Webová aplikace pro místní sportovní klub

Tomáš Janečka

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Janečka**
Osobní číslo:	**A17124**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Webová aplikace pro místní sportovní klub**
Téma práce anglicky:	**A Web Application Design for a Local Sports Club**

Zásady pro vypracování

1. Proveďte rešerši typu webové prezentace sportovního klubu.
2. Proveďte analýzu požadavků na aplikaci.
3. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
4. Realizujte navrženou aplikaci ve zvolené technologii.
5. Věnujte pozornost zabezpečení aplikace.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. GALLOWAY, Jon, Brad WILSON, K. Scott ALLEN a David MATSON. Professional ASP.NET MVC 5. Indianapolis, IN: Wrox, a Wiley brand, 2014. Wrox professional guides. ISBN 978-1118794753.
2. JOHNSON, Glenn. Programming in HTML5 with JavaScript and CSS3: training guide. Redmond, Wash.: Microsoft, 2013. ISBN 978-0735674387.
3. MUNRO, Jamie. ASP.NET MVC 5 with bootstrap and knockout.js: building dynamic, responsive web applications. Sebastopol: OReilly Media, 2015. ISBN 978-1491914397.
4. ANDERSON, Elijah. *Sql: The Ultimate Beginners Guide To Learn SQL In Less Than 24 Hours*. CreateSpace Independent Publishing Platform, 2016.
5. ASP.net MVC with entity framework and CSS. New York, NY: Springer Science+Business Media, 2016. ISBN 978-1484221365.

Vedoucí bakalářské práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **28. listopadu 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

PROHLÁŠENÍ

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 10.8.2020

Tomáš Janečka, v. r.
podpis diplomanta

ABSTRAKT

Bakalářská práce se zabývá tématem tvorby webové aplikace pro místní sportovní klub. Aplikace slouží především jako evidence stavu účastí uživatelů. Má sloužit k usnadnění organizace pravidelných sportovních akcí. Uživatelé si vytvoří vlastní účet a bude jim přiřazen jeden nebo více periodicky se opakujících srazů. V aplikaci také bude existovat jeden nebo více administrátorů, kteří budou moci spravovat srazy a přiřazovat do nich uživatele.

Klíčová slova: Webová aplikace, ASP.NET, sportovní klub, pravidelné setkání

ABSTRACT

This bachelor's thesis deals with the creation of a web application for a local sports club. Its main objective is to keep track of which participants are participating in the next meetup. It is meant to make meetup management easier. Users will be able to create their own user account and they will be assigned one or more meetups, which take place periodically. For every meetup, a user is able to let their teammates know whether they are participating in the upcoming meetup or not. There will also be one or more administrators, who will be able to manage meetups and assign users to them.

Keywords: Web application, ASP.NET, sports club, regular meetup

Děkuji panu doc. Petrovi Šilhavému za konzultace a připomínky.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Pořádání sportovních akcí vyžaduje určitou míru organizace. Může k ní docházet několik dní před samotnou akcí, či těsně před jejím začátkem. Existují také různé způsoby, jak organizovat. Pořadatel může vyvěsit plakáty, kontaktovat účastníky přes telefon nebo využít metody, jakými jsou třeba webové aplikace.

Místní badmintonový klub, kterého jsem součástí, využívá určitou webovou aplikaci, o které se zmíním posléze. Tuto aplikaci využívá již delší dobu. Časem jsem vypozoroval určité žádoucí, ale i nežádoucí vlastnosti této aplikace. Dále jsem si všiml při samotném konání akce, že by bylo dobré učinit ještě před jejím začátkem kroky, které by zaručily její hladký průběh. Často totiž dochází ke zmatečnému počínání při rozdělování hráčů na kurty mezi jednotlivými zápasy. Proto by bylo vhodné, kdyby aplikace uměla poskytnout soupisku zápasů, která obsahuje všechny přihlášené účastníky. Dále mnohokrát došlo k situaci, kdy hráč dorazil na setkání s dalším člověkem. Aktuální systém neposkytuje jinou možnost pro ohlášení hráče navíc než zanecháním zprávy. Přitom ale obsahuje počítadlo přihlášených osob. Tyto poznatky mě podnítily pro vytvoření užitečné aplikace, která by měla pomoci s organizací před a v průběhu akce.

Projekt má formu webové aplikace kvůli zabezpečení snadného přístupu uživatelů. Je využito technologií HTML, CSS a JavaScript pro tvorbu webových stránek. Pro zpracování požadavků uživatele a práci s daty je použit framework ASP.NET, jelikož poskytuje většinu potřebných nástrojů k zhotovení aplikace.

I. TEORETICKÁ ČÁST

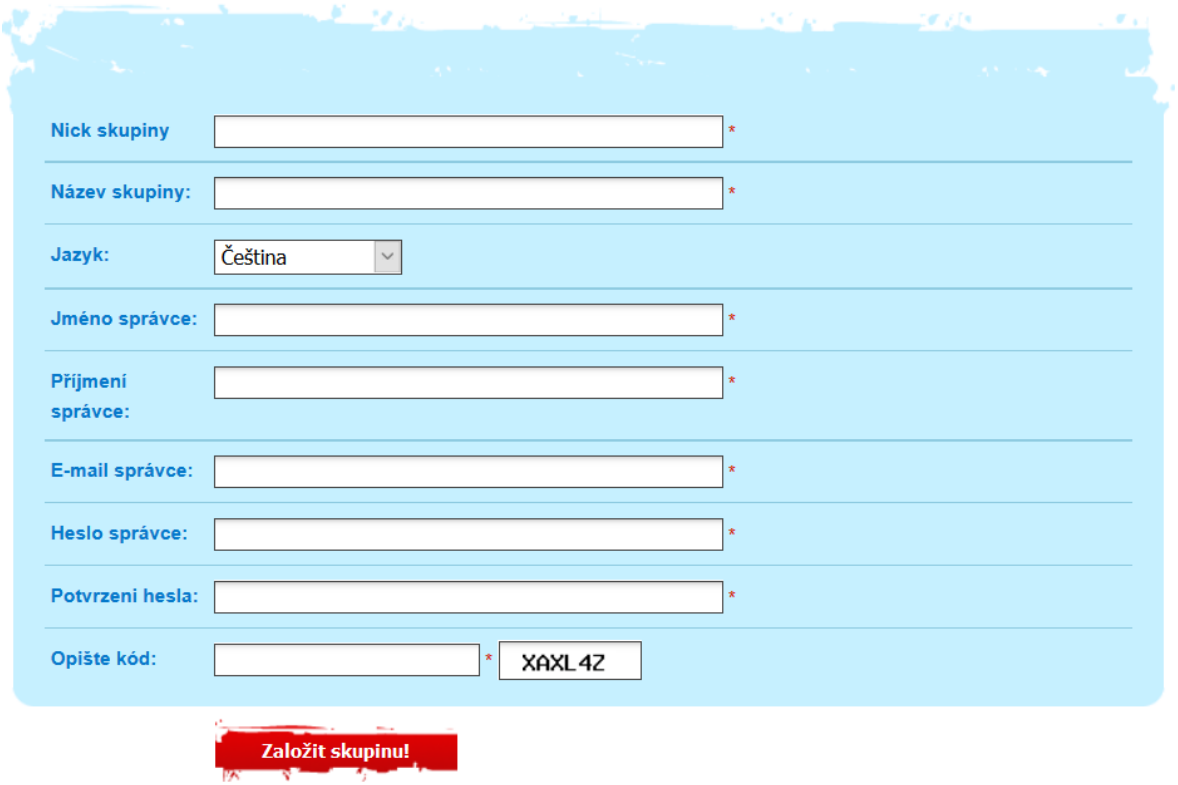
1 REŠERŠE APLIKACÍ

Aktuálně využívaným systémem pro organizaci je webová aplikace sejdemse.net. Tato aplikace stanovuje celou řadu žádoucích funkcí, které by se měly nacházet ve vypracovávané aplikaci.

Existují ale i další weby. V každé z níže uvedených aplikací může člověk bezplatně využívat jejích služeb. Většina z nich je v češtině, což je pro naše potřeby důležitý parametr.

1.1 Sejdemse.net

Jedná se o aktuálně využívanou aplikaci. K formuláři pro vytvoření srazu se člověk dostane z hlavní stránky po kliknutí na odkaz „Založ si svou skupinu“. Uživatel uvidí stránku s popisem nastavení a používání webu. Pod popisem se nachází samotný formulář. [1]



Obrázek 1: Sejdemse.net - registrace

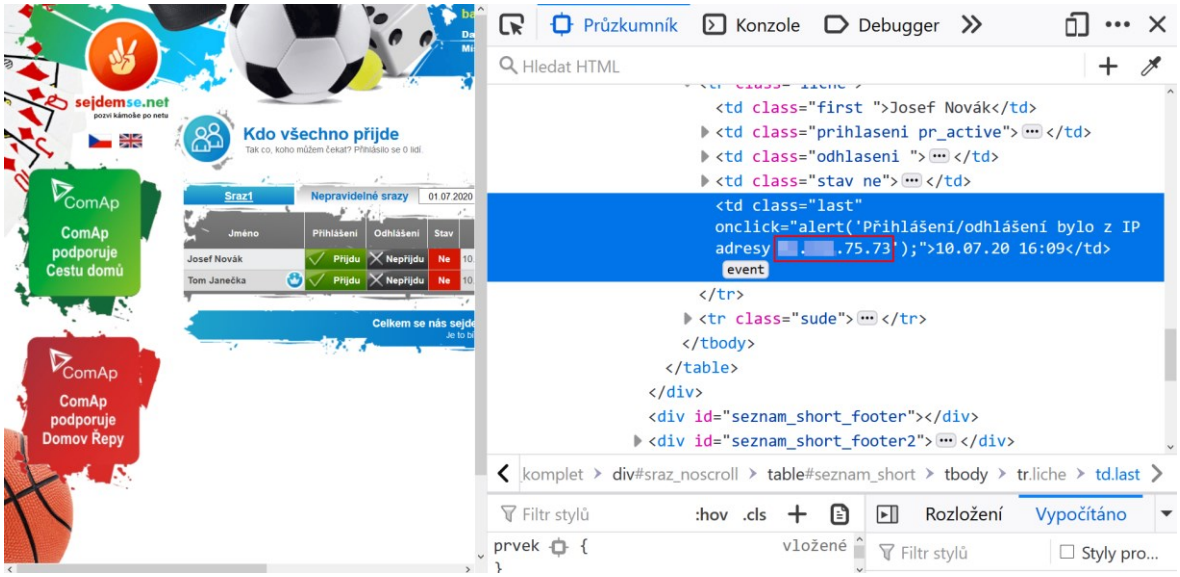
K přihlášení správce se používá „nicku skupiny“ a hesla. Aplikace dovolila u testovacího srazu použít heslo „abcd“. Běžnému uživateli ke zobrazení skupiny stačí navštívit odkaz, který je po vytvoření srazu zaslán správci skupiny na e-mail. Přístup do skupiny může být chráněn heslem, ale také nemusí. V takovém případě se pomocí odkazu dostane do skupiny kdokoliv. Tento člověk bude moci změnit stav účasti u každého, protože to aplikace

dovoluje, a také bude moci ve skupině zanechat zprávy, u kterých si může zvolit libovolné jméno odesílatele, a tudíž se může vydávat za někoho jiného.

Obrázek 2: Sejdemse.net – stránka srazu

Na stránce srazu uživatel vidí ostatní účastníky a také jejich stav účasti. Je evidováno také datum změny tohoto stavu. Mezi informace o srazu patří datum nadcházejícího srazu, místo konání a také počet všech účastníků, jejichž stav je nastaven na „Ano“. V této testovací skupině je na výběr ze dvou srazů: „Sraz1“ a jeden nepravidelný sraz, který lze vybrat z dropdown menu. Příspěvky od uživatelů jsou sdíleny napříč všemi srazy.

Pokud uživatel v prohlížeči využije funkce „Prozkoumat prvek“, v HTML kódu u elementu, který uchovává stav účasti, se nachází IP adresa, ze které došlo ke změně stavu účasti.



Obrázek 3: Sejdemse.net – IP adresa

V aplikaci lze vytvořit až 3 periodicky se opakující srazy. U každého lze nastavit počet týdnů nebo měsíců mezi jednotlivými konáními srazu. Minimální počet je 1 a maximální je 99. Dále si správce vybírá den a čas konání srazu. V případě měsíčního opakování si správce vybírá den v týdnu a týden. Kupříkladu se sraz bude opakovat každý třetí měsíc každé druhé úterý.

Obrázek 4: Sejdemse.net – nastavení srazu

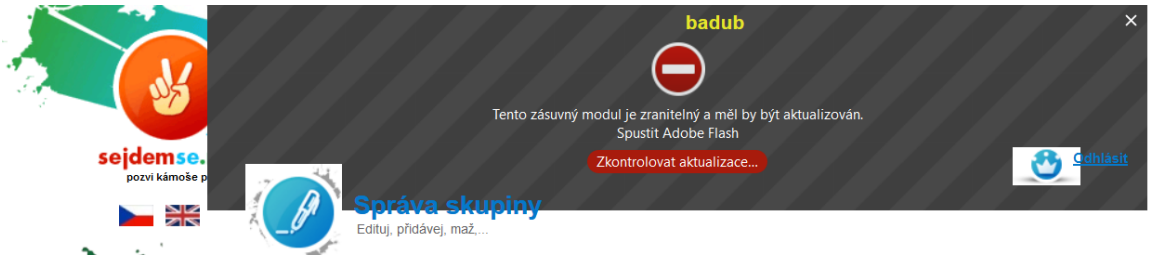
Správce také musí uvést místo srazu, počet hodin před konáním srazu, kdy bude zasláno upozornění a také ideální počet hráčů. Podle ideálního počtu se u každého srazu mění komentář u počtu přihlášených účastníků.

Správce má možnost vytvářet i nepravidelné srazy. Je u nich třeba zadat přímo datum konání. Může jich být vytvořen libovolný počet. Účastník může změnit svůj stav i po proběhnutí nepravidelného srazu.

Je k dispozici výpis uživatelů skupiny a také jejich e-mailů. Správce může upravovat údaje účastníků a také je mazat. Uživatel může být evidován jako náhradník. Náhradník má k dispozici stejné funkce jako běžný uživatel, pouze je ve výpisu všech účastníků srazu označen ikonou.

Existuje zde záložka s názvem „Statistika“. Bohužel se mi nepodařilo ani během několikaměsíčního používání tuto funkci vyzkoušet, protože je dočasně nedostupná.

Design webu není responzivní a jeho používání pomocí mobilního telefonu není ideální. Při neopatrném přibližování stránky může dojít k nechtěné změně stavu některého z účastníků. Nacházejí se zde také animace, které vyžadují Adobe Flash a na některých prohlížečích dochází k nechtěným jevům.

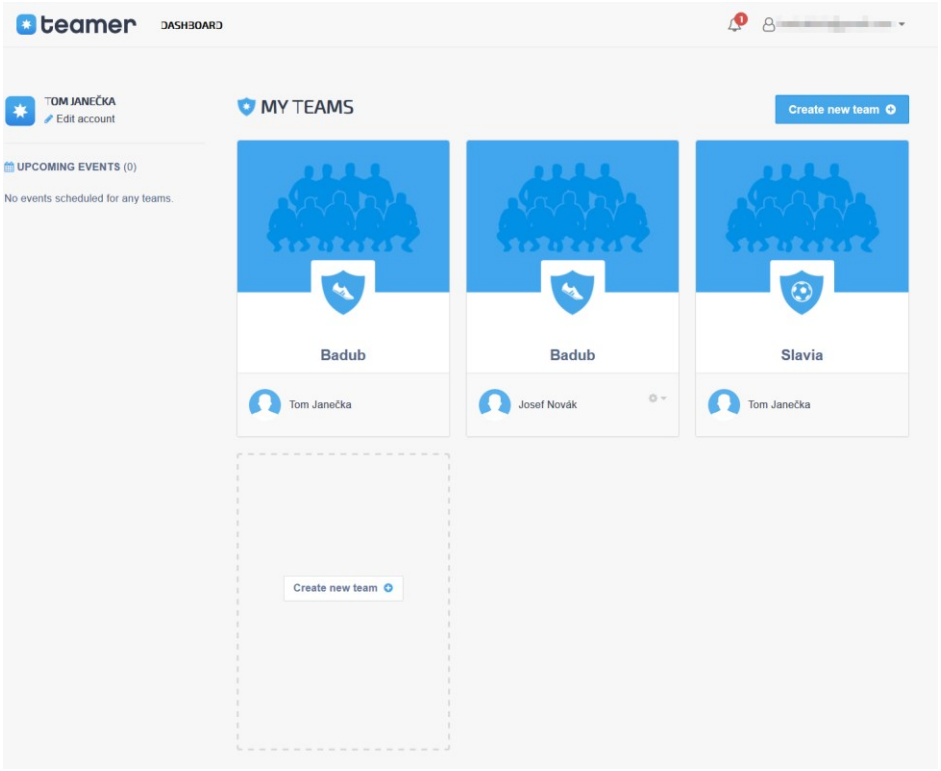


Obrázek 5: Sejdemse.net – Adobe Flash

1.2 Teamer.net

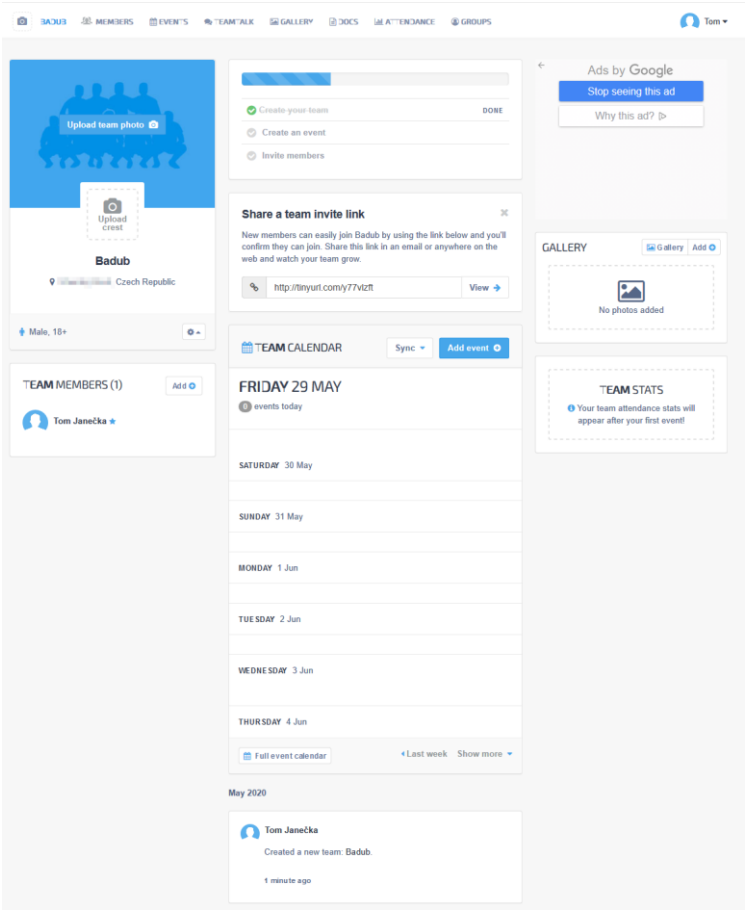
Tento web slouží ke správě sportovního týmu nebo klubu. Využívá ho 2 miliony uživatelů. [2]

Uživatel při registraci vyplňuje své celé jméno, adresu, rok narození, e-mail a heslo. Pro testovací sraz bylo opět akceptováno heslo „abcd“. V rámci každého účtu může existovat více lidí. Každý uživatel může vytvořit nový tým a stát se jeho správcem. Přitom může být členem libovolného počtu týmů.



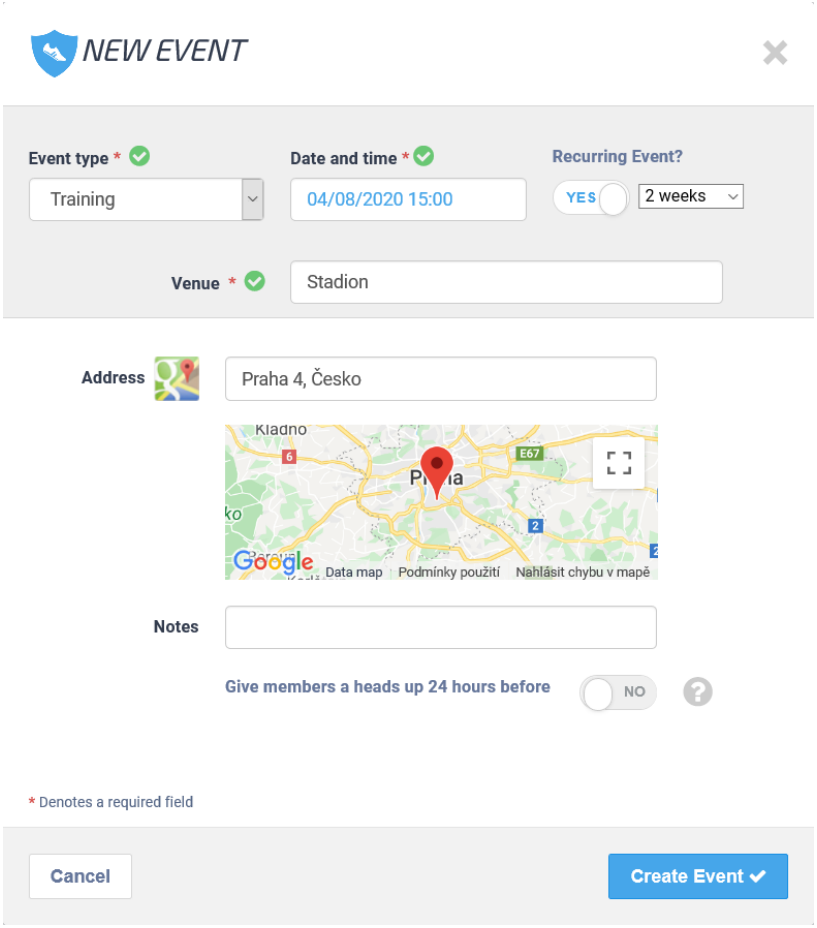
Obrázek 6: Teamer.net – přehled účtu s více uživateli

Při vytvoření týmu uživatel vyplňuje údaje jako název týmu, název klubu, druh sportu, věkovou skupinu členů, zda se jedná o chlapecký, dívčí nebo smíšený tým a adresu základny týmu. Aplikace se chová ke všem druhům sportu naprosto stejně.



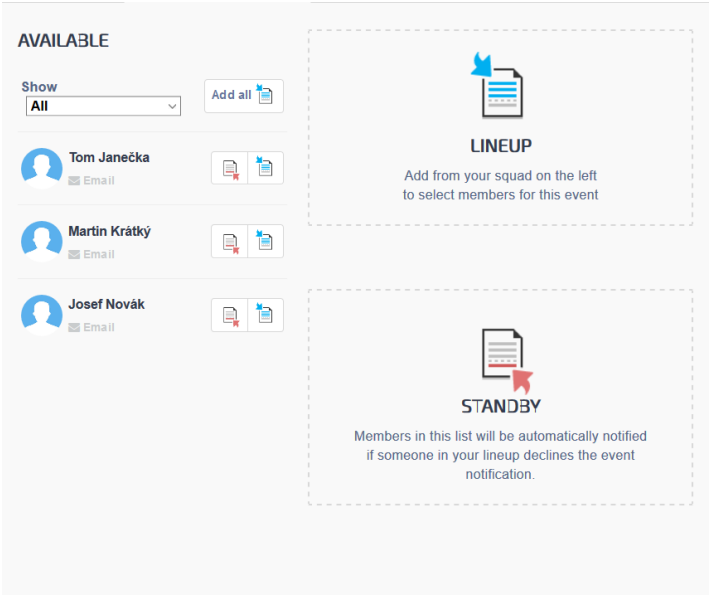
Obrázek 7: Teamer.net – přehled aktivity týmu

Po vytvoření týmu se zobrazí jeho přehled. Jsou k dispozici informace o členech, názvu a lokaci týmu, odkaz pro pozvání členů, seznam nadcházejících událostí a přehled celkové aktivity v týmu. Uživateli je doporučeno, aby jako další krok vytvořil novou událost.



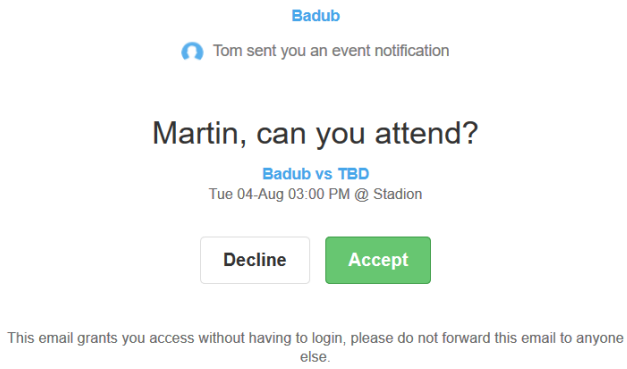
Obrázek 8: Teamer.net – nová událost

Uživatel si při tvorbě nové události vybere typ této události. Může se jednat kupříkladu o zápas, trénink, závod nebo turnaj. Dále je zvoleno datum. Je zde možnost, že se tato událost bude opakovat několik týdnů za sebou. Může být vybrána hodnota až 12 týdnů. Při zaškrtnutí možnosti opakující se události je v nabídce i položka „0 weeks“.



Obrázek 9: Teamer.net – přiřazování účastníků k události

U každé události správce týmu zařazuje členy do dvou skupin. První skupina jsou členové, kteří by se měli zúčastnit. Druhá skupina jsou náhradníci, kteří zastoupí členy z první skupiny, kteří zamítnou účast. Každému členovi musí být nejprve zasláno upozornění na nadcházející událost. Ten poté bude moci potvrdit nebo zamítnout účast.



Obrázek 10: Teamer.net – e-mail pro potvrzení účasti

Uživatelé si mohou prohlížet údaje o ostatních členech týmu. Dále mají k dipozici přehledný výpis nadcházejících událostí vytvořených správcem týmu. Lze si je zobrazit v kalendáři nebo seznamu a také je možné je filtrovat podle stavu účasti člena týmu nebo podle toho, jestli už událost proběhla. V rámci celého týmu je k dispozici funkce zasílání zpráv ostatním členům. Uživatelé mohou přidávat komentáře pod příspěvky a své příspěvky mazat. Správce týmu může smazat libovolný příspěvek. Existuje zde také

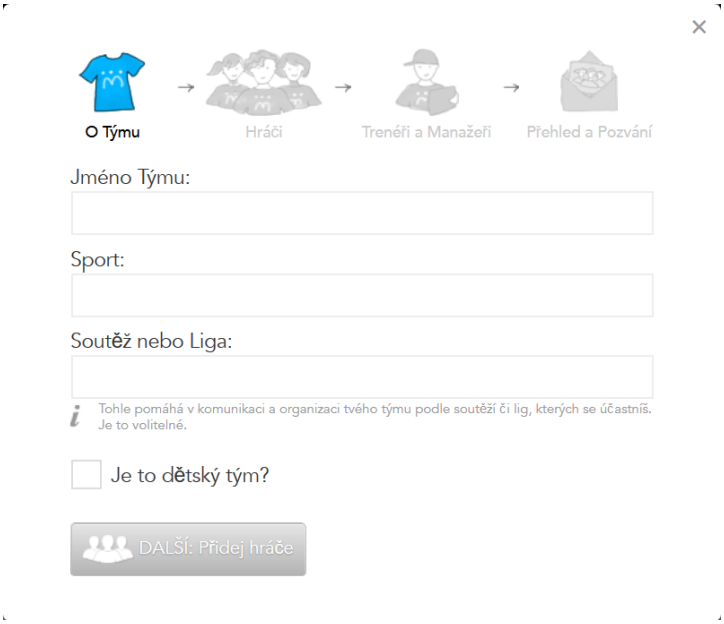
místnost, kam správce přidává obrázky viditelné pro všechny členy. Členové mají také přehled o historii účastí na událostech.

Přidání nových členů lze provést dvěma způsoby. Pro každý způsob již musí existovat uživatel, který má na webu svůj účet. První způsob je pozvání přes e-mail. Správce vyplní jméno a příjmení uživatele a jeho e-mail. Pozvaný uživatel dostane zprávu a po kliknutí na odkaz může vstoupit do týmu. Další způsob je poskytnutí unikátního odkazu uživateli, který pomocí něj vstoupí do týmu.

Aplikace poskytuje mnoho užitečných funkcí ale má také své nevýhody. Není k dispozici v českém jazyce. Na každé stránce se nachází reklamy. Členové týmu se nemohou na vytvořené události hlásit sami. Nelze snadno vytvořit události, které se opakují každý den nebo jednou za měsíc. Lze používat slabé heslo. Web neposkytuje responzivní design. V neposlední řadě také aplikace neposkytuje požadované funkce pro generování soupisky zápasů a pro přidání hosta na jednu nadcházející událost.

1.3 Teamstuff.net

Aplikace poskytuje velmi podobné funkce jako předchozí. Navíc je ale dostupná v českém jazyce. Při registraci je uživatel nejprve vyzván pro vyplnění e-mailu, jména a hesla. V testovacím týmu je třeba delšího hesla než v předchozích případech, a to alespoň šestimístné. Bylo zvoleno heslo „abcdef“. Poté hned dochází ke tvorbě týmu.

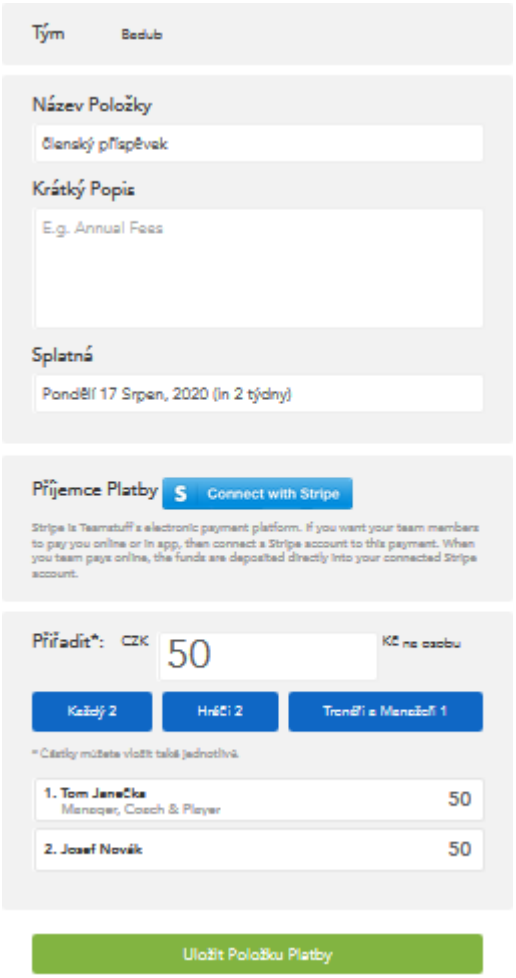


Obrázek 11: Teamstuff.com – nový tým

Vytvoření týmu je provedeno skrze modální okno s několika formuláři. Jedná se o velice intuitivní postup. Uživatel zde má možnost přímo pozvat členy do týmu zadáním jejich e-mailových adres.

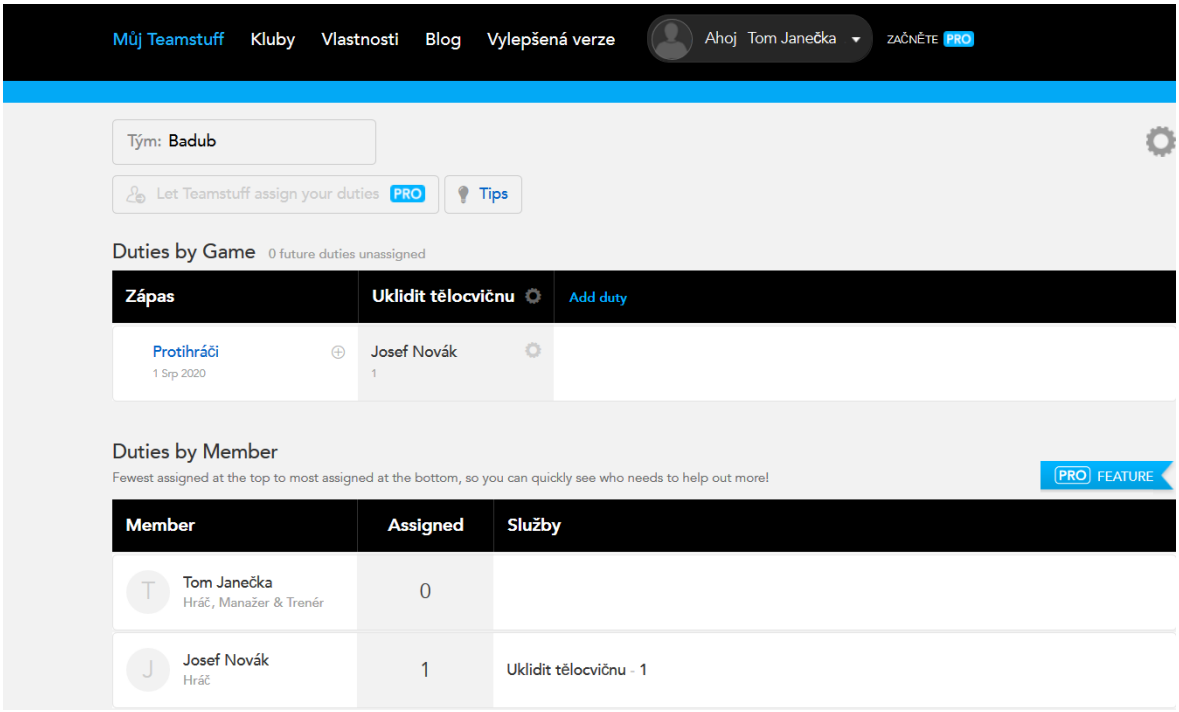
Obrázek 12: Teamstuff.com – první akce po vytvoření týmu

Uživatel má v menu na výběr z několika různých funkcí. Funkce „Aktivita“ eviduje veškeré změny, ke kterých došlo v týmech, jichž je uživatel součástí. V neplacené verzi aplikace je zobrazena aktivita za posledních 7 dnů. Dále má uživatel k dispozici kalendář, kde vidí nadcházející události. Přímo z kalendáře může potvrdit nebo zamítnout svoji účast na jednotlivých událostech. Změna účasti je prováděno pomocí jednoho tlačítka, které mění svůj stav. Výchozí stav je nastaven jako „?“. Hodnota výchozího stavu může být změněna. V takovém případě by se stav člena týmu mohl při vypsání události automaticky nastavit kupříkladu na „Ano“. Při klikání na tlačítko stavu účasti se cyklicky mění z „?“ na „Ano“, z „Ano“ na „Ne“ a z „Ne“ na „?“. Zároveň se změnou stavu účasti se mění seznam přihlášených členů.



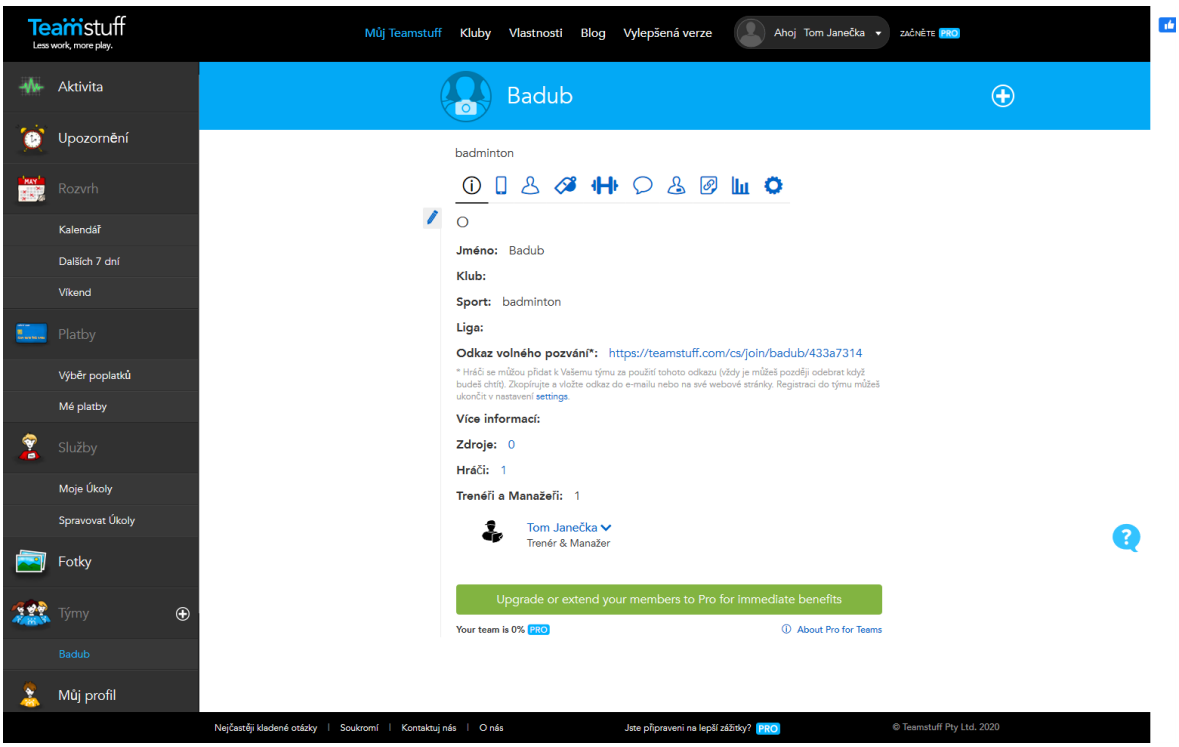
Obrázek 13: Teamstuff.com – nová platba

Aplikace nabízí možnost vybírání poplatků. Tato funkce je dostupná v bezplatné verzi aplikace. Správce může pouze evidovat provedené platby v hotovosti.



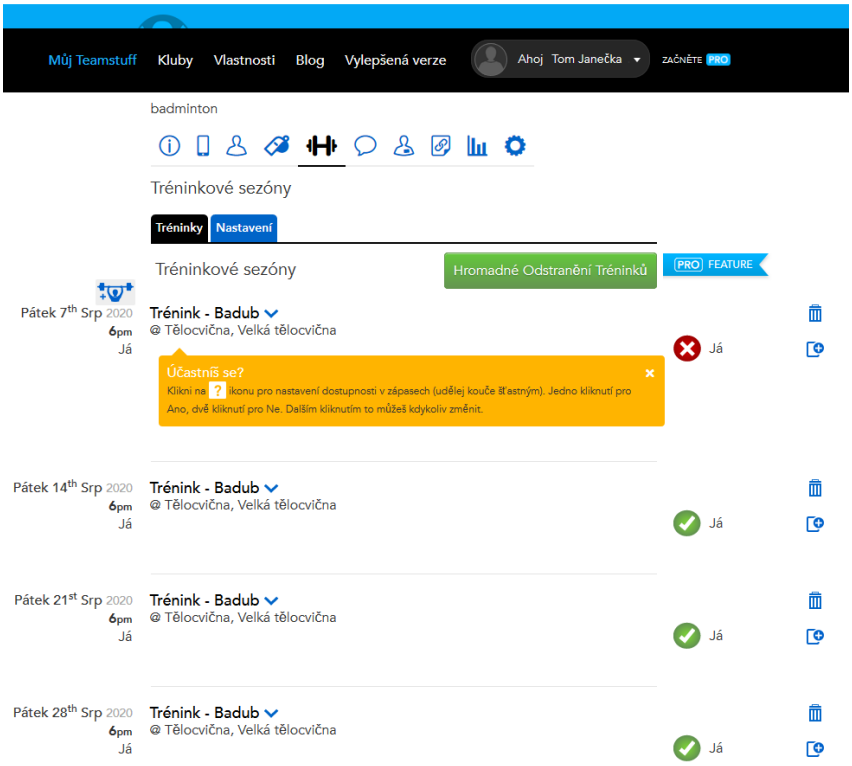
Obrázek 14: Teamstuff.com - úkoly

Správce má k dispozici vytvářet úkoly, které je potřeba provést před, během nebo po události. K úkolům správce přiřadí jednoho nebo více členů. Členové jsou upozorněni na to, že jim byl přiřazen úkol.



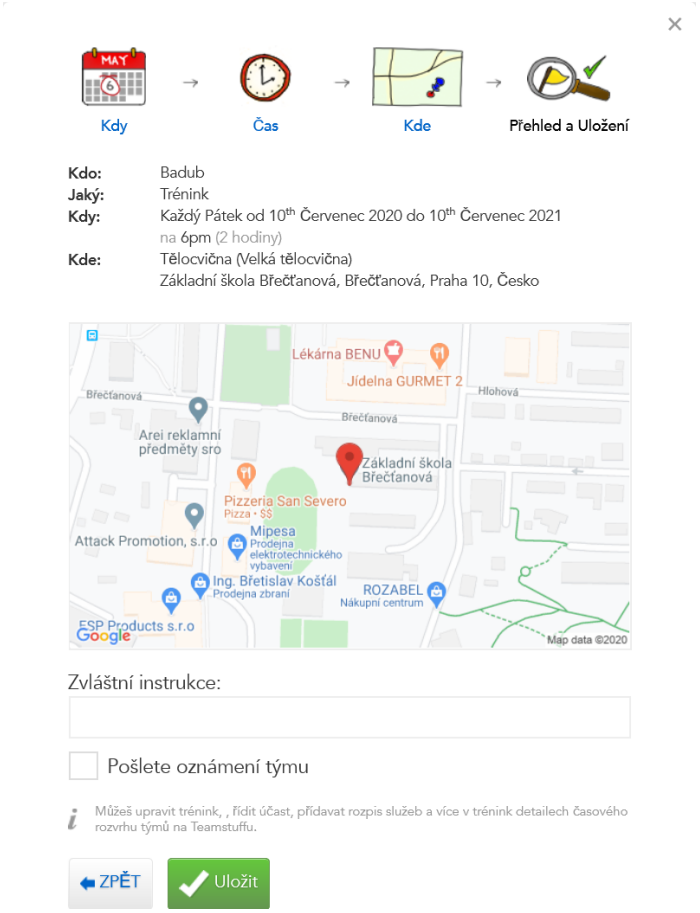
Obrázek 15: Teamstuff.com – přehled týmu

Uživatel si může vybrat požadovaný tým z menu a zobrazit si jeho přehled. U každého týmu je k dispozici několik záložek s různými informacemi. Správce týmu může v těchto záložkách upravovat odpovídající informace. Mezi záložky patří „Kontakt“, „Hráči“, „Naplánované zápasy“, „Naplánované tréninky“, „Zprávy“, „Dokumenty“, „Statistiky“ a „Nastavení týmu“.



Obrázek 16: Teamstuff.com – přehled tréninků

Tréninky mají v této aplikaci nejblíže k periodicky se opakujícím událostem. Jejich vytvoření probíhá podobným způsobem jako vytvoření týmu. Opět je zobrazeno modální okno s několika kroky. U formuláře se nacházejí přehledné vysvětlivky. Správce má na výběr z několika možností opakování. Může se jednat o jednorázový trénink, denní nebo týdenní trénink a o trénink každý druhý týden. U jednorázového tréninku se vybírá datum uskutečnění, u ostatních je vybráno počáteční a koncové datum. V dalším kroku se vybírá čas začátku a doba trvání. V české verzi jsou obě pole označena jako „Čas“. Může být ještě vybrána lokace a adresa místa, kde se trénink koná. Ke každému novému tréninku jsou automaticky přiřazeni všichni členové týmu.



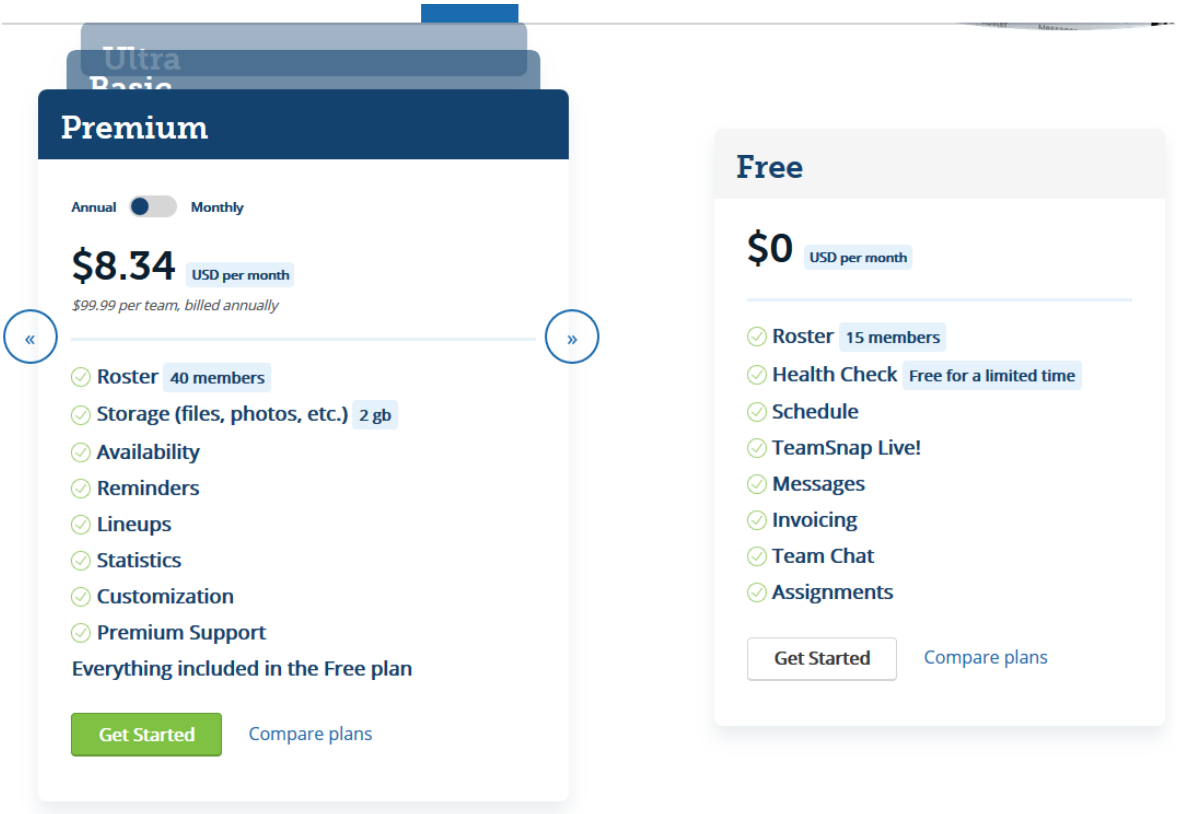
Obrázek 17: Teamstuff.com – přehled nového tréninku

Stejně jako v případě předchozí aplikace, není web responzivní. Oba systémy ale zdarma nabízí mobilní aplikaci. Obdobně také nenabízí možnost generování soupisky zápasů ani přidání hostů na nadcházející událost. Navíc může být stav účasti nastaven kromě „Ano“ a „Ne“ ještě na „?“, což znamená neznámý stav.

1.4 Teamsnap.com

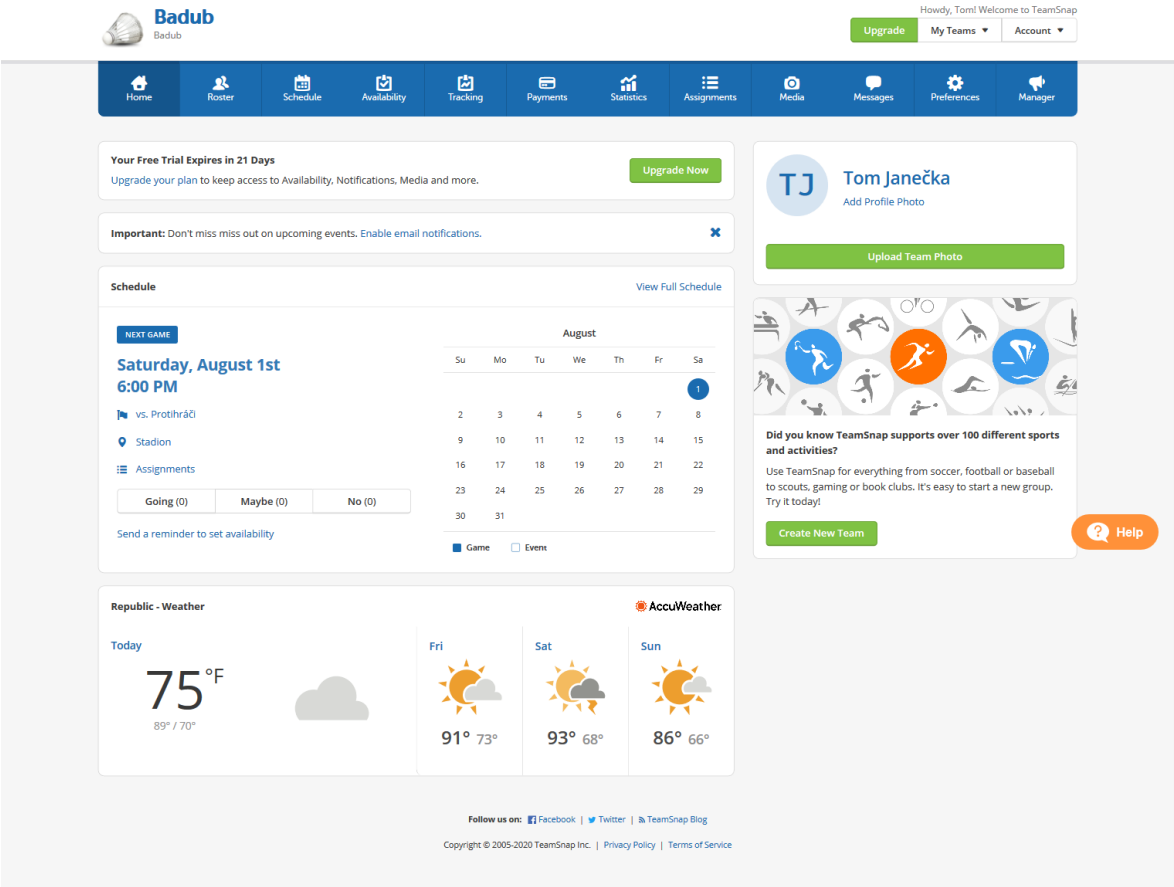
Další aplikace pro správu týmu je teamsnap.com. Využívá ji 23 milionů uživatelů a 19 tisíc sportovních organizací. [3]

Web je k dispozici pouze v anglickém jazyce. Stejně jako předchozí aplikace, nenabízí responzivní design, ale poskytuje mobilní aplikaci. Tento web se ale liší v tom, že i pro některé základní funkce musí být proveden upgrade na vyšší, placenou verzi. Aplikace vyžaduje silnější heslo. Pro testovací sraz bylo vybráno heslo „abcefgh1“.



Obrázek 18: Teamsnap.com – ceník a dostupné funkce

Při registraci uživatel vyplňuje e-mail, celé jméno, město a časovou zónu. Dále si vybírá svoji roli v týmu. Má na výběr ze správce týmu nebo jeho člena. Pokud vybere roli správce, je zobrazen formulář pro vytvoření týmu. V tomto formuláři se vybírá název týmu, prováděný sport, e-mailová adresa a heslo. Následují doplňující informace o týmu. Správce má možnost ihned přiřadit do týmu členy. Do formuláře vyplní jejich celé jméno a e-mail. Můžou být ihned zaslány pozvánky do týmu. Další krok je vytvoření prvního zápasu. Je třeba vyplnit název týmu protivníků, datum a čas zápasu a také jeho lokaci. Na závěr může správce poskytnout detaily o týmu. Může vybrat, zda se jedná o chlapecký, dívčí nebo smíšený tým, jestli je tým školní nebo rekreační a věkovou skupinu členů.

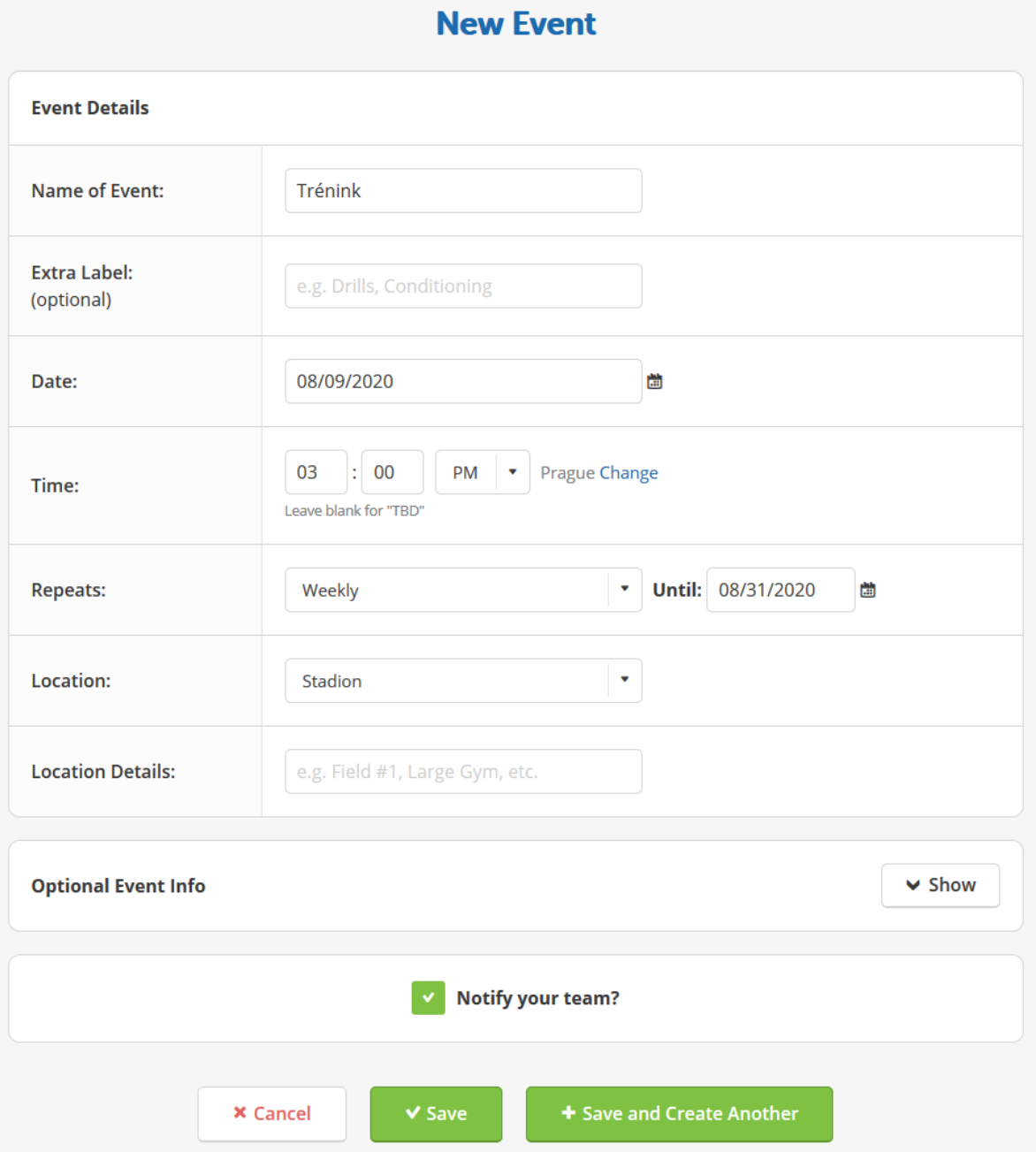


Obrázek 19: Teamsnap.com – přehled

Na hlavní stránce týmu jsou zobrazeny nadcházející události. Uživatel si může zobrazit tuto událost a nastavit svůj stav účasti. Nastavování stavu účasti a přehled všech účastí se po uplynutí zkušební verze stává nedostupným a je třeba upgradovat na placenou verzi aplikace.

Každý člen má k dispozici výpis všech ostatních členů. U každého člena vidí e-mailovou adresu a jeho role. Do bezplatných funkcí dále patří vytváření a evidence plateb, tvorba úkolů a přiřazování těchto úkolů ke členům a posílání e-mailů ostatním členům v rámci týmu.

Uživatelé mají možnost do týmu přidat nové členy. Tato funkce je dostupná ve výpisu uživatelů přes tlačítko „Add or Edit My Family Members“. Správce může přidávat jak nové členy, tak nové správce týmu.



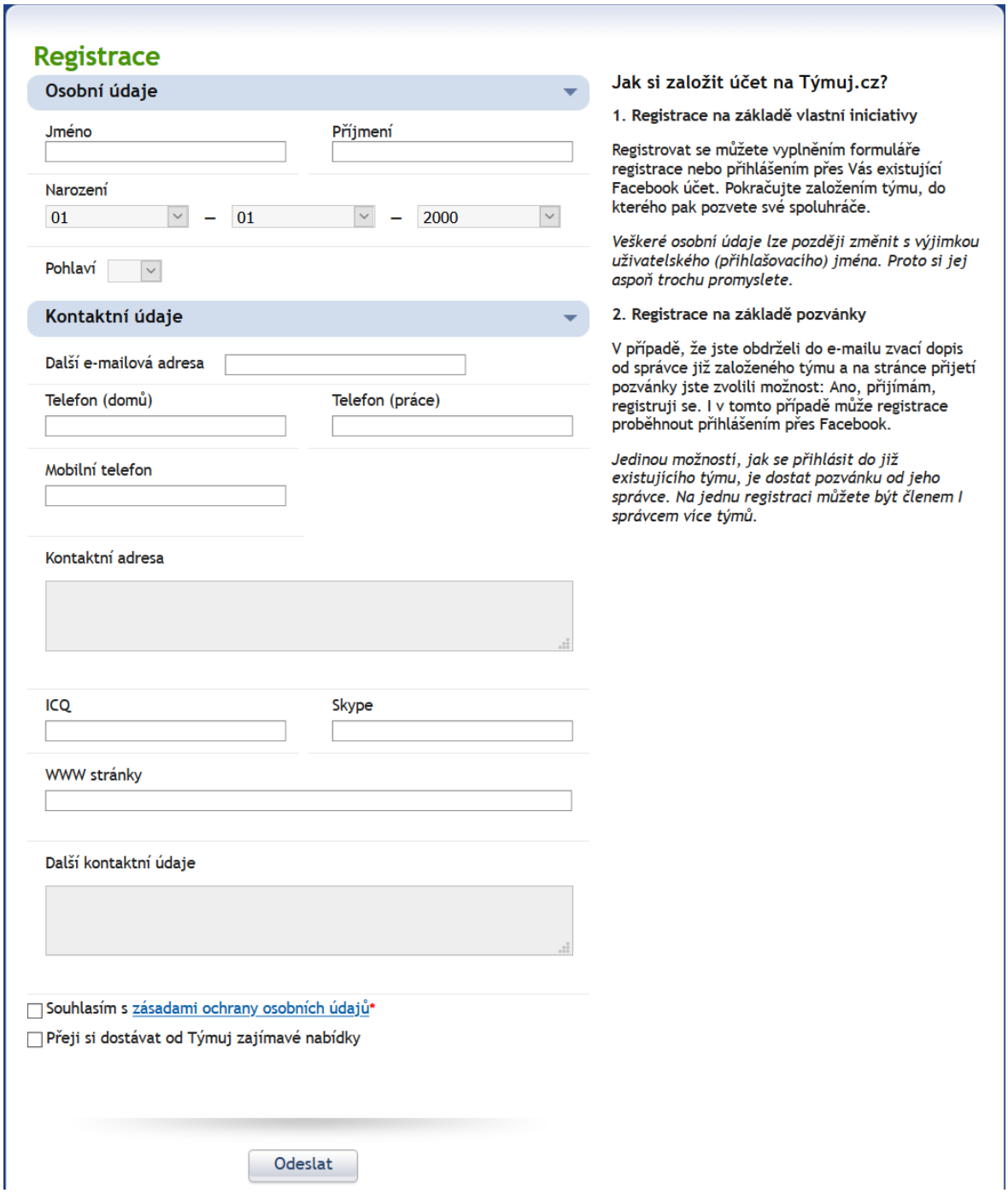
Obrázek 20: Teamstuff.com – nová událost

Správce vytváří nové zápasy nebo události. Událost je v tomto případě nejblíže pravidelně se opakujícímu srazu. Je třeba vyplnit název události a datum konání. Dále může být zvolena perioda opakování. Na výběr má správce z možností každý den a každý týden. V obou případech je třeba také vybrat datum konce opakování. Může být ještě zvolena lokace a další doplňující informace jako doba trvání události a čas příchodu před začátkem události.

Aplikace vyžaduje upgrade pro využití některých základních funkcí. Design webu není responzivní. Je zde opět možnost volby „?“ u potvrzování účasti. Aplikace opět nenabízí možnost generování soupisky zápasů ani přidání hostů na nadcházející událost.

1.5 Tymuj.cz

Jedná se o českou aplikaci pro lepší organizaci aktivit ve sportovních týmech a snadnou komunikaci jejich členů. [4]



Obrázek 21: Tymuj.cz – registrace

U registrace uživatel nejprve vyplní svoji přezdívku, e-mailovou adresu a heslo. Dále je třeba vyplnit ostatní osobní údaje. Pro testovací tým bylo zvoleno heslo „abcd“. Po registraci uživatel obdrží e-mail s jeho údaji.


Přihlašovací údaje Doručená pošta ×

Týmuj.cz <neodpovidat@tymuj.cz>
komu: mně

Dobrý den, [redacted],

děkujeme za Vaši registraci na Týmuj.cz. Dnes je Váš šťastný den:-)

Pro jistotu posíláme Vaše přihlašovací údaje.

Uživatelské jméno: [redacted]
Heslo: abcd

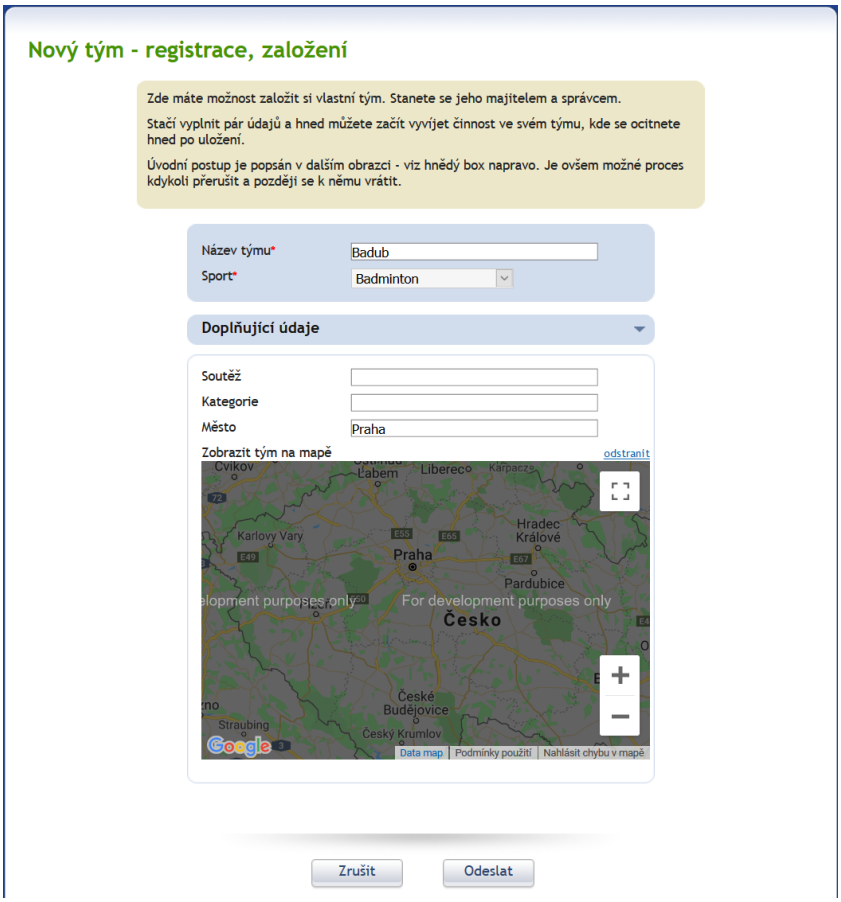
Po přihlášení na Týmuj.cz https://int.tymuj.cz můžete své heslo v osobním profilu změnit.

Úspěšné týmování přeje,

Týmuj.cz

Obrázek 22: Tymuj.cz - registrační e-mail

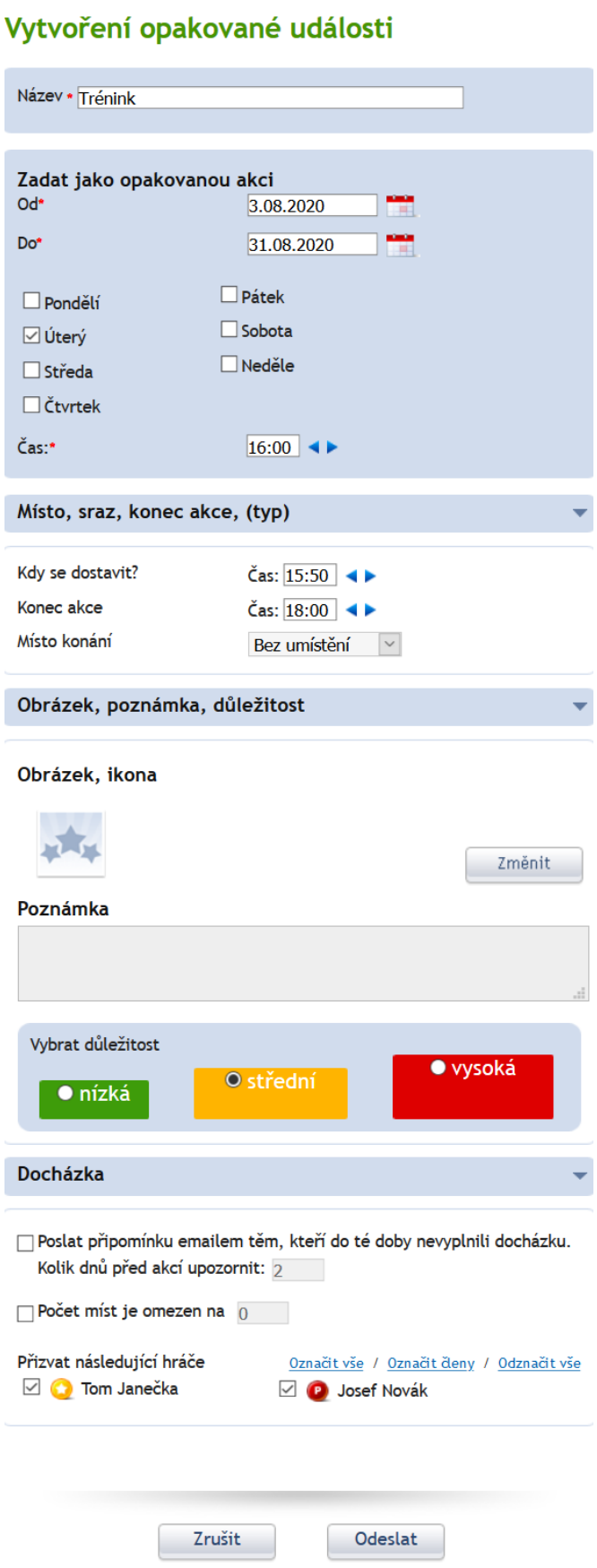
Uživatel je přesměrován na hlavní stránku, kde mu je nabídnuta možnost vytvořit nový tým.



Obrázek 23: Tymuj.cz - nový tým

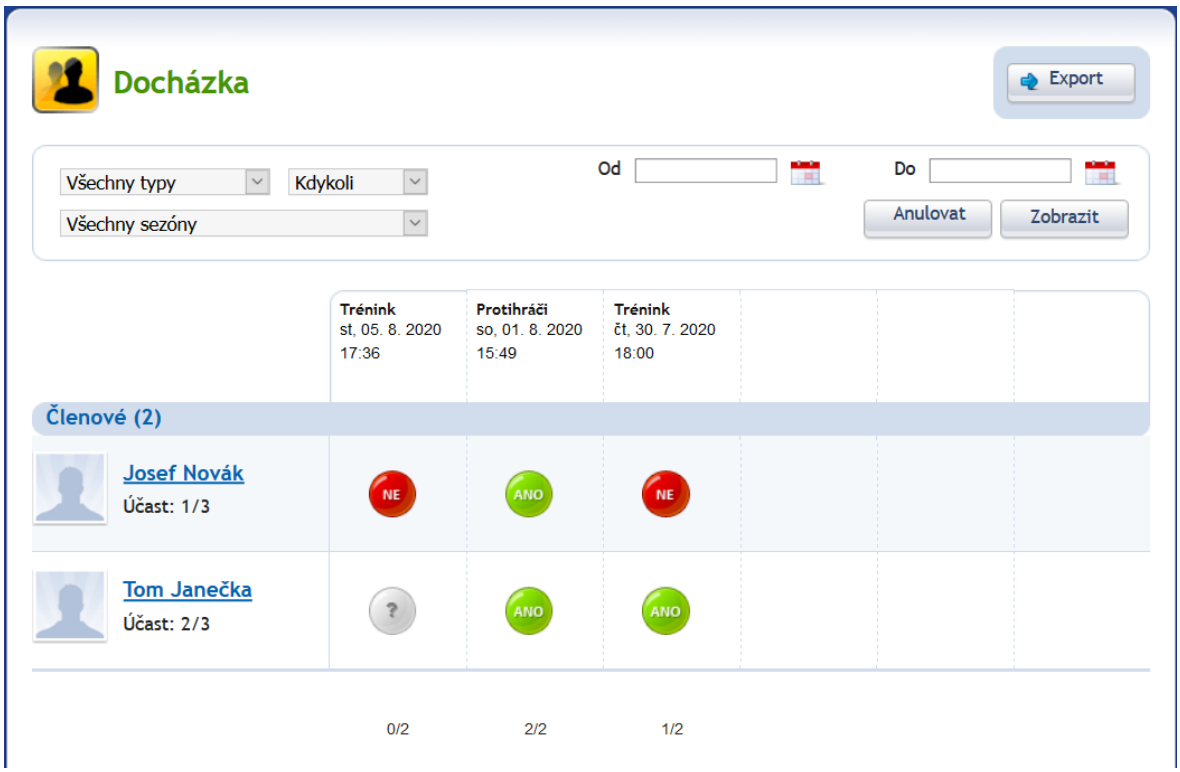
Uživatel je po úspěšném vytvoření týmu přesměrován na jeho přehled. Na této stránce jsou zobrazeny nejaktuálnější informace o dění v týmu. Je zde také spousta odkazů k dostupným funkcím a také popis každé funkce. Uživatelům je k dispozici výpis členů týmu. Každý člen může po kliknutí na ikonu u svého jména upravit své údaje. Správce může navíc členy odstranit.

Správce může vytvořit zápas nebo událost. U tvorby události existuje možnost „opakované události“. Opakovaná událost je v tomto případě nejblíže pravidelně se opakujícímu srazu. Správce musí vyplnit název události, datum začátku a konce opakování události a čas konání. Dále může vybrat jeden nebo více dnů v týdnu. Pokud není vybrán ani jeden den v týdnu a správce klikne na tlačítko „Odeslat“, žádná událost se nevytvoří. Při její tvorbě jsou vybráni členové týmu, kteří se mají události zúčastnit. Členové u každé události vybírají svůj stav účasti ze stavů „Ano“, „Ne“ a „?“. Při výběru libovolného stavu mohou členové ostatním zanechat krátký vzkaz.



Obrázek 24: Tymuj.cz - nová událost

Uživatelé si mohou zobrazit historii účastí na událostech a zápasech. Lze ji filtrovat podle typu, data nebo podle toho, zda událost již proběhla nebo ne.



Obrázek 25: Tymuj.cz – docházka

Správce může vytvářet úkoly, ke kterým jsou přiřazeni požadovaní členové. Dále jsou k dispozici funkce zasílání zpráv ostatním členům v rámci týmu a také galerie fotografií. Fotografii může nahrát každý člen týmu.

Možnost pozvat členy do týmu má k dispozici správce. U výpisu členů klikne na tlačítko „+ Spoluhráč“ a zobrazí se mu formulář pro pozvání člena. Musí zadat jeho e-mailovou adresu. Jméno není povinné. Nakonec má správce na výběr ze tří roli, které může novému členovi přiřadit. Existuje zde role „Host“. Má úplně stejná práva jako člen. Jen je jeho statut oddělen v seznamu uživatelů a pro účely docházky. [5]

Pozvanému členovi je zaslán e-mail s odkazem. Tento odkaz ho přesměruje do týmu. Může pozvání odmítnout nebo přijmout. V případě přijetí je vyžadováno, aby měl vytvořený svůj účet.

Obrázek 26: Tymuj.cz – pozvánka

1.6 Shrnutí

Většina zkoumaných aplikací umožňuje komukoliv si bezplatně vytvořit účet. Nabízejí jak webovou podobu, tak mobilní aplikaci. Aplikace dovolují vytvářet pravidelné události, na kterých můžou členové potvrdit nebo odmítnout svoji účast. Kromě aplikace sejdemse.net nabízí ostatní aplikace také funkce pro vkládání fotografií nebo dokumentů a také kalendář nadcházejících událostí. U některých také existuje možnost evidovat nebo provádět platby.

2 ROZBOR POUŽITÝCH TECHNOLOGIÍ

2.1 C#

Ke zpracování funkcionalit, které se budou vykonávat na straně serveru, byl použit jazyk C#. Jedná se o objektově orientovaný jazyk. Velmi často jsou v něm využívány třídy, které slouží pro popis reálného světa. Tyto třídy obsahují atributy a operace. Existují zde také rozhraní. Rozhraní definují předpisy metod a mohou být implementovány třídami. [6]

Jako editor bylo využito Microsoft Visual Studio 2019 Community.

2.2 ASP.NET Core

Jedná se o multiplatformní framework, který poskytuje prostředky pro vytváření webových aplikací. [7]

Při použití tohoto frameworku se často používá *dependency injection*. Jsou zde k dispozici prostředky pro registraci služeb, které budou při požadavku automaticky dodány. Při registraci služeb se také udává, jestli v rámci celé aplikace bude existovat pouze jedna instance dané služby nebo se při každém požadavku o ni vytvoří nová. [8]

```
services.AddTransient<AccountService>();
services.AddTransient<AdminService>();
services.AddTransient<ParticipantService>();
```

Obrázek 27: Ukázka registrace služeb

Dále při nastavování aplikace můžou být k registraci služeb použity rozšiřující metody, které existují u dané služby. Tímto způsobem je řešena registrace vlastního *DbContextu*. [8]

```
string connectionString =
configuration.GetConnectionString("SQLiteConnection");

services.AddDbContext<MeetupDbContext>(options =>
options.UseSqlite(connectionString));
```

Obrázek 28: Registrace vlastního DbContextu

2.2.1 ASP.NET Core Identity

ASP.NET Core Identity přidává do aplikace možnost využívat uživatelské účty. Uživatelé se přihlašují pomocí uživatelského jména a hesla nebo přes externí systémy, jako je Facebook, Google, Microsoft Account, Twitter a další. [9]

V tomto případě aplikace využívá k přihlášení e-mailu a hesla.

```
services.AddIdentity<Participant,
Role>().AddEntityFrameworkStores<MeetupDbContext>().AddDefaultTokenProviders(
);
```

Obrázek 29: Registrace služby pro správu uživatelských účtů

Pro úpravu parametrů, které jsou požadovány na uživatelské jméno, heslo a další je využita třída *IdentityOptions*.

```
services.Configure<IdentityOptions>(options =>
            {
                options.Password.RequireDigit = true;
                options.Password.RequireUppercase = true;
                options.Password.RequiredLength = 6;
                options.Password.RequireNonAlphanumeric = false;

                options.User.AllowedUserNameCharacters +=
"ěščřžýáíéňóťúůĚŠČŘŽÝÁÍÉÚŮŇŤÓ ";
                options.User.RequireUniqueEmail = true;
            });
```

Obrázek 30: Nastavení parametrů pro heslo a uživatele

2.3 MVC

MVC (Model-View-Controller) je návrhový vzor, který dělí aplikaci na tři různé části: modely, pohledy a kontrolery. Díky tomuto návrhovému vzoru docílíme toho, že každá část se stará o určitou věc. Uživatel komunikuje s kontrolerem, který načítá nebo upravuje data z modelu. Kontroler také vybírá, který pohled zobrazí uživateli a dodává data z modelu do jednotlivých pohledů. Toto rozdělení má zajistit snazší rozšiřování a testování kódu. [10]

V aplikace existují čtyři kontrolery: *AccountController*, *ParticipantController*, *AdminController* a *ErrorController*. AccountController se stará o požadavky týkající se uživatelských účtů, registrace a přihlašování. *ParticipantController* má na starosti poskytnout uživateli funkce, které mu dovolují pracovat s přiřazenými srazy. Je přístupný pouze registrovaným uživatelům s rolí „User“. *AdminController* obsahuje funkce dostupné administrátorovi, mezi které patří například správa vytvořených srazů a přiřazování srazů

k uživatelům. *ErrorController* zobrazuje vlastní chybové stránky v případě, že nastane neošetřená chyba v aplikaci.

```
[AutoValidateAntiforgeryToken]
    public class AccountController : Controller
    {
        private readonly AccountApplicationService accountApplicationService;

        public AccountController(AccountApplicationService
accountApplicationService)
        {
            this.accountApplicationService = accountApplicationService;
        }
        public IActionResult LogIn()
        {
            return View();
        }
        …
}
```

Obrázek 31: Ukázka AccountController

Pohledy bývají soustředěny do složky *Views*. Metoda *View()* se používá pro zobrazení určitého pohledu, který existuje ve struktuře projektu. Může jí být poskytnut název pohledu, ale také nemusí. Je zde využito pojmenovávací konvence. [11]

Pokud je v kontroleru *AccountController* zavolána metoda *LogIn()*, která vrací metodu *View()*, pak se očekává, že ve složce *Views* existuje složka *Account*, která obsahuje soubor s názvem *LogIn*.

2.4 Entity Framework Core

Entity Framework Core je odlehčená, rozšiřitelná, open-source a multiplatformní verze populárního Entity Frameworku, který pracuje s daty. Umožňuje vývojářům využívat objektů pro přístup k databázi. [12]

Pro přístup k datům byla vytvořena třída *MeetupDbContext*. Dědí od třídy *IdentityDbContext<Participant, Role, int>*, která oproti třídě *DbContext* vytvoří ještě výchozí strukturu tabulek pro uživatelský účet. Uživatel je představen třídou *Participant*, která dědí od třídy *IdentityUser<int>* a role třídou *Role*, která dědí od *IdentityRole<int>*. Datový typ *int* udává datový typ primárního klíče.

```
public class MeetupDbContext : IdentityDbContext<Participant, Role, int>
    {
        public DbSet<Participant> Participants { get; set; }
        public DbSet<Meetup> Meetups { get; set; }
        public DbSet<MeetupParticipantJunction> MeetupParticipantJunction {
get; set; }
        public DbSet<Message> Messages { get; set; }
        public DbSet<Guest> Guests { get; set; }
        public DbSet<HistoryRecord> HistoryRecords { get; set; }

        public MeetupDbContext(DbContextOptions<MeetupDbContext> options) :
base(options)
        {
            Database.EnsureCreated();

            Participants = Set<Participant>();
            Meetups = Set<Meetup>();
            Messages = Set<Message>();
            Guests = Set<Guest>();
            MeetupParticipantJunction = Set<MeetupParticipantJunction>();
            HistoryRecords = Set<HistoryRecord>();
        }
    }
```

Obrázek 32: MeetupDbContext

Kolekce *DbSet<T>* reprezentuje jednu tabulku a poskytuje funkce pro čtení, vkládání, úpravu a mazání dat. Vytvoření property *DbSet<Participant> Participants* zajistí, že bude vytvořena tabulka s názvem *Participants* a její sloupce budou odpovídat propertám třídy *Participant*. Vlastnosti sloupců lze upravovat umístěním atributů nad property ve třídě. Třída také obsahuje *navigační property* pro usnadnění přístupu k souvisejícím datům v jiných tabulkách.

```
public class Participant : IdentityUser<int>
    {
        [Required]
        public string FirstName { get; set; }
        [Required]
```

```
        public string LastName { get; set; }

        public Gender Gender { get; set; }

        public IList<Guest> Guests { get; set; }


        public IList<MeetupParticipantJunction> Junctions { get; set; }

    }
```

Obrázek 33: Třída Participant

2.5 SQLite

SQLite je knihovna, která implementuje relační databázový systém SQL. Může být použita jak pro soukromé, tak komerční účely zcela bezplatně. Jedná se o nejvíce nasazovanou databázi na světě. Celá databáze se všemi tabulkami, indexy a pohledy je obsažená v jediném souboru. [13]

Jedním ze zásadních rozdílů oproti klasickým SQL databázím je omezený počet datových typů. Nabízí jich pět. Patří mezi ně datový typ *null, integer, real, text* a *blob.* Datový typ *bool* a výčtové typy jsou ukládány jako *integer*. Datum a čas pak mohou být uloženy jako *text*, *real* nebo *integer*. [14]

Pro frameworky, které se zabývají práci s daty, bylo při tvorbě projektu třeba stáhnout dodatečné balíčky, aby mohlo být použito úložiště SQLite. Tyto balíčky poté poskytují metody, které jsou využity při registraci služeb.

```
string connectionString =
configuration.GetConnectionString("SQLiteConnection");
            services.AddDbContext<MeetupDbContext>(options =>
options.UseSqlite(connectionString));
services.AddHangfire(c => c.UseSQLiteStorage());
```

Obrázek 34: Ukázka nastavení využití úložiště SQLite

2.6 HTML

Hypertext Markup Language je jazyk, pomocí kterého definujeme strukturu našeho dokumentu. Skládá se z prvků, které obsahují určité části našeho obsahu. Prvky udávají, jak je jejich tělo prezentováno. Většina prvků má otevírací a zavírací značku. Prvky také obsahují atributy. Atributy obsahují dodatečné informace o prvku, u kterých není třeba, aby se nacházely v těle prvku. Prvek může obsahovat ve svém těle další prvky. Každý dokument se skládá z několika základních prvků. V prvku *<html></html>* jsou umístěna všechna data.

Je také označován jako kořenový prvek. V prvku *<head></head>* se nachází příkazy pro zahrnutí dalších souborů. Tento prvek obsahuje to, co není třeba, aby uživatel viděl. V prvku *<title></title>* se nachází titulek stránky. Prvek *<body></body>* obsahuje vše, co chceme uživateli prezentovat. [15]

Při tvorbě webových stránek u projektu bylo navíc využito syntaxe Razor. Jedná se o využití jazyku C# pro usnadnění vygenerování HTML kódu.

```
@Html.DropDownList("Gender", new List<SelectListItem>
{
new SelectListItem{ Value = Gender.Male.ToString(), Text =
MeetupResources.MaleName, Selected = true},
new SelectListItem{ Value = Gender.Female.ToString(), Text =
MeetupResources.FemaleName}
}
, new { @class = "form-control", name = "Gender" })
```

Obrázek 35: Ukázka syntaxe Razor

Každý příkaz syntaxe Razor začíná znakem @. Pokud je třeba více než jednoho příkazů, je třeba využít za znak @ přidat složené závorky. Uvnitř nich lze použít jazyk C#.

```
@if (Model.IsActive)
            {
                <span style="color: limegreen;"><i class="fa fa-check-circle
fa-lg"></i></span>
            }
            else
            {
                <span style="color: red;"><i class="fa fa-times-circle fa-
lg"></i></span>
            }
```

Obrázek 36: Ukázka podmínky v syntaxi Razor

2.7 JavaScript / jQuery

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, který se využívá ke tvorbě interaktivních webových stránek. Je interpretován na straně klienta. Dokáže přistoupit k prvkům v HTML souboru a upravit je. Také dokáže reagovat na vstup od uživatele. [16]

jQuery je knihovna JavaScriptu, která usnadňuje přístup k HTML prvkům, zpracování událostí, animace a asynchronní požadavky. [17]

V projektu je v mnoha případech využito asynchronního dotazování, zejména při odesílání formulářů. Toto nám dovoluje získat data ze serveru a prezentovat je uživateli, aniž by došlo k přesměrování stránky. Je zde také možnost deklarovat různé funkce pro úspěšný nebo neúspěšný požadavek.

```
$.ajax({
                    method: 'POST',
                    url: '@Url.Action("DeleteMessage",
MeetupResources.AdminControllerName)',
                    data: { messageId: messageId, __RequestVerificationToken:
token },
                    success: function (data) {
                        if (data.succeeded) {
                            messageRow.remove();
                        }
                        else
                            $('.failedAlert').css('display',
'flex').children('.alertText').text(data.errorMessage);
                    }
                });
```

Obrázek 37: Ukázka asynchronního požadavku

2.8 CSS

Jedná se o jazyk, pomocí kterého nastavujeme vzhled HTML prvků. Usnadňuje práci, protože můžeme styly přiřadit k více prvkům najednou. Definice stylů mohou být uloženy v externích souborech. [18]

Existuje několik způsobů, jak identifikovat určitý prvek. Lze využít název samotného prvku, id selektor nebo selektor třídy. K identifikaci všech prvků se používá znak *. Pokud chceme přiřadit jeden styl více druhům prvků, oddělíme jejich názvy čárkou. [18]

V projektu existuje několik externích souborů se styly. Prvním je soubor *site.css*. Tento soubor se vytvoří automaticky při vytvoření webové aplikace v Microsoft Visual Studiu. Další je vlastní soubor, který zajišťuje tmavý mód.

```
body {
    background-color: rgb(50, 50, 50);
    color: white;
}
```

Obrázek 38: Vlastní soubor se styly - darkStyle.css

Následuje soubor, ve kterém jsou vytvořeny vlastní styly pro určité prvky. Je zde stanoven vzhled a také změna velikosti prvku při změně velikosti stránky. Styly by měly být soustředěny do externích souborů kvůli znovupoužitelnosti. Můžeme je ale psát přímo k prvkům. Tento zápis má je specifičtější, a proto má větší důležitost. [19]

```
<div class="inviteSuccess" style="display: none; color:
forestgreen;">Pozvánka byla odeslána <i class="fa fa-check-circle"></i></div>

<div class="inviteFailed" style="display: none; color: lightcoral">Chyba při
odesílaní pozvánky <i class="fa fa-times-circle"></i></div>
```

Obrázek 39: Ukázka stylu u prvku

```
…
.meetup {
    max-width: 20rem;
    border: 2px solid white;
    border-radius: 6px;
    background-color: rgb(70,70,70);
    margin: 10px;
    display: inline-block;
}

    .meetup .header {
        padding-left: 5px;
        border-bottom: 1px solid white;
        display: flex;
    }

        .meetup .header .activityItem {
            display: flex;
            justify-content: flex-end;
        }

            .meetup .header .activityItem .notActive {
                padding: 5px;
                border-top-right-radius: 6px;
                background-color: red;
                border-left: 1px solid white;
                align-items: center;
                display: flex;
```

```
            }

            .meetup .header .activityItem .active {
                padding: 5px;
                border-top-right-radius: 6px;
                background-color: limegreen;
                border-left: 1px solid white;
                align-items: center;
                display: flex;
            }

    .meetup .meetupName {
        padding-right: 20px;
        display: flex;
        width: 100%;
    }

    .meetup .meetupItem {
        padding-right: 5px;
        padding-left: 5px;
        border-bottom: 1px solid white;
        display: block;
    }
…
```

Obrázek 40: Vlastní soubor se styly - customUtilities.css

2.9 Microsoft Azure

Aplikace je nasazena na serverech služby Microsoft Azure. Pro jejich využití je třeba vytvoření bezplatného účtu.

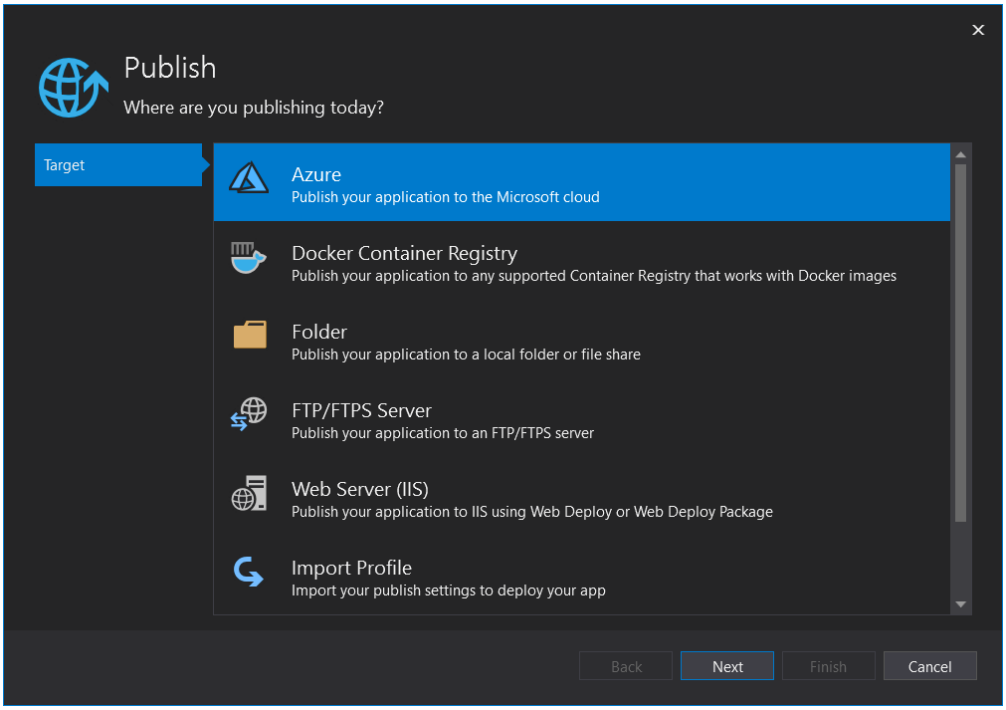
Jsou nabízeny určité služby, které jsou poskytovány každému, kdo má vytvořený účet. Patří mezi ně možnost si pomocí jednoho účtu vytvořit deset webových aplikací na bezplatné úrovni. [20]

Prvním krokem po vytvoření účtu je vytvoření *resource group*. Jedná se o souhrn prostředků, které souvisí se samotnou webovou aplikací. Může obsahovat všechny prostředky projektu, nebo pouze ty, které chceme spravovat jako skupinu. Také se zde nachází metadata o prostředcích. [21]

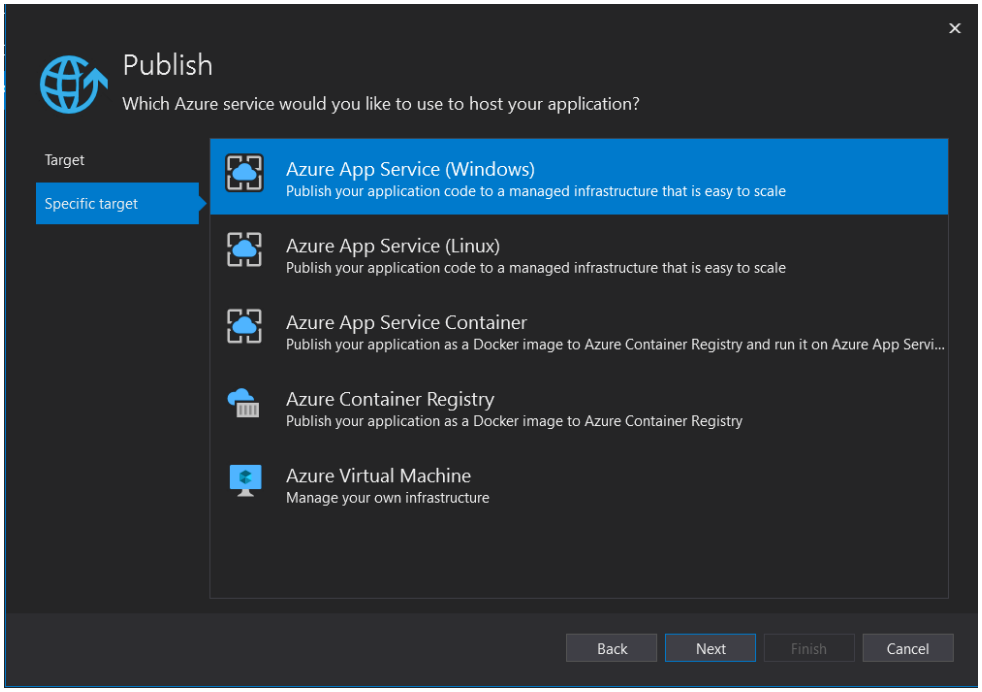
Další krok je vytvoření hostovacího plánu. Hostovací plán definuje soubor výpočetních prostředků, které jsou využity pro webové aplikace. Při jeho tvorbě je třeba si vybrat oblast na světě, kde budou tyto prostředky existovat. Podle toho, zda je zvolena placená úroveň, může být zvolen počet virtuálních počítačů, které poskytnou své prostředky a také jejich velikost. [22]

Po vytvoření hostovacího plánu můžeme vytvořit *app service*. Jedná se službu založenou na protokolu HTTP, která slouží k hostování webových aplikací, REST API a mobilních back-endů. Aplikace mohou běžet v prostředích založených na Linuxu nebo Windowsu. U každého *app service* je vybrán hostovací plán. [23]

U tohoto projektu byla zvolena bezplatná úroveň hostovacího plánu. Po přípravě prostředí lze naši webovou aplikace snadno nasadit pomocí Microsoft Visual Studia. Při prvním nasazení je třeba vytvořit profil.

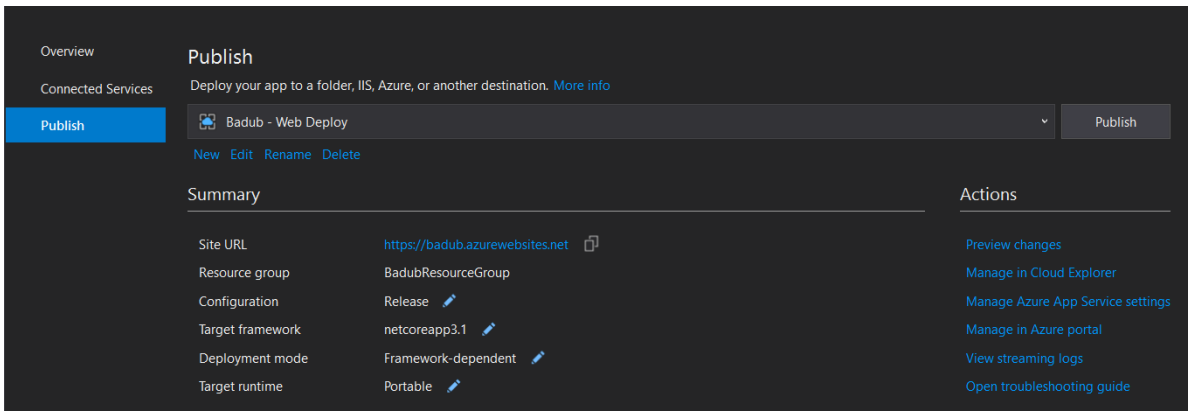


Obrázek 41: Vytvoření profilu 1.



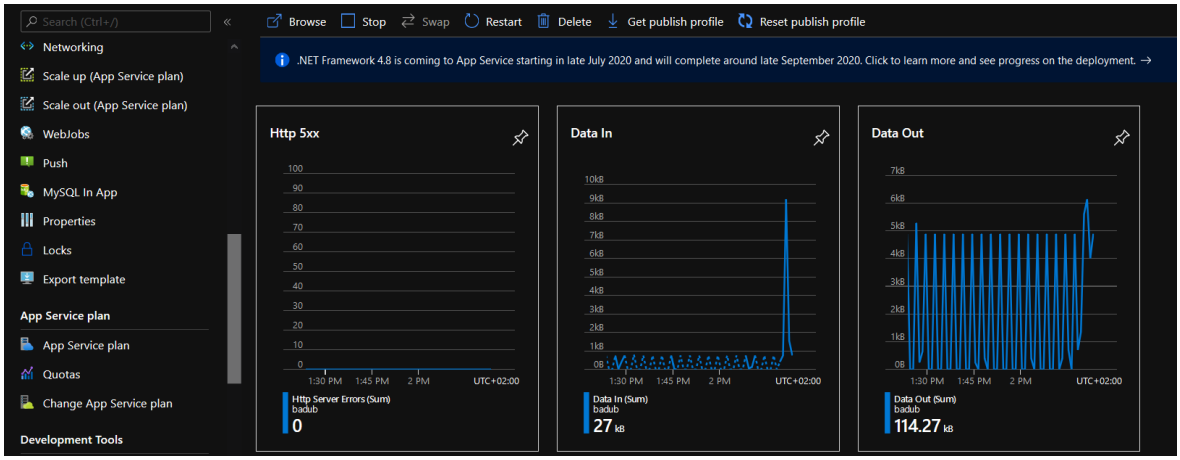
Obrázek 42: Vytvoření profilu 2.

V posledním kroku tvorby profilu je třeba se přihlásit pomocí svého účtu a následně vybrat požadovaný existující *app service*. Následně je vytvořen profil, který je využívám pro všechna následující nasazení.

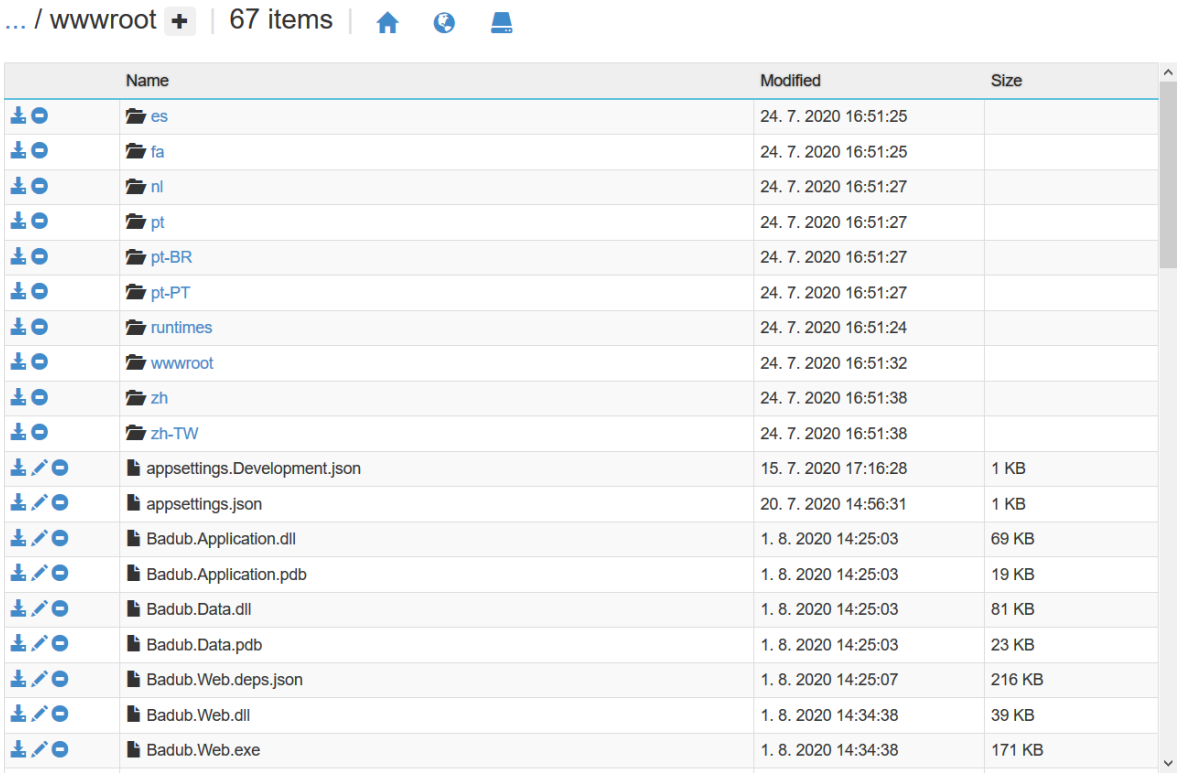


Obrázek 43: Profil pro nasazení

Na portálu Microsoft Azure je nám k dispozici přehled o běžícím *app service*. Z tohoto místa lze aplikace zastavit, restartovat nebo smazat. Jsou zde dostupné grafy četnosti požadavků, které byly zpracovány nebo grafy četnosti výskytu chyb. Existuje zde také služba pro procházení souborové struktury aplikace.



Obrázek 44: Přehled aktivity v aplikaci



Obrázek 45: Přehled souborů v aplikaci

2.10 Hangfire

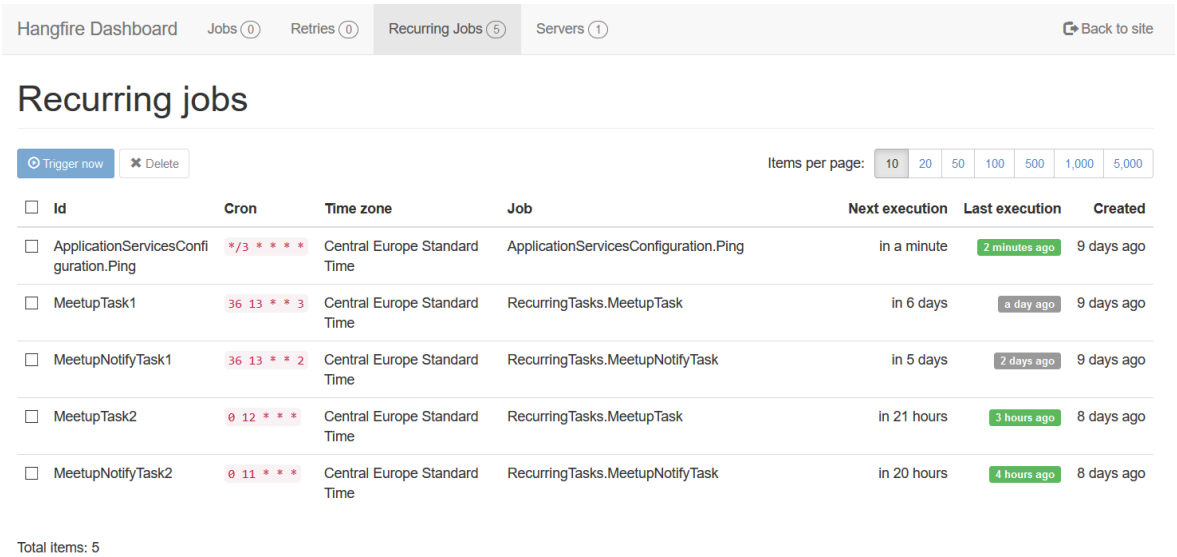
Hangfire je framework, který umožňuje vytvářet, zpracovávat a spravovat úlohy běžící v pozadí. Může se jednat o zaslání hromadných zpráv, údržba databáze nebo generace pravidelných výpisů o událostech. Jsou podporovány úlohy běžící krátkou i dlouhou dobu. Může se také jednat o jednorázové nebo pravidelně se opakující úlohy. Informace o těchto úlohách jsou uloženy v perzistentním úložišti našeho výběru. Pokud při průběhu úlohy dojde

k chybě, budou po krátké pauze provedeny další pokusy. Tento framework také nabízí službu ke sledování naplánovaných, probíhajících a provedených úloh. [24]

V aplikace je tento framework využit k provedeních pravidelně se opakujících úloh. První úloha je provedena právě v čas konání srazu. Druhá úloha je provedena před konáním srazu a zabezpečuje poslání upozornění na blížící se sraz. Jako perzistentní úložiště je opět využito SQLite. Pro využití této možnosti bylo třeba do projektu přidat dodatečný balíček. Služba pro sledování aktivity úloh vyžaduje určitou formu autentizace. Z toho důvodu byl přidán další balíček, který implementuje autentizaci pomocí jména a hesla.

```
services.AddHangfire(c => c.UseSQLiteStorage());

app.UseHangfireServer();
var hangfireSection = configuration.GetSection("HangfireDashboard");
string username = hangfireSection.GetValue<string>("Username");
string password = hangfireSection.GetValue<string>("Password");

app.UseHangfireDashboard(options: new DashboardOptions
{
    Authorization = new[] { new HangfireCustomBasicAuthenticationFilter {
User = username, Pass = password } }
});
```

Obrázek 46: Registrace a nastavení frameworku Hangfire



Obrázek 47: Výpis úloh

3 ZABEZPEČENÍ APLIKACE

3.1 SQL Injection

Je to technika vkládání kódu, která může způsobit poškození databáze. Jedná se o vložení SQL příkazu do již existujícího SQL dotazu pomocí formuláře na webové stránce. Po úspěšném útoku může dojít k přečtení citlivých dat z databáze nebo k jejich úpravě nebo smazání. K útoku může dojít, pokud je SQL dotaz dynamicky konstruován. [25]
Tomuto útoku lze předejít použitím dotazů s parametry. Další možností je využití ORM frameworků pro komunikaci s databází. [26]

```
public IEnumerable<Guest> GetGuests(int meetupId)
        {
            return dbContext.Guests.Where(g => g.MeetupId ==
meetupId).AsEnumerable();
        }
```

Obrázek 48: Ukázka dotazu

Při použití LINQ výrazů nedochází ke konstrukci dotazu manipulací s řetězcem nebo konkatenací, a proto tyto dotazy nejsou napadnutelné tradičními útoky SQL injection. [27]

3.2 CSRF

Jedná se o útok, kdy podvodný web pošle požadavek na zranitelnou stránku, kde je momentálně přihlášený uživatel. K útoku může dojít, pokud se přihlášený uživatel neodhlásí a je přesměrován na jinou stránku. Z této nové stránky je poté poslán požadavek na původní, zranitelný web. ASP.NET MVC využívá tokeny proti padělání pro zabránění těchto útoků. Pokud uživatel odešle formulář, požadavek musí obsahovat dva tokeny. Jeden ve formě cookies a druhý ve formě skrytého vstupu. Hodnoty těchto tokenů jsou generovány náhodně. Pokud požadavek neobsahuje oba tokeny, požadavek není povolen. Skryté tokeny jsou automaticky generovány, pokud se na stránce nachází formulář, jehož metoda je nastavena jako POST. Případně lze token vytvořit pomocí příkazu *@Html.AntiForgeryToken()*. [28]

```
function deleteGuest() {
            var deleteButton = $(this);
            var guestId = deleteButton.val();
            var token = $('input[name="__RequestVerificationToken"]').val();
```

```
            deleteButton.prop('disabled', true);

            $.ajax({
                method: 'POST',
                url: '@Url.Action("DeleteGuest",
MeetupResources.ParticipantControllerName)',
                data: { guestId: guestId, __RequestVerificationToken: token
},
                success: function (data) {
                    if (data.succeeded) {
                        var row =
deleteButton.parents('.participantItem').first();
                        row.remove();
                        setCounts();
                    }
                    else
                        $('.failedAlert').css('display',
'flex').children('.alertText').text(data.errorMessage);
                }
            }).done(() => {
                deleteButton.prop('disabled', false);
            });
        }
```

Obrázek 49: Ukázka odesílání formuláře s CSRF tokenem

Aby došlo ke kontrole tokenů na straně serveru, je nad kontrolery umístěn atribut *AutoValidateAntiforgeryToken*.

3.3 XSS

Jedná se o zranitelnost v aplikaci, která dovolí útočníkovi umístit klientské skripty do webových stránek. Tyto skripty bývají často psány pomocí JavaScriptu. Pokud dojde k zobrazení napadnuté stránky, útočníkův skript je proveden a umožní mu se zmocnit cookies a tokenů napadeného uživatele. Může také dojít k úpravě HTML prvků nebo k přesměrování na jinou stránku. K této zranitelnosti dochází, pokud webová aplikace příjme vstup od uživatele a pošle ho na výstup bez jakékoliv validace těchto dat. Této zranitelnosti lze předejít používáním kódování dat ještě před jejich umístěním do HTML prvku, do atributů HTML prvků nebo do kódu v JavaScriptu. [29]

```
@Html.Encode("\"<\"")
&quot;&lt;&quot;
```

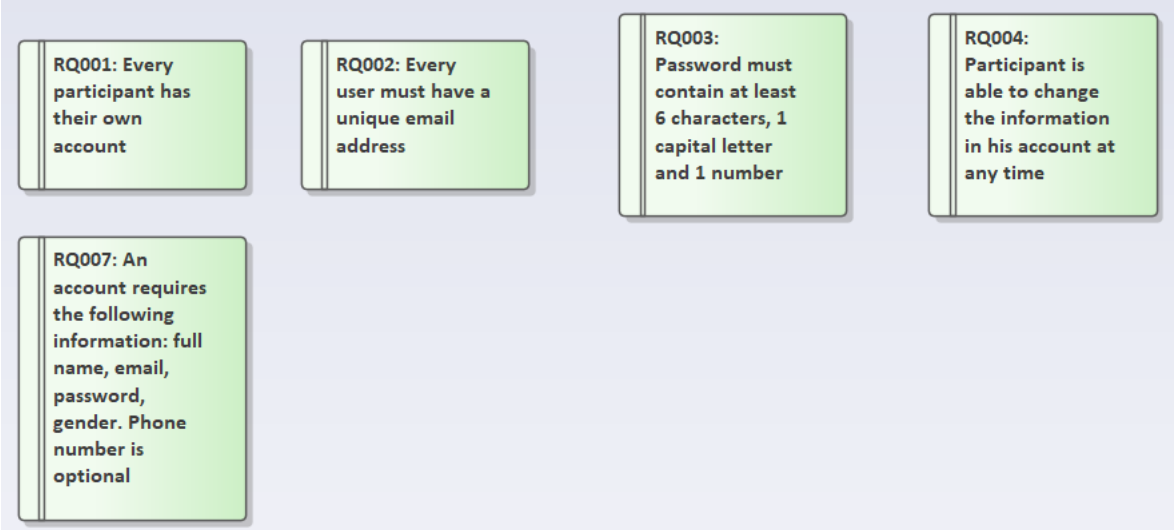
Obrázek 50: Ukázka kódování řetězců

II. PRAKTICKÁ ČÁST

4 REALIZACE

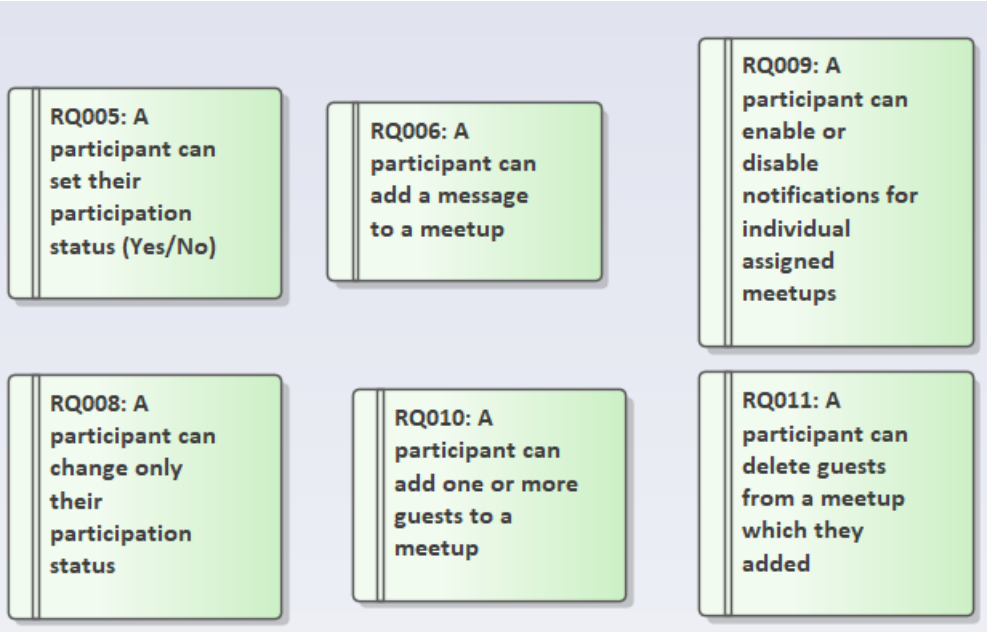
4.1 Požadavky

Každý uživatel má svůj vlastní účet, ke kterému se přihlásí pomocí svého e-mailu a hesla. E-maily v aplikaci musejí být unikátní. Uživatel má možnost kdykoliv změnit své údaje. Dále jeho účet uchovává jeho celé jméno, pohlaví a volitelně i telefonní číslo.



Obrázek 51: Požadavky na účet uživatele

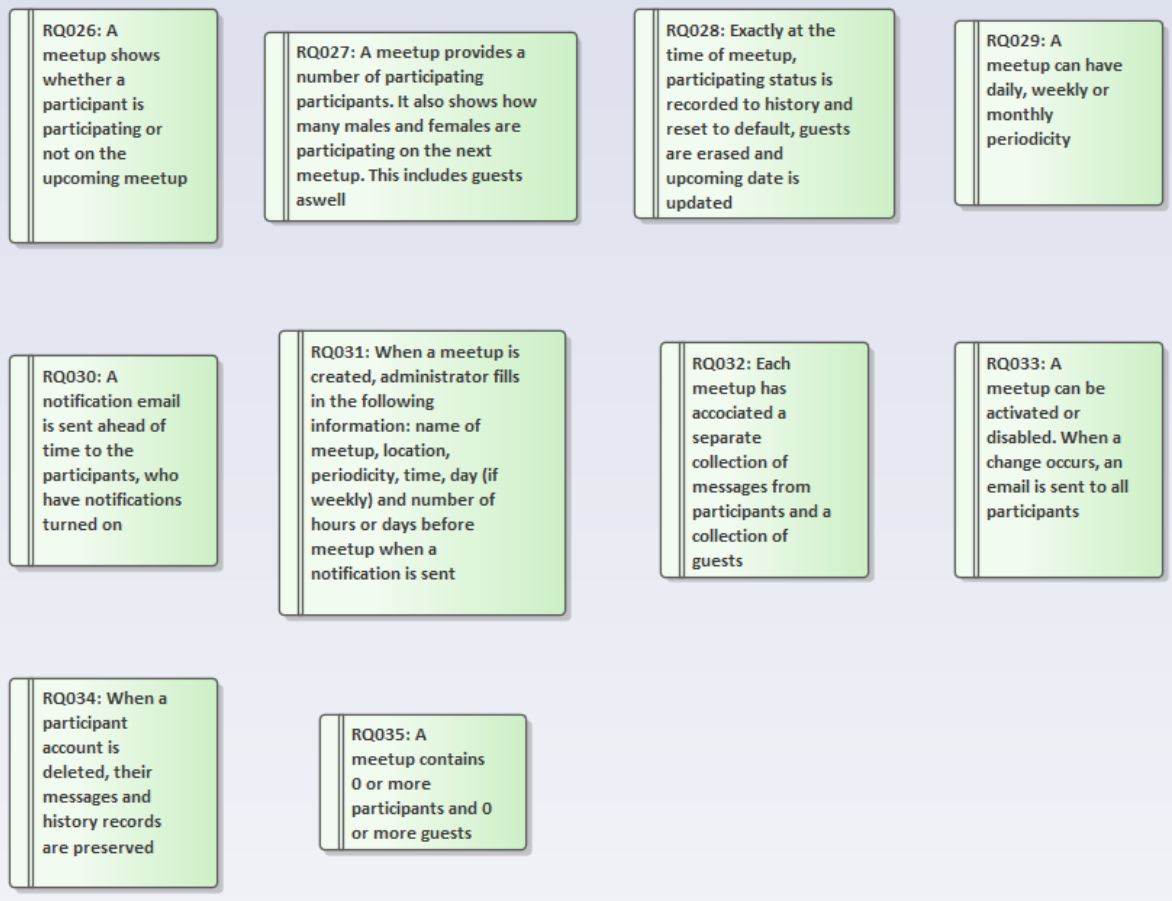
Aplikace umožňuje vytvořit jeden či více srazů. K těmto srazům můžou být přiřazeni uživatelé. Každý účastník srazu může nastavit svůj stav účasti. Na výběr má ze dvou možných stavů: „Ano“ a „Ne“. Každý účastník může ke srazu přidat jednoho nebo více hostů. Tímto dá ostatním najevo, že na příštím srazu mají očekávat další spoluhráče. Účastník může odstranit ty hosty, které sám přidal. Účastník dále může do srazu poslat zprávu, kterou vidí všichni ostatní účastníci. Tato zpráva zůstane ve srazu i po odstranění účtu uživatele. Účastník také může vypnout nebo zapnout upozornění na nadcházející srazy, které jsou pravidelně zasílány na e-mail.



Obrázek 52: Požadavky účastníka

Stránka jednoho srazu zobrazuje seznam účastníků. U každého jde vidět, zda se účastní nadcházejícího srazu. Také jsou poskytnuty údaje o srazu samotném, jako například jeho název, místo konání, čas, datum a den v týdnu nadcházejícího srazu. V neposlední řadě je zde také souhrn údajů o přihlášených účastnících a nahlášených hostech. V přehledu lze zjistit, kolik celkem účastníků má nastavenou účast do stavu „Ano“. Tito účastníci jsou rozděleni podle pohlaví. Údaje o počtu účastníků a hostů může být rozhodující v tom, zda k nadcházejícímu srazu vůbec dojde. Každý sraz má svoji kolekci účastníků, hostů a zpráv. Každý sraz se opakuje s určitou periodou. Tato perioda může být denní, týdenní a měsíční. U každého srazu musí být nastaven údaj, s jakým předstihem budou všichni účastníci informováni formou e-mailu o nadcházejícím srazu. Perioda opakování srazu udává, zda je tento údaj zadán v hodinách nebo dnech. E-mail je zaslán pouze těm účastníkům, kteří mají zapnuté upozornění u konkrétního srazu. Přesně v čas konání srazu jsou zaznamenány údaje o účasti jednotlivých účastníků do historie, jsou smazáni hosté z tohoto srazu a stav

účasti každého účastníka je nastaven do výchozího stavu (stav „Ne“).



Obrázek 53: Požadavky na sraz

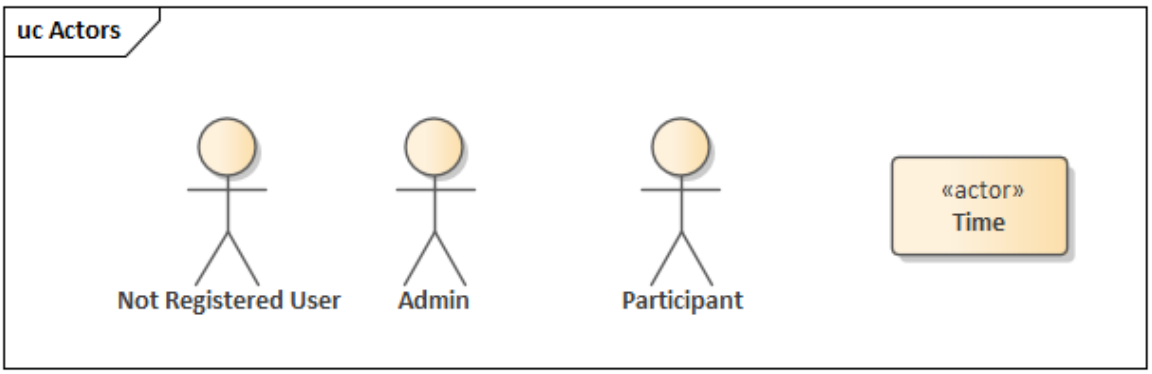
V aplikaci existuje alespoň jeden uživatel s administrátorskými právy. Tato práva jsou automaticky přiřazena uživateli, který se v rámci aplikace zaregistruje jako první. Tento uživatel může přidělit práva i dalším. Administrátor může vytvářet, mazat a upravovat srazy. Administrátor může mazat uživatelské účty a rozesílat zvací e-maily. Administrátor může přiřazovat srazy k uživatelům a také je z nich odebrat. Administrátor může zobrazit detaily i toho srazu, který mu nebyl jako uživateli přiřazen. Administrátor může mazat zprávy účastníků. Pokud administrátor zobrazí sraz, který mu byl přiřazen, má k dispozici stejné funkce, jako další účastníci. Navíc může pro sraz vygenerovat soupisku zápasů, která obsahuje všechny přihlášené účastníky a nahlášené hosty. Administrátor má přístup k historii účastí u jednotlivých srazů. Každý záznam v historii obsahuje jméno účastníka, datum uskutečněného srazu a také stav účasti. Administrátor může poslat hromadné oznámení jednomu nebo více účastníkům na jejich e-mail. Administrátor může deaktivovat nebo aktivovat sraz. Pokud je sraz neaktivní, účastnící nemohou měnit stav své účasti ani přidávat hosty, ale stále do něj mohou posílat zprávy. Při každé změně stavu srazu je všem

účastníkům zaslán informační e-mail a jejich stav je nastaven na „Ne“.

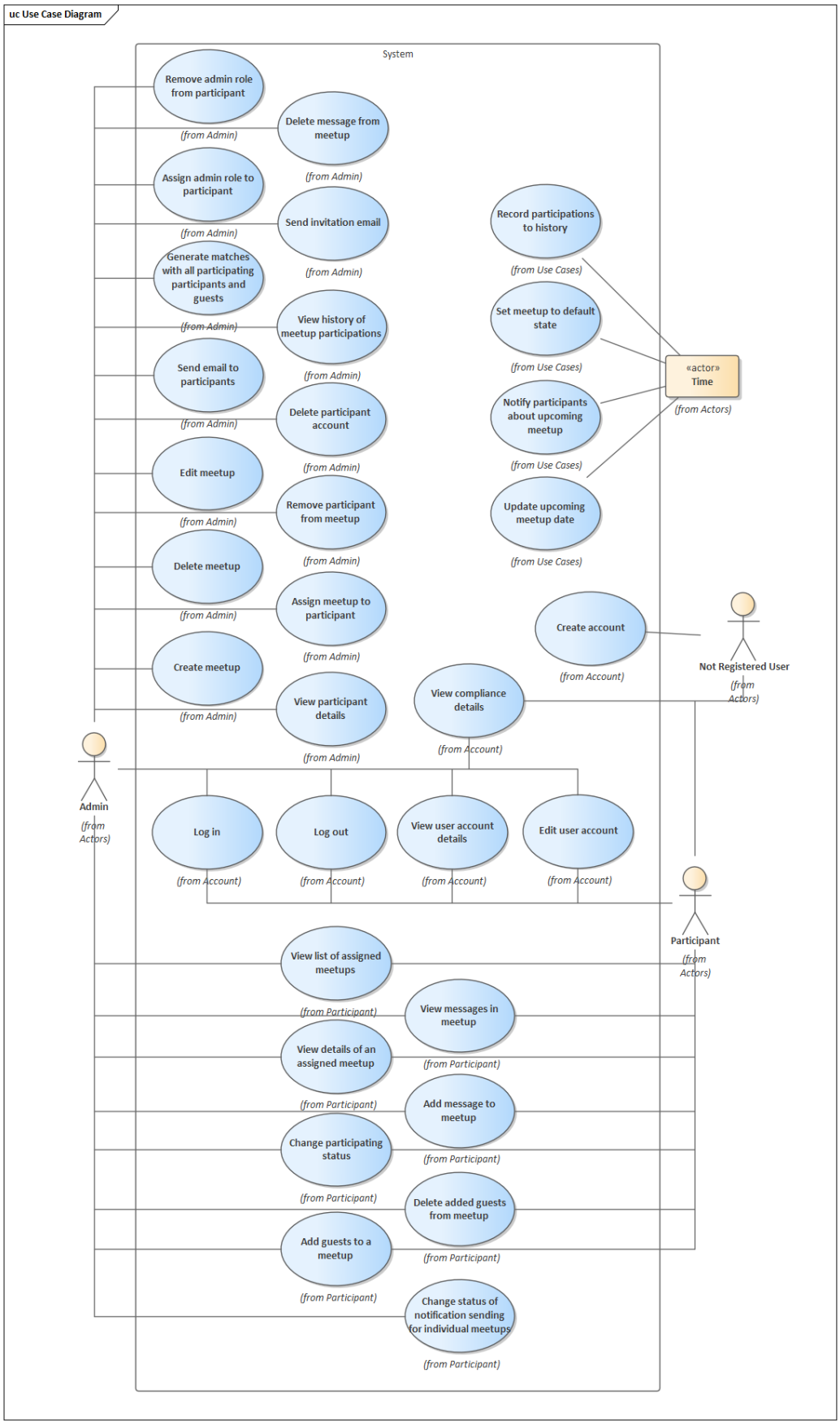
Obrázek 54: Požadavky administrátora

4.2 Use Case

Use case diagram nám dovoluje znázornit aktéry, kteří interagují se systémem. Každý aktér je v určitém vztahu s use casy, které může spouštět. Pomocí use case diagramu můžeme zjistit, zda systém nabízí funkce, které jsme definovali v požadavcích.



Obrázek 55: Aktéři



Obrázek 56: Use case diagram

4.2.1 Scénář: Registrace uživatele

Tabulka 1: Registrace uživatele

Název scénáře	Registrace uživatele
Aktéři	Neregistrovaný uživatel
Hlavní scénář	1. Neregistrovaný uživatel klikne na odkaz, který ho přesměruje na stránku s formulářem pro registraci 2. Neregistrovaný uživatel vyplní povinná pole a klikne na tlačítko „Registrovat“ 3. Systém provede validaci zadaných dat 4. Systém vytvoří uživatelský účet, přiřadí mu příslušné role a je přihlášen do systému
Vedlejší scénáře	3a. Zadané údaje nejsou validní 3b. E-mail je již evidován

4.2.2 Scénář: Zobrazit informace o zpracování osobních údajů

Tabulka 2: Zobrazit informace o zpracování osobních údajů

Název scénáře	Zobrazit informace o zpracování osobních údajů
Aktéři	Neregistrovaný uživatel, účastník nebo administrátor
Hlavní scénář	1. Uživatel klikne na odkaz, který ho přesměruje na stránku s informacemi o zpracování osobních údajů 2. Systém zobrazí stránku s informacemi
Vedlejší scénáře	

4.2.3 Scénář: Přihlášení

Tabulka 3: Přihlášení

Název scénáře	Přihlášení
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel klikne na odkaz, který ho přesměruje na stránku s formulářem pro přihlášení

	2. Uživatel vyplní e-mail a heslo 3. Systém provede validaci zadaných dat 4. Systém provede přihlášení 5. Systém uživatele přesměruje na stránku s výpisem srazů
Vedlejší scénáře	3a. Zadané údaje nejsou validní 4a. Přihlášení se nezdařilo

4.2.4 Scénář: Odhlášení

Tabulka 4: Odhlášení

Název scénáře	Odhlášení
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel klikne na tlačítko pro odhlášení z aplikace 2. Systém provede odhlášení
Vedlejší scénáře	

4.2.5 Scénář: Zobrazení dat účtu

Tabulka 5: Zobrazení dat účtu

Název scénáře	Zobrazení dat účtu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel klikne na tlačítko pro zobrazení informací o jeho účtu 2. Systém zobrazí formulář s údaji o účtu uživatele
Vedlejší scénáře	

4.2.6 Scénář: Změna dat účtu

Tabulka 6: Změna dat účtu

Název scénáře	Změna dat účtu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel klikne na tlačítko pro zobrazení informací o jeho účtu 2. Systém zobrazí formulář s údaji o účtu uživatele

	3. Uživatel klikne na tlačítko vedle požadovaného pole s údajem, který chce změnit 4. Systém odblokuje pole a povolí jeho úpravu 5. Uživatel provede změnu dat a klikne na tlačítko „Uložit“ 6. Systém provede validaci zadaných dat 7. Systém provede změnu dat a zobrazí hlášku o úspěchu provedené změny
Vedlejší scénáře	6a. Zadané údaje nejsou validní 6b. E-mail je již evidován

4.2.7 Scénář: Zobrazení výpisu přiřazených srazů

Tabulka 7: Zobrazení výpisu přiřazených srazů

Název scénáře	Zobrazení výpisu přiřazených srazů
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel klikne na tlačítko „Výpis setkání“ 3. Systém zobrazí seznam přiřazených srazů
Vedlejší scénáře	

4.2.8 Scénář: Zobrazení detailu jednoho srazu

Tabulka 8: Zobrazení detailu jednoho srazu

Název scénáře	Zobrazení detailu jednoho srazu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu
Vedlejší scénáře	2a. Uživatel se pokusí otevřít sraz, který mu nebyl přiřazen

4.2.9 Scénář: Zobrazení zpráv u jednoho srazu

Tabulka 9: Zobrazení zpráv u jednoho srazu

Název scénáře	Zobrazení zpráv u jednoho srazu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu 4. Uživatel klikne na tlačítko „Zprávy“ v liště menu 5. Systém zobrazí stránku se zprávami
Vedlejší scénáře	2a. Uživatel se pokusí otevřít sraz, který mu nebyl přiřazen

4.2.10 Scénář: Přidání zprávy ke srazu

Tabulka 10: Přidání zprávy ke srazu

Název scénáře	Přidání zprávy ke srazu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu 4. Uživatel klikne na tlačítko „Zprávy“ v liště menu 5. Systém zobrazí stránku se zprávami 6. Uživatel klikne na tlačítko „Přidat zprávu“ 7. Systém zobrazí formulář pro novou zprávu 8. Uživatel napíše do formuláře text zprávy a klikne na tlačítko „Přidat“ 9. Systém provede validaci dat 10. Systém přidá zprávu do kolekce zpráv srazu

Vedlejší scénáře	2a. Uživatel se pokusí otevřít sraz, který mu nebyl přiřazen 9a. Délka textu zprávy je mimo hranice

4.2.11 Scénář: Změna stavu účasti

Tabulka 11: Změna stavu účasti

Název scénáře	Změna stavu účasti
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu 4. Uživatel klikne na tlačítko „Ano“ nebo „Ne“ 5. Systém provede změnu stavu účasti u účastníka
Vedlejší scénáře	2a. Uživatel se pokusí otevřít sraz, který mu nebyl přiřazen

4.2.12 Scénář: Přidat hosta ke srazu

Tabulka 12: Přidat hosta ke srazu

Název scénáře	Přidat hosta ke srazu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu 4. Uživatel klikne na tlačítko „Přidat hosta“ 5. Systém zobrazí formulář při vyplnění údajů o hostovi 6. Uživatel vyplní údaje 7. Systém provede validaci dat 8. Systém přiřadí hosta ke srazu
Vedlejší scénáře	2a. Uživatel se pokusí otevřít sraz, který mu nebyl přiřazen

	7a. Zadané údaje nejsou validní

4.2.13 Scénář: Odstranit hosta

Tabulka 13: Odstranit hosta

Název scénáře	Odstranit hosta
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu 4. Uživatel klikne na tlačítko „Odstranit hosta“ 5. Systém odstraní hosta ze srazu
Vedlejší scénáře	2a. Uživatel se pokusí otevřít sraz, který mu nebyl přiřazen

4.2.14 Scénář: Změnit stav posílání notifikací u srazu

Tabulka 14: Změnit stav posílání notifikací u srazu

Název scénáře	Změnit stav posílání notifikací u srazu
Aktéři	Účastník nebo administrátor
Hlavní scénář	1. Uživatel se přihlásí 2. Uživatel si vybere požadovaný přiřazený sraz a klikne na jeho checkbox 3. Systém změní stav notifikací u uživatele
Vedlejší scénáře	

4.2.15 Scénář: Vytvořit sraz

Tabulka 15: Vytvořit sraz

Název scénáře	Vytvořit sraz
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Nový sraz“ v menu

	3. Systém zobrazí formulář pro vytvoření nového srazu 4. Administrátor vyplní údaje 5. Systém provede validaci dat 6. Systém vytvoří nový sraz
Vedlejší scénáře	5a. Zadané údaje nejsou validní

4.2.16 Scénář: Zobrazit informace o uživatelích

Tabulka 16: Zobrazit informace o uživatelích

Název scénáře	Zobrazit informace o uživatelích
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů
Vedlejší scénáře	

4.2.17 Scénář: Přiřadit sraz k uživateli

Tabulka 17: Přiřadit sraz k uživateli

Název scénáře	Přiřadit sraz k uživateli
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů 4. Administrátor vybere požadovaný sraz z dropdown menu a klikne na tlačítko „Přiřadit“ 5. Systém přiřadí sraz k uživateli
Vedlejší scénáře	

4.2.18 Scénář: Odstranit uživatele ze srazu

Tabulka 18: Odstranit uživatele ze srazu

Název scénáře	Odstranit uživatele ze srazu
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů 4. Administrátor vybere požadovaný sraz z dropdown menu a klikne na tlačítko „Odebrat“ 5. Systém odebere uživatele ze srazu
Vedlejší scénáře	

4.2.19 Scénář: Odstranit sraz

Tabulka 19: Odstranit sraz

Název scénáře	Odstranit sraz
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Odstranit sraz“ u požadovaného srazu 3. Systém zobrazí hlášku, zda chce administrátor skutečně sraz odstranit a po jeho souhlasu jej odstraní
Vedlejší scénáře	

4.2.20 Scénář: Upravit sraz

Tabulka 20: Upravit sraz

Název scénáře	Upravit sraz
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Upravit“ u požadovaného srazu

	3. Systém zobrazí formulář s informacemi o srazu 4. Administrátor změní požadované údaje 5. Systém provede validaci dat 6. Systém zaznamená údaje ke srazu
Vedlejší scénáře	5a. Zadané údaje nejsou validní

4.2.21 Scénář: Odstranit uživatelský účet

Tabulka 21: Odstranit uživatelský účet

Název scénáře	Odstranit uživatelský účet
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů 4. Administrátor vybere požadovaného uživatele a klikne u něj na tlačítko „Odstranit uživatele“ 5. Systém zobrazí hlášku, zda chce administrátor skutečně uživatele odstranit a po jeho souhlasu jej odstraní
Vedlejší scénáře	

4.2.22 Scénář: Zaslat hromadnou zprávu účastníkům

Tabulka 22: Zaslat hromadnou zprávu účastníkům

Název scénáře	Zaslat hromadnou zprávu účastníkům
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor si vybere požadovaný sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu

	4. Administrátor klikne na tlačítko „Zpráva účastníkům“ v menu 5. Systém zobrazí formulář pro zaslání hromadné zprávy 6. Administrátor vyplní předmět a znění zprávy a následně vybere účastníky, kterým bude zpráva zaslána 7. Systém provede validaci dat 8. Systém zašle zprávu na e-maily účastníků
Vedlejší scénáře	7a. Zadané údaje nejsou validní

4.2.23 Scénář: Zobrazení historie účastí

Tabulka 23: Zobrazení historie účastí

Název scénáře	Zobrazení historie účastí
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor si vybere požadovaný sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu 4. Administrátor klikne na tlačítko „Historie“ v menu 5. Systém zobrazí stránku se záznamy účastí srazu
Vedlejší scénáře	

4.2.24 Scénář: Vygenerování soupisky zápasů

Tabulka 24: Vygenerování soupisky zápasů

Název scénáře	Vygenerování soupisky zápasů
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor si vybere požadovaný sraz z výpisu setkání a klikne na tlačítko „Otevřít“ 3. Systém zobrazí stránku s informacemi o srazu

	4. Administrátor vyplní pole „Počet zápasů“ a klikne na tlačítko „Generovat soupisku“ 5. Systém zobrazí stránku s tabulkou vygenerovaných zápasů
Vedlejší scénáře	4a. Je zadána hodnota mimo rozsah 4b. Počet účastníků se stavem „Ano! nebo hostů je menší než 2

4.2.25 Scénář: Zaslání zvací zprávy

Tabulka 25: Zaslání zvací zprávy

Název scénáře	Zaslání zvací zprávy
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů 4. Administrátor klikne na tlačítko „Pozvat uživatele“ 5. Systém zobrazí formulář pro zaslání zvacího e-mailu 6. Administrátor vyplní e-mail uživatele, kterého chce pozvat a může také změnit již předvyplněnou zprávu nebo předmět zprávy a klikne na Tlačítko „Pozvat“ 7. Systém provede validaci dat 8. Systém zašle zprávu na zadaný e-mail a následně zobrazí hlášku o úspěchu odeslání
Vedlejší scénáře	7a. Zadané údaje nejsou validní

4.2.26 Scénář: Přiřadit uživateli roli administrátora

Tabulka 26: Přiřadit uživateli roli administrátora

Název scénáře	Přiřadit uživateli roli administrátora
Aktéři	Administrátor

Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů 4. Administrátor vybere požadovaného uživatele a klikne u něj na tlačítko „Přidat administrátora“ 5. Systém přiřadí uživateli roli administrátora
Vedlejší scénáře	

4.2.27 Scénář: Odebrat uživateli roli administrátora

Tabulka 27: Odebrat uživateli roli administrátora

Název scénáře	Odebrat uživateli roli administrátora
Aktéři	Administrátor
Hlavní scénář	1. Administrátor se přihlásí 2. Administrátor klikne na tlačítko „Výpis uživatelů“ v menu 3. Systém zobrazí stránku s výpisem všech evidovaných uživatelů, včetně jejich kontaktních údajů a seznamem přiřazených a nepřiřazených srazů 4. Administrátor vybere požadovaného uživatele a klikne u něj na tlačítko „Odebrat“ 5. Systém odebere uživateli roli administrátora
Vedlejší scénáře	

4.2.28 Scénář: Upravit datum následujícího srazu

Tabulka 28: Upravit datum následujícího srazu

Název scénáře	Upravit datum následujícího srazu
Aktéři	Čas
Hlavní scénář	1. Případ užití začíná právě v datum a čas uskutečnění srazu

	2. Systém zjistí periodu srazu a datum následujícího uskutečnění inkrementuje o určitou hodnotu
Vedlejší scénáře	

4.2.29 Scénář: Zaslat účastníkům upozornění na nadcházející sraz

Tabulka 29: Zaslat účastníkům upozornění na nadcházející sraz

Název scénáře	Zaslat účastníkům upozornění na nadcházející sraz
Aktéři	Čas
Hlavní scénář	1. Případ užití začíná právě v datum a čas, které jsou určeny odečtením hodnoty od data nadcházejícího srazu. Tato hodnota může být zadána ve dnech nebo hodinách 2. Systém zašle upozornění na e-mail všech účastníků srazu
Vedlejší scénáře	2a. Sraz není aktivní 2b. Účastník má vypnutý stav upozornění u srazu

4.2.30 Scénář: Nastavení výchozích hodnot srazu

Tabulka 30: Nastavení výchozích hodnot srazu

Název scénáře	Nastavení výchozích hodnot srazu
Aktéři	Čas
Hlavní scénář	1. Případ užití začíná právě v datum a čas uskutečnění srazu 2. Systém vymaže všechny hosty ze srazu a u každého účastníka nastaví stav účasti na „Ne“
Vedlejší scénáře	2a. Sraz není aktivní

4.2.31 Scénář: Zaznamenání účastníka do historie srazu

Tabulka 31: Zaznamenání účastníka do historie srazu

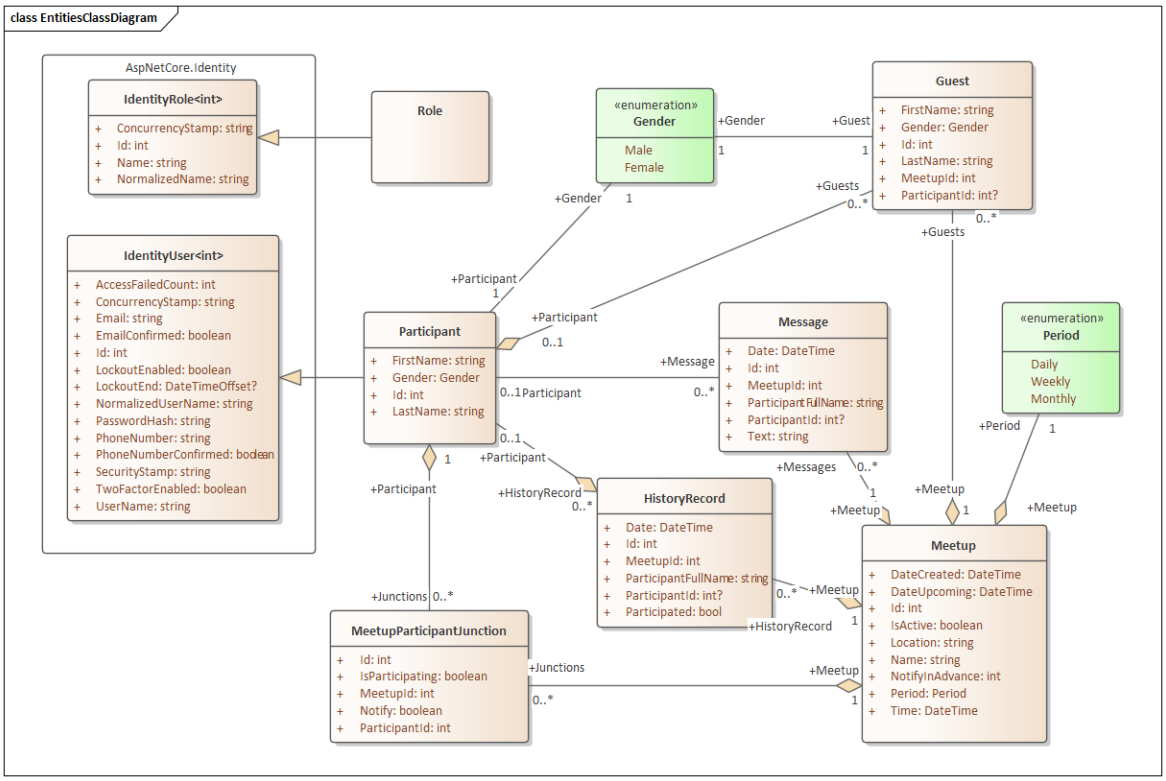
Název scénáře	Zaznamenání účastníka do historie srazu
Aktéři	Čas
Hlavní scénář	1. Případ užití začíná právě v datum a čas uskutečnění srazu 2. Systém zaznamená stav každého účastníka srazu do historie, včetně

	jeho jména a data uskutečnění srazu
Vedlejší scénáře	2a. Sraz není aktivní

4.3 Modely tříd

Po ujasnění funkcionalit systému jsou vytvořeny třídy, které reprezentují aktéry a také další entity, jako třeba jeden sraz, zpráva nebo záznam v historii. Do diagramu jsou také zahrnuty třídy webového projektu, které se slouží ke zpracování požadavků uživatelů a poskytují odpovídající HTML stránky.

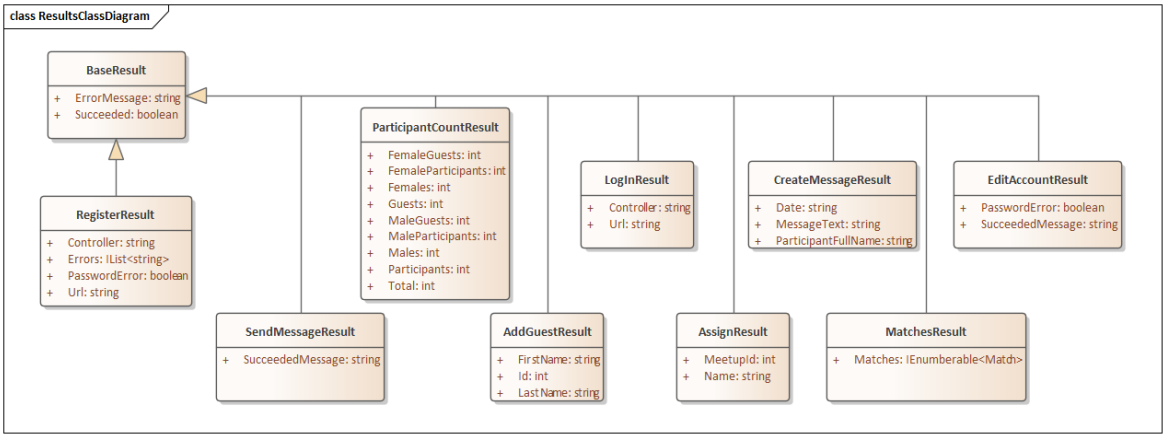
Vytvoření tříd pro entity by se také dalo považovat za návrh struktury databáze, protože je použit způsob návrhu code-first.



Obrázek 57: Diagram tříd – entity

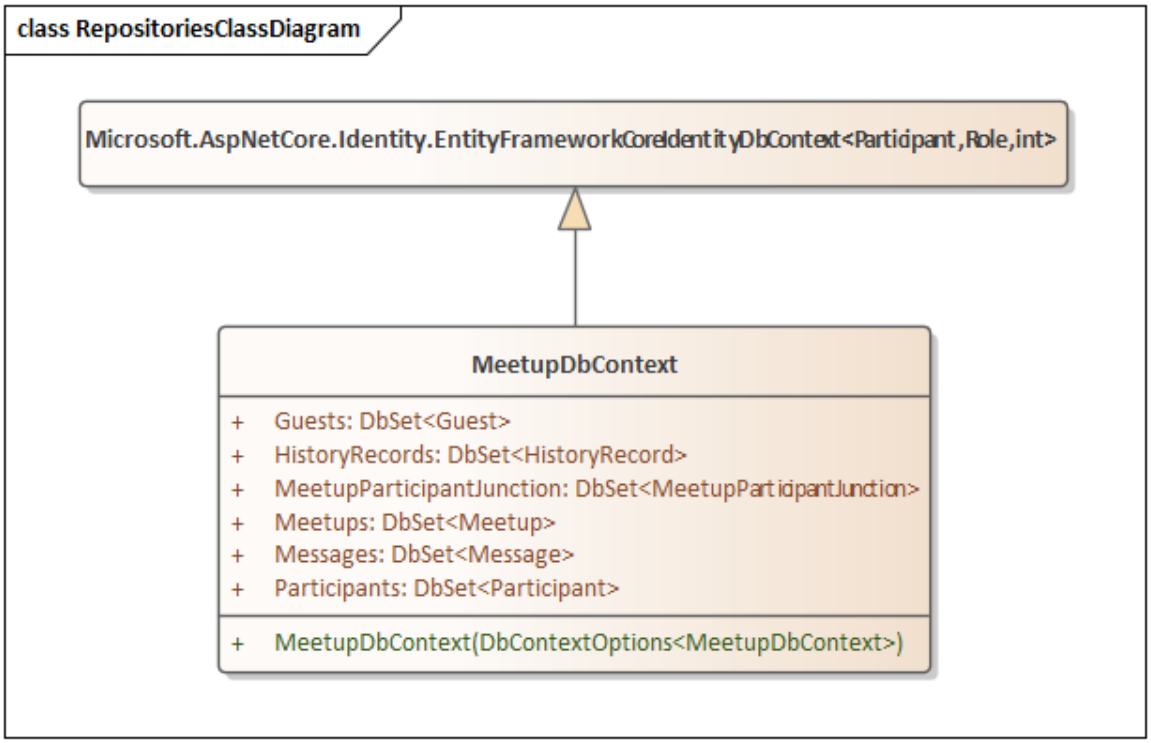
Třída *Participant* reprezentuje zaregistrovaného uživatele. Hlavním důvodem jejího vzniku je přidání potřebných atributů. S touto třídou je také spojena třída *Role*, která je v našem případě využita k omezení přístupu ke kontrolerům. Třída *Meetup* představuje jeden sraz. Vážou se k ní kolekce zpráv, hostů a záznamů do historie. Patří sem i kolekce třídy *Participant*, která je ale propojena přes třídu *MeetupParticipantJunction*. Tato třída

obsahuje dodatečné informace ohledné stavu účasti a zda si účastník přeje dostávat upozornění na tento konkrétní sraz.



Obrázek 58: Diagram tříd – výsledky operací

V rámci projektu jsou vytvořeny pomocné třídy, které obsahují informace o výsledcích různých operací. Každá operace může skončit buď úspěšně, nebo neúspěšně.
V neúspěšném případě je dodána také chybová zpráva, která je následně prezentována uživateli. Od základní třídy dědí další, specializované třídy, které mají dodatečně atributy.

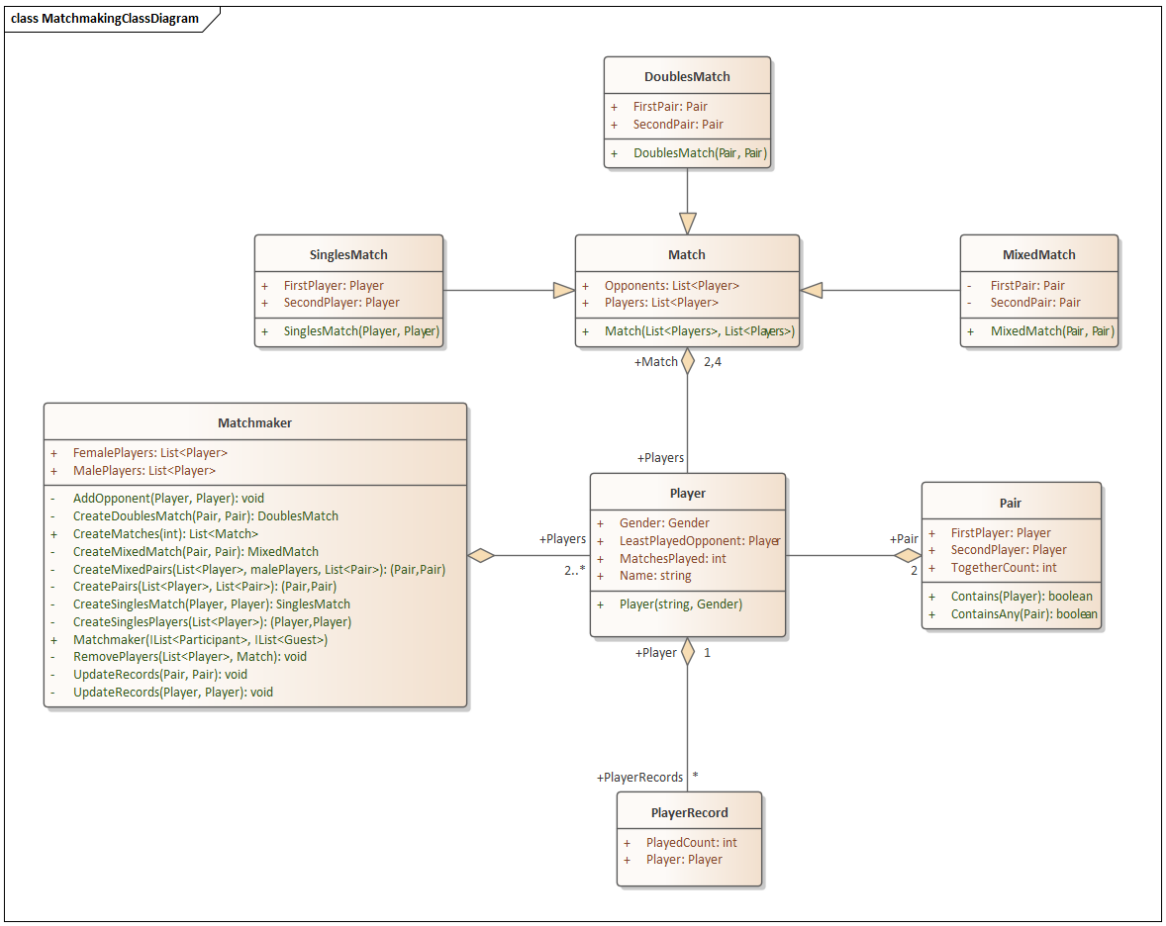


Obrázek 59: Diagram tříd – DbContext

Tato třída reprezentuje celou databázi aplikace. Uživatel s touto třídou interaguje prostřednictvím tříd *service*.

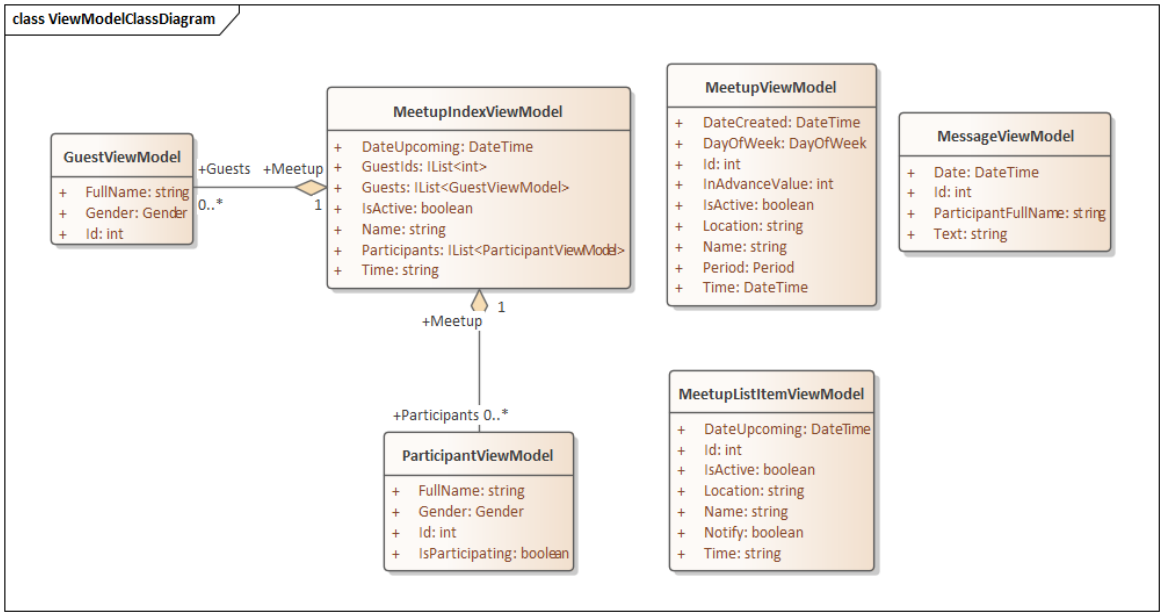
Obrázek 60: Diagram tříd – services

Každý *service* se stará o čtení, vytváření, úpravy a mazání dat z databáze. Třídy *service* jsou dále děleny podle toho, jaká je jejich oblast působení. Kupříkladu *AccountService* se zabývá výhradně daty, které se týkají účtu uživatele.



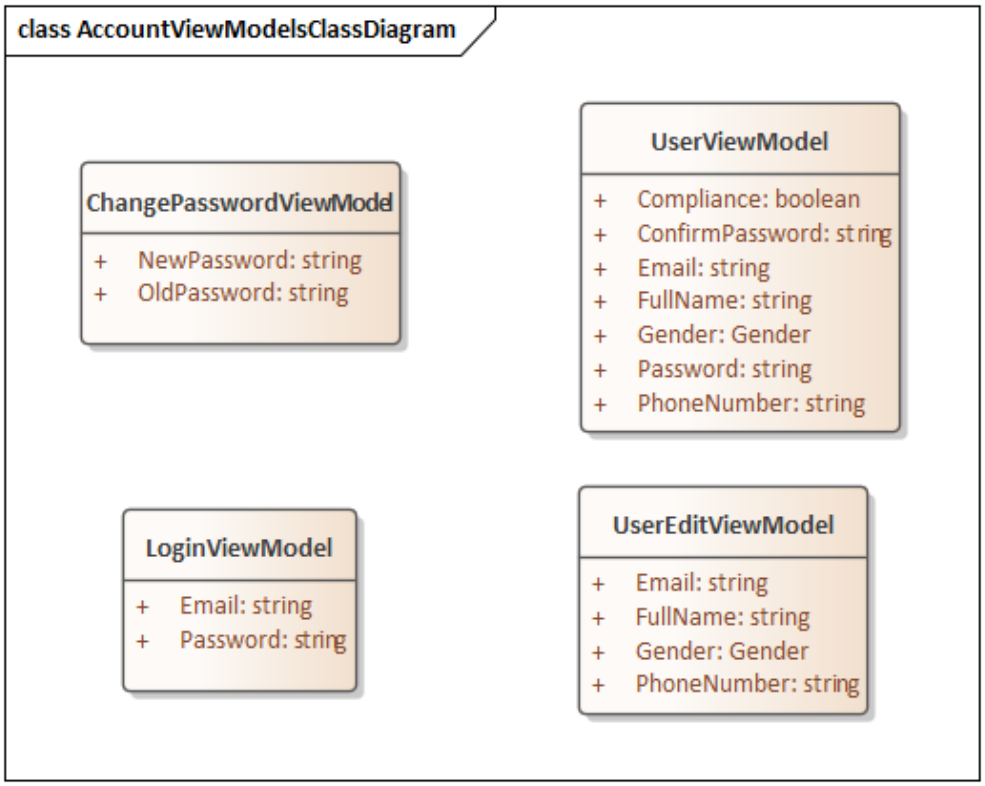
Obrázek 61: Diagram tříd – generování zápasů

V projektu byl vytvořen soubor tříd, který mi umožní snadněji namodelovat generování zápasů.

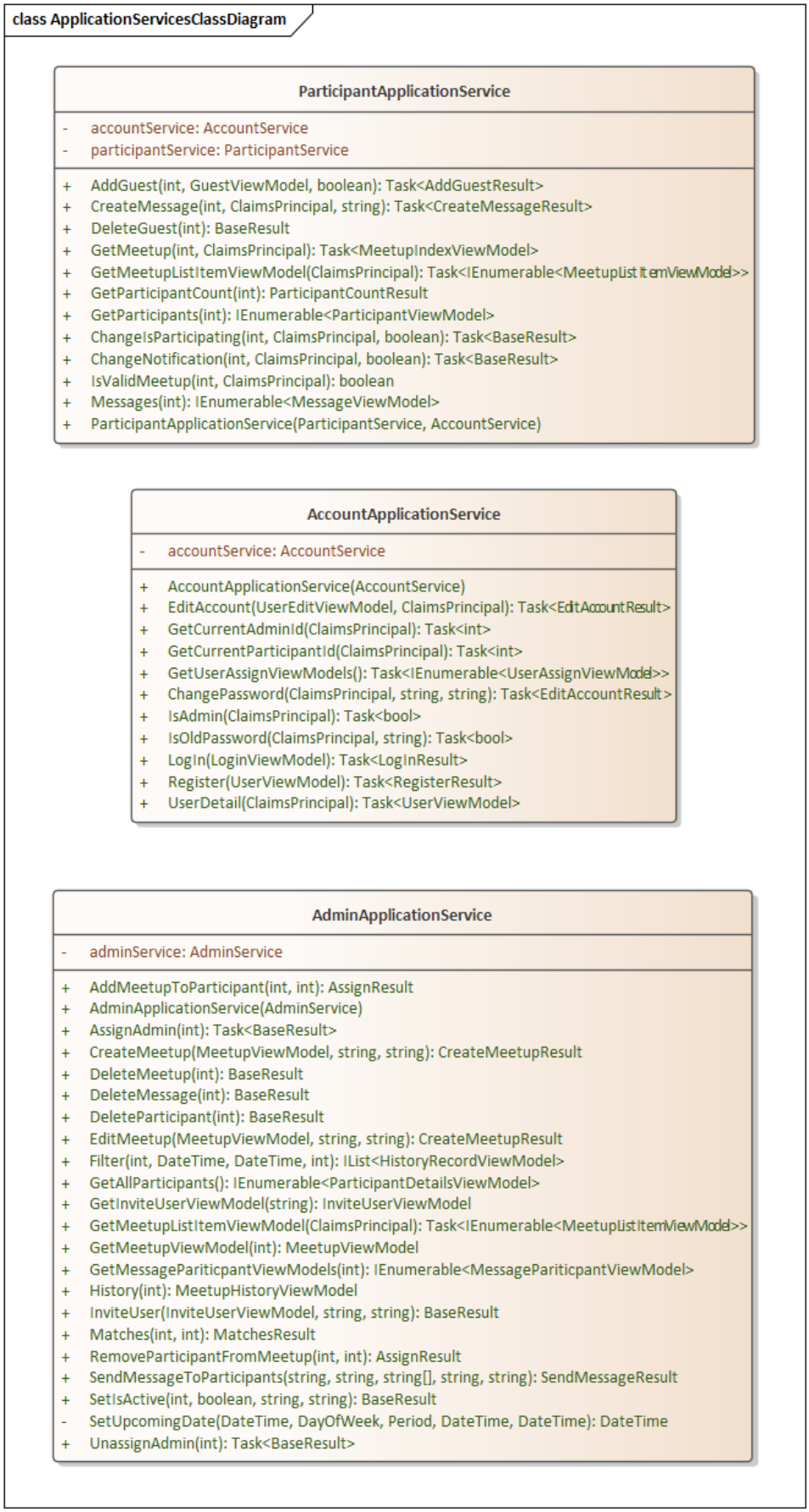


Obrázek 62: Diagram tříd – sdílené view modely

Ve vrstvě *Application* jsou k dispozici třídy, které slouží k prezentování dat uživateli. Často view modely obsahují jméno určité entity. Neobsahují ale všechny její atributy. Obsahují pouze ty atributy, které jsou důležité pro zobrazení. Dále také mohou view modely být použity pro poskytnutí dat uživatelem. View modely jsou poté buď konvertovány na entity, nebo jsou entity vyhledány pomocí dodaných dat.

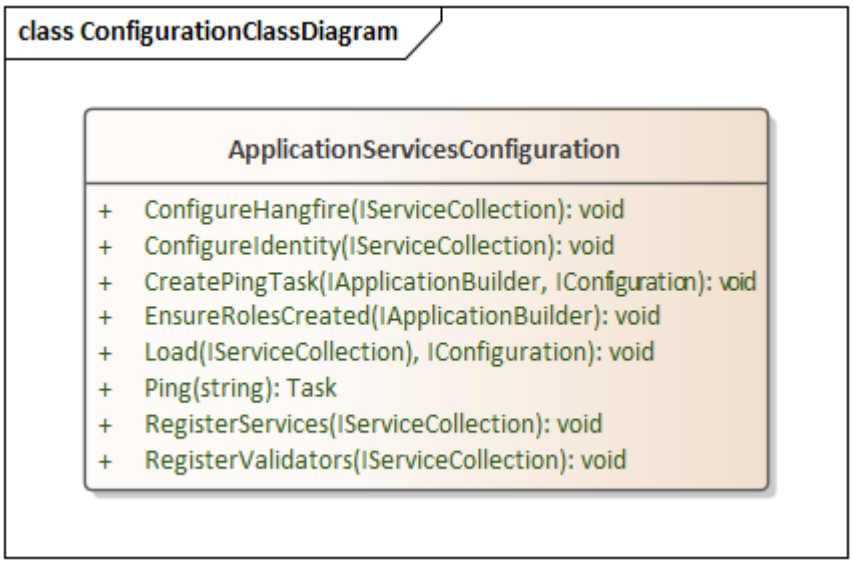


Obrázek 63: Diagram tříd – view modely týkající se účtu uživatele



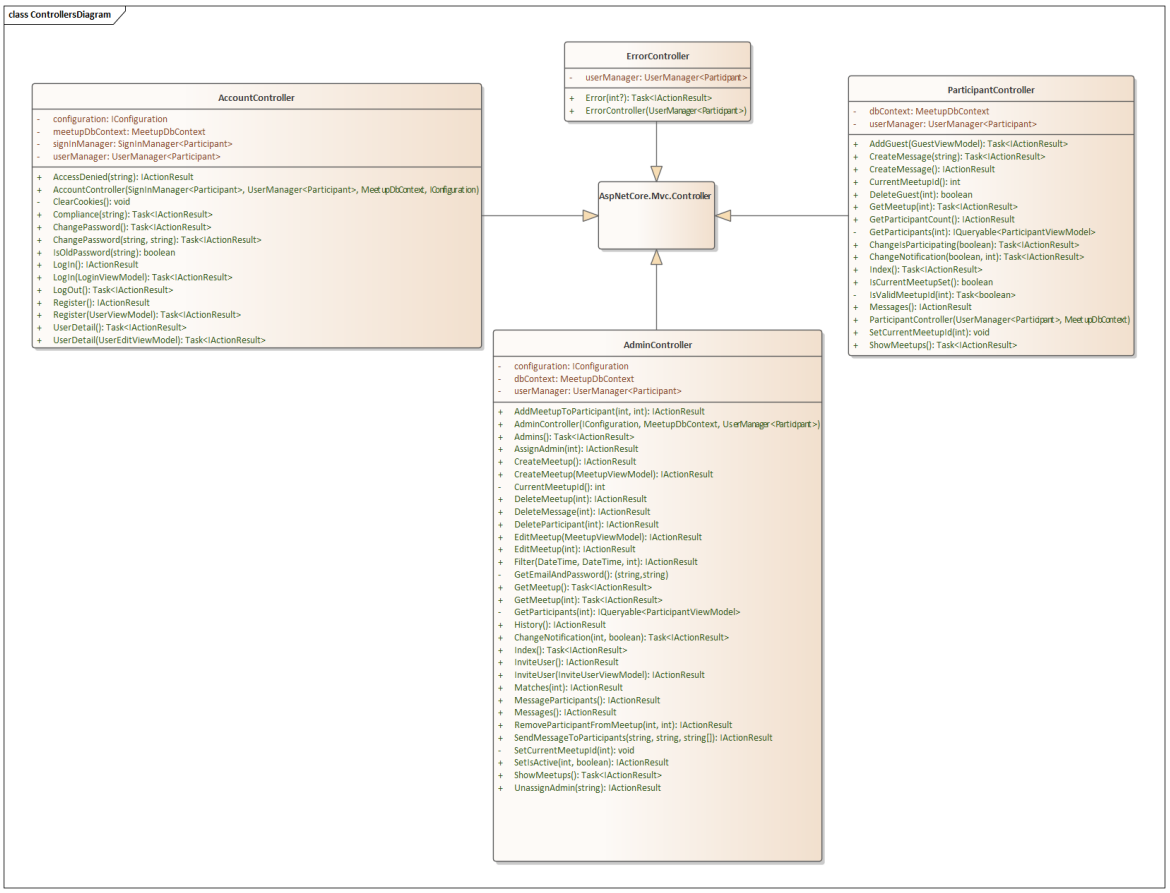
Obrázek 64: Diagram tříd – application services

Application service je mezistupeň mezi *service* a kontrolerem. Stará se především o zpracování dat získaných od uživatele, většinou z view modelů, a jejich převedení do formy, se kterou dokáže pracovat *service*.



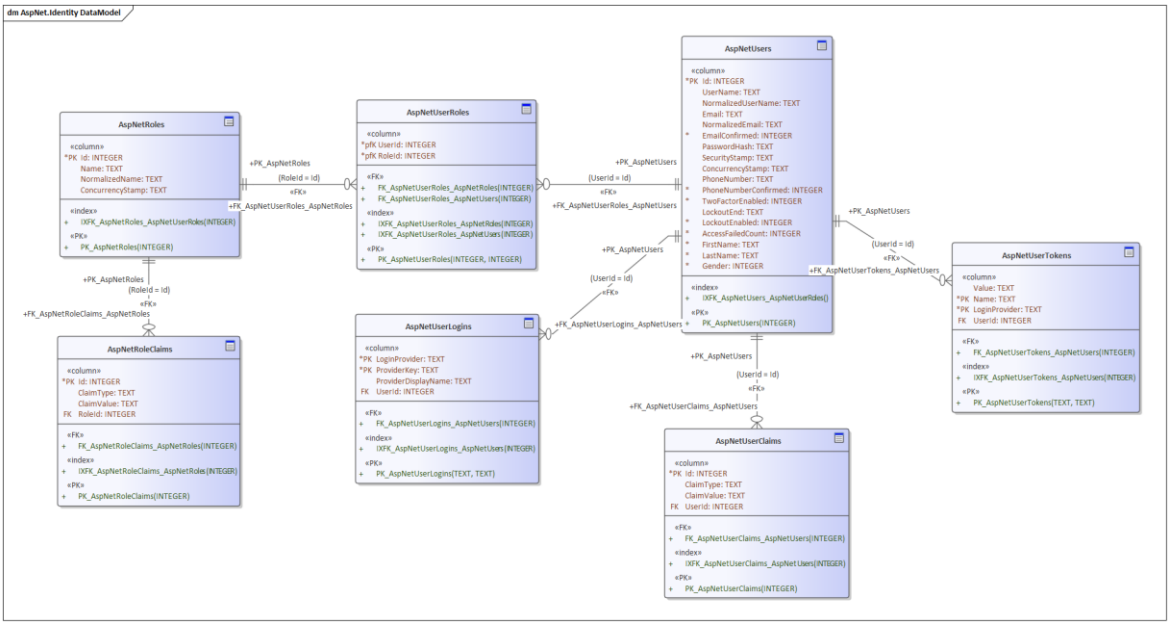
Obrázek 65: Diagram tříd – konfigurace

Nastavení samotné webové aplikace je soustředěno do jednoho místa, třídy *ApplicationServicesConfiguration*. Probíhá zde definování způsobu ukládání dat, nastavení požadavků na jméno a heslo uživatelů, definice rolí a také registrace frameworku sloužícího pro obstarání periodicky se opakujících událostí. Operace z této třídy jsou pak pouze volány ze třídy *Startup*.



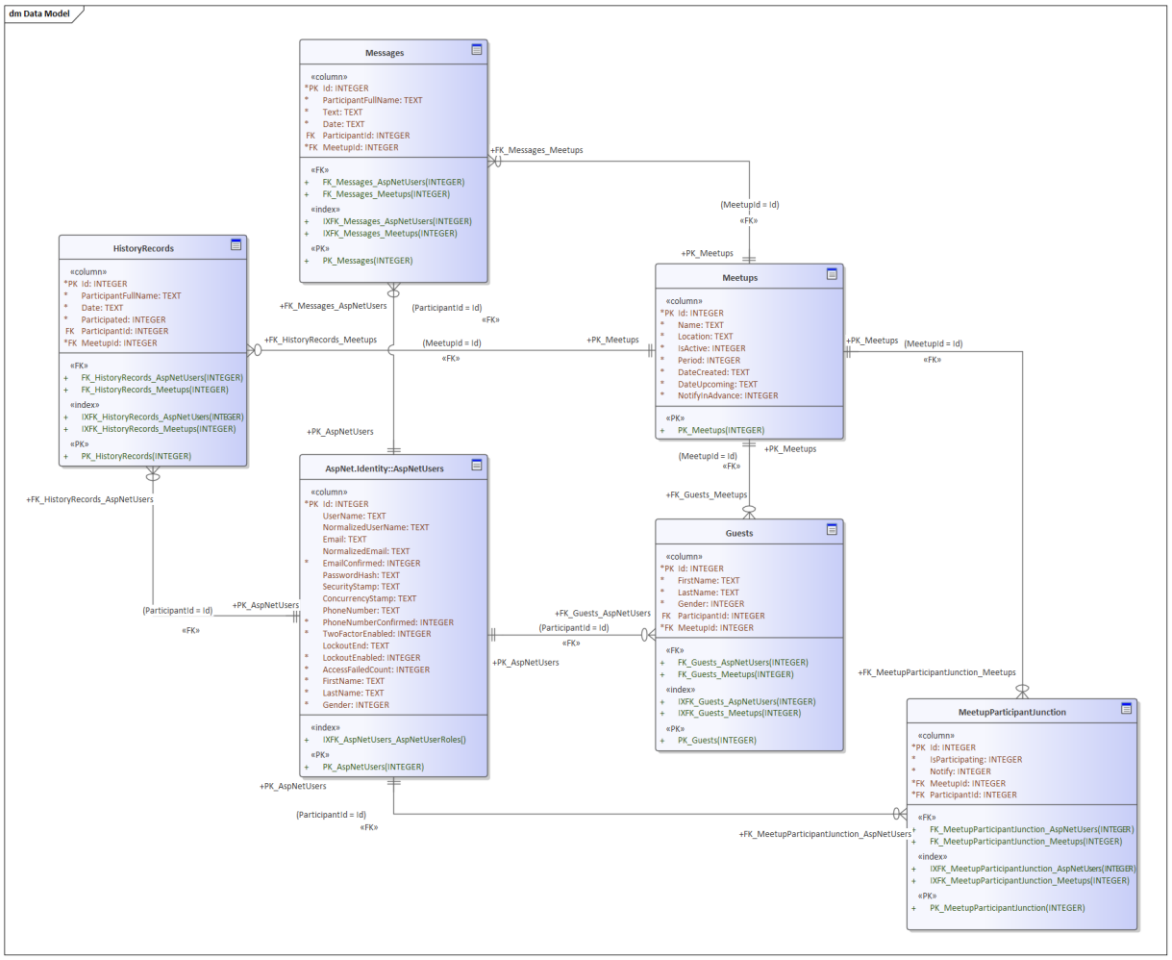
Obrázek 66: Diagram tříd – kontrolery

4.4 Datový model



Obrázek 67: Tabulky AspNet.Identity

Výše uvedené tabulky byly automaticky vygenerovány pomocí Entity Frameworku. Data v nich uložená se týkají především uživatelského účtu.



Obrázek 68: Tabulky entit

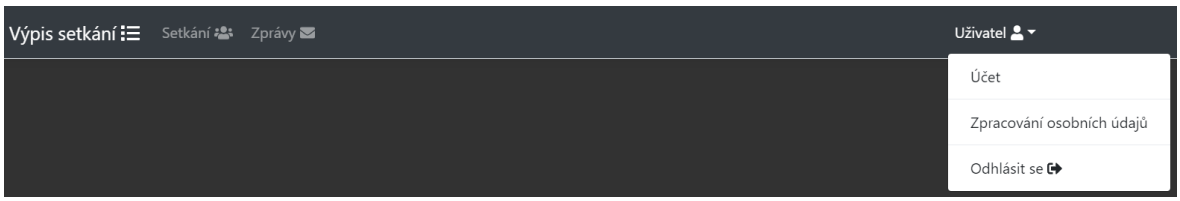
Tabulky a vztahy mezi nimi jsou z velké části odvozeny z definice tříd pro entity. Sloupce odpovídají názvům a datovým typům. Jednotlivé property lze navíc dekorovat atributy, které mohou upravit například název vygenerovaných sloupců.

4.5 Design webu

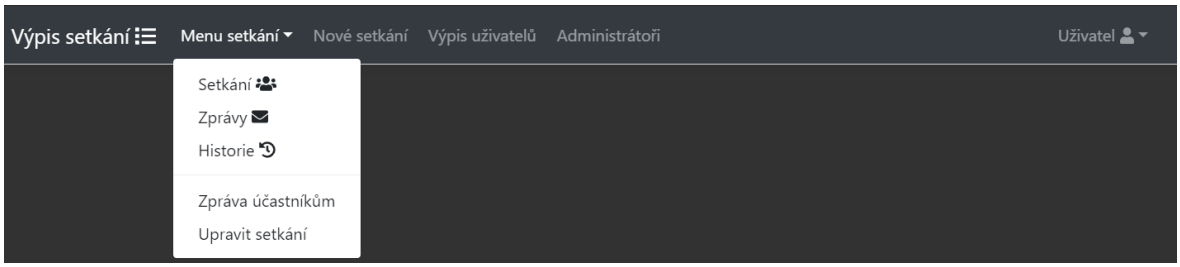
Rozvržení všech stránek kromě přihlášení a registrace je obdobné. Menu nahoře a hlavní obsah uprostřed. Byl zvolen tmavý motiv pro zmenšení námahy očí uživatelů. Dále jsou v aplikaci použity ikony.

4.5.1 Layout

Layout je soubor v projektu, který obsahuje základní soubory *.css* a *.js*. Dále může obsahovat elementy menu. Pomocí příkazu *RenderBody()* můžeme v hlavní části těla dokumentu zobrazit další libovolný HTML kód. Podmínka je, že daná stránka musí mít správně nastavenou proměnnou *Layout*.
V aplikaci jsou definovány dva layouty. Jeden pro účastníka a jeden pro administrátora. Liší se počet položek v menu.

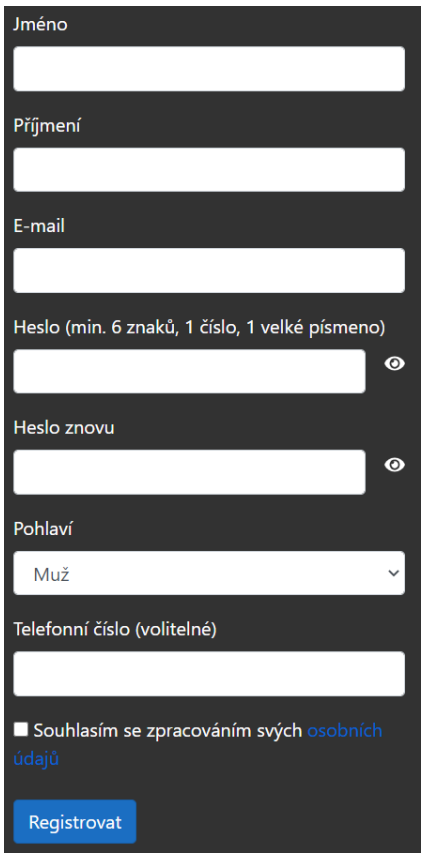


Obrázek 69: Menu účastníka



Obrázek 70: Menu administrátora

4.5.2 Stránka registrace



Obrázek 71: Formulář registrace uživatele

```
@model Badub.Application.Models.Account.UserViewModel
@using Badub.Application.Models
@using Badub.Application.Resources

@{
    Layout = null;
}
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <meta charset="utf-8" />
    <title>
        @(ViewBag.Title = ViewBag.RegisterTitle)
    </title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css"
/>
    <link rel="stylesheet" href="~/lib/fontawesome/all.css" />
    <link rel="stylesheet" href="~/css/site.css" />
    <link rel="stylesheet" href="~/css/darkStyle.css" />
</head>
```

Obrázek 72: Hlavička stránky registrace

```
<div class="form-group">
  <label asp-for="FirstName" class="control-label">Jméno</label>
  <input asp-for="FirstName" class="form-control" />
  <span asp-validation-for="FirstName" class="text-danger"></span>
</div>
```

Obrázek 73: Registrace – pole jména

Většina položek formulářů má podobnou strukturu. Jedna položka má svůj název, pole pro vstup od uživatele a místo, ve kterém se zobrazí případná chybová hláška pod vstupem pro uživatele.

```
<div class="form-group">
  <label asp-for="Gender" class="control-label">Pohlaví</label>
@Html.DropDownList("Gender", new List<SelectListItem>
{
  new SelectListItem{ Value = Gender.Male.ToString(), Text =
MeetupResources.MaleName, Selected = true},
  new SelectListItem{ Value = Gender.Female.ToString(), Text =
MeetupResources.FemaleName}
}, new { @class = "form-control", name = "Gender" })
</div>
```

Obrázek 74: Dropdown menu pro výběr pohlaví

Některé prvky lze vygenerovat použitím třídy *Html*. Zde je příklad vytvoření dropdown menu pro výběr pohlaví.

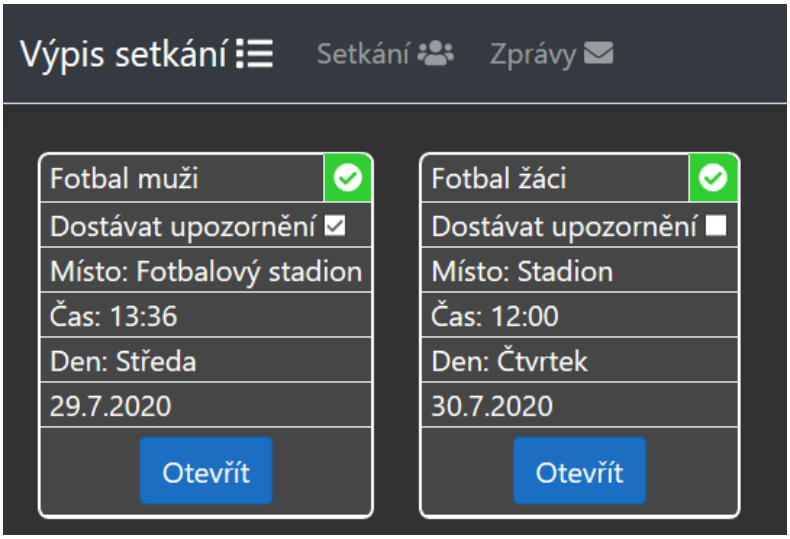
```
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/lib/fontawesome/all.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
    <script>
        $(document).ready(() => {
            $('.registerButton').on('click', register)
            $('.showPassword').on('click', togglePasswordVisibility);
            $(document).keypress(function (e) {
                var keyCode = e.keyCode ? e.keyCode : e.which;
                if (keyCode == '13')
                    register();
            });
            $('#complianceLink').click(showCompliance);
        });

        function register() {

            var button = $('.registerButton');
            var registerMessage = $('#registerErrorMessage');
            registerMessage.text('');
            var form = $('form');
            var formData = form.serialize();

            if (form.valid() == false) return;
var checked = $('#Compliance').is(':checked');
            if (checked == false) {
                $('.complianceMessage').text('Je třeba souhlasit se
zpracováním údajů');
```

```
                return;
            }

            button.prop('disabled', true);

            $.ajax({
                method: 'POST',
                url: '@Url.Action("Register")',
                data: formData,
                success: function (data) {
                    if (data.succeeded) {
                        window.location = data.url;
                    }
                    else {
                        if (data.passwordError) {
                            $('.passwordError').text(data.errors[0]);
                        }
                        else if (data.registrationFailedMessage) {
                            var message = data.registrationFailedMessage;
                            registerMessage.append(message);
                            registerMessage.append('<br>');
                        }
                        else {
                            data.errors.forEach((item, index) => {
                                registerMessage.append(item);
                                registerMessage.append('<br>');
                            });
                        }

                    }
                }
            }).done(() => {
                button.prop('disabled', false);
            });
        }
```

Obrázek 75: Skripty na stránce registrace

Jelikož na stránce registrace není nastaven *Layout*, je třeba nalinkovat soubory .css a .js manuálně. Samotná registrace je obstarána pomocí asynchronního požadavku do kontroleru. Ještě před odesláním formuláře dojde k validaci dat a případné zobrazení chybových hlášek. Po úspěšné registrace dojde k přihlášení uživatele a přesměrování na výpis přiřazených srazů.

4.5.3 Stránka výpisu setkání



Obrázek 76: Výpis přiřazených srazů

Přihlášený uživatel vidí výpis srazů, které mu byly přiřazeny. Vpravo nahoře vidí aktuální stav srazu. Může být aktivní nebo neaktivní. Dále může zapnout nebo vypnout upozorňování na nadcházející sraz.

```
@model IEnumerable<Badub.Application.Models.MeetupListItemViewModel>
@using System
@using Badub.Application.Resources

@{
    ViewBag.Title = "Seznam setkání";
    Layout = "~/Views/Participant/ParticipantLayout.cshtml";
}
```

Obrázek 77: Stránka výpisu srazů 1.

U této konkrétní stránky je použit *ParticipantLayout*. Je určen pro účastníky bez administrátorské role. Účastník musí nejdříve otevřít jeden ze srazů, aby se mu nastavil jako aktuálně zvolený. Teprve poté bude mít přístup na odkazy „Setkání“ a „Zprávy“ v menu.

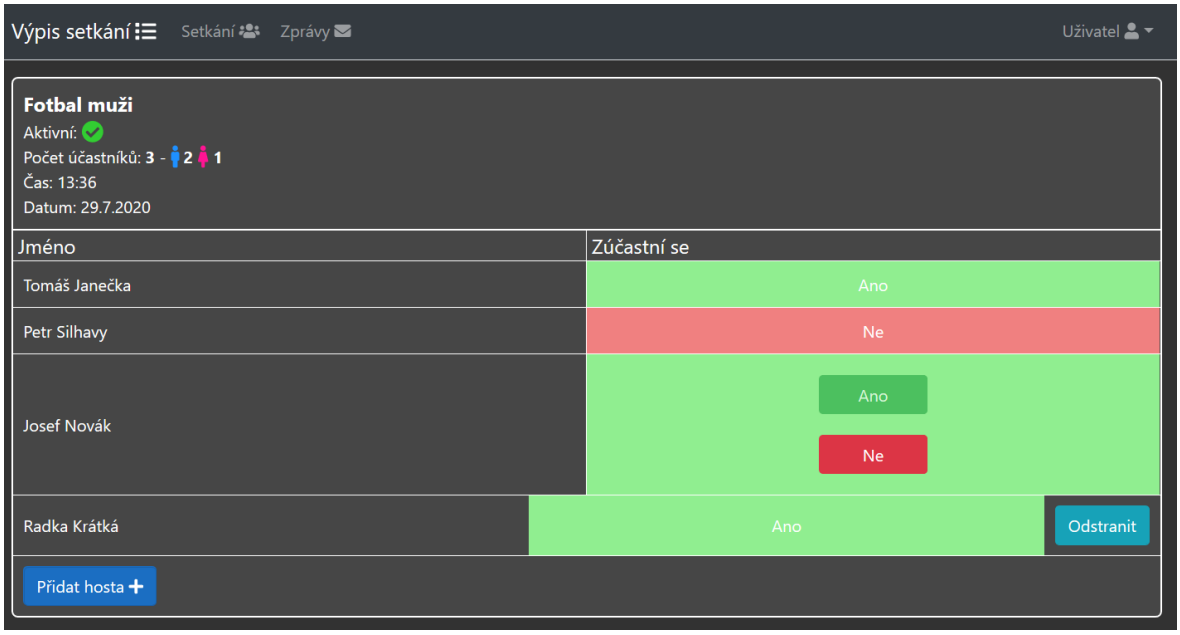
```
<div class="meetupTable">
    @if (Model.Count() == 0)
    {
        <h4>Nemáte žádné přiřazené srazy</h4>
    }
    @foreach (var meetup in Model)
    {
        <div class="meetup">
            <span class="header">
                <span class="meetupName">
                    @meetup.Name
                </span>
                <span class="activityItem">
                    @if (meetup.IsActive)
                    {
                        <span class="notActive" style="display: none;">
                            <i class="fa fa-times-circle"></i>
                        </span>
                        <span class="active">
                            <i class="fa fa-check-circle"></i>
                        </span>
                    }
                    else
                    {
                        <span class="notActive">
                            <i class="fa fa-times-circle"></i>
                        </span>
                        <span class="active" style="display: none;">
                            <i class="fa fa-check-circle"></i>
                        </span>
                    }
                </span>
            </span>
            <span class="meetupItem">
                <span>Dostávat upozornění</span>
                <span>
                    @Html.AntiForgeryToken()
                    <input type="checkbox" id="notifyCheckbox"
checked="@meetup.Notify" />
```

```
                </span>
            </span>
            <span class="meetupItem">
                <span>Místo: </span>
                <span>@meetup.Location</span>
            </span>
            <span class="meetupItem">
                <span>Čas: </span>
                <span>@meetup.Time</span>
            </span>
<span class="meetupItem">
                <span>Den: </span>
                <span>
                    @{
                        string dayOfWeekName = "";
                        switch (meetup.DateUpcoming.DayOfWeek)
                        {
                            case DayOfWeek.Monday:
                                dayOfWeekName = DateTimeResources.MondayName;
                                break;
                            case DayOfWeek.Tuesday:
                                dayOfWeekName =
DateTimeResources.TuesdayName;
                                break;
                            case DayOfWeek.Wednesday:
                                dayOfWeekName =
DateTimeResources.WednesdayName;
                                break;
                            case DayOfWeek.Thursday:
                                dayOfWeekName =
DateTimeResources.ThursdayName;
                                break;
                            case DayOfWeek.Friday:
                                dayOfWeekName = DateTimeResources.FridayName;
                                break;
                            case DayOfWeek.Saturday:
                                dayOfWeekName =
DateTimeResources.SaturdayName;
                                break;
```

```
                                case DayOfWeek.Sunday:
                                    dayOfWeekName = DateTimeResources.SundayName;
                                    break;


                            }
                        }
                        @dayOfWeekName
                    </span>
                </span>
                <span class="meetupItem">

<span>@meetup.DateUpcoming.ToString(DateTimeResources.DateFormat)</span>
                </span>
                <div class="meetupButtons">
                    <form method="post">
                        <button class="btn btn-primary getMeetupBtn"
type="button" value="@meetup.Id" name="id">Otevřít</button>
                    </form>
                </div>
            </div>
        }
</div>
```

Obrázek 78: Stránka výpisu srazů 2.

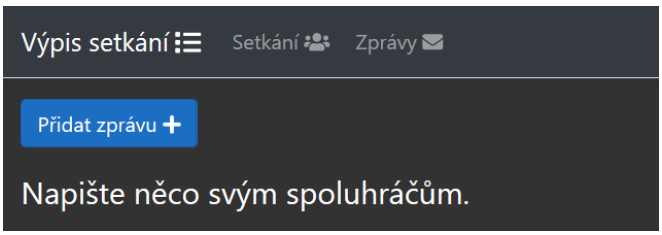
4.5.4 Stránka srazu



Obrázek 79: Stránka srazu 1.

Po otevření srazu uživatel vidí informace o srazu, přihlášených účastnících a nahlášených hostech. Účastník může změnit pouze svůj stav účasti. V tomto konkrétním srazu se nachází host, který byl nahlášen právě přihlášeným uživatelem. Je u něj proto možnost ho odstranit.

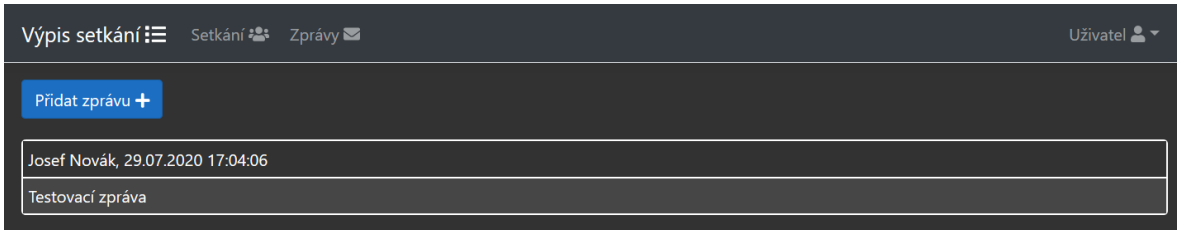
Formulář pro přidání hosta byl separován do *partial view*. Jedná se pouze o další soubor s HTML kódem. Poté ho lze umístit do jiného *view* pomocí *Html.PartialAsync()*. Tento formulář je stejný jak pro účastníka, tak administrátora. Měnit stav účasti a přidávat hosty lze pouze, pokud je stav srazu nastaven na „Aktivní“.



Obrázek 80: Zprávy srazu

Každý účastník srazu nebo administrátor může napsat zprávu ostatním. Každý sraz má svoji vlastní kolekci zpráv, které se nesdílejí napříč srazy.

Obrázek 81: Zprávy srazu – nová zpráva



Obrázek 82: Zprávy srazu – testovací zpráva

```
@model IEnumerable<Badub.Application.Models.MessageViewModel>
@using Badub.Application.Resources
@using Badub.Application.Models

@{
    ViewBag.Title = "Zprávy";
    Layout = "~/Views/Participant/ParticipantLayout.cshtml";
```

```
}

@await Html.PartialAsync("CreateMessage", new MessageViewModel())
<div class="messageTable">
    @if (Model.Count() == 0)
    {
        <h4 class="noMessagesHeader">@MeetupResources.NoMessagesMessage</h4>
    }
    @foreach (var message in Model)
    {
        <div class="message">
            <div class="messageHeaderSection">
                <span class="messageHeader">@message.ParticipantFullName,
@message.Date.ToString(DateTimeResources.FullDateTimeFormat)</span>
            </div>
            <div class="messageBody">@message.Text</div>
        </div>
    }
</div>
```

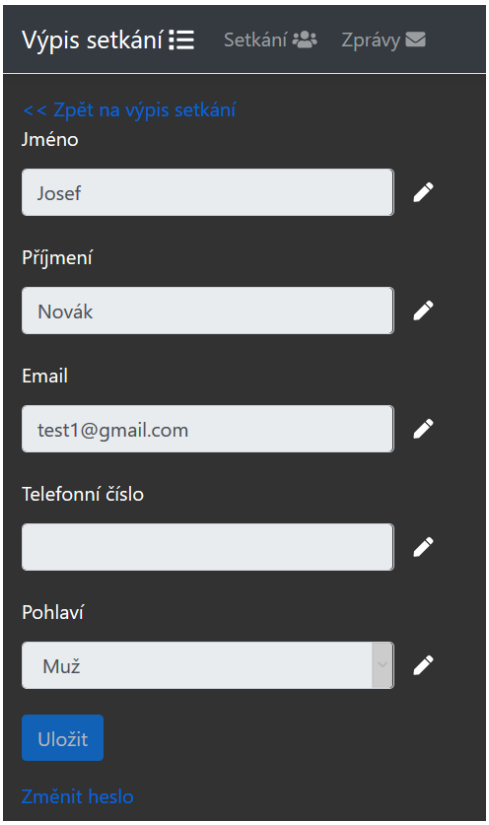
Obrázek 83: Zprávy srazu – kód

U zpráv je opět využito *partial view* pro vytvoření nové zprávy. Toto je především z důvodu validace nové zprávy.

```
@model Badub.Application.Models.MessageViewModel
<div class="addMessageSection">
    <button class="btn btn-primary showAddMessageFormBtn">Přidat zprávu <i
class="fa fa-plus"></i></button>
    <button class="btn btn-danger closeAddMessageFormBtn">Zavřít <i class="fa
fa-times"></i></button>
    <form class="addMessageForm" method="post">
        <textarea class="form-control messageTextArea" placeholder="Napište
zprávu..." asp-for="Text"></textarea>
        <span asp-validation-for="Text" class="text-danger"></span>
        <div class="input-group-append">
            <input type="button" class="btn btn-outline-primary
addMessageBtn" value="Přidat" />
        </div>
    </form>
</div>
```

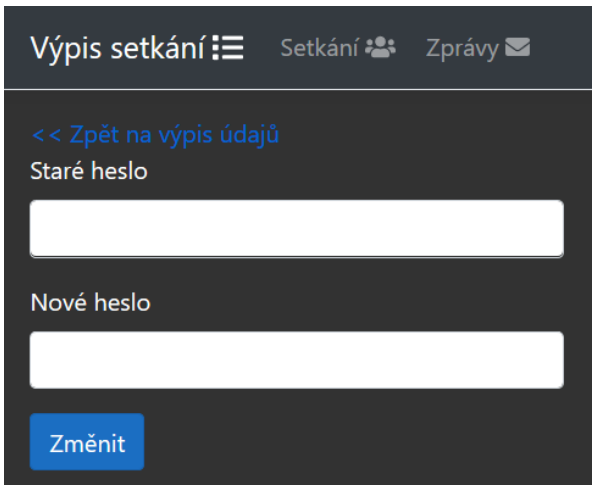
Obrázek 84: Nová zpráva

4.5.5 Zobrazení a úprava údajů účtu



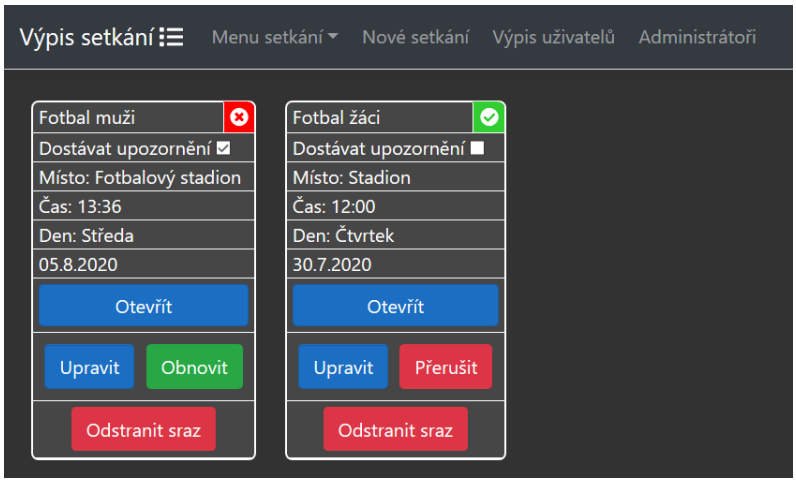
Obrázek 85: Přehled údajů účtu

Každý uživatel může kdykoliv změnit své údaje. Toto je provedeno na stránce jejich přehledu. U každého pole, které je na začátku deaktivováno, může uživatel kliknout na ikonu, které aktivuje pole a dovolí přepsat jeho hodnotu. Dochází zde ke kontrole toho, zda nově zadaný e-mail už není evidován. Kód formuláře je obdobný jako u registrace.
Ke změně hesla dochází na další stránce.



Obrázek 86: Změna hesla

4.5.6 Stránka výpisu setkání pro administrátora

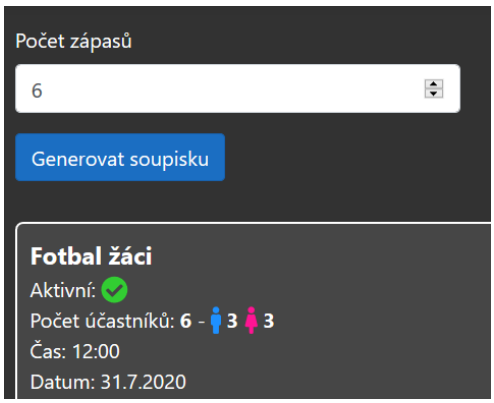


Obrázek 87: Výpis srazů – administrátor

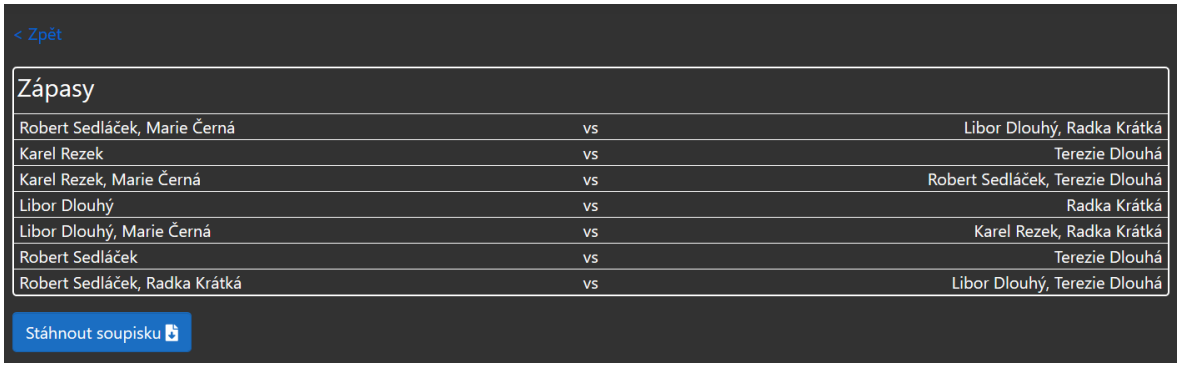
Administrátor má k dispozici podstatně více funkcí. Přímo z výpisu srazů může přerušit či obnovit jednotlivé srazy, dostat se na stránku pro úpravu informací srazu nebo sraz odstranit.

4.5.7 Stránka srazu pro administrátora

Administrátor si může prohlédnout veškeré existující srazy. Pouze u těch, které mu jsou přiřazeny může upravovat stav zasílání upozornění a měnit svůj stav účasti. Na stránce jednoho srazu administrátorovi přibývá možnost vygenerovat soupisku zápasů. Jsou do ní zařazeni pouze přihlášení účastníci a hosté.

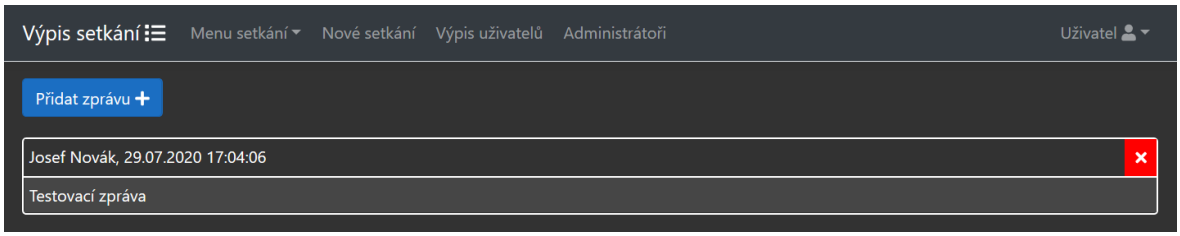


Obrázek 88: Formulář pro generování soupisky zápasů



Obrázek 89: Vygenerovaná soupiska

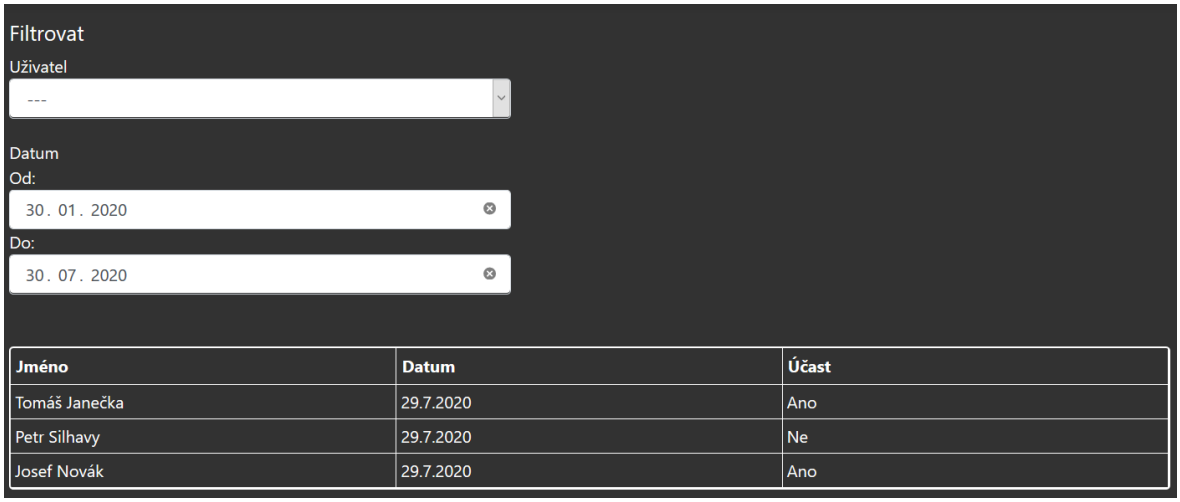
Stránka zpráv ve srazu je opět shodná se stránkou pro účastníka. Administrátor má navíc dostupnou funkci odstranění zpráv.



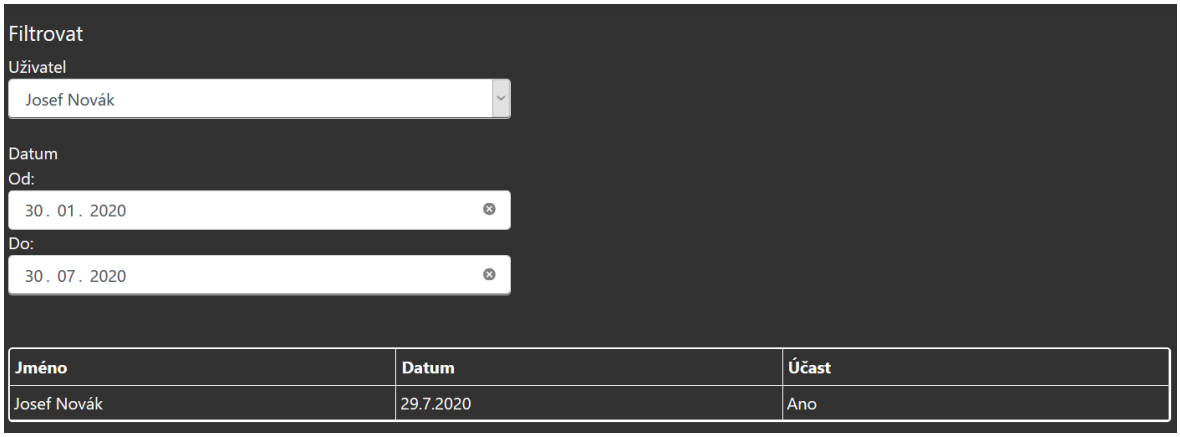
Obrázek 90: Zprávy srazu – administrátor

4.5.8 Stránka historie účastí

Každý sraz má svoji historii účastí. Administrátor díky ní může dohledat počty účastí v daném období nebo aktivitu jednotlivých účastníků. Je zde k dispozici funkce filtrace. Administrátor si může vybrat z filtrace podle období, účastníka nebo jejich kombinace.



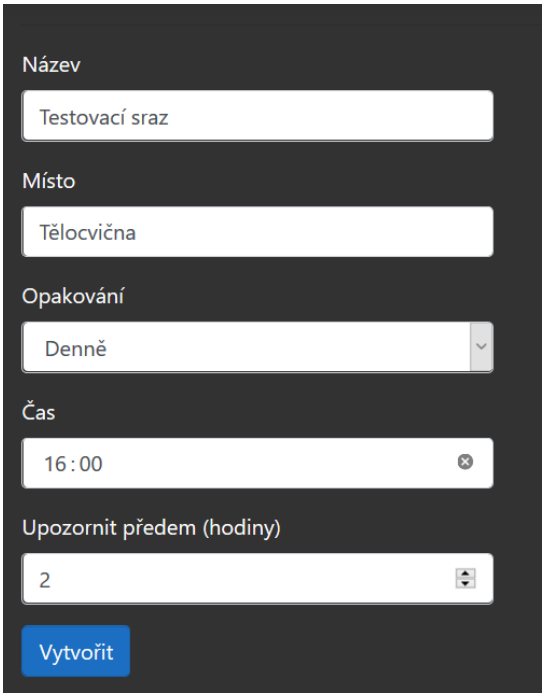
Obrázek 91: Historie účastí



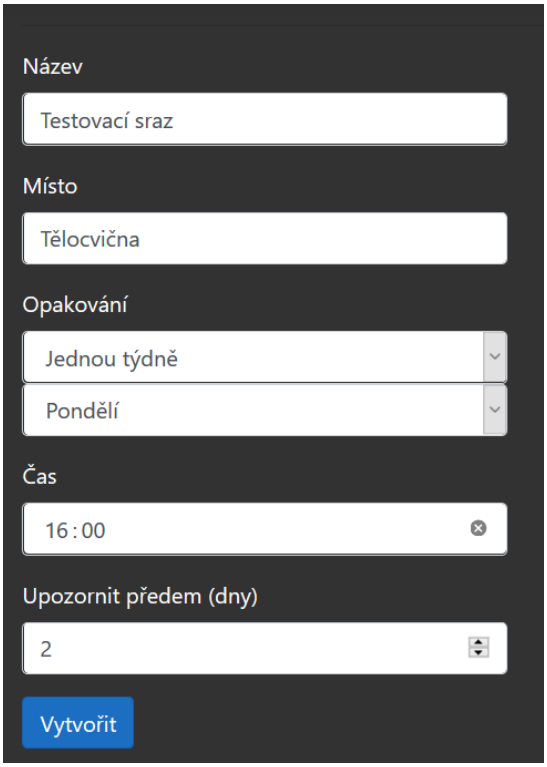
Obrázek 92: Historie účastí – filtrace podle účastníka

4.5.9 Stránka vytvoření srazu

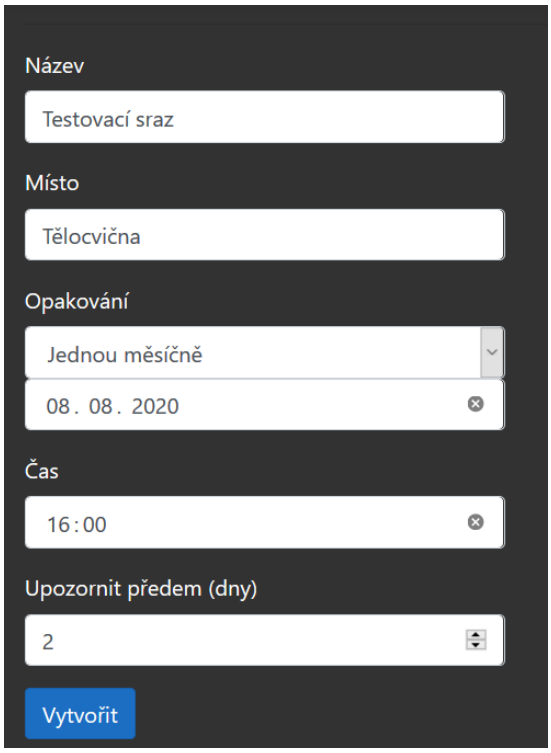
Dochází zde k vytvoření nového srazu, ke kterému mohou být později přiřazeni účastníci. Administrátor má na výběr ze tří možností opakování srazu – jednou denně, jednou týdně nebo jednou měsíčně.



Obrázek 93: Nový sraz – denně



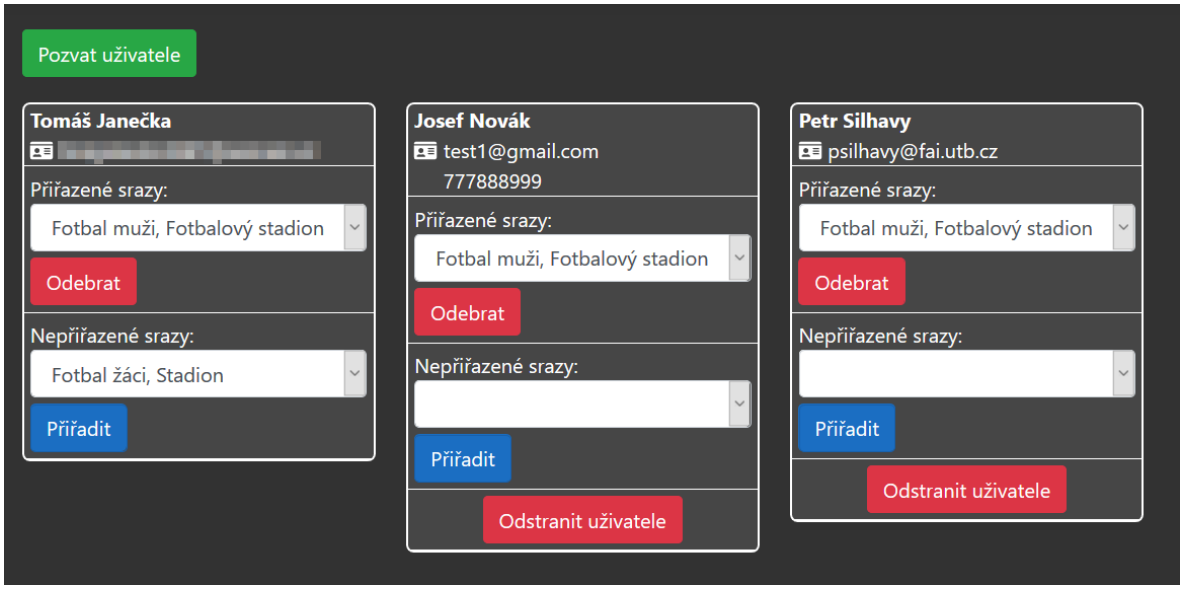
Obrázek 94: Nový sraz – týdně



Obrázek 95: Nový sraz – měsíčně

4.5.10 Stránka výpisu uživatelů

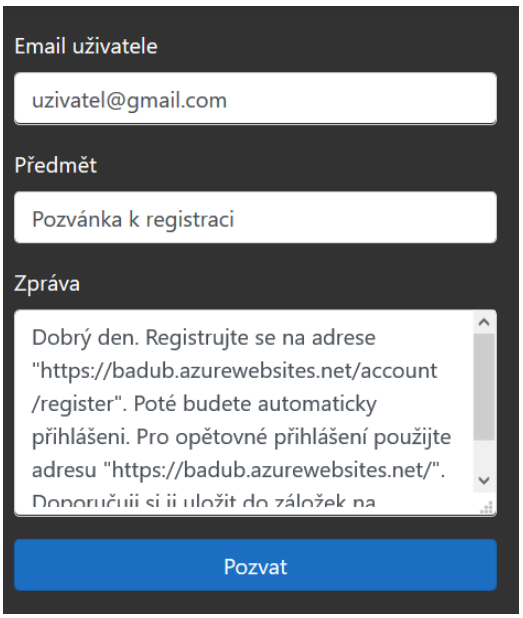
Administrátor má k dispozici přehled všech uživatelů, kteří se zaregistrovali. Má kontrolu nad tím, jaké srazy mají přiřazeny. Může také uživatele odstranit. U každého záznamu je uvedeno i jméno a kontaktní údaje.



Obrázek 96: Výpis uživatelů

4.5.11 Stránka pozvání uživatele

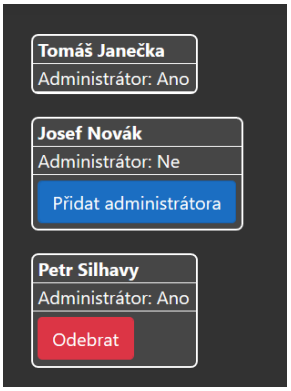
Administrátor může do aplikace zvát uživatele. Do formuláře musí zadat alespoň e-mail zvaného uživatele. Předmět a zpráva jsou automaticky vygenerovány s potřebnými informacemi. Administrátor tyto údaje může jakkoliv změnit. Po odeslání e-mailu se zobrazí hláška o tom, zda byl e-mail odeslán v pořádku či nikoliv.



Obrázek 97: Formulář pro pozvání uživatele

4.5.12 Stránka pro přidání administrátora

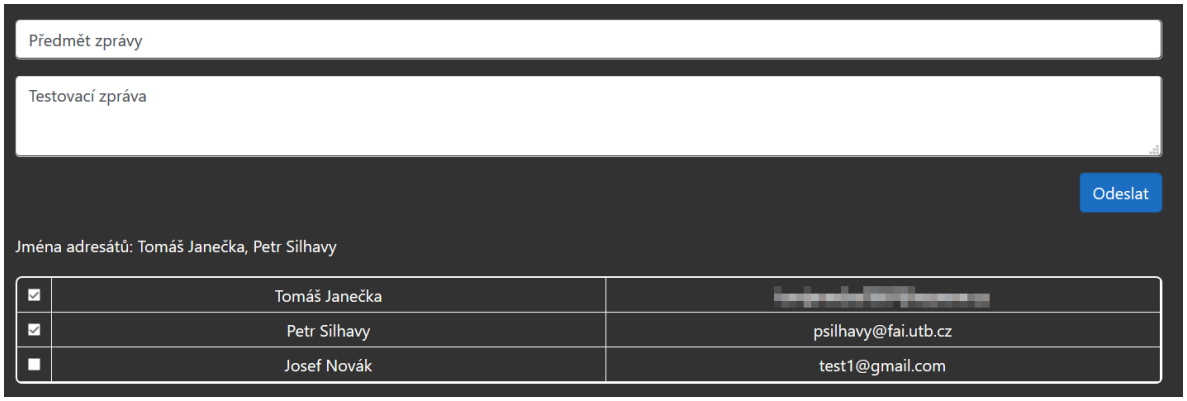
Administrátor může přidat nebo odebrat ostatním uživatelům roli administrátora.



Obrázek 98: Přidání role administrátora

4.5.13 Stránka pro zaslání hromadné zprávy

Administrátor může v rámci jednoho srazu zaslat hromadnou zprávu všem nebo jen některým účastníkům. E-mail bude zaslán těm účastníkům, u kterých administrátor v tabulce zaškrtnul checkbox.



Obrázek 99: Formulář a tabulka pro zaslání hromadné zprávy

4.5.14 Chybové stránky

Aplikace využívá možnosti vytvořit si vlastní stránky pro upozornění na chybový stav. Tato možnost je aktivována pomocí metody *UseStatusCodePagesWithReExecute()*. Je navíc možné prezentovat chybové stavy různě podle prostřední aplikace. Na jednotlivá prostředí se lze dotázat pomocí metod *IsDevelopment()*, *IsProduction()* a *IsStaging()*.

```
if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
```

```
            else
            {
                app.UseStatusCodePagesWithReExecute("/Error",
"?statusCode={0}");
            }
```

Obrázek 100: Výběr prezentace chyb podle prostředí

```
@{
    ViewBag.Title = "Stránka nenalezena";
    Layout = ViewBag.Layout;
}

<h4 class="text-danger mb-5">Stránka nebyla nalezena <i class="fa fa-
search"></i></h4>

<a class="btn btn-info" asp-action="ShowMeetups" asp-
controller="@ViewBag.Controller" asp-area="">Zpět na výpis srazů</a>
```

Obrázek 101: Stránka pro prezentaci chyby nenalezení stránky

```
[Route("/Error")]
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public async Task<IActionResult> Error(int? statusCode)
        {
            bool isAdmin = await
accountApplicationService.IsAdmin(HttpContext.User);

            if (isAdmin)
            {
                ViewBag.Controller = "Admin";
                ViewBag.Layout = "~/Views/Admin/AdminMeetupLayout.cshtml";
            }
            else
            {
                ViewBag.Controller = "Participant";
                ViewBag.Layout =
"~/Views/Participant/ParticipantLayout.cshtml";
            }

            if (statusCode.HasValue)
```

```
            {
                if (statusCode.Value == 404)
                    return View("NotFound");
                else if (statusCode.Value == 403)
                    return View("Forbidden");
                else if (statusCode.Value == 500)
                    return View("InternalError");
            }

            return View("NotFound");
        }
```

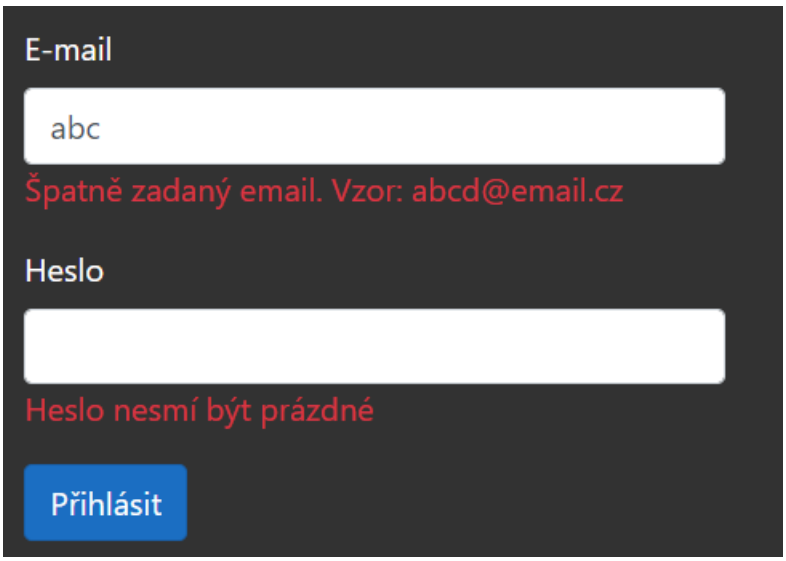
Obrázek 102: Metoda kontroleru – zpracování chybového kódu

4.6 Obstarání požadavků uživatelů

Uživatelé komunikují s aplikací prostřednictvím kontrolerů. Ty zpracovávají jejich požadavky a poskytují jim odpovídající webové stránky. Při spuštění jednotlivých operací jim také poskytují zpětnou vazbu ohledně jejich výsledků. Většina dotazů od uživatele je zasílána asynchronně a výsledek je poté zobrazen bez nutné aktualizace stránky.

Ještě před odeslání požadavků zpravidla dochází k validaci dat. V aplikaci existují validátory, které obsahují pravidla pro *view modely*. Může se jednat o nastavení minimální a maximální délky textu v textovém poli nebo o to, zda je zadána e-mailová adresa ve správném formátu. Na případnou chybu je uživatel upozorněn chybovou hláškou pod polem s chybnými daty.

4.6.1 Požadavek na přihlášení



Obrázek 103: Chybové hlášky při pokusu o přihlášení

```
public class LoginValidator : AbstractValidator<LoginViewModel>
    {
        public LoginValidator()
        {
            int minLength = ValidatorResources.EmailMinLength;
            int maxLength = ValidatorResources.EmailMaxLength;
            int passwordMaxLength = ValidatorResources.PasswordMaxLength;
            RuleFor(l => l.Email) NotEmpty().Length(minLength,
maxLength).WithMessage(ValidatorResources.MinMaxLengthMessage(minLength,
maxLength))
.EmailAddress().WithMessage(ValidatorResources.IncorrectEmailFormatMessage);
 RuleFor(l => l.Password)
.NotEmpty().WithMessage(ValidatorResources.PasswordEmptyMessage)
.MaximumLength(passwordMaxLength).WithMessage(ValidatorResources.TextTooLongM
essage(passwordMaxLength));
        }
    }
```

Obrázek 104: Validátor pro přihlášení

```
function login() {


            var button = $('.loginButton');
            var loginMessage = $('#loginErrorMessage');
            loginMessage.text('');
            var form = $('form');
            var formData = form.serialize();


            if (form.valid() == false) return;


            button.prop('disabled', true);
            $.ajax({
                method: 'POST',
                url: '@Url.Action("LogIn", "Account")',
                data: formData,
                success: function (data) {
                    if (data.succeeded) {
                        window.location = data.url;
                    }
                    else {
                        loginMessage.text(data.errorMessage);
                    }
                }
            }).done(() => {
                button.prop('disabled', false);
            });
        }
```

Obrázek 105: Funkce pro odeslání požadavku pro přihlášení

Po stisknutí tlačítka „Přihlásit“ je zavolána funkce *login()*. Samotné tlačítko se stane neaktivním, aby nedošlo k vícenásobnému odeslání formuláře. Poté proběhne validace

zadaných dat. Pokud vše odpovídá pravidlům, které jsou stanoveny, je zavolána funkce *serialize()* na proměnnou, která obsahuje HTML element formuláře. Tato funkce zapíše všechna data z formuláře do jednoho textového řetězce. Tato data jsou oddělena znakem *&*. Následně je vytvořen asynchronní požadavek. V něm je stanovena HTTP metoda, URL obsahující jméno kontroleru a metody, kterou chceme volat, samotná data z formuláře a nakonec funkce, která se provede při úspěšném provedení požadavku. Pokud dojde k úspěšnému přihlášení, uživatel je přesměrován na výpis přiřazených srazů. URL pro přesměrování je součástí výsledku operace přihlášení. Obsahuje ho třída *LogInResult*. Pokud přihlášení selhalo, je zobrazena chybová hláška. Nakonec, ať už byla operace provedena úspěšně nebo ne, je odblokováno tlačítko „Přihlásit“.

```
public IActionResult LogIn()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> LogIn(LoginViewModel loginViewModel)
        {
            var logInResult = await
accountApplicationService.LogIn(loginViewModel);

            if (logInResult.Controller != "")
                logInResult.Url = Url.Action("ShowMeetups",
logInResult.Controller);

            return Json(logInResult);
        }
```

Obrázek 106: Metody v kontroleru – přihlášení

```
public async Task<LogInResult> LogIn(string email, string password)
        {
            var user = await userManager.FindByEmailAsync(email);
```

```
            if (user == null)
                return new LogInResult(false,
AccountResources.UserNotFoundMessage, "");
            string errorMessage = "";

            try
            {
                var result = await signInManager.PasswordSignInAsync(user,
password, false, false);
                if (result.Succeeded)
                {
                    var roles = await userManager.GetRolesAsync(user);
                    string controller = "";

                    if (roles.Contains(AccountResources.AdminRole))
                        controller = MeetupResources.AdminControllerName;
                    else
                        controller =
MeetupResources.ParticipantControllerName;

                    return new LogInResult(true, "", controller);
                }
                else
                    return new LogInResult(false,
AccountResources.LoginFailedMessage, "");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
            }
```

```
            return new LogInResult(false, errorMessage, "");
        }
```

Obrázek 107: Metoda v account service – přihlášení

Account service využívá instanci *User managera* dodanou pomocí *dependency injection*. Údaje od uživatele jsou použity jako argumenty do metody *PasswordSignInAsync()*. Metoda vrací *SignInResult*, který obsahuje proměnnou udávající úspěch přihlášení. Pokud přihlášení proběhlo úspěšně, je ještě zjištěno, zda je uživateli přiřazena role administrátora. Podle toho se vybírá kontroler, na kterém se posléze zavolá metoda *ShowMeetups()*.

4.6.2 Požadavek zobrazení detailu srazu

Požadavek může být vyslán ze dvou míst. Pomocí tlačítka „Otevřít“ u srazu nebo z menu pomocí položky „Setkání“. V prvním případě je odeslán požadavek s id srazu. Musí nejdříve dojít ke kontrole, zda byl požadovaný sraz účastníkovi skutečně přiřazen. Po úspěšné kontrole je zapamatováno id vybraného srazu a účastník je přesměrován na stránku detailu srazu.

```
[HttpPost]
        public async Task<IActionResult> GetMeetup(int id)
        {
            string errorMessage = "";
            bool isValid = false;

            try
            {
                isValid = await IsValidMeetupId(id);
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return Json(new GetMeetupResult(false, errorMessage, ""));
            }

            if (isValid == false)
                return Json(new GetMeetupResult(false,
MeetupResources.MeetupAccessDeniedMessage, ""));
```

```
            SetCurrentMeetupId(id);

            return Json(new GetMeetupResult(true, "",
Url.Action(nameof(Index))));
        }
```

Obrázek 108: Metoda v kontroleru – zobrazení srazu 1.

```
public bool IsValidMeetup(Participant participant, int meetupId)
        {
            var participantJunctions =
dbContext.MeetupParticipantJunction.Where(j => j.ParticipantId ==
participant.Id);

            if (participantJunctions != null && participantJunctions.Count()
> 0)
            {
                var validMeetupIds = participantJunctions.Select(j =>
j.MeetupId);

                if (validMeetupIds.Contains(meetupId))
                    return true;
                else
                    return false;
            }
            else
                return false;
        }
```

Obrázek 109: Metoda v participant service – kontrola srazu

V případě že už byl vybrán sraz, je účastník přesměrován na metodu *Index()*.

```
public async Task<IActionResult> Index()
        {
            int currentMeetupId = CurrentMeetupId();

            if (currentMeetupId == -1) return
RedirectToAction(nameof(ShowMeetups));
            else
            {
                var viewModel = await
participantApplicationService.GetMeetup(HttpContext.User, currentMeetupId);
```

```
                ViewBag.CurrentParticipantId = await
accountApplicationService.GetCurrentParticipantId(HttpContext.User);

                return View(viewModel);
            }
        }
```

Obrázek 110: Metoda v kontroleru – zobrazení srazu 2.

```
public async Task<MeetupIndexViewModel> GetMeetup(ClaimsPrincipal principal,
int meetupId)
        {
            Meetup meetup = participantService.GetMeetup(meetupId);
            Participant participant = await
accountService.GetCurrentParticipant(principal);

            var participants =
participantService.GetParticipantsWithParticipating(meetupId).Select(r =>
r.Participant);
            var guests = participantService.GetGuests(meetupId);
            var guestIds =
participantService.GetGuestsOfParticipant(participant).Select(g => g.Id);

            var participantViewModels = GetParticipants(meetupId);
            var guestViewModels = guests.Select(g => new GuestViewModel
            {
                Id = g.Id,
                FirstName = g.FirstName,
                LastName = g.LastName,
                Gender = g.Gender == Data.Entities.Gender.Male ?
Models.Gender.Male : Models.Gender.Female
            });

            return new MeetupIndexViewModel
            {
                Name = meetup.Name,
                IsActive = meetup.IsActive,
                Time = meetup.Time.ToString(DateTimeResources.TimeFormat),
                DateUpcoming = meetup.DateUpcoming,
                GuestIds = guestIds,
                Participants = participantViewModels,
```

```
                Guests = guestViewModels
            };
        }
```

Obrázek 111: Metoda v participant application service – zobrazení srazu

4.6.3 Požadavek na změnu stavu účasti

Účastník srazu má na výběr ze dvou stavů účasti: „Ano“ a „Ne“. Pokud je jeho aktuální stav „Ne“, je možné kliknout pouze na tlačítko opačného stavu. Po kliknutí na jedno z tlačítek je odeslán požadavek na změnu stavu účasti.

```
function sendYes() {
            sendIsParticipating(true);
        }
        function sendNo() {
            sendIsParticipating(false);
        }

        function sendIsParticipating(isParticipating) {
            var token = $('input[name="__RequestVerificationToken"]').val();

            $.ajax({
                method: 'POST',
                url: '@Url.Action("ChangeIsParticipating",
MeetupResources.ParticipantControllerName)',
                data: { isParticipating: isParticipating,
__RequestVerificationToken: token },
                success: function (data) {
                    if (data.succeeded) {
                        toggleDisabled(isParticipating);
                        setCounts();
                    }
                    else {
                        $('.failedAlert').css('display', 'flex')
                            .children('.alertText').text(data.errorMessage);
                    }
                }
            });
        }
```

Obrázek 112: Funkce pro odeslání požadavku na změnu stavu účasti

K datům požadavku je přidán také CSRF token, který je vygenerován pokaždé, když má formulář nastavenou metodu na „POST“. Po úspěšné změně stavu je zablokováno tlačítko, které odstartovalo požadavek, a tlačítko opačného stavu se stane aktivním. Dále jsou přepočítány počty účastníků, kteří se zúčastní dalšího srazu. Pokud dojde k chybě, je zobrazena chybová hláška.

```
[HttpPost]
        public async Task<IActionResult> ChangeIsParticipating(bool
isParticipating)
        {
            int currentMeetupId = CurrentMeetupId();

            var output = await
participantApplicationService.ChangeIsParticipating(HttpContext.User,
currentMeetupId, isParticipating);

            return Json(output);
        }
```

Obrázek 113: Metoda v kontroleru – změna stavu účasti

```
public BaseResult ChangeIsParticipating(Participant participant, int
meetupId, bool isParticipating)
        {
            bool succeeded = false;
            string errorMessage = "";

            try
            {
                var junction = dbContext.MeetupParticipantJunction
                    .FirstOrDefault(j => j.ParticipantId == participant.Id &&
j.MeetupId == meetupId);

                if (junction != null)
                {
                    junction.IsParticipating = isParticipating;

                    int changes = dbContext.SaveChanges();

                    if (changes > 0)
                        succeeded = true;
```

```
                    else
                        errorMessage =
MeetupResources.IsParticipatingChangeFailedMessage;
                }
                else
                    errorMessage =
MeetupResources.IsParticipatingChangeFailedMessage;
            }
            catch (Exception e)
            {
                errorMessage =
MeetupResources.IsParticipatingChangeFailedMessage + " " + e.Message;
            }

            return new BaseResult(succeeded, errorMessage);
        }
```

Obrázek 114: Metoda v participant service – změna stavu účasti

4.6.4 Požadavek na přidání zprávy

Účastník klikne na tlačítko „Nová zpráva“ a zobrazí se mu formulář pro přidání nové zprávy. Po zadání zprávy a její validaci se nová zpráva zařadí na první místo v kolekci zpráv.

```
function addMessage() {

            var messageText = $('.messageTextArea').first().val();
            var messageTable = $('.messageTable').first();
            var form = $('.addMessageForm');
            var token = $('input[name="__RequestVerificationToken"]').val();

            if (form.valid() == false) return;

            $.ajax({
                method: 'POST',
                url:                                '@Url.Action("CreateMessage",
MeetupResources.ParticipantControllerName)',
                data: { text: messageText, __RequestVerificationToken: token
},
                success: function (data) {
                    if (data.succeeded) {
                        $('.noMessagesHeader').hide();
```

```
                        messageTable.before(
                            '<div class="message">'+
                                '<div class="messageHeaderSection">'+
                                    '<span      class="messageHeader">'      +
data.participantFullName +', '+ data.date +'</span>'+
                            '</div>' +
                            '<div  class="messageBody">'  +  data.messageText  +
'</div>' +
                            '</div>'
                        );
                    }
                    else
                        $('.failedAlert').css('display',
'flex').children('.alertText').text(data.errorMessage);
                }
            });
        }
```

Obrázek 115: Funkce pro přidání nové zprávy

```
[HttpPost]
        public async Task<IActionResult> CreateMessage(string text)
        {
            int currentMeetupId = CurrentMeetupId();

            var output = await
participantApplicationService.CreateMessage(HttpContext.User,
currentMeetupId, text);

            return Json(output);
        }
```

Obrázek 116: Metoda kontroleru – přidání nové zprávy

```
public async Task<CreateMessageResult> CreateMessage(ClaimsPrincipal
principal, int meetupId, string text)
        {
            Participant participant = await
accountService.GetCurrentParticipant(principal);
            var cestTimeZone =
TimeZoneInfo.FindSystemTimeZoneById(DateTimeResources.TimeZoneName);
            Message message = new Message
```

```
            {
                MeetupId = meetupId,
                Participant = participant,
                ParticipantFullName = participant.FirstName + " " +
participant.LastName,
                Text = text,
                Date =
TimeZoneInfo.ConvertTimeFromUtc(DateTime.Now.ToUniversalTime(), cestTimeZone)
            };
            var result = participantService.CreateMessage(message);
            return new CreateMessageResult(
                result.Succeeded,
                result.ErrorMessage,
                result.ParticipantFullName,
                result.MessageText,
                result.Date
                );
        }
```

Obrázek 117: Metoda participant application service – přidání nové zprávy

```
public CreateMessageResult CreateMessage(Message message)
        {
            bool succeeded = false;
            string errorMessage = "";
            string participantFullName = "";
            string messageText = "";
            string date = "";

            try
            {
                dbContext.Messages.Add(message);

                int changes = dbContext.SaveChanges();

                if (changes > 0)
                {
                    succeeded = true;
                    participantFullName = message.Participant.FirstName + " "
+ message.Participant.LastName;
                    messageText = message.Text;
```

```
                    date =
message.Date.ToString(DateTimeResources.FullDateTimeFormat);
                }
                else
                    errorMessage =
MeetupResources.CreateMessageFailedMessage;
            }
            catch (Exception e)
            {
                errorMessage = MeetupResources.CreateMessageFailedMessage + "
" + e.Message;
            }

            return new CreateMessageResult(succeeded, errorMessage,
participantFullName, messageText, date);
        }
```

Obrázek 118: Metoda participant service – přidání nové zprávy

4.6.5 Požadavek na přiřazení srazu k účastníkovi

Administrátor může ve výpisu všech uživatelů přiřadit srazy jednotlivým účastníkům. U požadovaného uživatele si vybere sraz z dropdown menu a klikne na tlačítko „Přiřadit“. Po úspěšném přiřazení se název srazu smaže z dropdown menu nepřiřazených srazů a přidá se do dropdown menu přiřazených srazů.

```
function addMeetup() {

        var assignBtn = $(this);
        var participantId = assignBtn.val();
        var meetupId = assignBtn.parent().prev().val();
        var token = $('input[name="__RequestVerificationToken"]').val();

        $.ajax({
            method: 'POST',
            url: '@Url.Action("AddMeetupToParticipant", "Admin")',
            data: { meetupId: meetupId, participantId: participantId,
__RequestVerificationToken: token },
            success: function (data) {
                if (data.succeeded) {
                    var user = assignBtn.parents('.user');
```

```
                    var unassignedMeetupsDropdown =
user.find('.unassignedMeetupsDropdown').first();
                    var assignedMeetupsDropdown =
user.find('.assignedMeetupsDropdown').first();
                    assignedMeetupsDropdown.prepend('<option value="' +
data.id + '">' + data.name + '</option>')

                    unassignedMeetupsDropdown.find(':selected').remove();
                    assignedMeetupsDropdown.find('option[value=' + data.id +
']').attr('selected', 'selected');
                }
                else
                    $('.failedAlert').css('display',
'flex').children('.alertText').text(data.errorMessage);
            }
        });
    }
```

Obrázek 119: Funkce pro přiřazení srazu

```
public AssignResult AddMeetupToParticipant(int meetupId, int participantId)
        {
            bool succeeded = false;
            string errorMessage = "";
            int id = meetupId;
            string name = "";

            try
            {
                var participant = dbContext.Participants.Where(p => p.Id ==
participantId).FirstOrDefault();
                var meetup = dbContext.Meetups.Where(m => m.Id ==
meetupId).FirstOrDefault();

                if (participant is null || meetup is null)
                    return new AssignResult(false,
MeetupResources.AddMeetupToParticipantFailedMessage, 0, "");

                MeetupParticipantJunction newJunction = new
MeetupParticipantJunction
                {
```

```
                    Meetup = meetup,
                    Participant = participant,
                    IsParticipating = false,
                    Notify = true
                };

                dbContext.MeetupParticipantJunction.Add(newJunction);
                int changes = dbContext.SaveChanges();

                if (changes > 0)
                {
                    succeeded = true;
                    name = $"{ meetup.Name }, { meetup.Location }";
                }
                else
                    errorMessage =
MeetupResources.AddMeetupToParticipantFailedMessage;
            }
            catch (Exception e)
            {
                errorMessage =
MeetupResources.AddMeetupToParticipantFailedMessage + " " + e.Message;
            }

            return new AssignResult(succeeded, errorMessage, meetupId, name);
        }
```

Obrázek 120: Metoda admin service – přiřazení srazu k účastníkovi

4.6.6 Požadavek na zaslání hromadné zprávy účastníkům srazu

```
function sendParticipantMessage() {

            var button = $('.createMeetupBtn');

            var form = $('.adminMessageForm');
            if (form.valid() == false) return;

            button.prop('disabled', true);

            var message = $('.messageTextArea').val();
            var subject = $('.subject').val();

            var list = $('.messageCheckbox:checked');
            var emails = [];
            list.each((i, item) => {
```

```
                var email = $(item).parent().next().next().text();
                emails.push(email);
            });
            var token = $('input[name="__RequestVerificationToken"]').val();

            $.ajax({
                method: 'POST',
                url: '@Url.Action("SendMessageToParticipants")',
                data: { message: message, emails: emails, subject: subject,
__RequestVerificationToken: token },
                success: function (data) {
                    if (data.succeeded) {
                        $('.succeededAlert').css('display',
'flex').children('.alertText').text(data.succeededMessage);
}
                    else
                        $('.failedAlert').css('display',
'flex').children('.alertText').text(data.errorMessage);
                }
            }).done(() => {
                button.prop('disabled', false);
            });
        }
```

Obrázek 121: Funkce pro zaslání hromadné zprávy

```
[HttpPost]
        public IActionResult SendMessageToParticipants(string message, string
subject, string[] emails)
        {
            (string email, string password) = GetEmailAndPassword();

            var output =
adminApplicationService.SendMessageToParticipants(message, subject, emails,
email, password);

            return Json(output);
        }
```

Obrázek 122: Metoda kontroleru – zaslání hromadné zprávy

```
public SendMessageResult SendMessageToParticipants(string message, string
subject, string[] emails, string email, string password)
        {
            EmailSender.EmailSender emailSender = new
EmailSender.EmailSender(email, password);
            (bool succeeded, string errorMessage) = emailSender.Send(emails,
message, subject);

            if
(errorMessage.Contains(EmailResources.SecureConnectionErrorMessagePart))
                errorMessage =
EmailResources.SecureConnectionResolutionMessage;

            return new SendMessageResult(succeeded, errorMessage,
EmailResources.EmailSentMessage);
        }
```

Obrázek 123: Metoda admin service – zaslání hromadné zprávy

```
public (bool, string) Send(IEnumerable<string> emails, string text, string
subject)
        {
            bool result = false;
            string errorMessage = "";

            try
            {
                SmtpClient client = new SmtpClient(SmtpClient);
                client.Port = 587;
                client.UseDefaultCredentials = false;
                client.Credentials = new NetworkCredential(Email, Password);
                client.EnableSsl = true;

                MailMessage message = new MailMessage();
                message.From = new MailAddress(Email);

                string addresses = "";
                for (int i = 0; i < emails.Count(); i++)
                {
                    if (i == (emails.Count() - 1))
                        addresses += emails.ElementAt(i);
                    else
                        addresses += emails.ElementAt(i) + ",";
                }

                message.To.Add(addresses);
                message.Subject = subject;
                message.Body = text;

                client.Send(message);
                result = true;
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
            }

            return (result, errorMessage);
        }
```

Obrázek 124: Metoda pro zaslání e-mailu na dané adresy

4.7 Periodické operace

V rámci běhu aplikace je nutné zajistit vykonání určitých operací v určité datum a čas. Jedná se o operaci, která se vykoná právě v čas uskutečnění srazu a také o operaci, která upozorní účastníky na blížící se sraz. Datum a čas jsou dány při tvorbě nového srazu nebo jeho úpravě.

```
function submitMeetupCreation() {
            var form = $('#createMeetupForm');
            if (form.valid() == false) return;


            var button = $('.createMeetupBtn');
```

```
            button.prop('disabled', true);
            var name = $('#Name').val();
            var location = $('#Location').val();
            var period = $('#Period').val();
            var dayOfWeek;
            var dateCreated;

            if (period == 'Weekly')
                dayOfWeek = $('.dayOfWeekSelector').val();
            else if (period == 'Monthly')
                dateCreated = $('.dateSelector').val();

            var time = $('#Time').val();
            var notify = $('#InAdvanceValue').val();
            var token = $('input[name="__RequestVerificationToken"]').val();

            $.ajax({
                method: 'POST',
                url: '@Url.Action("CreateMeetup", "Admin")',
                data: {
                    Name: name,
                    Location: location,
                    Period: period,
                    Time: time,
                    DayOfWeek: dayOfWeek,
                    DateCreated: dateCreated,
                    InAdvanceValue: notify,
                    __RequestVerificationToken: token
                },
                success: function (data) {
                    if (data.succeeded)
                        window.location = data.url;
                    else
                        $('.failedAlert').css('display',
'flex').children('.alertText').text(data.errorMessage);
                }
            }).done(() => {
                button.prop('disabled', false);
            });
```

```
        }
```

Obrázek 125: Funkce pro vytvoření nového srazu

```
public IActionResult CreateMeetup()
        {
            return View();
        }
[HttpPost]
public IActionResult CreateMeetup(MeetupViewModel viewModel)
        {
            (string email, string password) = GetEmailAndPassword();
            var result = adminApplicationService.CreateMeetup(viewModel,
email, password);
            if (result.Succeeded)
                result.Url = Url.Action(nameof(ShowMeetups),
MeetupResources.AdminControllerName);


            return Json(result);
        }
```

Obrázek 126: Metoda kontroleru – nový sraz

```
public CreateMeetupResult CreateMeetup(MeetupViewModel viewModel, string
email, string password)
        {
            string errorMessage = "";
            try
            {
                Data.Entities.Period period = Data.Entities.Period.Daily;
                switch (viewModel.Period)
                {
                    case Models.Period.Daily:
                        period = Data.Entities.Period.Daily;
                        break;
                    case Models.Period.Weekly:
                        period = Data.Entities.Period.Weekly;
                        break;
                    case Models.Period.Monthly:
                        period = Data.Entities.Period.Monthly;
                        break;
                }
                Meetup meetup = new Meetup();
```

```
                meetup.Name = viewModel.Name;
                meetup.Location = viewModel.Location;
                meetup.IsActive = true;
                meetup.Period = period;
                meetup.Time = viewModel.Time;
                meetup.DateCreated = DateTime.Now;
                meetup.NotifyInAdvance = viewModel.InAdvanceValue;
                meetup.History = new List<HistoryRecord>();
                meetup.DateUpcoming = SetUpcomingDate(viewModel.Time,
viewModel.DayOfWeek, viewModel.Period, viewModel.DateCreated, DateTime.Now);


                var result = adminService.CreateMeetup(meetup, email,
password);
                return new CreateMeetupResult(result.Succeeded,
result.ErrorMessage);
            }
            catch (Exception e)
            {
                errorMessage = MeetupResources.CreateMeetupFailedMessage + "
" + e.Message;
            }
            return new CreateMeetupResult(false, errorMessage);
        }
```

Obrázek 127: Metoda admin application service – nový sraz

```
public BaseResult CreateMeetup(Meetup meetup, string email, string password)
        {
            bool succeeded = false;
            string errorMessage = "";
            try
            {
                var entry = dbContext.Meetups.Add(meetup);
                int changes = dbContext.SaveChanges();
                if (changes >= 1)
                {
                    RecurringTasks.RecurringTasks recurringTasks = new
RecurringTasks.RecurringTasks(dbContext);
                    switch (meetup.Period)
                    {
                        case Period.Daily:
```

```
                            (succeeded, errorMessage) =
recurringTasks.CreateRecurringDailyMeetupTask(meetup.Time,
                                meetup.NotifyInAdvance, entry.Entity.Id,
email, password);
                            break;
                        case Period.Weekly:
                            (succeeded, errorMessage) =
recurringTasks.CreateRecurringWeeklyMeetupTask(meetup.Time,
                                meetup.DateUpcoming.DayOfWeek,
meetup.NotifyInAdvance, entry.Entity.Id, email, password);
                            break;
                        case Period.Monthly:
                            (succeeded, errorMessage) =
recurringTasks.CreateRecurringMonthlyMeetupTask(meetup.Time,
                                meetup.DateCreated, meetup.NotifyInAdvance,
entry.Entity.Id, email, password);
                            break;
                    }
                }
                else
                    errorMessage = MeetupResources.CreateMeetupFailedMessage;
            }
            catch (Exception e)
            {
                errorMessage = MeetupResources.CreateMeetupFailedMessage + "
" + e.Message;
            }
            return new BaseResult(succeeded, errorMessage);
        }
```

Obrázek 128: Metoda admin service – nový sraz

Pro každý sraz je nutné vygenerovat dva různé *cron* výrazy. Jeden pro *MeetupTask* a druhý pro *MeetupNotifyTask*.

```
public (bool, string) CreateRecurringDailyMeetupTask(DateTime time, int
hoursInAdvance, int meetupId, string email, string password)
        {
            bool succeeded = false;
            string errorMessage = "";
            TimeZoneInfo timeZoneInfo =
TimeZoneInfo.FindSystemTimeZoneById(DateTimeResources.TimeZoneName);
            string cron = DateTimeResources.DailyCron(time, 0);
```

```
            string jobId = MeetupResources.MeetupTaskName +
meetupId.ToString();
            try
            {
                RecurringJob.AddOrUpdate(
                    recurringJobId: jobId,
                    cronExpression: cron,
                    methodCall: () => MeetupTask(meetupId),
                    timeZone: timeZoneInfo
                );
                succeeded = true;
            }
            catch (Exception e)
            {
                succeeded = false;
                errorMessage =
MeetupResources.CreateRecurringTaskFailedMessage + " " + e.Message;
            }
            if (succeeded == false)
                return (succeeded, errorMessage);
            cron = DateTimeResources.DailyCron(time, hoursInAdvance);
            jobId = MeetupResources.MeetupNotifyTaskName +
meetupId.ToString();
            try
            {
                RecurringJob.AddOrUpdate(
                    recurringJobId: jobId,
                    cronExpression: cron,
                    methodCall: () => MeetupNotifyTask(meetupId, email,
password),
                    timeZone: timeZoneInfo
                );
                succeeded = true;
            }
            catch (Exception e)
            {
                succeeded = false;
                errorMessage =
MeetupResources.CreateRecurringTaskFailedMessage + " " + e.Message;
            }
            return (succeeded, errorMessage);
```

```
        }
```

Obrázek 129: Vytvoření operací, které se opakují denně

```
public void MeetupTask(int meetupId)
        {
            var meetup = dbContext.Meetups.Include(m => m.History).Include(m
=> m.Guests).Where(m => m.Id == meetupId).FirstOrDefault();
            int day = meetup.DateUpcoming.Day;
            meetup.DateUpcoming =
DateTimeResources.UpdateUpcomingDate(meetup.Period, meetup.DateUpcoming);
            if (day == DateTime.Now.Day && meetup.IsActive)
            {
                var junctions = dbContext.MeetupParticipantJunction.Include(j
=> j.Participant).Where(j => j.MeetupId == meetupId);
                foreach (var junction in junctions)
                {
                    meetup.History.Add(new HistoryRecord
                    {
                        Date = DateTimeResources.ZeroTime(DateTime.Now),
                        Meetup = meetup,
                        ParticipantFullName = junction.Participant.FirstName
+ " " + junction.Participant.LastName,
                        ParticipantId = junction.ParticipantId,
                        Participated = junction.IsParticipating
                    });

                    junction.IsParticipating = false;

                var guests = dbContext.Guests.Where(g => g.MeetupId ==
meetupId);
                if (guests != null && guests.Count() > 0)
                    dbContext.Guests.RemoveRange(guests);

                dbContext.SaveChanges();
            }
        }
```

Obrázek 130: MeetupTask

```
public void MeetupNotifyTask(int meetupId, string email, string password)
        {
            var meetup = dbContext.Meetups.Where(m => m.Id ==
meetupId).FirstOrDefault();
```

```
            if (meetup.IsActive)
            {
                var participants = dbContext.MeetupParticipantJunction
                    .Include(j => j.Participant).Where(j => j.MeetupId ==
meetupId && j.Notify).Select(j => j.Participant);
                if (participants != null && participants.Count() > 0)
                {
                    var emails = participants.Select(p => p.Email);
                    string message =
EmailResources.NotifyMessage(meetup.DateUpcoming);
                    string subject = EmailResources.NotifySubject;


                    EmailSender.EmailSender emailSender = new
EmailSender.EmailSender(email, password);
                    emailSender.Send(emails, message, subject);
                }
            }
        }
```

Obrázek 131: MeetupNotifyTask

ZÁVĚR

V rámci této práce se mi podařilo vypracovat aplikaci, která poskytuje rozšířenou funkcionalitu oproti právě využívanému systému. Vedoucí týmu má díky ní přehled o údajích uživatelů, o jejich účastech a může spravovat srazy. Účastnící mají k dispozici jednoduchý způsob zapisování na tyto srazy, k čemuž využívají svůj vlastní účet, nad kterým mají kontrolu a mohou ho kdykoliv upravit. Je jim umožněno nahlašovat hosty na nadcházející konání srazu a komunikovat s ostatními prostřednictvím zpráv.

V porovnání s aktuálním systémem se aplikace liší v mnoha směrech. Na rozdíl od něj je aplikace konstruována „na míru“ a není k dispozici každému, kdo by ji chtěl používat. Také některé funkce, které měly být zachovány, jsou implementovány jinak. Kupříkladu vybírání stavu účasti. V každém případě aplikace zachovává žádoucí funkce, které poskytuje aktuálně používaný systém.

Při vývoji webu nebylo nutné provádět žádné finanční investice. To samé platí pro jeho udržení v provozu. V tomto ohledu jsou oba systémy stejné. Uživatelům neúčtují žádné poplatky za používání. Navíc máme možnost kdykoliv změnit chování aplikace podle potřeby.

SEZNAM POUŽITÉ LITERATURY

1. *Formulář založení nové skupiny* [online]. [cit. 2020-08-04]. Dostupné z: http://sejdemse.net/index.php?id=2
2. *O aplikaci Teamer.net* [online]. [cit. 2020-08-03]. Dostupné z: https://join.teamer.net/about
3. *O aplikaci Teamsnap.com* [online]. [cit. 2020-08-04]. Dostupné z: https://www.teamsnap.com/
4. *O aplikaci Tymuj.cz* [online]. [cit. 2020-08-04]. Dostupné z: https://tymuj.cz/o-nas
5. *Uživatel, týmové role* [online]. [cit. 2020-08-04]. Dostupné z: old.tymuj.cz/napoveda-uzivatel-a-tymove-role.html
6. *A tour of the C# language* [online]. 26. 02. 2020 [cit. 2020-08-05]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/
7. *Introduction to ASP.NET Core* [online]. 17. 04. 2020 [cit. 2020-08-05]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1
8. SMITH, Steve, Scott ADDIE a Brandon DAHLER. *Dependency injection in ASP.NET Core* [online]. 21. 06. 2020 [cit. 2020-08-05]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1
9. RASTOGI, Pranav, Rick ANDERSON, Tom DYKSTRA, Jon GALLOWAY a Erik REITAN. *Introduction to Identity* [online]. [cit. 2020-08-05]. Dostupné z: https://aspnetcore.readthedocs.io/en/stable/security/authentication/identity.html
10. SMITH, Steve. *Overview of ASP.NET Core MVC* [online]. 12. 02. 2020 [cit. 2020-08-05]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-3.1
11. GALLOWAY, Jon. *Professional ASP.NET MVC 5*. 1. Indianapolis: Wiley, c2014. Wrox programmer to programmer (Wiley). ISBN 978-1118794753.
12. *Entity Framework Core* [online]. 27. 10. 2016 [cit. 2020-08-05]. Dostupné z: https://docs.microsoft.com/cs-cz/ef/core/

13. *About SQLite* [online]. [cit. 2020-08-05]. Dostupné z: https://www.sqlite.org/about.html
14. *Datatypes In SQLite Version 3* [online]. [cit. 2020-08-05]. Dostupné z: https://www.sqlite.org/datatype3.html
15. *HTML basics* [online]. [cit. 2020-08-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
16. Introduction. *MDN Web Docs* [online]. [cit. 2020-08-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript
17. What is jQuery? *JQuery* [online]. [cit. 2020-08-05]. Dostupné z: https://jquery.com/
18. CSS Introduction. *W3schools* [online]. [cit. 2020-08-06]. Dostupné z: https://www.w3schools.com/css/css_intro.asp
19. CSS Specificity. *W3schools* [online]. [cit. 2020-08-06]. Dostupné z: https://www.w3schools.com/css/css_specificity.asp
20. *Microsoft Azure* [online]. [cit. 2020-08-06]. Dostupné z: https://azure.microsoft.com/cs-cz/free/
21. Manage Azure Resource Manager resource groups by using the Azure portal. *Azure documentation* [online]. [cit. 2020-08-06]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/azure-resource-manager/management/manage-resource-groups-portal
22. Azure App Service plan overview. *Azure documentation* [online]. [cit. 2020-08-06]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/app-service/overview-hosting-plans
23. App Service overview. *Azure documentation* [online]. [cit. 2020-08-06]. Dostupné z: https://docs.microsoft.com/en-us/azure/app-service/overview
24. Overview. *Hangfire* [online]. [cit. 2020-08-06]. Dostupné z: https://www.hangfire.io/overview.html

25. SQL Injection. *OWASP* [online]. [cit. 2020-08-06]. Dostupné z: https://owasp.org/www-community/attacks/SQL_Injection

26. Protecting Against SQL Injection. *Hacksplaining* [online]. [cit. 2020-08-06]. Dostupné z: https://www.hacksplaining.com/prevention/sql-injection

27. Security Considerations (Entity Framework). *.NET documentation* [online]. 03/30/2017 [cit. 2020-08-06]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations

28. WASSON, Mike. Preventing Cross-Site Request Forgery (CSRF) Attacks in ASP.NET MVC Application. *Microsoft Docs* [online]. 12. 12. 2012 [cit. 2020-08-06]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/web-api/overview/security/preventing-cross-site-request-forgery-csrf-attacks

29. ANDERSON, Rick. Prevent Cross-Site Scripting (XSS) in ASP.NET Core. *Microsoft Docs* [online]. 02. 10. 2018 [cit. 2020-08-06]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/security/cross-site-scripting?view=aspnetcore-3.1

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
SQL	Structured Query Language
CSRF	Cross Site Request Forgery
XSS	Cross Site Scripting
ASP	Active Server Pages
LINQ	Language Interacted Query

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD

Dokumentace a zdrojové kódy aplikace.

- Dokumentace
    - Janečka – Dokumentace.docx
    - Janečka – Dokumentace.pdf
- Zdrojový kód
    - janecka.zip