

Autor **Petr Horák**

Obor **Informační technologie**

Vedoucí práce **Ing. Tomáš Kazda, DiS.**

Školní rok **2020/2021**

Střední průmyslová škola strojní   
a elektrotechnická a Vyšší odborná škola, Liberec 1, Masarykova 3

Revize aplikace Rentals

Maturitní práce

Anotace (Resumé)

Tato maturitní práce se zabývá odstraněním chyb a úpravami webové aplikace Rentals používané na SPŠSE a VOŠ Liberec. Aplikace slouží k evidenci výpůjček vybavení ze školního ateliéru. Výchozí verzí této aplikace byla aplikace Rentals vytvořená v rámci maturitní práce bývalého studenta SPŠSE a VOŠ Liberec Marka Honce ve školním roce 2018/19. Při provozu původní aplikace se však vyskytly chyby narušující její chod a funkčnost. Proto vznikla tato práce, která přináší řešení těchto chyb a zároveň řeší a vylepšuje další nedostatky původní verze.

Summary

This work focuses on elimination of errors and modifications of the Rentals web application used at SPŠSE and VOŠ Liberec. The application is used to record equipment borrowings from the school atelier. The initial version of this application was the Rentals application created within the graduation work of the former student of SPŠSE and VOŠ Liberec Marek Honc in the school year 2018/19. During the operation of the original application occurred errors that disrupted its operation and functionality. Therefore, this work was created. It provides a solution to these errors and at the same time solves and improves other shortcomings of the original version.

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní/ročníkovou práci vypracoval(a) sám(a) a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne

Petr Horák

Obsah

[Úvod 1](#_Toc64387018)

[1 Nasazení ve vývojovém prostředí 2](#_Toc64387019)

[2 Přenos databáze do testovacího prostředí 3](#_Toc64387020)

[3 Analýza zdrojových kódů 5](#_Toc64387021)

[4 Analýza a dokumentace chyb 6](#_Toc64387022)

[5 Oprava chyb a úprava stávajících řešení 7](#_Toc64387023)

[Závěr 8](#_Toc64387024)

[Seznam obrázků 9](#_Toc64387025)

[Použitá literatura 10](#_Toc64387026)

[A. Seznam přiložených souborů 1](#_Toc64387027)

[B. Další příloha 2](#_Toc64387028)

Úvod

Téma této práce jsem si zvolil, neboť splňuje vše podstatné, co jsem od dlouhodobé maturitní práce očekával. Mým cílem bylo, aby má práce byla prakticky využitelná, a aby čas strávený s její tvorbou byl přínosný jak pro mě, tak alespoň částečně pro mé okolí. Proto, když mi bylo nabídnuto pracovat na úpravě školního výpůjčkového systému, dlouho jsem neváhal.

Webová aplikace Rentals byla vytvořena pomocí programovacího jazyka C# a frameworku .NET Core. Aplikace je v provozu od roku 2019.

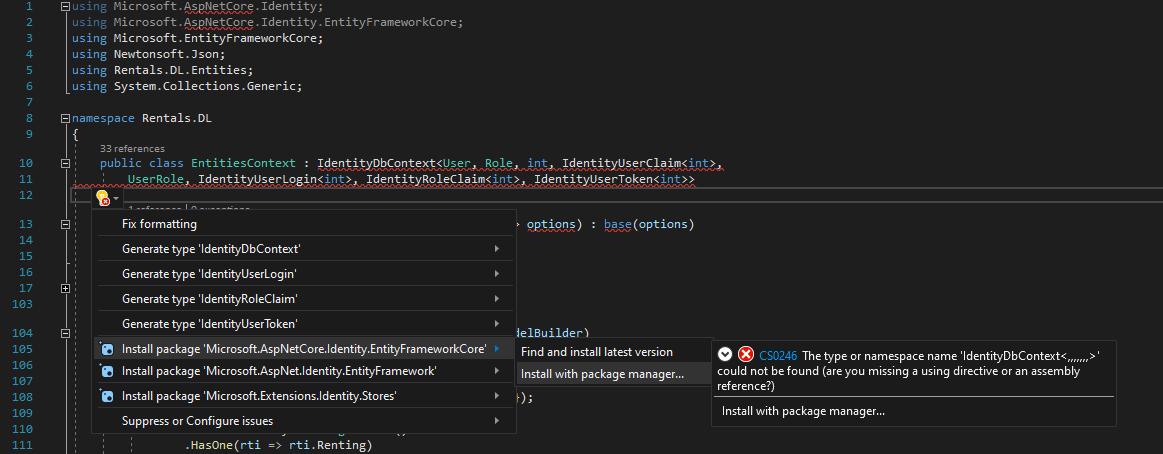
V tomto dokumentu popisuji postup, jakým jsem řešil úpravy původní verze aplikace, tak, aby nová verze byla plně funkční, méně problémová a přehlednější pro její uživatele.

1. Nasazení ve vývojovém prostředí

Jakožto vývojové prostředí jsem používal počítačový program Microsoft Visual Studio 2019, se kterým mám za dobu studia nemalé zkušenosti. Použití kombinace Visual Studia, C# a .NET Core se při debugování a práci s kódem ukázalo jako výhodné.

Výchozím stavem, ze kterého jsem při nasazení aplikace do vývojového prostředí vycházel byl repozitář na platformě GitHub obsahující zdrojové kódy původní aplikace Rentals. Data z repozitáře jsem pomocí funkce *Clone* importoval do Visual Studia.

Jak je již zmíněno výše, Visual Studio zprovoznění aplikace značně ulehčuje například svými chybovými hlášeními, které buď přímo odkazují na problém bránící chodu aplikace, nebo poskytují dobrý přehled nad tím, co je špatně. Postupoval jsem tedy podle tohoto seznamu chyb. Spuštění aplikace bránila absence NuGet balíčku *Microsoft.AspNetCore.Identity.EntityFrameworkCore.* Tento balíček do projektu Rentals.DL imlementuje ASP.NET Core Identity, což je API (Application Programming Interface), které umožňuje do aplikace zavést přihlašování jejich uživatelů uživatelskými účty. Z důvodu kompatibility je použita verze ASP.NET Core Identity 2.1.3.

Po přidání Identity balíčku již nebyly vývojovým prostředím hlášeny žádné chyby a bylo tak možné řešení sestavit.

Obrázek Příklad generovaného návrhu řešení vývojovým prostředím MS Visual Studio

1. Přenos databáze do testovacího prostředí

Dalším krokem po tom, když bylo řešení možné bez chyb sestavit, bylo připojení databáze.

Aplikace používá MSSQL (Microsoft SQL) databázi. Při přenosu databáze do testovacího prostředí jsem nejprve pomocí *SQL Server Object Explorer* panelu v MS Visual Studio vytvořil na lokálním serveru novou databázi. Z vlastností této nově vytvořené databáze jsem zkopíroval parametr *connection string*, který bylo třeba vložit do souboru *appsettings.json*, pro konfiguraci připojení projektu k databázi. Následně jsem použil příkaz *Update-Database*, který inicializuje nově vytvořenou databázi dle konfigurace obsažené v souboru s kontextem databáze (konkrétně *EntitiesContext.cs*).

* 1. Vložení dat do lokální databáze

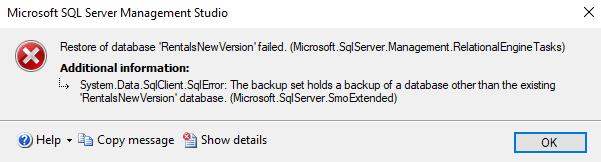
K co nejpřesnějšímu napodobení podmínek verze aplikace provozované na webu s verzí v testovacím prostředí bylo vhodné použít data přímo z aktivní databáze, kterou používá přímo webová aplikace.

* + 1. Vložení dat do databáze pomocí souboru formátu *.bak*

Abych naplnil nově nakonfigurovanou databázi daty, na kterých bych testoval funkčnost aplikace, měl jsem k dispozici soubor s daty z originální databáze ve formátu .*bak*. Tento formát souborů se používá k zálohování souborů. (1)

K přesunutí dat ze souboru *.bak* do databáze bylo třeba použít program *Microsoft SQL Server Management Studio*, který slouží ke správě Microsoft SQL Serveru a Microsoft SQL databází. Oproti *SQL Server Object Explorer* panelu v MS Visual Studio nabízí rozšířené funkce, mezi které patří i funkce *Restore*, která přijímá soubory formátu *.bak*. Tato funkce slouží k obnovení databáze z vybraného zdroje dat. V tomto případě byl zdrojem dat onen soubor formátu *.bak*. Po použití funkce *Restore* na nově vytvořenou databázi se však vyskytlo několik problémů.

Prvním problémem byly rozdílné verze serverů. SQL Server, na kterém byla v provozu databáze, ze které byla pořízena záloha v podobě zmiňovaného soubor formátu *.bak,* měl jinou verzi než ten lokální, na kterém byla vytvořena nová databáze. Po změně verze lokálního SQL serveru nastal další poněkud zásadnější problém.

Další chybová hláška programu *Microsoft SQL Server Management Studio* upozorňovala na skutečnost, že zálohu jedné databáze nelze vložit do databáze jiné, i když se svojí strukturou a vlastnostmi shodují.

Obrázek Chybová hláška programu Microsoft SQL Server Management Studio

* + 1. Vložení dat do databáze pomocí SQL příkazů

Chybové hlášení (viz Obrázek 2) naznačovalo, že dosavadní postup s využitím funkce *Restore* a vložením souboru formátu *.bak* úspěšné nebude. Data z původní databáze jsem proto dostal k dispozici v souborech ve formátu *.sql*. Tyto soubory obsahovali SQL příkazy jak pro vytvoření struktury databáze, tak pro naplnění databáze daty. Jelikož databáze, kterou jsem zkoušel naplnit daty s použitím funkce *Restore* a souboru *.bak*, už strukturu tabulek měla díky provedenému příkazu *Update-Database*, vytvořil jsem databázi novou, kterou jsem nakonfiguroval SQL příkazy ze souboru formátu .sql a dalším souborem .sql obsahujícím INSERT příkazy jsem do tabulek databáze vložil data.

Bylo možné použít první verzi databáze a zkusit ji pouze naplnit daty použitím INSERT příkazů, z důvodu zachování integrity a eliminaci potencionálních komplikací jsem se ale – jak je již zmíněno výše – rozhodl vytvořit databázi novou. Toto řešení se ukázalo jako úspěšné a po proběhnutí všech SQL příkazů byla databáze funkční a naplněna daty. Aplikace se již při spuštění z vývojového prostředí maximálně podobala jak vzhledem, tak funkčností verzi, jež byla v provozu na webu.

Další možností, jak naplnit lokální databázi daty, by bylo „seedování“ vlastních dat. Tento způsob jsem ale zavrhl kvůli časové náročnosti tvorby rádoby relevantních dat.

1. Analýza zdrojových kódů

Zdrojové kódy aplikace jsou psány programovacím jazykem C#. C# je jedním z nejrozšířenějších vysokoúrovňových programovacích jazyků. Používá se k objektovému programování. Byl vyvinut firmou *Microsoft*. (2)

Dále je zde využito frameworku *.NET Core*, který usnadňuje tvorbu aplikací v jazyce C#.

Struktura souborů se zdrojovými kódy aplikace je tvořena třemi projekty, které jsou popsány v podkapitolách níže.

* 1. Projekt *Rentals.Common*

Jedná se o nejméně rozsáhlou část aplikace obsahující například definici základních výčtových typů, které jsou použity v dalším projektu *Rentals.DL* obstarávajícím databázi aplikace. Dle těchto výčtových typů, které se definují klíčovým slovem *enum*, se v aplikaci stanovují například stavy výpůjček a role uživatelů.

V další části také obsahuje třídy, jejichž metody rozšiřují funkčnost základní datových typů používaných v dalších projektech aplikace.

* 1. Projekt *Rentals.DL*

Tento již poněkud rozsáhlejší projekt obsahuje všechny zdrojové kódy týkající se databáze. Využívá se zde struktury kódu, kdy mimo to, že tento projekt obsahuje třídy definující entity s jejich parametry, tak obsahuje také třídy a rozhraní definující metody pro každou z entit v databázi. Využití tohoto postupu v praxi znamená, že všechny dotazy na získání dat z databáze zůstávají v tomto projektu a v projektu *Rentals.WEB* obsahujícím logiku aplikace se pouze volají metody z *Rentals.DL*, jež tyto dotazy zahrnují.

* + 1. Struktura projektu

Pomyslným základem projektu je adresář tříd *Entities*, jež obsahují předpis databázové tabulky každé z entit. Vlastnosti těchto tříd předznamenávají sloupce tabulky. Jednotlivé vlastnosti pak lze označit validátory zvanými *Data Annotations*, které označují např. vlastnost, jež má sloužit jako identifikátor prvku. Jsou zde i třídy, které definují tzv. spojovací tabulky M:N vazby.

V adresáři s názvem *BussinessLogic* jsou třídy odpovídající třídám v adresáři *Entities*, které jsou o onu „Busssiness logiku“ rozšířeny, tzn. obsahují metody, které určují, jak data v daný tabulkách vytvářet a měnit.

Další částí projektu je adresář *Repositories*, která obsahuje třídy pro tabulky, ze kterých se pomocí metod obsažených v těchto třídách získávají data. Tím, že jsou všechny metody pro získávání dat jedné entity v jedné třídě v jednom souboru výrazně usnadňuje orientaci v souborech aplikace.

Každá třída v adresáři *Repositories* je závislá na odpovídajícím rozhraní nacházejícím se v adresáři *Interfaces*. Každá třída je totiž závislá na svém rozhraní, které osahuje deklarace metod obsažených v konkrétní třídě. Přes rozhraní třídy se také k metodám při jejich použití přistupuje.

Dále projekt obsahuje třídu *EntitiesContext.cs,* která konfiguruje databázi, určuje její tabulky a vztahy mezi nimi.

V adresáři *Migrations* se nachází automaticky generovaný kód, který provádí změny v databázi na základě kódu v souborech a třídách zmíněných výše.

* 1. Projekt *Rentals.Web*

V tomto projektu se nachází aplikace jako taková. Využívá se zde MVC (Model-View-Controller) architektury. To znamená, že každá zobrazovaná stránka v prohlížeči, která je generována dle kódu v souborech formátu *.cshtml* (View) má svůj model. Tento model představuje klasickou třídu, která obsahuje všechna data v podobě vlastností, ke kterým lze přistupovat v souborech formátu *.cshtml*. Onen zmíněný model se plní daty v kontroleru. Kontroler obsahuje metody vracející datový typ *ActionResult* s označením *Data Annotations* specifikujícím URL adresu. Takto označená metoda se poté volá pokaždé, když je v prohlížeči načtena stránka s definovanou URL adresou.

* + 1. Zákaznická část aplikace

Projekt je rozdělen do dvou oblastí. První z nich není ve struktuře souborů aplikace explicitně označena, adresáře se soubory do ní spadající jsou v hlavním adresáři projektu. Tato oblast primárně obstarává část aplikace věnovanou zákazníkům, obsahuje ale také například prostředí pro přihlašování uživatelů. I kvůli tomu je tato část aplikace přístupná všem úspěšně přihlášeným uživatelům. Uživatelé s rolí Zaměstnanec anebo Administrátor mají z této základní oblasti přístup do administrační části aplikace.

* + 1. Administrační část aplikace

Soubory týkající se této části aplikace se nachází v adresáři *Areas.* Oblast je od výchozí oblasti pro zákazníky oddělena. Má vlastní kontrolery i modely pro zobrazování dat. Webové rozhraní této oblasti aplikace se nachází na URL adrese */Admin*.

Uživatelé s rolemi Administrátor anebo Zaměstnanec mohou pomocí funkcí implementovaných funkcí spravovat aplikaci. To zahrnuje správu předmětů, výpůjček a přehled zákazníků.

1. Analýza a dokumentace chyb
2. Oprava chyb a úprava stávajících řešení

Závěr

Tak jsem to dokázal! A teď ještě, co jsem mohl udělat jinak a jak by se dalo v práci pokračovat.

Práce mi pomohla zdokonalit se v oblasti, která mě při studiu zaujala nejvíce, v tvorbě webových aplikací.

a navíc díky ní vznikla pro uživatele – studenty a personál SPŠSE a VOŠ – pohodlnější a méně problémová webová aplikace.

Seznam obrázků

[Obrázek 1 Příklad generovaného návrhu řešení vývojovým prostředím MS Visual Studio 2](file:///D:\Osobní%20-%20záloha%20(foto,%20kazety,%20důležité)\Škola\SPŠSE%20Liberec\MATURITA\MP2021-Petr-Horák-Revize_aplikace_Rentals.docx#_Toc65840742)

[Obrázek 2 Chybová hláška programu Microsoft SQL Server Management Studio 4](file:///D:\Osobní%20-%20záloha%20(foto,%20kazety,%20důležité)\Škola\SPŠSE%20Liberec\MATURITA\MP2021-Petr-Horák-Revize_aplikace_Rentals.docx#_Toc65840743)

Použitá literatura

1. **Wikipedia. *Bak file.* [Online] 18. Únor 2021. https://en.wikipedia.org/wiki/Bak\_file.**

**2. učitelé SPŠSE. Úvod. *SPŠSE a VOŠ Liberec.* [Online] 01. 09 2016. [Citace: 01. 09 2016.] https://www.pslib.cz.**

**3. nuget. [Online] Microsoft. https://www.nuget.org/.**

**4. Introduction to Identity on ASP.NET Core. *Microsoft Docs.* [Online] Microsoft , 15. 07 2020. https://docs.microsoft.com/cs-cz/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio.**

1. Seznam přiložených souborů

Text

1. Další příloha