

# Webová nadstavba ERP systému

Bc. Marcel Machala

---

Diplomová práce  
2014



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2013/2014

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Marcel Machala**  
Osobní číslo: **A12644**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**  
Forma studia: **kombinovaná**

Téma práce: **Webová nadstavba ERP systému**

Zásady pro vypracování:

1. Analyzujte problematiku a zpracujte literární rešerši na dané téma.
2. Na základě požadavků navrhnete strukturu databáze.
3. Navrhnete a vytvořte funkční webovou aplikaci.
4. Věnujte pozornost zabezpečení databáze a aplikace.
5. Proveďte testování a vyhodnocení aplikace z hlediska bezpečnosti a optimalizace.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. RIORDAN, R. M. Vytváříme relační databázové aplikace. Praha : Computer Press, 2000. 280 s. ISBN 80-7226-360-9.
2. CONNOLLY, T., BEGG, C., HOLOWCZAK, R. Databáze – Profesionální průvodce tvorbou efektivních databází. Computer Press, 2009, ISBN: 978-80-251-2328-7.
3. ESPOSITO, Dino. ASP.NET a ADO.NET: tvorba dynamických webových stránek. 1. vyd. Praha: Grada, 2003, 352 s. ISBN 802-47-0474-9.
4. PROSISE, Jeff. Programování v Microsoft .NET: webové aplikace v .Net Framework, C# a ASP.NET. Vyd. 1. Brno: Computer Press, 2003, xxi, 712 s. ISBN 807-22-6879-1.
5. MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně. Vyd. 1. Brno: Zoner Press, 2011, 2 sv. (880, 700 s.). ISBN 978-80-7413-131-81.
6. LACKO, L'uboslav. 1001 tipů a triků pro SQL. Vyd. 1. Brno: Computer Press, 2011, 416 s. ISBN 978-80-251-3010-0.
7. CASTRO, Elizabeth a Bruce HYSLOP. HTML5 a CSS3: názorný průvodce tvorbou WWW stránek. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.

Vedoucí diplomové práce:

**doc. Ing. Zdenka Prokopová, CSc.**

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

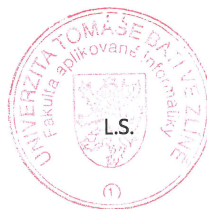
**21. února 2014**

Termín odevzdání diplomové práce:

**20. května 2014**

Ve Zlíně dne 21. února 2014

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



doc. Mgr. Roman Jašek, Ph.D.  
*ředitel ústavu*

### **Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### **Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 19. května 2014

.....  
podpis diplomanta

## **ABSTRAKT**

Diplomová práce se zabývá analýzou problematiky, návrhem struktury a vytvořením databáze, prostředky tvorby a samotnou tvorbou webové aplikace, která usnadní administraci školení a evidenci reklamací. Součástí aplikace je tzv. zákaznická zóna, kde jsou evidována vozidla zákazníka a jsou zde zobrazovány informace určené danému zákazníkovi. Aplikace je vytvořena třívrstvou architekturou MVC v jazyce ASP.NET za pomoci Entity Framework.

Klíčová slova:

MVC, Entity Framework, ASP.NET, SQL, databáze, HTML, Visual Studio, JavaScript, jQuery

## **ABSTRACT**

This dissertation occupies an analization of issues, including project of structure and creating of the database, devises of formation and its creation of web application, which facilitates administration of training and record of complaints. Part of the application is so called "customer zone" where the registered vehicles of the customers are displayed. The information is determined by the customer. Application is created by triple architecture MVC in ASP. NET language with help of Entity Framework.

Keywords:

MVC, Entity Framework, ASP.NET, SQL, database, HTML, Visual Studio, JavaScript, jQuery

Tímto chci poděkovat vedoucí mé diplomové práce paní doc. Ing. Zdence Prokopové, CSc. za cenné rady a připomínky při vedení mé práce. Zároveň chci poděkovat své manželce za významnou podporu po celou dobu studia.

# OBSAH

<b>ÚVOD</b> .....	<b>10</b>
<b>1 TEORETICKÁ ČÁST</b> .....	<b>12</b>
<b>1 WEBOVÉ APLIKACE</b> .....	<b>13</b>
1.1    STATICKE HTML STRÁNKY.....	13
1.1.1    Co to je HTML.....	13
1.1.2    Jak vytvořit HTML.....	14
1.1.3    Struktura HTML dokumentu.....	15
1.1.4    Řádkové značky HTML.....	15
1.1.5    Blokové značky HTML.....	17
1.1.6    Seznamy .....	17
1.1.7    Odkazy .....	18
1.1.8    Tabulky.....	19
1.1.9    Formuláře .....	19
1.1.10   JavaScript .....	20
1.2    DYNAMICKÉ HTML STRÁNKY .....	21
1.2.1    Skriptovací jazyky na straně serveru .....	21
<b>2 DATABÁZOVÉ SYSTÉMY</b> .....	<b>25</b>
2.1    DATABÁZE .....	25
2.2    SYSTÉM ŘÍZENÍ DATABÁZE (SŘBD) .....	25
2.3    DATABÁZOVÉ APLIKACE .....	25
2.4    NÁVRH DATABÁZE .....	26
2.5    VÝVOJ DATABÁZOVÝCH SYSTÉMŮ .....	26
2.5.1    Souborově orientované systémy.....	26
2.5.2    První generace SŘBD.....	27
2.5.3    Druhá generace SŘBD .....	28
2.5.4    Objektově orientovaný SŘBD – revoluční přístup.....	28
2.5.5    Objektově relační SŘBD – evoluční přístup.....	28
2.6    DATOVÉ MODELY .....	29
2.6.1    Logické modely.....	29
2.6.2    Modely fyzických dat .....	30
2.7    RELAČNÍ MODEL.....	30
2.7.1    Klíče relačních tabulek.....	30
2.8    NORMALIZACE .....	31
2.8.1    První normální forma (1NF) .....	31
2.8.2    Druhá normální forma (2NF) .....	32
2.8.3    Třetí normální forma (3NF) .....	32
<b>3 SQL</b> .....	<b>33</b>
3.1    MANIPULACE S DATY .....	33
3.1.1    SELECT .....	33
3.1.2    INSERT .....	34

3.1.3	UPDATE .....	35
3.1.4	DELETE.....	35
<b>4</b>	<b>BEZPEČNOST .....</b>	<b>36</b>
4.1	INJECTION.....	36
4.1.1	SQL Injection .....	36
4.1.2	LDAP Injection .....	37
4.1.3	OS Injection.....	37
4.2	KRÁDEŽ IDENTITY .....	38
4.2.1	Spočítat či uhádnout, brute-force attack.....	38
4.2.2	Odchytit existující SID.....	38
4.2.3	Vygenerovat vlastní SID a podstrčit ho oběti.....	39
4.3	CROSS-SITE SCRIPTING (XSS).....	39
4.4	CROSS-SITE REQUEST FORGERY (CSRF).....	39
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>41</b>
<b>5</b>	<b>WEBOVÉ STRÁNKY .....</b>	<b>42</b>
5.1	TECHNOLOGIE .....	42
5.2	VLASTNÍ TVORBA STRÁNEK.....	43
5.2.1	Potřebný software.....	43
5.2.2	Nový projekt.....	44
5.2.3	Náš projekt .....	46
<b>6</b>	<b>WEBOVÁ APLIKACE.....</b>	<b>63</b>
6.1	ŠKOLENÍ.....	63
6.1.1	Vytvoření školení .....	63
6.1.2	Vytvoření firmy .....	64
6.1.3	Migrace.....	66
6.1.4	Přihlášení na školení .....	68
6.1.5	Registrace do vlastní aplikace .....	72
6.1.6	Přihlášení.....	75
6.1.7	Správa uživatelských informací .....	76
6.1.8	Přihlášení na školení registrovaného uživatele .....	76
6.1.9	Odhlášení se ze školení .....	78
6.2	KONTAKTY .....	79
6.2.1	Mapa.....	79
6.2.2	Metoda Dealers() kontroleru NdSafController.....	80
6.3	EVIDENCE VOZIDEL .....	81
6.3.1	Seznam vozidel .....	82
6.3.2	Podrobnosti o vozidle.....	83
6.3.3	Smazání vozidla .....	84
6.3.4	Vložení nového vozidla.....	85
6.3.5	Editace vozidla .....	87
6.4	REKLAMACE.....	87
6.4.1	Seznam reklamací .....	89
6.4.2	Podrobnosti reklamace .....	89

6.4.3	Vložení nové reklamace .....	91
6.4.4	Editace a smazání reklamace.....	92
6.5	ZÁKAZNICKÁ ZÓNA .....	92
6.6	ADMINISTRAČNÍ ROZHRANÍ.....	93
6.6.1	Metoda Index.....	93
6.6.2	Tisk prezenčního listu .....	94
6.6.3	Rotativa .....	95
6.6.4	Metoda Uživatele .....	95
6.6.5	Metoda Vozidla.....	97
6.6.6	Metody Skoleni, Reklamace a Firmy .....	98
<b>7</b>	<b>BEZPEČNOST .....</b>	<b>99</b>
7.1	SQL INJECTION .....	99
7.2	KRÁDEŽ IDENTITY .....	99
7.3	CROSS-SITE SCRIPTING (XSS).....	99
7.4	CROSS-SITE REQUEST FORGERY (CSRF).....	100
	<b>ZÁVĚR .....</b>	<b>101</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>103</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>105</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>107</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>109</b>

## ÚVOD

V dnešní elektronické době se již stalo téměř nutností, aby velkoobchodní firma, která nabízí nějaké služby, měla svoji internetovou prezentaci. Pokud chce ale nějakým způsobem vyčnívat, musí jít dál – stát se pro stávající a současně i potenciální zákazníky nějak zajímavější, přitažlivější, nabídnout přidanou hodnotu - něco víc než nabízí ostatní na svých webových stránkách. Není tím myšlen e-shop, který je v některých odvětvích více na škodu než k užítku, protože nutí zákazníka vědět a učit se věci, které nechce a nepotřebuje ke svému byznysu. Zákazník chce najít v internetové prezentaci informace, které potřebuje ke svému rozhodnutí. Potřebuje možnost, aby mohl vyřešit svůj problém, když nastane.

Firma SAF-HOLLAND Czechia spol. s r.o., pro kterou je aplikace vytvářena, je dceřinou společností německo-amerického koncernu SAF-HOLLAND. Tento koncern patří k vedoucím výrobcům komponentů pro výrobu návěsů a přívěsů. Mimo jiné vyrábí nápravy a pérování s nosností nad 6,5 t, odstavné nohy a královské čepy pro návěsy, kuličkové dráhy pro přívěsy, a sedlové točny pro tahače. Tyto komponenty dodává výrobcům přímo výrobní závod. Každý výrobek má své sériové číslo, podle kterého je možné ho v koncernovém systému dohledat a zjistit, z jakých dílů se skládá, apod. Výrobce vyrobí vozidlo (návěs, přívěs, ...) a do svých záznamů může (a většinou to i dělá) zavést sériová čísla komponent, která na dané vozidlo namontoval. Tuto informaci ale nesdílí dál se zákazníkem, ani s dodavatelem komponent.

Nejvýznamnějšími zákazníky firmy SAF-HOLLAND Czechia jsou specializované obchodní sítě, servisní stanice a dopravní firmy. Tito zákazníci si často zavádějí vlastní evidenci vozidel, ale ukládají do ní informace pouze z technického průkazu vozidla, a to není pro identifikaci náhradních dílů komponentů koncernu SAF-HOLLAND dostatečné, protože o těchto komponentech se v technickém průkazu nepíše vůbec nebo velmi málo.

Když zákazník shání náhradní díl, může být vozidlo již staré a označení komponent již nemusí být zřetelné. Proto bylo navrženo, aby vznikla aplikace, kde budou moci uživatelé vést evidenci vozidel, tak aby bylo možné identifikovat náhradní díly komponentů koncernu SAF-HOLLAND.

Tato evidence bude samozřejmě sloužit i firmě, která tak získá přehled o skladbě svých produktů u zákazníků, s možností tvorby individuálních nabídek, apod.

Dále bude aplikace obsahovat on-line formulář pro přihlášení na školení, které firma SAF-HOLLAND Czechia pravidelně pořádá. Součástí administrační prostředí bude možnost tisku prezenční listiny. Nyní se veškerá evidence vede v Excelu, který je ve sdíleném úložišti a může tak dojít k nekonzistentním úpravám. Navíc se údaje několikrát přepisují, takže může dojít k chybě.

Dále aplikace bude obsahovat on-line formulář na řešení reklamací. V současné době probíhá reklamační vyplňování papírového formuláře s několikerým ručním přepisováním do různých systémů. Při tomto ručním přepisování může dojít k zanesení chyby, která se táhne celým případem a v konečném důsledku může vést k zamítnutí reklamační, která by v případě, že by bylo všechno přepsáno bez chyb, byla uznána. Dalším faktorem pro zavedení on-line formuláře je možnost validace zadaných dat ještě v průběhu jejich zadávání. Zákazník tak může mít jistotu, že zadal všechny nutné informace, pro správné vyřízení celé reklamační.

V teoretické části této práce se seznámíme se základními možnostmi tvorby webových aplikací, se základy databází a jejich využitím u webových aplikací a principy zabezpečení webových aplikací a databází.

V praktické části bude popsán postup primárního cíle této práce – tvorba webové aplikace, jak je uvedeno výše.

## **I. TEORETICKÁ ČÁST**

## 1 WEBOVÉ APLIKACE

Webové (internetové) aplikace jsou takové aplikace, které využívají pro komunikaci s uživatelem okno webového prohlížeče. Tyto aplikace pracují v režimu klient-server, a začaly ve velkém nahrazovat aplikace pracující ve stejném režimu, které používaly pro komunikaci s uživatelem vlastní okno klienta. Tento klient se označuje jako tlustý klient, protože obsahuje logiku a zpracovává data přijaté ze serveru. Naproti tomu se prohlížeč označuje jako tenký klient. Neobsahuje totiž žádnou logiku a datům ze serveru nerozumí. Server musí všechny data zpracovat a připravit pro prohlížeč.

### 1.1 Statické HTML stránky

Základem pro zobrazování informací (textu, obrázků,...) v prohlížeči jsou statické HTML stránky. Říká se jim statické, protože zobrazují vždy přesně to, co je uloženo v html souboru na serveru.

Proto, aby prohlížeč zobrazil data ve formátu, který chceme, musíme je opatřit formátovacími značkami, tzv. tagy. Tyto značky říkají prohlížeči, jak má text zobrazit, případně umožňují vložit do textu obrázek nebo obrázek. Jsou to vlastně příkazy jazyka HTML.

#### 1.1.1 Co to je HTML

HTML (HyperText Markup Language) je „programovací“ jazyk internetových stránek. Navrhl ho v roce 1990 Tim Berns-Lee při práci na informačním systému pro CERN, zároveň napsal první webový prohlížeč, který nazval *WorldWideWeb*. V roce 1991 CERN zprovoznil svůj první web.

V říjnu 1994 založil Tim Berns-Lee při MIT (Massachusetts Institute of Technology) World Wide Web Consortium (W3C), které má pomáhat rozvíjet protokoly a zajistit dlouhodobý růst webu.

HTML je aktuálně ve verzi 5, která má oproti minulým verzím velkou podporu multimédií.

### 1.1.2 Jak vytvořit HTML

- **Jakýkoliv textový editor**

HTML stránku můžeme napsat v jakémkoliv textovém editoru, např. Poznámkový blok (Notepad). Html soubor je totiž jednoduchý textový soubor, který mimo zobrazovaného textu obsahuje i tagy jazyka HTML. Důležité je dokument uložit jako čistý text s příponou *\*.html*. Není vhodné použít editor, který text formátuje, jako je MS Word nebo OpenOffice Writer, protože tyto editory přidávají do souboru svoje formátovací značky, které internetový prohlížeč nedokáže interpretovat.

Psaní internetových stránek tímto způsobem je tím těžší, čím je stránka složitější. Neexistuje tady žádná pomůcka, která by pomohla psaní HTML souboru.

- **WYSIWYG Editor**

Další možností jak vytvořit HTML stránku je použití WYSIWYG (What you see is what you get – co vidíš, to dostaneš) editoru. Tento editor umožňuje tvorbu webové stránky stejně jako dokument ve Wordu. Píšeme text, pomocí nástrojů ho formátujeme, můžeme vložit obrázek nebo tabulku. Editor automaticky tvoří zdrojový kód.

Tímto způsobem dokáže vytvořit vzhledné stránky opravdu každý. Velkým problémem takto vytvořeného kódu je nekvalita kódu a jeho problematická kontrola. Zdrojový kód bývá často nekvalitní a obsahuje mnoho duplicit.

- **Speciální vývojové prostředí - IDE**

Tyto vývojové prostředí (IDE - Integrated Development Environment) umožňují psát kód stejně jako textové editory, ale na rozdíl o nich obsahují různé pomůcky pro zvýraznění syntaxe, použití tagů HTML. Umí zobrazit náhled rozepsané HTML stránky, apod.

Existuje velké množství speciálních IDE, některé jsou specializované pro psaní pouze v jednom jazyce, jiné jsou univerzální pro více jazyků. Některá IDE jsou zdarma, tzv. freeware, jiné jsou komerční – za jejich použití je nutné zaplatit vývojářům.

Za všechny jmenujme např. Eclipse, NetBeans, PSPad (český), Microsoft Visual Studio Express for Web. Každý program má své klady a zápory.

### 1.1.3 Struktura HTML dokumentu

Kód HTML dokumentu se dělí na dvě hlavní části: hlavičku a tělo. Celý kód je uzavřený mezi tagy `<html>` a `</html>`. Mimo tyto tagy je pouze první řádek s tagem `<!DOCTYPE>`, který uvádí specifikaci DTD, která je podle standardu W3C povinná, ale prohlížeče umí stránku zobrazit i bez tohoto tagu.

- **Hlavička**

Hlavička je část html souboru mezi tagy `<head>` a `</head>`. Hlavička může obsahovat několik nepovinných tagů, a to:

- `<title>Nadpis</title>` - nastavuje „Nadpis“ jako titulek stránky
- `<base>` - nastavuje základní cestu pro relativní odkazy ve stránce
- `<link>` - slouží ke spojení s jiným souborem, např. CSS stylem. Má několik atributů (`rel`, `href`, `type`,...)
- `<meta>` - obsahuje informace o stránce, např. kódování znaků
- `<script>` - slouží k připojení externího skriptu (např. JavaScript)

Příklad kompletní hlavičky:

```
<head>
<meta http-equiv="content-type" content="text/html;
      charset=windows-1250">
<title>Titulek stránky</title>
<link rel="stylesheet" href="styly/muj-styl.css" type="text/css">
<script src="skripty/muj-skript.js"></script>
</head>
```

[8]

- **Tělo**

Tělo dokumentu je část html souboru mezi tagy `<body>` a `</body>`. Sem zapisujeme text, který chceme zobrazit na stránce a formátovacími tagy. Tag `<body>` může obsahovat nepovinné parametry (např. `bgcolor`, `background`, `link`, `vlink`, ...), které nastaví formáty celého dokumentu. Dnes se nedoporučují používat, a nahrazují se použitím CSS vlastností.

### 1.1.4 Řádkové značky HTML

Řádkové značky poznáme podle toho, že jejich ukončovací tag nezalamuje text na nový řádek. Jsou to zejména značky, které formátují text. Formátování textu na webových stránkách HTML můžeme rozdělit na dvě skupiny:

- Fyzické formátování – používá se pro nastavení vzhledu, tj. velikosti, barvy, dekorace, apod.
  - Logické formátování – používá se pro logické dělení stránky podle obsahu
- Patří sem i tag `<span> </span>`, který sám text nijak neformátuje, ale takto označený text můžeme lehce formátovat pomocí CSS a JavaScriptu.

- **Fyzické formátování**

Nastavit vzhled můžeme opět dvěma způsoby (případně oběma najednou):

- Pomocí tagů – např. text v kurzívě vypíšeme tagy `<i>text</i>`
- Pomocí CSS – používá se tag `<style>` nebo atribut `style`

Pro fyzické formátování využíváme například tyto značky: `<b></b>` pro tučný text, `<i></i>` pro kurzívu, `<strike></strike>` pro přeškrtnuté písmo, `<sub></sub>` pro dolní index, `<sup></sup>` pro horní index, `<u></u>` pro podtržené písmo, a mnohé další.

Všechny tyto značky se dají nahradit kaskádovými styly.

Formátování pomocí tagů je již zastaralé a má mnohá omezení oproti použití CSS, proto se dále budu věnovat pouze formátování pomocí stylů CSS. CSS je zkratka Cascading Style Sheets (česky „kaskádové styly“).

CSS poprvé implementoval Microsoft do svého prohlížeče Internet Explorer 3.0 v roce 1996. Pomocí CSS můžeme definovat písma, barvy, velikost, ale kromě toho třeba vzhled rámečků, odrážek, velikost okrajů, přesné pozice prvků, atd.

CSS definuje vzhled pro celý dokument, můžeme tak na jednom místě rychle a jednoduše změnit vzhled celého webu. CSS dokáže změnit chování a vzhled i standardních tagů HTML.

Pokud použijeme JavaScript můžeme dynamicky měnit styly jednotlivých komponent (CSS a JavaScript se propojí pomocí identifikátoru „id“). [8]

- **Logické formátování**

Tagy logického formátování formátují text nejen fyzicky, ale přiřazují mu logický význam. Pro lepší pochopení můžeme porovnat tagy `<b>` a `<strong>`. Pokud pomocí CSS nezměníme výchozí formátování, oba tagy řeknou prohlížeči, že má text zobrazit tučným

fontem. Tag `<strong>` navíc ale řekne prohlížeči, že se jedná o něco, co má být zvýrazněno. Pokud tedy prohlížeč neumí zobrazit tučný text, tag `<b>` zůstane bez povšimnutí, ale tag `<strong>` se prohlížeč pokusí zvýraznit třeba jinou barvou, případně jinak, bude-li se stránka interpretovat jinak než vizuálně.

Jedná se o tyto tagy: `<abbr>` (zkratka, její vysvětlení se zapisuje do parametru *title*), `<acronym>` (zkratka, která se nečte po písmenech, ale jako slovo), `<cite>` (řádková citace, většinou se vykresluje kurzívou), `<code>` (úsek kódu, vypisuje se neproporcionálním písmem), `<del>` (odstraněný text, vykresluje se přeškrtnutým fontem), `<ins>` (přidaný text, vykreslí se jako podtrhnutý), `<kdb>` (zápis z klávesnice, neproporcionální font), `<samp>` (výstup z programu, taktéž neproporcionální font), `<span>` (vytváří blok textu, kterému lze přidělit styl pomocí atributů *style*, *id* nebo *class*; sám nijak neformátuje text), `<strong>` (zvýraznění, tučné písmo), `<q>` (citace, atribut *cite* obsahuje zdroj citace) a `<var>` (proměnná, kurzíva).

Tyto značky nemají alternativu v CSS a není možné je tímto způsobem nahradit. [8]

### 1.1.5 Blokové značky HTML

Blokové značky se vyznačují tím, že jejich ukončovací tag zalomí text na další řádek. Mezi blokové značky můžeme zařadit tagy: `<p>` (odstavec), `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` (nadpisy), `<br />` (přechod na nový řádek), `<div>` (blokový `<span>` - důležitý pro tvorbu layoutu stránky), `<hr />` (vodorovná čára), `<pre>` (zobrazuje všechny zapsané znaky i všechny mezery, řádkování, tabulátory) a `<blockquote>` (víceřádková citace). [8]

### 1.1.6 Seznamy

Seznam je několik položek, které se zapisují do řádků pod sebou. Seznam může být číslovaný nebo odrážkový. I když si seznam představujeme jako řádky pod sebou, je možné ho pomocí CSS změnit na seznam vedle sebe na řádek. Tohoto se často využívá pro vytvoření základního menu.

- **Číslovaný seznam**

Číslovaný seznam se vytváří tagem `<ol> </ol>`. Tento tag má volitelné parametry *start*, který nastavuje počáteční číslo seznamu a *type*, který nastavuje typ číslování („A“ – velká

abeceda, „a“ – malá abeceda, „i“ – malé římské číslice, „I“ – velké římské číslice a „1“ – arabské číslice). Každá položka seznamu musí být uzavřena tagem `<li> </li>`.

- **Odrážkový seznam**

Odrážkový seznam se vytváří tagy `<ul>` a `</ul>`. Položky seznamu jsou stejně jako u číslovaného uzavřeny tagem `<li> </li>`. Atributem *type* tagu `<ul>` je možné nastavit typ odrážky. Máme možnosti „square“ (čtvereček), „circle“ (kolečko) a „disc“ (puntík). Nebo můžeme použít „none“ pro seznam bez odrážek. Pomocí CSS lze seznam ještě vylepšit – můžeme nastavit obrázek místo odrážek. [8]

- **Vnořené seznamy**

Jednotlivé seznamy je možné do sebe nezávisle vnořovat, a to tak, že mezi tagy `<ol> </ol>` nebo `<ul> </ul>` vložíme další tagy `<ol> </ol>` či `<ul> </ul>`. Oba druhy seznamů můžeme jakkoliv střídat, pouze musíme dbát na správné ukončování, tzn. seznam otevřený jako poslední musíme jako první uzavřít. [8]

- **Seznam definic**

Dalším typem seznamu je seznam definic. Zapisuje se mezi tagy `<dl> </dl>`. Seznam definic tvoří dvojice tagů `<dt> </dt>` (mezi tyto tagy zapisujeme **pojem**) a `<dd> </dd>` (mezi tyto tagy zapisujeme **definici pojmu**). [8]

### 1.1.7 Odkazy

Odkazy dělají internet internetem. Pomocí nich je možné pohybovat se mezi jednotlivými stránkami. Tato jednoduchost v prvopočátku zaujala uživatele a stála za masivním rozvojem internetu.

Odkaz vytvoříme tagy `<a> </a>`. Text odkazu napíšeme mezi tyto tagy. Je možné místo textu použít obrázek. Tag `<a>` má atribut *href*. Do tohoto atributu zapisujeme cíl odkazu. Cílem odkazu může být absolutní adresa ([www.seznam.cz](http://www.seznam.cz)), může to být relativní adresa na stejném webu (`../index.html`) nebo odkaz na stejné stránce (`#top`). [8]

### 1.1.8 Tabulky

Tabulky se v dřívějších dobách využívaly k vytvoření layoutu stránky. Tento postup je dnes zavrhován a nahrazuje se vytvořením layoutu pomocí elementů `<div>`. Tabulky se používají k tomu, k čemu byly původně určeny, tj. k reprezentaci dat.

Tabulka se vytvoří tagy `<table>` `</table>`. Vlastnosti tabulky je možné definovat atributy (zastaralý způsob – nedoporučuje se) nebo pomocí stylů CSS. Každá tabulka se skládá z jednoho nebo více řádků. Řádek se zapisuje mezi tagy `<tr>` `</tr>`. Každý řádek se skládá z jedné nebo více buněk. Obsah buňky zapíšeme mezi tagy `<td>` `</td>`. Buňky můžeme slučovat po sloupcích pomocí atributu *colspan* (a) nebo po řádcích atributem *rowspan*.

V prvním řádku tabulky můžeme místo tagů `<td>` `</td>` použít tagy `<th>` `</th>`. Obsah mezi těmito tagy je považován za hlavičku tabulky a bude formátován jiným stylem.

Mezi tagy `<caption>` `</caption>` můžeme zapsat nadpis tabulky. [8]

### 1.1.9 Formuláře

Formulářem nazýváme část webové stránky, kterou může uživatel vyplnit a odeslat nám ke zpracování. Aby bylo možné data zpracovat, je nutné zajistit, aby na serveru byl existoval skript, který zaslané data dokáže analyzovat a zpracovat. O možnostech skriptování na straně serveru si povíme v další části, která se věnuje dynamickým HTML stránkám.

Formulář je část kódu mezi tagy `<form>` `</form>`. Tag `<form>` má několik atributů, povinný je však jenom jeden, a to *action*. Do tohoto atributu se zapisuje adresa skriptu, který pracovává data z formuláře. Dalším atributem je *method*. Tento atribut může nabývat dvou hodnot: GET nebo POST. Jedná se o metodu zpracování. Při metodě POST se data posílají „skrytě“ v hlavičce stránky. Metodou GET se data posílají v adrese stránky, takže jsou vidět v adresním řádku prohlížeče. Metoda GET je výchozí, takže se použije, pokud parametr *method* nezadáme explicitně.

Do formuláře vkládáme různé komponenty, do kterých může uživatel něco zapsat (textové pole) nebo vybrat z nabízených možností (rozevírací seznam, checkboxy, skupina radiobuttonů, ...).

Tyto komponenty můžeme samozřejmě použít i mimo formulář, ale v tom případě se server nedoví, co uživatel udělal. Je možné změnu těchto prvků zachytit pomocí JavaScriptu a adekvátně zareagovat, aniž by bylo nutné informovat o tom server.

Formulář musí vždy obsahovat odesílací tlačítko. Bez tohoto tlačítka by nebylo možné zavolat ovládací skript na serveru a odeslat data. Tlačítko vytvoříme tagem `<input />` s parametrem `type` nastaveným na hodnotu „submit“. Dalším parametrem je `value`, který obsahuje nápis zobrazený na tlačítku. Formulář může obsahovat tlačítko, které vytvoříme parametrem `type` s hodnotou „reset“. Toto tlačítko vymaže všechny uživatelem zadané hodnoty ve formuláři, případně je nastaví na výchozí hodnoty.

### 1.1.10 JavaScript

JavaScript je programovací jazyk, který umožňuje provádění svého kódu přímo v prohlížeči. Zapisuje se přímo do kódu HTML, případně se v HTML kódu uloží odkaz na soubor, který JavaScriptový kód obsahuje.

Často se zaměňuje JavaScript a Java. Sice jsou to oba multiplatformní objektově orientované jazyky a mají podobnou syntaxi, ale tím podobnost končí. Jazyk Java vyvinula společnost Sun Microsystems v roce 1995. JavaScript byl napsán Brendanem Eichem ze společnosti Netscape. Ohlášen byl ve stejném roce jako Java, ale o pár měsíců později. Existují různé implementace, například ActionScript (Macromedia Flash) nebo JScript (Microsoft .NET).

JavaScript se dá použít jednoduše pro zobrazování a skrývání obsahu, ale i složitěji pro načítání dat ze serveru na pozadí a dynamicky měnit zobrazenou stránku.

Pro usnadnění práce vzniklo velké množství knihoven JavaScriptu. Mezi nejznámější patří *jQuery* (<http://jquery.com>), *MooTools* (<http://mootools.net>), *YUI* (<http://yuilibary.com>).

JavaScript je interpretovaný jazyk, tzn. že kód není nutné před prováděním kompilovat. Využívá objekty, které obsahuje prohlížeč i objekty HTML kódu. Závisí na prohlížeči – většina prohlížečů JavaScript podporuje, ale je možné ho uživatelsky zakázat. Navíc, každý prohlížeč může interpretovat kód trochu jinak. Tento problém se snaží řešit zmíněné knihovny, které se snaží dosáhnout maximálně konzistentního chování ve všech prohlížečích.

Největším omezením JavaScriptu, které je nejdůležitějším zabezpečením, je že nemůže přistupovat k lokálním souborům, ani je nemůže zapisovat. Toto zabezpečení je implemen-

továno, protože JavaScript se spouští automaticky po načtení stránky, a pokud by se načel nebezpečný kód, automaticky by se spustil a mohl provádět záškodnickou činnost.

- **Externí JavaScript**

Externím JavaScriptem nazýváme takový skript, který se načítá z externího souboru pomocí tagu `<script type="text/javascript" src="skript.js">`  
`</script>`. Často se tento kód píše mezi tagy `<head>` a `</head>`, tedy do hlavičky. Tento postup, pokud to není nevyhnutelně nutné, je považován za nevhodný, protože pokud prohlížeč načítá a provádí kód JavaScriptu, nedělá nic jiného, tj. nenačítá kód ani nezobrazuje text. Proto je nevhodnější toto načítání kódu napsat až na poslední řádek, těsně před ukončovací tag `</body>`. Toto řešení nelze použít, pokud se JavaScript využívá již pro první zobrazení stránky.

- **Vložený JavaScript**

Vložený JavaScript je takový, který vkládáme přímo do HTML kódu. Kód JavaScriptu vložíme mezi tagy `<script type="text/javascript">` a `</script>`. Tento kód můžeme vložit kamkoliv do HTML kódu, musíme mít ale na zřeteli, že kód skriptu se bude načítat v průběhu provádění HTML kódu. Pokud bude kód JavaScriptu dlouhý, případně náročný, přestane prohlížeč vykonávat HTML kód a nebude pokračovat ve vykreslování stránky.

## 1.2 Dynamické HTML stránky

Dynamické stránky jsou takové, které nejsou uloženy na serveru přímo ve formátu HTML, ale jako skript, který generuje kód pro prohlížeč až na základě aktuálního stavu prohlížeče a požadavku uživatele. Stránka se stejnou adresou se tak může dvěma uživatelům zobrazit úplně rozdílně. Stránky často obsahují data, která jsou uložena v relačních databázích.

### 1.2.1 Skriptovací jazyky na straně serveru

V dřívějších dobách běžely na serverech téměř výhradně CGI skripty v Perlu. V současnosti je nejběžnějším jazykem PHP, kterému sekunduje jazyk ASP.NET. Dalším používaným je JSP.

Tyto skripty fungují tak, že uživatel zadá adresu skriptu, který vygeneruje HTML stránku, kterou pošle prohlížeči.

- **JavaServerPage – JSP**

Jak název napovídá, jedná se jazyk založený na jazyce Java. Vyvinula jej společnost Sun Microsystems v roce 1999.

Kód jazyka JSP se zapisuje mezi tagy `<% a %>`. Soubory mají koncovku `*.jsp`.

JSP lze použít jako samostatné stránky nebo jako „View“ v designu Model-View-Controller. Model obvykle reprezentuje JavaBeans a Java servlety (např. Apache Struts) jako Controllery. Tato architektura se označuje jako Model2.

- **PHP**

Jazyk PHP byl vydán v roce 1996. Zkratka PHP je rekurzivní zkratka PHP: Hypertext Pre-processor. Její původní význam byl Personal Home Page.

PHP je nejčastěji využívaný serverový skriptovací jazyk. Je to především kvůli licenční politice, která je velmi otevřená – systém je zdarma, je multiplatformní a podporuje mnoho různých databázových systémů. Navíc je na Internetu volně k dispozici nespočetně tutoriálů a učebních textů.

PHP je vhodné pro malé projekty (jak vyplývá z původního významu zkratky), stejně jako pro obrovské projekty jako jsou webové aplikace – *Wikipedia*, *Wordpress*, *eBay*, a další.

Kód jazyka PHP se zapisuje do tagu `<?php kód_PHP ?>`.

Jazyk PHP je slabě typovaný, tzn. že nevyžaduje striktně zadaný typ proměnné. Obsahuje automatické přetypování.



Obr. 1 Logo PHP

Pro PHP existuje nepřehledné množství frameworků, které usnadňují programování. Mají například nadefinované ošetření vstupů, připravené příkazy pro jednoduchou práci s databází. Framework můžeme definovat jako soubor knihoven, podpůrných programů, různých API,...

Mezi nejznámější frameworky pro PHP patří Zend Framework ([www.framework.zend.com](http://www.framework.zend.com)), CakePHP ([www.cakephp.org](http://www.cakephp.org)), český framework Nette ([www.nette.org](http://www.nette.org)), Symfony ([www.symfony.com](http://www.symfony.com)), ...

- **ASP.NET**

Jazyk ASP.NET je nástupcem jazyka ASP. I když je jeho název odvozený od jeho předchůdce, mají spolu společnou jen část názvu. Základem ASP.NET je .NET Framework stejně jako jazyka C#. První verze ASP.NET byla uvolněna v roce 2002.

Obrovskou nevýhodou jazyka ASP.NET je jeho platformní závislost. Pro jeho běh je nutný Microsoft IIS Server, který běží pouze pod Microsoft Windows. Existuje sice projekt Mono, který multiplatformní a podporuje i .NET Framework. Ale funkčnost je stále hodně diskutabilní.

Možná z tohoto důvodu není ASP.NET rozšířený stejně jako PHP, i když kvalitativně mezi nimi asi nejsou velké rozdíly. Někdo tvrdí, že ASP.NET se nehodí pro menší projekty, ale dle mého názoru to není pravda. ASP.NET je vhodný pro malý osobní web úplně stejně jako PHP.

Stejně jako PHP je na webu spousta tutoriálů, ale jsou převážně v angličtině. Dále se mnohem obtížněji hledají free hostingy s podporou ASP.NET.

ASP.NET je postaven celý na .NET Frameworku, proto není možné ho srovnávat s čistým PHP, i když několik pokusů již proběhlo. Pro ASP.NET existuje framework ASP.NET MVC, který umožňuje vytvářet weby podle architektury MVC (Model-View-Controller). Druhou možností jak psát web jazykem ASP.NET je tzv. WebForms. Tato metoda se podobá psaní desktopové aplikace.



Obr. 2 Logo MS .NET MVC 5

Dalším často používaným frameworkem je EntityFramework, který usnadňuje použití databáze v projektech.

Správněji řečeno ASP.NET není programovací jazyk, ale platforma. Jazykem této platformy je Visual Basic nebo dnes častěji Visual C#. Skript má tvar HTML stránky s vloženými částmi kódu v dané syntax.

## 2 DATABÁZOVÉ SYSTÉMY

Databázový systém je tvořený databází (nebo databázemi) a systémem řízení databáze. Některé zdroje ještě přidávají i databázové aplikace, ale dle mého názoru tyto do databázového systému nepatří, protože databázový systém bude fungovat i bez nich.

### 2.1 Databáze

V současnosti je databáze, jednoduše řečeno, úložiště perzistentních dat. Tato data jsou dostupná současně mnoha uživatelům. Kromě samotných dat obsahuje databáze i popis těchto dat. Proto je databáze také definována jako *sebepopisující kolekce integrovaných záznamů*. Tato vlastnost databází poskytuje tzv. nezávislost dat – přidáme-li do existující databáze nové struktury, nebo existující změníme, nijak to neovlivní databázové aplikace (pokud přímo nepracují s tím, co bylo změněno). [2]

### 2.2 Systém řízení databáze (SŘBD)

Systém řízení databáze (angl. DataBase Management System – DBMS) je software, který komunikuje s uživateli, databázovými aplikacemi a se samotnou databází. SŘBD umožňuje uživatelům a aplikacím vkládat, mazat, upravovat a vyhledávat data v databázi, většinou pomocí dotazovacího jazyka, např. SQL (Structured Query language).

Mezi nejznámější SŘBD patří MySQL, MS SQL Server a Oracle.



Obr. 3 Logo MS SQL Serveru



Obr. 4 Logo MySQL

### 2.3 Databázové aplikace

Databázová aplikace je program, který komunikuje s databází vyvoláním odpovídajícího požadavku pro SŘBD. [2]

Uživatelé pracují s databází prostřednictvím databázových aplikací, například informační systém, ve kterém uživatel vystaví fakturu, uloží několik záznamů do databáze. Uživatel vlastně ani nemusí vědět, že pracuje s databází.

Databázové aplikace lze psát v programovacích jazycích třetí generace (C++, C#, Java,...) nebo v některém vyšším jazyce čtvrté generace. V dnešní době se čím dál častěji jedná o webové aplikace.

## 2.4 Návrh databáze

Během návrhu databáze vzniká struktura databáze. Návrh databáze pro funkčnost systému klíčový. Špatně navržená databáze bude vytvářet chyby a vracet chybné informace. Při špatném návrhu databáze může být databázová aplikace sebelepší, ale bude zobrazovat data. Musíme tedy myslet nejprve na data a na aplikaci až v druhé řadě.

## 2.5 Vývoj databázových systémů

### 2.5.1 Souborově orientované systémy

Tyto systémy sahají zpět do 60. let minulého století. Byly prvními pokusy zpracovat manuální kartotékový systém, který je uspokojivým řešením, pokud je malý počet položek, nebo velký počet položek, které nemusíme nijak zpracovávat nebo nepotřebujeme křížové odkazy. Toto křížové hledání tvoří jádro dnešních databázových systémů.

Souborově orientované systémy mají mnoho nevýhod:

- *Programově-datová závislost* – Datové struktury jsou vloženy do aplikačních programů. Jakákoliv změna v datech si vyžádá změnu celé aplikace, provádění změn je složité a často zdrojem chyb.
- *Duplikace dat* – Každé oddělení firmy používá vlastní datové aplikace a vlastní datové struktury, a i když používají stejné entity, zadává si je každé oddělení zvlášť. Duplikace je často nežádoucí, protože vede k plýtvání (čas, peníze, úložný prostor), může vést ke ztrátě integrity (data nejsou konzistentní).
- *Oddělení a izolace dat* – Když jsou data izolována v oddělených souborech, je obtížnější je v kritický okamžik zpřístupnit. Pro dodání určité informace musí být přístupných několik souborů.

- *Fixní dotazy* – Každý dotaz musí být zapracovaný do aplikace. To vede ke dvěma věcem. Jednou byl dotaz pevný a nebyla možnost neplánovaných dotazů, podruhé došlo k enormnímu nárůstu souborů a aplikačních programů.
- *Omezené sdílení dat* – Protože má každé oddělení firmy své souborové systémy, je sdílení dat omezené. Vedení firmy nemůže získat celkový obraz o jejím stavu.

Všechny tyto nevýhody řeší nasazení SŘBD. [2]

### 2.5.2 První generace SŘBD

Hierarchický a síťový přístup označujeme jako *první generaci SŘBD*. Tato generace má zásadní nevýhody:

- Je třeba napsat složité programy i pro zodpovězení jednoduchých dotazů k jednomu záznamu.
- Mají minimální datovou závislost, takže aplikace nejsou chráněny před změnami formátu dat.
- Neexistuje široce přijímaný teoretický základ.

- **Hierarchické SŘBD**

První hierarchický SŘBD byl IMS (Information Management System) firmy IBM (původně byl vyvinut pro program Apollo), který uchovává záznamy pomocí invertované stromové struktury, která je výhodná při použití sekvenčních paměťových zařízení, zejména magnetických pásek.

IMS stále patří mezi nejrozšířenější databázové systémy sálových počítačů.

- **Síťové SŘBD**

Firma General Electric vyvinula SŘBD IDS (Integrated Data Store) pro vytvoření databázového standardu CODASYL (CONference on DATA SYstem Languages). Tento systém řeší potřebu reprezentace komplexnějších datových struktur, než které bylo možné modelovat pomocí hierarchických struktur.

Nejpoužívanějším síťovým SŘBD je IDMS/R firmy Computer Associates.

### 2.5.3 Druhá generace SŘBD

Relační databáze se označují jako *SŘBD druhé generace*.

- **Relační SŘBD**

První zmínka o relačních databázích je z r. 1970, kdy E. F. Codd z IBM Research Laboratory vydal pojednání v časopise *Communications of ACM*. Šlo nejprve pouze o teoretický základ. Pojednání bylo ale vhodně načasováno a reagovalo na nevýhody první generace SŘBD. Data viděl uživatel jako jednotlivé tabulky a k dispozici byly silné prostředky pro práci s těmito daty. Klíčovým krokem byl SQL – standardní dotazovací jazyk relačních databází.

I když ani relační databáze není bez nedostatků (zejména co se týká modelovací schopnosti), jedná se o nejpoužívanější typ databází – Oracle, MS SQL Server a IBM DB2 představují mnoha miliardové (v dolarech) odvětví. A s největší pravděpodobností to tak ještě dlouhou dobu zůstane.

### 2.5.4 Objektově orientovaný SŘBD – revoluční přístup

OOSŘBD je založen na objektově orientovaném přístupu. OOSŘBD se objevily nejprve ve strojírenství, získaly si ale oblibu i ve finančních a telekomunikačních aplikacích.

Pro objektově orientovaný datový model bylo navrženo několik různých definic. Tyto definice nejsou příliš výmluvné. Neexistuje totiž žádný objektově orientovaný model dat ekvivalentní podkladovému modelu dat relačních SŘBD.

### 2.5.5 Objektově relační SŘBD – evoluční přístup

Mnoho dodavatelů RSŘBD si uvědomilo nebezpečí a možnosti OOSŘBD. Shodli se, že tradiční RSŘBD nejsou vhodné pro pokročilé aplikace a je třeba přidat další funkčnost. Tři hlavní dodavatelé RSŘBD – Oracle, Microsoft a IBM – rozšířili své systémy do podoby ORSŘBD. Tyto hybridní systémy jsou velmi působivé, zachovávají zkušenosti z RSŘBD a přidávají novinky z OOSŘBD.

Tyto rozšířené ORSŘBD si vyžádaly novou verzi jazyka SQL. Vznikly tedy nové standardy SQL:2003 a SQL:2006.

ORSŘBD rozšiřuje relační model dat mimo jiné *opakovanou použitelností*. To znamená, že standardní funkce jsou poskytovány centrálně a nemusí je mít každá aplikace. Například aplikace může používat prostorová data, která slouží k reprezentaci bodů a čar, a funkce, které počítají vzdálenosti mezi body, úhly apod. Pokud bude tyto funkce poskytovat server, není potřeba je definovat v aplikaci.

Nevýhodou ORSŘBD je jeho složitost a s tím spojené zvýšené náklady. Zastánci čistě relačního přístupu jsou přesvědčení, že dojde ke ztrátě čistoty a jednoduchosti relačního modelu. [2]

## 2.6 Datové modely

Datový model je souhrn pravidel pro reprezentaci logické organizace dat v databázi.

### 2.6.1 Logické modely

Logické modely popisují data na úrovni konceptuální a na úrovni pohledů. Jsou to modely založené na záznamech a modely založené na objektech.

- **Hierarchický model**

Záznamu u hierarchického modelu se říká **segment**. Ze segmentu lze vytvářet odkazy pouze jedním směrem od předků k následníkům. Vzniká tak typická stromovitá struktura. Tato koncepce byla vhodná při použití magnetických pásek. S příchodem magnetických disků se zavedly nové pojmy: *fyzická databáze* (odpovídá původní koncepci) a *logická databáze*. Logická databáze vznikne spojením více fyzických databází s přidáním nových odkazů.

Nevýhodou tohoto modelu je složité vkládání a rušení záznamů.

- **Síťový model**

Síťový model je v podstatě zobecnění hierarchického modelu, který doplňuje o mnohonásobné vztahy, tzv. sety. Oba modely ale vznikly nezávisle na sobě. Sety spojují záznamy, a toto spojení může být realizováno na jeden nebo víc záznamů. Přístup na spojené záznamy je bez vyhledávání.

Nevýhodou tohoto modelu je nepružnost a obtížná změna její struktury.

- **Relační model**

Data v tomto modelu mají pravidelnou strukturu a jsou logicky strukturované do *relací* (tabulek). Sloupec v relaci se nazývá *atribut*. Řádek v relaci se nazývá *datová n-tice*.

Relační model řeší mimo jiné vazbu m:n, se kterou měly předchozí modely problémy. Tento model je v současnosti nejčastěji používán u komerčních SŘBD.

### 2.6.2 Modely fyzických dat

Popisují data na fyzické úrovni, postihují aspekty implementace.

## 2.7 Relační model

Protože je relační model v současnosti nejpoužívanější, podíváme se na něj podrobněji. Jak jsme si již řekli, data jsou v relačním modelu uspořádána do tabulek (relací). Tyto tabulky mají vlastnosti:

- Každá tabulka má jméno, které je v celé databázi jedinečné.
- Každá buňka tabulky obsahuje přesně jednu hodnotu.
- Každý sloupec (atribut) tabulky má jedinečné jméno (v tabulce).
- Všechny hodnoty ve sloupci jsou ze stejné domény (množina přípustných hodnot).
- Pořadí sloupců v tabulce nemá význam.
- Každý záznam je jedinečný, neexistují duplicitní záznamy. [1]

### 2.7.1 Klíče relačních tabulek

Protože každý záznam v tabulce je jedinečný, musí existovat sloupec nebo kombinace sloupců, které tuto jedinečnost zajišťují. Tyto sloupce se nazývají klíče.

- **Superklíč**

Superklíč je sloupec nebo množina sloupců, které jednoznačně identifikují záznam (řádek) v relaci (tabulce). Superklíč může obsahovat sloupce navíc, které pro jednoznačné určení záznamu nejsou potřebné.

- **Kandidátní klíč**

Kandidátní klíč je takový superklíč, který obsahuje minimální počet sloupců pro jednoznačnou identifikaci záznamu. Jako takový musí být **jedinečný**, tzn. v každém záznamu určuje hodnota klíče pouze jeden záznam. Dále musí být **neredukovatelný**, tj. žádná podmnožina klíče nezajistí jednoznačné určení záznamu.

- **Primární klíč**

Primární klíč je kandidátní klíč, který je vybraný, aby jednoznačně určoval záznamy v tabulce. Protože v tabulce neexistují duplicitní záznamy, vždy existuje primární klíč. V nejlepším případě je to jeden sloupec, v nejhorším případě to jsou všechny sloupce.

Kandidátní klíče, které nejsou vybrány za primární klíč, nazýváme *alternativní klíče*.

- **Cizí klíč**

Pokud se sloupec objeví ve více než v jedné tabulce, většinou to představuje vztah mezi oběma tabulkami. Cizí klíč je sloupec nebo skupina sloupců v jedné tabulce, která odpovídá kandidátnímu klíči některé tabulky.

I když se jedná o redundanci, kterou v databázi nevidíme rádi, v tomto případě se jedná o chtěnou redundanci, která nám reprezentuje vztahy mezi tabulkami. [1]

## 2.8 Normalizace

Normalizace je technika pro vytváření databáze s minimální redundancí. V roce 1972 techniku normalizace popsal „vynálezce“ relačního modelu pro podporu návrhu databází. Hlavním cílem návrhu databáze je seskupit sloupce do tabulek, aby se minimalizovala redundance. Ušetříme tak úložnou kapacitu a předejdeme problémům s *anomáliemi aktualizace*:

- Anomálie vkládání
- Anomálie vymazání
- Anomálie modifikace

### 2.8.1 První normální forma (1NF)

Tabulka, v níž každý průsečík sloupce a záznamu obsahuje jen jednu hodnotu.

První normální forma je jedinou normální formou, která je kriticky důležitá pro vytvoření vhodných tabulek relační databáze. Všechny ostatní normální formy jsou volitelné. Pro správný a bezproblémový chod (anomálie aktualizace) se doporučuje používat třetí normální formy (3NF).

Normalizace se provádí obvykle oddělením složených nebo redundantních položek do nové tabulky.

### 2.8.2 Druhá normální forma (2NF)

Tabulka, která je v 1NF a ve které jsou hodnoty každého sloupce, který není součástí primárního klíče, determinovány **všemi** hodnotami sloupců, které tvoří primární klíč.

Druhá normální forma se týká pouze tabulek, kde primární klíč tvoří dva a více sloupců. Tabulka, která je v 1NF a jejíž primární klíč tvoří jeden sloupec je automaticky v 2NF.

Normalizace se provádí obvykle dekompozicí (rozdělením tabulky) do více tabulek.

### 2.8.3 Třetí normální forma (3NF)

Tabulka, která již je v 1NF a 2NF a ve které všechny hodnoty ve sloupcích, které nepatří k primárnímu klíči, jsou determinovány **pouze** sloupci primárního klíče a nejsou determinovány žádnými jinými sloupci.

Neexistuje žádný neklíčový sloupec, který je tranzitivně závislý na některém klíči.

Existují normální formy, které jsou vyšší než 3NF – Boyce-Coddova normální forma (BCNF), čtvrtá normální forma (4NF) a pátá normální forma (5NF). [1]

## 3 SQL

SQL (Structured Query Language – strukturovaný dotazovací jazyk) je nejrozšířenějším komerčním databázovým jazykem. Původně ho vyvinuli ve výzkumné laboratoři IBM v San José.

Standard SQL ISO má dvě hlavní součásti:

- DDL (Data Definitiv Language, jazyk pro definici dat) – pro definici struktury databáze a kontrolu přístupu k datům
- DML (Data Manipulation Language, jazyk pro manipulaci s daty) – pro vyvolání a aktualizaci dat [6]

### 3.1 Manipulace s daty

Pro manipulaci s daty budeme používat příkazy SELECT, INSERT, UPDATE a DELETE.

#### 3.1.1 SELECT

Příkaz SELECT zobrazí data jedné nebo více tabulek. Jde o nejpoužívanější příkaz SQL.

Jeho syntaxe je:

```
SELECT [DISTINCT | ALL] [*] [sloupec [AS aliasSloupce]]  
[,...]  
FROM tabulka [aliasTabulky] [...]  
[WHERE podmínka]  
[GROUP BY seznamSloupců] [HAVING podmínka]  
[ORDER BY seznamSloupců];
```

Povinná klauzule je pouze FROM, která říká z které tabulky (kterých tabulek) se budou data zobrazovat.

- **Výběr všech řádků**

Příkazem `SELECT * FROM Firma;` vypíšeme všechny řádky a sloupce tabulky *Firma*. Vypisovat všechny sloupce není často nutné, můžeme tedy vypsát jen sloupce, které nás zajímají, když hvězdičku nahradíme seznamem sloupců: `SELECT Jmeno, Mesto, Ico`

FROM Firma;. Tento příkaz vypíše všechny řádky a sloupce se jmény *Jmeno*, *Mesto* a *Ico* z tabulky *Firma*.

Nevypíšeme-li žádný kandidátní klíč, je možné, že se nám vypíšou některé řádky duplicitně. Pokud o to nestojíme, použijeme klauzuli DISTINCT, která výpis omezí tak, že se neobjeví žádná duplicita. Např. SELECT DISTINCT Mesto FROM Firma; vypíše seznam všech měst, ve kterých je nějaká firma z naší tabulky. Každé město ale bude vypsáno pouze jedenkrát, i když je v něm více firem.

- **Výběr řádků**

Tak jako nás často (či spíše skoro vždy) nezajímají všechny sloupce tabulky, nás často nezajímají všechny řádky. K jejich omezení se používá klauzule WHERE následovaná podmínkou, při jejímž splnění bude řádek zobrazen.

Např. příkaz SELECT Jmeno, Ico, PocetZam FROM Firma WHERE PocetZam > 10; zobrazí uvedené sloupce a řádky s firmami, které mají více než 10 zaměstnanců. V podmínce můžeme použít všechny běžné logické operátory (= <> <= => > < AND OR NOT). U složitějších podmínek je doporučeno používat závorky, aby se předešlo dvojznačností.

Pro vyhledávání v textovém řetězci slouží klíčové slovo LIKE a speciální znaky %, ' a ,\_'. Procento představuje libovolný počet (i žádný) libovolných znaků a podtržítka představuje právě jeden libovolný znak. Pro výpis všech firem, jejichž jméno začíná písmenem „F“ a má libovolný počet znaků použijeme příkaz SELECT Jmeno, Ico, Mesto FROM Firma WHERE Jmeno LIKE 'F%';

Výsledky příkazu SELECT nejsou obecně nijak seřazeny. Klauzulí ORDER BY můžeme tyto řádky seřadit podle zadaných sloupců. Klauzule ORDER BY musí být poslední klauzulí příkazu SELECT. Například rozšíříme-li minulý příkaz na SELECT Jmeno, Ico, Mesto FROM Firma WHERE Jmeno LIKE 'F%' ORDER BY Mesto; seřadí výstup abecedně podle města. [6]

### 3.1.2 INSERT

Příkaz INSERT slouží k vkládání nových záznamů do tabulky. Jeho syntaxe je INSERT INTO tabulka [(sloupce)] VALUES (hodnoty); *tabulka* je jméno tabulky, do které nový záznam vkládáme. Nepovinný parametr *sloupce* je seznam sloupců, jejichž

hodnoty zadáváme do nového záznamu. Pokud není seznam sloupců uveden, předpokládá se kompletní seznam všech sloupců v implicitním pořadí. Pokud jsou některé sloupce neuvedeny, bude uložena jejich výchozí hodnota nebo hodnota *null*. Příklad příkazu INSERT:

```
INSERT INTO Firma VALUES ('Férová firma', '1234567', 'Praha', 20);
```

[6]

### 3.1.3 UPDATE

Příkaz UPDATE slouží pro aktualizaci záznamů v tabulce. Syntaxe je UPDATE tabulka SET jmenoSloupce = novaHodnota [, jmenoSloupce2 = novaHodnota2 ...] WHERE [omezovaciPodminka];

Do klauzule SET se zadává jméno sloupce (sloupců), které mají být aktualizovány. V nepovinné klauzuli WHERE můžeme omezit řádky, které budou aktualizovány (platí stejná pravidla jako v příkazu SELECT). [6]

### 3.1.4 DELETE

Příkaz DELETE slouží k vymazání záznamů (řádků) z tabulky. Syntaxe tohoto příkazu je DELETE FROM tabulka [WHERE podmínka]; tabulka je opět jméno tabulky, ve které se mají záznamy mazat. Pokud není zadána klauzule WHERE, budou vymazány všechny řádky tabulky (**nevratně – nedají se obnovit z koše!**). Při zadání podmínky budou vymazány pouze záznamy, které splňují podmínku.

Příkaz DELETE FROM Firma WHERE Jmeno = 'Férová firma'; vymaže všechny záznamy, ve kterých je *Jmeno* „Férová firma“. [6]

## 4 BEZPEČNOST

Webové aplikace jsou neustále vystaveny různým útokům. Internet je vlastně neustále prohledáván různými roboty, kteří zkouší různé formy útoku a hledají zranitelnosti aplikací. Mezi nejčastější útoky patří tzv. *injection*. Další častá forma útoku je *krádež identity*. Poslední skupinou útoků, o které se chci zmínit, jsou *cross-site* útoky.

### 4.1 Injection

Při této formě útoku posílá útočník místo dotazů, které systém očekává, různé zprávy, které při nevhodně napsaném ovládacím skriptu mohou vést k výpisu (nebo smazání) celé databáze, případně převzetí kontroly nad systémem.

#### 4.1.1 SQL Injection

K SQL Injection dochází nejčastěji přes formuláře webová aplikace. Útočník zadává do formuláře části SQL dotazů a snaží se z chybových hlášení zjistit skladbu databáze a vypsat její obsah.

Například do pole pro heslo zadá *ahoj or 'x' = 'x*. Pokud není aplikace ošetřena proti tomuto typu útoku, může být heslo vyhodnoceno jako správné a útočník bude přihlášen k účtu, který může obsahovat různé (i důvěrné) informace, které může útočník zneužít.

Další možností SQL Injection je změna URL adresy nebo podstrčením **cookies**.

Jedná se o nejčastější a relativně dobře známou techniku útoku, pořád existuje mnoho webových aplikací, které jsou tímto útokem zranitelné. Kodér musí ošetřit všechny vstupy aplikace. Toto ošetření se provádí obvykle dvakrát – poprvé na klientu, přímo ve webové stránce v prohlížeči (většinou pomocí JavaScriptu) a podruhé na serveru pomocí skriptu PHP nebo ASP.NET. První ošetření se provádí spíše pro komfort uživatele – uvidí, zda všechny vstupy, které zadal, jsou validní; útočník snadno tuto validaci obejde (vypnutím JavaScriptu v prohlížeči).

Na straně serveru má jazyk PHP implementovány funkce `is_string()` a `is_numeric()`, které ověřují, zda vstup obsahuje řetězec nebo číslo. Pokud očekáváme řetězec, je nutné využít tzv. *escapování* – pro databázi MySQL funkcí `mysql_real_escape_string($_GET[„id“])`. Od verze PHP 5 je implementována

třída PDO, která reprezentuje spojení mezi PHP a databázovým serverem. Tato třída se stará o oddělení příkazů SQL a zadaných proměných. Další možností je využití různých frameworků, které mají implementovány tuto ochranu, např. Nette Framework, Zend Framework a další...

Jazyk ASP.NET má pro tuto ověření tzv. **validátory**. Všechny validátory ASP.NET provádějí kontrolu jak na klientu (validátor vytváří kód JavaScriptu), tak na serveru. Nejuniverzálnějším validátorem je *RegularExpressionValidator*. Tomuto validátoru se zadává parametr *ValidationExpression*, který obsahuje regulární výraz, a tento se porovnává s výrazem, který zadal uživatel.

Kromě toho lze použít *CustomValidator*, kterému zadáme vlastní kontrolní algoritmus. Pokud chceme použít tuto kontrolu i na klientu, musíme napsat i JavaScriptovou funkci, kterou zadáme jako parametr *ClientValidationFunction*. Funkce, která ověřuje zadání na serveru, se zadává jako parametr *OnServerValidate*.

#### 4.1.2 LDAP Injection

LDAP je protokol pro dotazování a modifikaci adresářových služeb. Adresářové služby se často využívají pro ukládání uživatelských dat. Adresář je vlastně speciální databáze, která je určena pro rychlé čtení a vyhledávání, ale pouze na občasnou modifikaci a záznam (je pomalý).

LDAP Injection funguje téměř stejně jako SQL Injection, ale místo databáze a SQL dotazů útočí na adresáře pomocí LDAP dotazů.

Proto je stejná i obrana, a to kontrola všech vstupů od uživatelů.

#### 4.1.3 OS Injection

Jedná se o útok, kdy systému útočník posílá příkazy na úrovni OS. V případě, že napadený uživatel má v systému vyšší práva, využívá útočník těchto práv k ovládnutí systému.

Obrana je stejná jako u všech Injection útoků, validace vstupů uživatelů, v tomto případě navíc důsledné využívání nastavení uživatelských práv.

## 4.2 Krádež identity

Tímto útokem může útočník získat přístup k účtu uživatele, který je momentálně přihlášený. Pokud se jedná o administrátora, může útočník převzít kontrolu nad systémem. Jedná se o útok, při kterém útočník získá tzv. *Session ID*, což je náhodný řetězec, který identifikuje jednotlivé uživatele webové aplikace. Pomocí tohoto identifikátoru server rozpozná, od koho přišel požadavek, a podle toho vrací relevantní data.

Získáním SID (Session ID) útočník získá stejný přístup jako jeho oběť. Jsou 3 možné cesty jak získat SID.

### 4.2.1 Spočítat či uhádnout, brute-force attack

Tuto možnost můžeme jako kodéři minimalizovat generováním vhodného SID. Musí být dost dlouhé, náhodné, časově omezené...

### 4.2.2 Odchytit existující SID

SID si server a klient můžou předávat jako:

- a) parametr v URL
- b) skryté formulářové pole
- c) cookies

add a) Toto je rozhodně nejnebezpečnější způsob. Adresy URL se po cestě sítí logují na řadě serverů a routerů. Stejně tak se ukládá do historie prohlížeče a jeho záložek. Při prokliku na odkaz prohlížeč předává novému serveru v hlavičce původní URL včetně SID.

add b) Platí víceméně bod a), protože pokud je formulář posílaný serveru pomocí metody GET jsou formulářová data (včetně skrytých) uložena v adrese URL (platí bod a) doslova), při použití metody POST jsou formulářová data v hlavičce, kterou prohlížeč posílá serveru.

add c) Nejbezpečnější způsob předávání SID. Jednou z mála možností jak odchytit toto SID je pomocí XSS (viz kapitola 3.3). Lze se tomu bránit nastavením vlastnosti cookie HTTPOnly. Cookie s touto vlastností není dostupná JavaScriptem. Další možností je odchytit cookie sledováním datového provozu. Dnes je provozováno mnoho

nezabezpečených wi-fi sítí, kde nezabezpečená data doslova létají vzduchem. Jako vývojář webových aplikací se můžeme bránit pouze tak, že naše aplikace bude využívat pouze šifrovaný kanál protokolem https.

#### 4.2.3 Vygenerovat vlastní SID a podstrčit ho oběti

Útočník podstrčí oběti svůj odkaz, který již obsahuje permanentní cookie (SID). Když se oběť přihlásí do aplikace, získá útočník stejným odkazem stejný přístup do aplikace jako má jeho oběť.

Obranou je generování nového SID po přihlášení, případně při každém požadavku odeslaném na server. Další možností je dodatečné ověřování – sledujeme IP adresu, prohlížeč, rozlišení obrazovky,... tj. vlastnosti, které by se neměly mezi jednotlivými dotazy na server měnit.

### 4.3 Cross-Site Scripting (XSS)

Útočník posílá na vstup vlastní JavaScriptový kód, který stránka vykoná. Může tak získat data z databáze, převzít identitu jiného uživatele (krádež SID), případně až převzít kontrolu nad celým systémem.

Tento JavaScriptový kód může útočník vložit do stránek, jako příklad se často uvádí diskusní fórum nebo návštěvní kniha. Tento kód se uloží do úložiště dat, a vykonává se při každém zobrazení webové stránky. Kód nemusí být dlouhý, dobře bude fungovat i jednoduchý odkaz na skript kdekoliv v Internetu, např. `<script src="http://www.zaskodnickyyweb.net/xss.js"></script>`.

Obrana je opět relativně jednoduchá, a to validace výstupů do webových stránek. Jazyk PHP má funkci `htmlspecialchars()`, u ASP.NET se opět využívají různé validátory podobně jako u Injection. Musí se odfiltrovat potencionálně nebezpečné znaky, jako jsou `<` a `>`, díky kterým HTML předpokládá, že se jedná o *tag*. [16]

### 4.4 Cross-site Request Forgery (CSRF)

Útočník přiměje uživatele, aby spustil jím připravenou stránku, která na pozadí provede samotný útok, tj. provede akci jménem uživatele v aplikaci, ve které je uživatel přihlášen.

Například může změnit heslo, odeslat email, apod. Odkaz může být maskovaný jako neexistující obrázek, aby se hned provedl:

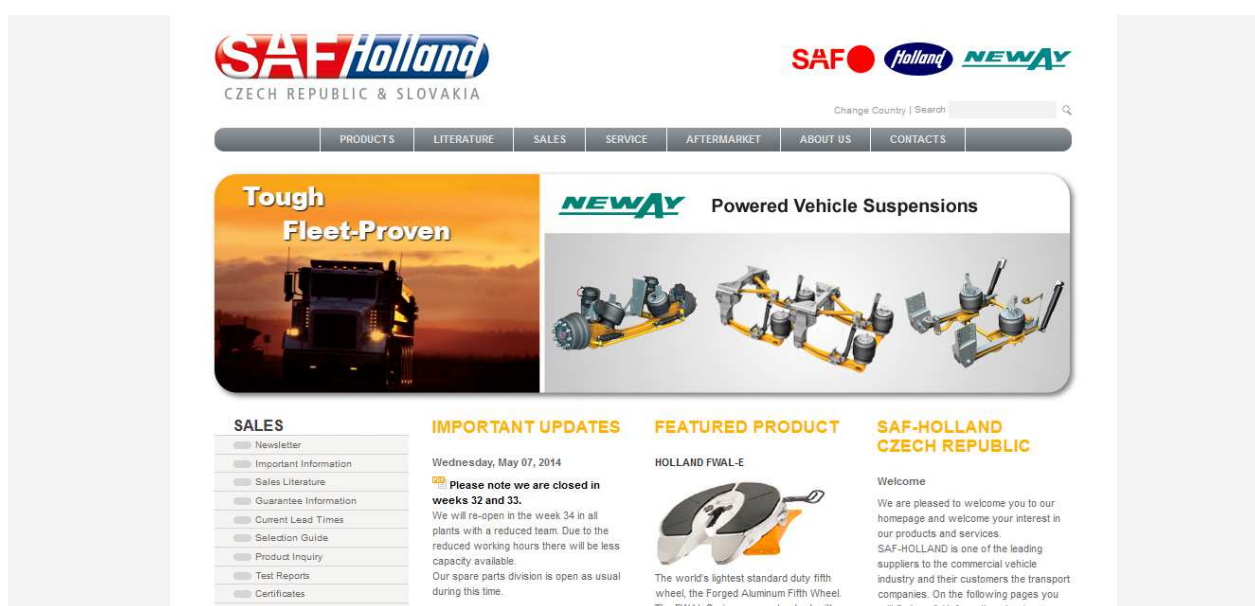
```
<img  
scr="http://www.vasmail.cz/sendmail?to=kamarad@jehomail.cz&subject  
=Spam&message=nejakanedulezitezprava>. Pokud je oběť přihlášená do mailové  
schránky (existuje platné SID cookie emailové schránky), tak pokud je správná syntaxe, se  
odešle email, aniž by o tom oběť věděla.
```

Jako nejlehčí způsob obrany se zdá vyžadovat data metodou POST, ale toto je možné snadno podvrhnout pomocí JavaScriptu. Opakem tedy opravdu funkční obranou je požadovat potvrzení operace pomocí SMS nebo alespoň CAPTCHA. Další možností obrany je nastavení co nejkratší doby platnosti SID. Je to ale na úkor komfortu uživatele. Po vypršení platnosti SID bude aplikace vyžadovat přihlášení (i při skrytém požadavku). Ještě je třeba zmínit metodu skrytých polí, kdy v každém formuláři je jedno skryté pole, které obsahuje jednoznačný identifikátor. Tento identifikátor se samozřejmě odesílá s každým odesláním formuláře a server může ověřit, zda formulář poslal ten, kdo měl. [17]

## **II. PRAKTICKÁ ČÁST**

## 5 WEBOVÉ STRÁNKY

Oficiální webové stránky firmy SAF-HOLLAND Czechia [www.safholland.cz](http://www.safholland.cz) spravuje její mateřská společnost SAF-HOLLAND GmbH se sídlem v Německu. Firma SAF-HOLLAND Czechia nemá tedy možnost obsah stránek jakkoliv ovlivnit – stránky dokonce nejsou v českém jazyce. Z tohoto důvodu bylo rozhodnuto o vytvoření „neoficiálních“ stránek firmy [www.saf-holland.cz](http://www.saf-holland.cz). Design webových stránek by měl na první pohled vypadat podobně jako oficiální stránky, samozřejmě o obsahu bude rozhodovat společnost SAF-HOLLAND Czechia.



Obr. 5 Oficiální webové stránky

### 5.1 Technologie

Vyvíjená aplikace bude spolupracovat s ERP systémem K2, který běží na serverech společnosti SAF-HOLLAND Czechia, jejichž operační systém je Windows Server 2008R2 a systém K2 využívá SŘBD Microsoft SQL Server. Bylo proto rozhodnuto, aby byla možná co nejsnadnější spolupráce mezi oběma aplikacemi, že webové stránky poběží přímo na serverech společnosti, a to na webovém serveru IIS 7.5, který součástí OS Windows Server 2008R2. Protože firma vlastní licence pro SŘBD Microsoft SQL Server bude použita data-báze taktéž uložena na serveru společnosti. Server je do internetu připojen linkou s dostatečnou kapacitou pro předpokládaný provoz internetových stránek. Tím máme pohromadě všechny potřebné prostředky pro provozování webových stránek. Máme internetovou adre-

su, máme připojení do internetu, máme webový server a (protože budeme využívat databáze) máme SŘBD.

## 5.2 Vlastní tvorba stránek

Stránky budou obsahovat důležité informace, které chce firma SAF-HOLLAND Czechia sdělovat svým zákazníkům. Tyto informace, které se týkají sortimentu koncernu SAF-HOLLAND, vydává mateřský podnik v několika světových jazycích (mezi nimi bohužel chybí čeština), se v české pobočce přeloží a přidají na „neoficiální“ stránky. Informace, které se netýkají sortimentu koncernu, se získávají od výrobců a dodavatelů jednotlivých komponentů. Je tedy předpoklad, že budou relativně často přibývat další a další. Je tedy vhodné, pro snadné přidávání, využití databáze, která bude obsahovat jednotlivé články (informace).

Proto je vhodné vytvořit dynamický web. Tento poběží na webovém serveru IIS firmy Microsoft, je proto maximálně vhodné využití serverové skriptovacího jazyka ASP.NET, který je pro tento webový server nativní.

### 5.2.1 Potřebný software

Jak bylo řečeno v úvodu, pro napsání webových stránek stačí poznámkový blok. Není to ale nejvhodnější volba, zejména u složitějších projektů.

- **IDE**

Microsoft nám usnadnil rozhodování, protože uvolnil svůj vývojový nástroj Visual Studio ve verzi Express jako freeware. Omezení, která tato verze obsahuje, jsou tak speciální, že jsem se dosud nesešel s něčím, co by ve verzi Express nefungovalo. Součástí Visual Studio Express for Web je i webový server IIS Express a MS SQL Server Express. Obsahuje tedy všechno pro vytvoření funkční webové aplikace.

- **Grafický editor**

Aby byl web přitažlivý, musí mít jistou grafickou úroveň a bude tedy obsahovat i nějaké obrázky. Protože ne vždy máme k dispozici všechny potřebné grafické objekty v požadované kvalitě, potřebujeme vhodný editor obrázků.

Já používám program GIMP, který dostupný zdarma včetně zdrojových kódů pod licenci GPL.

GIMP je editor rastrové grafiky, který obsahuje mnoho rastrových nástrojů. Umí pracovat s vrstvami. Existuje pro něj mnoho zásuvných modulů, které vytváří bohatá uživatelská komunita. Umožňuje export do mnoha grafických formátů. Navíc je kompletně přeložený do češtiny.

- **Firebug**

Firebug je doplněk prohlížeče Firefox, který používám pro ladění kódu. Dokáže vypsat informace o jednotlivých objektech na stránce – HTML, CSS. Umožňuje jejich editaci. Analyzuje běh skriptů na stránkách, obsahuje rozšířenou konzolu JavaScriptu. Hlídá dotazy, které stránka vyvolala. Umí procházet i DOM.

### 5.2.2 Nový projekt

Po spuštění Visual Studia vybereme *New Project* z menu *File* pro vytvoření nového projektu. Jako jazyk skriptu vybereme **C#** a framework **ASP.NET MVC4 Web Application** a projekt nějak vhodně pojmenujeme. Potvrdíme OK a v dalším okně zvolíme šablonu **Internet Application**, jako zobrazovací engine Razor (zobrazovací engine ASPX se téměř nevyužívá, má mírně odlišnou syntax a má nižší podporu frameworku). Framework MVC je vhodný pro testy řízený vývoj aplikací, k vytvoření testů slouží volba **Create a unit test project**, v tomto případě je nebudeme využívat. Potvrdíme OK.

Pro vytvoření webových stránek a webové aplikace jsem si vybral framework MVC. Framework MVC představuje třívrstvou architekturu **Model – View – Controller**. Tato architektura dělí projekt na logické celky:

- Model – model – obsahuje data a business logiku
- View – pohled – je kód, který má na starosti generování zobrazovacího kódu, zejména \*.html
- Controller – kontroler – zachytává požadavky prohlížeče (většinou speciální kontroler, který požadavky předává na další kontroler), předává tyto požadavky na požadovaný model a spojí ho se správným pohledem, který vrátí data pro zobrazení prohlížeči.

Visual studio vytvoří nový projekt podle šablony. V okně **Solution Explorer** vidíme skladbu projektu. Všimněme si složek *Controllers*, *Models* a *Views*. Samozřejmě obsahují jednotlivé díly architektury MVC.

Ve složce **App\_Start** je soubor *RouteConfig.cs*. V tomto souboru se nastavuje směrování požadavků prohlížeče. Najdeme tady třídu *RouteConfig* se statickou metodou *RegisterRoutes*. Funkce *MapRoute* v této metodě nastavuje směrování tzv. *FirstControlleru*, tj. říká, jak se mají směřovat požadavky prohlížeče. V parametru *name* je jméno směrování, v parametru *url* je skladba adresy z prohlížeče, a v *defaults* výchozí hodnoty, pokud nejsou v prohlížeči zadány.

Z výchozích hodnot

```
routes.MapRoute(  
    name: "Default",  
    url: "{controller}/{action}/{id}",  
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }  
);
```

vyčteme, že pokud nezadáme za webovou adresou nic, načte se kontroler **Home** a provede akci **Index** bez parametrů. Pokud nezadáme akci, provede se akce *Index* zadaného kontroleru. Většinou je toto výchozí nastavení vhodné, a platí to i pro tento náš případ. Je ale možné nastavit více směrování. [13]

Podíváme se tedy do složky *Controllers*, která obsahuje kontrolery pro náš web. V šabloně je **AccountController** a **HomeController**. *AccountController* obsahuje kód pro přihlašování a odhlašování a změnu hesla. Teď nás zajímá *HomeController*. Pokud ho nemáme otevřený, dvojklikem ho otevřeme.

Vidíme, že se jedná o třídu odvozenou od třídy *Controller*, která obsahuje veřejné metody, které vrací hodnotu typu *ActionResult*. Podíváme-li se na první metodu *Index()*, vidíme, že nečeká žádný vstup, vloží do proměnné *ViewBag.Message* textový řetězec a vrací funkci *View()*, které zavolá pohled se stejným jménem, jako je jméno metody. **ViewBag** je speciální objekt, který se předává z kontroleru do pohledu.

Podíváme se tedy na pohled. Klikneme pravým tlačítkem do vybrané metody a z kontextového menu zvolíme *Go To View*. Tím se nám otevře správný pohled.

V pohledu vidíme, známý HTML kód, který obsahuje části skriptu. Skript se uvozuje znakem „@“. HTML kód je jednoduchý a srozumitelný.

V kódu ale chybí hlavička `<head> </head>`. A tag `<body> </body>`. Jak je možné, že prohlížeč stránku zobrazí? V šabloně je použitý tzv. layout. Otevřeme složku Views a dále složku Shared. Tady najdeme soubor `_Layout.cshtml`. Tento soubor určuje vzhled webových stránek. Pokud při vytváření pohledu použití layoutu nezakážeme, vždy se provede tento skript a pohled se vloží na místo příkazu `@RenderBody()`.

Abychom dodrželi koncepci MVC, měl by veškerý kód a výpočty provádět kontroler a výsledky předávat do pohledu. Pohled by měl být jenom „hloupý“ zobrazovač.

### 5.2.3 Náš projekt

- `_Layout.cshtml`

Vytvoříme tedy nový projekt MVC v jazyce C# a pojmenujeme si ho např. saf-holland.

Nejprve si vytvoříme layout stránek, otevřeme si tedy soubor `_Layout.cshtml` ve složce Shared ve složce Views. A upravíme ho, aby nám vyhovoval.

Budeme potřebovat nějaké obrázky, vložíme je do složky Images přes SolutionExplorer. V hlavičce změníme titulek webové stránky. Dále si vytvoříme hlavičku stránky, která bude obsahovat loga firmy, možnost přihlášení a registrace, hlášku o datu a svátku a statické menu.

Obrázek zobrazíme klasickým tagem `<img />`. Do parametru `src` vložíme funkci `@Url.Content("~/Images/obrazek.jpg")`. Menu vytvoříme pomocí nečíslovaného seznamu, do jehož položek nasázíme položky menu.

Nastavení vzhledu, velikosti, pozic apod. nastavíme pomocí kaskádových stylů CSS, soubor `site.css` je ve složce Content. Pomocí CSS nastavíme i vzhled a chování menu.

Aktuální datum získáme vlastností `Today` datového typu `DateTime`. Pomocí JavaScriptu vypíšeme, kdo má daný den svátek.

Poté spustíme vykreslení požadovaného pohledu příkazem `@RenderBody`.

Pod pohled vykreslíme patičku, kde budeme vypisovat copyright.

- **Databáze**

Jak jsme si řekli, články, které budeme zobrazovat, budeme ukládat do databáze. Kde ale vezmeme databázi?

Pro práci s databází použijeme **Entity Framework**, což je tzv. ORM (objektově relační mapování). Všechny databázové tabulky se přímo namapují na třídy C#, v kódu pak pracujeme jenom s objekty a SQL dotazy do databáze generuje framework na pozadí.

Existují dvě možnosti jak pracovat s Entity Frameworkem. Jedna se jmenuje *Code First*. Vytvoříme nejprve třídy C# a framework z nich vytvoří databázové tabulky, druhou možností je vytvořit nejprve databázi a framework z ní vytvoří třídy v C#. Tato metoda se nazývá *Database First*.

Já jsem zvolil první možnost. Napíšeme tedy třídy **Menu.cs**, **Podmenu.cs**, **Oddil.cs**, **Pododdil.cs** a **Clanek.cs**. Klikneme pravým tlačítkem na složku Models (sem umístíme naše třídy), vybereme Add... a v dalším okně zvolíme Class.

Menu.cs :

```
public class Menu
{
    public Menu()
    {
        PodmenuList = new List<Podmenu>(); //vytvoří relaci 1:n
    }

    public int Id { get; set; } // jedinečný identifikátor
    public string Nadpis { get; set; } // zobrazovaný text
    public string MenuControl { get; set; } //jméno kontroleru
    public bool Platne { get; set; } // je menu platné
    public virtual ICollection<Podmenu> PodmenuList { get; set; } //virtuální parametr, který odkazuje na třídu Podmenu
}
```

Podmenu.cs

```
public class Podmenu
{
    public Podmenu()
    {
        OddilList = new List<Oddil>(); // vytvoří relaci 1:n
        this.Clanky = new HashSet<Clanek>(); // vytvoří relaci n:m
    }
    public int Id { get; set; }
    public string Nadpis { get; set; }
    public string PodmenuController { get; set; } //jméno kontroleru
    public string PodmenuAction { get; set; } // jméno metody kontroleru
    public int MenuId { get; set; } //cizí klíč třídy Menu.cs
    public bool Platne { get; set; }
    public virtual Menu Menu { get; set; } //virtuální parametr pro vytvoření relace 1:n s třídou menu
    public virtual ICollection<Oddil> OddilList { get; set; }
    public virtual ICollection<Clanek> Clanky { get; set; }
}
```

Oddil.cs:

```
public class Oddil
{
    public Oddil()
    {
        PododdilList = new List<Pododdil>();
        this.Clanky = new HashSet<Clanek>();
    }
    public int Id { get; set; }
    public string Nadpis { get; set; }
    public string OddilController { get; set; }
    public string OddilAction { get; set; }
    public int PodmenuId { get; set; }
    public bool Platne { get; set; }

    public virtual Podmenu Podmenu { get; set; }
    public virtual ICollection<Pododdil> PododdilList { get; set; }
    public virtual ICollection<Clanek> Clanky { get; set; }
}
```

Pododdil.cs

```
public class Pododdil
{
    public Pododdil()
    {
        this.Clanky = new HashSet<Clanek>();
    }
    public int Id { get; set; }
    public string Nadpis { get; set; }
    public string PododdilController { get; set; }
    public string PododdilAction { get; set; }
    public int OddilId { get; set; }
    public bool Platne { get; set; }
    public virtual Oddil Oddil { get; set; }
    public virtual ICollection<Clanek> Clanky { get; set; }
}
```

Clanek.cs

```
public class Clanek
{
    public int Id { get; set; }
    public string Nadpis { get; set; } //zobrazovaný nadpis článku
    public DateTime DatumPridani { get; set; } //Datum poslední změny
    public string Abstrakt { get; set; } //zobrazovaný abstrakt článku
    public string Obrazek { get; set; } //relativní adresa obrázku článku
    public string Text { get; set; } //relativní adresa textu článku
    public string Odkaz { get; set; } //relativní odkaz na článek
    public string Pdf { get; set; } //relativní odkaz na pdf verzi článku
    public bool TOP { get; set; }
    public bool Platne { get; set; }
    public virtual ICollection<Oddil> Oddily { get; set; }
    public virtual ICollection<Pododdil> Pododily { get; set; }
    public virtual ICollection<Podmenu> Podmenus { get; set; }
}
```

Aby VS mohlo s vytvořenými třídami pracovat, musíme projekt sestavit. Klikneme pravým tlačítkem na jméno projektu v Solution Exploreru a zvolíme Rebuild. Projekt se tak zkompile s nově přidanými třídami. Projekt musíme zkompileovat po každé změně v třídě nebo kontroleru, aby se tyto změny promítly do spuštěné aplikace.

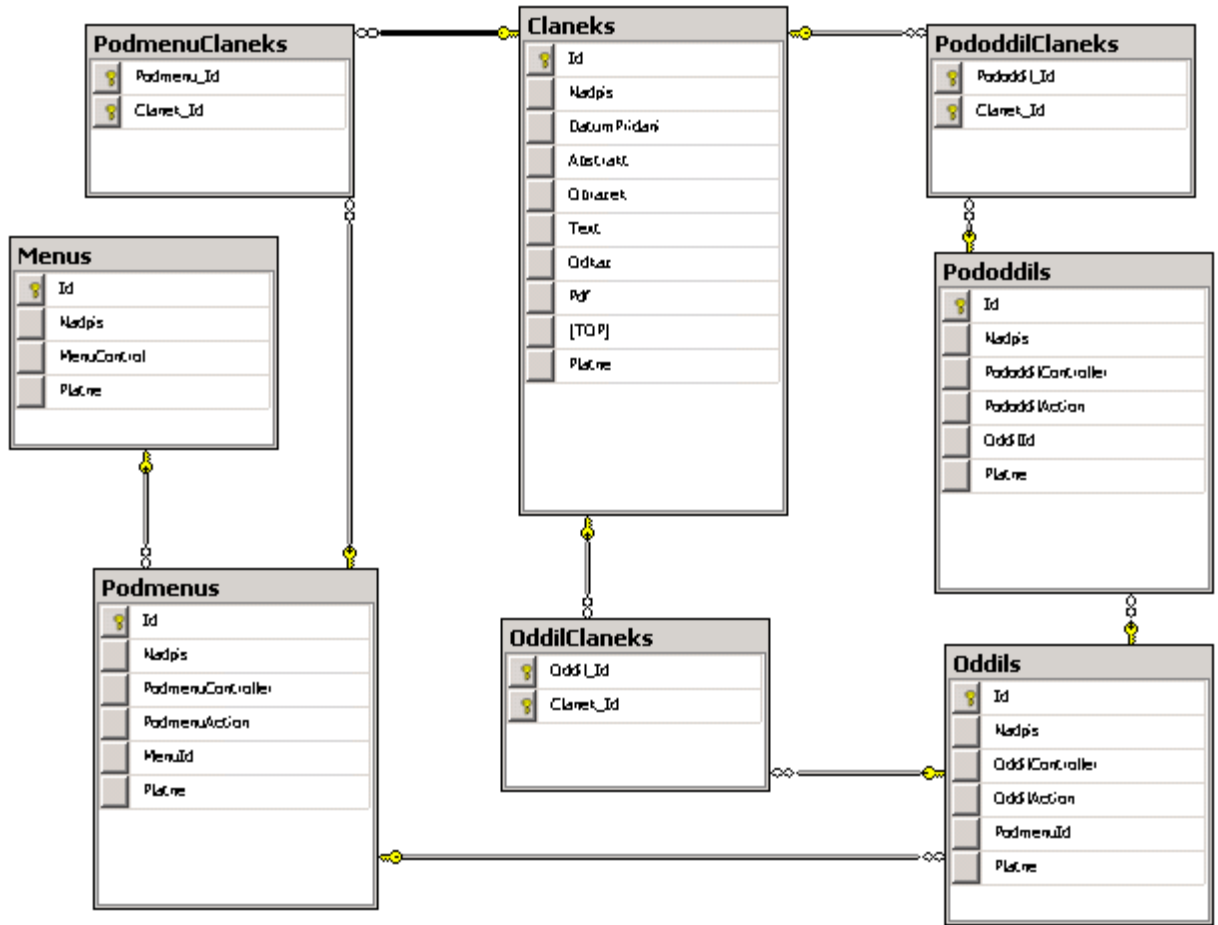
Přidáme si kontroler, který bude s třídami pracovat. Klikneme pravým tlačítkem na složku Controllers a zvolíme Add Controller. V nově otevřeném okně vybereme MVC5 Controller with views, using Entity Framework.

Jako jméno zvolíme **ClanekController**, jako třídu zvolíme naši třídu Clanek a jako datový kontext jediný existující. Necháme zatrhnutou možnost Generate views (VS vygeneruje standardní pohledy pro kontroler) a Use a layout page (vygenerované pohledy budou obsahovat náš připravený layout). Dialog potvrdíme.

Vygeneruje se ClanekController, který již obsahuje několik metod, a to: *Index*, *Details*, *Delete*, *Create* a *Edit*. Jejich názvy přesně vystihují, co provádějí. Když se podíváme do složky Views, najdeme tu složku Clanek, která obsahuje pohledy pro tyto metody.

Nyní aplikaci spustíme (zelená šipka v horní liště, případně F5). Aplikace přeloží a spustí se IIS (jeho ikona se zobrazí v liště). V adresním řádku bude adresa localhost:53076 (port může být jiný, generuje ho IIS pro každou relaci). Jak víme, zavolá se HomeController a jeho metoda Index(). Pro zavolání našeho nového kontroleru přepíšeme adresu na localhost:53076/Clanek. Tato adresa zavolá kontroler ClanekController a jeho metoda Index(). Protože je databáze prázdná zobrazí se pouze hlavička tabulky.

Aplikaci zastavíme a přepneme se do Server Exploreru. Přes Data Connections najdeme naše připojení k databázi. Mezi tabulkami najdeme naše tabulky. Navíc jsou tam i pomocné tabulky pro relace n:m mezi tabulkami. Diagram vytvořených tabulek vypadá asi takto:



Obr. 6 Schéma tabulek databáze pro funkční menu

Taková databáze se vygenerovala pouze zadáním několika tříd. A kontroler má funkční kód. Zkusme opět spustit aplikaci a do adresy zadat localhost:53076/Clanek/Create. Otevře se formulář, který je plně funkční:



Obr. 7 První funkční formulář naší aplikace

Ve formuláři jsou funkční i validátory, které nám nedovolí formulář odeslat, pokud zadáme neplatnou hodnotu.

Vzhledově nevypadá formulář moc vzhledně, tak ho upravíme. V `ClanekController` klikneme pravým tlačítkem do metody `Create` a vybereme `Go To View`. Tento pohled upravíme. Datum přidání nebude vyplňovat, ale doplníme ho automaticky při uložení – v pohledu smažeme celý kód, který zobrazuje toto políčko (i validátor):

```
<div class="form-group">
  @Html.LabelFor(model => model.DatumPridani, new { @class = "control-label col-
md-2" })
  <div class="col-md-10">
    @Html.EditorFor(model => model.DatumPridani)
    @Html.ValidationMessageFor(model => model.DatumPridani)
  </div>
</div>
```

V tomto případě musíme ovšem upravit i kontroler. Otevřeme `ClanekController` a najdeme metodu `Create` (`Clanek clanek`). Tato metoda očekává, že se jí vrátí prohlížeče kompletní třída `Clanek`. Vymažeme tedy z parametru `Include DatumPridani`, ale před kontrolou, zda je model platný musíme objekt **clanek** doplnit. To uděláme řádkem `clanek.DatumPridani = DateTime.Today;`

Pro abstrakt budeme potřebovat více prostoru: přepíšeme `@Html.EditorFor(model => model.Abstrakt)` na `@Html.TextAreaFor(model => model.Abstrakt)`.

Pomocí CSS vytvoříme tlačítko a posuneme formulář dál od okraje.

The screenshot shows a web page with a navigation bar at the top containing 'CZECH REPUBLIC & SLOVAKIA' and a date 'Dnes je 14.05.2014 a svátek má Bonifác'. Below the navigation bar are several menu items: 'PRODUKTY SAF-HOLLAND', 'NÁHRADNÍ DÍLY SAF-HOLLAND', 'OSTATNÍ PRODUKTY', 'SERVIS', 'ZÁKAZNICKÁ ZÓNA', 'O NÁS', and 'KONTAKT'. The main content area is titled 'Vložit nový článek' and contains the following form elements:

- Nadpis**: A text input field.
- Abstrakt**: A large text area for the abstract.
- Obrazek**: A text input field.
- Text**: A text input field.
- Odkaz**: A text input field.
- Pdf**: A text input field.
- TOP**: A checkbox.
- Platne**: A checkbox.
- Vložit**: A button to submit the form.

At the bottom right of the page, there is a link labeled 'Seznam článků'.

Obr. 8 Formulář pro vkládání článků

Vidíme, že na pár kliknutí a téměř bez psaní kódu máme funkční formulář, pomocí kterého můžeme vkládat data do databáze.

Formulář má nyní jednu vadu. Řekli jsme, že články budou přiřazeny k podmenu a oddílům. To tento formulář ještě neumí.

Nejprve si musíme naplnit tabulky Menu, Podmenu, Oddil a Pododdil daty. Protože se jedná o málokdy měněné pole, nebudeme vytvářet kontroler, ale vyplníme tabulky ručně.

Když je budeme mít naplněné, můžeme vytvořit dynamické menu po levé straně. Menu vytvoříme pomocí několika vnořených seznamů.

V kontroleru u každé metody přidáme tento kód:

```
var podmenu = db.Podmenus.ToList();
ViewBag.podmenu = podmenu;
var menu = db.Menus.ToList();
ViewBag.menu = menu;
```

Do pohledu přidáme tento kód, který nám vytvoří boční menu:

```
<div id="side_menu">
  <ul>
    @foreach (var item in ViewBag.menu){
      <li class="nadpis">
        <a href="/@item.MenuControl">@item.Nadpis.ToString()</a>
      </li>
      foreach (var poditem in ViewBag.podmenu){
        <li class="odkaz">
          <ul>
            @if (item.Id == poditem.MenuId){
              <li>
                <a
                  href="/@poditem.PodmenuController/@poditem.PodmenuAction>
                  @poditem.Nadpis.ToString()</a>
              </li>
            }
          </ul>
        </li>
      }
    }
  </ul>
</div>
```

Nyní přidáme možnost doplnit sekce, do kterých články patří. Do kontroleru Claneek, do metody Create() přidáme několik řádků, které do ViewBagu přidají seznamy sekcí:

```
ViewBag.oddilclaneek = new SelectList(db.Oddils, "Id", "Nadpis");
ViewBag.podmenuclaneek = new SelectList(db.Podmenus, "Id", "Nadpis");
ViewBag.pododdilclaneek = new SelectList(db.Pododdils, "Id", "Nadpis");
```

Tyto seznamy zobrazíme v pohledu jako rozbalovací pole:

```
<div class="form-group">
  @Html.LabelForModel("Podmenu", new { @class = "control-label col-md-2" })
  <div class="col-md-10">
    @Html.DropDownList("podmenuclaneek", "žádné")
  </div>
</div>
```

```

        </div>
    </div>
    <div class="form-group">
        @Html.LabelForModel("Oddíl", new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownList("oddilclanek", "žádný")
        </div>
    </div>
    <div class="form-group">
        @Html.LabelForModel("Pododdíl", new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownList("pododdilclanek", "žádný")
        </div>
    </div>
</div>

```

Zpracování těchto údajů musíme opět přidat do kontroleru *Create(Vozidlo vozidlo)*. A to tak, že ho rozšíříme. Nyní bude očekávat ještě další proměnné – *Create(Vozidlo vozidlo, int? podmenuclanek, int? oddilclanek, int? pododdilclanek)*. Dále musíme do kontroleru přidat kód, který tyto data zpracuje, pokud je nějaká hodnota zadána, přidá záznam do databáze.

```

if (podmenuclanek != null) db.Podmenus.Find(podmenuclanek).Clanky.Add(clanek);
if (oddilclanek != null) db.Oddils.Find(oddilclanek).Clanky.Add(clanek);
if (pododdilclanek != null) db.Pododdils.Find(pododdilclanek).Clanky.Add(clanek);

```

Tím jsme dostali funkční formulář pro vkládání článků. Ještě nám můžeme upravit české nadpisy na české. Toto změníme přímo ve třídě *Clanek.cs*. Přidáme nad řádek člena, kterého chceme přejmenovat [*Display(Name="zobrazovaný text")*].

```

[Display(Name="Obrázek")]
public string Obrazek { get; set; }

```

Obr. 9 Formulář pro vkládání článků do platných sekcí

Nyní máme opravdu český funkční formulář. Samozřejmě ale nechceme, aby k tomuto formuláři měli přístup všichni návštěvníci webu. Proto do kontroleru nad metodu, která tento formulář zobrazí, napíšeme omezení [`Authorize(Roles="admin")`]. To zajistí, že se na tuto stránku dostane pouze přihlášený uživatel, který má přidělenou roli „admin“. Na přihlašování a role se podíváme, až budeme vytvářet aplikaci a zákaznickou zónu.

Dále upravíme pohled pro úpravu článku, na kterou se dostaneme zadáním adresy **localhost:53076/Clanek/Edit/id**, kde id je číslo článku v databázi. Vše funguje podle pravidel routování: kontroler = Clanek, akce = Edit a parametr = id. Pohled upravíme, aby vypadal stejně jako pohled pro vytvoření nového článku. Rozdíl bude pouze v tom, že framework načte data článku z databáze a předvyplní je do formuláře. Po potvrzení, kontroler ověří data a změny zapíše do databáze. Protože opět nechceme, aby tuto metodu volal každý návštěvník stránek, opět před metodu napíšeme direktivu omezení.

Dalším pohledem, který upravíme je **Delete**, který zajišťuje mazání článku z databáze. Protože je možné, že příkaz k smazání byl zvolen neúmyslně, bývá dobrým zvykem zobrazit dotaz na smazání dvakrát. I na toto framework myslel a pokud do prohlížeče zadáme adresu `localhost:53076/Clanek/Delete/id` (id má stejný význam jako v metodě Edit), zobrazí se stránka s dotazem, zda si jsme opravdu jisti smazáním a článek se smaže až po potvrzení.



Obr. 10 Automaticky vygenerovaný formulář pro smazání článku

Tato stránka se nám samozřejmě nelíbí, takže opět upravíme pohled. Klikneme pravým tlačítkem do metody Delete a zvolíme Go To View. A upravíme pohled do námi požadovaného vzhladu.



Obr. 11 Formulář pro vymazání článku

My ale články mazat nechceme. Jenom je označíme jako neplatné. Proto musíme upravit metodu `DeleteConfirmed` v kontroleru. Změníme řádek, který vymazává záznam z databáze `db.Claneks.Remove(clanek);`, tak aby označil záznam jako neplatný: `clanek.Platne = false;` Články, které nejsou platné, se budou vypisovat pouze administrátorovi, takže je uživatel stránek nevidí.

Další metodou vytvořeného kontroleru je **Index**. Je to výchozí metoda, která se zavolá, pokud není zadána žádná akce. Nyní chceme, aby vypsala všechny články a odkazy k jejich správě. Pokud nyní zavoláme tuto metodu zadáním adresy `localhost:53076/Clanek`, získáme výchozí zobrazení, které opět poupravíme, aby nám zapadlo do šablony.

Vytvořit nový

Nadpis	Datum změny	Abstrakt	TOP	Platný článek	
Info č.1 - Připevnění příruby kola - náboj (Hub-Unit)	15.5.2014 10:51:08	Změna šroubového spoje	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<a href="#">Upravit</a>   <a href="#">Podrobnosti</a>   <a href="#">Smazat</a>
Info č.2 - Těleso brzdy KNORR SB6.../SB7... s ocelovým krytem	4.7.2000 0:00:00	Přechod z pryžového krytu na ocelový	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Upravit</a>   <a href="#">Podrobnosti</a>   <a href="#">Smazat</a>
test intradisc	23.6.2000 0:00:00	asfeiafa	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Upravit</a>   <a href="#">Podrobnosti</a>   <a href="#">Smazat</a>
test intradrum	23.5.2001 0:00:00	eafaeuhasfhau	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<a href="#">Upravit</a>   <a href="#">Podrobnosti</a>   <a href="#">Smazat</a>
Test	14.5.2014 0:00:00	test do královských čepů	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<a href="#">Upravit</a>   <a href="#">Podrobnosti</a>   <a href="#">Smazat</a>

Obr. 12 Výpis seznamu článků

Poslední metodou je **Details**, který zobrazí podrobnosti článku, v našem případě samotný článek. Protože text článku je v externím HTML souboru, vypadá tento detail otřesně. Otevřeme si tedy pohled Details a upravíme ho. Vymažeme hlavičky, a místo adresy textu článku tento text zobrazíme, použijeme k tomu příkaz `@Html.RenderPartial(jménoSouboru)`, který připojí daný soubor. Stránka pro `id=1` bude vypadat asi takto:

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Královské čepy
- Kuličkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité Informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky
- Gumové zástěrky
- Systém AntiSpray
- Tažné zařízení
- Odstavné nohy a navijáky
- Příčky a oje
- Ventily a EBS
- Ostatní

**SERVIS**

- Reklamacé
- Školení

### Info č.1 - Připevnění příruby kola - náboj (Hub-Unit)

**Změna šroubového spoje**

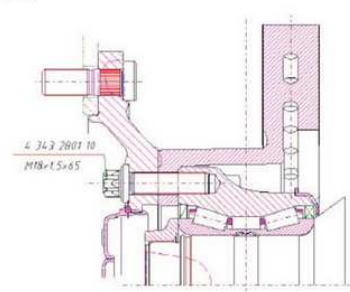
Doposud 12 x šestihranných šroubů s nákrůžkem M 18 x 1,5, ident. č. SAF 4 343 2916 10  
Způsob dotažení: moment dotažení 450 Nm

Nově 12 x šroubů TORX s nákrůžkem M 18 x 1,5, ident. č. SAF 4 343 2801 10  
Způsob dotažení: předběžné dotažení 50Nm přes kříž, úhel otáčení 90° dotáhněte přes kříž (1 1/2 rožku hlavy), šrouby a styčné plochy bez tuku a oleje!  
Alternativa: moment dotažení 450 Nm. Šrouby a styčné plochy bez tuku a oleje!

Pozor: Při tomto způsobu musíte šrouby při každé další montáži vyměnit.

Nářadí: Vnitřní TORX E 24  
Pohon 3/4"

Ident. č. SAF 4 343 2801 10



4 343 2801 10  
M18x1.5x65

[Stáhnout pdf](#)

15.05.2014

Obr. 13 Detail článku

Když si nyní prohlédneme kontroler, všimeme si, že pouze jediná metoda je přístupná někomu jinému než administrátorovi. Můžeme tedy nastavit, že celý kontroler bude přístupný pouze administrátorovi, a pouze jedna metoda bude přístupná komukoliv. Nad kontroler napíšeme již známe omezení: `[Authorize(Roles="admin")]`, u všech metod ho můžeme smazat, a k metodě `Details` napíšeme `[AllowAnonymous]`. Tím máme vytvořený základ naší webové prezentace.

Samozřejmě nemůže po uživateli chtít, aby zadávali čísla článků, když chtějí nějaký zobrazit. Proto máme systém menu, který vypíše seznam článků patřících do jednotlivých sekcí. V těchto článcích již bude pro uživatele snazší se orientovat a otevřít si požadovaný článek.

Vytvoříme si pro každou položku menu kontroler. Máme tedy **ProduktyController**, **NdSafController**, **OstatniController**, **ServisController**, **OController** a **KontaktController** (ten zatím vytvářet nebudeme). Pro zákaznickou zónu využijeme již vytvořený kontroler **AccountController**. Při vytváření nás Visual Studio nutí dodržovat konvence (jméno kontroleru musí končit „Controller“) tím, že při vytváření navrhuje jména a přidává k nim `Controller`. Jako třídu vybereme vybereme naši třídu `Clanek`, kromě kontroleru `OController` (tady vytvoříme kontroler `MVC5 Controller – Empty`). Protože nebudeme potřebovat automaticky generované pohledy (již je máme v kontroleru `ClanekController`), odznačíme volbu `Generate views`.

V kontroleru `Produkty` si ponecháme metodu `Index()` a ostatní vymažeme. Tato metoda se zavolá vždy, když někdo klikne v menu na „Produkty SAF-HOLLAND“. Teď když na ni klikneme, dostaneme chybové hlášení, které nám oznámí, že nebyl nalezen pohled.

V aplikaci / došlo k chybě serveru.

*The view 'Index' or its master was not found or no view engine supports the searched locations. The following locations were searched:*

```
~/Views/Produkty/Index.aspx
~/Views/Produkty/Index.ascx
~/Views/Shared/Index.aspx
~/Views/Shared/Index.ascx
~/Views/Produkty/Index.cshtml
~/Views/Produkty/Index.vbhtml
~/Views/Shared/Index.cshtml
~/Views/Shared/Index.vbhtml
```

**Popis:** Při provádění aktuálního webového požadavku došlo k neošetřené výjimce. Další informace o chybě a o jejím původu v kódu naleznete v trasování zásobníku.

**Podrobnosti o výjimce:** System.InvalidOperationException: The view 'Index' or its master was not found or no view engine supports the searched locations. The following locations were searched:

```
~/Views/Produkty/Index.aspx
~/Views/Produkty/Index.ascx
~/Views/Shared/Index.aspx
~/Views/Shared/Index.ascx
~/Views/Produkty/Index.cshtml
~/Views/Produkty/Index.vbhtml
~/Views/Shared/Index.cshtml
~/Views/Shared/Index.vbhtml
```

Obr. 14 Chybějící pohled

Vytvoříme tedy pohled, klikneme pravým tlačítkem do metody Index v kontoleru, a zvolíme Add View... Protože na stránce vytvoříme seznam vhodných článků, jako šablonu zvolíme List a jako třídu ClaneK a necháme vygenerovat pohled. Vygeneruje se pohled stejný, jako byl vygenerovaný pohled v kontroleru ClaneKController. Pohled upravíme, přidáme obrázek. Rozdělíme na tři sloupce, v levém vytvoříme podmenu, v pravém zobrazíme posledních 5 článků (podle data poslední změny) a v prostředním články označené jako TOP.

Pro podmenu v levém sloupci nám potřebné informace kontroler již předává, protože je potřebujeme pro zobrazení menu úplně vlevo. Použijeme cyklus foreach, kterým projdeme všechny položky podmenu. Pokud patří do menu Produkty, vytvoříme odkaz:

```
@foreach (var item1 in ViewBag.podmenu)
{
    if (item1.MenuId == 3)
    {
        <div class="submenu">
            @{var odkaz = "../" + item1.PodmenuController + "/" + item1.PodmenuAction;}
            <a href=@odkaz>@item1.Nadpis</a>
        </div>
    }
}
```

Pro zobrazení novinek využijeme podobnou konstrukci:

```
<div id="novinky">
    <div class="velkynadpis">
        Novinky
    </div>
    @foreach (var item in Model)
    {
        <div class="datum">
```

```

        @item.DatumPridani.Date.ToString("dd.MM.yyyy")
    </div>
    <div class="novinkynadpis">
        <a href=@item.Odkaz>@item.Nadpis</a>
    </div>
    <div class="abstrakt">
        @item.Abstrakt
        @if (item.Pdf != null)
        {<a class="float-right" href=@item.Pdf target="_blank">pdf</a>}
    </div>
    }
</div>

```

Co je ale Model? Model je objekt, který posílá kontroler do pohledu. Přepneme se tedy do kontroleru a přidáme následující řádek:

```

var clankyall=db.Claneks.Where(p=>p.Platne).Where(p=>!p.Top)
.OrderByDescending(p=>p.DatumPridani).ToList();

```

Tento řádek vytvoří proměnnou *clankyall*, do které uloží seznam (*ToList()*) z databáze (*db.Claneks*), které jsou platné (*Where(p=>p.Platne)*), které nejsou TOP (*Where(p=>!p.TOP)*) seřazené sestupně podle data změny (*OrderByDescending(p=>p.DatumPridani)*).

Dále vytvoříme proměnnou *clanky*, která bude obsahovat prázdný seznam článků. Budeme procházet seznam *clankyall*, a pokud bude článek patřit do správné skupiny, přidáme ho do seznamu *clanky*. Tento seznam omezíme na 5 položek. Nakonec ho předáme pohledu příkazem `return View(clanky)`. Tento seznam *clanky* je tedy onen Model v pohledu. Stejnou konstrukci použijeme pro sloupec TOP, vybereme pouze články, které mají nastavený atribut *TOP*. A tento seznam předáme pohledu v objektu *ViewBag*.

I v kontrolerech *NdSafController* a *OstatniController* vymažeme všechny metody a upravíme metodu *Index* stejně jako v kontroleru *ProduktyController*. A vytvoříme i stejné pohledy *Index*. Samozřejmě upravíme podmínky, aby zobrazované články patřily k danému menu.



Obr. 15 Pohled Index kontroleru ProduktyController

Nyní vytvoříme metody pro položky podmenu. V kontroleru ProduktyController vytvoříme metodu **Napravy()**. Tato metoda vrací navíc v objektu ViewBag seznam oddílů, které potřebujeme pro rozbalení obou menu. Zbytek metody odpovídá metodě **Index()** s tím rozdílem, že kontrolujeme, zda článek patří do podmenu. Pohledu vracíme opět seznam článků. Pohled bude opět podobný, navíc rozbalíme menu – vypíšeme i oddíly vybraného podmenu. Stejně tak vytvoříme i metody pro ostatní podmenu v menu Produkty, a stejně tak zpracujeme menu Ostatní produkty – vytvoříme metody v kontroleru OstatniController a pohledy.



Obr. 16 Pohled Index kontroleru CepyController

Protože se v kontroleru nemůže vnořit do více úrovní, a přitom bychom se potřebovali dostat ještě o úroveň níž, vytvoříme si pro každou položku podmenu, kterou potřebujeme dále rozdělit, nový kontroler, tedy **NapravyController** a **TocnyController**.

U těchto kontrolerů nebudeme potřebovat žádnou automaticky generovanou metodu. Vytvoříme pouze metody pro nižší úroveň, kterou potřebujeme zobrazit. Abychom nemuseli vytvářet ještě další kontrolery pro poslední úroveň, předáme tuto poslední úroveň jako parametr id. Pro nápravy je to vzduchové pérování, vytvoříme tedy metodu **Vzduchove(string id)**, atd. Stejně pro točny.

Kontroler bude opět podobný těm předchozím. Do objektu ViewBag přidáme další seznam pro rozbalení další úrovně menu. Dále musíme kód rozdělit na situaci, kdy bude zadán parametr a kdy nebude.

```
if (id == null)
{
    var clankyall = db.Claneks.OrderByDescending(p => p.DatumPridani).
        Where(e => e.Platne == true).Distinct().ToList();
    var clanky = new List<Clanek>();
    foreach (Clanek clanek in clankyall)
    {
        foreach (var item in clanek.Oddily)
        {
            if (item.Id == 1)
            {
                clanky.Add(clanek);
            }
        }
    }
    return View(clanky);
}
if (id != null)
{
    var pomo = db.Pododdils.Where(e => e.PododdilAction == id).Single();
    int para = pomo.Id;
    var clankyall = db.Claneks.OrderByDescending(p =>
        p.DatumPridani).Where(p => p.Platne).Distinct().ToList();
    var clanky = new List<Clanek>();
    foreach (Clanek clanek in clankyall)
    {
        foreach (var item in clanek.Pododily)
        {
            if (item.Id == para)
            {
                clanky.Add(clanek);
            }
        }
    }
    ViewBag.podnadpis = pomo.Nadpis;
    return View(clanky);
}
```

Pohled bude stejný a zobrazí seznam článků, které mu předá kontroler. Opět vytvoříme stejně i ostatní metody a pohledy pro oba kontrolery.

Pro náhradní díly je dělení trochu odlišné. Do kontroleru přidáme metody **Identifikace()**, **POD()**, **Navody()**, **Info()** a **Dealers()**.

Metoda **Identifikace()** volá pohled, kterému nepředává žádné informace (kromě ViewBag, kde máme seznamy pro menu). Pohled **Identifikace** neobsahuje žádný speciální kód (kromě menu) a obsahuje pouze HTML tagy.

Metoda **POD()** je na tom stejně, její pohled navíc obsahuje JavaScript, který otevře nové okno s adresou aplikace **POD** na webu mateřské firmy, a vrátí se zpět. Pokud je JavaScript zakázaný zobrazí se odkaz pro otevření této aplikace.

Metody **Navody()** a **Info()** odpovídají metodám předchozích kontrolerů **ProduktyController** a **OstatniController()** stejně jako pohledy, zobrazují seznamy článků, které metoda předává.

K metodě **Dealers()** se vrátíme později při tvorbě aplikace.

Posledním kontrolerem, který nás nyní zajímá, je **OController**. Bude mít pouze jednu metodu, a to **Index()**, která zavolá pohled, který obsahuje obyčejný HTML soubor.

Máme tedy vytvořené webové stránky, které využívají databázi pro zobrazení menu a článků, které k menu patří. Zatím se tedy sice jedná o dynamické webové stránky, ale nenazval bych je webovou aplikací. Nemají žádnou hlubší interakci s uživatelem.

## 6 WEBOVÁ APLIKACE

Abychom mohli webové stránky nazvat aplikací, měly by v interakci s uživatelem něco dělat. Často evidují. Evidují různá data. Naše aplikace bude evidovat školení a přihlášené účastníky na ně. Další částí bude evidence vozidel uživatelů. Poslední částí aplikace bude evidence reklamací. Do každého pohledu zapracujeme naše menu na levé straně, proto ho už nebudu zmiňovat.

### 6.1 Školení

#### 6.1.1 Vytvoření školení

První část bude evidence školení. V první řadě budeme potřebovat rozšířit naši databázi. Protože jsme začali metodou *Code First*, budeme v ní i pokračovat. Do našeho projektu přidáme třídy **Skoleni.cs** a **TypSkoleni.cs**.

```
public class TypSkoleni
{
    public int Id { get; set; } //jednoznačný identifikátor
    [Display(Name="Téma školení")]
    public string Nazev { get; set; } //téma školení
    [Display(Name="Popis školení")]
    public string Popis1 { get; set; } //popis školení
    public string Popis2 { get; set; }
    public string Popis3 { get; set; }
    public string Popis4 { get; set; }
}

public class Skoleni
{
    public int Id { get; set; } //jednoznačný identifikátor
    [Display(Name="Téma školení")]
    public int TypSkoleniId { get; set; } //cizí klíč
    [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}", ApplyFormatInEditMode =
true)]
    [Display(Name="Datum školení")]
    public DateTime Datum { get; set; } // datum školení
    [Display(Name = "Téma školení")]
    public virtual TypSkoleni TypSkoleni {get; set;} // virtuální parametr pro
vazbu 1:n
}
```

Dále vytvoříme **SkoleniController**, jako třídu zvolíme naši novou třídu **Skoleni**. Necháme vygenerovat standardní pohledy, při správě je využijeme. Metoda **Index()** předá v Modelu do pohledu seznam školení – pro uživatele od dnešního dne, pro administrátora všechny. Přes **ViewBag** přidáme seznam, který obsahuje typy školení.

Pohled vypíše pevný text s obecnými informacemi o školení. Dále použijeme foreach cyklus pro výpis typů školení. Pro každý typ vypíšeme jeho informace a vnořeným foreach cyklem projdeme seznam školení a pro daný typ vypíšeme termíny a ke každému termínu odkaz pro přihlášení na toto školení.

Pro větší přehlednost, termíny pomocí JavaScriptu skryjeme a vytvoříme tlačítko, pomocí kterého je můžeme zase zobrazit.

Pro platby v CZK: 50084046/6200 *Commerzbank AG, Praha*  
 Pro platby v EUR: 70069767/8050 IBAN: SK28 8050 0000 0000 7006 9767 BIC: COBASKBXXXX *Commerzbank AG, Bratislava*  
 Celková částka = počet osob x 2.178 Kč nebo x 74 EUR v případě SK zákazníka, VS: Vaše IČO, KS: 0308.  
 Na základě zaplaceného poplatku obdržíte obratem daňový doklad + potvrzení účasti s upřesňujícími informacemi.  
 V případě potřeby si vyžádejte zálohovou fakturu.  
 O úspěšně absolvovaném školení obdrží účastníci SAF-HOLLAND certifikát.

**Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND**

- účastník se seznámí se všemi agregáty, všemi nápravami a s technologií SAF INTEGRAL
- možnosti údržby a opravy, přiřazení ND příslušnému typu nápravy
- práce s domovskými stránkami SAF-HOLLAND a se servisní literaturou
- praxe- práce a opravy na kotoučových brzdách, rozbor a odstranění závad ve speciálních případech

Zobrazit termíny

**Identifikace, objednávání a řízení náhradních dílů a opravářských sad SAF-HOLLAND**

- účastník se seznámí se systematickým přiřazením náhradních dílů k aktuálním produktům SAF-HOLLAND a jejich rozšířením na trhu, rady pro zvlášť přehledné a hospodárné udržování skladu
- doporučení pro skladové zásoby s důrazem na opravy
- práce s SAF elektronickým katalogem náhradních dílů, identifikace náhradních dílů
- práce s domovskými stránkami SAF-HOLLAND

Skrýt termíny

Datum školení	
10.11.2014	<a href="#">Přihlásit na školení</a>
11.11.2014	<a href="#">Přihlásit na školení</a>

**Řešení reklamací SAF-HOLLAND - ohledání, zpracování, vyřízení**

- účastník se seznámí se všemi agregáty, všemi nápravami a s technologií SAF INTEGRAL včetně záruky

Obr. 17 Pohled Index kontroleru SkoleniController

## 6.1.2 Vytvoření firmy

Aby se bylo možné přihlašovat, budeme potřebovat databázi ještě rozšířit. Protože víme, že zákazníci firmy jsou zase pouze firmy, přidáme třídu **Firma.cs**. Firma bude obsahovat adresu, aby databáze splňovala NF, vytvoříme třídy **Mesto.cs** a **Stat.cs**. I když zákazníci budou firmy, na školení se budou přihlašovat jednotliví lidé. Vytvoříme tedy ještě třídu **Zamestnanec.cs**. A k ní budeme potřebovat třídu **Titul.cs** a třídu **Funkce.cs**

```
public class Stat
{
    public int Id { get; set; }
    public string Nazev { get; set; }
    public string Zkratka { get; set; }
}
```

```
public class Mesto
{
    public int Id { get; set; }
    public string Nazev { get; set; }
    [Display(Name="PSČ")]
    public string PSC { get; set; }
    public int? KrajId { get; set; }
    public int? StatId { get; set; }
    public virtual Kraj Kraj { get; set; }
    [Display(Name="Stát")]
    public virtual Stat Stat { get; set; }
}

public class Firma
{
    public Firma()
    {
        Zamestnanci = new List<Zamestnanec>();
    }
    public int Id { get; set; }
    [Display(Name="Jméno firmy")]
    public String Nazev { get; set; }
    public String Ulice { get; set; }
    [Display(Name="č.p.")]
    public string Cp { get; set; }
    [Display(Name="č.o.")]
    public string Co {get; set;}
    public int MestoId { get; set; }
    public bool Dealer { get; set; }
    public bool Servis { get; set; }
    public bool Doprava { get; set; }
    [Display(Name="IČO")]
    [MinLength(8)]
    [MaxLength(8)]
    public String Ico { get; set; }
    [Display(Name="DIČ")]
    [StringLength(10)]
    public String Dic { get; set; }
    public DateTime Vytvoreni { get; set; }
    public string VytvorilId { get; set; }
    public float? Lat { get; set; }
    public float? Lng { get; set; }
    [Display(Name="Město")]
    public virtual Mesto Mesto { get; set; }
    public virtual ICollection<Zamestnanec> Zamestnanci { get; set; }
}

public class Titul
{
    public int Id { get; set; }
    [Display(Name="Titul")]
    public string NazTitul { get; set; }
}

public class Funkce
{
    public int Id { get; set; }
    [Display(Name="Funkce")]
    public string NazFunkce { get; set; }
}
```

```

public class Zamestnanec
{
    public int Id { get; set; }
    public int? TitulId { get; set; }
    [Required(ErrorMessage="Jméno musí být zadáno")]
    [Display(Name="Jméno")]
    public string Jmeno { get; set; }
    [Required(ErrorMessage="Příjmení musí být zadáno")]
    [Display(Name="Příjmení")]
    public string Prijmeni { get; set; }
    public string Telefon { get; set; }
    public string Fax { get; set; }
    public string Mobil { get; set; }
    [DataType(DataType.EmailAddress)]
    public string Email { get; set; }
    public int FirmaId { get; set; }
    public string UserId { get; set; }
    [Display(Name="Datum narození")]
    public DateTime? DatumNarozeni { get; set; }
    [Display(Name="Datum vytvoření")]
    public DateTime Vytvoreni { get; set; }
    [Display(Name="Je platný?")]
    public bool Platny { get; set; }
    public virtual Funkce Funkce { get; set; }
    public virtual Firma Firma { get; set; }
    public virtual Titul Titul { get; set; }
    public virtual ICollection<Skoleni> Skolenis { get; set; }
}

```

Navíc musíme rozšířit třídu **Skoleni.cs**.

```

public class Skoleni
{
    public Skoleni()
    {
        this.Zamestnanci = new HashSet<Zamestnanec>();
    }
    public int Id { get; set; }
    [Display(Name="Téma školení")]
    public int TypSkoleniId { get; set; }
    [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}", ApplyFormatInEditMode =
true)]
    [Display(Name="Datum školení")]
    public DateTime Datum { get; set; }
    [Display(Name = "Téma školení")]
    public virtual TypSkoleni TypSkoleni {get; set;}
    [Display(Name="Účastníci školení")]
    public virtual ICollection<Zamestnanec> Zamestnanci { get; set; }
}

```

Tím vytvoříme vazbu n:m mezi tabulkami Skoleni a Zamestnanec.

### 6.1.3 Migrace

Pokud se pokusíme nyní aplikaci spustit, dostaneme chybové hlášení.

## V aplikaci / došlo k chybě serveru.

*The model backing the 'ApplicationDbContext' context has changed since the database was created. Consider using Code First Migrations to update the database (<http://go.microsoft.com/fwlink/?LinkId=238269>).*

**Popis:** Při provádění aktuálního webového požadavku došlo k neošetřené výjimce. Další informace o chybě a o jejím původu v kódu naleznete v trasování zásobníku.

**Podrobnosti o výjimce:** System.InvalidOperationException: The model backing the 'ApplicationDbContext' context has changed since the database was created. Consider using Code First Migrations to update the database (<http://go.microsoft.com/fwlink/?LinkId=238269>).

**Zdrojová chyba:**

```
Řádek 21:      {
Řádek 22:          DateTime dnes = DateTime.Now;
Řádek 23:          var skolenis = db.Skolenis.Where(s => s.Datum > dnes).Include(s => s.TypSkoleni);
Řádek 24:          if (User.IsInRole("admin")) skolenis = db.Skolenis.Include(s => s.TypSkoleni);
Řádek 25:          var oddil = db.Oddils.ToList();
```

**Zdrojový soubor:** c:\Users\lmachala.SAF CZ\Documents\Visual Studio 2013\Projects\saf-holland\saf-holland\Controllers\SkoleniController.cs **Řádek:** 23

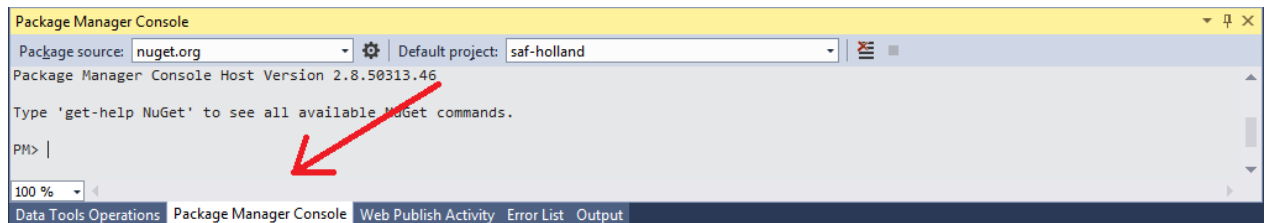
**Trasování zásobníku:**

```
[InvalidOperationException: The model backing the 'ApplicationDbContext' context has changed since the database was created. Consider using Code Fi
System.Data.Entity.<>c__DisplayClass1.<InitializeDatabase>b__0() +130
System.Data.Entity.Internal.MigrationsChecker.IsMigrationsConfigured(InternalContext internalContext, Func`1 databaseExists) +59
System.Data.Entity.CreateDatabaseIfNotExists`1.InitializeDatabase(TContext context) +193
```

Obr. 18 Chybové hlášení po změně ve třídě Skoleni

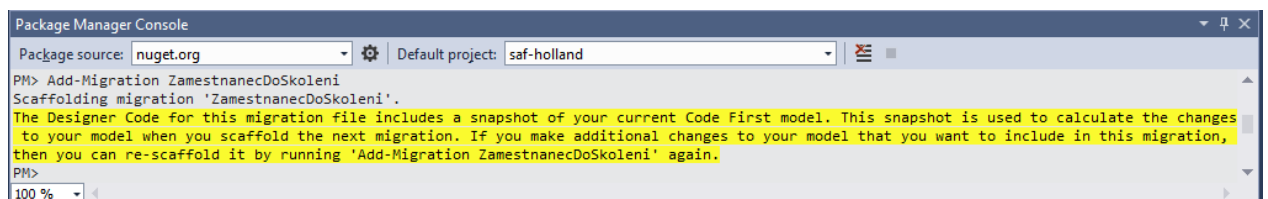
Změnili jsme třídy (Skoleni.cs), ale nepromítli jsme změny do databáze. K tomu existuje nástroj, který se jmenuje migrace.

Přepneme se do **Package Manager Console** (pokud ji nevidíme, zobrazíme ji přes menu Tools => NuGet Package Manager => Package Manager Console).



Obr. 19 Package Manager Console

Do příkazové řádky zadáme příkaz *Enable-Migrations*. Tento příkaz přidá do projektu adresář *Migrations*, a do něj konfigurační soubor **Configuration.cs**. Novou migraci vytvoříme příkazem *Add-Migration*. Za tento příkaz přidáme své označení migrace. V našem případě zadáme Add-Migration ZamestnanecDoSkoleni.



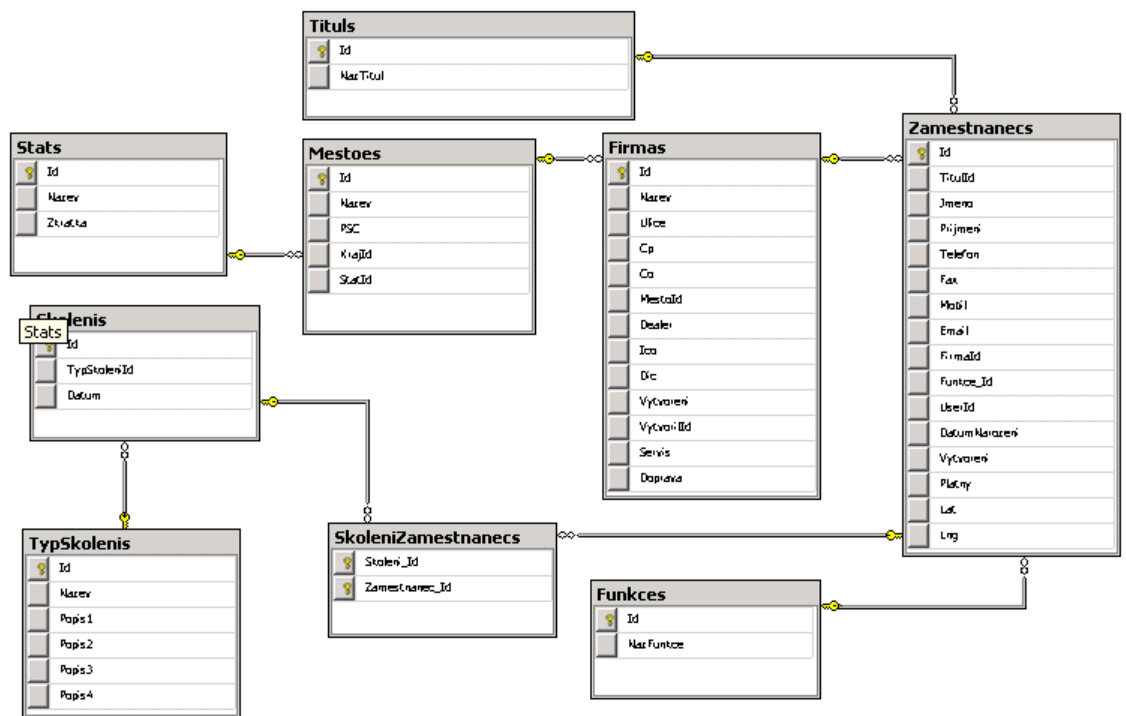
Obr. 20 Migrace

Do adresáře Migrations přibude nová třída, která se má jméno ve tvaru `yyyyymmddhhmsss_ZamestnanecDoSkoleni`. Třída obsahuje dvě metody **Up()** a **Down()**. Tyto metody obsahují kód, který se vykoná při aktualizování na novější verzi (Up), případně při obnově na starší verzi (Down). Zatím se ale neprovedly žádné změny v databázi. Ty se provedou až po provedení příkazu *Update-Database*.

```
PM> Update-Database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying explicit migrations: [201405151736212_ZamestnanecDoSkoleni].
Applying explicit migration: 201405151736212_ZamestnanecDoSkoleni.
Running Seed method.
PM> |
```

Obr. 21 Aktualizace databáze

Nyní již jsou tabulky v databázi upraveny. A aplikace se bez problému spustí.



Obr. 22 Schéma databázových tabulek pro evidenci školení

#### 6.1.4 Přihlášení na školení

Přidáme tedy do kontroleru metodu pro přihlášení na školení. Metoda se jmenuje **Prihlasiť(int? id)**, id je interní číslo školení. Kontroler předá pohledu školení, které najde podle id.

Přidáme pohled, použijeme šablonu **Detail** a třídu **Skoleni**. Upravíme vzhled a přidáme formulář. Pro vytvoření formuláře použijeme direktivu `@using Html.BeginForm`. Formulář vypadá nějak takto.

Obr. 23 Přihlášení na školení

Protože jsme si formulář vytvořili sami, musíme se sami postarat o validaci, tj. o kontrolu, zda uživatel zadal to, co očekáváme, nebo zadal nesmysly. Validaci uděláme dvakrát. Poprvé v JavaScriptu, ať zbytečně neposíláme data serveru a podruhé na serveru, protože nevíme, zda uživatel nemá JavaScript vypnutý.

Přidáme tedy do formuláře kód, který sice v normálním stavu nic nevypíše, ale v případě chyby v zadání se přihlásí o slovo.

```
@Html.TextBox("Jmeno")
<span id="jmenokontrola" class="kontrola"></span>
```

Klikneme pravým tlačítkem na složku Scripts a vybereme Add => JavaScript. Protože ASP.NET automaticky přidává do projektu knihovnu jQuery (vytváří s její pomocí validátory), neexistuje jediný důvod, proč této knihovny nevyužít. Vložíme tedy skript

```
$(document).ready(function () {
    $("#prihlasit").submit(function () {
        return CheckForm(this);
    });});
```

Tento skript se automaticky spustí při načtení kompletního dokumentu. Provede to, že před odesláním formuláře, který má id *prihlasit*, zavolá funkce **CheckForm**, která vrací *true* nebo *false*. Při *false* se formulář neodešle.

Zbývá napsat funkci **CheckForm**. Vytvoříme si proměnné, do kterých načteme hodnoty polí formuláře. Nastavíme rámeček na standardní parametry, a vymažeme pole, ve kterých můžou být hlášení z minulého volání skriptu. Vytvoříme proměnnou *chyba* a nastavíme ji na nulu. Nyní procházíme jednu proměnnou za druhou, a ověřujeme zda, její hodnota odpovídá tomu, co očekáváme. Pokud ne, nastavíme červený rámeček kolem špatně zadaného pole, a vypíšeme chybové hlášení, a navýšíme hodnotu proměnné *chyba*. Protože většina polí má typ string, většinou řešíme, zda bylo něco zadáno nebo ne. Výjimku tvoří email, kde kontrolujeme, zda skladba řetězce odpovídá emailu. Další výjimkou jsou telefonní čísla, zde kontrolujeme, zda byly zadány pouze číslice. To u IČO kontrolujeme také, ale navíc i zda je číslo platné. Skladba IČO má svoje pravidla, poslední číslice je kontrolní.

Protože necháváme uživatele zadávat datum, je vhodné vytvořit nějaký nástroj pro zadání data. Každý uživatel zadává datum podle svého zvyku, jeden píše mezi čísly lomítka, jiný tečky, další píše měsíc slovy, atd. Abychom nemuseli kontrolovat, jaký styl má který uživatel, vložíme do stránek nástroj, kde si uživatel datum vybere a do formuláře se vloží ve formátu, který nám vyhovuje. Tento nástroj obsahuje knihovna jQuery, respektive její nástavba jQuery UI. Jmenuje DatePicker a vkládá se pomocí JavaScriptu. Má relativně velké možnosti nastavení zobrazení a je jednoduchý na použití.

```
$(function () {
    $("#datumnarozeni").datepicker({
        showWeek: false,
        firstDay: 1,
        changeMonth: true,
        changeYear: true,
        maxDate: "-17Y",
        minDate: "-118Y",
        showAnim: "fold",
        yearRange: "c-100:c+1"
    },
    $.datepicker.regional["cs"]);
});
```

Pokud je po prověření všech polí *chyba* rovno nule, formulář se odešle. Pokud někde byla chyba navýšena, skript vrátí *false* a formulář se neodešle.

Pokud je všechno v pořádku (případně v prohlížeči JavaScript nefunguje), odešlou se data na server, kde je přijme náš kontroler svojí druhou metodou **Prihlasit**. Tato metoda očeká-

vá zadání všech hodnot do formuláře. Nejprve vytvoříme nový seznam řetězců, pojmenujeme ho `errors: List<string> errors = new List<string>()`;

Procházíme všechny parametry, jeden po druhém a kontrolujeme, odpovídá tomu, co očekáváme. Pokud ano, přidáme chybové hlášení do seznamu: `errors.Add("Jméno musí být vyplněno!");` Takto projdeme všechny parametry. Pokud není na konci seznam `errors` prázdný, přidáme seznam do objektu `ViewBag` a zobrazíme formulář znovu. Navíc vypíšeme i chybové hlášky.

```
if (ViewBag.error != null)
{
    <div class="kontrola">
        <ul>
            @foreach (var item in ViewBag.error)
            {
                <li>
                    @item
                </li>
            }
        </ul>
    </div>
}
```

16.06.2014  
Téma školení  
Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND

- Jméno musí být vyplněno!
- Příjmení musí být vyplněno!
- Nějakou funkci určitě máte...
- Alespoň jeden telefon je nutné zadat!
- Email musí být vyplněn!
- Datum narození musí být vyplněno!
- Jméno firmy musí být vyplněno!
- Ulice musí být vyplněna!
- Město musí být vyplněno!
- IČO musí být vyplněno!
- Opravdu jste neplátce DPH?? V tom případě vyplňte 'neplátce', prosím.

Titul  Jméno  Příjmení  Funkce

Telefon  Fax  Mobil  Email

Telefon zadejte v mezinárodním formátu 0042x xxx xxx xxx Fax zadejte v mezinárodním formátu 0042x xxx xxx xxx Mobil zadejte v mezinárodním formátu 0042x xxx xxx xxx

Datum narození

Firma  Ulice  Č.p./Č.o.  Město

IČO  DIČ

Obr. 24 Vrácený formulář po validaci na serveru

Pokud je vše v pořádku, vytvoříme novou proměnnou třídy `Zamestnanec`. Přiřadíme jí zadané hodnoty, přidáme zaměstnance do seznamu účastníků školení a přidáme záznam do databáze do tabulky `Zamestnanec`. Dále vytvoříme novou proměnnou třídy `Firma`. Přiřadí-

me jí zadané hodnoty, přiřadíme nově vytvořeného zaměstnance a uložíme záznam do databáze do tabulky Firma. Ještě před tím zkontrolujeme, zda firma se stejným IČO již v databázi neexistuje. Pokud ano, přiřadíme zaměstnance do stávající firmy a nezakládáme novou.

```
foreach (var item in firmyall)
{
    if ((newfirma.Ico != null) && (newfirma.Ico == item.Ico))
    {
        newzamestnanec.Firma = item;
        db.Zamestnanecs.Add(newzamestnanec);
        if (errors.Count == 0)
        {
            db.SaveChanges();
        }
        ViewBag.error = errors;
        ViewBag.prihlasen = 2;
        return View();
    }
}
db.Firmas.Add(newfirma);
newzamestnanec.Firma = newfirma;
db.Zamestnanecs.Add(newzamestnanec);
db.Skolenis.Find(id).Zamestnanci.Add(newzamestnanec);
if (errors.Count == 0)
{
    db.SaveChanges();
}
```

Nyní se tedy může kdokoliv přihlásit na školení. Když ale máme v databázi vytvořenou tabulku, ve které je možné evidovat lidi, využijme ji pro evidenci uživatelů.

### 6.1.5 Registrace do vlastní aplikace

Uživatel, který se bude chtít přihlásit na více školení, je nyní nucen vždy vyplnit kompletní formulář. I když jeho údaje a údaje firmy máme v databázi uloženy. Proto je vhodné umožnit mu přihlášení do aplikace, a poté se přihlásí na libovolný počet školení pouze jedním kliknutím.

Využijeme k tomu vytvořený **AccountController** a jeho metody a pohledy.

Zajímat nás bude metoda **Register**. Po jejím zavolání se zobrazí jednoduchý formulář, který je vytvořen pohledem Register.cshmtl. Tento formulář je připravený pro vytvoření uživatelského účtu. My si ho rozšíříme pro vytvoření záznamu v tabulce Zamestnanec. Přidáme tedy do formuláře pole, které jsou stejné s poli u přihlašování na školení. Formulář pro registraci vypadá tedy takto.

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Královské čepy
- Kuličkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky
- Gumové zástěrky
- Systém AntiSpray
- Tažné zařízení
- Odstavné nohy a navijáky
- Příčky a oje
- Ventily a EBS
- Ostatní

**SERVIS**

- Reklamacce
- Členění

### Registrace

Titul <input type="text"/>	Jméno <input type="text"/>	Příjmení <input type="text"/>	Funkce <input type="text"/>
Telefon 0042 <input type="text"/> <small>Telefon zadejte v mezinárodním formátu 0042x xxx xxx xxx</small>	Fax 0042 <input type="text"/> <small>Fax zadejte v mezinárodním formátu 0042x xxx xxx xxx</small>	Mobil 0042 <input type="text"/> <small>Mobil zadejte v mezinárodním formátu 0042x xxx xxx xxx</small>	Email <input type="text"/>
Datum narození <input type="text"/>	Firma <input type="text"/>	Ulice <input type="text"/>	Č.p./Č.o. <input type="text"/>
IČO <input type="text"/>	DIČ <input type="text"/>	Město <input type="text"/>	

**Uživatelské jméno**

**Heslo**

**Zopakujte heslo**

Obr. 25 Registrace nového uživatele

Protože jsme si opět formulář vytvořili sami, musíme se postarat o validaci. Validace proběhne stejným způsobem jako u přihlašování na školení. Tzn. nejprve pomocí skriptu pouze v prohlížeči, v případě, že kontrola proběhne v pořádku nebo neproběhne vůbec, pošlou se data na server, kde proběhne kontrola druhá<sup>1</sup>.

<sup>1</sup> Ve skutečnosti probíhají kontroly tři. Třetí kontrola probíhá na databázovém serveru, kde se kontroluje, zda je možné data zapsat do daného sloupce. Pokud vrátí aplikace chybové hlášení.

Obr. 26 Validace na straně klienta - JavaScript

Pokud všechny kontroly proběhnou v pořádku, vytvoří se nový záznam v tabulce Zamestnanec, pokud neexistuje v databázi firma se stejným IČO, vytvoří se nový záznam v tabulce Firma, k této firmě se přiřadí vytvořený zaměstnanec. Dále se vytvoří nový záznam v tabulce AspNetUsers. Jedná se o tabulku uživatelů, kterou vytvořilo Visual Studio z použité šablony. Tato tabulka obsahuje sloupce Id (jednoznačný identifikátor), UserName (uživatelské jméno), PasswordHash (uložený otisk hesla). Další sloupce v naší aplikaci zatím nevyužijeme.

Name	Data Type	Allow Nulls	Default
Id	nvarchar(128)	<input type="checkbox"/>	
UserName	nvarchar(MAX)	<input checked="" type="checkbox"/>	
PasswordHash	nvarchar(MAX)	<input checked="" type="checkbox"/>	
SecurityStamp	nvarchar(MAX)	<input checked="" type="checkbox"/>	
Discriminator	nvarchar(128)	<input type="checkbox"/>	

Keys (1)  
 PK\_dbo.AspNetUsers (Primary Key, Clustered: Id)  
 Check Constraints (0)  
 Indexes (0)  
 Foreign Keys (0)  
 Triggers (0)

Obr. 27 Tabulka AspNetUsers

Protože nikdy není vhodné ukládat heslo v otevřeném textu, ukládáme tzv. hash hesla. Hash je zjednodušeně řečeno jednocestně zakódované heslo. Jednocestně protože se současnou technikou není možné z hashe zpětně vypočítat původní heslo. Má důležitou

vlastnost – ze stejného vloženého řetězce nám hashovací funkce vždy vrátí stejný hash. Proto aplikace dokáže uživatele přihlásit, i když nezná jeho heslo (zná hash jeho hesla).

### 6.1.6 Přihlášení

Přihlášení do aplikace mají na starosti metody **Login** kontroleru AccountController. První, jejímž vstupním parametrem je řetězec s url adresou, ze které se tato metoda volá, pouze zavolá pohled Login.cshtml, kterému předá v objektu ViewBag svůj vstupní parametr.

Formulář je jednoduchý, obsahuje pouze pole pro zadání uživatelského jména a hesla, a checkbox pro vytvoření permanentní cookie, kdy si prohlížeč zapamatuje přihlášení. V tomto případě necháme ověření na šabloně.



The image shows a web form for logging in. On the left, there is a partial view of a table with columns like 'ity', 'ND', and 'KTY'. The main form is titled 'Přihlášení' and contains the following elements:

- A text input field for 'Uživatelské jméno' (Username) with a red asterisk indicating it is required. Below it is the red text 'Zadejte své uživatelské jméno'.
- A text input field for 'Heslo' (Password) with a red asterisk. Below it is the red text 'Zadejte Vaše heslo'.
- A checkbox labeled 'Zapamatovat?' (Remember?).
- A button labeled 'Přihlásit' (Login).
- A link at the bottom right that says 'Zaregistrujte se, pokud' (Register if).

Obr. 28 Validace přihlašovacího formuláře

Pohled obsahuje i kód, pro přihlášení pomocí jiných externích služeb (Facebook, Google+ nebo Twitter). Tento kód zatím označíme jako poznámku pro případ, že v budoucnu budeme chtít tuto možnost využít.

Druhá metoda přijímá zadané údaje, provede ověření, v případě platných údajů uživatele přihlásí a vrátí na stránku, ze které na přihlášení přišel, v opačném případě zobrazí formulář znovu s chybovým hlášením.

### 6.1.7 Správa uživatelských informací

Že je uživatel přihlášen, pozná v pravém horním rohu, kde se zobrazují odkazy pro přihlášení a registraci, pokud uživatel není přihlášený. A odkazy pro odhlášení pro správu uživatelských informací, pokud přihlášený uživatel je.



Obr. 29 Nepřihlášený uživatel



Obr. 30 Přihlášený uživatel

Správu uživatelského profilu má na starost metoda **Manage** kontroleru AccountController. Je podobná metodě Register a my ji upravíme úplně stejně, stejně tak i její pohled. Pohled původně obsahoval pouze pole pro změnu hesla, tj. zadání původního hesla, nového hesla a ověření nového hesla. My přidáme další pole o zaměstnanci a o firmě, do kterých načteme patřičná data z databáze. Při odeslání provedeme stejnou validaci jako v případě registrace jak v prohlížeči JavaScriptem, tak opět na serveru kontrolerem. Pokud vše proběhne v pořádku, uložíme změněné údaje do databáze, v opačném případě vypíšeme chybové hlášení, aby uživatel věděl, co má opravit.

### 6.1.8 Přihlášení na školení registrovaného uživatele

Upravíme nyní metodu a pohled **Index** kontroleru SkoleniController, tak aby se registrovaný uživatel mohl snadno na školení přihlásit, aby se mohl odhlásit, a mohl se podívat na detaily školení, které dostupné neregistrovanému uživateli nejsou.

V metodě Index připišeme kód, který do objektu ViewBag přidá seznam školení, na kterých je přihlášený aktuální uživatel.

```
foreach (var item in zamestnanci)
{
    if (uzivatelId == item.UserId)
    {
        zamestnanec = item;
        var skoleni = zamestnanec.Skolenis.ToList();
        ViewBag.skoleni = skoleni;
    }
}
```

Do výpisu školení přidáme odkaz na detail školení a na odhlášení ze školení. Odkaz na detail zobrazíme pouze přihlášenému uživateli, a odkaz na odhlášení pouze pokud je na daném školení přihlášen.

```

@if (Request.IsAuthenticated)
{
    <td>
        @Html.ActionLink("Detail", "Details", new { id = item1.Id })
    </td>
}
@if (Request.IsAuthenticated)
{
    foreach (var skol in ViewBag.skoleni)
    {
        if (skol.Id == item1.Id)
        {
            <td>
                @Html.ActionLink("Odhlásit se ze školení", "Odhlasit", new { id
                = item1.Id })
            </td>
        }
    }
}

```

Dále musíme upravit pohled **Přihlase**ni, aby pro přihlášeného uživatele nezobrazoval kompletní formulář.

Protože předpokládáme, že jeden uživatel (vedoucí) bude přihlašovat více lidí, kteří si nebudou vytvářet vlastní účty v aplikaci, dáme uživateli možnost přihlásit více zaměstnanců ze stejné firmy. Přidáme tedy dvě tlačítka, jedním se přihlásí na školení aktuální uživatel, druhé zobrazí seznam zaměstnanců, jejichž data již jsou v databázi uloženy a patří ke stejné firmě jako aktuální uživatel. Zároveň s tímto seznamem se zobrazí formulář pro zadání nového zaměstnance.

```

<div class="col-md-10">
    <input type="submit" name="sebe" id="sebe" class="button" value="sebe" />
    <input type="submit" name="sebe" id="nesebe" class="button" value="kolegu" />
</div>

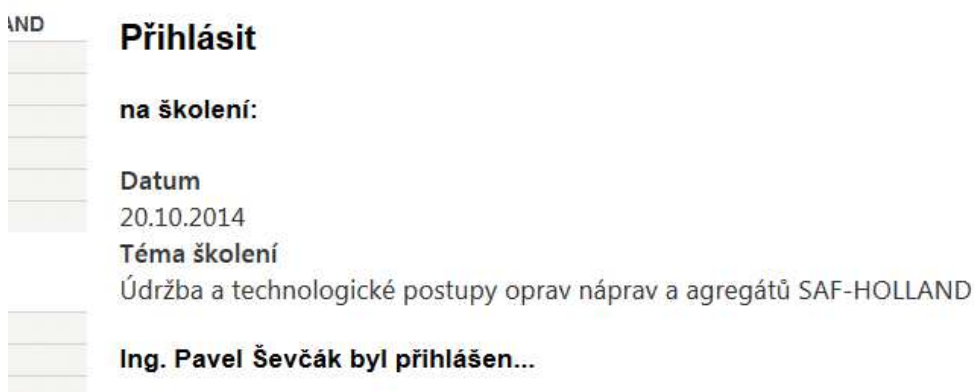
```

Kuličkové dráty	<b>Datum</b>	20.10.2014		
Ostatní	<b>Téma školení</b>	Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND		
<b>HRADNÍ DÍLY</b>	<b>Chcete přihlásit:</b>			
<b>F-HOLLAND</b>	<b>Kolegové</b>	<input type="text"/>		
Jak identifikovat ND	<b>Nový kolega:</b>			
Parts On Demand	<b>Titul</b>	<b>Jméno</b>	<b>Příjmení</b>	<b>Funkce</b>
Montážní návody	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Důležité informace	<b>Telefon</b>	<b>Fax</b>	<b>Mobil</b>	<b>Email</b>
Kde koupit	0042	0042	0042	
<b>TATNÍ PRODUKTY</b>	<small>Telefon zadejte v mezinárodním formátu 0042x xxx xxx xxx</small>	<small>Fax zadejte v mezinárodním formátu 0042x xxx xxx xxx</small>	<small>Mobil zadejte v mezinárodním formátu 0042x xxx xxx xxx</small>	
System osvětlení	<b>Datum narození</b>	<input type="text"/>		
Plastové blatníky a skříňky				
Gumové zástěrky	<input type="text"/>			
System AntiSpray	<input type="text"/>			
Tažné zařízení	<input type="text"/>			
Odstavné nohy a navijáky				
Příčky a oje				
Ventily a EBS				
Ostatní				
<b>RVIS</b>	<input type="button" value="Přihlásit"/>			
Reklamacce				
Školení				

[Zpět na seznam školení](#)

Obr. 31 Přihlášení na školení

V případě, že uživatel klikne na tlačítko sebe, kontroler přidá aktuálního uživatele do seznamu uživatelů přihlášených na školení. Kliknutím na tlačítko kolegu se zobrazí další formulář, uživatel může vybrat ze seznamu nebo vyplnit formulář. Po kliknutí na tlačítko přihlásit, kontroler prověří, zda je vybrán záznam ze seznamu, nebo vyplněný formulář. Podle toho buď přidá vybraného zaměstnance do seznamu přihlášených na aktuální školení, nebo vloží nový záznam do tabulky Zamestnanec, přidá ho seznamu zaměstnanců firmy, aktuálně přihlášeného uživatele a přidá ho na seznam přihlášených na aktuální školení.



Obr. 32 Potvrzení přihlášení na školení

### 6.1.9 Odhlášení se ze školení

Přihlášený uživatel se může ze školení odhlásit kliknutím na odkaz Odhlásit se ze školení. To zavolá Metodu **Odhlasit** kontroleru SkoleniController. Ta zavolá v prvním kroku pohled Odhlasit, kterému předá dané školení. Tento pohled zobrazí dotaz na potvrzení odhlášení.



Obr. 33 Odhlášení se ze školení

Po potvrzení je přihlášený uživatel smazaný ze seznamu přihlášených účastníků.

```
foreach (var item in zamestnanci)
{
    if (uzivatelId == item.UserId)
    {
        db.Skolenis.Find(id).Zamestnanci.Remove(item);
        db.SaveChanges();
        ViewBag.hlaseni = "Právě jste se odhlásil...";
        ViewBag.odhlasen = 1;
    }
}
```

## 6.2 Kontakty

Nyní se vrátíme ke kontaktům, které jsme přeskočili. Kontroler **KontaktController** bude mít pouze jednu metodu, a to metodu **Index**. Předpokládejme, že první firmou v databázi (tedy firmou, jejíž Id bude rovno jedné) je firma SAF-HOLLAND Czechia. Přidáme do objektu ViewBag tuto firmu a v modelu předáme pohledu seznam jejích zaměstnanců.

```
var firma = db.Firmas.Find(1);
ViewBag.SAF = firma;
ViewBag.Mesto = db.Mestoos.Where(p => p.Id == firma.MestoId).Single();
var zamestnanecs = db.Zamestnanecs.Where(z=>z.Firma.Id == firma.Id).Include(z =>
z.Firma).Include(z => z.Titul);
return View(zamestnanecs.ToList());
```

Pohled vypíše informace o firmě z ViewBag, a prochází model, při čemž vypisuje informace o jednotlivých zaměstnancích. Až projde všechny zaměstnance, přidáme pomocí GoogleMaps API mapu, na kterou přidáme značku na správné adrese.

### 6.2.1 Mapa

Vložení mapy je nyní díky API od Google velmi snadné. V pohledu pouze vytvoříme prostor pro mapu pomocí tagu `<div> </div>`. Před tím musíme vložit skript uložený na webu googlu, který mapu zobrazí: `<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>`. K tomu přidáme náš skript, který skriptu od Google řekne, jak má mapu zobrazit: `<script type="text/javascript" src="/Scripts/kontakt.js"></script>`

```
function initialize() {
    var myOptions = {
        zoom: 15,
        center: new google.maps.LatLng(49.0584474, 17.4602263),
        myTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById('map_canvas'), myOptions);
    var options = {
        position: new google.maps.LatLng(49.0586210, 17.4601080),
    };
};
```

```

var marker = new google.maps.Marker(options);
marker.setMap(map);
}

```

API je na webových stránkách dobře popsáno a není problém vytvořit jakékoliv nastavení mapy. Je důležité upozornit, že API je zdarma pouze za určitých omezení. V naší aplikaci nám tato omezení nijak nevadí.

stern AntiSpray
žně zařízení
řstavné nohy a navijáky
čky a oje
ntily a EBS
řtatní
<b>IS</b>
klamace
olení

*Logistika a odbyt***Marcel Machala**

Tel: +420 572 540 903

Fax: +420 572 540 933

Mob: +420 572 540 933

E-mail: [marcel.machala@safholland.cz](mailto:marcel.machala@safholland.cz)*Obchodně technický zástupce prodej náhradních dílů***Martin Chmelík**

Tel: +420 572 557 189

Fax: +420 572 540 933

Mob: +420 724 064 458

E-mail: [martin.chmelik@safholland.cz](mailto:martin.chmelik@safholland.cz)*Skladník***Michal Habarta**

Tel: +420 572 557 188

Fax: +420 572 540 933

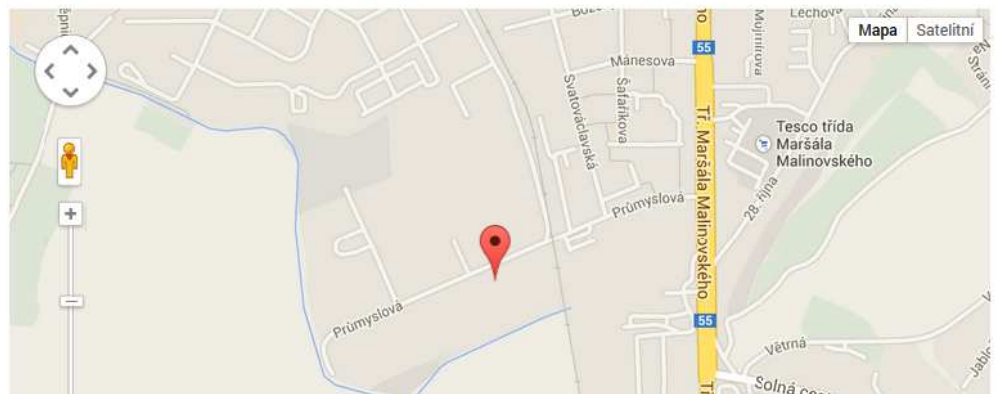
Mob: +420 572 540 933

E-mail: [sklad@safholland.cz](mailto:sklad@safholland.cz)*Skladník***Miroslav Strýček**

Tel: +420 572 557 188

Fax: +420 572 540 933

Mob:

E-mail: [sklad@safholland.cz](mailto:sklad@safholland.cz)

Obr. 34 Zobrazení mapy

## 6.2.2 Metoda Dealers() kontroleru NdSafController

Další částí, kterou jsme přeskočili, je metoda **Dealers()** kontroleru **NdSafController**. Přeskočili jsme ji, protože jsme neměli vytvořenou tabulku **Firma**. Na této stránce se vypíše partnerské firmy s adresou a kontakty, dále se zobrazí na mapě.

Aby bylo možné firmy lépe členit, přidáme do databáze tabulku **Kraj**. Vytvoříme tedy třídu **Kraj.cs**. Tato třída bude jednoduchá, bude obsahovat pouze dva atributy, **Id** a **Nazev**. Dále upravíme třídu **Mesto**, tak aby nám vznikla vazba 1:n.

Metoda **Dealers()** předá pohledu seznam firem, které mají parametr **Dealer** nastavený na **true**. Pomocí **ViewBag** předáme seznam krajů.

Vytvoříme tedy pohled. Nejprve vytvoříme formulář, který použijeme pro filtrování firem. Ve formuláři bude rozbalovací seznam krajů a pole, do kterého zadáme část jména firmy. Po odeslání se opět zavolá stejná metoda, která omezí vrácený seznam firem podle zadaných údajů.

```
if (!String.IsNullOrEmpty(id))
{
    dealers = dealers.Where(s => s.Nazev.Contains(id));
}
if (!String.IsNullOrEmpty(kraj))
{
    dealers = dealers.Where(s => s.Mesto.Kraj.Nazev == kraj);
}
```

Tento omezený seznam opět dostane pohled, který firmy zobrazí. Pod zobrazený seznam opět přidáme mapu stejným způsobem, jako když jsme ji vkládali do kontaktů.

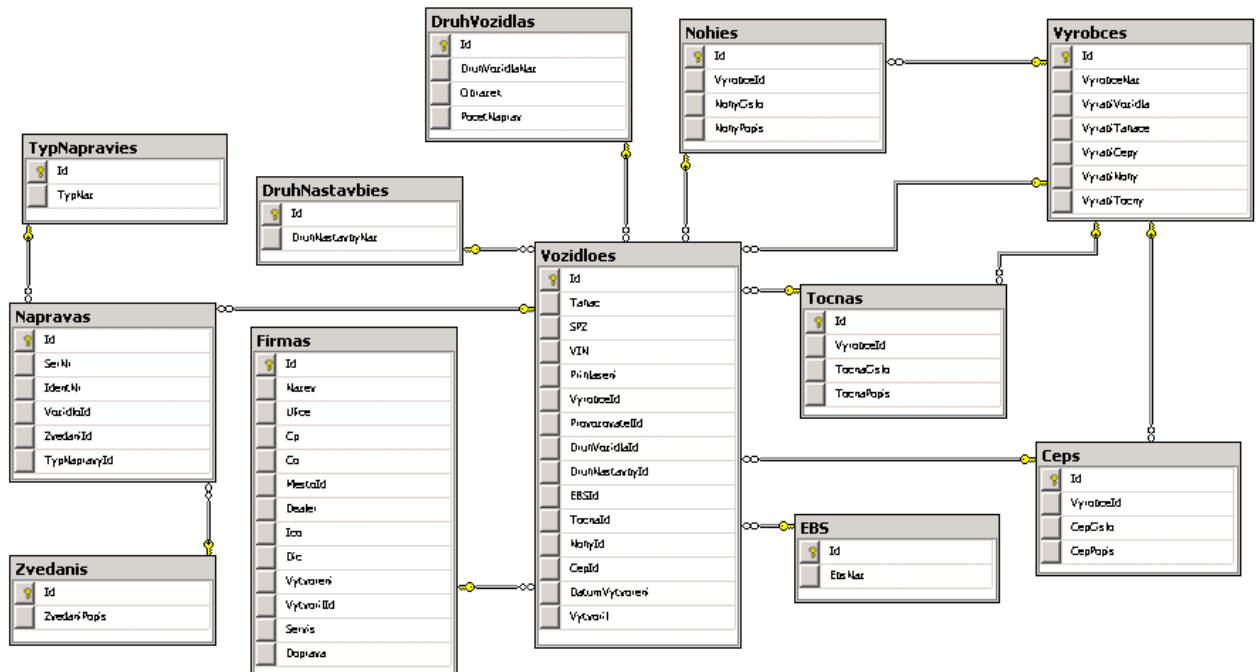
The screenshot displays a web interface for finding a dealer. At the top left, there is a sidebar with a list of services: 'Návěsové točny', 'Odstavné nohy', 'Královské čepy', 'Kuličkové dráhy', 'Ostatní', 'HRADNÍ DÍLY', 'SAF-HOLLAND', 'Jak identifikovat ND', 'Parts On Demand', 'Montážní návody', 'Důležité informace', 'Kde koupit', 'TATNÍ PRODUKTY', 'Systém osvětlení', 'Plastové blatníky a skříňky', 'Gumové zástěrky', 'Systém AntiSpray', 'Tažné zařízení', 'Odstavné nohy a navijáky', 'Příčky a oje', 'Ventily a EBS', 'Ostatní', 'RVIS', 'Reklamace', and 'Školení'. The main content area is titled 'Kde koupit' and features a search form with a dropdown for 'Kraj:' and a text input for 'Firma: SAF'. A 'Hledat' button is next to the input. Below the search form, the results for 'SAF-HOLLAND Czechia spol. s r.o.' are shown, including the address 'Průmyslová 1152, 68601 Uherské Hradiště'. A map of the location is displayed below the text, showing the area around Uherské Hradiště with a red pin marking the dealer's location. The map includes street names like 'Průmyslová', 'Mánesova', 'Svatováclavská', 'Safarikova', 'T. Maršála Malinovského', '28. října', 'Větrná', 'Solná cesta', and 'Pod Vinohrady'. The map also shows a 'Tesco třída Maršála Malinovského' and a 'Mapa' button. The bottom of the map shows 'Data map ©2014 Google' and 'Podmínky použití'.

Obr. 35 Zobrazení servisů

### 6.3 Evidence vozidel

Další částí naší aplikace je evidence vozidel. Pro tuto část budeme potřebovat další tabulky do databáze. Vytvoříme tedy nové třídy **Vozidlo.cs**, **EBS.cs**, **Cep.cs**, **Tocna.cs**, **Vyrobce.cs**, **Nohy.cs**, **DruhVozidla.cs**, **DruhNastavby.cs**, **Naprava.cs**, **Zvedani.cs** a **TypNapravy.cs**. Většina tříd je velmi jednoduchá a slouží pro splnění NF.

Vytvoříme nový kontroler **VozidloController**. Jako třídu modelu zvolíme novou třídu **Vozidlo** a necháme vytvořit pohledy. Přidáme migraci (*add-migrations Vozidlo*) a zaktualizujeme databázi (*update-database*).



Obr. 36 Schéma databázových tabulek pro evidenci vozidel

Protože tvořit evidenci vozidel nepůjde bez registrace, připišeme omezení `[Authorize]` nad kontroler, tzn. pokud zavolá jakoukoliv metodu tohoto kontroleru anonymní uživatel, bude přesměrován na přihlašovací stránku. Tuto direktivu může zrušit napsáním direktivy `[AllowAnonymous]` nad metodu.

### 6.3.1 Seznam vozidel

Metoda **Index** předává pohledu seznam vozidel firmy přihlášeného uživatele.

```
var zamestnanec = db.Zamestnanecs.Where(u => u.UserId == user.Id).Single();
var firma = zamestnanec.FirmaId;
vozidloes = db.Vozidloes.Where(f => f.Provozovatel.Id == firma).Include(v =>
v.Cep).Include(v => v.DruhNastavby).Include(v => v.DruhVozidla).Include(v =>
v.EBS).Include(v => v.Nohy).Include(v => v.Provozovatel).Include(v =>
v.Tocna).Include(v => v.Vyrobcie);
```

Pohled zobrazí odkaz pro vložení nového vozidla, a prochází jednotlivé položky předaného seznamu a vypisuje je to tabulky. Ke každému řádku přidává odkazy pro detail, úpravu a smazání položky.

Pro zjednodušení hledání přidáme filtrovací pole. Filtrovat lze podle registrační značky a podle VIN. Po kliknutí na tlačítko Filtr se zadaná data odešlou stejným způsobem Index, do které jsme přiřadili kód pro zpracování těchto dat.

```
if (!String.IsNullOrEmpty(spz))
    {vozidloes = vozidloes.Where(s => s.SPZ.Contains(spz));}
if (!String.IsNullOrEmpty(vin))
    {vozidloes = vozidloes.Where(v => v.VIN.Contains(vin));}
```

Nyní metoda vrátí seznam omezený filtrem.

Registrační značka	VIN	Výrobce	Tahač?	Nástavba	
test11	testovacivin12349	PANAV	<input type="checkbox"/>	Skříň	<a href="#">Detail</a>   <a href="#">Upravit</a>   <a href="#">Smazat</a>

Obr. 37 Filtrovaný seznam vozidel

### 6.3.2 Podrobnosti o vozidle

Další metodou je **Details**, která zobrazí podrobnosti o vybraném vozidle, které se nevlezly do seznamu. Do pohledu předává jedno dané vozidlo. Dále přes objekt ViewBag předává uživatelské jméno toho, kdo vozidlo zadal do systému a seznam náprav, které jsou přiřazeny vozidlu.

Pohled vypíše všechny parametry vozidla, dále namontované příslušenství a pokud se nejedná o tahač, vypíše nápravy a jejich parametry.

<ul style="list-style-type: none"> <li>rávy a agregáty</li> <li>řivové točny</li> <li>řavné nohy</li> <li>řvské řepy</li> <li>řkové dráhy</li> <li>řtní</li> </ul>	<p><b>Registrační značka</b> test11</p> <p><b>VIN</b> testovacivin12349</p> <p><b>Tahač?</b> <input type="checkbox"/></p> <p><b>Výrobce</b> PANAV</p> <p><b>Nástavba</b> Skříň</p> <p><b>Druh Vozidla</b> 4-nápravový návěš</p> <p><b>Datum přihlášení</b> 07.05.2014</p> <p><b>Výrobce EBS</b> Knorr</p>																				
<p><b>DNÍ DÍLY</b></p> <p><b>OLLAND</b></p> <ul style="list-style-type: none"> <li>identifikovat ND</li> <li>On Demand</li> <li>řtažní návody</li> <li>řžitě informace</li> <li>koupit</li> </ul> <p><b>NÍ PRODUKTY</b></p> <ul style="list-style-type: none"> <li>řm osvětlení</li> <li>řové blatníky a skříňky</li> <li>řové zástěrky</li> <li>řm AntiSpray</li> <li>ř zařízení</li> <li>řavné nohy a navijáky</li> <li>řy a oje</li> <li>řily a EBS</li> <li>řtní</li> </ul>	<table border="1"> <thead> <tr> <th></th> <th>Výrobce</th> <th>Číslo</th> <th>Popis</th> </tr> </thead> <tbody> <tr> <td><b>Královský řep</b></td> <td>JOST</td> <td>KZ1012</td> <td>2,5"</td> </tr> <tr> <td><b>Odstavné nohy</b></td> <td>haacon</td> <td></td> <td>S2000+</td> </tr> </tbody> </table>		Výrobce	Číslo	Popis	<b>Královský řep</b>	JOST	KZ1012	2,5"	<b>Odstavné nohy</b>	haacon		S2000+								
	Výrobce	Číslo	Popis																		
<b>Královský řep</b>	JOST	KZ1012	2,5"																		
<b>Odstavné nohy</b>	haacon		S2000+																		
	<p style="text-align: center;"><b>Nápravy</b></p> <table border="1"> <thead> <tr> <th>Identifikační číslo</th> <th>Sériové číslo</th> <th>Typ nápravy</th> <th>Zvedání</th> </tr> </thead> <tbody> <tr> <td>12345678901</td> <td>123084789</td> <td>B19-22S</td> <td>Bez zvedání</td> </tr> <tr> <td>12345678901</td> <td>111073121</td> <td>B19-22S</td> <td>Nevím</td> </tr> <tr> <td>12345677888</td> <td>13140230121</td> <td>B9-22S</td> <td>Dvoustranné zvedání</td> </tr> <tr> <td>123456677832</td> <td>11111111111</td> <td>B19-22S</td> <td>Jednostranné zvedání</td> </tr> </tbody> </table>	Identifikační číslo	Sériové číslo	Typ nápravy	Zvedání	12345678901	123084789	B19-22S	Bez zvedání	12345678901	111073121	B19-22S	Nevím	12345677888	13140230121	B9-22S	Dvoustranné zvedání	123456677832	11111111111	B19-22S	Jednostranné zvedání
Identifikační číslo	Sériové číslo	Typ nápravy	Zvedání																		
12345678901	123084789	B19-22S	Bez zvedání																		
12345678901	111073121	B19-22S	Nevím																		
12345677888	13140230121	B9-22S	Dvoustranné zvedání																		
123456677832	11111111111	B19-22S	Jednostranné zvedání																		

Obr. 38 Podrobnosti o vozidle

### 6.3.3 Smazání vozidla

Pohled **Delete** zobrazí detail vybrané položky spolu s dotazem na potvrzení smazání, tlačítkem na potvrzení smazání nebo zrušení smazání. Metoda `Delete()` je vlastně úplně stejná jako metoda `Details()`. Předává pohledu vybrané vozidlo a ve `ViewBag` nápravy vozidla a uživatelské jméno toho, kdo vozidlo vytvořil. Pohled je též téměř totožný. Přidává navíc formulář s tlačítkem `Smazat` a odkazem zpět.

Při potvrzení smazání se zavolá metoda **DeleteConfirmed**. Tato metoda projde všechny nápravy vozidla a odebere je ze seznamu náprav vozidla, pak vozidlo odebere ze seznamu vozidel firmy. Ve skutečnosti v databázi nic nesmažeme, už z toho důvodu, že jak vozidlo, tak nápravy mohou být uloženy v některé reklamaci, která by se tak dostala do nekonzistentního stavu.

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Vozidlo vozidlo = db.Vozidloes.Find(id);
    var firmaid = vozidlo.ProvozovatelId;
    Firma firma = db.Firmas.Find(firmaid);
    var naprawy = vozidlo.Naprawy.ToList();
```

```
foreach(var item in napravy)
{
    vozidlo.Napravy.Remove(item);
}
firma.VozidloList.Remove(vozidlo);
db.SaveChanges();
return RedirectToAction("Index");
}
```

### 6.3.4 Vložení nového vozidla

Další metodou tohoto kontroleru je metoda **Create()**. Tato metoda má opět dvě verze, první, která zobrazí formulář, a druhou, která provádí validaci a ukládá data do databáze. První verze předává ve ViewBag všechny rozbalovací seznamy pro atributy, které se vybírají z jiných tabulek.

Pohled, který byl vytvořen ze šablony, vypadá dost nevzhledně, proto si ho upravíme k obrazu svému. I když formulářové pole poskládáme do logických celků, stále je polí poměrně dost a formulář může být nepřehledný. Použijeme tedy JavaScript pro skrytí polí, které nejsou podle aktuálního stavu formuláře potřeba – např. pokud je vozidlo tahač, nezajímají nás nápravy, nezadává se nástavba ani nohy; pokud není vozidlo tahač, zobrazují se pole pro zadání náprav podle druhu vozidla.

### Vložení nového vozidla

Tahač?

Registrační značka	VIN	Výrobce	Datum přihlášení
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
EBS	Točna		
<input type="text"/>	<input type="text"/>		

[Zpět na seznam vozidel](#)

Obr. 39 Vložení nového vozidla - tahač

## Vložení nového vozidla

Tahač?

Registrační značka  VIN  Výrobce  Datum přihlášení

Druh Vozidla  Druh Nástavby

2-nápravový návěš

EBS  Odstavné nohy  Královský čep

Identifikační číslo Sériové číslo Typ nápravy Zvedání nápravy

1.náprava

2.náprava

[Zpět na seznam vozidel](#)

Obr. 40 Vložení nového vozidla - přívěs

Tak jako u vkládání jiných dat musíme provést validaci zadaných dat. Nejprve to uděláme JavaScriptem přímo v prohlížeči stejně jako v předchozích případech. V případě chyby v zadání žádná data serveru neodešleme. V případě, že skript žádnou chybu nenajde, data se odešlou serveru, kde je přijme druhá metoda **Create**.

U výběrů z rozbalovacích menu kontrolujeme, zda uživatel nějakou možnost vybral. U zadávaných atributů kontrolujeme, zda jsou zadané, případně délku řetězce (VIN má vždy 17 znaků), případně skladbu zadaných řetězců (sériové číslo nápravy má pevně danou skladbu).

Pokud zadaná data projdou validací v kontroleru, přidají se do databáze zadané nápravy a vozidlo. K vozidlu se přiřadí nápravy.

```

if (errors.Count == 0)
{
    if (pocetnaprav > 0) { db.Napravas.Add(naprava1); vozidlo.Napravy.Add(naprava1); }
    if (pocetnaprav > 1) { db.Napravas.Add(naprava2); vozidlo.Napravy.Add(naprava2); }
    if (pocetnaprav > 2) { db.Napravas.Add(naprava3); vozidlo.Napravy.Add(naprava3); }
    if (pocetnaprav > 3) { db.Napravas.Add(naprava4); vozidlo.Napravy.Add(naprava4); }
    if (pocetnaprav > 4) { db.Napravas.Add(naprava5); vozidlo.Napravy.Add(naprava5); }
}
db.SaveChanges();
vozidlo.DatumVytvoreni = DateTime.Now;
var userId = db.Users.Where(n => n.UserName == User.Identity.Name).Single().Id;
vozidlo.Vytvoril = userId;

```

```

var zamestnanec = db.Zamestnanecs.Where(i => i.UserId == userId).Single();
var provozovatel = zamestnanec.Firma;
vozidlo.Provozovatel = provozovatel;
db.Vozidloes.Add(vozidlo);
db.SaveChanges();
return RedirectToAction("Index");
}

```

### 6.3.5 Editace vozidla

Poslední metodou kontroleru VozidloController je **Edit()**. Edit je podobná metoda metodě Create(). Jako Create má i Edit dvě verze, kde první zobrazuje formulář a druhá zpracovává zadaná data. Od metody Create se liší pouze tím, že pohled Create do formuláře doplní data vybraného vozidla. Stejně jako u metody Create musíme v metodě Edit provést validaci zadaných dat. Druhá verze metody Edit data do databáze na rozdíl od metody Create nepřidává, ale mění je.

## 6.4 Reklamace

Další částí naší aplikace je část pro evidenci reklamací. Pro tuto část budeme potřebovat ještě několik tabulek v naší databázi. Přidáme proto třídy **Reklamace.cs** a **MontazniPolo-ha.cs**.

```

public class Reklamace
{
    public int Id { get; set; }
    [Display(Name="Číslo reklamace")]
    public string Cislo { get; set; }
    [Display(Name = "Číslo reklamace SAF")]
    public string Me { get; set; }
    [Display(Name="Reklamace ND?")]
    public bool ReklamaceNd { get; set; }
    [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}", ApplyFormatInEditMode = true)]
    [Display(Name="Datum opravy")]
    public DateTime DatumOpravy { get; set; }
    [Display(Name="Servis")]
    public int ObjednavatelId { get; set; }
    public virtual Firma Objednavatel { get; set; }
    [Display(Name="Přívěs/návěs")]
    public int? VozidloId { get; set; }
    public virtual Vozidlo Vozidlo { get; set; }
    [Display(Name="Tahač")]
    public int? TahacId { get; set; }
    public virtual Vozidlo Tahac { get; set; }
    [Range(1,10000000)]
    [Display(Name="Stav km")]
    public int? KmVozidlo { get; set; }
    [Range(1,10000000)]
    [Display(Name="Najeté km s namontovaným ND")]
    public int? KmNd { get; set; }
    [Display(Name="Reklamovaný díl")]
}

```

```

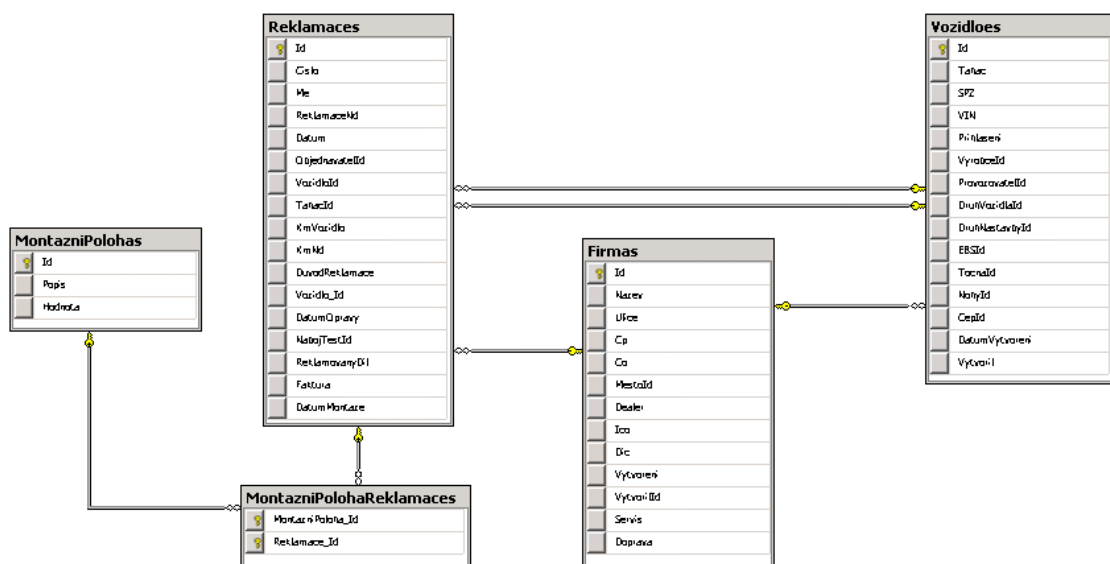
public string ReklamovanyDil { get; set; }
public string Faktura { get; set; }
[DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}", ApplyFormatInEditMode = true)]
[Display(Name="Datum montáže ND")]
public DateTime? DatumMontaze { get; set; }
[Display(Name = "Důvod reklamace - popis závady")]
[DataType(DataType.MultilineText)]
public string DuvodReklamace { get; set; }
[ScaffoldColumn(false)]
[DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}", ApplyFormatInEditMode = true)]
public DateTime Datum { get; set; }
[Display(Name="Vyberte polohu reklamovaného dílu")]
public virtual ICollection<MontazniPoloha> MontazniPolohy { get; set; }
}

public class MontazniPoloha
{
    public MontazniPoloha()
    {
        this.Reklamaces = new HashSet<Reklamace>();
    }
    public int Id { get; set; }
    public string Popis { get; set; }
    public int Hodnota { get; set; }
    public virtual ICollection<Reklamace> Reklamaces { get; set; }
}

```

Vytvoříme kontroler ReklamaceController, jako modelovou třídu použijeme samozřejmě třídu Reklamace, necháme vygenerovat pohledy.

Přidáme novou migraci a zaktualizujeme databázi.



Obr. 41 Schéma databázových tabulek pro evidenci reklamací

Jak vidíme, třída reklamace bude nejsložitější, už jenom protože obsahuje dvě vazby na tabulku vozidel. Je to z toho důvodu, že v reklamaci může zadáný jak tahač, tak přípojné vozidlo.

Protože opět nechceme, aby nám reklamace zadávali a prohlíželi neregistrovaní uživatelé, napíšeme před kontroler direktivu [Authorize].

#### 6.4.1 Seznam reklamací

Metoda **Index()** slouží k výpisu reklamací, ve kterých se nějakým způsobem vyskytuje firma přihlášeného uživatele – je buď provozovatelem vozidla v reklamaci, případně je servisem, který reklamaci zpracovává. Kód bude tedy složitější.

```
var reklamaces1 = reklamaces.Where(f => f.ObjednavatelId == firma);
var reklamaces2 = reklamaces.Where(f => f.Tahac.ProvozovatelId == firma);
var reklamaces3 = reklamaces.Where(f => f.Vozidlo.ProvozovatelId == firma);
var reklamaces = reklamaces1.Concat(reklamaces2).Concat(reklamaces3);
return View(reklamaces.ToList());
```

Pohled tedy dostane seznam reklamací splňující podmínky. Vypíše seznam reklamací do řádků, jejich atributy do sloupců. Jelikož je atributů mnoho, nevejdou se nám do stránky. Všechny atributy ale není potřeba vypsat. Zobrazíme je v detailu. Upravíme tedy kód, aby se zobrazené atributy vešly do stránky, a bylo možné reklamaci identifikovat.

PRODUKTY SAF-HOLLAND		Reklamace					<a href="#">Vložit novou reklamaci</a>
<input type="checkbox"/> Nápravy a agregáty		Číslo reklamace SAF	Číslo reklamace	Registrační značka	Datum	Důvod reklamace - popis závady	
<input type="checkbox"/> Návěsové točny				nezadáno / test2	08.05.2014	afjlhasf	<a href="#">Upravit</a> <a href="#">Detail</a>
<input type="checkbox"/> Odstavné nohy			1	nezadáno / aaaa	08.05.2014	afasdfa	<a href="#">Upravit</a> <a href="#">Detail</a>
<input type="checkbox"/> Královské čepy			afjaúslđjaúsd	nezadáno / aba1	08.05.2014	sdjůa	<a href="#">Upravit</a> <a href="#">Detail</a>
<input type="checkbox"/> Kulčkové dráhy				nezadáno / test2	08.05.2014	afjlhasf	<a href="#">Upravit</a> <a href="#">Detail</a>
<input type="checkbox"/> Ostatní							
NÁHRADNÍ DÍLY SAF-HOLLAND							
<input type="checkbox"/> Jak identifikovat ND							
<input type="checkbox"/> Parts On Demand							
<input type="checkbox"/> Montážní návody							
<input type="checkbox"/> Důležité informace							
<input type="checkbox"/> Kde koupit							

Obr. 42 Výpis reklamací

#### 6.4.2 Podrobnosti reklamace

Další metodou je metoda **Details**, která čeká na vstupu interní číslo (Id) reklamace. Metoda je jednoduchá: podle čísla reklamace ji vyhledá v databázi a předá pohledu, který vypíše všechny parametry.

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Kráiovské čepy
- Kuličkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky
- Gumové zástěrky
- Systém AntiSpray
- Tažné zařízení
- Odstavné nohy a navijáky
- Příčky a oje
- Ventily a EBS

### Podrobnosti

#### Reklamacce

---

**Zadavatel reklamacce**  
abad

**Registrační značka tahače**  
Registrační značka vozidla  
nova

**Číslo reklamacce**  
aneafsf

**Číslo reklamacce SAF**  
ME50010123

**Reklamacce ND?**

**Datum**  
08.05.2014

**Stav km**  
1111

**Najeté km s namontovaným ND**  
**Důvod reklamacce - popis závady**  
ajfdúljajdiaoj

[Upravit](#) | [Zpět](#)

Obr. 43 Podrobnosti o reklamaci – zobrazení šablony

Tento výpis není pro naše potřeby vůbec vhodný. Potřebujeme vypsát i některé údaje o vozidlech, apod. Pohled této metody tedy celý přepracujeme, aby zobrazoval všechny potřebné informace. Aby byl více přehledný zapracujeme všechny informace do tabulky.

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Kráiovské čepy
- Kuličkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky
- Gumové zástěrky
- Systém AntiSpray
- Tažné zařízení
- Odstavné nohy a navijáky
- Příčky a oje
- Ventily a EBS
- Ostatní

**SERVIS**

- Reklamacce
- Školení

### Podrobnosti

#### Reklamacce

---

<b>Číslo reklamacce SAF</b>	ME50010123	<b>Datum</b>	08.05.2014	<b>Číslo reklamacce</b>	aneafsf
<b>Objednavatel (zpravidla servis)</b>		abad afsasfd 68601 Uherské Hradiště		<b>Datum opravy</b>	01.05.2014
<b>Provozovatel vozidla</b>		pepe pp 1 68601 Uherské Hradiště			
<b>Výrobce vozidla</b>	PANAV	<b>Registrační značka vozidla</b>	nova	<b>Číslo podvozku</b>	12345678901234568
<b>Datum 1. přihlášení do evidence vozidel</b>		02.05.2014	<b>Kilometrový výkon celkem</b>		1111
	<b>Identifikační číslo</b>	<b>Sériové číslo</b>	<b>Typ nápravy</b>		
<b>1.náprava</b>	12345678901	111043111	B9-22S		
<b>Druh Vozidla</b>	1-nápravový návés	<b>Montážní polohy</b>	První náprava vlevo První náprava vpravo		
<b>Nástavba</b>	Skříň				
<b>Výrobce EBS</b>	Hallex				
<b>Tažné vozidlo</b>	<b>Výrobce</b>	<b>1.uvedení do provozu</b>			
	EBS	VIN			
<b>Důvod reklamacce - popis závady</b>	ajfdúljajdiaoj				

[Upravit](#) | [Zpět](#)

Obr. 44 Podrobnosti o reklamaci – upravené zobrazení

Nyní máme v pohledu všechny potřebné informace v přijatelném formátu.

### 6.4.3 Vložení nové reklamace

Aby bylo možné zobrazit detail reklamace, musíme nejdřív nějakou reklamaci v aplikaci vytvořit. K tomu samozřejmě slouží metoda **Create()**. Opět má dvě varianty, první nepřijímá žádný argument a zobrazuje formulář pro zadání reklamace. Druhá přijímá všechny zadané hodnoty, které zkontroluje a v případě, že jsou data v pořádku zapíše do databáze.

Metoda **Create()** pro zobrazení formuláře předává v objektu **ViewBag** pohledu všechny potřebné rozbalovací seznamy. Protože je dat, které je potřeba zadat mnoho. Použijeme JavaScript pro skrytí polí, a zobrazujeme je postupně při zadávání dat. Tak zůstane formulář přehledný.

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěšové točny
- Odstavné nohy
- Královské žepy
- Kuličkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky

**Vložit novou reklamaci**

Číslo reklamace  Datum opravy  Reklamační ND?

Návěs/přívěs  Tahač

Vyberte ze seznamu nebo ▼ Vyberte ze seznamu nebo ▼

Stav km

Důvod reklamace - popis závady

Vložit

[Seznam reklamací](#)

Obr. 45 Vložení nové reklamace – výchozí pohled

Pokud zvolíme, že se jedná o reklamaci ND, zobrazí se pole pro zadání čísla ND, čísla faktury, data montáže ND a počtu najetých km s ND. Montážní polohy se zobrazí až po vybrání přípojného vozidla z rozbalovacího seznamu. Pole pro zadání nového vozidla se zobrazí, pouze pokud vybereme volbu *nový* z rozbalovacího seznamu. Atd.

The screenshot shows a web application interface for creating a new complaint. On the left, there is a sidebar with several menu items, each with a radio button:

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Královské čepy
- Kuličkové dráty
- Ostatní

Below these are three sections:

- NÁHRADNÍ DÍLY SAF-HOLLAND**
  - Jak identifikovat ND
  - Parts On Demand
  - Montážní návody
  - Důležité informace
  - Kde koupit
- OSTATNÍ PRODUKTY**
  - Systém osvětlení
  - Plastové blatníky a skříňky
  - Gumové zástěrky
  - Systém AntiSpray
  - Tažné zařízení
  - Odstavné nohy a navijáky
  - Příčky a oje
  - Ventily a EBS
  - Ostatní
- SERVIS**
  - Reklamace
  - Školení

The main form, titled "VLOŽIT NOVOU REKLAMACI", contains the following fields:

- Číslo reklamace:
- Datum opravy:
- Reklamace ND?:
- Návěs/přívěs:
- Tahač:
- Poznávací značka:
- VIN:
- Datum přihlášení:
- Výrobce:
- Dodavatel EBS:
- Točna:
- Provozovatel:
- Servis:
- Stav km:
- Najeté km s namontovaným ND:
- Reklamovaný díl:
- Faktura:
- Datum montáže ND:

At the bottom right of the form is a button labeled "Vložit".

Obr. 46 Vložení nové reklamace po výběru tahače

Při odeslání formuláře tlačítkem *Vložit* se nejprve spustí kontrola skriptem v prohlížeči. Pokud kontrola nenajde chybu data se odešlou na server, kde je přijme druhá metoda `Create()` a provede jejich validaci. Tato metoda může přidávat do databáze nejen reklamace, ale i nové vozidla, nápravy a firmy. Je to naše nejkompexnější a nejsložitější metoda.

#### 6.4.4 Editace a smazání reklamace

Reklamaci již neumožníme editovat, protože zadané data budou předávána na centrálu, která je bude dál zpracovávat. Pouze v administračním režimu umožníme přidat číslo reklamace SAF. Ze stejného důvodu nepovolíme mazání reklamací z databáze.

### 6.5 Zákaznická zóna

Pro vytvoření zákaznické zóny využijeme **AccountController**, který již máme vytvořený, respektive jeho metodu **Index()**.

Tato metoda předá pohledu firmu aktuálně přihlášeného uživatele. Pohled zobrazí informace o firmě, a tlačítka k přesměrování na změnu uživatelských informací, na seznam vozidel a na seznam reklamací.



Obr. 47 Zákaznická zóna

Do dalších verzí se zde plánují další informace určené jednotlivým zákazníkům, např. speciální nabídky, speciální ceníky, apod.

## 6.6 Administrační rozhraní

Pro administrační rozhraní budeme potřebovat nový kontroler, pojmenujme ho **AdminController**. Klikneme pravým tlačítkem na složku Controllers, vybereme Add => Controller. V dalším okně zvolíme MVC5 Controller – Empty. Tím si vytvoříme prázdný kontroler, který má předpřipravenou metodu **Index()**.

Protože nebudeme vytvářet speciální odkaz pro přístup do administrace, přesměrujeme administrátora na toto rozhraní po kliknutí na přístup do zákaznické zóny - nepředpokládáme, že roli administrátora dostane nějaký zákazník, takže nám to v ničem nevadí.

```
if (User.IsInRole("admin"))
{
    return Redirect("~/admin");
}
```

### 6.6.1 Metoda Index

Pohled **Index** může zobrazovat různé informace, které by měli uživatelé s právy administrátora vidět. Zatím bude možnost tisknout prezenční listiny účastníků školení. Předáme tedy pohledu objektem ViewBag seznam školení.

Pohled je jednoduchý, zobrazí několik odkazů na další metody kontroleru a (jak bylo zmíněno) seznam školení s možností tisku prezenčních listů, který se vypíše v cyklu foreach, který prochází seznam školení z ViewBag.

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Královské čepy
- Kulíčkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité Informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky
- Gumové zástěrky
- Systém AntiSpray

## Administrace

Zde můžete editovat všechna data

[Uživatelé](#) [Firmy](#) [Reklamacce](#) [Školení](#) [Vozidla](#)

### Školení - prezenční listy

Datum	Typ školení	
19.05.2014	Řešení reklamací SAF-HOLLAND - ohledání, zpracování, vyřízení	<a href="#">Prezenční list</a>
20.05.2014	Řešení reklamací SAF-HOLLAND - ohledání, zpracování, vyřízení	<a href="#">Prezenční list</a>
16.06.2014	Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND	<a href="#">Prezenční list</a>
17.06.2014	Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND	<a href="#">Prezenční list</a>
20.10.2014	Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND	<a href="#">Prezenční list</a>
21.10.2014	Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND	<a href="#">Prezenční list</a>
10.11.2014	Identifikace, objednávání a řízení náhradních dílů a opravářských sad SAF-HOLLAND	<a href="#">Prezenční list</a>
11.11.2014	Identifikace, objednávání a řízení náhradních dílů a opravářských sad SAF-HOLLAND	<a href="#">Prezenční list</a>

Obr. 48 Administrační rozhraní

## 6.6.2 Tisk prezenčního listu

Pro zobrazení prezenčního listu napíšeme novou metodu **Ucastnici**. Tato metoda má jeden vstupní parametr, a to interní číslo (Id) školení. Najde toto školení v databázi a předá ho pohledu.

Pohled vytvoříme kliknutím pravým tlačítkem do metody a výběrem Add View... z menu. Jako šablonu zvolíme Details, jako třídu Skoleni. Protože tato stránka bude sloužit k tisku, a nechceme na ní mít naše menu, pozadí, atd., odznačíme zatržítka u Use a layout page. Tím se nám vygeneruje kompletní stránka včetně `<!DOCTYPE>` a hlavičky `<head>` `</head>`.

Upravíme vzhled stránky tak, jak chceme, aby prezenční list vypadal.

### Školení 17.06.2014



Téma školení: Údržba a technologické postupy oprav náprav a agregátů SAF-HOLLAND

Č.	Jméno	Příjmení	Datum narození	Firma	Pozice	Podpis
1	Marcel	Machala		SAF-HOLLAND Czechia spol. s r.o.	Logistika a odbyt	
2	Martin	Chmelik		SAF-HOLLAND Czechia spol. s r.o.	Obchodně technický zástupce prodeje náhradních dílů	
3	Michal	Habarta		SAF-HOLLAND Czechia spol. s r.o.	Skladník	
4						

Obr. 49 Prezenční list

### 6.6.3 Rotativa

Tento prezenční list můžeme vyexportovat do pdf. Pro export do pdf lze použít několik nástrojů. Já zvolil **Rotativa**. Je snadná na použití, je zdarma (licence MIT) a lehce se přidá do projektu.

Přepneme se *Package Manager Console* a napíšeme příkaz *Install-Package Rotativa*. Tento příkaz stáhne balíček z internetového úložiště, a nakopíruje všechny potřebné soubory do projektu.

Napíšeme další metodu, kterou nazveme **TiskPrezencka** se stejným parametrem jako metoda *Ucastnici*. Metoda vrací *ActionAsPdf*, který má jako parametr metodu, která se zavolá a převede do pdf. Metoda má další parametry, které určují vzhled výsledného pdf souboru.

```
public ActionResult TiskPrezencka(int Id)
{
    var skol = db.Skolenis.Find(Id);
    string file = "Prezenční list " + skol.Datum;
    return new ActionAsPdf("Ucastnici", new { Id = Id })
    {
        FileName = file,
        PageOrientation = Orientation.Landscape,
        PageSize = Size.A4,
        IsBackgroundDisabled = true
    };
}
```

### 6.6.4 Metoda Uzivatele

Metoda **Uzivatele()** posílá pohledu seznam registrovaných uživatelů. Pohled vytvoříme jako *List* jako třídu zvolíme *Zamestnanec*, protože každý uživatel musí mít záznam v tabulce *Zamestnanec*. Vygenerovaný pohled si upravíme, aby zapadl do našeho zobrazení.

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Královské čepy
- Kulíčkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité informace
- Kde koupit

**SOUBŘÍ PRODUKTY**

#### Uživatelé

Příjmení:  Firma:

Jméno	Příjmení	Telefon Mobil	Email	Firma	
Marcel	Machala	+420 572 540 903 +420 721 838 458	<a href="mailto:marcel.machala@safholland.cz">marcel.machala@safholland.cz</a>	SAF-HOLLAND Czechia spol. s r.o.	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>
Michal	Habarta	+420 572 557 188	<a href="mailto:sklad@safholland.cz">sklad@safholland.cz</a>	SAF-HOLLAND Czechia spol. s r.o.	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>
aa	pepi	00420 572 557 189	<a href="mailto:pepe@pepe.pepe">pepe@pepe.pepe</a>	pepe	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>

Obr. 50 Výpis uživatelů

Pro správu uživatelů si vytvoříme nový kontroler `UzivatelController`, který bude obsahovat metody pro správu uživatelů, tj. `Edit`, `Details` a `Delete`. Vstupem do těchto metod je `Id`, které metoda vyhledá v databázi a zobrazí daný pohled s vybraným záznamem.

`Details` vypíše dostupné informace o uživateli (jeho `username`, jeho přiřazené role).

<ul style="list-style-type: none"> <li><input type="checkbox"/> Kuličkové dráhy</li> <li><input type="checkbox"/> Ostatní</li> </ul>	<p><b>Jméno</b> Marcel</p> <p><b>Příjmení</b> Machala</p> <p><b>Jméno firmy</b> SAF-HOLLAND Czechia spol. s r.o.</p> <p><b>Funkce</b> Logistika a odbyt</p> <p><b>Telefon</b> +420 572 540 903</p> <p><b>Fax</b> +420 572 540 933</p> <p><b>Mobil</b> +420 721 838 458</p> <p><b>Email</b> <a href="mailto:marcel.machala@safholland.cz">marcel.machala@safholland.cz</a></p> <p><b>Datum narození</b> 19.8.1982 0:00:00</p> <p><b>Datum vytvoření</b> 1.1.1900 0:00:00</p> <p><b>Je platný?</b> <input checked="" type="checkbox"/></p> <p><b>Uživatelské jméno</b> marcel</p> <p><b>Role</b> admin</p>
<p><b>NÁHRADNÍ DÍLY</b> <b>SAF-HOLLAND</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Jak identifikovat ND</li> <li><input type="checkbox"/> Parts On Demand</li> <li><input type="checkbox"/> Montážní návody</li> <li><input type="checkbox"/> Důležité informace</li> <li><input type="checkbox"/> Kde koupit</li> </ul> <p><b>OSTATNÍ PRODUKTY</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Systém osvětlení</li> <li><input type="checkbox"/> Plastové blatníky a skříňky</li> <li><input type="checkbox"/> Gumové zástěrky</li> <li><input type="checkbox"/> Systém AntiSpray</li> <li><input type="checkbox"/> Tažné zařízení</li> <li><input type="checkbox"/> Odstavné nohy a navijáky</li> <li><input type="checkbox"/> Příčky a oje</li> <li><input type="checkbox"/> Ventily a EBS</li> <li><input type="checkbox"/> Ostatní</li> </ul> <p><b>SERVIS</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Reklamacce</li> <li><input type="checkbox"/> Školení</li> </ul>	

[Upravit](#) | [Zpět](#)

Obr. 51 Podrobnosti o uživateli

Metoda `Edit()` zobrazí stejný formulář, ale v editovatelné formě. Pro jednotlivé role vytvoří checkboxy, tím se dají uživateli přiřadit role.

<ul style="list-style-type: none"> <li><input type="checkbox"/> Ostatní</li> </ul>	<p><input type="text" value=""/></p>
<p><b>NÁHRADNÍ DÍLY</b> <b>SAF-HOLLAND</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Jak identifikovat ND</li> <li><input type="checkbox"/> Parts On Demand</li> <li><input type="checkbox"/> Montážní návody</li> <li><input type="checkbox"/> Důležité informace</li> <li><input type="checkbox"/> Kde koupit</li> </ul> <p><b>OSTATNÍ PRODUKTY</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Systém osvětlení</li> <li><input type="checkbox"/> Plastové blatníky a skříňky</li> <li><input type="checkbox"/> Gumové zástěrky</li> <li><input type="checkbox"/> Systém AntiSpray</li> <li><input type="checkbox"/> Tažné zařízení</li> <li><input type="checkbox"/> Odstavné nohy a navijáky</li> <li><input type="checkbox"/> Příčky a oje</li> <li><input type="checkbox"/> Ventily a EBS</li> <li><input type="checkbox"/> Ostatní</li> </ul> <p><b>SERVIS</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Reklamacce</li> <li><input type="checkbox"/> Školení</li> </ul>	<p><b>Jméno</b> <input type="text" value="Michal"/></p> <p><b>Příjmení</b> <input type="text" value="Habarta"/></p> <p><b>Funkce</b> <input type="text" value="Skladník"/></p> <p><b>Telefon</b> <input type="text" value="+420 572 557 188"/></p> <p><b>Fax</b> <input type="text" value="+420 572 540 933"/></p> <p><b>Mobil</b> <input type="text" value=""/></p> <p><b>Email</b> <input type="text" value="sklad@safholland.cz"/></p> <p><b>Firma</b> <input type="text" value="SAF-HOLLAND Czechia spol. s r.o."/></p> <p><b>Datum narození</b> <input type="text" value="9.12.1982 0:00:00"/></p> <p><b>Datum vytvoření</b> <input type="text" value="1.1.1900 0:00:00"/></p> <p><b>Je platný?</b> <input checked="" type="checkbox"/></p> <p><b>Role</b> <input type="checkbox"/> admin <input type="checkbox"/> PowerUser</p>

Obr. 52 Editace uživatele

Po odeslání formuláře probíhá validace JavaScriptem, a posléze druhá validace na serveru, tak jako u všech metod, které přijímají data od uživatele.

Metoda Delete, po potvrzení požadavku na smazání uživatelského účtu, označí zaměstnance svázaného s účtem, jako neplatného, a vymaže uživatelský účet z databáze.

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Zamestnanec zamestnanec = db.Zamestnanecs.Find(id);
    zamestnanec.Platny = false;
    var user = db.Users.Where(i => i.Id == zamestnanec.UserId).Single();
    db.Users.Remove(user);
    db.SaveChanges();
    return RedirectToAction("Uzivatele", "Admin");
}
```

### 6.6.5 Metoda Vozidla

Metoda **Vozidla()** předává pohledu seznam vozidel. Pohled vytvoříme jako List a jako třídu zvolíme Vozidlo. Pohled upravíme, aby se nám zobrazené sloupce vešly do stránky. Přidáme odkazy k jejich editaci, smazání a detailu. Odkazy už nasměrujeme na kontroler VozidloController a jeho metody.

Protože v tomto pohledu budeme zobrazovat úplně všechna vozidla, přidáme filtrování. Do pohledu přidáme formulář, který bude obsahovat pole pro zadání registrační značky, VIN, provozovatele a rozbalovací seznamy s výrobcí a nastávkami. Do metody přidáme kód, který přijme data zadaná do polí formuláře, a na jejich základě omezí vrácený seznam vozidel.

```
if(!string.IsNullOrEmpty(spz))
{
    vozidla = vozidla.Where(s => s.SPZ.Contains(spz));
}
if (!string.IsNullOrEmpty(vin))
{
    vozidla = vozidla.Where(s => s.VIN.Contains(vin));
}
if (!string.IsNullOrEmpty(vyrobce))
{
    vozidla = vozidla.Where(s => s.Vyrobce.VyrobceNaz == vyrobce);
}
if (!string.IsNullOrEmpty(nastavba))
{
    vozidla = vozidla.Where(s => s.DruhNastavby.DruhNastavbyNaz == nastavba);
}
if (!string.IsNullOrEmpty(provozovatel))
{
    vozidla = vozidla.Where(s => s.Provozovatel.Nazev.Contains(provozovatel));
}
```

```

}
vozidla = vozidla.OrderBy(s => s.SPZ);
return View(vozidla);

```

**PRODUKTY SAF-HOLLAND**

- Nápravy a agregáty
- Návěsové točny
- Odstavné nohy
- Královské čepy
- Kuličkové dráhy
- Ostatní

**NÁHRADNÍ DÍLY SAF-HOLLAND**

- Jak identifikovat ND
- Parts On Demand
- Montážní návody
- Důležité informace
- Kde koupit

**OSTATNÍ PRODUKTY**

- Systém osvětlení
- Plastové blatníky a skříňky
- Gumové zástěrky
- Systém AntiSpray
- Tažné zařízení
- Odstavné nohy a navijáky
- Příčky a oje
- Ventily a EBS
- Ostatní

### Vozidla

Registrační značka:  VIN:

Výrobce:  Nastavba:  Provozovatel:  Hledat

[Vložit nové](#)

Registrační značka	VIN	Výrobce	Nastavba	Provozovatel	
aaaa	12345678901234567	Schmitz	Sklopěč	pepe	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>
aba1	12345678901234567	SCANIA		pepe	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>
nova	12345678901234568	PANAV	Skříň	pepe	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>
test11	testovacivin12349	PANAV	Skříň	SAF-HOLLAND Czechia spol. s r.o.	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>
test2	testovacivin12345	SCANIA		SAF-HOLLAND Czechia spol. s r.o.	<a href="#">Upravit</a> <a href="#">Podrobnosti</a> <a href="#">Smazat</a>

Obr. 53 Filtrovaný výpis vozidel v administračním rozhraní

### 6.6.6 Metody Skoleni, Reklamace a Firmy

Tyto metody jsou velmi podobné metodě předchozí. Rozdíl je pouze v třídě, kterou použijeme. Podle toho se pak samozřejmě změní pole, podle kterých seznamy filtrujeme. Odkazy vždy směřujeme na kontroler dané třídy.

## 7 BEZPEČNOST

### 7.1 SQL Injection

Naše aplikace není zranitelná tímto útokem. Nikde v naší aplikaci nepoužíváme přímý zápis SQL. Všechny dotazy do databáze řešíme přes *Entity Framework*.

### 7.2 Krádež identity

Proti tomuto útoku chráníme naši aplikaci tím, že SID předáváme v cookie s vlastností *HttpOnly*. SID je navíc dlouhý 450 znaků, čímž je chráněn před brutte-force útokem, tj. vyzkoušením všech možností.

### 7.3 Cross-Site Scripting (XSS)

Veškeré vstupy MVC Framework testuje a v případě, že existuje podezření na XSS (vstup obsahuje tagy jazyka HTML) tento vstup nepřijme a vrací výjimku *HttpRequestValidationException*.

My můžeme tuto výjimku odchytnout a přiměřeně na ni reagovat. Je to lepší než zobrazit standardní hlášení. Otevřeme si soubor *Global.asax.cs* v našem projektu. Přidáme do něj novou metodu, kterou pojmenujeme **Application\_Error**.

```
protected void Application_Error(object sender, EventArgs e)
{
    var context = HttpContext.Current;
    var exception = context.Server.GetLastError();
    if (exception is HttpRequestValidationException)
    {
        context.Server.ClearError();
        Response.Clear();
        Response.StatusCode = 200;
        Response.Redirect("../Error500");
        Response.End();
        return;
    }
}
```

Tato metoda odchytné výjimky na serveru. Pokud je výjimka *HttpRequestValidationException*, zrušíme výjimku, pošleme prohlížeči odpověď 200 (OK) a přesměrujeme na námi vytvořenou stránku, které zobrazí naše hlášení a provede další akce (přesměrování zpět na formulář).

The screenshot shows the SAF-HOLLAND website interface. On the left, there is a menu titled 'PRODUKTY SAF-HOLLAND' with several items: 'Nápravy a agregáty', 'Návěsové točny', 'Odstavné nohy', 'Kráčkové čepy', and 'Kuličkové dráhy'. The 'Kuličkové dráhy' item is highlighted with a yellow background. On the right, there is a dark grey banner with the text: 'Bohužel při provádění Vašeho požadavku došlo k chybě, kterou nedokážeme identifikovat. Nyní Vás vrátíme zpět. Zkontrolujte Vaše zadání a zkuste to znovu.' Above the banner, there are logos for SAF, holland, and NEWAY, along with the text 'Dnes je 17.05. 2014 a svátek má Aneta'.

Obr. 54 Chybové hlášení

Pokud někdy chceme v dotazu přijímat i HTML tagy, musíme nad metodu, která má tyto tagy přijímat napíšeme direktivu `[ValidateInput(false)]`. Tím v této metodě kontrolu vypneme (samozřejmě můžeme direktivu napsat před kontroler – pak platí pro všechny metody kontroleru). Tím se ale naše aplikace stane potenciálním cílem XSS. Microsoft vydal knihovnu, která nám pomůže naši aplikaci ochránit. Tuto knihovnu můžeme nainstalovat z konzoly **Package Manager Console** příkazem *Install-Package AntiXSS*. Tato knihovna obsahuje třídu **Sanitizer**, jejíž metoda **GetSafeHtmlFragment** vymaže z řetězce potenciálně nebezpečný kód.

My v naší aplikaci žádný kód v HTML přijímat nebude, proto nemusíme tuto knihovnu instalovat.

## 7.4 Cross-site Request Forgery (CSRF)

Proti tomuto útoku se v naší aplikaci bránit musíme. Do každého formuláře, který odesílá data na server, přidáme *AntiForgeryToken* příkazem `@Html.AntiForgeryToken()`. Po při zobrazení stránky v prohlížeči je na jeho místě skryté pole, které obsahuje jednorázový unikátní řetězec.

Před každou metodu, která přijímá data z tohoto formuláře, přidáme direktivu `[ValidateAntiForgeryToken]`. Metoda potom, dříve než přijme data, zkontroluje, zda souhlasí tento řetězec. Pokud nesouhlasí nebo úplně chybí, aplikace vyhodí výjimku *HttpAntiForgeryException*. Tuto výjimku odchytíme a zareagujeme v naší metodě **Application\_Error** v souboru *Global.asax.cs*.

## ZÁVĚR

Nedávno firma SAF-HOLLAND Czechia dostala z mateřské firmy výsledky průzkumu, ze kterého vyplynulo, že nejméně spokojení jsou zákazníci s řešením reklamací, dále s možností identifikace náhradních dílů na své vozidla. Vlastním šetřením firmy SAF-HOLLAND Czechia bylo zjištěno, že nespokojenost z důvodu (ne)možnosti identifikace náhradních dílů vyplývá spíše z nevědomosti zákazníků. Bohužel není možné z údajů v technickém průkazu vozidla identifikovat náhradní díly z důvodů, které byly řečeny v úvodu této práce. Jako částečné řešení tohoto problému slouží první část vytvořené aplikace.

Problém týkající se reklamací je hlubší, zejména proto, že firma řeší reklamace i na produkty, které vlastně neprodala a nemá o nich informace. Zákazníci proto musí dodat mnoho informací, na co nejsou z reklamací jiných dílů zvyklí, a z toho pramení první nespokojenost. Toto nejsme schopni nijak ovlivnit. Druhá část nespokojenosti je z neinformovanosti a délky trvání reklamačního řízení. Tuto část by měla ovlivnit naše aplikace. Aplikace může uživatele informovat o průběhu reklamačního řízení. Délku reklamačního řízení ovlivní kontrola zadaných informací. Když budou zadány správně online, může se reklamace ihned zpracovat.

Třetí část aplikace – školení je takovou „třešničkou“. Za organizovaná školení získává firma SAF-HOLLAND Czechia od svých zákazníků nespočet pochval. Malým rozšířením databáze o dvě tabulky, získáváme online aplikaci, která usnadní administraci školení a zvýší komfort zákazníků.

Celkem má aplikace ušetřit čas jak zaměstnancům, tak i zákazníkům firmy SAF-HOLLAND Czechia a zvýšit spokojenost zákazníků se službami.

Práce samotná seznamuje s principem vytváření jednoduchých webových aplikací v jazyce ASP.NET třívrstvou architekturou MVC, a krok za krokem ukazuje tvorbu jednotlivých částí aplikace.

Jako důležitou se nakonec ukázala volba verze vývojového nástroje, kdy je potřeba, pro bezproblémové nasazení, použít stejné verze vývojových nástrojů a aplikačních serverů. V práci bylo použito VS Express 2013, které obsahuje SQL Server Express 2012 a aplikace má běžet na SQL Server 2008. Nepodařilo se mi dostat lokální testovací data na aplikační

server z důvodu nekompatibility obou verzí. Pro testování musela být data do aplikačního serveru vložena ručně.

**SEZNAM POUŽITÉ LITERATURY**

- [1] RIORDAN, R. M. *Vytváříme relační databázové aplikace*. Praha : Computer Press, 2000, 280 s. ISBN 80-7226-360-9.
- [2] CONNOLLY, T., BEGG, C., HOLOWCZAK, R. *Databáze - Profesionální průvodce tvorbou efektivních databází*. Computer Press, 2009, ISBN: 978-80-251-2328-7.
- [3] ESPOSITO, Dino. *ASP.NET a ADO.NET: tvorba dynamických webových stránek*. 1. vyd. Praha: Grada, 2003, 352 s. ISBN 802-47-0474-9.
- [4] PROSISE, Jeff. *Programování v Microsoft .NET: webové aplikace v .Net Framework, C# a ASP.NET*. Vyd. 1. Brno: Computer Press, 2003, xxi, 712 s. ISBN 807-22-6879-1.
- [5] MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. *ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně*. Vyd. 1. Brno: Zoner Press, 2011, 2 sv. (880, 700 s.). ISBN 978-80-7413-131-81.
- [6] LACKO, Ľuboslav. *1001 tipů a triků pro SQL*. Vyd. 1. Brno: Computer Press, 2011, 416 s. ISBN 978-80-251-3010-0
- [7] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.
- [8] MACHALA, Marcel. *Studijní materiál z matematiky ve formátu HTML*. Zlín, 2012. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Ostravský Jan, RNDr. CSc.
- [9] MICROSOFT. *The Official Microsoft ASP.NET Site: MVC* [online]. © 2014 [cit. 2014-05-08]. Dostupné z: [www.asp.net/MVC](http://www.asp.net/MVC)
- [10] THE JQUERY FOUNDATION. *JQuery* [online]. 2014 [cit. 2014-05-08]. Dostupné z: [www.jquery.com](http://www.jquery.com)
- [11] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.

- [12] HOGAN, Brian P. *HTML5 a CSS3: výukový kurz webového vývojáře*. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-80-251-3576-1.
- [13] *Devbook.cz* [online]. © 2014 [cit. 2014-05-08]. Dostupné z: <http://www.devbook.cz/tvorba-webu-v-asp-dot-net-tutorialy-zdrojove-kody>
- [14] REFSNES DATA. *W3schools.com: ASP.NET MVC* [online]. © 1999-2014 [cit. 2014-05-08]. Dostupné z: [http://www.w3schools.com/aspnet/mvc\\_intro.asp](http://www.w3schools.com/aspnet/mvc_intro.asp)
- [15] BOZIO, Giorgio. Rotativa. GITHUB, Inc. *GitHub* [online]. © 2014 [cit. 2014-05-18]. Dostupné z: <https://github.com/webgio/Rotativa>
- [16] Pokročilé techniky XSS. KÜMMEL, Roman. SOOM.cz [online]. Dostupné z: <http://www.soom.cz/clanky/485--Pokrocile-techniky-XSS>
- [17] Cross Site Request Forgery. KÜMMEL, Roman. SOOM.cz [online]. Dostupné z: <http://www.soom.cz/clanky/484--Cross-Site-Request-Forgery>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

API	Application Programming Interface
ASP	Active Server Page
CGI	Common Gateway Interface – protokol pro propojení aplikací se serverem
CSS	Cascading Style Sheets – kaskádové styly
CSRF	Cross-Site Request Forgery – jedna z metod útoku do web. aplikací
DBMS	DataBase Management System = SŘBD
DDL	Data Definition Language – jazyk pro definici dat – množina příkazů jazyka SQL
DML	Data Manipulation Language – jazyk pro manipulaci s daty – množina příkazů jazyka SQL
ERP	Enterprise resource planning – informační systém firmy, který spojuje a zastřešuje rozdílné činnosti ve firmě, např. logistiku, prodej, účetnictví, mzdy, apod.
GPL	General Public Licence – všeobecná veřejná licence
HTML	HyperText Markup Language – hypertextový značkovací jazyk
IDE	Integrated Development Environment – vývojové prostředí
JSP	JavaServer Pages – jazyk pro psaní dynamických HTML
MVC	Model-View-Controller
ND	Náhradní díl
NF	Normální forma
PHP	PHP: Hypertext Preprocessor, původně Personal Home Page
SQL	Structured Query Language – strukturovaný dotazovací jazyk
SŘBD	Systém Řízení Báze Dat
URL	Uniform Resource Locator
VS	Visual Studio – IDE od firmy Microsoft

W3C            World Wide Web Consortium

WYSIWYG    What you see is what you get – co vidíš, to dostaneš

XSS            Cross-Site Scripting

**SEZNAM OBRÁZKŮ**

Obr. 1 Logo PHP.....	22
Obr. 2 Logo MS .NET MVC 5 .....	23
Obr. 3 Logo MS SQL Serveru .....	25
Obr. 4 Logo MySQL .....	25
Obr. 5 Oficiální webové stránky .....	42
Obr. 6 Schéma tabulek databáze pro funkční menu .....	50
Obr. 7 První funkční formulář naší aplikace.....	50
Obr. 8 Formulář pro vkládání článků.....	51
Obr. 9 Formulář pro vkládání článků do platných sekcí.....	53
Obr. 10 Automaticky vygenerovaný formulář pro smazání článku .....	54
Obr. 11 Formulář pro vymazání článku .....	55
Obr. 12 Výpis seznamu článků .....	56
Obr. 13 Detail článku.....	56
Obr. 14 Chybějící pohled.....	58
Obr. 15 Pohled Index kontroleru ProduktyController .....	60
Obr. 16 Pohled Index kontroleru CepyController.....	60
Obr. 17 Pohled Index kontroleru SkoleniController.....	64
Obr. 18 Chybové hlášení po změně ve třídě Skoleni.....	67
Obr. 19 Package Manager Console.....	67
Obr. 20 Migrace .....	67
Obr. 21 Aktualizace databáze .....	68
Obr. 22 Schéma databázových tabulek pro evidenci školení.....	68
Obr. 23 Přihlášení na školení.....	69
Obr. 24 Vracený formulář po validaci na serveru.....	71
Obr. 25 Registrace nového uživatele .....	73
Obr. 26 Validace na straně klienta - JavaScript.....	74
Obr. 27 Tabulka AspNetUsers.....	74
Obr. 28 Validace přihlašovacího formuláře.....	75
Obr. 29 Nepřihlášený uživatel .....	76
Obr. 30 Přihlášený uživatel.....	76
Obr. 31 Přihlášení kolegy na školení .....	77

Obr. 32 Kolega byl přihlášen .....	78
Obr. 33 Odhlášení se ze školení.....	78
Obr. 34 Zobrazení mapy .....	80
Obr. 35 Zobrazení servisů.....	81
Obr. 36 Schéma databázových tabulek pro evidenci vozidel .....	82
Obr. 37 Filtrovaný seznam vozidel.....	83
Obr. 38 Podrobnosti o vozidle .....	84
Obr. 39 Vložení nového vozidla - tahač .....	85
Obr. 40 Vložení nového vozidla - přívěs.....	86
Obr. 41 Schéma databázových tabulek pro evidenci reklamací .....	88
Obr. 42 Výpis reklamací .....	89
Obr. 43 Podrobnosti o reklamaci – zobrazení šablony .....	90
Obr. 44 Podrobnosti o reklamaci – upravené zobrazení .....	90
Obr. 45 Vložení nové reklamace – výchozí pohled .....	91
Obr. 46 Vložení nové reklamace po výběru tahače .....	92
Obr. 47 Zákaznická zóna .....	93
Obr. 48 Administrační rozhraní .....	94
Obr. 49 Prezenční list.....	94
Obr. 50 Výpis uživatelů .....	95
Obr. 51 Podrobnosti o uživateli .....	96
Obr. 52 Editace uživatele.....	96
Obr. 53 Filtrovaný výpis vozidel v administračním rozhraní .....	98
Obr. 54 Chybové hlášení.....	100

## SEZNAM PŘÍLOH

- A. CD-ROM s elektronickou verzí DP a zdrojovými kódy aplikace