Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj webové aplikace pro evidenci lezeckých záznamů

Vypracoval: Jakub Cink
Vedoucí práce: Mgr. Radim Remeš, Ph.D.

České Budějovice 2022

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub CINK**
Osobní číslo:	**E19025**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vývoj webové aplikace pro evidenci lezeckých záznamů**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvoření webové aplikace (website) fungující jako lezecký tréninkový deník pro uživatele. Website bude umožňovat správu jejích uživatelů a přidělování uživatelům různých rolí (např. administrátor, redaktor, registrovaný uživatel, apod.). Uživatelé si budou moci vést svůj lezecký tréninkový deník, kam budou zaznamenávat např. vylezené cesty, stěny nebo skalní oblasti, včetně obtížnosti, stylu lezení, své plánované i dosažené cíle, záznamy z tréninků, své poznámky. Aplikace bude nabízet zobrazení přehledů výkonu za různá časová období a další vybraná statistická data. Aplikace bude vyvíjena s použitím softwarových komponent ADO.NET.

Metodický postup:

1. Studium odborné literatury.
2. Návrh, popis vývoje a implementace aplikace.
3. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
4. Závěr.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Albahari, J. (2021). *C# 9.0 in a Nutshell.* Sebastopol, CA (USA): O'Reilly.
2. Dean, J. (2019). *Web programming with HTML5, CSS, and JavaScript.* Burlington, MA (USA): Jones & Bartlett Learning.
3. Seidl, M., Scholz, M., Huemer, C., & Kappel, G. (2015). *UML @ Classroom: An Introduction to Object-Oriented Modeling.* Cham, CH: Springer.
4. Sharp, J. (2018). *Microsoft Visual C# Step by Step.* Upper Saddle River, NJ (USA): Pearson Education.
5. Skeet, J. (2019). *C# in Depth.* Shelter Island, NY (USA): Manning.

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **25. března 2021**
Termín odevzdání bakalářské práce: **15. dubna 2022**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 31. března 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

…… ……

Datum Podpis studenta

Poděkování

Mé poděkování patří vedoucímu mé bakalářské práce panu Mgr. Radimu Remešovi, Ph.D., za vstřícnost a cenné připomínky při vypracování bakalářské práce.

Obsah

1 Úvod

S rozvojem vysokorychlostního Internetu se otevřely možnosti pomocí internetového prohlížeče k využívání velmi komplexních internetových stránek, které mohou nahrazovat některé programy, které jinak měl uživatel nainstalované lokálně na svém počítači. Tyto stránky nazýváme webové aplikace. (Jak na internet, c2022)

Téma vývoje webové aplikace jsem si zvolil, protože jsem se chtěl zdokonalit ve vývoji webových aplikacích a vyzkoušet si vlastní návrh, následnou implementaci aplikace a její následné nasazení v reálném prostředí. Rovněž jsem vášnivým lezcem a chtěl jsem vytvořit aplikaci nejen pro mě, ale pro další potencionální uživatele, kteří by mohli svoje lezecká data uchovávat, sledovat svůj výkonnostní pokrok a dále s vytvořenými daty pracovat podle uvážení daného uživatele.

V teoretické části této bakalářské práce se věnuji definici webových aplikací, databází a definicí samotného horolezeckého sportu a jednotlivých disciplín. V dalších kapitolách se zabývám popisem technologie pro vývoj webových aplikací.

V praktické části popisuji vývoj aplikace na platformě ASP.NET Core, která nese název Lezecký deník.

Pro usnadnění orientace v textu používám pro názvy tříd, metod, proměnných souvisejících s vývojem aplikace a pro názvy typů proměnných odlišný font. Pro názvy souvisejících s výslednou aplikací využívám kurzívu.

1.1 Cíl práce

Cílem práce je vývoj webové aplikace, jež dovoluje jejím uživatelům evidovat různá data, která souvisí se sportovním lezením a následně tyto data zobrazovat. V případech, kde to má smysl, tyto data editovat. Dále aplikace umožní uživatelům zaregistrovat vlastní účet, pod kterým se budou do aplikace přihlašovat a budou tyto data pod tímto účtem vkládat. Uživatelům aplikace také dává možnost evidovat a plánovat tréninkové jednotky a definovat pro tyto uživatele cíle a tyto cíle se budou vyhodnocovat jako splněné či nesplněné. Následně se tyto cíle budou ukládat a zobrazovat. Aplikace též uživateli dovolí si nechat vizualizovat jím zadaná data do celkového přehledu. K účtům uživatelů se budou dát navázat různé role, které budou na základě těchto rolí rozšiřovat či omezovat funkce v aplikaci. Rovněž aplikace umožní tyto role měnit určeným rolím pro tuto funkci.

2 Úvod do problematiky

2.1 Webová aplikace

Webová aplikace je program, který je uložen na vzdáleném serveru, předáván je přes počítačovou síť Internet nebo případně přes interní síť a pomocí rozhraní prohlížeče internetových stránek je zobrazován na klientském zařízení. Za webovou aplikaci můžeme považovat jakoukoliv webovou stránku, která obsahuje komponentu, která provádí nějakou funkci pro uživatele. Proto, aby mohla webová aplikace fungovat, je zapotřebí webový server, aplikační server a databázi. Webový server spravuje požadavky od klienta a aplikační server dokončí požadovanou úlohu. Databázi lze použít k uložení dat a informací. Mezi hlavní výhody patří, že dovoluje více uživatelům přístup ke stejné verzi aplikace. Webové aplikace není potřeba instalovat, fungují na různých zařízeních a může s k nim přistupovat přes více prohlížečů. (TechTarget, c2006-2022)

2.2 Databáze

Databáze je kolekce informací, které společně existují nějakou delší dobu. Slovo databáze běžně odkazuje na kolekci dat, jež je spravována systémem řízení bází dat (zkráceně SŘBD, v angličtině se používá zkratka DBMS z database management system). (Garcia-Molina et al., 2008)

Od SŘBD očekáváme, že dovolí uživateli vytvářet nové databáze a specifikovat jejich schéma, pomocí specializovaného jazyka DDL (data-definition language). Současně umožní uživateli provádět dotazy a modifikovat data pomocí dotazovacího jazyka DDL (data-manipulation language). Také dovoluje ukládat velké množství dat. V případě chyb a výpadků umožňuje obnovu dat a v případě používání databáze více uživateli sjednává přístup těchto uživatelů současně, tak aby nedošlo k nežádoucím problémům a narušení dat. (Garcia-Molina et al., 2008)

Základem struktury databáze je datový model. Jedná se o soubor nástrojů pro popsání dat a vztahů mezi nimi. Datový model nám poskytuje způsob, jak popsat návrh databáze na fyzické, logické a zobrazovací úrovni. Datový model můžeme klasifikovat do čtyř kategorií.

- Relační model, který využívá kolekci tabulek uchovávajíc jednak data a také vztahy mezi tabulkami.

- Entitně vztahový model (E-R model), kde data jsou založeny na vnímání reálného světa, které odpovídají kolekci objektů nazývány entity.
- Objektově založený datový model může být vnímán jako rozšíření E-R modelu, kde je přidáno zapouzdření, metody a identifikátor objektu.
- Polostrukturovaný datový model, který dovoluje jednotlivým datovým položkám se stejným datovým typem rozdílné vlastnosti.
- Historicky stojí za zmínku síťový datový model a hierarchický datový model, které předcházeli relačnímu modelu. (Silberschatz et al., c 2006)

2.3 Horolezectví a sportovní lezení

„Ve svém nejpůvodnějším významu vyjadřuje pojem horolezectví aktivitu vedoucí k výstupu na vrchol hory. Slovo horolezectví ovšem v současnosti v češtině vyjadřuje celý soubor dalších činností a specializovaných sportovních disciplín.“ (Kublák, 2014)

2.3.1. Formy horolezectví

Pojem sportovní horolezectví lze rozdělit na několik podskupin podle různých hledisek.

Jedním ze zásadních hledisek je takzvané „etické pojetí“. Zahrnuje v sobě prvky motivace, přání, názorů, zvyků. Tento termín má velký vliv na způsob a provozování horolezeckých aktivit. Díky tomuto termínu rozdělujeme horolezectví na tradičně (klasicky) pojímané a gymnasticky pojímané. Podle intenzity horolezeckého výkonu lze taktéž dělit tento sport do tří kategorií, a to vrcholová, výkonností a rekreační. Další hledisko je roční období, kdy rozdělujeme na horolezectví letní a zimní. Podstatným hlediskem je také prostředí provozování horolezectví a tím mohou být skály, hory, velehory s věčným sněhem a ledem, umělá lezecká stěna, apod. Poslední formou dělení je na závodní a nezávodní. (Kublák, 2014)

Pod pojmem sportovní horolezectví si můžeme díky výše zmíněných hlediscích a jejich kombinací rozlišit všechny aktivity, které pod pojem sportovní horolezectví patří. Nicméně mezi jednotlivými aktivitami a disciplínami není jasná hranice a dochází běžně k velkému prolínání a splývání různých disciplín. (Kublák, 2014)

2.3.2. Tradiční pojetí horolezectví a gymnastické pojetí horolezectví

Tradiční pojetí horolezectví, někdy též nazývané klasické, ovlivňuje podobu řady disciplín. Základním principem je požadavek, aby přirozený terén zůstal po výstupu člověka v původním stavu. Člověk zde má vykonat výstup svými schopnostmi, a ne pomocí techniky. Naproti tomu stojí požadavek o ochraně lidského života a zdraví. Tento požadavek jde proti předchozímu a je mu nadřazen tudíž je možné zasahovat do přírody a např. vytvořit trvalé jistící prostředky. Naproti tomu gymnastické pojetí horolezectví má hlavní princip ze zvládnutí lezeckého pohybu nutného k přelezení určité části výstupu. S tímto principem se pojí soutěživost a neustále zlepšování v překonávání více náročných obtížností. Z této snahy se zrodilo závodní lezení, které si oblíbilo prostředí umělých horolezeckých stěn. (Kublák, 2014)

2.3.3. Horolezecké disciplíny

Horolezení se dělí na různé horolezecké disciplíny jsou to:

- **Skalní lezení a pískovcové lezení** – Jedná se dnes o nejdominantnější formu horolezectví. Určujícím faktorem pro tento typ je prostředí, kde se tato činnost vykonává a tím jsou skály. Ve skalách lze pojmout horolezectví tradičně nebo gymnasticky. V rámci České republiky se skalní horolezectví dělí na dvě části, a to na pískovcové a na lezení po ostatních skalních terénech tvořenými jinými horninami než je pískovec.
- **Bouldering** – Jedná se o především lezení na balvany nebo malé skalky, které zpravidla nepřesahují výšku 2–5 metrů. Leze se bez jištění a v případě pádu se odskakuje na zem. Tato disciplína se přiblížila svým charakterem ke skutečné gymnastice.
- **Ledové horolezectví** – Pro tento typ horolezectví je terénem led, kdy se dnes hlavně provozuje na zamrzlých vodopádech v horských údolích během zimy.
- **Horské a velehorské horolezectví** – Tato forma horolezectví je určena taktéž prostředím. Tím prostředím jsou hory, k upřesnění se používá rozčlenění na hory, střední velehory, vysoké velehory.
- **Zimní horolezectví** – Za disciplínu zimní horolezectví se považuje, že během zimního období terén díky zimě zcela změnil svůj charakter.
- **Vysokohorská turistika** – Nejpůvodnější forma horolezectví. Jednou z hlavních motivací je zde vystoupat na vrchol hory.

- **Zajištěná cesta (klettersteig, via ferrata)** – Jde o disciplínu nejrazantnějšího projevu rekreační úrovně horolezectví. Jedná se o horolezeckou trasu, která je zpřístupněna a zajištěna umělými prostředky jako jsou železné kramle, žebříky, řetězy anebo jistící ocelová lana.
- **Umělé stěny** – První umělé stěny začaly vznikat s rozvojem gymnastického pojetí horolezectví. První motivací bylo uniknout vlivu prostředí a druhou získat tréninkové prostředí pro výkonnostní rozvoj lezců.
- **Skialpinismus** – Tento termín označuje všechny horolezecké aktivity provozované v horách s lyžemi ve volném terénu.
- **Interdisciplíny** – Pokud prvky horolezectví zasahují do jiných sportů či aktivit. Mluvíme např. o termínech jako je speleoalpinismus, paraglidealpinismus nebo canyoning. (Kublák, 2014)

2.3.4. Klasifikační stupnice

Jako každý sport musí existovat stanovená pravidla pro hodnocení výkonů. Horolezecký výkon má dvě části. Za prvé je to obtížnost, která určuje náročnost překážky, jenž je nutná k překonání. Druhou složkou je náročnost, která značí, jak bude lezec po výkonu vyčerpán. Existuje několik stupnic, které se samostatně vyvíjeli a používají se pro různé disciplíny. Samotný proces pro udělování stupně obtížnosti je proces dlouhodobý a subjektivní. (Kublák, 2014)

Jedná se o stupnice:

- Klasifikační stupnice UIAA
- Klasifikační stupnice technického lezení
- Britská klasifikační stupnice
- Klasifikační stupnice USA
- Francouzská klasifikační stupnice
- Západoalpská klasifikační stupnice
- Ruská klasifikační stupnice a Ruská lezecká stupnice
- Skotská stupnice pro lezení v ledu
- Klasifikační stupnice pro lezení v ledu
- Mezinárodní klasifikační stupnice pro lezení v ledu
- Klasifikační stupnice WI pro obtížnosti lezení v ledu
- Klasifikační stupnice pro moderní mixové lezení

- Boulderingové klasifikační stupnice – Francouzká stupnice Fb a americká stupnice V (Kublák, 2014)

Klasifikační stupnice UIAA

„**I** – Lehké. Nejjednodušší forma skalního lezení, ne však již pouze a bezvýhradně chodecký terén. K zabezpečení rovnováhy je třeba rukou.

II – Mírně těžké: Začátek lezení, při kterém je vyžadována technika tří pevných bodů.

III – Středně těžké: Na exponovaných místech je jíž doporučováno mezijištění.

IV – Těžké: Jsou nezbytné lezecké zkušenosti, úseky tohoto stupně již obvykle vyžadují více mezijištění.

V – Velmi těžké: Lezení již klade značné nároky na trénovanost lezce. Mnohdy se již jedná o převislé úseky.

VI – Neobyčejně těžké: Nezbytná je dobrá technika a spolehlivé jištění.

VII – Mimořádně těžké: Velká expozice se často spojuje s malými možnostmi jištění, i výborní lezci potřebují pro každý druh skály speciální přípravu, aby výstupy tohoto stupně vylezli bez pádu.

VIII – X – Stupňování předchozích obtížností, vyžaduje již velmi specifický trénink. Obvykle je tato obtížnosti nedostupná lezcům, kteří netrénují na umělé stěně a nevěnují značnou část svého tréninkového plánu specifickému posilování. Běžné lezení v těchto stupních obtížnosti je vyhrazeno vrcholovým sportovcům.

XI – Současná hranice lezeckých možností. Zpravidla je nezbytné předchozí nacvičování cesty, a ani špičkoví lezci nejsou schopni úseky tohoto stupně opakovat často. K překonání jsou nezbytné ideální podmínky, špičková forma a naprosté soustředění na výkon.“ (Kublák, 2014)

Stupnice UIAA je směrem nahoru otevřená. To znamená, že může dojít k rozšíření. Také se pro upřesnění a co nejpřesnější rozlišení používají ještě znaménka + a -. (Kublák, 2014)

Na obrázku 1 jsou vyobrazeny některé klasifikační stupnice a jejich vzájemný převodní vztah.

Obrázek 1 Klasifikační stupnice

UIAA	Fr	USA	Saxony	UK	AU		Fr	USA
							bouldering	
I	1	5.2	I	moderate	11			
II	2	5.3	II	difficult				
III	3	5.4	III	very difficult	12			
IV	4	5.5	IV	4a	13			
V-	5	5.6	V	4b	14			
V		5.7	VI	4c	15		Fb3	
V+		5.8	VII	5a	16		Fb4	V0
VI-	5+	5.9	VIIb	5b	17		Fb5a	
VI	6a	5.10a	VIIc		18		Fb5b	V1
VI+	6a+	5.10b	VIII	5c	19		Fb5c	
VII-	6b	5.10c	VIIIb		20		Fb6a	
VII	6b+	5.10d	VIIIc	6a	21		Fb6a+	V2
VII+	6c	5.11a	IX		22		Fb6b	
VIII-	6c+	5.11b	IXb		23		Fb6b+	
VIII	7a	5.11c	IXc	6b	24		Fb6c	V3
VIII+	7a+	5.11d	X		25		Fb6c+	V4
IX-	7b	5.12a	Xb	6c	26		Fb7a	V5
IX	7b+	5.12b	Xc		27		Fb7a+	V6
IX+	7c	5.12c	XI	7a	28		Fb7b	V7
X-	7c+	5.12d	XIb		29		Fb7b+	V8
X	8a	5.13a	XIc	7b	30		Fb7c	V9
X+	8a+	5.13b			31		Fb7c+	V10
XI-	8b	5.13c			32		Fb8a	V11
XI	8b+	5.13d			33		Fb8a+	V12
XI+	8c	5.14a			34		Fb8b	V13
	8c+	5.14b			35		Fb8b+	V14
	9a	5.14c			36		Fb8c	V15
	9a+	5.14d						
		5.15a						

Zdroj: (Kublák, 2014)

2.4 Stavba a organizace lezeckého tréninku

Trénink organizujeme do cyklů a tyto tréninkové cykly dělíme na základě délky trvání, a to na:

- Makrocyklus – Jedná se o nejdelší cyklus. Jeho ideální délka je jeden rok, ale používají se i kratší, což může být půlroční cyklus nebo delší a ten může dosahovat dvou až čtyř let.
- Mezocyklus – Převádí dlouhodobé záměry makrocyklu do praxe. V ideálním případě je jejich délka 4 týdny, ale opět se v praxi používají kratší nebo delší.

- Mikrocyklus – Mezocyklus je tvořen mikrocykly. Jejichž ideální délka je 7 dní a shoduje se tudíž s týdnem. Opět se v praxi můžeme setkat s kratším nebo delším mikrocyklem.
- Tréninková jednotka – Je nejmenší a nejdůležitější součást zvyšováním výkonosti. Má různou délku a může se kumulovat do bloků a fází nebo v průběhu dne vícekrát opakovat. (Tefelner, 1999)

3 Použité nástroje a technologie

3.1 Vývojové prostředí

Jako vývojové prostředí jsem zvolil od firmy Microsoft Visual Studio a jeho aktuálně nejnovější verzi, tudíž Visual Studio 2022.

Integrované vývojové prostředí nebo také zkráceně IDE z anglického *integrated development environment* je software podporující mnoho aspektů softwarového vývoje. Visual Studio IDE se může využít jako platforma k úpravám, ladění a sestavování kódu a následně k publikaci aplikace. Dále Visual Studio nabízí na rozdíl od ostatních IDE navíc kompilátory, nástroje pro dokončování kódu, grafické návrháře a další nástroje pro vývoj softwaru. (G. Lee et al., 2021)

3.2 ASP.NET Core

Jedná se o nejnovější evoluci od Microsoftu na populární webový framework ASP.NET. První verze ASP.NET Core byla představena v červnu 2016 (Lock, 2018). Jeho aktuálně nejnovější verze je verze 6.0 představená v listopadu 2021. (Roth, 2021)

Microsoft v dokumentaci uvádí, že ASP.NET Core je multiplatformní, vysoce výkonný, open source framework pro vývoj moderních, cloudových a k internetu připojených aplikací. S tímto frameworkem je možné vyvinout aplikace, služby a mobilní backend. Tuto aplikaci nasadit na cloud nebo lokálně a je spustitelný na .NET Core. (Roth et al., 2022)

3.2.1. Návrhové vzory

ASP.NET Core nabízí několik návrhových vzorů. Jsou to:

- ASP.NET Core Web App Rator Pages
- ASP.NET Core Web App MVC (Model-View-Controller)
- ASP.NET Core Web API
- ASP.NET Core s Angular
- ASP.NET Core s React.js
- ASP.NET Core s React.js a Redux (Khalid, 2020)

3.2.2. MVC

Jako návrhový vzor jsem si zvolil MVC (Model-View-Controller), protože odděluje aplikaci do třech hlavních skupin komponentů.

Jedná se o návrhový vzor, který definuje výše zmíněné tři komponenty. V modelu se implementují datové entity a jejich přístupnost. View je odpovědný za informace, které se zobrazují uživateli a controller využívá data modelu a předává je view. Controller je také zodpovědný za obdržení žádosti od prohlížeče k provedení akce a následně na tento požadavek vrací odpověď. Za to, jak bude odpověď vypadat, zodpovídá view. Pomocí jazyka HTML a Razor syntaxe je view definováno a data, která budou zobrazena si controller přebírá z modelu, která jsou finálně pomocí view vykreslena jako výstup. (Nagel, 2018)

Model

Objekty modelu reprezentují data aplikace a obsahují business logiku aplikace. Objekty modelů často načítají a ukládají stav modelu do databáze. (Microsoft, 2020)

View

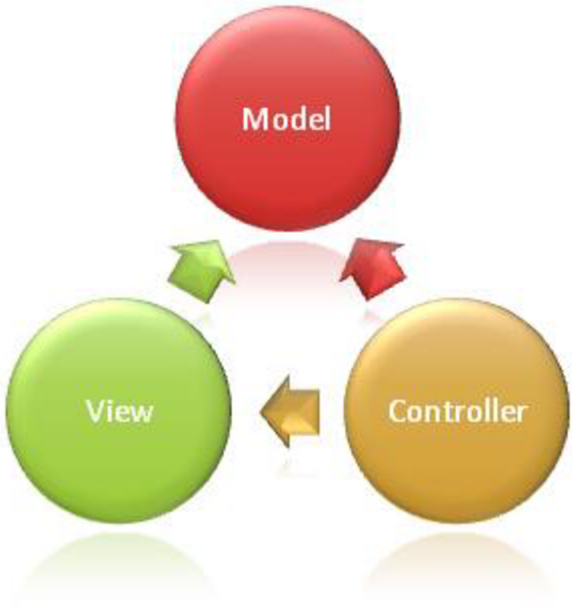
View je zodpovědný za prezentování obsahu uživateli. Využívá se zde Razor syntaxe pro vkládání .NET kódu do HTML prvků. Ve view by se měla nacházet pouze šablona k vykreslení. Jakákoliv logika by zde měla být pouze minimální a pokud se zde nachází, měla by se vztahovat k prezentování vykresleného obsahu. (Smith & Brock, c2022)

Controller

Controller se vypořádává s uživatelskými interakcemi, zpracovává uživatelské požadavky a na základě těchto požadavků vybírá view, který se zobrazí. Dále pracuje s modely. V návrhovém vzoru MVC je controller počátečním vstupním bodem a je zodpovědný, který view se má vykreslit a s jakým modelovým typem pracovat. (Smith & Addie, 2022)

Následující diagram zobrazuje hlavní komponenty architektury MVC:

Obrázek 2 MVC architektura



Zdroj: (Smith, 2022)

3.3 Programovací jazyky

3.3.1 C#

C# je moderní vysokoúrovňový, objektově orientovaný a typově zabezpečený programovací jazyk. Jazyk C# vychází z rodiny C jazyků a je velmi podobný programovacím jazykům C, C++, Java a JavaScript. (A tour of the C# language, 2022)

C# má bohatou implementaci objektově orientovaného paradigmatu. Ta zahrnuje zapouzdření, dědičnost a polymorfismus. Další implementace objektově orientovaného přístupu je jednotný typový systém. Dále C# disponuje třídami a rozhraními. Všechny funkcionality jsou metodami. Dále se zde nacházejí vlastnosti, které zapouzdřují určitý stav objektu a eventy, které působí při změně stavů objektů. Programovací jazyk C# je také silně typově zabezpečený. To znamená, že typy mohou komunikovat skrz protokol pouze tak, jak jsou definovány. (Albahari, 2021)

Jazyk také spoléhá za běhu programu na automatickou správu paměti. Komponent CLR (Common Language Runtime) obsahuje garbage collector a spouští se jako součást programu. Garbage collector uvolňuje paměť pro objekty, na které již není odkazováno. (Albahari, 2021)

C# je možné spustit na platformách Windows, Linux a macOS, Android, iOS a na Windows 10 zařízeních. (Albahari, 2021)

3.3.2 HTML

HTML je značkovací jazyk, který je interpretován webovým prohlížečem. Definuje elementy k zobrazení jako jsou nadpisy, tabulky, seznamy nebo textová pole. (Nagel, 2018)

3.3.3 CSS

CSS neboli zkratka z anglických slov *Cascading Style Sheets*, česky také kaskádové styly. Definují vzhled stránky a vzhled jednotlivých HTML prvků. Pomocí CSS můžeme elementy označovat buď pomocí ID, jména nebo můžeme definovat CSS třídu pro určité HTML značky. (Nagel, 2018)

3.3.4 Razor syntaxe

Razor je syntaxe značek pro vkládání kódu, který obsahuje části kódu založených na frameworku .NET do webových stránek. Skládá se z HTML prvků, Razor značkovače, který označuje C# část kódu. Soubory obsahující Razor mají příponu *.cshtml*. (Anderson et al., 2022)

Razor používá k přechodu na C# znak @. Kde začíná znak @ tam začíná C# kód. Engine automaticky detekuje konec C# kódu tam, kde najde začátek HTML tagu. (Nagel, 2018)

Ukázka kódu 1 Razor syntaxe

```
<h1>@name</h1>
```

Zdroj: Autor

3.4 Entity Framework Core

Entity Framework Core – zkráceně EF Core – je knihovna, kterou vývojáři používají k přístupu k databázím. EF Core je navrhnut jako objektově relační zobrazovač (zkratka O/RM z anglického object-relational mapper). O/RM pracuje mezi dvěma světy, mezi relačními databázemi s vlastním API a objektově orientovaným světem. Hlavním přínosem je, že vývojáři mohou psát kód pro přístup k databázím v jazyce, který mohou znát lépe než jazyk SQL. (Smith, [2021])

Entity Framework Core dovoluje dotazování v těchto různých databázích, zahrnují SQL Server, Oracle, MySQL, PostgreSQL a SQLite. (Albahari, 2021)

3.5 Bootstrap

Bootstrap je open-source frontendový framework, který obsahuje šablony založené na HTML a CSS, pro typografii, formuláře, tlačítka, tabulky a mnoho dalších prvků. (W3schools: Bootstrap 5 Tutorial, c1999-2022)

Protože existují knihovny, které nabízejí předem připravené open source šablony a na oficiálních stránkách Bootstrapu je poskytována open source knihovna s ikonami, rozhodl jsem se zvolit si tento framework jako frontend pro moji aplikaci.

4 Metodika

Pro vývoj webové aplikace jsem se rozhodl využít širokou paletu technologií a služeb, kterou nabízí firma Microsoft. Tyto technologie a služby jsou distribuovány ve formě open-source. Pro vývoj aplikace jsem si vybral framework ASP.NET Core, protože nejvíce zkušeností mám s programovacím jazykem C# a platformou .NET, kterou je ASP.NET Core součástí a zároveň je tento framework napsán v programovacím jazyku C#.

Jako integrované vývojové prostředí jsem si zvolil Microsoft Visual Studio, které nabízí několik předem připravených šablon, jež usnadní vývoj a toto IDE disponuje širokou škálu podpory pro vývoj na tomto frameworku. Jako architektonický vzor jsem se rozhodl vybrat vzor MVC, tedy Model-View-Controller. V pozdější fázi vývoje jsem také dospěl k tomu využít cizí knihovny prostřednictvím balíčků NuGet.

Jelikož jsem potřeboval nějakým způsobem vizualizovat data do grafů, vybral jsem si pro tento účel JavaScriptovou knihovnu Google Charts, která se velice snadno používá. V oficiální dokumentaci je velice podrobně a srozumitelně popsána.

Při řešení přístupu k databázi jsem došel k rozhodnutí, že z původního zamýšleného použití ADO.NET k přístupu k datům bude lepší použít Entity Framework Core, protože nabízí modelování na základě entit a používá větší abstrakci pří vývoji. Zároveň používá komunikační vrstvu ADO.NET data provider pro standartní ADO.NET mechanismy. Toto rozhodnutí se mi při vývoji aplikace osvědčilo a výrazně usnadnilo vývoj.

Výslednou aplikaci bylo potřeba nasadit do reálného prostředí. Opět jsem si vybral technologii od Microsoftu, a to Microsoft Azure, což je cloudový server. Ve Visual Studiu je integrovaný proces k nasazení a publikování, proto výsledné publikování je velice snadné. Zároveň Microsoft nabízí zdarma na dvanáct měsíců používání cloudové databáze Azure SQL Database a volně poskytuje používání služby Azure App Service pro nasazení aplikace.

5 Vývoj aplikace

5.1 Požadavky

Pro tvorbu webové aplikace bylo nejprve potřeba si stanovit jednotlivé požadavky na funkčnost. Konkrétně co přesně webová aplikace má dělat a jak ji budou různí uživatelé používat. Prvním a zásadním požadavkem bylo, aby každý uživatel měl svůj profil, ze kterého bude mít přístup jednak do aplikace a zadruhé si pod tímto účtem mohl přidávat různé typy záznamů do aplikace a následně je zobrazovat. Tímto mi vyvstaly další dva požadavky, a to na ukládání záznamů uživatelem a zobrazování záznamů uživateli. K tomu je dále nezbytné tyto data někam ukládat, tudíž je samozřejmě potřeba databáze, kde se budou tyto data shromažďovat. Konkrétně se jedná o záznamy vylezených cest, uživatelem stanovené cíle a jeho záznamy z tréninků.

V rámci vývoje aplikace jsem se rozhodl omezit na jednu horolezeckou disciplínu, a tou je lezení na umělých stěnách. Nicméně aplikaci lze využívat k vytváření záznamů k více typům horolezeckých disciplín. Po zvážení omezeních v podobě databáze skalních oblastí jsem se zaměřil ve výsledné implementaci aplikace pouze na jednu horolezeckou disciplínu, konkrétně už výše zmíněnou disciplínu lezení na umělých stěnách.

Na českých stěnách je obvyklé používat označení v arabských číslicích, případně se znaménkem zpřesňující obtížnost konkrétní lezecké cesty.

V následujících podkapitolách popisuji jednotlivé stránky a jejich funkce a následně se odkazuji na popis konkrétních implementací.

5.2 Vzhled aplikace

Vzhled neboli také výsledný frontend aplikace jsem vyřešil pomocí implementace open source knihovny Bootswach, která nabízí předem připravené formáty vzhledu stránek a je založena na knihovně Bootstrap. Abych tento vzhled stránky přidal do projektu, vytvořil jsem v adresáři wwwroot/css nový css soubor s tímto souvisejícím kódem. Do hlavičky HTML dokumentu, který je obsažen v částečném zobrazení _Layout.cshtml, jsem dodal tag link k tomuto css souboru. Druhou funkcionalitou, jež jsem přidal pro zpřehlednění a zlepšení vzhledu aplikace, jsou ikony u různých tlačítek. Bootstrap nabízí velkou knihovnu ikon a použití je velice snadné. Pomocí sítě pro doručování obsahu (anglicky též pod zkratkou CDN tedy *content delivery network*) se přidá

tag link a následně se dají tyto ikony používat v zobrazeních. Příklad tagu, který zobrazuje ikonu: `<i class="bi bi-název-ikony"></i>`

5.3 Rozložení webové aplikace

Webovou aplikaci lze rozložit na několik samostatných stránek, které plní různé funkce. Tyto stránky se zobrazují v horní liště. První z nich je domovská stránka, která slouží jako vstupní bod pro uživatele. První stránka s názvem *Přehled* zobrazuje celkový přehled uživatele. Další stránka se nazývá *Přidat záznam*, která slouží jako odesílací formulář k vytvoření záznamu v aplikaci uživatelem. Následující stránkou v liště je zobrazování dat, které souvisí se záznamy konkrétního přihlášeného uživatele. Podobnou funkci má i další stránka, která se v navigační liště nazývá *Záznamy komunity*. Tato stránka ukazuje, podobně jako předešle zmíněná stránka, záznamy s tím rozdílem, že zobrazuje statistiky všech uživatelů, resp. jejich průměry, nejlepší výkon a počet záznamů. Jednou z posledních stránek je *Administrátorský panel*, který je pro běžného uživatele schován a nachází se zde rozcestník pro úpravu webové aplikace, jednoduché psaní článků a editaci uživatelů.

5.4 Domovská stránka

Na domovské stránce se nachází dvě funkce. První z nich je upravovatelná stránka pomocí administrátorského panelu, a tou druhou je výpis posledních publikovaných článků, případně pod tlačítkem *Všechny články* se nachází výpis všech publikovaných článků uložených v aplikaci. Články se přidávají a publikují taktéž v administrátorském panelu.

V adresáři Views jsou uložena všechna zobrazení pro aplikaci. Každá stránka má svůj adresář, v němž se nachází jednotlivé soubory zobrazení s příponou `.cshtml`, ze kterých se jednotlivé stránky skládají.

Domovská stránka má adresář Views/Home, kde se nachází tyto zobrazení:

- `Article.cshtml` – slouží k zobrazení konkrétního článku.
- `Index.cshtml` – zobrazí domovskou stránku s výše zmíněními funkcemi.
- `Privacy.cshtml` – zobrazí stránku se zásadami ochrany osobních údajů.
- `ShowAllArticle.cshtml` – zobrazuje seznam všech publikovaných článků.

5.5 Stránka přehled

Stránka nazvaná *Přehled* slouží jako měsíční přehled pro uživatele. Na této stránce uživatel uvidí seznam jeho lezeckých záznamů, jeho dnešní tréninkovou jednotku, jeho stanovené cíle daného měsíce, kolik má splněno cílů z celkového počtu. Současně je zde liniový graf, který znázorňuje počet splněných cílů v závislosti na času, respektive na měsíční frekvenci. Grafy běží pomocí JavaScriptové knihovny Google Charts. Blíže popisuji tuto implementaci v kapitole 5.17.

V adresáři Views/Overview se nachází jedno zobrazení a to:

- `Index.cshtml` – zobrazuje výše zmíněné funkcionality. Obsahuje jedno částečné zobrazení `_LineChart.cshtml`, které vykresluje výše zmíněný graf. O částečných zobrazeních pojednávám v kapitole 5.12.

5.6 Stránka přidat záznam

Další stránkou je *Přidat záznam*. Přidat záznam může pouze přihlášený uživatel. Záznam se týká lezeckého výkonu. Jedná se o jednoduchý formulář, kam uživatel vloží data a následně je odešle aplikaci.

Toto view se nachází v adresáři Views/AddRecord a obsahuje jedno zobrazení:

- `Index.cshtml` – zobrazí formulář pro vložení dat uživatelem, které souvisí se zdolanou cestou na umělé stěně.

5.7 Stránka list záznamů

Stránka, která je pojmenovaná *List záznamů*, zobrazuje na levé straně tabulku se záznamy právě přihlášeného uživatele. Po pravé straně se uživateli zobrazují dva grafy. První zobrazuje koláčový graf s procentuálním rozložením jednotlivých obtížností, které bere z modelu `Record` a ze sloupce `Difficulty` a následně je upravuje do podoby koláčového grafu. Druhým grafem je graf liniový, který zobrazuje počet záznamů za určitý den. Pod tímto grafem se ještě nachází tabulka s průměrnou obtížností, nejtěžší vylezenou cestou a počtem zdolaných cest. Podle dvou zadaných rozsahů se dynamicky průměr a další ukazatele přepočítávají.

Zobrazení je zde jedno a nachází se v adresáři View/Record:

- `Index.cshtml` – zobrazí tabulku se všemi vloženými záznamy uživatele. Konkrétně jde o záznam `Record`, který je blíže popsán níže v kapitole 5.15.4. Nachází se zde i dvě částečná zobrazení, a to `_PieChart.cshtml` a `_LineChart.cshtml`.

5.8 Stránka globální statistiky

Stránka *Globální statistiky* zobrazuje, podobně jako předešlá stránka *List záznamů,* záznamy uskutečněných lezeckých cest. S rozdílem, že ukazuje data všech uživatelů. Respektive zobrazuje průměr, nejvyšší výsledek a počet záznamů jednotlivých uživatelů. Podobně jako u předešlé stránky, i zde jsou po pravé straně dva grafy, které vykreslují data všech uživatelů. První je koláčový graf a zobrazuje průměry všech uživatelů. Druhý je liniový graf a zobrazuje počet záznamů za určité datum.

Opět se zde nachází jedno zobrazení a nachází se v adresáři Views/Global:

- `Index.cshtml` – zobrazí tabulku se všemi uživateli a jejich statistickými záznamy. Konkrétně jde o záznamy `StatisticData`, které jsou blíže popsány níže v kapitole 5.15.5. Nachází se zde i dvě částečná zobrazení, a to `_PieChart.cshtml` a `_LineChart.cshtml`.

5.9 Stránka plánování cílů

Stránka *Plánování cílů* zobrazuje tabulku s vytvořenými cíli uživatele. V základu se vykresluje aktuální rok a je možné vyfiltrovat celý seznam za předešlé roky. Tabulka obsahuje průměry, nejvyšší plánované obtížnostní výkony a počet plánovaných vylezených lezeckých cest. Také umožňuje tento cíl vytvořit pod tlačítkem *Vytvořit nový cíl*, kde se zobrazí formulář pro vytvoření tohoto cíle.

Zobrazení se nachází v adresáři Views/Goal a obsahuje:

- `Create.cshtml` – zobrazí formulář pro vytvoření nového cíle.
- `Index.cshtml` – zobrazí tabulku s cíli.

5.10 Stránka záznamy tréninků

Na stránce *Záznamy tréninků* je možné pod tlačítkem *Vytvořit novou tréninkovou jednotku,* jak už název napovídá, otevřít formulář k zaznamenání nové tréninkové jednotky. Popis jednotlivých atributů se nachází v kapitole 5.15.7. Rovněž se pod touto

stránkou zobrazí tabulka s těmito tréninkovými jednotkami a u každé je možné konkrétní záznam editovat, nechat si zobrazit detaily a popřípadě vymazat. Každé jednotlivé tlačítko přesune uživatele na nové zobrazení, kde se jednotlivé funkce dají provést. Na stránce je možné mezi tréninkovými jednotkami vyhledávat pomocí textového pole nebo pomocí upřesnění přesného data. Současně lze data filtrovat pomocí tlačítek na liště tabulky podle datumu nebo splněných a nesplněných tréninků.

Zobrazení se nachází v adresáři Views/Training a obsahují následující zobrazení:

- `Create.cshtml` – zobrazí formulář pro vytvoření nové tréninkové jednotky.
- `Delete.cshtml` – umožní vymazat konkrétní tréninkovou jednotku.
- `Details.cshtml` – zobrazí detaily konkrétní tréninkové jednotky.
- `Edit.cshtml` – zobrazí formulář s předvyplněnými daty podle položky, kterou se uživatel rozhodl upravit a umožní tuto položku editovat.
- `Index.cshtml` – zobrazí tabulku se všemi záznamy tréninkových jednotek.

5.11 Stránka administrátorský panel

Na stránce, která je pojmenovaná *Administrátorský panel* a přístup do ní mohou mít pouze uživatelé, kteří mají přiřazené role *SuperAdmin*, *Admin* anebo *Editor*, se nachází seznam odkazů. Prvním odkazem je odkaz *Přidat článek,* kam mají přístup všechny role až na výjimku *User*. Zde je možné vytvářet články, které se zobrazují na úvodní stránce. Další funkcí je pod odkazem s názvem *Zobrazit napsané články a případně je editovat nebo smazat* je možné uživatelem napsané články editovat, publikovat a případně mazat. Třetím tlačítkem je *Editovat úvodní stránku,* kam mají přístup role *SuperAdmin* a *Admin*. Umožnuje pomocí WYSIWYG editoru, který je popsán v kapitole 5.18, upravovat vzhled a informace na úvodní stránce aplikace. Posledním odkazem je *Nastavit role uživatelům*, kde má *SuperAdmin* a *Admin* možnost nastavit role ostatním uživatelům. Nicméně role *Admin* zde má omezení v podobě pomyslné hierarchie, kdy z pochopitelných důvodů nemůže měnit ani přidávat roli *SuperAdmin*.

Zobrazení se nachází v adresáři Views/Admin a obsahuje:

- `CreateArticle.cshtml` – zobrazení slouží pro vytváření nebo úpravu článku. Používá se zde open source WYSIWYG editor Trumbowyg, který rozšiřuje textové pole o funkce WYSIWYG editoru a tato implementace je blíže popsána v 5.18.

- `EditMainPage.cshtml` – zobrazení zobrazí stránku pro editování úvodní stránky neboli Home Page. Opět je zde použit WYSIWYG editor Trumbowyg.
- `ChangeRole.cshtml` – zobrazení slouží pro změnu role konkrétního uživatele.
- `Index.cshtml` – zobrazí rozcestník pro další interakce. Role editor je zde omezena pouze na dvě funkce.
- `ShowArticle.cshtml` – zobrazí tabulku se všemi napsanými články pro konkrétního uživatele. V tabulce je možné hledat podle názvu článku.
- `ShowRoles.cshtml` – zobrazí tabulku se všemi uživateli a jejich přiřazenými rolemi.

5.12 Částečná zobrazení

V projektu používám také několik částečných zobrazení neboli anglicky partial views. Jedná se o soubor značek Razor s příponou `.cshtml`, který vykresluje HTML výstup v rámci vykreslení výstupu jiného souboru značek. Používají se v případě, pokud je potřeba rozdělit obsáhlé zobrazení na několik menších a pomáhají při odstraňování duplicitního kódu. (Smith et al., 2022)

Nachází se v adresáři Views/Shared a projekt jich používá několik:

- `_Layout.cshtml` – je zde použito rozložení vzhledu aplikace pro všechny stránky. A také obsahuje hlavičku HTML dokumentu. Dále je zde definován header pro celou aplikaci a footer.
- `_LineChart.cshtml` – nachází se zde implementace JavaScriptové knihovny Google Charts. Popis této implementace je v kapitole 5.17.
- `_LoginPartial.cshtml` – částečné zobrazení vygenerované a používané pomocí Identity.
- `_PieChart.cshtml` – nachází se zde implementace JavaScriptové knihovny Google Charts.
- `_ValidationScriptsPartial.cshtml` – obsahuje validační scripty ve formě částečného zobrazení.

Další dvě částečná zobrazení jsou v adresáři View a jsou to `_ViewImports.cshtml` a `_ViewStart.cshtml`.

- `_ViewImports.cshtml` – zde se nachází centralizovaný seznam použitých using direktiv v rámci zobrazeních a inicializuji zde přidáním ještě tag helpers.

- `_ViewStart.cshtml` – zde se nachází část kódu, který je potřeba spustit před každým zobrazením. V mém projektu je nastaveno, že pro každé zobrazení se používá částečné zobrazení _Layout. (Smith & Brock, 2022)

5.13 Identity

Z dalších velkých součástí aplikace je registrování nového uživatele, přihlášení stávajícího, přiřazování rolí k uživatelům a dalších pomocných funkcí. To je docíleno pomocí ASP.NET Core Identity, jedná se o API, které podporuje výše zmíněné funkcionality.

Jedním ze způsobů, jak přidat do projektu toto API, je pravým klikem na projekt následně v tomto seznamu vybrat položku *Nová vygenerovaná položka* a zde zvolit *Identita*. Pak se nám zobrazí okno s možnostmi, jež chceme přesně přidat. Podle zaškrtnutých políček se vygeneruje do adresáře Areas podadresář Identity. Vygenerovaný obsah se vygeneruje jako Razor Class Library.

Identity umožňuje také autorizaci na základě rolí. Identita s rolemi se přidává jako rozšiřující služba ve třídě `Program.cs`. Jako metoda `AddRoles<IdentityRole>()`. Celá část kódu, která přidává službu Identity s rozšířením o role a dalším nastavením do projektu, je v ukázce níže.

Ukázka kódu 2 Přidání služby Identity v `Program.cs`

```
builder.Services.AddDefaultIdentity<ApplicationUser>().AddRoles<IdentityRole>()
.AddDefaultTokenProviders().AddEntityFrameworkStores<ApplicationDbContext>();
```

Zdroj: Autor

Seznam všech rolí jsem přidal do statické třídy `Role` a přidal do adresáře Models/StaticData. V aplikaci se nachází čtyři role, a to *SuperAdmin*, *Admin*, *Editor* a *User*. Při registraci nového uživatele je přiřazena role *User* a změnu role lze provést v administrátorském panelu, který je popsán v kapitole výše s názvem Stránka administrátorský panel.

Přidání kontroly rolí se provádí pomocí atributu `Authorize`, který se přidá buď ke controlleru nebo k akční metodě. Níže je ukázka omezení vstupu roli *User* na controlleru `AdminController`.

Ukázka kódu 3 Authorize

```
[Authorize(Roles = Role.RoleAdmin + "," + Role.RoleSuperAdmin + "," + Role.RoleEditor)]
public class AdminController : Controller
```

Zdroj: Autor

5.14 Datový model

V adresáři Models se nachází datový model aplikace. Vytvořil jsem několik EF Core entit, které jsou specifikovány níže.

Entity framework Core podporuje dva vývojové přístupy. První je Code-First a druhý Database-First.

Existuje ještě třetí vývojový přístup, a to Model-first. Ten je ale momentálně podporován pouze v Entity Framework a v budoucnosti Microsoft neplánuje jeho přidání. (Microsoft, 2021)

Code-first přístup, jak už název napovídá, je přístup, při kterém EF Core API vytvoří databázi a relační tabulky na základě migrace a jsou konfigurovány v modelových třídách. (Entity Fremework Core, c2020)

Migrace se vytvoří po zadání příkazu do konzole správce balíčků, a to příkazem `add-migration JmenoMigrace` a databázi je následně potřeba aktualizovat pomocí příkazu update-database. Tento příkaz zajistí, aby databáze byla aktuální a korespondovala s poslední migrací.

Ukázka kódu 4 Příkazy na vytvoření migrace a aktualizace databáze

```
PM> add-migration NovaMigrace
```

```
PM> update-database
```

Zdroj: Autor

V přístupu Database-First EF Core API vytvoří databázi na základě stávající databáze pomocí příkazů EF Core. Tento přístup je značně omezen, protože EF Core nepodporuje visual designer nebo wizard. (Entity Fremework Core, c2020)

5.15 Seznam entit

V projektu se nachází několik databázových entit, které aplikace používá pro svůj výsledný chod. Blíže jsou popsány v kapitolách níže.

Celý databázový model je zobrazen v příloze označenou jako Příloha C.

Na obrázku níže se nachází ukázka implementace entity `StatisticData`.

Ukázka kódu 5 Implementace modelu `StatisticData`

```
public class StatisticData
{
    [Key]
    public string? UserId { get; set; }
    [ForeignKey("UserId")]
    [ValidateNever]
    public ApplicationUser? User { get; set; }

    public string? Average { get; set; }

    public string? Highest { get; set; }

    public int Count { get; set; }
}
```

Zdroj: Autor

5.15.1 ApplicationUser

Tato entita je zde pro případné rozšíření obsahu, který už spravuje třída `IdentityUser`, jež tento model dědí. Také se díky tomuto modelu lépe pracuje při další implementaci, kde je potřeba pracovat s entitou `ApplicationUser`.

5.15.2 Article

V aplikaci entita `Article` se používá pro vytváření a ukládání článků do databáze.

Atribut	Datový typ	Popis
`Id`	`Int`	Primární klíč, slouží k jednoznačné identifikaci článku.
`Title`	`String`	Název článku.
`Content`	`String`	Obsah článku
`Published`	`Bool`	Boolová hodnota, která je defaultně nastavená na false a určuje, jestli se článek zveřejní.
`UserId`	`String`	Cizí klíč, který slouží k identifikaci uživatele, který článek napsal.

5.15.3 MainPage

Entita `MainPage` slouží pro úpravu a uložení vzhledu a informací, které se zobrazují na úvodní stránce.

Atribut	Datový typ	Popis
`Title`	`String`	Primární klíč, slouží k jednoznačné identifikaci `MainPage`.
`Content`	`String`	Obsah `MainPage`, kterou uživatel s oprávněním může nastavit.

5.15.4 Record

Entita `Record` nebo v překladu záznam. Slouží k ukládání dat, které se týkají vylezení cesty uživatelem.

Atribut	Datový typ	Popis
`Id`	`Int`	Primární klíč, slouží k jednoznačné identifikaci záznamu.
`Name`	`String`	Název cesty, kterou uživatel zdolal a zadává jí do systému.
`Difficulty`	`String`	Hodnota, kterou vybírá uživatel. Jedná se o obtížnosti podle klasifikační stupnice UIAA. Vysvětlená v kapitole 2.3.4.
`ModifyDifficulty`	`Int`	Hodnota, která se převádí pomocí třídy `ConverterDifficulty`. Jedná se o pomocnou hodnotu pro výpočet průměrů apod.
`DateRecord`	`DateTime`	Datum, kdy byla cesta zdolána. Tuto hodnotu zadává uživatel. Hodnota je specifikována na formát: dd:mm:yyyy
`UserId`	`String`	Cizí klíč, který slouží k identifikaci uživatele, který záznam vytvořil.

5.15.5 StatisticData

Tato entita slouží pro ukládání statistik jednotlivých uživatelů na základě výpočtu z jejich uložených záznamů. Výpočet probíhá v případě, kdy je přidán nový záznam do tabulky `Record`.

Atribut	Datový typ	Popis
`UserId`	`String`	Primární klíč a zároveň cizí klíč. Je zde použit vztah *one-to-one*. Slouží k jednoznačné identifikaci uživatele.
`Average`	`String`	Průměr uživatele ukládán ve formě `string` podle klasifikační stupnice UIAA (např. „5+”). Vypočítáván z `ModifyDifficulty`.
`Highest`	`String`	Nejvyšší dosažená obtížnost ukládán ve formě `string` podle klasifikační stupnice UIAA (např. „5+”). Získáván z `ModifyDifficulty`.
`Count`	`Int`	Počet celkových záznamů, které uživatel zadal do aplikace.

5.15.6 Goal

Entita `Goal` se používá pro vytváření cílů uživatele. Uživateli nechávám volnost při vytváření těchto cílů. Kontrola atributu `Achieved` pro splnění cíle probíhá vždy začátkem nového měsíce.

Atribut	Datový typ	Popis
`Id`	`Int`	Primární klíč. Slouží k jednoznačné identifikaci cíle.
`Average`	`String`	Průměr uživatele ukládán ve formě `string` podle klasifikační stupnice UIAA (např. „5+”). Hodnota získávána vložením uživatele.

`Highest`	`String`	Nejvyšší dosažená obtížnost ukládána ve formě `string` podle klasifikační stupnice UIAA (např. „5+”).
`Count`	`Int`	Počet záznamů, které chce mít uživatel minimálně vloženy na konci měsíce.
`Month`	`DateTime`	Slouží k označení měsíce ve, kterém se daný cíl má splnit.
`Achieved`	`Bool`	Označuje, zda je daný cíl splněn či nikoliv.
`UserId`	`String`	Cizí klíč sloužící k identifikaci uživatele, který cíl vytvořil.

5.15.7 Training

Tato entita se používá pro vytváření tréninkových jednotek uživatelem. Opět zde nechávám uživateli volnost při vytváření těchto jednotek.

Atribut	**Datový typ**	**Popis**
`Id`	`Int`	Primární klíč. Slouží k jednoznačné identifikaci tréninkové jednotky.
`Date`	`DateTime`	Označuje, kdy uživatel plánuje danou tréninkovou jednotku vykonat (popřípadě, kdy byla tréninková jednotka uskutečněna).

`Focus`	`String`	Označuje zaměření tréninkové jednotky. Podle Tefelnera existuje pět oblastí, které se dají v lezení trénovat. Jsou to: • Síla • Vytrvalostní aerobní (AE) trénink • Vytrvalostní anaerobní (AN) trénink • Vytrvalostní anaerobní – aerobní (AN/AE) trénink • Všeobecně vytrvalostní trénink (Tefelner, 1999) Jsou předány v konstruktéru třídy pomocí proměnné `ChooseFocus` typu `List<SelectListItem>`.
`Description`	`String`	Uživateli umožňuje přidat konkrétní popis tréninkové jednotky.
`Time`	`DateTime`	Uživateli dovoluje přidat dobu trvání tréninkové jednotky.
`Note`	`String`	Možnost přidání poznámky k tréninkové jednotce.
`IsDone`	`Bool`	Označuje, pokud danou tréninkovou jednotku uživatel splnil či nikoliv.

`UserId`	`String`	Cizí klíč sloužící k identifikaci uživatele, který tréninkovou jednotku vytvořil.

5.16 ViewModels

Existují tři různá řešení, jak dostat informace z controlleru do view. První jako *strongly typed model* objekt. Druhá metoda je pomocí typu *dynamic* a třetí za použití *ViewBag*. (Anderson, 2020)

Zvolil jsem si řešení *strongly typed view*, kdy bylo potřeba si v podadresáři Models/ViewModels vytvořit modely, které pomocí klíčového slova se syntaxí Razor a názvem tohoto ViewModelu přidávám na začátek komponentu zobrazení. Příklad přidání řádku s inicializací `ViewModelu: @model JménoViewModelu`.

Popisy těchto modelů, které aplikace využívá, jsou následující:

`ErrorViewModel` – Používá se v zobrazení `Error.cshtml`, které se nachází v adresáři Views/Shared a je vygenerováno při založení projektu. Obsahuje tyto datové typy:

- `RequestId - string`
- `ShowRequestId - bool`

`GlobalViewModel` – Používá se v zobrazení `Index.cshtml`, které se nachází v adresáři Views/Global. Používá tyto datové typy:

- `StatisticData - IEnumerable<StatisticData>` - data obsahují položky pro tabulku.
- `AvarageAndCount - Dictionary<string, int>` - tato proměnná předává zformátovaná data do částečného zobrazení, které má za úkol vygenerovat koláčový graf.
- `DatesAndCounts - Dictionary<string, int>` - tato proměnná předává zformátovaná data do částečného zobrazení, které má za úkol vygenerovat liniový graf.

`MainPageViewModel` – Používá se v zobrazení `Index.cshtml`, které vykresluje úvodní stránku. Obsahuje tyto datové typy:

- `MainPage - MainPage` – obsahuje data z databáze pro zobrazení upraveného vzhledu úvodní stránky.
- `Articles - IEnumerable<Article>` – obsahuje data pro zobrazení názvu článků na hlavní stránce.

`OverviewViewModel`– Používá se v zobrazení `Index.cshtml` uloženým v adresáři Overview pro vykreslení přehledu. Obsahuje tyto datové typy:

- `public IEnumerable<Record> Records` – obsahuje data z entity `Record` pro zobrazení tabulky s těmito daty.
- `public IEnumerable<Training> Training` – používá se pro vykreslení tabulky s dnešními tréninkovými jednotkami.
- `public IEnumerable<Goal> Goal` – předává data pro vykreslení cílů v aktuálním měsíci.
- `public int DoneGoals` – používá se pro vykreslení všech splněných cílů.
- `public int AllGoals` – používá se pro vykreslení všech uložených cílů.
- `public Dictionary<string, int> GoalMonthAndCount` – používá se pro předání dat do liniového grafu.

`RecordViewModel` – Tento ViewModel je určen pro zobrazení `Index.cshtml` v adresáři Record.

- `Record - IEnumerable<Record>` – předávaná data se používají pro vykreslení tabulky.
- `DifficultyAndCounts - Dictionary<string, int>` – používá se pro předání dat do koláčového grafu.
- `DatesAndCounts - Dictionary<string, int>` – data se předávají do liniového grafu.
- `Statistics - IEnumerable<StatisticData>` – používá se pro tabulku obsahující individuální statistiku.

`ShowRolesViewModel` – Používá se v zobrazení `ShowRoles.cshtml` a `ChangeRole.cshtml`.

- `UserId - String` – předává identifikaci uživatele do tagu <a> přesněji v atributu `asp-route-userId`, kde toto `userId` pomáhá při výběru uživatele, nad kterým se má spustit zvolená operace. Buď změnit roli nebo vymazat.
- `UserName - String` – používá se pro vykreslení jména uživatele.

- `Roles - IEnumerable<string>` – používá se pro zobrazení momentálně nastavené role uživatele.
- `ChooseRole - List<SelectListItem>` – používá se pro příkaz `@Html.DropDownListFor`, který vykreslí nabídku s více možnostmi. Nastavení hodnot probíhá v konstruktoru této třídy.

5.17 Google Charts

Pro vizualizaci dat do grafů využívám JavaScriptovou knihovnu Google Charts, která poskytuje několik různých variant typů grafů, jak vizualizovat data. V projektu jsem si vystačil s koláčovým a liniovým grafem.

V oficiální dokumentaci je tato knihovna popsána jako způsob, jak vizualizovat data na webové stránce. Můžeme zde použít jednoduché grafy či i více komplexní grafy díky velké paletě již předem připravených grafů. Tyto grafy jsou navrženy jako třídy JavaScriptu a jde jim dále měnit vzhled podle aktuální potřeby. Grafy jsou vykreslovány pomocí technologie HTML5/SVG, tím je zajištěna kompatibilita mezi prohlížeči a přenositelnost mezi různými platformami. Všechny typy grafů se naplňují daty pomocí třídy DataTable, což usnadňuje vývoj. Tato třída poskytuje také metody pro třídění, úpravu a filtrování dat a lze ji naplnit přímo z webové stránky, databáze nebo jakéhokoliv poskytovatele dat podporujícího protokol Chart Tools Datasource. (Google Charts, 2019)

Implementaci této knihovny řeším ve dvou částečných zobrazeních, a to v `_PieChart.cshtml` a `_LineChart.cshtml`, kde nejprve pro jednotlivé grafy inicializuji použití knihovny Google Charts pomocí CDN a následně inicializuji načtení vizualizace konkrétního typu grafu. V následujících řádcích kódu přidávám do proměnné `data` pomocí foreach cyklu data, která předávám tomuto zobrazení pomocí modelu typu `Dictionary<string, int>`. Dalším krokem je vytvoření proměnné `options` s nastavením grafu. Na konci provádím deklaraci instance `chart` a na této instanci volám metodu `draw`, kde do argumentů předávám proměnnou s názvem `data` s daty a druhou proměnou `options` s nastavením grafu. Na závěr mimo tag `<script>` je potřeba přidat značku `<div>` s `id`, které je dříve uvedeno při vytvoření instance `chart`.

Oba grafy se iniciují v mé implementaci poměrně stejně až s rozdílem při načtení konkrétního grafu. Další rozdíl je v nastavení tohoto grafu v proměnné `options` a

při vytvoření nové instance `chart`, kdy je potřeba volat správnou metodu s názvem již načteného grafu.

Ukázka kódu 6 Inicializace proměnné chart

```
        var chart = new google.visualization.PieChart(document.getElementById('pie_chart_div'));
        chart.draw(data, options);
        }
</script>

<div id="pie_chart_div"></div>
```

Zdroj: Autor

5.18 WYSIWYG editor

WYSIWYG je akronym anglické věty *What you see is what you get* (česky *Co vidíš to dostaneš*). Jedná se o způsob editace, při kterém podoba dokumentu je stejná jako vzhled při úpravě. Tento editor přidávám v aplikaci, kde uživatel potřebuje mít v textovém poli více formátovacích funkcí. Využívám pro tento účel open source WYSIWYG editor Trumbowyg, který již obsahuje předem připravenou českou lokalizaci. Implementace do projektu je velice jednoduchá, stačí na webových stránkách (https://alex-d.github.io/Trumbowyg/documentation/) stáhnout balíček, ten následně přidat do adresáře wwwroot/lib.

Následná implementace, která je potřeba vytvořit, je na ukázce kódu níže:

Ukázka kódu 7 Implementace editoru Trumbowyg

```
@section Scripts {
    <script src="~/lib/Trumbowyg-main/dist/trumbowyg.js"></script>
    <script type="text/javascript" src="~/lib/Trumbowyg-main/dist/langs/cs.js"></script>

    <script>
        $('#Description').trumbowyg({
    lang: 'cs'
    });
    </script>
}
```

Zdroj: Autor

5.19 Návrhový vzor Repository a Unit of Work

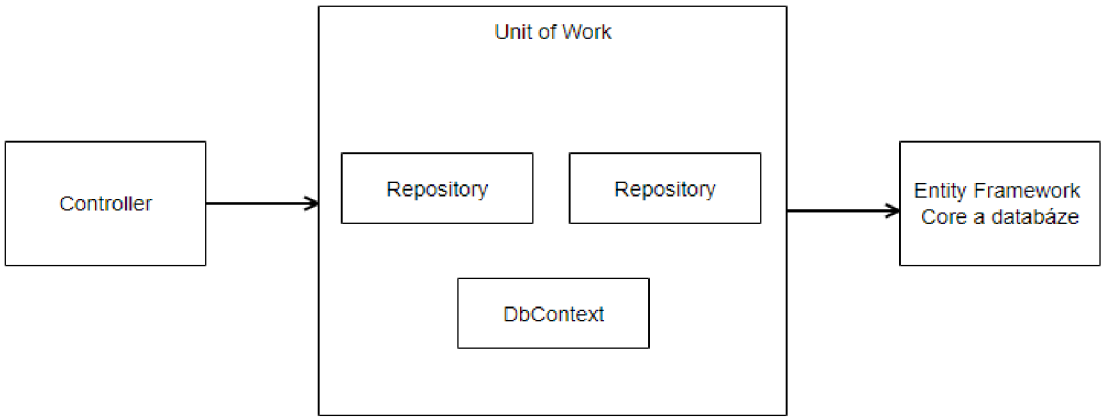
V implementaci využívám návrhový vzor Repository, který zprostředkovává vrstvu mezi aplikací a logikou k přístupu k datům. To znamená, že Repository pattern izoluje veškerý přístupový kód k datům od zbytku aplikace. Výhodou je, že pokud

potřebujeme v budoucnu provést nějakou změnu, uděláme to na jednom místě. (Repository Pattern In C#, 2021)

Unit of Work je spojen se zefektivněním implementace Repository vzoru. Používá se k seskupení jedné nebo více operací do jedné transakce nebo takzvané jednotky práce (Unit of Work). Tím je zajištěno, že všechny operace buď projdou nebo selžou jako jedna jednotka. Unit of Work také obsahuje metodu SaveChanges na jedné DbContext instanci. To znamená, že pokud dvě entity používají stejnou instanci a jedna nebude fungovat, druhá se také neuloží. Tak je zabezpečena konzistence databáze. Tudíž, když se metoda SaveChanges zavolá, bude provedena pro obě úložiště. (Dot Net Tutorials, n. d.)

Na obrázku níže můžeme vidět ukázku, jak controller komunikuje s Unit of Work a Unit of Work s Entity Framework Core a s databází.

Obrázek 3 Repository pattern s Unit of Work



Zdroj: Autor, upraveno podle (Dykstra, 2021)

Na výše zobrazeném obrázku je vidět, že controller nebude komunikovat s třídou Entity Framework přímo, ale prostřednictvím vrstvy repository. Konkrétně v tomto případě s Unit of Work a následně s Entity Framework Core a databází.

Výhody tohoto vzoru jsou:

- Jak už je zmíněno výše, přístupový kód k databázi je na jednom místě, a tudíž je lépe spravován a lze snadno implementovat jakékoliv změny.
- Poskytuje flexibilní architekturu.

- Testování controllerů se stává snadnější, protože testovací framework nemusí běžet na skutečném kódu k přístupu k databázi.
- Jedním z nejdůležitějších aspektů je oddělení mezi aktuální databází, dotazy a ostatní datově přístupovou logikou od zbytku aplikace. (Repository Pattern In C#, 2021)

Implementace návrhového vzoru Repository se nachází v adresáři Repository spolu s implementací návrhového vzoru Unit of Work, který zastřešuje všechny funkce Repository do kompaktního celku. V implementaci se používá velice jednoduše přes proměnnou typu třídu `UnitOfWork`.

5.20 Controller

Každá stránka má vlastní controller, který se stará o interakci uživatelů, pracuje s modelem, a nakonec vykreslí view, které se na závěr procesu zobrazí uživateli. Controllery jsou uloženy v adresáři Controllers. Každý controller obsahuje metody, jejichž názvy korespondují s názvy zobrazeními. To zaručí, že nemusíme v implementaci metody vracet metodu `View()` s argumentem k příslušným zobrazením a vše vyřeší sám controller. Tím je také výrazně zpřehledněn kód. Nicméně v mém projektu existují také případy, kde jsem se rozhodl pro opětovné použití zobrazení. Jako argument v těchto situacích bylo potřeba předat název zobrazení ve formě `string`.

Microsoft ve své dokumentaci uvádí, že controller by neměl být příliš komplikovaný a mít mnoho povinností, aby se logika controlleru nestala nepřehlednou a příliš komplexní. Pokud by se toto stalo, měla by se tato logika vytlačit pryč z controlleru. (Smith, 2022) Zde se v tomto doporučení mírně rozcházím, nicméně přesto si myslím, že ve výsledné implementaci se mi povedlo dosáhnout toho, že controllery jsou přehledné, nejsou příliš složité a nemají mnoho povinností.

Každý controller obsahuje několik metod. Některé z nich popíši níže. Všechny controllery obsahují deklarovaný konstruktor, kde inicializuji proměnnou `_unitOfWork`, která v sobě obsahuje všechny potřebné metody pro získání dat z databáze.

V projektu se nachází tyto controllery:

- `AddRecordController`
- `AddRecordController`
- `GlobalController`

- `GoalController`
- `HomeController`
- `OverviewController`
- `TrainingController`

Každý se stará o jednu stránku a v nich obsažené metody korespondují s vytvořenými views v jednotlivých adresářích s shodují se s názvy, které slouží pro zobrazování obsahu uživateli. Jednotlivé metody obsahují implementace pro CRUD operace (z anglické zkratky create, read, edit a delete tedy vytvořit, číst, editovat a smazat). Pokud nebyly některé operace potřebné, jsou vynechány. V některých controllerech dochází k výpočtům dat.

Níže popíši dva vybrané controllery, konkrétně RecordController a TrainingController, na kterých blíže objasním jednotlivé metody. Ostatní controllery jsou velmi podobné či implementují některé další funkce. Lze si je podrobněji prohlédnout v kódu aplikace, nacházející se v příloze A.

`RecordController` – Stará se o stránku List záznamů, kde zobrazuje data získaná z databáze. Podobný controller je `GlobalController`, který se odlišuje pouze ve výsledné implementaci získávaných dat.

- `public IActionResult Index()` – na základě Id uživatele přiřadí data z databáze do instance třídy `RecodViewModel`, kterou vrací jako argument v metodě `View()`.
- `private Dictionary<string, int> GetDataToPieChart()` – privátní metoda, jež se volá v metodě `Index()`. Získává a formátuje data pro koláčový graf.
- `private Dictionary<string, int> GetDataToLineChart()` – privátní metoda, která se volá v metodě `Index()`. Získává a formátuje data pro liniový graf.

`TrainingController` – Stará se o stránku *Záznamy tréninků*, kde zobrazuje data získaná z databáze. Obsahuje všechny CRUD operace.

- `public IActionResult Index(string sortOrderDate, string sortOrderIsDone, string search, DateTime searchDate)` – na základě vložených argumentů od uživatele upraví data získaná z databáze anebo je zobrazí v základním nastavení a doručí je do zobrazení.
- `public IActionResult Details(int? id)` – na základě id získá konkrétní detail tréninku pro vykreslení.

- `public IActionResult Create()` – zobrazí view `Create.cshtml`.
- `public IActionResult Create([Bind("Id,Date,Focus,Description,Time,Note,IsDone,UserId")] Training training)` – vytvoří nový záznam na základě vložených dat.
- `public IActionResult Edit(int? id)` – vykreslí zobrazení `Edit.cshtml` s předpřipraveným formulářem.
- `public IActionResult Edit(int id, [Bind("Id,Date,Focus,Description,Time,Note,IsDone,UserId")] Training training)` – upraví záznam na základě změn zadané uživatelem.
- `public IActionResult Delete(int? id)` – na základě id vykreslí zobrazení `Delete.cshtml`, které obsahuje data na podkladě id tohoto záznamu a rovněž nabízí tlačítko pro vymazání.
- `public IActionResult DeleteConfirmed(int id)` – odstraní záznam a vrátí uživatele zpět na `Index.cshtml`.
- `private bool TrainingExists(int id)` – pomocná privátní metoda, která kontroluje existenci záznamu s id.

5.21 Pomocné třídy

V adresáři Utility se nachází dvě třídy sloužící jako pomocné třídy pro zpřehlednění projektu a pomoci při implementaci jiných metod. Jsou to tyto dvě třídy:

První třída z nich je `ConverterDifficulty.cs`, jež je řešena jako statická třída. Obsahuje dvě metody, a to `GetIntFromDifficultyString()`, která jako argument potřebuje `string difficulty` a vrací `int`. Funkcí této metody je přeměna stringové podoby lezecké obtížnosti na číselnou stupnici začínající od jedničky. Tato konverze slouží pro další výpočty, a to hlavně průměru lezecké obtížnosti uživatelů. Druhou metodou je `GetStringFromDifficultyInt()`. Jedná se o protipólnou metodu k prvně zmíněné metodě. Jako argument potřebuje `int`, který představuje už konvertovanou lezeckou stupnici v celých číslech a vrací odpovídající `string` v podobě UIAA stupnice. Tyto metody jsou používány tam, kde je zapotřebí přeměnit stringovou podobu stupnice UIAA se znaménkem + nebo - (např. „4+“) na celočíselnou hodnotu. Metoda `GetStringFromDifficultyInt()` zajišťuje opačný proces pro přeměnu číselné hodnoty zpět na stringovou podobu. Celočíselná hodnota je dále používána při výpočtech.

Druhou třídou je `EmailSender.cs`, který obsahuje jedinou metodu, a to `SendEmailAsync()`. V metodě je implementováno zasílání emailů uživatelům. Pro své fungování využívá dvě knihovny, jež jsou implementované pomocí balíčků NuGet, a to MailKit a MimeKit. Blíže jsou tyto knihovny popsány v kapitole 5.22, kde uvádím všechna přidaná rozšíření do projektu. Tato metoda je používána ve více třídách. První z nich je třída `Register.cshtml.cs`, která doručuje email nově registrovanému uživateli o potvrzení účtu a ve třídě `ForgotPassword.cshtml.cs` odesílá email při změně hesla, také implementuje tuto funkci ve třídě `Email.cshtml.cs`.

5.22 NuGet balíčky

V projektu využívám několik cizích knihoven a oficiální knihovny vytvořené Microsoftem.

Knihovny od Microsoftu jsou následující:

- **Microsoft.AspNetCore.Identity.EntityFrameworkCore** – přidává EF Core popsaný v předešlých kapitolách.
- **Microsoft.AspNetCore.Identity.UI** – výchozí integrované uživatelské rozhraní Razor Pages.
- **Microsoft.EntityFrameworkCore.SqlServer** – poskytovatel databáze Microsoft SQL Server pro Entity Framework Core. Tato databáze byla použita při vývoji aplikace.
- **Microsoft.EntityFrameworkCore.Tools** – základní nástroje pro EF Core pro konzoli NuGet Packet Manager. (Např. výše zmíněné Add-Migration, Update-Database)
- **Microsoft.Extensions.Identity.Core** – systém pro vytváření přihlašování a rolí.
- **Microsoft.Extensions.Identity.Stores** – systém pro vytváření přihlašování a rolí.
- **Microsoft.VisualStudio.Web.CodeGeneration.Design** – nástroj pro generování kódu.

Seznam knihoven třetích stran:

- **ExpressiveAnnotations** – je malá .NET a JavaScriptová knihovna, která poskytuje úplné na anotacích založené mechanismy podmíněného ověřování entit. (Nuget.org, 2022)

- **MailKit** – jedná se o open source multiplatformní mail-klient knihovnu založenou na .NET, založenou na MimeKit a také optimalizovanou pro mobilní zařízení. (Nuget.org, 2022)
- **MimeKit** – je C# knihovna, která umožňuje použití pro vytváření a parsování zpráv pomocí Multipurpose Internet Mail Extension (MIME), jak je definováno v specifikacích IETF. (Nuget.org, 2022)

5.23 Nasazení

Pro nasazení v reálném prostředí jsem zvolil řešení od Microsoftu, a to cloudovou službu Azure.

Azure je veřejná cloudová platforma od společnosti Microsoft. Využívá se k vytváření, hostování a škálování webových aplikací prostřednictvím datových center Microsoftu. Microsoft Azure obsahuje tři hlavní části, a to výpočetní část, úložiště a strukturní část. (GEM System, c2020)

Pro mé potřeby nasazení aplikace jsem využil služby Azure App Service a Azure SQL Database. První zmíněná služba je k dispozici zdarma s omezením na 10 webových aplikací s 1 GB úložištěm. Druhá služba Azure SQL Database je dostupná po 12 měsíců zdarma s omezením na 250 GB. (Azure, c2022)

Publikování aplikace je poměrně jednoduchý proces přímo ve Visual Studiu v záložce S*estavit*, kde se nachází tlačítko *Publikovat*. K tomuto kroku je nutné mít ověřený účet u Microsoftu a pod tímto účtem se přihlásit. Po výběru možnosti *Publikovat* se zobrazí okno pro výběr cíle. Zde je na výběr z několika možností. Cílím na Azure, proto volím tuto možnost. V dalším okně je potřeba vybrat konkrétní cíl. Zvolil jsem si zde Azure App Service (Windows), neboť tato služba umožňuje publikovat kód aplikace do spravované infrastruktury. Dále je nezbytné zvolit konkrétní instanci služby app service, kde bude aplikace nasazena, současně je zde možné si novou instanci vytvořit.

Rovněž bylo nutné vytvořit novou službu, kterou je cloudová databáze. Ta lze přidat v sekci *závislosti služby*, kde jsem udělal novou závislost, a to na službě Azure SQL Database. Následně bylo nezbytné povolit pro tuto databázi migraci při publikování, aby se na základě definované migrace vytvořil databázový model odpovídající mému navrženému modelu. Toto nastavení lze vybrat po zvolení tlačítka s ikonkou pera

a v novém okně pod názvem *Migrace rozhraní Entity Framework* se zaškrtne nastavení *Použít tuto migraci při publikování*.

Při použití tlačítka *Publikovat* se vytvoří build aplikace a podle nastavení dojde k nasazení aplikace do reálného prostředí. Pokud je nutné nasadit novou aktuálnější verzi aplikace, stačí použít znovu tlačítko *Publikovat* a nasadí se tato aktuálnější verze aplikace. Po tomto procesu je webová aplikace dostupná na této webové adrese: https://lezeckydenik.azurewebsites.net/.

6 Testování

Testování aplikace probíhalo ve dvou fázích. V obou případech bylo použito manuální testování. První fáze se uskutečnila současně během vývoje na lokálním zařízení, kdy jsem po implementaci konkrétní stránky otestoval její funkcionalitu a následně odstranil chyby.

Druhá fáze probíhala po nasazení aplikace do reálného prostředí na cloud Azure, kde se tester držel testovacích scénářů. Tyto scénáře vycházejí z požadavků na aplikaci. Nachází se v příloze B. Pomocí těchto scénářů byla otestována finální funkčnost aplikace. Případně nalezené chyby byla následně opraveny.

7 Závěr

Ve své bakalářské práci se zabývám vývojem webové aplikace pro evidenci lezeckých záznamů. V teoretické části práce se zaobírám vymezením pojmů webové aplikace, databáze, horolezectví, respektive pojmu sportovní lezení a definuji stavbu a organizaci lezeckých tréninků. Na základě těchto informací navrhuji jednotlivé funkce ve webové aplikaci. V další kapitole popisuji použité nástroje a technologie. Těmi jsou použité integrované vývojové prostředí Microsoft Visual Studio, technologie pro vývoj aplikace ASP.NET Core, návrhové vzory a použitý návrhový vzor MVC. V dalších kapitolách popisuji využité programovací jazyky, použitý objektově-relační mapovač Entity Framework Core pro přístup k datům do databáze a open-source knihovnu Bootstrap.

V praktické části se věnuji popisu požadavků na aplikaci, popisu vývoje vzhledu aplikace neboli frontendu aplikace. Dále charakterizuji výsledné rozložení stránek webové aplikace jednotlivých funkcionalit a zobrazení, které se výsledně vykreslují uživateli. Následně popisuji datový model, implementaci controllerů a dalších implementací použitých technologií pro vývoj této webové aplikace.

Cílem práce bylo vytvořit webovou aplikaci, která umožní uživatelům evidovat data, která souvisí se sportovním lezením. Tyto data dává možnost zobrazovat a případně evidovat. Dále aplikace dovoluje uživatelům založit účet, pod kterým se přihlašují do aplikace a vkládají svá data. Současně aplikace umí k těmto účtům přidávat role a případně je podle potřeby měnit.

Vyvinul a navrhnul jsem webovou aplikaci založenou na technologii ASP.NET Core a dalších technologiích, které jsou popsány v této práci. Využil jsem integrovaného vývojového prostředí Visual Studio. Aplikace umožňuje registraci uživatelů a následné přihlášení pod tímto vytvořeným účtem. Webová aplikace umí uživatelům přidávat role k těmto účtům a případně je možné oprávněným uživatelům tyto role měnit. Uživatelé si mohou vést svůj lezecký deník, který je primárně zaměřen na lezeckou disciplínu lezení na umělých stěnách. Nicméně s jistým omezením lze aplikaci použít i na evidenci záznamů u dalších lezeckých disciplín, jako je skalní horolezectví a specifický druh pískovcové horolezení. Tento deník je rozdělen na několik částí. První část je zaměřená na zaznamenávání uskutečněných lezeckých cest a následné zobrazování těchto dat pro individuálního uživatele nebo jako souhrn všech uživatelů. Druhou částí je

plánování cílů a třetí seskupení dat je zaměřeno na plánování či evidenci záznamů z tréninkových jednotek. Webová aplikace nabízí celkový přehled, který zobrazuje vybraná statistická data. V jednotlivých částech je možné tyto data filtrovat a vyhledávat. Webová aplikace má také pro ukázku jednotlivých rolí dvě funkce, a to úpravu vzhledu domovské stránky a přidávání článků. Tyto funkce lze používat v *Administrátorském panelu*, kam je omezen přístup pro jednotlivé role. Rovněž je zde možnost měnit role uživatelům.

Výsledná webová aplikace je publikovaná na cloudovou službu Microsoft Azure a je dostupná na webové adrese: https://lezeckydenik.azurewebsites.net/

I. Summary and keywords

This paper describes a design and development of a web application for creating and saving climbing records. Also, users of this application are allowed to have different roles, which allows users to be admins and editors, who has opportunity to use the application in different way, create their own goals, training units and see their overviews based on records. For development of this web application were used IDE Visual Studio. On top of that, it is based on ASP.NET Core technology and deployed on Microsoft Azure cloud service. The theoretical part mainly describes web application in general, databases and developer tools. In the second practical part were described requirements for the web application, front-end of the application. Furthermore, were described overall layouts of the web application and functionality of individual pages. As next, were described data model and design of the controllers. In the end of this chapter were described how the web application is published to cloud service from Microsoft, which is called Microsoft Azure in IDE Visual Studio.

Key words: web application, ASP.NET Core, database, MVC, C#, climbing

II. Seznam použitých zdrojů

Albahari, J. (2021). *C# 9.0 in a nutshell: the definitive reference*. O'Reilly.

Anderson, R. (2020). *Dynamic v. Strongly Typed Views*. Microsoft Technical documentation. Retrieved April 1, 2022, from https://docs.microsoft.com/en-us/aspnet/mvc/overview/views/dynamic-v-strongly-typed-views

Anderson, R., Mullen, T., & Vicarel, D. (2022). *Razor syntax reference for ASP.NET Core*. Microsoft Technical documentation. Retrieved April 1, 2022, from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-6.0

A tour of the C# language. (2022). Microsoft Technical documentation. Retrieved April 1, 2022, from https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Azure. (c2022). Retrieved April 2, 2022, from https://azure.microsoft.com/cs-cz/free/?ezid=WisA_FQ_Try

Dot Net Tutorials. (n. d.). Retrieved April 2, 2022, from https://dotnettutorials.net/lesson/unit-of-work-csharp-mvc/

Dykstra, T. (2021). *Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application*. Microsoft Technical documentation. Retrieved April 2, 2022, from https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

Entity Fremework Core. (c2020). Entity Framework Tutorial. Retrieved April 1, 2022, from https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx

Garcia-Molina, H., D. Ullman, J., & Widom, J. (2009). *Database Systems: The Complete Book* (Second Edition). Pearson Education.

GEM System. (c2020). Retrieved April 2, 2022, from https://www.gemsystem.cz/reseni-a-sluzby/integrace-a-vyvoj/poznejte-microsoft-azure-platform-a-jeho-moznosti-pro-vas-business/

G. Lee, T. (2022). *Welcome to the Visual Studio IDE*. Microsoft: Documentation. Retrieved February 10, 2022, from https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022#editions

Google Charts. (2019). Retrieved April 2, 2022, from https://developers.google.com/chart/interactive/docs

Jak na internet. (c2022). Retrieved March 31, 2022, from https://www.jaknainternet.cz/page/1262/webove-aplikace/

Khalid, A. (2020). *ASP.NET Core 2 - Architecture And Design Pattern Ideology*. C# Corner. Retrieved February 10, 2022, from https://www.c-sharpcorner.com/article/asp-net-core-2-architecture-design-pattern-ideology/

Kublák, T. (2014). *Horolezecká metodika - 1. díl - Základy*. Tomas Kublak - MMPublishing. https://play.google.com/store/books/details?id=2U7vBQAAQBAJ

Lock, A. (2018). *ASP.NET Core in Action*. Manning Publications Co.

Microsoft. (2020). *ASP.NET MVC Overview*. Microsoft Technical documentation. Retrieved February 10, 2022, from https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview

Microsoft. (2021). *Compare EF Core & EF6*. Microsoft Technical documentation. Retrieved April 1, 2022, from https://docs.microsoft.com/cs-cz/ef/efcore-and-ef6/

Nagel, C. (2018). *Professional C# 7 and .NET Core 2.0*. Wrox, John Wiley.

Nuget. (2022). Retrieved April 1, 2022, from https://www.nuget.org/packages/ExpressiveAnnotations/

Nuget. (2022). Retrieved April 1, 2022, from https://www.nuget.org/packages/MailKit/

Nuget. (2022). Retrieved April 1, 2022, from https://www.nuget.org/packages/MimeKit/

Repository Pattern In C#. (2021). Retrieved April 1, 2022, from https://www.linkedin.com/pulse/repository-pattern-c-pawan-verma

Roth, D. (2021). *Announcing ASP.NET Core in .NET 6*. Microsoft .NET Blog. Retrieved February 10, 2022, from https://devblogs.microsoft.com/dotnet/announcing-asp-net-core-in-net-6/

Repository Pattern In C#. (2021). Retrieved April 1, 2022, from https://www.linkedin.com/pulse/repository-pattern-c-pawan-verma

Roth, D., Anderson, R., & Luttin, S. (2022). *Overview to ASP.NET Core*. Microsoft Technical documentation. Retrieved February 10, 2022, from https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

Silberschatz, A., Korth, H. F., & Sudarshan, S. (c 2006). *Database system concepts* (5. ed). McGraw-Hill Higher Education.

Smith, J. P. ([2021]). *Entity Framework Core in action* (Second edition). Manning.

Smith, S., & Addie, S. (2022). *Handle requests with controllers in ASP.NET Core MVC: What is a Controller?*. Microsoft Technical documentation. Retrieved February 10, 2022, from https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/actions?view=aspnetcore-6.0

Smith, S., & Brock, D. (c2022). *Views in ASP.NET Core MVC*. Microsoft Technical documentation. Retrieved February 10, 2022, from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-6.0

Smith, S., & Brock, D. (2022). *Layout in ASP.NET Core*. Microsoft Technical documentation. Retrieved April 1, 2022, from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/layout?view=aspnetcore-6.0

Smith, S., Jendoubi, M., Anderson, R., & Sauber, S. (2022). *Partial views in ASP.NET Core*. Microsoft Technical documentation. Retrieved March 31, 2022, from https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial?view=aspnetcore-6.0

Tefelner, R. (1999). *TRÉNINK SPORTOVNÍHO LEZCE*. Rudolf Tefelner, Lelekovice 382, 664 31.

TechTarget: Web application (Web app). (c2006-2022). Retrieved March 31, 2022, from https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

V., P. (2021). *Repository Pattern In C#*. LinkedIn. Retrieved April 1, 2022, from https://www.linkedin.com/pulse/repository-pattern-c-pawan-verma

W3schools: Bootstrap 5 Tutorial. (c1999-2022). Retrieved February 10, 2022, from https://www.w3schools.com/bootstrap5/index.php

III. Seznam obrázků a tabulek

Seznam tabulek

IV. Seznam ukázek kódů

V. Seznam příloh

VI. Přílohy

A. Zdrojový kód

Celý zdrojový kód aplikace je přístupný na této webové adrese:
https://github.com/huttt10/LezeckyDenik

B. Testovací scénáře

Tabulka 1 Testovací scénáře

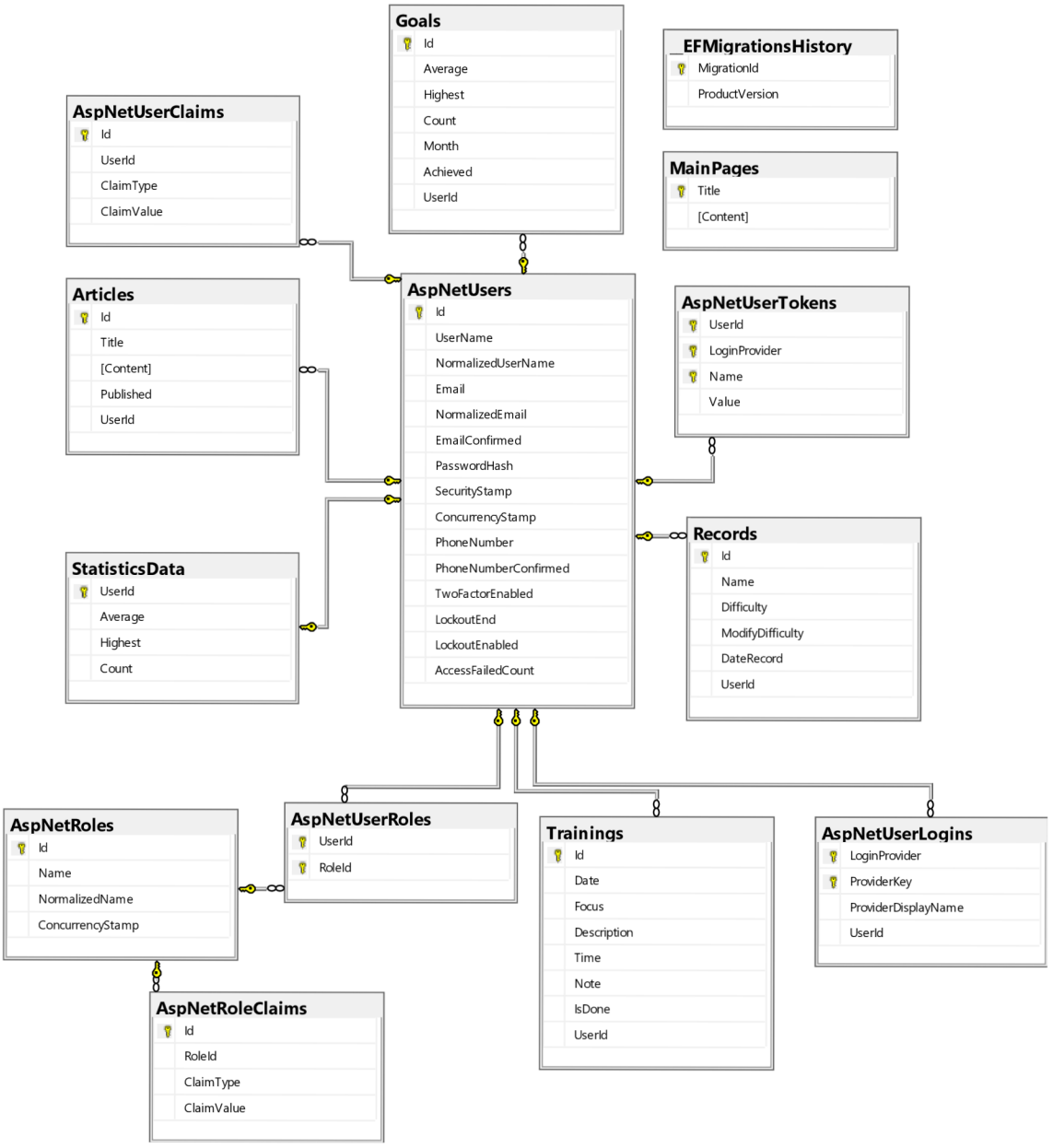
Akce	**Předpoklady**	**Instrukce**	**Očekávaný výsledek**
Registrace	-	Zadejte *email*, *heslo* a *potvrzení hesla*, poté klikněte na tlačítko *Přihlásit se*.	Uživateli se vytvoří nový účet a dostane emailem zprávu o potvrzení účtu.
Přihlášení	Uživatel je registrovaný.	Zadejte *email* a *heslo* poté klikněte na tlačítko *Přihlásit se*.	Uživatel byl přihlášen.
Přidat záznam	Uživatel je přihlášený.	Zadejte *Název*, *Obtížnost*, *Datum vylezení*.	Záznam se přidal do databáze.
Zobrazit *List záznamů*	Uživatel je přihlášený. Uživatel má přidaný alespoň jeden záznam o výstupu.	Zobrazit stránku *List záznamů*.	Zobrazí se všechny záznamy uživatelů.
Zobrazit *Globální statistiky*	Uživatel je přihlášený. Uživatel má přidaný alespoň jeden záznam o výstupu.	Zobrazit stránku *Globální statistiky*.	Zobrazí se všechny záznamy všech uživatelů.
Přidat cíl	Uživatel je přihlášený.	Na stránce *Plánování cílů*. Kliknout na tlačítko *Vytvořit nový cíl*. Vyplnit formulář a klinout na tlačítko *Vytvořit*.	Záznam se přidal do databáze.
Pokračování na další stránce.			

Zobrazit *Plánování cílů*	Uživatel je přihlášený. Uživatel má přidaný alespoň jeden záznam cíle.	Zobrazit stránku *Plánování cílů*.	Zobrazí se všechny záznamy cílů přihlášeného uživatele.
Přidat tréninkovou jednotku	Uživatel je přihlášený.	Na stránce *Záznamy tréninků*. Kliknout na tlačítko *Vytvořit novou tréninkovou jednotku.* Vyplnit formulář. Kliknout na tlačítko *Vytvořit.*	Záznam se přidal do databáze.
Zobrazit tréninkovou jednotku	Uživatel je přihlášený.	Zobrazit stránku *Záznamy tréninků*.	Zobrazí se všechny záznamy tréninkových jednotek aktuálně přihlášeného uživatele.
Editovat tréninkovou jednotku	Uživatel je přihlášený. Existuje tréninková jednotka.	Kliknout na tlačítko *Edit*. Změnit konkrétní údaje. Kliknout na tlačítko *Uložit*.	V databázi se provede konkrétní změna tréninkové jednotky.
Zobrazit detail tréninkové jednotky	Uživatel je přihlášený. Existuje tréninková jednotka.	Kliknout na tlačítko *Detail.*	Zobrazí se detail vybraného záznamu.
Smazat tréninkovou jednotku	Uživatel je přihlášený. Existuje tréninková jednotka.	Kliknout na tlačítko *Delete.*	Tréninková jednotka se vymaže z databáze.
Zobrazit *Přehled*	Uživatel je přihlášený. Od každého druhu záznamu existuje alespoň jeden s aktuálním datem.	Zobrazit stránku *Přehled.*	Zobrazí se všechny záznamy přihlášeného uživatele.
Pokračování na další stránce.			

Přidat článek	Uživatel je přihlášený. Má roli *Editor* nebo *Admin* nebo *SuperAdmin*.	Vyplnit textová pole *Název článku* a *obsah článku*. Kliknout na tlačítko *Uložit*.	Článek se uloží do databáze.
Editovat článek	Uživatel je přihlášený. Má roli *Editor* nebo *Admin* nebo *SuperAdmin*. Existuje alespoň jeden článek.	V *Seznamu všech napsaných článků* se klikne na tlačítko *Edit*. Provede se změna textu článku a klikne se na tlačítko *Uložit*.	V databázi se provede konkrétní změna dat u záznamu článku.
Editovat úvodní stránku	Uživatel je přihlášený. Má roli *Admin* nebo *SuperAdmin*.	V *Administrátorském panelu* se klikne na tlačítko *Editovat úvodní stránku*. Provede se změna textu a klikne se na tlačítko *Uložit*.	V databázi se provede konkrétní změna dat u entity `MainPage`.
Změnit roli uživateli.	Uživatel je přihlášený. Má roli *Admin* nebo *SuperAdmin*.	V *Nastavit role uživatelům a smazat uživatele.* Klikne na tlačítko *Změnit roli.* Z nabídky vybere jinou roli. Kline na tlačítko *Změnit*.	V databázi se provede konkrétní změna role u uživatele.

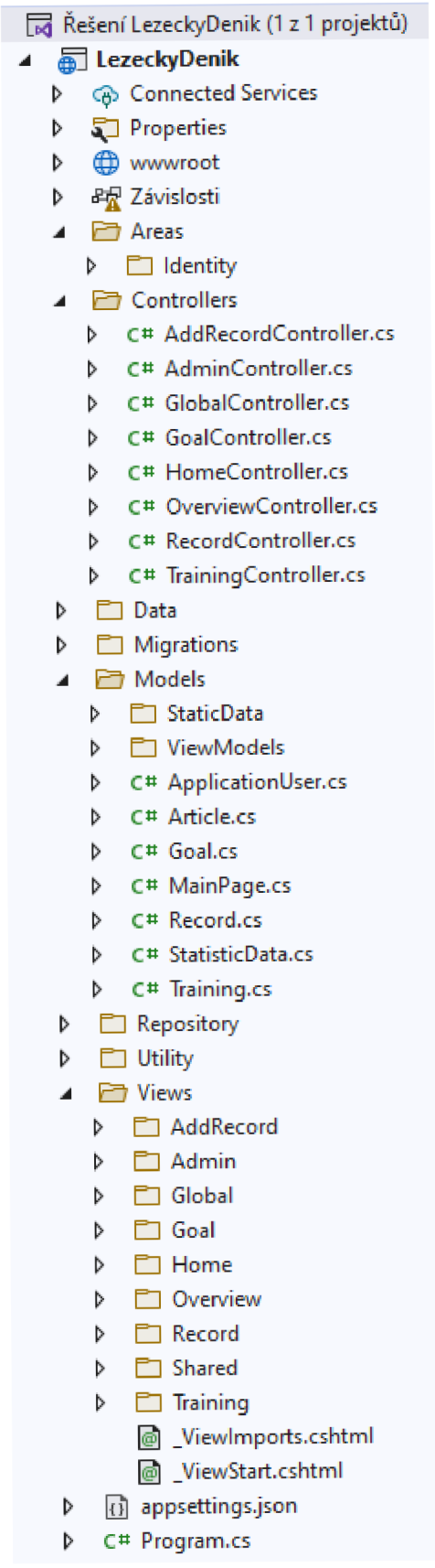
Zdroj: Autor

C. Databázový model



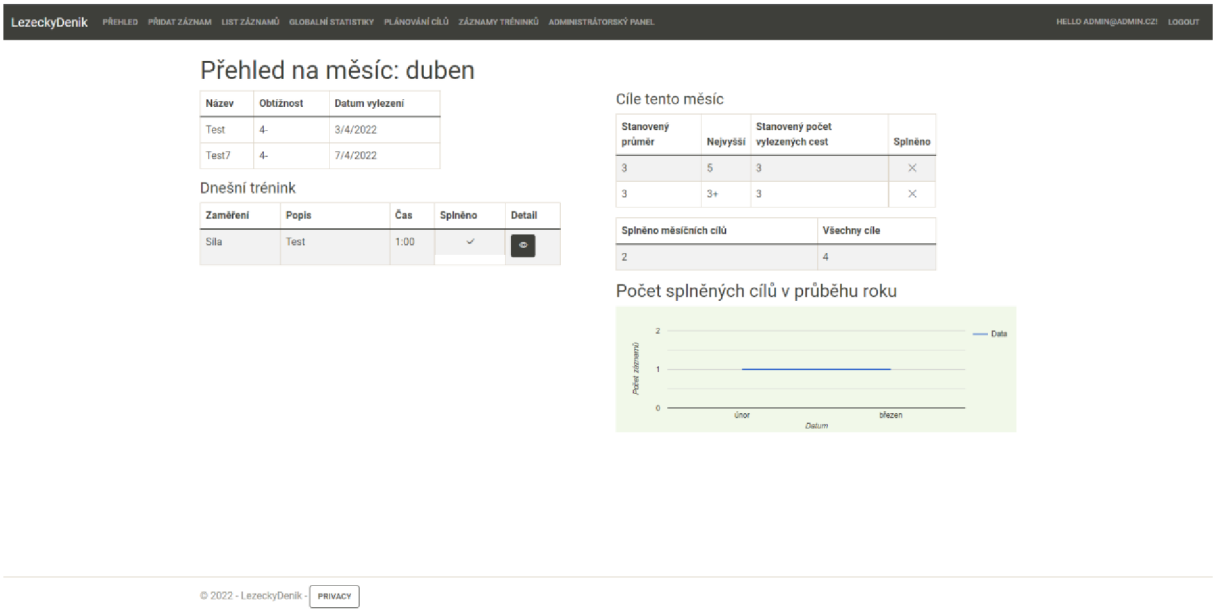
Zdroj: Autor

D. Struktura aplikace



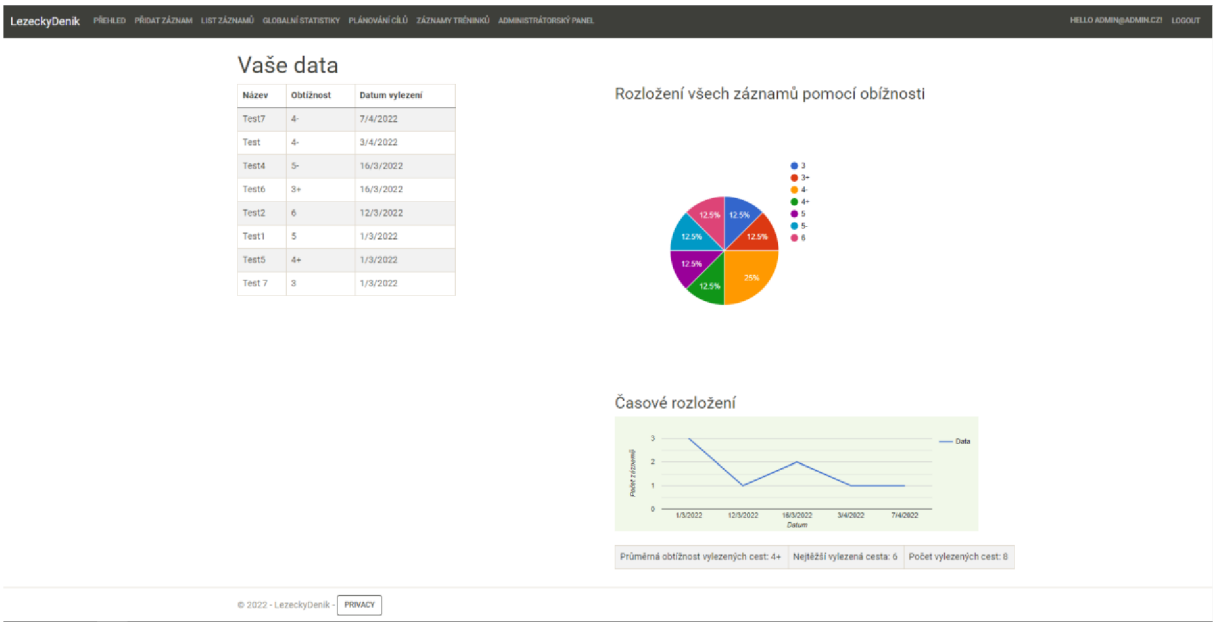
Zdroj: Autor

E. Snímek stránky *Přehled*



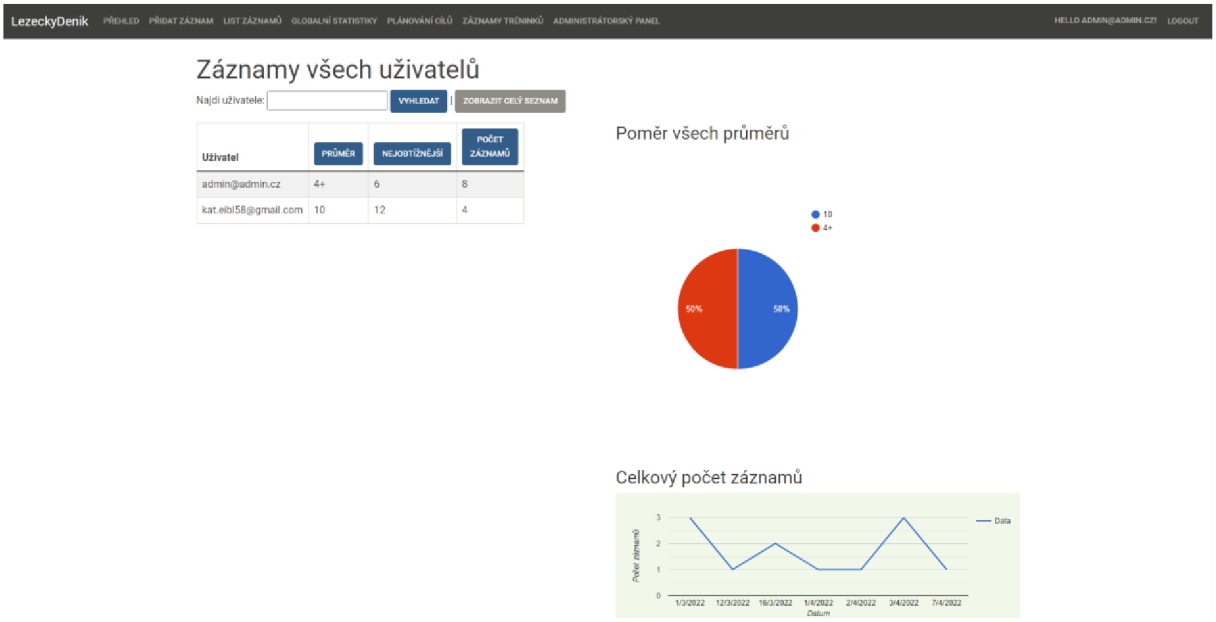
Zdroj: Autor

F. Snímek stránky *List záznamů*



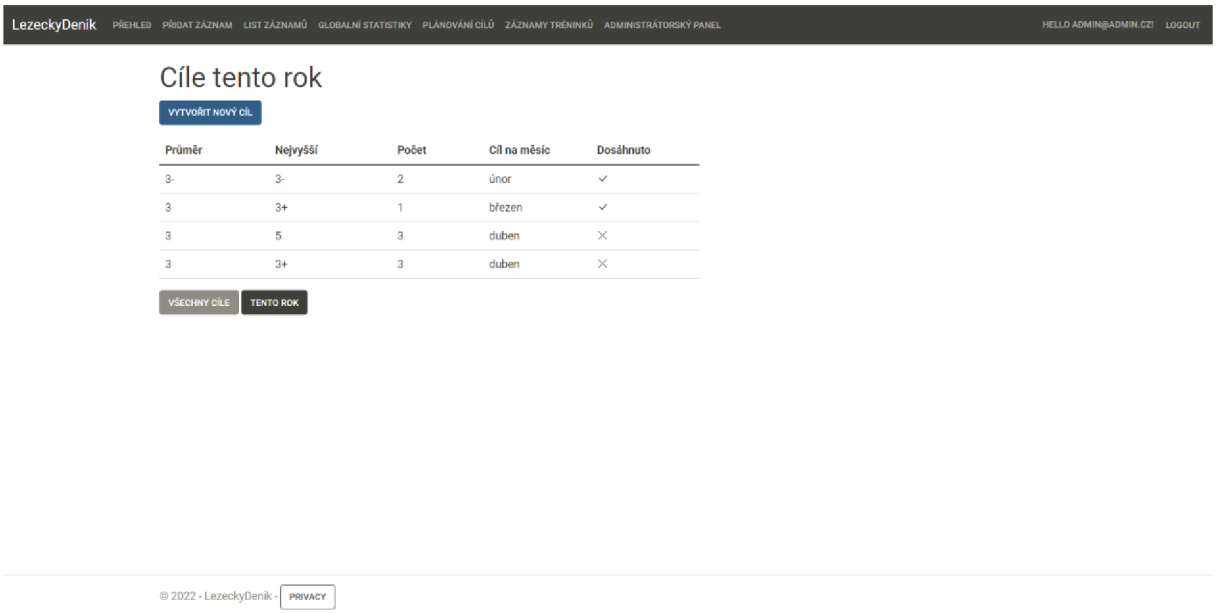
Zdroj: Autor

G. Snímek stránky *Globální statistiky*



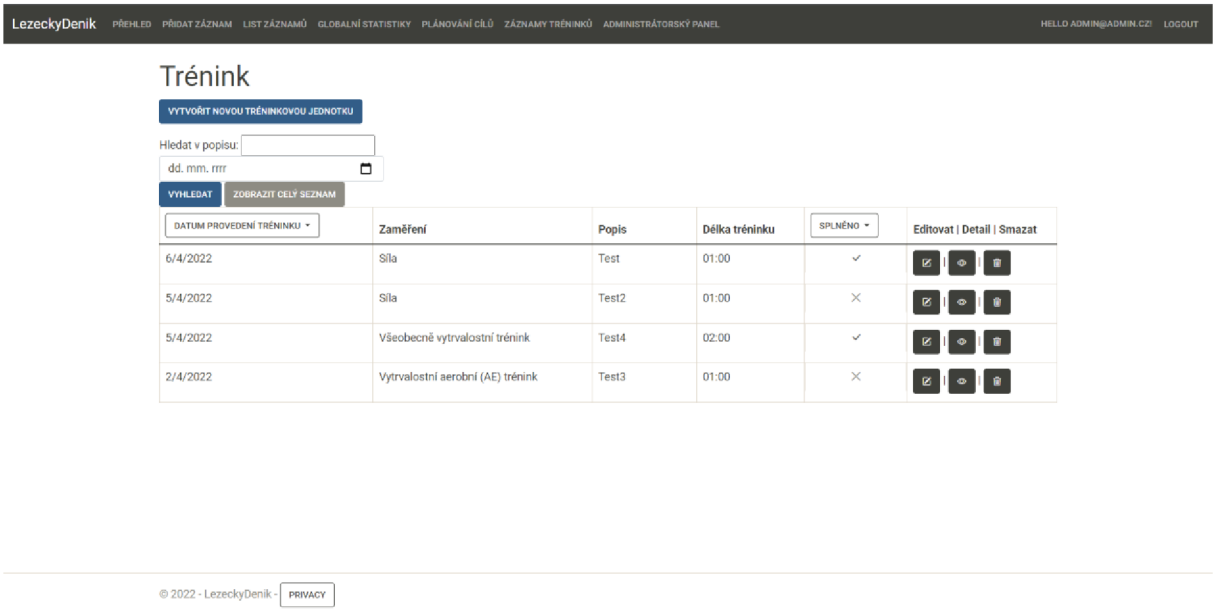
Zdroj: Autor

H. Snímek stránky *Globální statistiky*



Zdroj: Autor

CH. Snímek stránky *Záznamy tréninků*



Zdroj: Autor