebo do čistej textovej podoby. Takisto sa objavuje pri vytváraní výstupu tlače, kde XSL formátovacie objekty môžu byť prevedené do rôznych formátov. Za XSLT stojí konzorcium **World Wide Web Consortium** (W3C) a jeho zatiaľ poslednou špecifikáciou je verzia 2.0 z roku 2007. [8]

2.8 Vývojové prostredie

Pre implementáciu systému je dôležité si vhodne zvoliť vývojové implementačné prostredie. Najlepšou voľbou pri tvorbe webovej aplikácie na báze .NET sa javí prostredie z dielne Microsoft, ktorý s technológiou .NET vyrukoval. Prostredie **Microsoft Visual Studio** bude tak vysokou oporou pri vývoji aplikácie na tejto báze, spolu s pokročilým debugovacím nástrojom a prispôsobeným editorom, ktorý obsahuje. Prínosom bude takisto jednoduchá práca s CSS. Microsoft Visual Studio je primárnym a popredným vývojovým nástrojom pre programovanie s technológiou .NET. [3]

Kapitola 3

Návrh riešenia

Návrh aplikácie je sprevádzaný rozdeľovaním a plánovaním prác s vytváraním diagramov pomocou vývojových prostriedkov. Kapitola sa najprv na návrh pozerá z pohľadu získavania informácií a rozdelenia aplikácie na logické celky, čo prinesie štrukturalizáciu prác na aplikácii. Taktiež sa tu nachádzajú zobrazenia diagramu prípadov užitia a konceptuálneho dátového modelu, doprevádzané detailným popisom.

3.1 Získavanie informácií

Ešte pred začiatkom zamýšľania sa nad systémom a jeho návrhom je dôležité zoznámiť sa čo najlepšie s problematikou výpočtu cien energií, ale taktiež so samotnými dodávateľmi a distribútormi, ich pôsobnosťou, rozdelením, a spôsobmi a frekvenciami zverejňovania a aktualizovania ich cien a vnútorných prerozdelení.

V tejto fáze zohráva veľkú úlohu prosté zisťovanie a hromadenie množstva rôznych užitočných informácií. Hlavným zdrojom oficiálnych vyhlásení, pravidiel a aktuálnych, ale aj minulých hodnôt regulovaných cien je samotný **Energetický regulačný úrad Českej republiky** a jeho webová stránka. Je taktiež vo veľkej miere zdrojom informácií a kontaktov na dodávateľov, či zdrojom popisov princípov niektorých kategorizácií či výpočtov. Väčšina dodávateľov má svoj osobitý spôsob a štýl zverejňovania cien a kategorizácie, je preto nutné si osvojiť vlastný systém pre zefektívnenie a zjednotenie získaných informácií od dodávateľov. Cieľom pre nájdenie a osvojenie si vlastného systému je zjednodušenie a zefektívnenie evidencie dát a údajov. Vlastný systém evidencie by mal zodpovedať informatívnosti a obsahu štýlov distribútorov a dodávateľov, avšak so zameraním a prispôsobením

pre integráciu v systéme. Možnosťou je tiež osvojenie si kvalitných a efektívnych spôsobov niektorého z distribútorov, respektíve dodávateľov.

Taktiež rôzne druhy energií majú okrem rôznych cien a princípov výpočtov aj rôzne číselníky a ich potrebný počet pre prácu s cenami energie. Počet číselníkov sa takmer vždy u energií značne líši. Dôležité je zamerať sa na spoločné znaky jednotlivých energií a ich číselníkov a snažiť sa pri zachovaní všetkých informácií čo najlepšie zjednotiť evidenciu.

Ďalším zdrojom informácií sú v tomto prípade veľmi jednoduché kalkulátory pre výpočet ceny spotreby nachádzajúce sa prevažne na stránkach dodávateľov, alebo iné malé pomôcky, ktorých funkčnosť bude systém rozširovať a zjednocovať. Hlavnou a najdôležitejšou informáciou, ktorú tieto malé pomôcky nesú, je spôsob výpočtu spotreby za obdobie a zvolené obdobie, pre ktoré je spotreba počitaná. Všímaním si detailov aj u takýchto malých pomôcok a kalkulátorov s veľmi obmedzenými vlastnosťami a funkčnosťou je možné predísť chybám pri implementácií a nájsť vhodnú a prospešnú mieru možností nastavovania parametrov.

Po získaní dostatočného množstva informácií z rôznych zdrojov je možné vyhnúť sa problematickým konštrukciám pri návrhu a implementácií a zjednodušiť si prácu vďaka optimalizovaniu spôsobov a princípov, ktoré budú využívané.

3.2 Rozdelenie aplikácie na logické celky

Pri informačnom systéme akým je systém pre evidenciu spotreby energií v domácnosti, v ktorom užívatelia využívajú centrálne všeobecné dáta pre ďalšiu funkčnosť, je dôležité systém rozdeliť na niekoľko logických častí.

Návštevníkovi sa sprístupní vždy časť systému, v ktorej má práva sa pohybovať. Osobné údaje užívateľov, nastavovanie a zadávanie dát určených všetkým užívateľom sú citlivé časti systému. Je preto dôležité, aby boli prístupné len po autentifikácii užívateľa v systéme. Preto sa pri návšteve systému vždy najskôr spustí stránka bez konkrétnych dát, len s možnosťami na vstup do hlbších častí systému využívajúcich tiež súkromné údaje. Bežný návštevník sa z nej môže dozvedieť viac o systéme, prihlásiť sa doň pomocou osobných prihlasovacích údajov, čím mu bude sprístupnená užívateľská časť systému, spolu s jeho osobnými údajmi a dátami, ktoré do systému zadal, alebo v prípade, že návštevník zatiaľ nemá v systéme založený účet, práve v tejto časti systému k jeho založeniu môže pristúpiť a využívať tak

po prihlásení funkcie systému vyžadujúce prihlásenie doň. Taktiež je možné, po zadaní platného mena užívateľského účtu na úvodnej stránke, nechať si vygenerovať a zaslať nové heslo pre vstup do systému v prípade jeho zabudnutia.

V systéme takéhoto typu je veľmi dôležité, aby disponoval aktuálnymi a validnými spoločnými dátami, ktoré budú využívať užívatelia pri ich práci so systémom a sledovaní svojich odberov. Pre rýchle a pohodlné vkladanie dát do systému je preto vhodné, aby systém mal svoju vlastnú administračnú časť, v ktorej bude možné efektívne pracovanie s centrálnymi dátami. Administračná časť by mala mať určených svojich vlastných správcov, ktorí do nej budú mať prístup, a taktiež by bola pre systém prínosná možnosť prideľovania oprávení už priamo v ňom, existujúcimi správcami. Týmito opatreniami sa odbúrajú v podstate všetky manuálne prístupy a modifikácie databázy. Všetky databázové objekty, spolu s dátami v nich, by mali byť prístupné priamo z aplikácie. Pravdaže administračná časť systému vedie k zjednodušeniu a zefektívneniu práce s číselníkmi a hodnotami systému len v prípade, že je dostatočné prehľadná, intuitívna, rýchla a vyvaruje sa chybám a možnostiam zadania nekonzistentných dát do databázy.

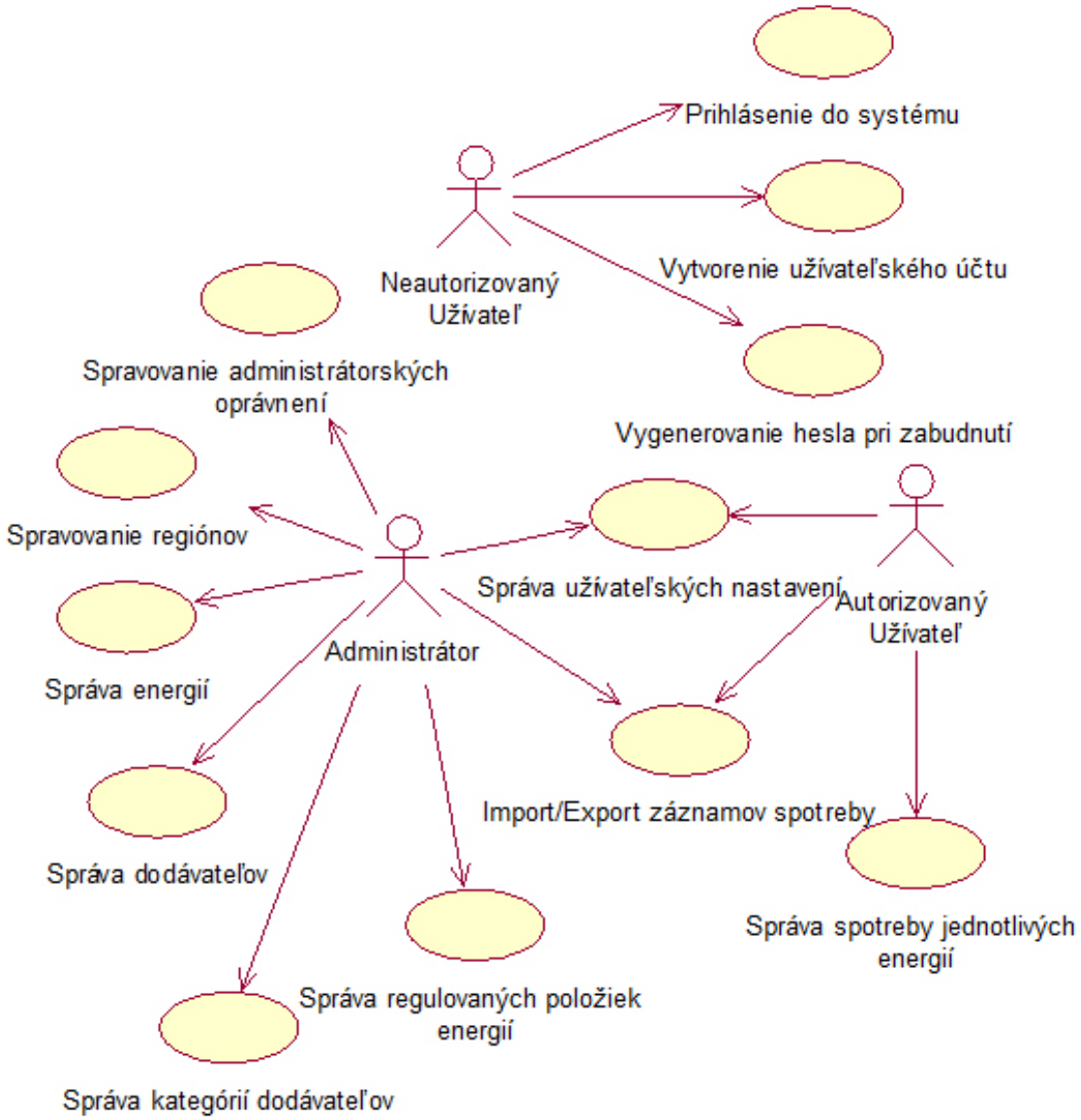
Vzhľadom na to, že systém bude pracovať s viacerými typmi energií, pričom s každou jednou bude zrejme manipulovať rozličným spôsobom a hlavne výhradne, je dôležité tiež sprehľadniť užívateľskú časť systému vhodným rozdelením na jednotlivé energie a ostatné možnosti systému.

Keď si toto rozvrhnutie nakoniec zosumarizujeme, systém sa bude teda skladať z troch logických častí. Úvodnej, prístupnej každému, slúžiacej ako vstupná brána do ostatných častí systému, administračnej, prístupnej len určeným vybraným užívateľom a užívateľskej, prístupnej registrovaným užívateľom pre využívanie služieb a funkcií evidencie a monitorovania systému.

3.3 Use-Case diagram

Pri vytváraní informačného systému hrajú veľkú rolu diagramy jazyka UML, ktoré modelujú rôzne aspekty systému z rôznych pohľadov na ne. Jedným z týchto diagramov je aj **diagram prípadov užitia** systému (Use-Case diagram). Na systém sa pozerá z pohľadu jeho hraníc a možností, pričom modeluje účastníkov systému spolu s možnými interakciami medzi nimi a systémom.

Systém pre evidenciu spotreby energií v domácnostiach okrem bežného užívateľa vyžaduje tiež namodelovanie užívateľa s administrátorskými (spravovateľskými) právamami. O rozlišovanie druhu užívateľského účtu v systéme sa stará autentifikácia pri prihlasovaní do systému. Tá po zistení užívateľských práv automaticky nasmeruje užívateľa do časti systému pre neho určenej. Takisto systém sa podieľa na inteakciách generovaním grafov a výpočtami cien odberov za určité obdobia.



Obrázok 3.1: Use-Case diagram

Hlavné možnosti účastníkov v tomto systéme sú zakreslené v diagrame spolu s väzbami. Užívateľ využíva funkcie a zobrazenia systému pre získavanie pre neho potrebných informácií, o systém sa nemusí nijako starať ani ho udržiavať. Hodnoty zadávané užívateľom slúžia jemu samotnému, využívajú sa pri výpočtoch a zobrazovaní, avšak tieto konkrétne hodnoty sú užívateľovi vlastné a nebudú v systéme používané pri iných užívateľoch, či v úlohe všeobecných dát. Užívateľ má možnosť nastavovať a meniť nastavenia svojho užívateľského účtu ako regionálne informácie, osobné dáta, heslo k účtu a určenie energií, ktoré budú monitorované. Užívateľ môže taktiež do systému naimportovať pomocou XML dokumentu dáta o jeho odberoch energií, a pri každej konkrétnej energii spravovať a pridávať jeho odbery energie.

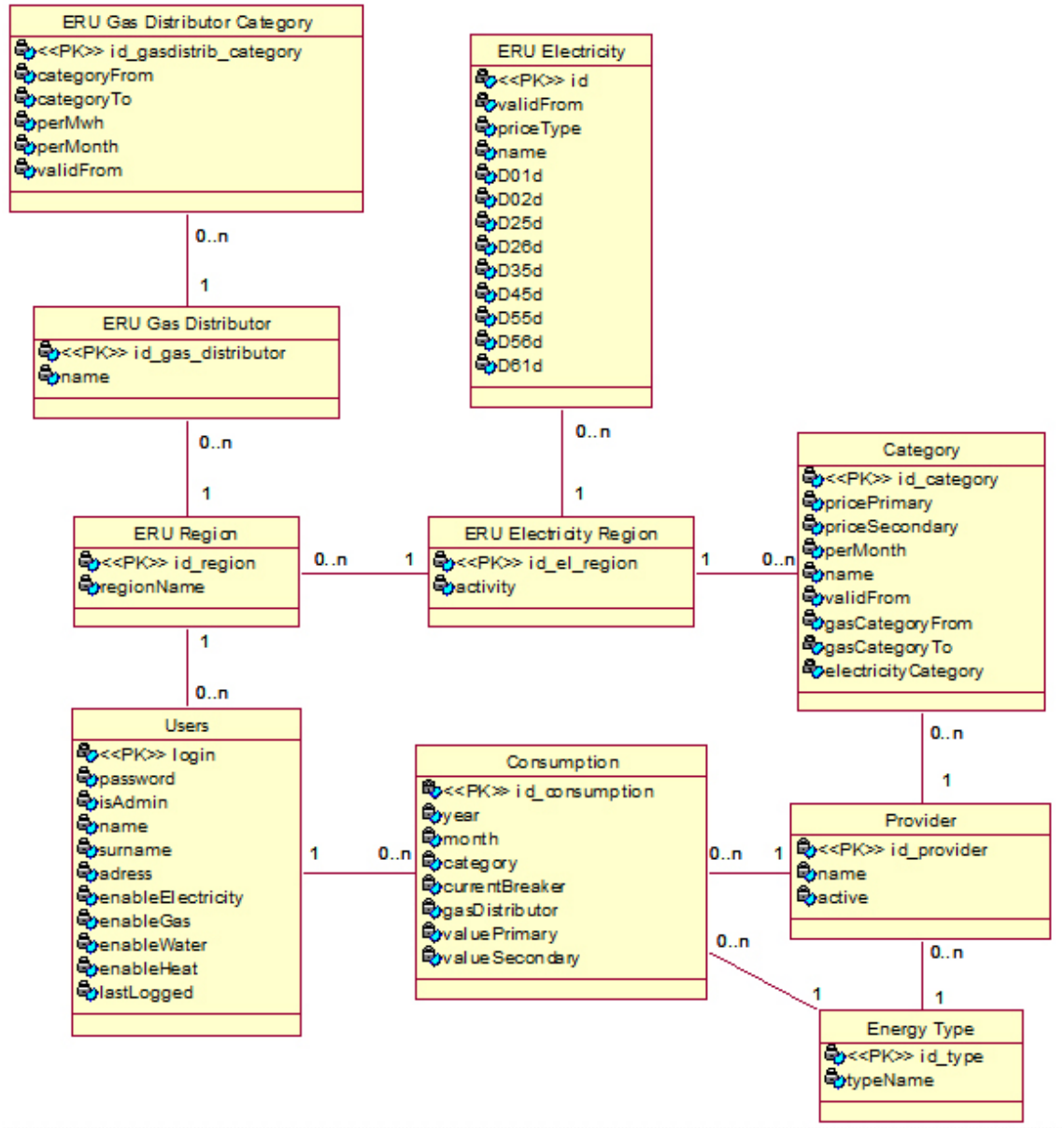
Udržiavanie, pre užívateľov spoločných dát, má na starosti ďalší účastník systému, ktorým je užívateľ s administrátorskými právami. Ten sa o dáta stará, manipuluje s nimi a zaisťuje ich prístupnosť využitiu pre bežných užívateľov. Pomocou administračnej časti systému vkladá informácie o energiách, dodávateľoch, dodávateľských kategóriách a distribútoroch do databázy. Pod právomoc administrátorského účtu spadajú tiež regulované hodnoty a ceny energií, a taktiež spravovanie administrátorských práv samotných. Administrátor môže však, rovnako ako užívateľ, meniť nastavenia svojho účtu a importovať dáta o jeho odberoch, ktoré by však boli zobrazené a využiteľné až po odobratí jeho administrátorských práv a jeho vstupe do časti systému pre registrovaných užívateľov bez administrátorskych práv.

K dôležitým interakciám so systémom patrí tiež vygenerovanie nového hesla užívateľom v prípade jeho zabudnutia, a taktiež vytváranie nového užívateľského účtu, ku ktorému môže pristúpiť ktokoľvek bez autentifikácie v systéme.

3.4 Entity-Relationship diagram

Entity-Relationship diagram (ER diagram) slúži pre návrh uloženia dát aplikačnej domény a ich vzťahov na vyššej úrovni abstrakcie. Patrí medzi stavové diagramy, vzhľadom na to, že zachytáva dátový model aplikácie. Prezentuje stav systému na dátach uchovávaných v systéme, ktoré sú perzistentne uložené na dátovom zdroji. Na rozdiel od Use-case diagramu nemodeluje účastníkov a ich interakcie, ale model uloženia hodnôt využívaných v interakciách a ich vzájomné súvislosti.

V tomto prípade pozostáva diagram z 10 na sebe závislých entitných množín zreťazujúcich informácie o odobraných spotrebách, či regulovaných cenách a kategorizácii:



Obrázok 3.2: Entity-Relationship diagram

- **Energy Type** je obrazom energií, s ktorými systém pracuje. Jednoznačne sú identifikované atribútom *id_type* a pomenovanie energie nesie atribút *typeName*.
- **Users** je entita modelujúca užívateľa systému s atribútmi určujúcimi jeho užívateľské nastavenia. Spolu s prihlasovacími údajmi užívateľa v podobe atribútov *login* a

password tu sú uchovávané jeho osobné informácie v podobe mena, priezviska, adresy a dátumu posledného prihlásenia do systému v zodpovedajúcich atribútoch *name*, *surname*, *adress* a *lastLogged*. Dôležitým je atribút *isAdmin*, ktorý určuje oprávnenie užívateľa na vstup do administračnej časti aplikácie. Atribúty *enableElectricity*, *enableGas*, *enableWater*, *enableHeat* uchovávajú nastavenie monitorovania jednotlivých energií.

- **Consumption** predstavuje entitu, ktorá uchováva spotrebu určitej energie v priebehu mesiaca. Je určená vlastným unikátnym identifikátorom *id_consumption*, no kombinácia atribútov *year*, *month* pre určitého užívateľa a určitý typ energie musí byť taktiež unikátna. Zamedzí sa tak existencii viacerých záznamov pre záznam spotreby v mesiaci určitého roku a nejednoznačnosti pri určovaní, ktorý záznam má byť použitý pri evidencii spotreby. Pre záznam o spotrebe elektrickej energie je nutné naplnenie atribútov *category* a *currentBreaker*, ktoré špecifikujú ceny využité pri výpočte. Atribút *category* predstavuje distribučnú sadzbu elektrického odberu, atribút *currentBreaker* zas typ ističa prijímajúceho dodávku. Tieto atribúty sú využiteľné pri iných energiách pre ukladanie ich vlastných kategorizácií alebo iných hodnôt, rovnako ako atribút *gasDistributor* primárne špecifikujúci distribútora u záznamov o spotrebe plynu. Atribúty *valuePrimary* a *valueSecondary* slúžia pre uchovanie veľkosti mesačnej spotreby v prípade dvojzložkovej spotreby, v prípade jednozložkovej spotreby je primárne využiteľný atribút *valuePrimary*.

- **Provider** modeluje dodávateľa určitej energie. Na jeho jednoznačnú identifikáciu slúži atribút *id_provider*, atribút *name* uchováva obchodné meno dodávateľa a atribút *active* určuje aktivitu dodávateľa na trhu. Zrušením aktivity dodávateľa je možné odstaviť ho z ponuky dodávateľov pre zvolenie dodávateľa energie užívateľmi v systéme.

- **Category** uchováva kategórie, ktoré ponúkajú dodávatelia zákazníkom, spolu s cenami za ich používanie, bližším určením a platnosťou. Kategória je identifikovaná jednoznačne identifikátorom v podobe atribútu *id_category*. Atribúty *pricePrimary* a *priceSecondary* obsahujú ceny, ktoré si účtuje dodávateľ za zložky energie v prípade dvojzložkovej ceny, v prípade jednozložkovej ceny sa primárne používa na uchovanie ceny atribút *pricePrimary*. Mesačný poplatok za kategóriu v prípade, že pre konkrétnu energiu existuje, predstavuje atribút *perMonth*. Platnosť záznamu, a teda cien

kategórie, určuje atribút *validFrom*, atribút *name* je voľným pomenovaním kategórie. Atribúty *gasCategoryFrom*, *gasCategoryTo* a *electricityCategory* bližšie určujú kategórie spotreby. Prvé dva tak činnia pre spotrebu plynu, tretí pre elektrinu.

- **ERU Region** ukladá regióny, do ktorých je distribúcia primárne rozdelená. Každý región je identifikovaný hodnotou atribútu *id_region* a pomenovanie regiónu, s ktorým sa vyskytuje, nesie atribút *regionName*.

- **ERU Gas Distributor** je entita pre distribútorov plynu jednoznačne identifikovaná atribútom *id_gas_distributor*. Pre uloženie obchodného mena distribútora slúži atribút *name*.

- **ERU Gas Distributor Category** predstavuje kategóriu cien distribútora plynu jednoznačne identifikovanú atribútom *id_gasdistrib_category* a určenú intervalom ročnej spotreby plynu užívateľom s hranicami uloženými v atribútoch *categoryFrom* a *categoryTo*. Interval je sprava uzavretý. Platnosť cien určuje atribút *validFrom*, pričom samotné ceny sú určené atribútmi *perMwh* a *perMonth*, kde prvý predstavuje cenu za spotrebovanú jednotku energie a druhý mesačný poplatok za odbery v kategórii.

- **ERU Electricity Region** vystupuje v systéme s menom uloženým priamo v jednoznačnom identifikátore, ktorým je atribút *id_el_region*. Atribút *activity* určuje aktivitu regiónu a jeho možnosť zvolenia administrátorom pri vkladaní dodávateľskej kategórie z ponuky regiónov distribúcie elektriny.

- **ERU Electricity** vo svojej štruktúre uchováva regulované ceny pre distribúciu elektriny pre jednotlivé kategórie odberov a typy kapacity ističov prijímajúcich elektrinu. Jednoznačne sa idetifikuje atribútom *id*, platnosť položky je určená atribútom *validFrom*, *priceType* určuje charakter ceny v podobe typu ističa alebo iných poplatkov, ktorý môže byť popísaný vo voliteľnom atribúte *name*. Atribúty *D01d* až *D61d* zodpovedajú kategórii, ktorú má spotreba užívateľa určenú.

Každý užívateľ môže mať niekoľko záznamov o spotrebe energie, každý záznam o spotrebe je určitého typu energie a má svojho dodávateľa. Každý dodávateľ je dodávateľom jedného typu energie a energiu dodáva v niekoľkých kategóriách, či už na základe kategorizácie v rámci energie, alebo iba s úmyslom prideľovania platnosti jednotlivým cenovým sadzbám. Dodávateľské kategórie pre elektrickú energiu majú pridelený elektrický región

distribúcie, ostatné typy energií s touto skutočnosťou nepracujú. Užívateľ odoberá energie v niektorom z regiónov Českej republiky podľa prerozdelenia Energetickým regulačným úradom Českej republiky. Každý región spadá do niektorého z elektrických regiónov distribúcie, a pre každý z týchto elektrických regiónov distribúcie existuje niekoľko záznamov o regulovaných hodnotách. Od regiónov sa odvíja aj rozdelenie distribútorov plynu, z ktorých každý pôsobí v inom regióne. Každý distribútor plynu má niekoľko záznamov regulovaných cien, ktoré sú prerozdelené do kategórií podľa ročného odberu užívateľa.

Kapitola 4

Implementácia

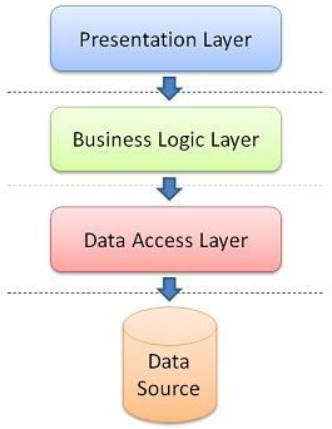
Na začiatku kapitoly je priblížená trojvrstvová architektúra aplikácie spolu s podrobným popisom jednotlivých vrstiev, niekedy doprevádzaným ukážkami. V podkapitole venovanej prezentačnej vrstve aplikácie sa zameranie týka na grafického užívateľského rozhrania, ktoré popisuje z hľadiska štruktúry, metód návrhu, využitých koncepcií, prvkov a všetko to dopĺňa obrázkami vzhľadu kľúčových častí aplikácie. Pozornosť sa tiež venuje lokalizácii aplikácie do iných jazykov a odchytávaniu výnimiek generovaných aplikáciou. V závere kapitoly sú priblížené spôsoby testovania aplikácie a zhodnotená jeho úspešnosť a prínos pri ošetrovaní chýb.

4.1 Trojvrstvová architektúra

Aplikácia je vytvorená na základe modelu trojvrstvovej architektúry, čo ju robí ľahko rozšíriteľnou, a taktiež ľahko udržiavateľnou. Každá vrstva je nezávislá a z pohľadu zdrojových kódov ju tvorí samostatný projekt. Jednotlivé projekty sú zreťazené referenciami, čo s použitím vzájomných menných priestorov dovoľuje vyšším vrstvám využívať súčasti nižších vrstiev.

V prípade systému pre evidenciu spotreby energií je konkrétne rozdelenie vrstiev a ich obsah nasledovný:

1. **Prezentačná vrstva** – stojí najvyššie, komunikuje a interaguje s užívateľom prostredníctvom GUI, tvorená formulármi, ošetruje chyby a výnimky a upozorňuje na ne.



Obrázok 4.1: Trojvrstvová architektúra. Obrázok prevzatý z [6].

2. **Aplikačná vrstva** – obsahuje celú aplikačnú logiku aplikácie, triedy s metódami pre výpočty a manipulácie, triedy pre vytvorenie objektov s dátami užívateľa, triedy a metódy pre spojenie s dátovou vrstvou.
3. **Dátová vrstva** – trieda s napojením na databázu prostredníctvom LINQ, trieda pre pracovanie s XML.

4.2 Dátová vrstva a dátové zdroje

Úlohou dátovej vrstvy je zabezpečiť priame pripojenie na dátové zdroje, z ktorých aplikácia čerpá. V prípade systému pre evidenciu spotreby energií sa jedná o dva zdroje.

Prvým, hlavným zdrojom dát, je databáza. Prevedením diagramu ER na relačný model, sa získa schéma databáze obsahujúca navyše aj cudzie kľúče a vzťahové množiny, je samotná databáza vytvorená v **SQL Server 2008 Express**. Tvorba databázy je možná priamo v použitom vývojovom prostredí. Po vytvorení jednotlivých entít sú im, okrem vytvorenia atribútov, nastavené rôzne vlastnosti, počnúc od unikátnosťou a primárnymi kľúčmi. Pre identifikáty záznamov entitných množín je, vo väčšine prípadov, využité automatické generovanie hodnoty atribútu pomocou inkrementácie. Niektorým atribútom je výhodné nastaviť prednastavené hodnoty, tie budú použité v prípade, že pri vytváraní záznamu nie je špecifikovaná hodnota konkrétneho atribútu. Využiteľné je to hlavne pre atribúty, ktoré nemusia využívať všetky typy energií, a teda bývajú pri vytváraní záznamu často zamlčané. Pre čo najefektívnejšie uloženie záznamov, rešpektujúce však najmä pres-

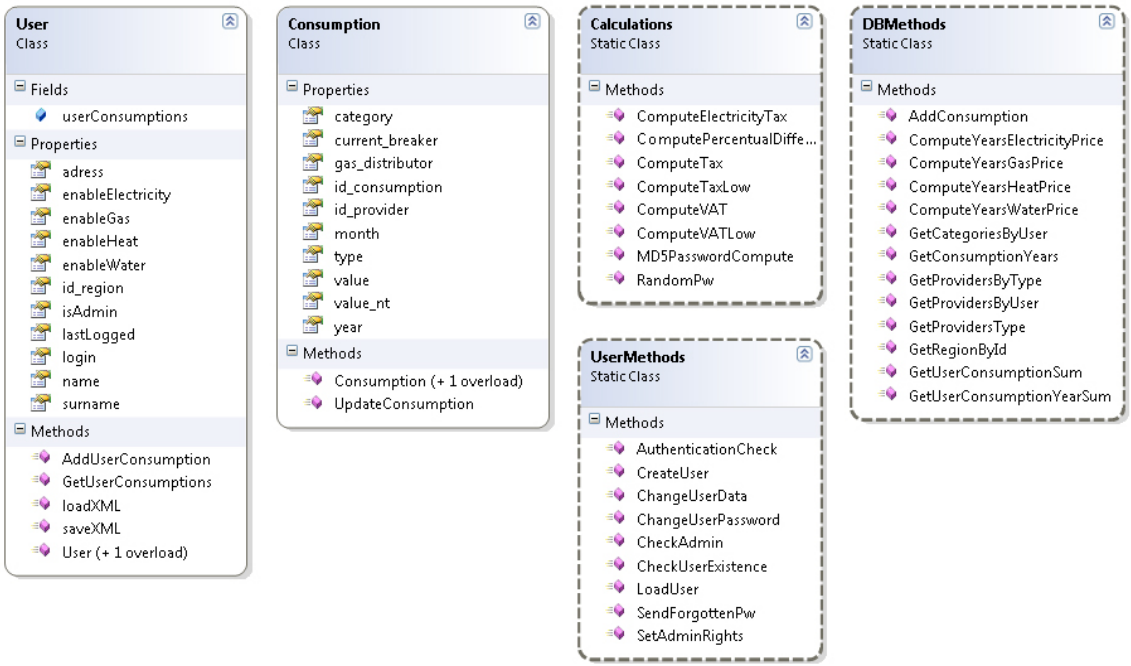
nosť uloženia, vzhľadom na to, že veľká časť atribútov je využívaná a dosadzovaná do vzorcov pri výpočtoch, je dôležité zvoliť vhodné dátové typy jednotlivých atribútov. Pre celé čísla je volený typ *int*, čísla s desatinnou časťou ako ceny a spotreby sú vždy typu *money*, respektíve *decimal*, s presnosťou na 4 desatinné miesta. Tieto typy sú dostačujúce pre implementáciu desatinných čísel, typ *real* by mohol spôsobiť chyby v presnosti hodnôt a zaberal by väčšie množstvo pamäte. Pri pravdivostných hodnotách sa využíva typ *bit* a textové reťazce sú v závislosti od ich charakteru typu *nchar* s daným počtom znakov alebo typu *nvarchar* s premenlivým počtom znakov. Tieto typy tiež podporujú uloženie znakov s diakritikou. V databáze sú taktiež implementované **parametrizované uložené procedúry**, pomocou ktorých sa získavajú a vkladajú dáta do databázy. Dáta špecifikujú na základe vstupných parametrov procedúr následne využitých v príkaze *SELECT*. Uložené procedúry sú vhodným riešením aj z hľadiska zvýšenia bezpečnosti, a taktiež z praktických dôvodov. V databáze nie sú z dôvodu bezpečnosti, uložené priamo heslá užívateľov, ale len reťazce po ich hashovaní pomocou *MD5* algoritmu. Pri kontrole hesiel sa teda porovnávajú reťazce s *MD5* hodnotami hesiel, nie samotné heslá.

Druhým dátovým zdrojom je XML, využívané pre importovanie a exportovanie záznamov databázy o spotrebách užívateľa. Súbory *.xml* sú pred importovaním najskôr nahrané do priečinka *users* pod názvom prihlasovacieho mena užívateľa, a taktiež pri exporte sú v spomínanom priečinku najskôr vytvárané.

Dátovú vrstvu aplikácie predstavuje projekt *DataLayer* obsahujúci dve triedy. O pripojenie na databázu sa stará generovaná trieda *LINQ* (Language Integrated Query), ktorá mapuje do aplikácie jednotlivé entity a uložené procedúry pre ich rýchle a efektívne použitie. Po vytvorení triedy LINQ je možné metódou drag-and-drop priamo zo zobrazenia databázy vybrať entity a uložené procedúry, ktoré majú byť namapované. Kód vygenerovaný pomocou LINQ pozostáva z časti predlohy pre schému (*Database.dbml.layout*) a tzv. „designer“ časti (*Database.designer.cs*), obsahujúcej samotný kód pripojenia a využívania databázy. LINQ je veľmi vhodným riešením z hľadiska vyvarovania sa chýb a optimalizácie, spojenie s databázou je pri jeho použití bezpečnejšie a veľmi efektívne. S XML pracuje trieda *XML.cs*, pomocou serializácie a deserializácie záznamov o odberoch. Vo svojich dvoch metódach implementuje ukladanie časti objektu užívateľa do XML a jej načítavanie z, už predtým vytvoreného, XML súboru.

4.3 Aplikačná vrstva

Projekt *BusinessLayer* je aplikačnou vrstvou systému. Je tvorený triedami *User.cs* a *Consumption.cs*, umožnujúcimi vytvorenie objektu užívateľa spolu s jeho spotrebami pre potreby manipulácie s nimi a statickými triedami implementujúcimi správanie systému, výpočty a naviazanie na dátovú vrstvu. Volanie uložených procedúr prebieha cez vytvorenie dátového kontextu triedy LINQ a jeho následné využívanie. Objekt užívateľa, spolu s jeho nastaveniami a zoznamom spotrieb, sa po prihlásení do systému načíta z databázy a následne uloží do poľa *Session*, pre rýchlu manipuláciu s často vyžadovanými dátami a obmedzenie neustálej komunikácie s databázou za účelom týchto údajov. Platnosť *Session* je nastavená explicitne v konfiguračnom súbore prezentačnej vrstvy na 60 minút. Po uplynutí tejto doby od prihlásenia sa záznamy zneplatnia a užívateľ bude znovu vyzvaný k prihláseniu.



Obrázok 4.2: Diagram tried (class diagram) aplikačnej vrstvy.

Jednou zo statických tried je trieda *UserMethods.cs* implementujúca statické metódy pre prácu s užívateľskými údajmi. Metódy implementujú vytváranie nového užívateľa, zmenu užívateľských nastavení, kontroly práv a existencie užívateľa, načítanie užívateľského profilu

do objektu aplikačnej vrstvy a zasielanie zabudnutého hesla na e-mailovú adresu. Všetky z týchto metód, pri získavaní a zmene dát, pracujú s dátovým kontextom LINQ.

```
public static bool AuthenticationCheck(string UserName, string Password)
{
    DatabaseDataContext db = new DatabaseDataContext();
    GetPasswordResult result;

    try
    {
        result = db.GetPassword(UserName).First();
    }
    catch
    {
        return false;
    }

    // Check password validity through MD5
    if (Calculations.MD5PasswordCompute(Password).Equals(result.password.ToString()))
        return true;
    else
        return false;
}
```

Obrázok 4.3: Ukážka metódy pre autentifikáciu užívateľa.

Ďalšou, implementujúcou výhradne statické metódy, je trieda *DBMethods.cs*, pokrývajúca implementačne všetky metódy, ktoré pracujú s databázovým prístupom, okrem metód zameraných na užívateľa a jeho profil. Väčšinou sa jedná o metódy vyťahujúce dáta z databázy za účelom ich zobrazenia a zaradenia do kontextu a metódy vkladajúce záznamy do databázy. Nachádzajú sa tu však aj rozsiahle metódy na vypočítavanie cien spotreby pre každú energiu. Každá z týchto metód slúži, tak pri výpočte ročných cien za spotreby, ako aj pri výpočte cien alternatívnych riešení na stránke porovnania alternatívnych dodávateľov energie. Použitie sa posudzuje na základe prichádzajúcich parametrov metódy. Pre výpočet cien elektriny a plynu je nutné využitie aj regulovaných cien, spolu s cenami, ktoré si účtujú dodávatelia energie. Metódy pre výpočet cien si vyberajú správne a aktuálne údaje o cenách v danom období, a v prípade ich nedostupnosti korektne hlásia ich absenciu. Vybraný záznam o cene vždy zodpovedá kategorizácii spotreby a platnosť ceny obdobiu, v ktorom bola energia spotrebovaná.

Poslednou je trieda *Calculations.cs*, taktiež implementujúca iba statické metódy. Tieto metódy však slúžia ako pomocné výpočtové metódy a riešia prepočítavanie hodnôt, výpočty daní k cenám, percentuálnych rozdielov cien, a tiež sa tu nachádza metóda pre

vygenerovanie náhodného hesla, volaná pri aktivovaní formulára o zabudnutom hesle, spolu s metódou na výpočet hashovanej hodnoty hesla pomocou *MD5* algoritmu. Metódy v tejto triede nepracujú s dátovým kontextom.

4.4 Prezentačná vrstva

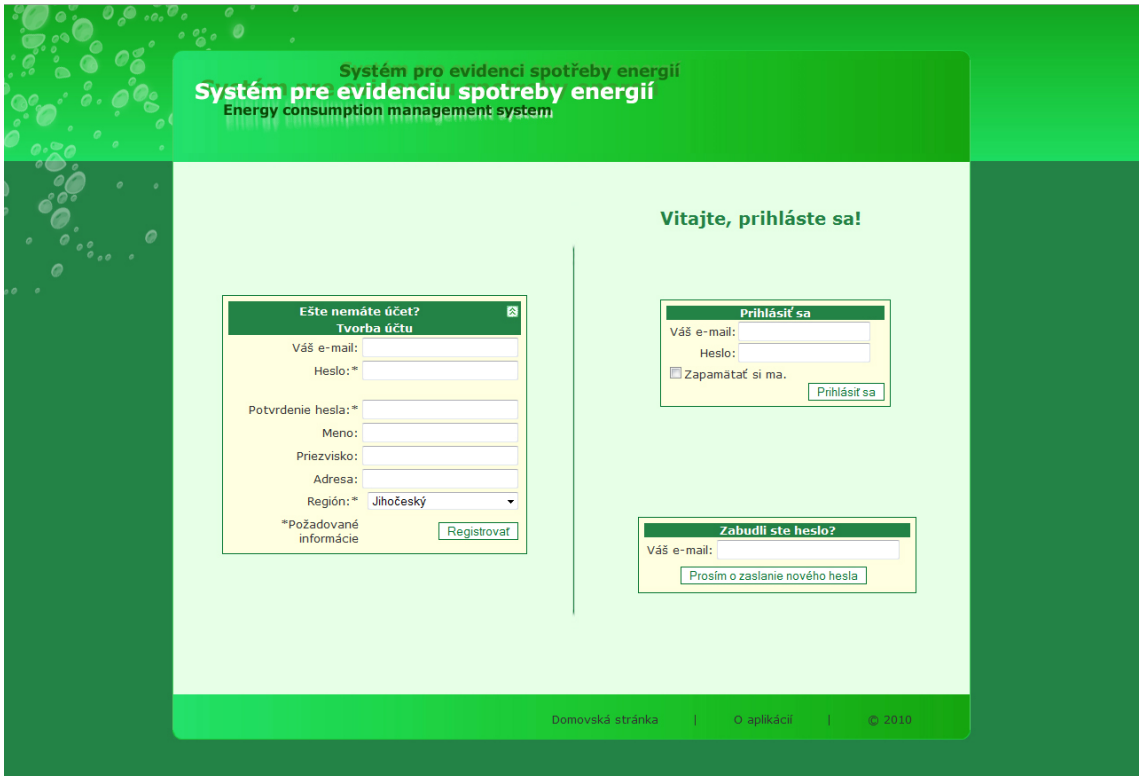
Prezentačnou vrstvou systému je samotný projekt webovej prezentácie. Jej úlohou je komunikácia s užívateľom a riešenie výnimiek, ktoré sem môžu postupne prichádzať až z dátovej vrstvy.

Úvodným webovým formulárom je stránka *index.aspx*. Jediným formulárom administratívnej časti je *admin.aspx*, v ktorom sa nachádzajú prostriedky ku všetkým dostupným administračným úkonom. Hlavným formulárom po prihlásení užívateľa bez administrátorských oprávnení je *home.aspx*, poskytujúci zhrňujúci pohľad na aktivity užívateľa. Stránky jednotlivých energií majú prefix v podobe názvu energie, ktorej sú venované (*electricity.aspx*, *electricity_alternatives.aspx*, atď.).

Formuláre prezentačnej vrstvy používajú kód v oddelenom súbore, to znamená, že každému formulárovému súboru zodpovedá súbor rovnakého názvu typu *.cs*, implementujúci správanie formulára a vysporiadanie sa s udalosťami, ktoré sa počas interakcie s ním môžu objaviť. V metóde pomenovanej *Page_Load* každej stránky, vyžadujúcej prihlásenie, sa nachádza kód, ktorý implementuje kontrolu stavu užívateľovho prihlásenia, prípadne jeho práva pre vstup do administračnej časti. Metóda *Page_Load* je volaná udalosťou načítania stránky a predstavuje jedno z viacerých štádií životného cyklu stránky ASP.NET. V prezentačnej vrstve sa nachádza taktiež súbor *XSLTTransform.xslt*, ktorý je schémou pre prevedenie XML súboru obsahujúceho údaje o spotrebách užívateľa na jednoduchú webovú stránku s výpisom týchto spotrieb. Samotnú transformáciu v rámci systému volá formulár *transformed.aspx*, prístupný po importovaní spotrieb do systému užívateľom a zobrazení tlačítka pre zobrazenie importovaných spotrieb z formulára *ie.aspx*, slúžiaceho okrem importu taktiež na export spotrieb. Formulár *settings.aspx* umožňuje zmeny užívateľských nastavení a formulár *about.aspx* má čisto informačnú funkciu.

4.4.1 Grafické užívateľské rozhranie

Grafické užívateľské rozhranie (GUI) aplikácie je zobrazením prezentačnej vrstvy aplikácie. Užívateľ s ním komunikuje a interaguje za účelom vykonávania jeho požiadaviek a pokynov. GUI by malo svojim vzhľadom užívateľa zaujať a motivovať ho k práci s aplikáciou, preto musí byť jednoduché a intuitívne. Rozvrhnutie vzhľadu systému je sofistikované, prehľadné a nie je nijako pre užívateľa rušivé. Taktiež hierarchické rozdelenie a umiestnenie jednotlivých ponúk menu je veľmi intuitívne.



Obrázok 4.4: Úvodná stránka webovej aplikácie.

Hlavné rozvrhnutie aplikácie je definované pomocou *Master pages*, stránkami slúžiacimi ako predloha pre iné stránky. Použité sú dve, v sebe zanorené, predlohy. Prvá definuje hlavné rozdelenie vzhľadu aplikácie na viditeľné oddelenie hlavičky, obsahu a zapätia stránky spolu s ich veľkosťami a malým menu v zápätí. Menu obsahuje odkaz na domovskú stránku a stránku s informáciami o aplikácii. Druhá predloha dopĺňa prvú o menu v hlavičke stránky, obsahujúce odkazy na stránky importovania a exportovania záznamov, nastavenia užíva-

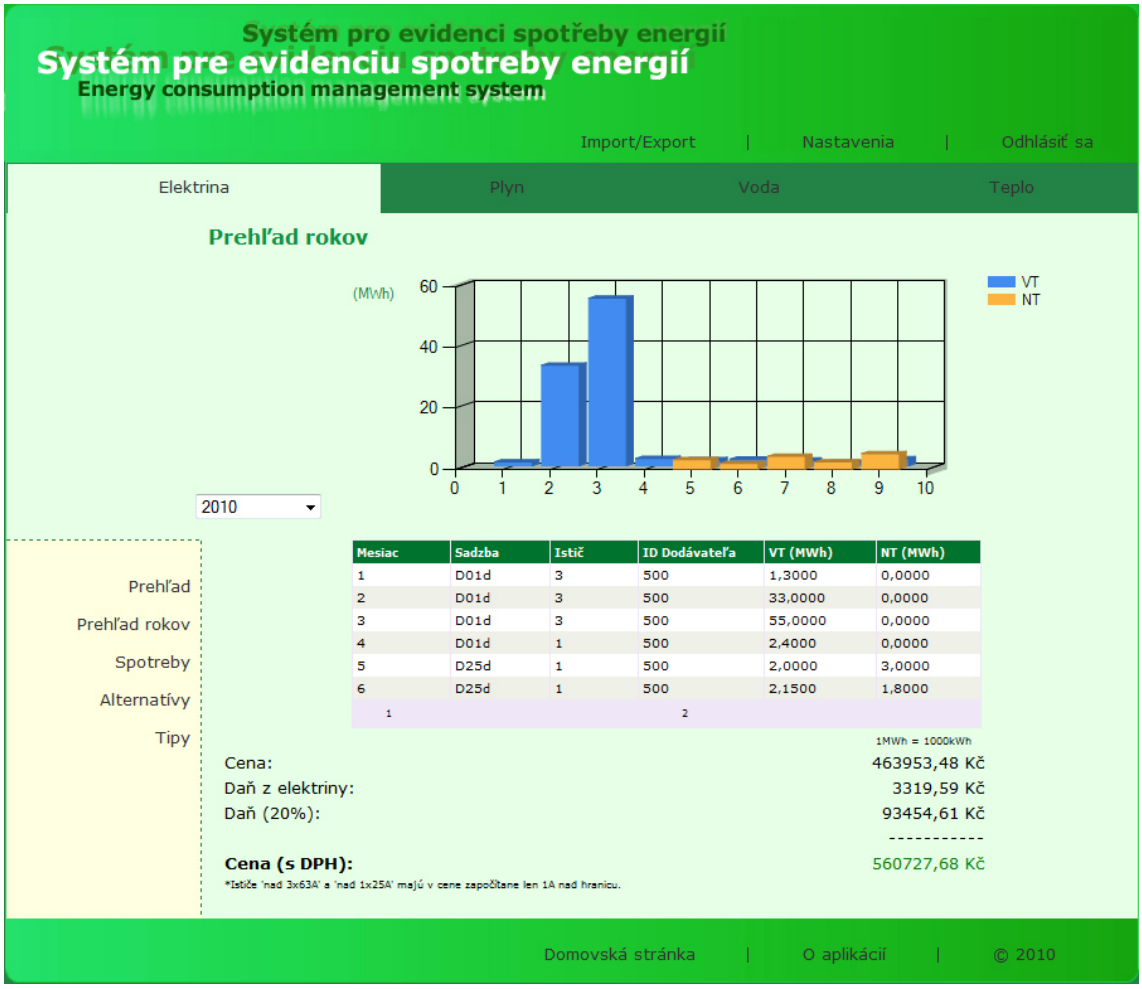
teľského profilu a odkaz pre odhlásenie z aplikácie. Taktiež pôvodnú predlohu dopĺňa o menu energií, ktoré obsahuje odkazy na monitorovacie stránky jednotlivých energií. Stránky jednotlivých energií obsahujú potom ešte sekundárne menu implementované v podobe *Web User Control*, pre navigáciu stránok pracujúcich s konkrétnou energiou.

Obrázok 4.5: Ukážka administračnej časti aplikácie.

Definícia vzhľadu GUI je založená na princípe vzhľadových tém, tzv. *Themes*, ktoré podporujú rýchle zmeny vzhľadu aj za chodu aplikácie. Vzhľadové témy sa nachádzajú v špecializovanom priečinku aplikácie *App_Themes* a obsahujú vlastné obrázky, súbory kaskádových štýlov a súbor *.skin*, umožňujúci centralizované zmeny vzhľadu serverových ovládacích prvkov. Pri definovaní vzhľadu je nutné klásť dôraz na konštrukcie, ktoré nie sú

pre internetové prehliadače problematické.

Pre spestrenie a zatraktívnenie vzhľadu aplikácie je použitý balík *AJAX Control Toolkit*, obsahujúci množstvo prídavných prvkov podporujúcich technológiu AJAX. Hlavné využitie si našiel v administrátorskej časti prvok *TabContainer*, ktorý je vlastne súborom záložiek s asynchronným dotazovaním. Už na úvodnej stránke je badateľný prvok *CollapsiblePanel*, ktorý umožňuje animované vysúvanie pridaného obsahu stránky. Stránka nastavení je založená na prvku *Accordion*, ponúkajúcom vkusné triedenie jednotlivých užívateľských nastavení.



Obrázok 4.6: Ukážka stránky pre zobrazenie spotreby za zvolený rok.

Aplikácia obsahuje taktiež dynamicky generované grafy, ktoré sú doležitým prvkom

pre ilustratívne a pútavé zobrazenie užívateľových spotrieb energie. Použitý je ovládací prvok *Microsoft Chart*. Zdrojom dát pre grafy je dátový kontext LINQ, na ktorý sú priamo naviazané.

4.4.2 Kontrola vstupných dát a odchytávanie výnimiek

Kontrola vstupných dát má za úlohu zamedziť vzniku nekonzistentných záznamov v databáze. Pri každom mieste zadávania dát do systému, ktoré reprezentujú najmä textové polia, je kontrolované, či sú dáta v požadovanom formáte a rozsahu. Táto kontrola je vykonávaná pomocou **regulárnych výrazov**. V serverových ovládacích prvkoch *ListView*, ktoré sú hlavným prvkom spotrestredkujúcim manipuláciu s dátami v databáze, sú jednotlivé vstupy kontrolované priamo dátovým kontextom a metódami pre obslúženie udalostí vkladania a úprav, ktoré sú modifikované pre obslúženie výnimiek vyskytujúcich sa pri vykonávaní operácie. Prvky *ListView* sú naviazané na databázové tabuľky priamo pomocou dátového zdroja LINQ.

Výnimky sú mechanizmom pre vyvolávanie a detekovanie rôznych chybových stavov aplikácie počas vykonávania funkcií. Pomocou výnimiek je možné tieto stavy korektne ošetriť, bez narušenia funkčnosti a chodu aplikácie. Mechanizmus je založený na využívaní blokov *try-catch-finally*. V *try* bloku je umiestnený kód, ktorý je potenciálnym zdrojom výnimky, blok *catch* túto výnimku zachytáva a rieši jej spracovanie. Blok *finally* je nepovinný a jeho kód sa vykoná v prípade, že bola zachytená výnimka, ale rovnako aj v prípade, že nedošlo k výnimke. Dôležitým je príkaz *throw*, vyvolávajúci výnimku, respektíve zaisťuje jej zachytenia v mieste, odkiaľ bola funkcia volaná, a takto postupne sa výnimka môže dostať k určitému miestu, kde je s ňou naložené v rámci udržania správneho chodu aplikácie. Miestom pre ošetrovanie výnimiek je v systéme prezentačná vrstva, výnimky z nižších vrstiev sa do nej dostávajú ich postupným preposielaním pomocou príkazu *throw*.

Zachytávanie a následné ošetrenie výnimiek je dôležité hlavne pri práci s databázou a dátovým kontextom. Dátová vrstva vzhľadom na napojenie na dátové zdroje, ktoré môže kedykoľvek zlyhať, prípadne sa prerušiť, je najväčším zdrojom výnimiek. Výskytu výnimiek spôsobených nesprávnosťou dátových typov a ich použitia je možné zamedziť zaistením konzistencie dát uložených v dátovom zdroji.

4.5 Lokalizácia

Napriek tomu, že aplikácia svojim zadaním špecifikuje jej zameranie na Českú republiku, jej potenciálni užívatelia môžu uprednostňovať iný jazyk ako je čeština. Typickým príkladom sú príslušníci iných národov žijúci v Českej republike. Ďalšou možnosťou je postupné možné rozšírenie aplikácie aj do iných štátov. Pri vytváraní viacjazyčnej webovej aplikácie sú jednotlivé jazykové adaptácie ťažené zo zdroja jednej stránky, ktorý si vyžaduje omnoho menej údržby v porovnaní s riešením v podobe oddelenej stránky na oddelenom sídle. Lokalizácia spočíva v automatickom využívaní súborov *.resx*, ktoré vyplýva zo špecifickej menovacej dohody a sídli v špecializovaných adresároch v rámci aplikácie. Špecializovaným adresárom obsahujúcim lokalizačné súbory je *App_LocalResources*. Táto automatizovaná podpora dovoľuje pridávať zdrojové súbory jednotlivých lokalizácií bez akýchkoľvek kompilačných krokov alebo satelitných assemblies, jednoduchým pridaním ďalšieho *.resx* súboru pre konkrétnu jazykovú adaptáciu a formulár. Vo formulári je potom pri každom lokalizovanom texte uvedený identifikátor, ktorý mu prideľuje hodnotu v lokalizačnom súbore.

Jazyk lokalizácie je automaticky načítaný podľa regionálnych nastavení jazykov na operačnom systéme, respektíve v internetovom prehliadači, na ktorom je aplikácia spúšťaná. **Visual Studio 2008** umožňuje automaticky vygenerovať súbor *.resx* spolu s naviazaním textu na jednotlivé identifikátory, v niektorých špecifických prípadoch je však stále nutné niektoré referencie doplniť. Pre každý formulár je nutné vygenerovať toľko súborov *.resx* a preložiť ich, v koľkých jazykoch má byť formulár dostupný.

Aplikácia je dostupná v českej, slovenskej a anglickej jazykovej verzii, pričom prednastaveným hlavným jazykom je slovenčina.

4.6 Testovanie

Testovanie aplikácie prebiehalo aj priebežne počas implementácie, vždy po ukončení implementácie väčšieho celku. Principiálne išlo o zadávanie nesprávnych a hraničných hodnôt systému s očakávaním korektnej reakcie, prípadne chybového výpisu aplikácie.

V administračnej časti spočívalo testovanie v pridávaní, upravovaní a odstraňovaní hodnôt jednotlivých tabuliek odrážajúcich databázu aplikácie. Databáza sa nesmela dostať do nekonzistentného stavu a ukladať chybne zadané hodnoty. Napríklad, pri odstraňovaní položky z databázy je dôležité správanie a reakcia v prípade, že položka databázy je cudzím

kľúčom do inej tabuľky a existujú pre neho záznamy.

V užívateľskej časti bolo testovanie zamerané najmä na zle zadané, či hraničné vstupné hodnoty a hodnoty null objavujúce sa v rôznych častiach systému v rôznych kontextoch. Taktiež bolo testované vypršanie platnosti poľa Session, ktoré uchováva údaje o prihlásenom užívateľovi. Pri postupnom ladení systému boli všetky známe chyby a výnimky ošetrené, systém ich vždy doprevádza výstižným chybovým hlásením a aplikácia sa udržiava v korektnom stave. Dôležitá je odpoveď systému na požiadavku užívateľa o poskytnutie údajov, ktoré v databáze chýbajú. Táto situácia sa vyskytuje najmä pri výpočte cien za spotreby a porovnávaní alternatív, ak chýbajú cenové údaje pre niektorého z dodávateľov v danom časovom období. V užívateľskej časti bola taktiež testovaná korektnosť práce s XML pri importovaní a exportovaní.

Všetky nájdené chyby a chybné hodnoty boli vhodne ošetrené a sú doprevádzané výstižným hlásením, popisujúcim príčinu pre užívateľa. Aplikácia bola taktiež testovaná v internetových prehliadačoch Internet Explorer, Mozilla Firefox a Google Chrome. Zobrazenie v každom z prehliadačov je korektné a v základoch sa neodlišuje. Taktiež funkčnosť komponentov využitých v systéme a jednotlivých ovládacích prvkov nie je nijako obmedzená v závislosti na internetovom prehliadači.

Kapitola 5

Možnosti rozšírenia

Po splnení požiadaviek na systém a jeho funkčnosť a následnom doladení a ošetrení vo všetkých aspektoch je možné sa zamerať na zlepšenia, ktoré by boli systému prospešné a posunuli by ho na ešte vyššiu úroveň práce s ním, pokrytia dát a energií.

Už od ranných prác na návrhu aplikácie bolo cieľom vytvoriť aplikáciu, ktorá bude ľahko modifikovateľná a rozšíriteľná v čo najviac ohľadoch. Tento cieľ pomáha z veľkej časti realizovať architektúra aplikácie, ktorá svojou trojvrstvovou koncepciou dovoľuje rýchle zmeny aj v merítku kompletných vrstiev. Samozrejme, pre budúce rozšírenia je uspôsobená aj databáza, ktorá už aj prostredníctvom administračnej časti systému dovoľuje pridávanie hodnôt, ktoré zaisťujú rozšírenie. Rozšíriteľnosť začína už na hlavnej premennej tohto systému v podaní typov energie. Databáza sa pridaniu nových energií nebráni, a po menších úpravách a vytvorení formulárov pre ňu je možné ju plnohodnotne používať.

Design aplikácie nie je nijako pevne viazaný a umožňuje jednoduché pridanie ďalších prvkov a položiek bez fatálnych modifikácií. Taktiež kompletná zmena designu je ľahko realizovateľná modifikáciami prezentačnej vrstvy. Design aplikácie je založený na princípe vzhľadových tém, pomocou ktorých je ľahké okamžite meniť vzhľad aplikácie aj za jej chodu užívateľským nastavením. Aplikácia obsahuje však momentálne iba jednu vzhľadovú tému, takisto ovládanie zmeny vzhľadu by bolo nutné doimplementovať. Pridaním algoritmov do aplikačnej vrstvy a ich následným naviazaním na prvky prezentačnej vrstvy je taktiež možné zakomponovať novú funkcionalitu. Vhodným by sa javilo, napríklad, archivovanie cien za minulé spotreby a systém rôznych upozornení, napríklad, na rôzne nové cenové ponuky dodávateľov, či iné zmeny cien. Taktiež rozšírenie v podobe ďalších metód pridávania spotrieb užívateľa aj externe, bez prístupu do systému, by bolo veľmi vhodným riešením.

Aplikácia podobného typu má tiež vysoký medzinárodný potenciál, a vzhľadom na existujúcu lokalizáciu aplikácie je možné kedykoľvek rýchlo pridať ďalšie nové jazyky. V prípade, že by aplikácia mala vysoké ciele, nebolo by problémom podľa jej vzoru implementovať podobné systémy pre iné krajiny, prípadne pridať možnosti iných krajín do aplikácie a zvýšiť tak jej maximálnu využiteľnosť.

Kapitola 6

Záver

Cieľom tejto bakalárskej práce bolo popísať implementáciu webovej aplikácie pre evidenciu a sledovanie spotreby energií v domácnostiach a zobrazovať užívateľom zadané údaje o spotrebe vo forme grafov a tabuliek. Taktiež mal tento systém ponúkať varianty iných dodávateľov energie k zvoleným. Aplikácia mala umožňovať sledovanie minimálne elektriny, plynu a vody.

Aplikácia bola implementovaná so snahou predčiť existujúce a priniesť nové komplexné a efektívne riešenie. Funkcionalita bola v niektorých ohľadoch doplnená, hlavným doplnkom bolo pridanie monitorovania a evidencie spotreby tepla. Aplikácia dokáže spravovať centrálne dáta využiteľné pre všetkých užívateľov, a takisto umožňuje každému z nich spravovať a archivovať vlastné dáta. Pri jednotlivých energiách vypočítava ceny za ich spotreby, ilustruje časový priebeh spotreby grafmi a cenovo porovnáva zvolené dodávateľské riešenie s inými. Taktiež je možné užívateľské dáta pridávať importovaním, prípadne ich zálohovať a prenášať exportovaním zo systému.

Pri návrhu aplikácie a vrstvovej architektúry som využil znalosti získáné zo softvérového inžinierstva. Pri implementácii som taktiež využil predošlé skúsenosti s koncepciou objektovo-orientovaného programovania a vedomosti z databázových systémov.

Vypracovaním tejto práce som sa podrobne oboznámil s technológiou ASP.NET a naučil som sa ju efektívne využívať spolu so všetkými výhodami a možnosťami, ktoré .NET Framework ponúka. Prínosom do budúcnosti pre mňa bude získaná zručnosť práce s jazykom C#, zdokonalenie mojich schopností návrhu aplikácie, tvorby a práce s databázou.

Literatúra

[1] Jak se skládá cena elektřiny? - Skupina ČEZ [online]. Dostupné na: http://www.cez.cz/cs/pro-zakazniky/faktury-a-platby/jak-se-sklada-cena-elektriny.html, [cit. 2010-05-14].

[2] Overview of the .NET Framework [online]. Dostupné na: http://msdn.microsoft.com/en-us/library/a4t23ktk%28v=VS.100%29.aspx, [cit. 2010-05-14].

[3] Visual Studio 2008 Professional [online]. Dostupné na: http://www.microsoft.com/slovakia/msdn/produkty/vstudio/Professional/default.mspx, [cit. 2010-05-14].

[4] ms973842.dotnet_movingjavaapps_01%28en-us,MSDN.10%29.gif (Obrázok GIF) [online]. Dostupné na: http://i.msdn.microsoft.com/ms973842.dotnet_movingjavaapps_01%28en-us,MSDN.10%29.gif, Posledná modifikácia: 1.5.2007 [cit. 2010-05-14].

[5] Introduction to OMG's Unified Modeling Language [online]. Dostupné na: http://www.omg.org/gettingstarted/what_is_uml.htm, Posledná modifikácia: 18.6.2009 [cit. 2010-05-14].

[6] 3tier_2.jpg (Obrázok JPEG) [online]. Dostupné na: http://weblogs.asp.net/blogs/fredriknormen/WindowsLiveWriter/UsingWebServicesina3tierarchitecture_134F6/3tier_2.jpg, Posledná modifikácia: 5.11.2008 [cit. 2010-05-14].

[7] Evjen, B.: *ASP.NET 3.5 v jazycích C# a Visual Basic : programujeme profesionálně*. Computer Press, 2009, iSBN 978-80-251-2069-9.

[8] Holzner, S.: *XSLT Příručka internetového vývojáře*. Computer Press, 2002, iSBN 80-7226-600-4.

[9] Kanisová, H.: *UML srozumitelně*. Computer Press, 2006, iSBN 80-251-1083-4.

[10] Kolář, D.: *Principy programovacích jazyků a OOP :IPP*. Fakulta informačních technologií VUT v Brně, 2008.

[11] Robinson, S.: *C# : programujeme profesionálně*. Computer Press, 2003, iSBN 80-251-0085-5.

[12] Staníček, P.: *CSS Kaskádové styly - Kompletní průvodce*. Computer Press, 2003, iSBN 80-7226-872-4.

[13] Walters, R.: *Mistrovství v Microsoft SQL Server 2008*. Computer Press, 2009, iSBN 978-80-251-2329-4.

Príloha A

Obsah CD

Priložené CD obsahuje:

- kompilovateľné zdrojové kódy implementácie aplikácie,
- predkompilovaná aplikácia pripravená k nasadeniu,
- skript pre vytvorenie ukážkovo predvyplnenej databázy,
- projektová dokumentácia aplikácie,
- jednoduchý manuál ovládania aplikácie vo formáte PDF,
- tento dokument vo formáte PDF,
- kompilovateľné zdrojové kódy tohoto dokumentu spolu s obrázkami.