Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Tvorba podpůrné webové aplikace pro hotelový systém s využitím ASP.NET a MVC

Patrik Bláha

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Patrik Bláha

Informatika

Název práce

Tvorba podpůrné webové aplikace pro hotelový systém s využitím ASP.NET a MVC

Název anglicky

Development of support web application for hotel IS using ASP.NET and MVC

Cíle práce

Cílem práce je navrhnout a implementovat webovou aplikaci sloužící jako podpůrný prostředek pro správu složitějšího webového IS. Výsledná aplikace bude implementována s využitím technologie Microsoft ASP.NET a MVC 5.

Metodika

Metodika práce je založena na analyticko-syntetickém přístupu. Bude provedena analýza odborných informačních zdrojů a na základě syntézy zjištěných poznatků a požadavků na vytvářený systém bude navržena a implementována webová aplikace sloužící jako podpůrný prostředek při správě rozsáhlého webového IS. Postup návrhu, implementace, testování a nasazení bude popsán a zhodnocen.

Doporučený rozsah práce

35-40 stran

Klíčová slova

webová aplikace, C#, .NET, MVC, webové stránky, třídy, LINQ

Doporučené zdroje informací

ASP.NET MVC | The ASP.NET Site [online]. 2015-09-19. 2015 [cit. 2015-09-19]. Dostupné z: http://www.asp.net/mvc

GALLOWAY, Jon. Professional asp.net mvc 5. 1st edition. Indianapolis, IN: John Wiley and Sons, 2014, pages cm. ISBN 11-187-9475-3.

JOHNSON, Bruce. Professional visual studio 2013. Indianapolis, Indiana: Wrox, 2014, 1 online zdroj (1102 pages). ISBN 978-1-118-83205-9.

MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně. Vyd. 1. Brno: Zoner Press, 2011, 700 s. Encyklopedie Zoner Press. ISBN 978-80-7413-145-5.

Referenční dokumentace jazyka C# [online]. 2015 [cit. 2015-09-21]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/618ayhy6.aspx

---

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 06. 03. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Tvorba podpůrné webové aplikace pro hotelový systém s využitím ASP.NET a MVC" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 14.3.2016 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce panu Ing. Jiří Brožkovi, Ph.D. za ochotnou pomoc a odborné vedení při zpracování práce.

Tvorba podpůrné webové aplikace pro hotelový systém s využitím ASP.NET a MVC

Souhrn

Práce se zabývá vytvořením jednoduché podpůrné aplikace hotelového systému se zaměřením na použití technologií ASP.NET MVC s jazykem C# a poznatků a frameworků získaných při konání praxe. V první části této práce je uvedení do technologií ASP.NET, MVC a použitých frameworků a technologií jako je Bootstrap, LINQ, Entity Framework a další. Použité frameworky značně zefektivňují a usnadňují psaní této podpůrné aplikace. První část je dále zaměřena na průběh kompilace pomocí ASP.NET a je zde vysvětlena struktura aplikace při použití technologie MVC. Druhá část této práce je zaměřena na vytvoření podpůrné aplikace pro hotelový systém. Je zde provedena analýza na základě požadavků na tuto aplikaci. Po této analýze je podrobně popsaný technický postup vytvoření aplikace. V první části vytváření podpůrné aplikace je uvedeno, jak se pomocí Visual Studia vytváří základní projekt a struktura MVC. Je zde vysvětleno propojení aplikace s databázovým systémem za pomocí Entity Frameworku a to při vytváření nové databáze a její struktury a zároveň i propojení s již existující databází. Dále je zde uvedena implementace frameworku pro řízení uživatelských účtů. V poslední části je tvorba aplikace zaměřena na aplikaci stylů a jsou zde uvedeny grafické výstupy aplikace.

Klíčová slova: webová aplikace, C#, .NET, MVC, webové stránky, třídy, LINQ

Development of support web application for hotel IS using ASP.NET and MVC

Summary

This thesis deals with creating a supporting web application for the existing hotel information system and is focused to use ASP.NET and MVC technologies with C# language and knowledge received in my practical experience. In the first part of this thesis is an introduction to the ASP.NET MVC and used technologies and framework such as Bootstrap, LINQ, Entity Framework and more. Used frameworks makes writing this supporting application much easier. The first part of this thesis is also aimed at compiling using ASP.NET and here is explained the structure of applications with using ASP.NET technology. The second part is focused on creating this hotel supporting web application for hotel system. There is an analysis based on the requirements for this application. After this analysis is described the technical process of creating this application in detail. The first of creating this support web application is how to use Visual Studio to create a base web application project and MVC structure. It explains linking application with database system using Entity Framework and creating a new database and its structure as well as creating the connection with a new database. There is indicated the implementation with framework manage the users accounts. The last part is the creation of application focused on the styles of this web application and there are some graphical presentations.

Keywords: web application, C#, .NET, MVC, web pages, classes, LINQ

Obsah

Seznam obrázků

1 Úvod

U rostoucích informačních systémů je potřeba vytvořit podpůrné aplikace, které se starají o jeho správu. S růstem takového informačního systému je potřeba tomuto systému odlehčit, vzhledem k rostoucím datům a složitosti zpracovávaných informací.

Zároveň s tím v informačním systému roste počet uživatelů a s tím jsou kladeny požadavky na správu těchto uživatelů, to vede k urychlení zpracovávání požadavků na podporu informačního systému.

Jedním z rostoucích informačních systémů je i hotelový systém. S rostoucím počtem zákazníků rostou požadavky na výkon, tedy rychlost tohoto informačního systému, podporu uživatelů a monitoring celkové funkcionality hotelového informačního systému.

Tato práce je zaměřena na vytvoření podpůrné aplikace hotelového informačního systému pro monitoring funkcionality informačního systému a zajištění potřebné podpory. Práce je nejprve zaměřena na bližší popis a uvedení programovacího jazyka C#, uvedení návrhových vzorů a použitých frameworků. V druhé části je uvedena doprovodná aplikace.

2 Cíl práce a metodika

S rostoucím vývojem hotelového systému je potřeba vytvořit podpůrnou aplikaci, která bude mít za úkol správu tohoto informačního systému spolu s implementovanou správou a řízením uživatelských účtů v této podpůrné aplikaci. K dosažení tohoto úkolu bude zapotřebí čtyř cílů, které postupně sestaví celou aplikaci.

Prvním cílem bude popis a seznámení se s frameworky umožňující implementaci a s technickými prostředky posléze použitých při tvorbě této aplikace. Druhým cílem bude následovat interview s vlastníkem hotelového systému k ustanovení základních požadavků. Následně bude provedena analýza požadavků doprovázená technickou analýzou.

Následovat bude třetí cíl, který bude zastřešovat implementaci podpůrné hotelové aplikace. Zde bude podrobně uveden postup implementace analyzovaných požadavků z druhého cíle.
Posledním, tedy čtvrtým cílem bude vysvětlení možných postupů testování a vysvětleno nasazení aplikace na webový server.

V teoretické části bude provedena analýza odborných informačních zdrojů souvisejících s frameworky a technologickými postupy. Na základě syntézy analýzy odborných informačních zdrojů bude proveden sběr požadavků od vlastníka hotelového systému a pomocí analýzy a technické analýzy bude navržena implementace.
Praktickou částí bude následovat implementace, kde bude uveden podrobný postup včetně ukázek. Na základě výsledků implementace bude vyhodnoceno testování aplikace a následné nasazení na webový server.

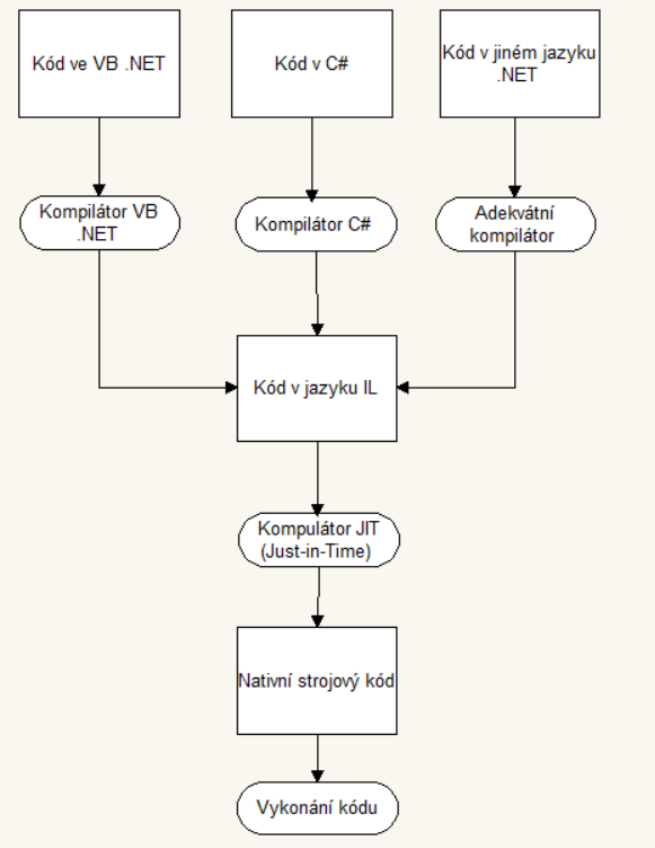
3 Přehled řešené problematiky

3.1 ASP.NET

Již od prvního vydání se ASP.NET výrazně lišilo od jiných produktů firmy Microsoft i konkurenčních platforem. Je zde sedm pilířů, které vystihují odlišnost od tehdejších platforem.

První pilíř vystihuje integraci ASP s .NET frameworkem. Integrace .NET frameworku umožňuje použití velkého množství funkčních částí zahrnujících desetitisíce typů (4, s.25). Desetitisíce těchto tříd jsou uspořádány do hierarchického kontejneru, kterému se říká jmenný prostor, v anglickém názvu *namespace* (4, s. 26). Umožňuje využívat velké množství již správně a efektivně sepsaného kódu. Právě díky tomu není nutné většinu algoritmů v aplikaci psát znovu dokola. Jako příklad bude uvedena třída klienta, tedy rozhraní, pomocí kterého je možné se pomocí počítačové sítě propojit s jinou aplikací, spuštěné na třeba druhém konci světa. Pokud není použit .NET framework a je potřeba toto propojení uskutečnit, bylo by nutné, po vyloučení frameworků konkurenčních firem, sepsat vlastní aplikaci, která by uskutečňovala navázání spojení od samého základu. Tedy validovat vstupní informace a přeposílat dotazy paket po paketu. S použitím již zmíněného frameworku se pouze použije již implementované rozhraní, vybere se adresa cílového počítače, nastaví se preferované atributy (např. autorizace) a spojení se uskuteční. Ve výsledku spojení uskuteční knihovna .NET frameworku jejíž výstupem bude předem zpracovaný výstup. Bez jakýchkoli pochyb tedy lze říci, že se jednalo skutečně o převratný Framework, který programátorům velmi ulehčil práci.

Druhý pilíř se zabývá kompilací samotného kódu. Popisuje dvě etapy, kterými prochází proces celé kompilace. V první etapě se napsaný kód (v našem případě v jazyce C#) zkompiluje do přechodného jazyka Microsoft Intermediate Language (MSIL, nebo také IL). První etapou kompilace tedy vzniknou soubory IL, kterým se říká *asembly*. Díky tomu je možné stejnou aplikaci psát ve více možných jazycích. Druhá etapa nastává před tím, než se kód doopravdy vykoná (4, s. 26). Celý proces obou kompilací je znázorněn v následujícím obrázku (Obrázek 1).

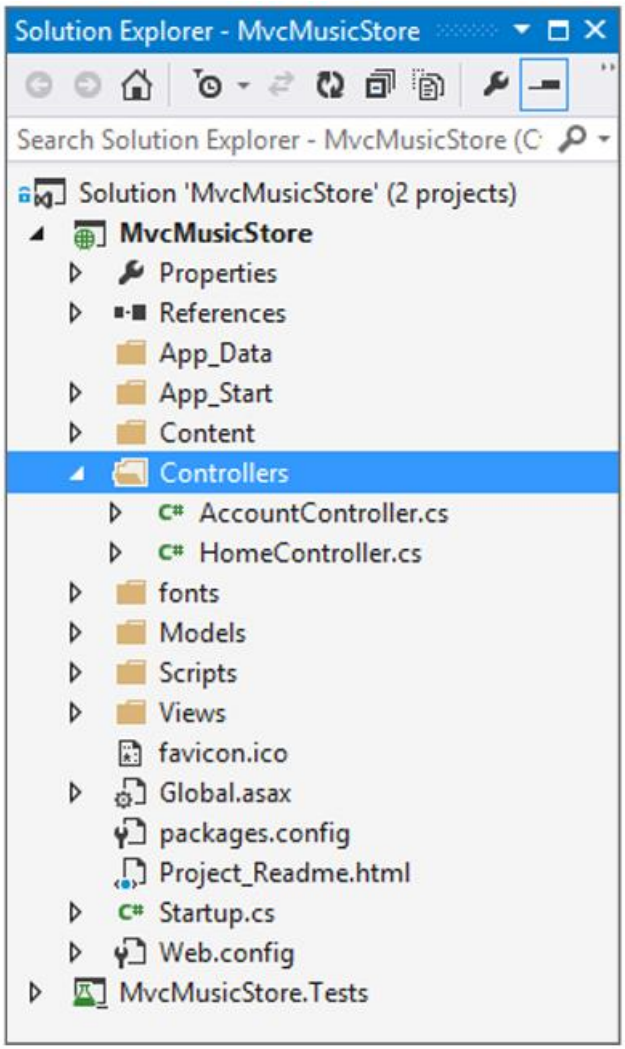


Obrázek 1: Kompilační proces ASP.NET (zdroj: 4, s. 27)

Jak už napovídá druhý pilíř, třetím pilířem je fakt, že je ASP.NET vícejazyčné. Díky tomu může programátor programovat v jím preferovaném programovacím jazyce. Čtvrtý pilíř poukazuje na to, že ASP.NET běží uvnitř společného runtime jazyků, díky tomu je možný například multitasking nebo strukturované generování chyb. ASP.NET je objektově orientované, dle pátého pilíře, tedy má plný přístup k objektům .NET Frameworku (4, s. 30). Šestý pilíř odkazuje na fakt, že ASP.NET podporuje všechny prohlížeče. V tomto pilíři bych rád poukázal na podporu technologie AJAX (asynchronní javascript a XML), ta umožňuje dynamicky měnit obsah stránky při běhu aplikace, bez nutnosti obnovovat celou webovou stránek. A posledním, tedy sedmým pilířem je snadné nasazování a konfigurace aplikace. Díky tomu nemusí programátor složitě konfigurovat nastavení aplikace při jejím nasazení, ale pouze zkopíruje soubory do příslušného adresáře.

3.2 MVC

V první řadě je dobré zmínit, že se jedná o velice oblíbenou softwarovou architekturu, jejíž snahou je oddělit datovou strukturu, řídící logiku a uživatelské rozhraní. Toto je zřejmé přímo z názvů, MVC znamená strukturu Model, View a Controller, znázorněnou adresářovou strukturou Visual Studia na následujícím obrázku (Obrázek 2). V českém překladu je to tedy model, zobrazení a kontrolér. Model nám definuje datovou strukturu, tedy přímo objekty, které jsou předávány do zobrazení. Zobrazení definuje výstup, tedy to co se zobrazí uživateli na obrazovce. Kontrolér, jak už název vypovídá, řídí celý proces. Tedy vyčká na požadavek od uživatele pro zobrazení nějaké stránky, zajistí potřebná data, které pomocí modelu předá přímo do zobrazení.



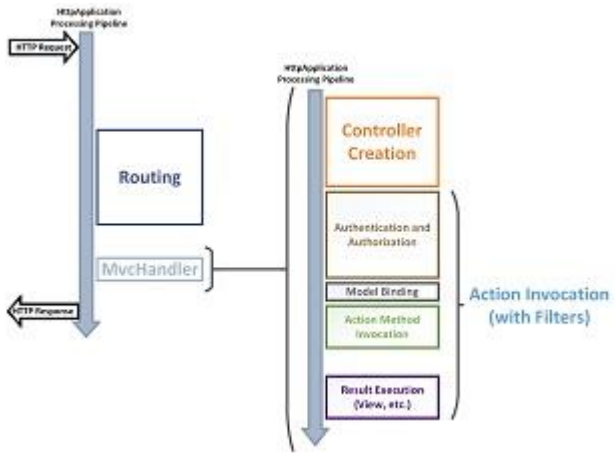
Obrázek 2: Adresářová struktura MVC ve Visual Studiu (zdroj: 2, s. 25)

Příkladem může být jednoduchá stránka, která obsahuje dva textové vstupy. Jejím úkolem má být násobení obou vstupů. Model tedy bude obsahovat tři proměnné (číslo 1,2 a výsledek), na zobrazení bude struktura toho, co uživatel vidí (včetně HTML struktury a definovaného css, popřípadě meta konfigurací) a kontrolér bude obsahovat logiku (v našem případě násobení obou vstupů). V případě takto definované struktury se jedná o strukturu MVC.

Pojem MVC ale neznamená pouze softwarovou architekturu. Nese sebou i spoustu dalších užitečných funkcionalit jako je přímá podpora Bootstrapu, manažera identity (přihlašování uživatelů) a spoustu dalších popsaných více do detailů v knize ASP.NET MVC 5 (2, s.11).

Další funkcionalitou, kterou MVC nabízí, jsou HTML pomocníci (z anglického názvu *helpers*). Těmito pomocníky jsou metody, které se dají vyvolat za pomocí globální proměnné na každém zobrazení (2, s. 114). Jednou z těchto metod je funkce BeginForm, pomocí které je možné parametrizovat formulář, který bude za pomocí této metody vykreslen do HTML struktury. Při použití těchto metod se stává kód použitý v zobrazení mnohem přehlednější. Při nastavování tohoto formuláře se musí definovat atribut, který stanoví, zda se mají položky odeslané formulářem odeslat pomocí *query stringu* nebo do hlavičky HTTP žádosti (2, s. 111). Metoda, která odesílá data pomocí *query stringu* se nazývá GET (v překladu získej) a metoda, která odesílá data pomocí HTTP žádosti, se nazývá POST (v překladu pošli).

Všechny výše uvedené funkcionality, které MVC hrají nedílnou součást životního cyklu celé aplikace znázorněný na následujícím obrázku (Obrázek 3).



Obrázek 3: Životní cyklus aplikace MVC (Zdroj: 1, s. Lifecycle of an ASP.NET MVC 5 Application | The ASP.NET Site)

3.3 LINQ

Pomocí tohoto jazyka je možné nahradit iterační logiku, jako je například cyklus foreach, deklarativním výrazem LINQ (4, s. 580). Jedná se o programovací model, který sjednocuje a usnadňuje implementaci libovolného přístupu k datům (7, s. 23). Díky tomu je možné kód napsaný v cyklu foreach poměrně zkrátit (např. místo 6 ti řádků cyklu napíšeme pouze jeden řádek s dotazem LINQ).

Tento jazyk velmi usnadňuje psaní iterativních dotazů. Nejlépe se tento jazyk vysvětluje přímo na příkladech. Pokud bude potřeba vybrat například 6 osob z nějaké množiny osob, napíše se v LINQ tento jednoduchý dotaz: var sestOsob = osoby.Take(6);. Řešit podobný problém některým z cyklů by znamenalo složitě vymýšlet cyklus s omezujícími podmínkami na počet. LINQ tedy značně zefektivňuje psaný kód, který se zároveň stává více přehledným.

Existují zde dva typy použití dotazu LINQ. První z nich, uveden ve zmíněné ukázce, se nazývá LINQ to *objects*, tedy dotazování se přímo na objekty. Druhým typem je LINQ to *entities*. V příkazech LINQ to *entities* se používají klíčová slova, jako jsou *from*, *in*, *where*, a *select* (4, s. 580).

Další výhodou LINQ je jeho integrované řazení a filtrování. Filtrování se provádí za pomocí podmínky *where* a definované podmínky spolu s kombinací AND a OR (4, s. 586). Nechybí zde ani seskupování, to je provedeno pomocí příkazu *group by* a vhodně zvoleného klíče k seskupení (4, s. 587).

Entity framework dále také nabízí operátor select (vyber). Ten umožňuje promítat výsledky dotazu do definovaného objektu prostřednictvím rozhraní IEnumerable<TResult> (7, s. 67). Díky tomu můžeme z množiny atributů na daném projektu vybrat pouze ty atributy, které budeme skutečně potřebovat a snížit tím nároky na paměť při ukládání těchto výsledků do paměti počítače.

3.4 Ostatní použité frameworky

S těmito frameworky jsem se setkal během konání mé bakalářské práce a jejich použití přináší přehlednější a efektivnější použití kódu a tak má pozitivní vliv na celý informační systém.

3.4.1 Entity Framework a migrace databáze

Tento Framework umožňuje po krátké konfiguraci rychlé a efektivní propojení aplikace s databázovým systémem. Konfigurace a možnosti entity frameworku jsou velice rozsáhlé a propojení nemusí nutně vyžadovat pouze databázový server, může být použit jakýkoli datové úložiště (6, s. 9). Při vykonávání mé bakalářské práce bude použita metodika zvaná *Code-First*. Ta, jak už název napovídá, počítá s tím, že se celý databázový model včetně všech závislostí definuje pomocí tříd (6, s. 9).

Celá struktura databáze je tedy uložena ve strukturovaných třídách, ze kterých se pomocí entity frameworku generují migrace (12). Migrace jsou soubory tříd obsahující zaznamenané změny modelu tříd, které si vytváří a udržuje entity framework. Každá migrace musí mít programátorem definovaný název, který entity framework doplní o časové razítko.

V prvním kroku se nadefinuje struktura třídy, následně se spustí příkaz entity frameworku s názvem migrace, pro vygenerování migrace. Entity framework vygeneruje migraci s definovaným názvem a časovým razítkem. Po spuštění příkazu se celá migrace aplikuje na databázi. Aplikování probíhá tak, že si entity framework vygeneruje SQL skript, který posílá na databázový server. Po aplikaci migrace si entity framework aplikované migrace zapíše do vlastní tabulky, kde si udržuje aplikované migrace, aby věděl, které migrace již aplikoval.

Ve stručnosti se dá říci, že se jedná o nástroj, který překládá programovací jazyk ve formátu databázového modelu a dotazy na dané objekty do SQL příkazů, které následně pouští oproti databázi a výsledky předává přímo v objektech (6, s. 9).

3.4.2 Fluent Security

S tímto frameworkem jsem se plně setkal při vykonávání mé bakalářské praxe. Jedná se o framework, který řídí přístup uživatelských účtů.

V souboru *Web.config* webové aplikace je definován odkaz na databázi. V databázi musí být připravená struktura tří tabulek. Jedna tabulka obsahuje uživatelské účty, druhá tabulka obsahuje uživatelské role a třetí tabulka obsahuje vazby mezi těmito dvěma tabulkami. Dále je zde třída, která obsahuje konfigurace povolených kontrolérů a jejich akcí, na které má uživatel právo přistoupit (10, s. Getting started).

3.4.3 Security Guard

Instalací tohoto frameworku se vytvoří základní struktura podporující správu a přihlašování uživatelů. Vytvoří se kompletní struktura MVC, tedy kontrolér, zobrazení a modely potřebné pro správu uživatelů a jejich přihlašování (9, s. SecurityGuard.MVC5 1.0.5).

3.4.4 Bootstrap

Tento framework se týká předem definovaných stylů a strukturovaného HTML. První Bootstrap byl původně vyvinut firmou Twitter (2, s. 13). Ve výchozím nastavení je v MVC 5 použita výchozí šablona (2, s. 13).

S tímto frameworkem jsem se seznámil na bakalářské praxi, kdy byl analyzován design výsledné aplikace s požadavkem na co nejnižší náklady. Bootstrap se dá plně implementovat do aplikace C# ASP.NET MVC pomocí předpřipravených balíků a správce balíků zmíněném v kapitole 3.3. Po instalaci se plně integruje do aplikace. Sám o sobě se tento framework pyšní plně vyvinutým designem, který je, jako většina designů v dnešní době, plně responzivní. Responzivního designu dosahuje za použití *media queries* aplikovaných na vlastní design (2, s. 468).

Díky své responzivitě jsou tedy všechny komponenty plně přizpůsobené všem zařízením (mobilním telefonům, tabletům, počítačům). Mít responzivní design je v dnešní době velice důležité právě kvůli již zmíněným různým typům zařízení, na kterých má být aplikace použita. Disponuje velkým množstvím předem definovaných HTML struktur, tedy komponent (11, s. Components).

Mezi tyto komponenty patří například různé skupiny tlačítek, posuvníků, struktury textových vstupů a mimo jiné i předem dané struktury menu, které se dají velmi jednoduše použít. Zároveň obsahuje balík základních ikon, které se dají použít pomocí css tříd. V balíku ikon jsou například ikony pro zvuk, odhlášení a přihlášení uživatele a spoustu dalších ikon (11, s. Components).

Dále bootstrap nabízí celou škálu css stylů (11, s. CSS) a javascriptových funkcionalit (11, s. JavaScript), které uvádí na svých stránkách (11, s. Bootstrap).

4 Vlastní řešení

4.1 Analýza uživatelských požadavků aplikace

4.1.1 Interview s vlastníkem hotelového systému

Vlastník hotelového systému uvedl, že prioritou podpůrné aplikace bude přihlašování jednotlivých uživatelů této aplikace na adrese */prihlasit-se*, kde se uživatel bude moci přihlásit svým uživatelským jménem a heslem. Dále uvedl, že grafické zapracování aplikace musí mít minimální finanční náklady, protože nechce investovat do designérů a HTML kodérů. Další předností aplikace zmínil nutnost hromadné emailové rozesílky na hotely z databáze hotelů, která má být odesílána ihned při vytvoření. Jako další požadavek uvedl jednoduchý přehledný report rezervací z hotelového systému a zdůraznil, že z databáze hotelů může být přístup pouze ke čtení dat, tedy nesmí do ní být vykonán žádný zásah.

4.1.2 Analýza uživatelského rozhraní

4.1.2.1 Menu podpůrné aplikace

Menu s jednotlivými položkami se bude nacházet v horní části aplikace.

4.1.2.2 Přihlášení uživatelů do podpůrné aplikace

Uživatelům bude umožněn vstup do podpůrné aplikace pouze pomocí jejich přihlašovacího jména a hesla. Vstupní obrazovka pro přihlášení bude na adrese */prihlasit-se* a bude obsahovat pole pro uživatelské jméno, heslo a tlačítko „Přihlásit se“.

4.1.2.3 Hromadná emailová rozesílka

V menu podpůrné aplikace bude položka „Emailová rozesílka“.
Sekce hromadné rozesílky bude obsahovat:

- Textové pole odesílatele zprávy
- Textové pole pro předmět zprávy
- Obsah zprávy
- Seznam hotelů a jejich emailových adres. U každého bude možnost hotel zvolit. Pokud bude zvolen, odešle se na něj hromadný email.

4.1.2.4 **Přehledný report s počtem rezervací**

V menu bude položka s názvem „Report rezervací“. Po otevření této sekce se přímo zobrazí jednoduchá tabulka. Tabulka bude obsahovat pouze název hotelu na levé straně a počet rezervací na straně pravé.

4.1.3 Technická analýza požadavků

4.1.3.1 **Hromadná emailová rozesílka**

- Rozesílka se bude rozesílat synchronně

4.1.3.2 **Report rezervací**

- Report rezervací se musí generovat v reálném čase.

4.1.3.3 **Databáze**

- V databázi hotelového systému nesmí být ze strany podpůrné aplikace žádný zásah. Obsah je určen pouze ke čtení.

4.1.4 Hromadná emailová rozesílka

- Synchronní rozesílání
  Po odeslání formuláře se rozesílka ihned začne rozesílat.

4.1.5 Report rezervací

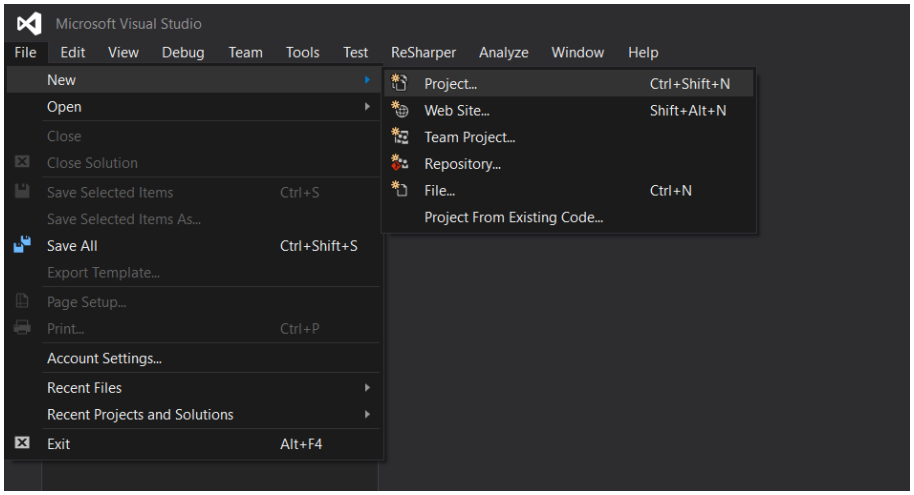
- Generování v reálném čase
  Po otevření reportu se vždy načtou nová data z databáze.

4.1.6 Databáze

- Nová databáze
  Pro udržení obou databází od sebe bude vytvořena nová databáze, která bude zastřešovat data používaná přímo podpůrnou aplikací.
- Databáze hotelového systému
  Pomocí Entity Frameworku bude vytvořen druhý přístup kvůli oddělení obou databází. Z této databáze se bude pouze číst.

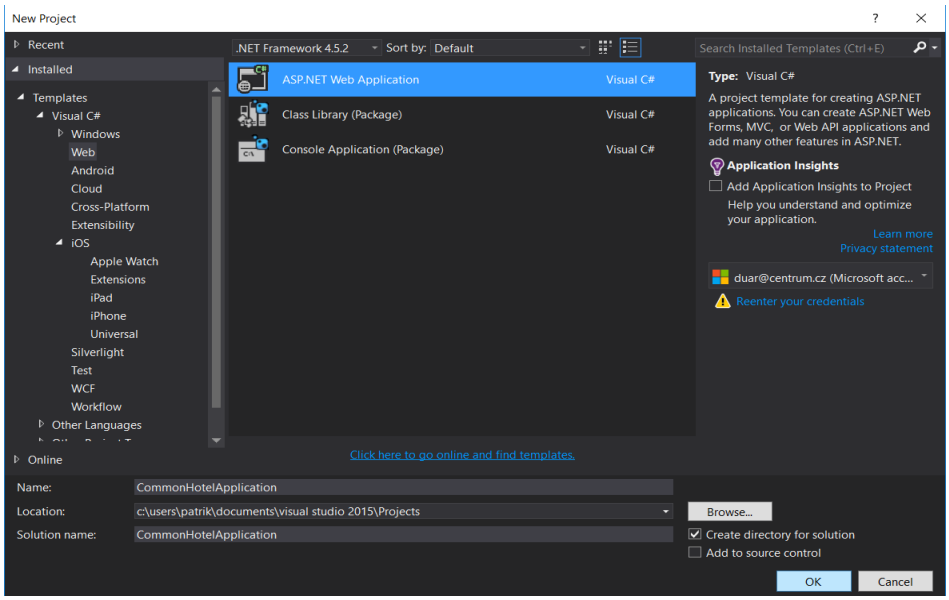
4.2 Vytvoření základní struktury aplikace

Po otevření aplikace Visual Studio se přidá nový projekt pomocí dialogového okna Soubor -> Nový -> Projekt, postupem uvedeným na následujícím obrázku (Obrázek 4).



Obrázek 4: Otevření okna pro vytvoření nového projektu (zdroj: autor)

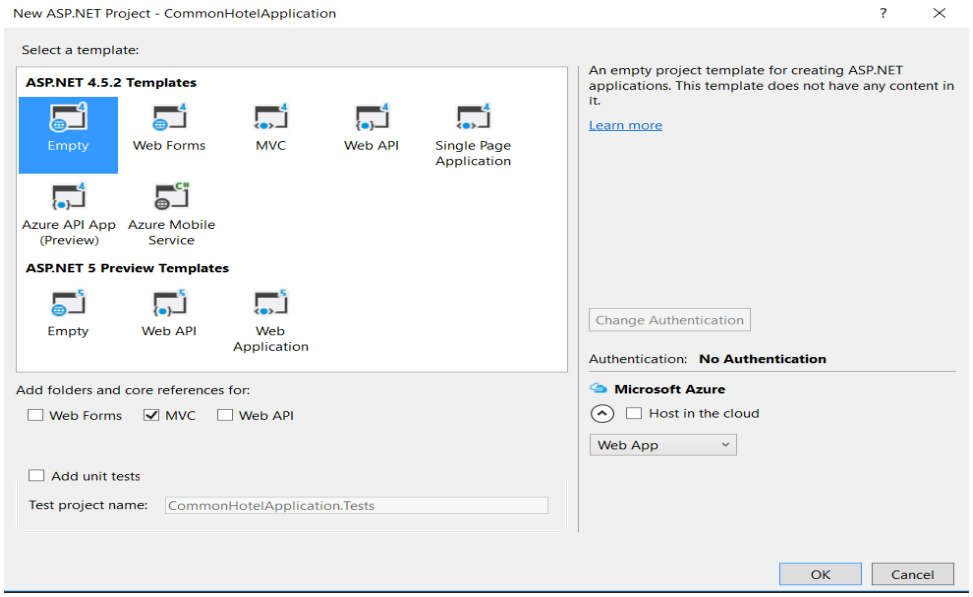
V následujícím okně bude vybráno vytvoření aplikace ASP.NET a vyplněn její název jako na následujícím obrázku (Obrázek 5).



Obrázek 5: Výběr aplikace ASP.NET (zdroj: autor)

Následně už zbývá jen zvolit, že jde o typ aplikace MVC, aby Visual Studio vytvořilo správnou strukturu a nastavilo reference (3, s. 397). Může být vybráno z několika různých

typů šablon. V tomto případě bude zvolena prázdná aplikace, strukturu bude definována samostatně. Výběr prázdného projektu MVC je znázorněn na obrázku (Obrázek 6).



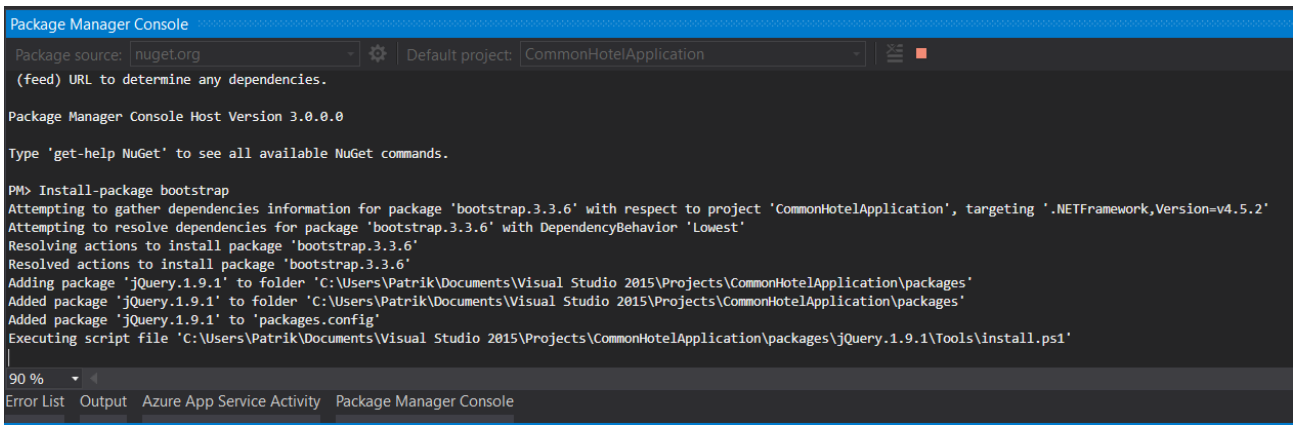
Obrázek 6: Výběr prázdného projektu MVC (zdroj: autor)

4.3 Instalace frameworků a pluginů

Nyní je vytvořena základní struktura podpůrné hotelové aplikace a bude následovat instalace zvolených frameworků. Nejprve se nainstaluje Bootstrap popsaný v kapitole ostatní frameworky. Při zvolení jiné šablony, než prázdné, by se tento framework přidal automaticky. Bohužel by byli přidány i jiné reference, které by ani nemuseli být využité a proto se instaluje dodatečně.

4.3.1 Instalace Bootstrapu pomocí Package Manager Console

Balíky frameworků se dají instalovat dvěma způsoby. Jedním z nich je konzole, která bude použita v této práci. Pro instalaci balíku s frameworkem Bootstrap se jednoduše otevře tato konzole pomocí kontextového okénka View -> Other Windows -> Package Manager Console (8, s. 31), která se nachází v dolní části Visual Studia jak ukazuje obrázek (Obrázek 7). Následně se spustí příkaz *Install-Package bootstrap* (9, s. Bootstrap CSS).



Obrázek 7: Instalace Bootstrapu pomocí konzole pro správu balíčku (zdroj: autor)

4.3.2 Instalace ostatních balíčků

Stejným způsobem se nainstalují i ostatní balíčky kromě balíčku s Entity Framework, ten bude doinstalován později.

4.4 Sestavení přihlašování uživatelů do podpůrné aplikace

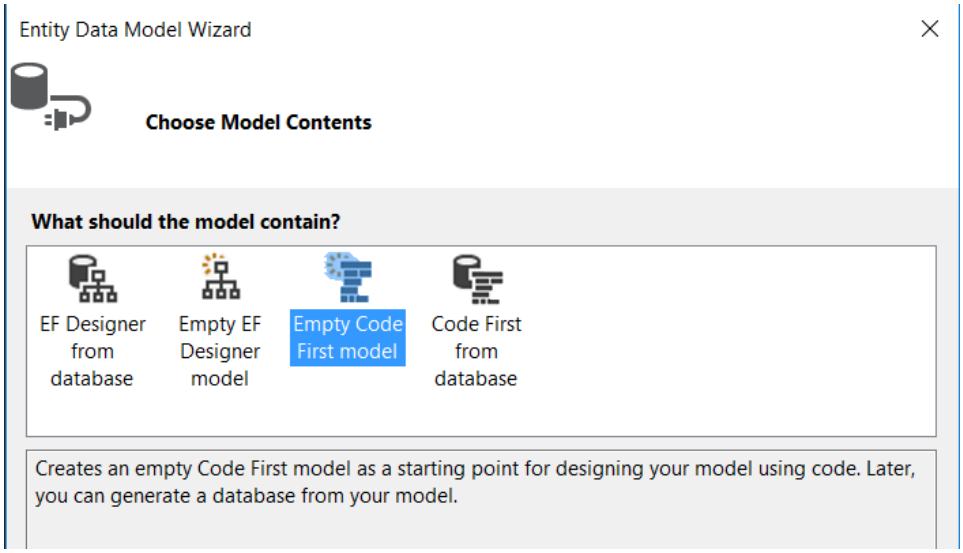
V této chvíli je připravený základ aplikace se všemi frameworky, které budou pro realizaci podpůrné hotelové aplikace potřeba a může se tak začít formovat základní infrastruktura. Základem je vytvoření databázové struktury, první migrace a autorizace v podpůrné aplikaci.

4.4.1 Vytvoření základní databázové struktury

Pro přehlednost se bude model databáze udržovat v samostatném projektu. Pomocí dialogového okna pro vytvoření nového projektu popsaného v kapitole 4.2, se vytvoří prázdný projekt, do kterého se bude vkládat infrastruktura nové databáze. Nový projekt se bude jmenovat Model, aby bylo z názvu zřejmé, že obsahuje model databáze. A následně bude v tomto projektu pomocí konzole správce balíčků nainstalován Entity Framework.

4.4.1.1 Základní nastavení Entity Frameworku

V novém projektu se musí vytvořit propojení databáze a nastavit prázdný model *Code-First*. To bude provedeno tak, že se pravým tlačítkem klikne na nový projekt a zvolí se přidat – novou položku. Následně se vybere *Entity Data Model*. To spustí průvodce (Obrázek 8), ve kterém bude zvolena položka *Empty Code First Model*. Tím se určí vytvoření nové čisté databáze.



Obrázek 8: Výběr prázdného modelu Code First (zdroj: autor)

4.4.1.2 Vytvoření struktury modelu databáze

Nyní přichází na řadu vytvoření struktury databáze. V projektu *Model* se vytvoří nová složka s názvem *Entities*. Do této složky se přidají dvě třídy. Jedna s názvem *User* a druhá s názvem *Role*. Definice tříd je uvedena v referenční příručce jazyka C# (5, s. class (Referenční dokumentace jazyka C#)).

4.4.1.3 Třída User

V této tabulce se budou evidovat následující položky:

- UserId (celé číslo, primární klíč)
- Username (Uživatelské jméno)
- PasswordHash (pole bytů)
- PasswordSalt (pole bytů)

Atributy Password vycházejí z definice struktury databáze pro Security Guard – autorizaci uživatele. Dále se přidá pomocí typového interface ICollection vazba na uživatele (obdobně se tak provede ve třídě pro role), tím se vytvoří vazba m:n mezi oběma tabulkami.

4.4.1.4 **Třída Role**

Tabulka Role bude obsahovat následující položky:

- RoleId (celé číslo, primární klíč)
- Name (jméno role)

Výsledná struktura uživatele vypadá takto:

```
public class User
{
    [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int UserId { get; set; }

    public string Username { get; set; }

    public byte[] PasswordHash { get; set; }

    public byte[] PasswordSalt { get; set; }

    public virtual ICollection<Role> Roles { get; set; }
}
```

4.4.1.5 **Zařazení modelů do kontextu databáze**

Nyní už zbývá pouze říci entity frameworku, že se jedná o modely databáze, tzv. *DbSets*. Po přidání *Code First* v kapitole 4.4.1.1. se vytvořila třída, která dědí od třídy *DbContext* a obsahuje základní instrukce pro přidání těchto *DbSets*. Výsledný zápis těchto setů bude následující:

```
public virtual DbSet<User> Users { get; set; }
public virtual DbSet<Role> Roles { get; set; }
```

4.4.1.6 **Vytvoření migrace**

Povolení migrace
Nejprve je důležité povolit migrace. To se zařídí snadno pomocí konzole správy balíčků příkazem *Enable-Mirations* (12, Enable Migrations). Důležité je mít v tomto kroku zvolený jako výchozí projekt v konzole správce balíčků právě Model, aby se tyto příkazy aplikovali pro tento projekt.

Vytvoření migrace
Vytvoření migrace je velice jednoduché. Pomocí konzole správce balíčků se spustí následující příkaz *Add-Migration* "NazevNasiMigrace" (12, Generating & Running Migrations). Entity Framework automaticky vygeneruje třídu pro aplikaci změn modelu databáze a výslednou třídu po přidání migrace otevře.

Aplikování migrace
V tomto kroku je potřeba vytvořit prázdnou databázi na databázovém serveru a upravit nastavení *Web.config* tak, aby se mohl Entity Framework připojit k serveru. Následně se aplikují vytvořené migrace pomocí konzole pro správu balíčků a příkazu *Update-Database* (12, Generating & Running Migrations).

Po tomto kroku se v databázi vytvořili čtyři nové tabulky. Tři pro role, uživatele a vazby mezi nimi a jedna pro historii migrací, která udržuje stav aktuální migrace.

4.4.2 Nastavení Security Guard

Musí se nastavit následující dvě konfigurace, aby Security Guard věděl o nastavení tabulek v databázi:
<CFMembershipSettings dbContext="Model.CommonHotelApplicationModel, Model" userObject="Model.Entities.User, Model" roleObject="Model.Entities.Role, Model" keyType="Int16" userTable="Users" roleTable="Roles" allowLoginWithEmail="false" useEmailAsUsername="false" />
A dále se musí přidat konfigurační sekce, tím se definuje, že se má použít právě toto nastavení.

4.4.3 Vytvoření vstupní obrazovky

Každá aplikace musí mít někde svůj začátek. V tomto případě se jako vstupní bod vytvoří prázdná obrazovka, na které bude vytvořeno jednoduché menu.

4.4.3.1 Vytvoření základní struktury domácího kontroléru

Podle modelu MVC s postupem definovaným v knize ASP.NET MVC 5 (2, s. 39) bude vytvořen podle návodu jednoduchý domácí kontrolér (anglicky zvaný *Home Controller*). Jelikož je žádoucí pouze jedna vstupní stránka, z návodu bude převzata pouze část týkající se výchozí stránky, tzv. *Index*.

4.4.4 Nastavení Fluent Security

V tuto chvíli je vytvořena stránka pro přihlášení a zároveň vstupní obrazovka. Pomocí Fluent Security se budou definovat pravidla přístupu k těmto stránkám. Bude se tedy řídit autorizace a pomocí podmínek bude definováno, na jaké stránky může přihlášený uživatel a na které ne. Pomocí Fluent Security se zároveň určí se kterou rolí má uživatel právo na danou akci nebo kontrolér.

Bude nastaven plný přístup pro stránku s přihlášením a za použití třídy kontroléru *BaseController,* který se vytvořil při přidání domácího kontroléru, bude zamezeno všem stránkám, dědícím tento kontrolér anonymní přístup.
Toto nastavení bude zaneseno do třídy *Global.asax* kde jsou veškeré inicializace, které se provádí při startu aplikace. Výsledné nastavení je ukázáno na následujícím obrázku (Obrázek 9).

```
namespace CommonHotelApplication
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            SecurityConfigurator.Configure(configuration =>
            {
                configuration.Advanced.SetDefaultResultsCacheLifecycle(Cache.PerHttpRequest);
                configuration.GetAuthenticationStatusFrom(() => HttpContext.Current.User.Identity.IsAuthenticated);
                configuration.GetRolesFrom(Roles.GetRolesForUser);
                configuration.ForAllControllersInheriting<Controllers.BaseController>().DenyAnonymousAccess();
                configuration.For<Controllers.SGAccountController>(c => c.LogOn()).Ignore();
                configuration.For<Controllers.SGAccountController>(c => c.ForgotPassword()).Ignore();
                configuration.For<Controllers.SGAccountController>(c => c.ForgotPasswordSuccess()).Ignore();
            });
            GlobalFilters.Filters.Add(new HandleSecurityAttribute(), 0);
        }
    }
}
```

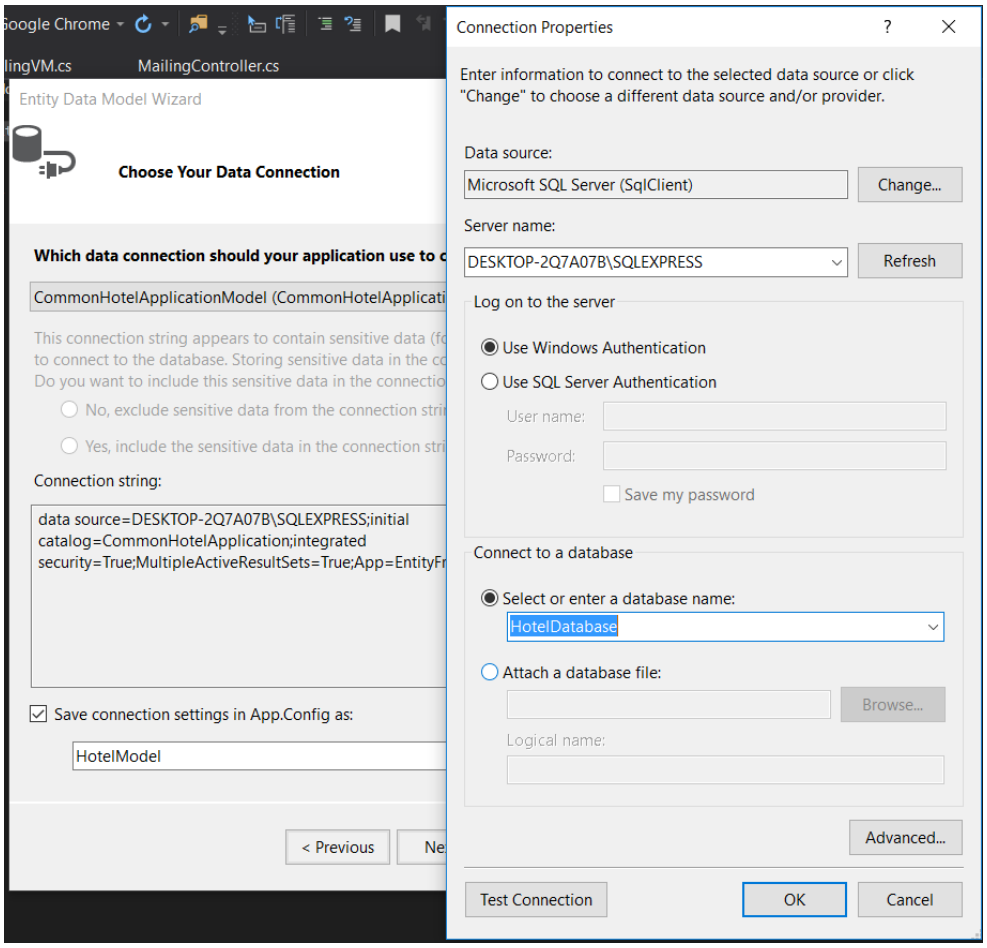
Obrázek 9: Nastavení základních pravidel autorizace (zdroj: autor)

4.5 Propojení aplikace s již existující databází hotelů

Pro připojení již existující databáze se bude postupovat podobně jako v kapitole 4.4.1.1, místo nové databáze ale bude vybrána *Code First* z již existující databáze.

4.5.1 Nastavení připojení k existující databázi

Průvodce přidáním existující databáze se zeptá na název serveru, na kterém je databáze spuštěna. Budou zadány potřebné informace, jako je adresa serveru, uživatelské jméno a heslo. Následně bude vybrán název databáze jako na obrázku (Obrázek 10).



Obrázek 10: Nastavení Code First na existující databázi (zdroj: autor)

4.5.2 Výběr tabulek

V následujícím kroku se Entity Framework dotáže na tabulky, které budou v tomto importu existující databáze použity. V tomto případě bude potřeba pouze tabulka hotelů a tabulka rezervací. Tabulka hotelů bude použita v hromadné rozesílce emailů a obě tabulky budou použity při generování reportu. Po dokončení Entity Framework vygeneruje třídu s již nastaveným databázovým kontextem.

4.5.3 Nastavení Connection stringu

Po vytvoření tohoto propojení s existující databází je důležité si uvědomit, že Entity Framework vygeneroval Connection string (nastavení obsahující informace k propojení s databázovým serverem) do *App.config* v projektu Model, na databázi ale bude přistupovat podpůrná webová aplikace a tak je potřeba toto nastavení ještě přenést do konfigurace *Web.config* podpůrné aplikace.

4.6 Hromadná emailová rozesílka

4.6.1 Vytvoření základní struktury MVC

4.6.1.1 Model

Bude vytvořena třída s názvem Mailing.cs obsahující následující položky:

- Hotels (list, seznam hotelů z databáze)
- SelectedHotelEmails (pole vybraných emailových adres hotelů)
- Subject (předmět zprávy)
- Body (tělo zprávy)

V tomto kroku je důležité počítat s nastavením atributů těchto proměnných. MVC umožňuje pojmenovávat proměnné, tedy nastavit jim atributy názvu. Tyto názvy se následně budou vypisovat při vykreslování proměnných v zobrazení. Syntaxe je velmi jednoduchá, nad každou proměnnou, kterou má být pojmenována, bude přidán do hranatých závorek atribut *Display* a jako jeho vstup bude zvolen *Name*. Výchozí syntaxe může vypadat například takto:

```
[Display(Name = "Předmět")]
public string Subject { get; set; }
```

4.6.1.2 **Zobrazení (View)**

Jednoduchý formulář s výpisem hodnot v modelu a tlačítkem sloužícím pro odeslání daného formuláře.

V první řadě bude nastaven titulek stránky a vytvořena struktura pro formulář pomocí MVC metod. Ta je definována pomocí extension metody *BeginForm* na proměnné *Html*. Vstupními parametry této metody jsou jméno akce, kontroléru a typ požadavku (requestu), který je použit pro odeslání formuláře. V tomto případě je to výše zmíněný POST.

Také se nesmí zapomenout na to, že se bude v tomto zobrazení používat model definovaný pro hromadnou emailovou rozesílku. Tato závislost se předá do zobrazení pomocí syntaxe MVC @model. Dále se použije HTML syntaxe Bootstraperu pro zobrazení základního setu položek (*Fieldsetu*) a do něj se doplní položky. Pro vykreslení položek nabízí MVC několik jednoduchých metod, které se dají pomocí proměnné Html použít přímo v zobrazení. Jsou to metody *NameFor*, *TextBoxFor* a *TextAreaFor*. Tyto metody vykreslí přímo HTML strukturu k proměnný z našeho modelu. Výsledek zápisu je uveden na následujícím obrázku (Obrázek 11).

```
@model CommonHotelApplication.Models.MailingVM
@{
    ViewBag.Title = "Hromadná emailová rozesílka";
}

@using (Html.BeginForm("Index", "Mailing", null, FormMethod.Post))
{
    <div class="row">
        <fieldset class="full">
            <legend>Hromadná emailová rozesílka</legend>
            <div class="row-wrapper">
                <div class="span5">
                    <div class="form-section" id="mailing-form">
                        <dl class="dl-horizontal">
                            <dt>@Html.NameFor(m => m.Sender) *</dt>
                            <dd>@Html.TextBoxFor(m => m.Sender)</dd>
                            <dt>@Html.NameFor(m => m.Subject) *</dt>
                            <dd>@Html.TextBoxFor(m => m.Subject)</dd>
                            <dt>@Html.NameFor(m => m.Body) *</dt>
                            <dd>@Html.TextAreaFor(m => m.Body)</dd>
                        </dl>
                    </div>
                    <br />
                </div>
            </div>
        </fieldset>
    </div>

    <div id="tabs">
        <table class="hotels-view block">
            <tr>
                <th>Jméno hotelu</th>
                <th>Emailová adresa</th>
                <th></th>
            </tr>
            @foreach (var hotel in Model.Hotels)
            {
                <tr>
                    <td>@hotel.Name</td>
                    <td>@hotel.Email</td>
                    <td><input type="checkbox" name="@Html.NameFor(m => m.SelectedHotelEmails)" value="@hotel.Email"/></td>
                </tr>
            }
        </table>
        </div>
    <div class="form-actions" style="margin-left: 0;">
        @Html.ActionLink("Zpět", "Index", "Home", new { @class = "btn btn-primary" })
        <button class="btn btn-success">Odeslat</button>
    </div>
}
```

Obrázek 11: Implementace zobrazení hromadné emailové rozesílky (zdroj: autor)

4.6.1.3 **Kontrolér (Controller)**

Nejprve se implementuje základní struktura, tedy třída, která dědí třídu BaseController vytvořenou doplňkem SecurityGuard kvůli řízení přístupů uživatelů. Na kontroléru se vytvoří dvě akce s názvem Index. Jedna bude ohodnocena atributem HttpGet a druhá atributem HttpPost. Tato implementace je nezbytná kvůli typu formuláře, který budeme odesílat metodou POST dle standardu HTTP.

4.6.2 Implementace logiky kontroléru

4.6.2.1 **Implementace metody GET**

V první řadě budeme implementovat základní inicializaci, tedy data, která se předají uživateli ve chvíli, kdy vstoupí na zobrazení. Připravíme kontroléru proměnnou _hotelModel, která je typu HotelModel. Tento typ nám vygeneroval Entity Framework v předchozím kroku. Pomocí této proměnné se aplikace jednoduše dotáže databáze na hotely, které v hotelové databázi jsou, a následně je předá do proměnné v předpřipraveném modelu.
V tomto případě bude seznam hotelů jediné, co se má uživateli v prvním kroku zobrazit.

4.6.2.2 **Implementace metody POST**

V druhé řadě se bude implementovat co se má stát, pokud uživatel odešle formulář. Je důležité v tomto kroku počítat alespoň s jednoduchými validacemi, jako je nevyplnění některého z povinných polí, nebo vyplnění hodnoty ve špatném formátu (jako je například emailová adresa).

Implementace validací
Pro implementaci validací se bude používat proměnná MVC s názvem ModelState. V překladu se proměnná nazývá „stav modelu“. Ta obsahuje informace o tom, zda je nebo není model validní. Na této proměnné je implementována metoda AddModelError. Má dva vstupní parametry, jméno proměnné, ke které se chyba vztahuje, toto pole může být i prázdné, a chybovou hlášku, které se má vypsat. Pokud je první pole prázdné, chybová hláška se vztahuje k celému modelu.

Budou přidány tři chybové hlášky. Pole předmět a obsah zprávy nesmí být prázdné a zároveň musí být vybraný alespoň jeden hotel, na který se má email rozeslat.

Zpracování ne/validního požadavku

Po tom, co požadavek projde přes validace, se aplikace zeptá proměnné Modelstate, zda je model validní. Proměnná Modelstate obsahuje atribut IsValid, pokud je nastavený na hodnotu false, tak se uživateli vrátí zobrazení s tímto modelem, aby mu mohli být chyby zobrazeny. V opačném případě se provede odeslání emailu a uživatel bude přesměrován na domovskou stránku.

K odeslání emailu bude použita již předpřipravená funkcionalita, implementovaná ve sdílených knihovnách .NET Framweorku. Již nyní je zřejmé, jak .NET Framework zjednodušuje práci díky svým obsáhlým knihovnám. Výsledný kód je zobrazen na následujícím obrázku (Obrázek 12).

Validace, které se předávají do stavu modelu, musí být následně přidány do zobrazení, respektive v zobrazení se musí zařídit jejich výpis. To bude realizováno pomocí rozšířených metod a proměnné Html, zmíněné v kapitole 4.6.1.2 a to za použití metody ValidationMessageFor, ta zařídí, že se do zobrazení podpůrné hotelové aplikace v případě, že stav modelu obsahuje chybovou hlášku k dané proměnné, vypíše validační hláška.

```
[HttpPost]
public ActionResult Index(MailingVM vm)
{
    if (string.IsNullOrEmpty(vm.Subject))
    {
        ModelState.AddModelError("Subject","Předmět zprávy nesmí být prázdný.");
    }
    if (string.IsNullOrEmpty(vm.Sender))
    {
        ModelState.AddModelError("Sender", "Odesílatel zprávy nesmí být prázdný.");
    }
    if (string.IsNullOrEmpty(vm.Body?.ToHtmlString()))
    {
        ModelState.AddModelError("Body", "Tělo zprávy nesmí být prázdné.");
    }
    if (vm.SelectedHotelEmails == null || !vm.SelectedHotelEmails.Any())
    {
        ModelState.AddModelError("SelectedHotelIds", "Vyberte alespoň jeden hotel.");
    }
    if (!ModelState.IsValid)
    {
        vm.Hotels = _hotelModel.Hotels.ToList();
        return View(vm);
    }
    foreach (var hotelEmail in vm.SelectedHotelEmails)
    {
        var mail = new MailMessage("info@pc-zone.cz", hotelEmail);
        var client = new SmtpClient
        {
            Port = 25,
            DeliveryMethod = SmtpDeliveryMethod.Network,
            UseDefaultCredentials = true
        };
        mail.Subject = vm.Subject;
        mail.Body = vm.Body.ToHtmlString();
        mail.IsBodyHtml = true;
        client.Send(mail);
    }

    return RedirectToAction("Index", "Home");
}
```

Obrázek 12: Implementace metody POST hromadné rozesílky emailů (zdroj: autor)

Nyní je implementována hromadná rozesílka emailů na hotely uložené v databázi hotelového systému.

4.7 Přehledný report rezervací

4.7.1 Vytvoření Modelu

Obdobně jako v předchozí kapitole bude vytvořen model, který bude obsahovat pouze proměnnou typu slovník (*Dictionary*). Tato proměnná byla zvolena na základně požadavků reportu. Slovník je proměnná, která má dva typové parametry a to klíč, který je vždy unikátní a hodnotu ke klíči. Vzhledem k faktu, že budou výsledky vždy název hotelu a počet rezervací, je zde zřejmé, že hotel bude vždy unikátním klíčem a počet rezervací bude jeho hodnota. Toto bude jediná proměnná, které bude pro výpis potřeba.

4.7.2 Vytvoření zobrazení (View)

K vytvoření zobrazení bude potřeba pouze jednoduchého cyklu foreach, který zařídí výpis všech hodnot a HTML struktury tabulky. Bude provedeno základní nastavení jako v předchozí kapitole, tedy bude nastaven titulek stránky a závislost na model.

4.7.3 Vytvoření a implementace kontroléru (Controller)

V tomto kontroléru bude pouze jedna metoda typu GET, jelikož mají být data pouze zobrazena.

4.7.3.1 **Implementace metody**

U tohoto reportu budou zobrazena pouze data, bude vytvořena akce, která bude typu GET. Tako akce se vždy dotáže databáze pomocí závislosti na databázový model, implementované v kapitole 4.6.2.1 a vrátí pouze jedno zobrazení.

4.7.3.2 **Dotaz na data z databáze**

V této kapitole bude využita metoda LINQ zmíněná v kapitole 3.3 a implementuje se tak vytažení požadovaného reportu z hotelové databáze. V první řadě budou vytvořeny dvě proměnné, jedna s hotely a druhá s rezervacemi. Tyto dvě proměnné se budou pomocí metody LINQ spojovat (za pomocí příkazu join), to nám vytvoří strukturu SQL Join, kterou se Entity Framework dotáže do databáze. Následně bude sepsán dotaz za pomocí metody GroupBy, který dle syntaxe SQL data setřídí do skupin podle zvoleného klíče. Klíčem bude název hotelu. Pro výpočet počtu rezervací, je ještě nezbytné použít metodu výběru (Select), kterou bude vybrán vždy jeden klíč a k němu aplikována metoda spočítej (Count), která k danému

klíči spočítá počet výskytů. Tím budou vytažena všechna potřebná data. Výsledný kód je uveden na následujícím obrázku (Obrázek 13).

```
namespace CommonHotelApplication.Controllers
{
    public class ReportController : BaseController
    {
        private readonly HotelModel _hotelModel;

        public ReportController()
        {
            _hotelModel = new HotelModel();
        }

        [HttpGet]
        public ActionResult Index()
        {
            var hotels = _hotelModel.Hotels;
            var rezervations = _hotelModel.Resezvations;
            var joined = hotels.Join(rezervations, hotel => hotel.HotelId, reservation => reservation.Hotel_Id,(hotel, resezvation) => new {hotel, resezvation});
            var grouped = joined.GroupBy(x => x.hotel.Name).Select(g => new {g.Key, count = g.Key.Count()});
            return View(new ReportVM
            {
                ReportData = grouped.ToDictionary(g => g.Key, g => g.count)
            });
        }
    }
}
```

Obrázek 13: Filtrování dat reportu z hotelové databáze (zdroj: autor)

4.8 Úprava výchozích šablon

ASP.NET vytvořil sadu základních zobrazení, které jsou použité jako šablony. Tyto šablony se nazývají „Layout“. V základní aplikaci vytvořil pouze jednu takovou šablonu, která obsahuje hlavičku a patičku stránky. Tato šablona bude upravena tak aby, odpovídala požadavkům aplikace. V této výchozí šabloně existuje metoda *RenderBody*, ta umožní výpis stánek na kterých je tato šablona použita, na to místo, kde je tato metoda použitá. Výchozí šablonu používají automaticky všechny stránky, které v aplikaci jsou, pokud programátor nenastaví jinak. Výchozí šablona také obsahuje odkazy na použité styly, javascripty a ostatní funkcionality.

4.9 Vytvoření menu

Menu bude vytvořeno za pomocí předem definované HTML struktury (11, s. Components) a stylů Frameworku Bootstrap. Odkazy na metody (akce kontroléru) budou vytvořeny za pomoci rozšířené metody na proměnné Html od MVC s názvem *ActionLink*, neboli odkaz na akci. Do vstupních parametrů této metody patří název kontroléru a akce na tomto kontroléru. Výsledný zápis je vidět na následujícím obrázku (Obrázek 14).

```
<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
                <li>@Html.ActionLink("Domů", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })</li>
                <li>@Html.ActionLink("Rozesílka hromadných emailů", "Index", "Mailing", new { area = "" }, new { @class = "navbar-brand" })</li>
                <li>@Html.ActionLink("Report", "Index", "Report", new { area = "" }, new { @class = "navbar-brand" })</li>
            </ul>
        </div>
    </div>
</nav>
```

Obrázek 14: Menu vytvořené za pomocí Bootstrapu a rozšířených metod MVC (zdroj: autor)

4.10 Úprava názvů URL

Dalším krokem při vytváření podpůrné aplikace hotelového systému je úprava názvů generované url za pomocí tzv. atributů směrování (Routing Attributes).

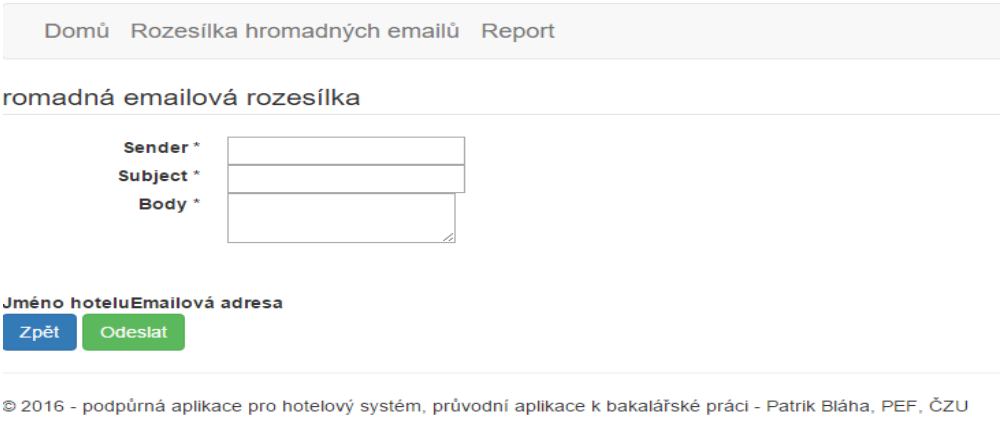
Tyto atributy se jednoduše zapíší před název akce v kontroléru nebo před třídu kontroléru. Jako vstupní parametr se udává požadovaný název části url. Aktuální adresa url pro přihlášení pomocí kontroléru Security Guard je „SGAccount/LogOn“, z této adresy vyplívá, že výchozí směrování MVC používá vždy název kontroléru, ovšem bez dodatečného názvu Controller, který každá třída obsahující kontrolér musí mít a název metody. Pomocí atributů směrování bude nastaven název kontroléru na prázdný a název metody pro přihlášení na „prihlasit-se“.

Aby bylo možné toto realizovat, je potřeba dát vědět nastavení směrování MVC, že místo výchozího směrování, které realizuje právě to, že se směruje pomocí názvu kontroléru a jeho akcí, bude používat směrování pomocí těchto MVC atributů. To bude realizováno jednoduchou modifikací nastavení routování ve složce „App_Start“ a třídě „RouteConfig“. Aktuální funkcionalita směrování bude smazána a přidána funkcionalita, která předá MVC informaci o tom, že má použít směrovací atributy. To realizuje následující syntaxe: *routes.MapMvcAttributeRoutes();*.

Nyní je akce přihlášení na adrese ,“prihlasit-se“. Tyto atributy je potřeba nastavit na všech kontrolérech a akcích.

4.11 Úprava stylů aplikace za pomocí Bootstrap Frameworku

V tento okamžik je připravené vše, co se po technické stránce podpůrné aplikace týká. Zbývá už jen upravit styl aplikace. Po vytvoření této aplikace a použití Bootstrap Frameworku je důležité styl poupravit ať už za použití dalších struktur frameworku nebo vlastních stylů. Pro ukázku je zde znázorněno, jak vypadá hromadná rozesílka emailů před závěrečnou stylizací (Obrázek 15).



Obrázek 15: Ukázka hromadné emailové rozesílky bez úpravy stylů (zdroj: autor)

Zde je patrné, že Bootstrap Framework automaticky ve výchozím nastavení odsadil menu od začátku stránky. Zároveň neudal velikost textových polí, ani výpis chybových hlášek, což je nežádoucí. Pro přepsání těchto stylů bude vytvořen jeden vlastní soubor stylů css, který bude definovat dodatečné styly. Je důležité, aby se styly nemodifikovali přímo ve stylech generovaných Bootstrap Frameworkem, ale aby se toto řešilo přes vlastní stylový soubor. Při budoucím updatu by se styly aplikované ve třídách stylů Bootstrap Frameworku smazali. To by vedlo ke ztrátě definovaných stylů.

Upravené styly budou přidány pomocí nového souboru, který bude uložen do složky s názvem „Content“. Tu MVC automaticky vytvořilo pro všechny soubory spojené se stylizací aplikace. Do této složky tedy bude vytvořen nový soubor s názvem „Style.css“. Kliknutím na složku „Content“ pravým tlačítkem Visual Studio automaticky nabídne přidání položky „Style sheet“, tedy soubor stylů. Dále stačí jen zadat zvolený název a soubor se automaticky vytvoří.

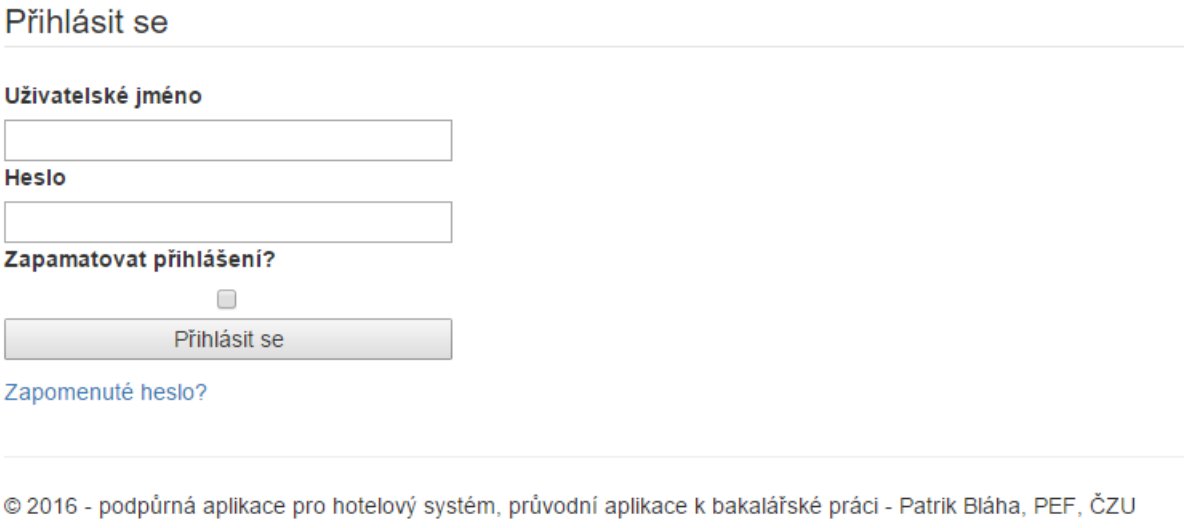
Po vytvoření tohoto souboru můžou být přidány změnové styly. Aby se aplikovali do aplikace, je důležité je přidat odkazem do výchozí šablony aplikace popisované v kapitole 4.8.

4.12 Závěrečné úpravy

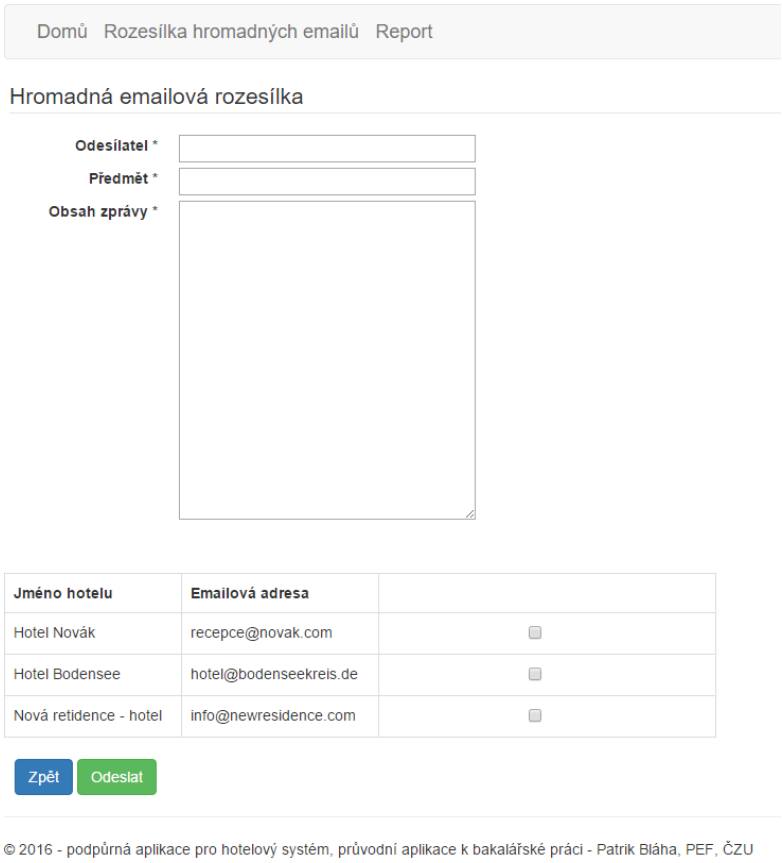
Závěrem je potřeba provést ještě pár základních úprav. Při implementaci frameworku správy uživatelů je potřeba výchozí zobrazení upravit tak, aby se výchozí šablona nezobrazovala při přístupu na přihlašovací stránku a zároveň s tím je potřeba přepsat tyto zobrazení do českého jazyka. Tento framework je vždy definován po výchozí instalaci v anglickém jazyce.

4.13 Výsledná aplikace

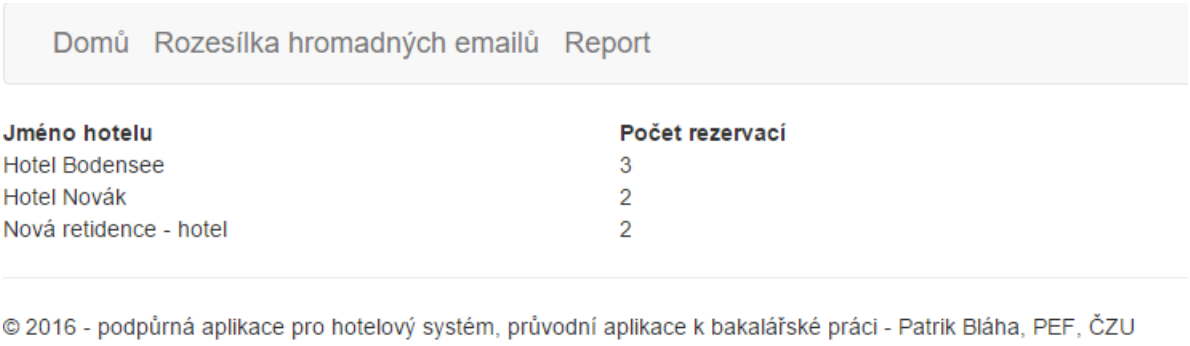
Výsledkem je stylovaná, přehledná aplikace, které zastřešuje přihlášení uživatelů do systému (Obrázek 16), hromadnou rozesílku emailů na předem určený seznam hotelů z databáze hotelového systému (Obrázek 17) a přehledný report s počtem rezervací v hotelovém systému (Obrázek 18).



Obrázek 16: Přihlašovací stránka uživatelů podpůrné aplikace (zdroj: autor)



Obrázek 17: Hromadná rozesílka emailů po úpravách stylů (zdroj: autor)



Obrázek 18: Přehledný report rezervací po úpravách stylů (zdroj: autor)

4.14 Testování

Následujícím krokem je testování celé aplikace. V podnicích s větším počtem programátorů se k tomuto úkonu využívá lidí na pozicích testera. U menších aplikací se nevyplatí testování třetí osobou, a proto aplikaci testuje programátor přímo při implementaci aplikace. U této podpůrné aplikace hotelového systému je velikost naimplementovaného kódu minimální, a proto není potřeba rozsáhlého testování. Při testování této nebyly na lokálním vývojovém počítači zjištěny žádné problémy. I tak se ale musí celá aplikace otestovat znovu po nasazení.

4.15 Nasazení aplikace na webový server

Nasazení aplikace na webový server je pomocí Visual Studie velmi jednoduché. Vše se provádí přes kontextové menu při kliknutí na projekt pravým tlačítkem. Příkazem *Publish* zobrazí Visual Studio nové okno obsahující profily, které jsou předem definované a tlačítko vlastního profilu (*Custom*), kterým se dá definovat vlastní typ vystavování aplikace na webový server. Vystavovat se dá základními třemi metodami. První z nich je vystavení přímo a webový server pomocí přihlašovacích údajů k tomuto serveru. Druhý z nich je pomocí FTP přístupu, který se autorizuje pomocí uživatelského jména a hesla k účtu FTP. Třetí z těchto možností je vystavení pomocí vygenerovaných souborů přímo na systém souborů lokálního počítače.

Při nasazování této aplikace bude použita metoda třetí. Po vytvoření profilu pomocí okna pro vystavování Visual Studia bude vybrána položka *File System* a vybráno umístění na lokálním počítači. Po stisknutí tlačítka *Publish* se do tohoto umístění vygenerují soubory. Tyto soubory se zkopírují na webový server předpřipravený pro tuto webovou aplikaci.

5 Zhodnocení výsledků

Vytváření webových aplikací pomocí ASP.NET MVC za použití podpůrných frameworků je značně jednoduché. Velké množství možných rozšíření a technologií, které se dají doinstalovat, značně usnadňuje vytváření aplikací, které se díky tomu dokáží rychle rozrůstat.

Výsledkem této práce je základní infrastruktura podpůrné aplikace hotelového systému, která se dala snadno realizovat právě za pomocí zmíněných frameworků a technologií. Díky knihovnám ASP.NET je psaní aplikací velice snadné. Knihovny obsahují velké množství již naimplementovaných funkcionalit. Není nutné řešit problémy s autentizací a řízením uživatelských účtů, díky použitému frameworku. Propojení s databází je velmi jednoduché a přehledné. Za pomoci Entity Frameworku se tedy dá velice jednoduše udržovat struktura databáze a tak nevznikají žádné chyby při implementaci a změnách databázové struktury. Vzhledem k dvojí kompilaci ASP.NET se dá proces nasazování aplikace a udržování aplikace automatizovat.

Dle poznatků z praxe by se dalo pár věcí ještě vylepšit. Existují automatizované funkcionality, které se například starají o výpisy tabulek, kde se proces výpisu (jako u reportu rezervací) dá plně implementovat dalším frameworkem, který se navíc automatizovaně stará o filtrování položek a umožňuje tak i vyhledávání.

6 Závěr

Prvním cílem byla analýza základních a dodatečných frameworků použitých při implementaci podpůrné hotelové aplikace. Byly zde zapracovány principy používané při tvorbě aplikace. V kapitole 3.1 bylo vysvětleno rozhodnutí použít ASP.NET pro jeho obsáhlou knihovnu plnou funkcionalit a vysvětlen princip kompilace. V kapitole 3.2 byl vysvětlena struktura MVC a metodika při používání této softwarové architektury zapracovaná v celé části podpůrné hotelové aplikace.

V následující kapitole 3.3 byla vysvětlena funkcionalita dotazů tvořených za pomoci tvorby těchto dotazů pomocí LINQ, použitých při tvorbě reportu rezervací. Dále zde proběhla analýza dodatečných frameworků, které ulehčili práci při psaní této podpůrné aplikace. To bylo zapracováno v teoretické podobě v kapitole 3.4 a jejich použití bylo reflektováno při implementaci aplikace.

Druhým cíl sestával z interview s majitelem hotelového systému a na základě jeho požadavků byla provedena analýza v kapitole 4.1.2 a technická analýza v kapitole 4.1.3.

Třetím cílem byla provedena implementace podpůrné aplikace na základě analýzy provedené v kapitole 4.1. Byl zde podrobně vysvětlen postup implementace s názornými příklady za použití syntézy frameworků z prvního cíle a jejich instalace. Byl zde vysvětlen postup založení základní struktury aplikace popsaný v kapitole 4.1 a instalace dodatečných frameworků za použití konzole pro instalaci balíčků popsané v kapitole 4.2. Následně zde byl uveden postup implementace přihlašování uživatelů za pomoci frameworků SecurityGuard a zapracování vstupního zobrazení, implementace databází a následné nastavení práv za pomocí Fluent Security. Toto bylo zapracováno formou podrobného postupu s obrázky v kapitole 4.3. Následně bylo uvedeno vytvoření hromadné emailové rozesílky a reportu. Hromadná emailová rozesílka byla zapracována v kapitole 4.5. Přehledný report rezervací byl zapracován v kapitole 4.6, bylo zde podrobně popsáno použití metodiky frameworku LINQ. Zároveň s tím zde byla zapracována názorná ukázka výstupu uživatelského zobrazení před aplikací Bootstrap Frameworku v kapitole 4.10 a po aplikaci tohoto frameworku v kapitole 4.12.

V posledním, tedy čtvrtém cíli bylo popsáno testování aplikace popsané v kapitole 4.13 a její následné nasazení na webový server, při použití tří různých typů nasazení v kapitole 4.14.

7 Seznam použitých zdrojů

1. ASP.NET MVC | The ASP.NET Site [online]. 2015-09-19. 2015 [cit. 2015-09-19]. Dostupné z: http://www.asp.net/mvc

2. GALLOWAY, Jon. Professional asp.net mvc 5. 1st edition. Indianapolis, IN: John Wiley and Sons, 2014, pages cm. ISBN 11-187-9475-3.

3. JOHNSON, Bruce. Professional visual studio 2013. Indianapolis, Indiana: Wrox, 2014, 1 online zdroj (1102 pages). ISBN 978-1-118-83205-9.

4. MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně. Vyd. 1. Brno: Zoner Press, 2011, 700 s. Encyklopedie Zoner Press. ISBN 978-80-7413-145-5.

5. Referenční dokumentace jazyka C# [online]. 2015 [cit. 2015-09-21]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/618ayhy6.aspx

6. KANJILAL, Joydip. *Entity Framework tutorial*. 2008. ISBN 1847195229.

7. PIALORSI, Paolo a Marco RUSSO. Microsoft LINQ: kompletní průvodce programátora. Vyd. 1. Brno: Computer Press, 2009. ISBN 978-80-251-2735-3.

8. ARH, Damir a Dejan DAKIC. NuGet 2 essentials. Birmingham: Packt Publishing, 2013. ISBN 978-1-78216-587-3.

9. *NuGet Gallery | Home* [online]. .NET Foundation, 2016 [cit. 2015-12-11]. Dostupné z: https://www.nuget.org/

10. *FluentSecurity : Fluent Security configuration for ASP.NET MVC* [online]. Kristoffer Ahl, 2016 [cit. 2016-01-19]. Dostupné z: http://www.fluentsecurity.net/

11. *Bootstrap · The world's most popular mobile-first and responsive front-end framework.* [online]. Twitter, 2016 [cit. 2016-02-01]. Dostupné z: http://getbootstrap.com/

12. Entity Framework Code First Migrations. *Výuka pro vývojáře na webu Microsoft Developer Network | MSDN* [online]. Microsoft, 2015 [cit. 2016-01-10]. Dostupné z: https://msdn.microsoft.com/en-us/data/jj591621

8 Přílohy

Příloha A CD s přiloženými soubory podpůrné webové aplikace a hotelové databáze