

Studijní program: B2646/Informační technologie

Studijní obor: 1802R007/Informační technologie

Autor práce: Michal Červinka

Vedoucí práce: Ing. Jan Hybš

Single-page Aplikace sloužící pro komunikaci mezi uživateli

Anotace

Cílem projektu je vytvořit responzivní multiplatformní webovou SPA (Single-page aplikaci) sloužící pro okamžitou komunikaci mezi uživateli. Projekt bude využívat architekturu server/klient, kde server, realizovaný pomocí frameworku ASP.NET, poskytuje API, které bude klient využívat. Komunikace v reálném čase bude zajištěna pomocí webových socketů.

Prohlášení

Byl jsem seznámen s tím, že na mou ročníkovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé ročníkové práce pro vnitřní potřebu TUL.

Užiji-li ročníkovou práci nebo poskytnu-li licenci k jejímu vy-užití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do je-jich skutečné výše.

Ročníkovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé ročníkové práce a konzultantem.

V Liberci dne …………………………….

Podpis: ……………………………………

Poděkování

Pan inženýr Jan Hybš byl díky svým zkušenostem a radám skvělý vedoucí této ročníkové práce a rád bych mu tímto způsobem poděkoval za jeho ochotu a trpělivost.

Obsah

Úvod

Tato práce podrobně popisuje proces vývoje aplikace pro okamžité posílání zpráv mezi 2 uživateli přes internet. Jak už ze zadání vyplývá, tato aplikace je vyvinuta pomocí frameworku ASP.NET, který je pro vývoj aplikace s takovými požadavky jeden z nejlepších kandidátů. Výsledkem tohoto projektu je aplikace, která umožňuje přihlásit již existujícího uživatele, zaregistrovat nového uživatele, odhlásit uživatele a pro každého uživatele nabídnout možnost napsat okamžitou zprávu jakémukoliv jinému uživateli. Z těchto požadavků nepřímo vyplývají další požadavky jako např. Bezpečné ukládání hesel, nepřihlášený uživatel nemůže nikomu napsat zprávu atd. Součástí práce je i databáze, která se nachází na školním serveru a obsahuje informace o všech založených účtech a o všech zaslaných zprávách. V neposlední řadě se tato práce také zabývá všemi technologiemi, které byli použity pro vývoj aplikace splňující všechny dané požadavky ze zadání.

1. Použité technologie
   1. Microsoft Visual Studio 2019

Microsoft Visual Studio 2019 je vývojové prostředí od společnosti Microsoft. Dá se použít pro vývoj konzolových aplikací, webových stránek/aplikací, grafického rozhraní atd. Obsahuje editor kódu s podporou IntelliSense (např. dokončování slov) a velmi dobře zvládnutou funkci redaktorování (provádění změn, tak aby neměli vliv na chování kódu). Součástí Visual Studia 2019 je i debugger, který nám nabízí významnou pomoc při hledání chyb v našem kódu. Já jsem Visual Studio při práci se zadáním projektu používal jako prostředek pro vývoj webové aplikace sloužící pro okamžitou komunikaci mezi uživateli, a to jak serverovou část, tak i část na straně klienta. K vývoji serverové části této aplikace jsem použil framework ASP.NET, který je součástí .NET frameworku a Visual Studio 2019 mi umožňuje s ním pracovat.

* 1. Microsoft SQL Server Management Studio 18

Microsoft SQL Server Management Studio 18 (dále jen SQL Server) je analytický systém pro tvorbu a správu relačních databází v jazyce T-SQL. Tento jazyk je proprietární rozšíření jazyka SQL od společnosti Microsoft. V tomto projektu byl SQL Server použit pro tvorbu a přístup k databázi, která v sobě ukládá informace o uživatelích a zprávách mezi uživateli. Databáze byla uložena na školním serveru dostupným pro studenty (147.230.21.34).

* 1. ASP.NET

ASP.NET je součástí .NET Frameworku (soubor technologií a softwarových produktů). Je nástupcem technologie ASP (Active Server Pages). Jde o framework, který umožňuje vývoj serverových částí dynamických webových aplikací a služeb. Pro vývoj aplikace, které se tato práce zabývá, byl použit právě tento framework, protože je to jedna z nejlepších možností, které lze použít pro vývoj aplikace splňující zadané podmínky a také protože součástí zadání bylo použít právě framework ASP.NET. Konkrétně byl použit ASP.NET Core, což je nástupce ASP.NET, který je obsažen v novém vydání .NET Core. Jako softwarová architektura byl použit Model-View-Controller, což je architektura, kterou ASP.NET podporuje (konkrétně ASP.NET MVC).

1. Použití
   1. Úvodní strana

Při první návštěvě webové aplikace uživatel pravděpodobně zavítá na úvodní stránku. Tato stránka je velice jednoduchá, kromě vítacího nadpisu se zde nachází pouze tlačítko pro přihlášení existujícího uživatele do aplikace a tlačítko pro registraci nového uživatele. Tyto tlačítka nás po kliknutí přesměrují na příslušnou stránku pomocí správné URL adresy (Uniform Resource Locator), ve které je za doménovou adresou serveru zapsáno umístění zdroje na serveru (v tomto případě stránky). URL adresa samotné úvodní strany má 2 verze. Buď lze napsat doménovou adresu serveru + /home anebo stačí jen doménová adresa, jelikož ASP.NET uznává /home za úvodní stranu, a z důvodu zvýšení intuitivity proto podporuje i druhou variantu.

* 1. Registrace

Při první návštěvě je také velmi pravděpodobné, že uživatel nemá založený žádný účet. Nepřihlášený uživatel nemůže používat většinu služeb, které mu tato aplikace poskytuje, a proto je potřeba vytvořit si nový profil. Na stránku pro registraci se dá dostal buď z úvodní stránky nebo zadáním správné URL adresy, což je doménová adresa + /register. Na této stránce se nachází 2 textová pole, jedno pro zadání jména nového uživatele a druhé pro zadání hesla. Po kliknutí na tlačítko “login” aplikace zkontroluje, jestli již v databázi neexistuje uživatel se stejným jménem jako je to, co nový uživatel zadal do prvního textového pole (uživatelská jména totiž jsou a musí být unikátní). Pokud ano, registrace se neuskuteční a uživateli se objeví varování informujíce ho o nemožnosti použití daného jména. Kontrola síly hesla by zde mohla být také implementovaná, nicméně není, z důvodu rychlejšího testování (tato implementace by byla velice jednoduchá). V případě úspěšné registrace se jméno a heslo nového uživatele zapíší do databáze (heslo je samozřejmě převedeno na bezpečně zahashovaný tvar viz. níže). Uživatel může nyní používat všechny služby aplikace.

* 1. Přihlašování

V případě, že uživatel již má vytvořený účet, je potřeba umožnit mu přihlásit se k danému účtu. K tomu slouží přihlašovací stránka s URL adresou doménová adresa + /login. Vzhledově se vcelku podobá předchozí stránce pro registraci, funkcionalitu má ale jinou. Po zadání uživatelského jména a hesla se aplikace pokusí najít v databázi uživatele, který má stejné oba údaje. Pokud žádného nenajde, proces přihlašování se nezdaří a obdobně jako při nezdařené registraci se uživateli objeví varování o nesprávnosti údaje či obou údajů. V případě, že aplikace v databázi najde uživatele s totožnými údaji, osoba je přihlášena právě jako tento uživatel a od této chvíle je oprávněna používat všechny služby aplikace.

* 1. Chat

Hlavním účelem této aplikace je posílání zpráv mezi 2 uživateli. Stránka určená k chatu má URL adresu doménová adresa + /chat a neoprávněný uživatel se na ní nedostane. Pro získání oprávnění je potřeba se přihlásit. Chatování v této aplikaci funguje formou Instant Messaging, jelikož zasílání zpráv je okamžité (Pokud jeden uživatel napíše druhému uživateli, který je přihlášený a zrovna se kouká na zprávy od prvního uživatele, druhý uživatel uvidí zprávu okamžitě). Na této stránce může uživatel vidět seznam všech ostatních uživatelů, kterým může napsat. Na každého uživatele v seznamu lze kliknout a zobrazit si všechny zprávy, které si daní uživatelé poslali a také textové pole pro zadání a následné zaslání nové zprávy. Také se tu nachází možnost odhlášení. Po kliknutí na odhlašovací odkaz je uživatel odhlášen a jeho oprávnění pro vstup na tuto stránku je odebráno. Pro opětovné získání daného oprávnění je potřeba se znovu přihlásit buď k tomu samému účtu nebo k nějakému jinému.

1. Vývoj aplikace
   1. Databáze

Jako první krok pro vývoj zadané chatovací aplikace bylo vytvořit relační databázi (Jiný druh databáze v dnešní době nemá smysl). V našem případě tato databáze nemusí být nijak složitá, jediné, co si potřebujeme ukládat jsou informace o uživatelích a zprávy mezi nimi. Jsou tedy potřeba jen dvě entity (tabulky) – uživatelé a zprávy. Vztah mezi těmito entitami je následující: K uživateli může patřit n zpráv, ale také k němu nemusí patřit žádná, zatímco ke zprávě jsou přidružení právě 2 uživatelé (odesílatel a příjemce). Každý uživatel má několik atributů: Unikátní celočíselný identifikátor, uživatelské jméno, které je pro každého uživatele unikátní, hash hesla, do kterého se ukládá hash hesla vygenerovaný hashovým algoritmem viz. níže. V žádném případě do databáze neukládáme heslo v normálním nezabezpečeném tvaru! Každá zpráva má opět několik atributů: Unikátní celočíselný identifikátor, text zprávy a jelikož jde o slabou entitu, musí obsahovat i cizí klíč k silné entitě (k uživateli), jinak by nemohla existovat). V našem případě má cizí klíče 2, 1 odkazující na odesílatele dané zprávy a 1 odkazující na příjemce.

* 1. Entity framework

Po založení nového projektu ASP.NET MVC ve Visual Studiu je potřeba propojit databázi s aplikačním modelem, abychom mohli s databází jednoduše komunikovat. K tomu se v .NET projektech v drtivé většině používá Entity Framework. Tento rámec nám umožňuje pracovat s databází na vyšším levelu abstrakce, protože z entit a atributů nám dokáže vytvořit třídy a vlastnosti, tudíž nám umožňuje pracovat s daty jako s objekty. K vytvoření takového propojení můžeme použít 2 způsoby. Buď tzv. model-first (vytvoříme model a necháme podle něj Entity Framework vygenerovat databázi) anebo tzv. database-first (vytvoříme databázi a necháme podle ní Entity Framework vygenerovat model). Já jsem pro tento konkrétní projekt použil způsob database-first.

* 1. Controllery

Controllery jsou v ASP.NET MVC architektuře od toho, aby se starali o všechny příchozí URL requesty (požadavky). Controller je třída, která obsahuje veřejné metody (tzv. action methods). Tyto metody reagují na příchozí požadavky, získájí ze služeb, popřípadě z modelu potřebná data a vrátí příslušnou odpověď. V ASP.NET MVC všechny controllery musí končit slovem „Controller“ a být potomkem třídy Controller z knihovny Systém.Web.Mvc. V tomto konkrétním projektu máme celkem 4 controllery. HomeController, který se stará o požadavky z úvodní stránky (v našem případě tam žádné ani být nemusí). LoginController, který se stará o přihlašování existujících uživatelů, o kontrolu správnosti přihlašovacích údajů a o přesměrování do chatovací aplikace v případě správných údajů. RegisterController, který je podobný předchozímu controlleru. Tento controller se stará o registraci nových uživatelů. Posledním a nejrozsáhlejším controllerem je ChatController. Ten se stará o zobrazování všech uživatelů, kterým lze poslat zprávu, zobrazování konverzace mezi přihlášeným a navoleným uživatelem, posílání zpráv mezi uživateli a odhlašování uživatele. Tento controller má autorizovaný vstup (viz. níže) tj. nemůže s ním komunikovat nepříhlášený uživatel.

* 1. Views

Views (pohledy) jsou UI (user interfaces). Používají se k zobrazování chtěných dat uživateli. Pohledy komunikují s controllery, ke kterým jsou přiděleny. V případě ASP.NET MVC se pohledy nacházejí ve složkách nesoucí stejné jméno jako controller, ke kterým patří (např. view se jménem index ve složce Home bude komunikovat s metodou Index() v HomeControlleru). Controller může mít na starosti více než 1 pohled ale pohled patří právě pod 1 controller. V případě našeho projektu potřebujeme pohledy pro domovskou stránku, přihlašování, registraci, zobrazení uživatelů pro přihlášeného uživatele a zobrazení možnosti napsat jistému uživateli zprávu.

* 1. SignalR

ASP.NET Core SignalR je knihovna, která umožňuje přidávání real-time funkcí do webových aplikací a služeb (zpracovávání funkcí v reálném čase). Pro správný provoz zadané chatovací aplikace potřebujeme SignalR pro posílání zpráv. Když jeden uživatel pošle zprávu druhému uživateli, jemu se zpráva musí zobrazit ve chvíli kdy přijde a bez refreshování stránky (pouze pokud je dotyčný uživatel online a má rozkliknutého uživatele, který mu právě zprávu poslal). K tomu použijeme tzv. hub (rozbočovač) (v našem případě ChatHub). Tento hub se nachází na serverové části aplikace a čeká až se ho klient na něco zeptá. V našem případě čeká až některý klient pošle zprávu jinému klientovi. V případě, kdy tato situace nastane, ChatHub příjme informaci o odeslání zprávy a danou zprávu odešle jak příjemci, tak i zpět odesílateli. Oba klienti na druhé straně neustále čekají na příchozí zprávu od hubu, která jim oznámí, co za zprávu jim přišlo. Kromě toho nabízí klientům možnost zjistit jejich id, které je potřeba pro rozeznání uživatelů. Zpráva se nezobrazí uživateli, pokud daná zpráva nebyla určena pro něj anebo pokud nemá uživatel rozkliknutého právě toho uživatele, který mu zprávu poslal. Další záležitosti jako např. zápis zprávy do databáze atd. signalR neřeší, ten je pouze pro real-time zobrazení chtěných informací.

* 1. Bezpečnostní opatření

Tato webová aplikace ukládá osobní data o uživatelích (uživatelské jméno a heslo) a proto je potřeba implementovat jistou ochranu osobních dat. V žádném případě nesmíme ukládat hesla v normálním tvaru tak jak jsou, v případě nějakého útoku, či nechtěnmu úniku dat by každý hned věděl hesla uživatelů. Proto musíme ukládat hesla pouze jako tzv. hashe (otisky). Hash je výstup tzv. hashovacího algoritmu, jejímž vstupem je textový řetězec obsahující uživatelovo heslo. Tyto algoritmy jsou dělaný tak, aby fungovali pouze jednostraně (tzn. z hesla můžeme získat hash, ale z hashe už nemáme žádnou možnost, jak získat zpět původní heslo). Ze stejného hesla vytvoříme stejný hash a pouze drobná změna v hesle se projeví velkým způsobem na výstupním hashi. Výstupní hashe jsou pokaždé stejně dlouhé. V databázi pod políčkem „Heslo“ budeme ukládat tyto hashe. Hashovacích algoritmů je více, ale v našem případě je použit algoritmus SHA256, což je v dnešní době dobrý algoritmus. Poté musíme ještě implementovat tzv. salt, protože některá jednoduchá hesla se dají podle jejich hashů na internetu dohledat. Salt je pouze textový řetězec náhodných znaků, který se přidá před či za heslo a dohromady se to zahashuje. Tímto způsobem se z hesla stane neobvyklý řetězec znaků, který už podle hashe dohledat nejde.

Závěr

Práce splňuje všechny požadavky, které byli obsaženy v zadání (viz. Anotace). Při vývoji praktické části projektu se objevila řada problémů, které by mě vůbec nenapadli, že se s nimi mohu někdy střetnout, nicméně jsem je zdárně vyřešil a práci úspěšně dokončil. Získal jsem spoustu cenných zkušeností, zejména v oblasti vývoje aplikací v ASP.NET frameworku, ale i mimo ni. Při vývoji databáze (ačkoliv šlo o jednoduchý návrh databáze) jsem získal pár nových vědomostí.