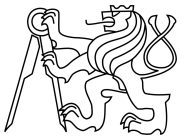
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém evidence účastníků akcí pro Klub českých turistů
Student:	Štěpán Adámek
Vedoucí:	Ing. Jan Čermák
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je návrh a implementace specializovaného backendového řešení evidence účastníků akcí pro Klub českých turistů vč. administračního rozhraní.

Požadované případy užití jsou:
1. Administrace systému - správa uživatelů, akcí a vztahů mezi nimi.
2. Odbavení účastníků turistických akcí předem před akcí, na začátku, v průběhu a v závěru akce.
3. Export dat z akce - jmenný seznam účastníků a jejich objednaných služeb, průchod kontrolními body.

Backend bude přijímat data od klientských aplikací ze strojově čitelného identifikátoru (QR kód na nových průkazkách klubu).

1. Seznamte se s problematikou, formulujte funkční a nefunkční požadavky na řešení.
2. Proveďte průzkum existujících řešení.
3. Navrhněte architekturu vlastního řešení.
4. Diskutujte a zvolte vhodnou implementační platformu.
5. Proveďte implementaci, zdokumentujte a otestujte ji.
6. Na základě testů dokončete aplikaci, zhodnoťte ji a naznačte další směr vývoje.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. prosince 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Systém evidence účastníků akcí pro Klub českých turistů

Štěpán Adámek

Vedoucí práce: Ing. Jan Čermák, katedra inženýrské informatiky FSv

15. května 2017

Poděkování

Rád bych poděkoval všem, kteří mě podporovali při psaní této práce. Děkuji vedoucímu práce Ing. Janu Čermákovi za cenné rady při psaní této práce a hlavně možnost spolupracovat na tak zajímavém projektu. Dále bych chtěl poděkovat všem, kteří mě podporovali po celou dobu studia, zejména mým přátelům, rodině, přitelkyni a členům kapely The Latecomers.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Adámek, Štěpán. *Systém evidence účastníků akcí pro Klub českých turistů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem práce je vytvořit návrh systému evidence účastníků akcí pro Klub českých turistů a na jeho základě vytvořit funkční systém, krerý poskytne webové API pro navazující aplikace. Práce popisuje vývoj aplikace od analýzy až po její testování a obsahuje API referenci.

Klíčová slova informační systém, .NET Core, MVC, webová aplikace, API, OAuth2, OpenId, C#

Abstract

The aim of this thesis is to design a system, that registers event participants for the Czech Tourists Club and create a functional system based on this design. System should provide a web API for follow-up applications. This work describes the application development from analysis to testing and contains API reference.

Keywords information system, .NET Core, MVC, web application, API, OAuth2, OpenId, C#

Obsah

Seznam obrázků

Úvod

Klub českých turistů sdružuje příznivce turistiky již od roku 1888. Mezi jeho hlavní činnosti patří organizace turistických akcí, se kterou má velké zkušenosti. Ne vždy je však organizace tak rychlá a efektivní, jak by mohla být s použitím moderních technologii.

Tyto důvody vedly k vytvoření vlastního systému pro správu akcí, který má za úkol organizátorům usnadnit jejich organizaci. Hlavním cílem systému je evidence přihlášek a její následné využití pro odbavování účastníků akcí. Dále má systém za úkol zpracování údajů o průchodu kontrolními stanovišti. Tato práce se věnuje pouze tvorbě backendu výsledného systému.

Práce začíná analýzou současného stavu řešení a požadavků na nově vznikající systém, aby bylo jasné, co všechno má umět. Tato analýza proběhla především na základě schůzek se zprostředkovatelem komunikace mezi mnou a Klubem českých turistů Ing. Janem Čermákem – vedoucím této práce. Na základě analýzy byl vytvořen seznam účastníků systému a model případů užití.

Návrh vychází z předchozí analýzy a klade si za cíl především vytvořit kvalitní rozhraní API, které je klíčovou součástí celého systému. Jako první jsou zvoleny vhodné technologie na základě požadavků, poté je vytvořen návrh datových tříd, které odpovídají modelu případů užití.

Realizace začíná volbou vhodných nástrojů pro vývoj a podpůrné činnosti. Dále je zde popsán vzhled výsledné aplikace, schéma databáze a předpokládaný model nasazení. Velký důraz je kladen na API referenci, která je naprosto zásadní pro vývoj navazujících aplikací.

V poslední části je rozebráno, jak probíhalo testování systému a jeho výsledky. Nakonec jsou zde nastíněny možnosti dalšího vývoje – jaké aplikace by bylo vhodné ještě vyvinout v návaznosti na tento systém.

Kapitola 1

Specifikace cíle

1.1 Cíl práce

Cílem této práce je navrhnout a vytvořit backendové řešení nového informačního systému pro organizaci a správu akcí Klubu českých turistů, který využije strojově čitelná data na nových průkazkách členů klubu. Jeho uživateli budou organizátoři z řad KČT. Systém má mít administrační rozhraní a webové API, které umožní dalším aplikacím ze vznikajícího systému přístup k datům.

Hlavní motivací pro vývoj takovéto aplikace je fakt, že v současné době v rámci KČT neexistuje žádný centralizovaný systém, který by řešil problematiku správy akcí organizovaných klubem. Předpokládá se, že takovýto systém značně zpříjemní a zrychlí organizaci při minimálních nákladech – QR Kódy vytištěné na průkazkách členů.

Kapitola 2

Analýza

V této části je popsán současný stav řešení a jeho problémy. Následuje analýza funkčních a nefunkčních požadavků. Analýza vychází především z rozhovorů s prostředníkem mezi mnou a KČT Ing. Janem Čermákem – vedoucím této práce. Také byly využity již existující přihlášky na akce v tištěné podobě.

2.1 Současný stav

V současné době jsou všechna data zpracována ručně. Členové klubu sice mají nové průkazky se strojově čitelnými údaji v podobě QR kódu, ale ty se k registraci ani identifikaci členů zatím nepoužívají. Přihlášku na akci je možné vytisknout a vyplněnou zaslat poštou. Ukázku takovéto přihlášky nalezneme na obrázcích 2.1 a 2.2. Dále se lze registrovat pomocí elektronické pošty nebo webového formuláře (nasbíraná data musí být potom také zpracována ručně). K identifikaci přihlášených v místě konání pak slouží seznam účastníků, ve kterém organizátoři účastníky vyhledávají ručně. Pokud přijde někdo bez registrace, potom musí být registrován až na místě. Kontrola účastníků v průběhu akce na kontrolních stanovištích opět probíhá bez použití strojově čitelných údajů a organizátoři musí kontrolovat údaje na průkazkách nebo kartách účastníků, které dostanou při odbavení.

Další možností registrace na některé akce klubu je použití aplikace třetí strany – např. Google Forms [3], které se využívají zejména pro registraci účastníků na interní akce. Probíhá také testovací využití RFID tagů, které ale není moc úspěšné kvůli jejich vysoké ceně.

2.1.1 Nevýhody současného stavu

- Absence komplexního a jednotného systému, který by mohly využít všechny odnože KČT.

PŘIHLÁŠKA NA POCHOD ZA POSLEDNÍM PUCHÝŘEM 2017
odešlete pořadatelům nejpozději do 15. 9. 2017

Jméno _____ Příjmení _____
Adresa _____
Datum narození _____ Členské číslo průkazu KČT _____
Telefon _____ E-mail _____

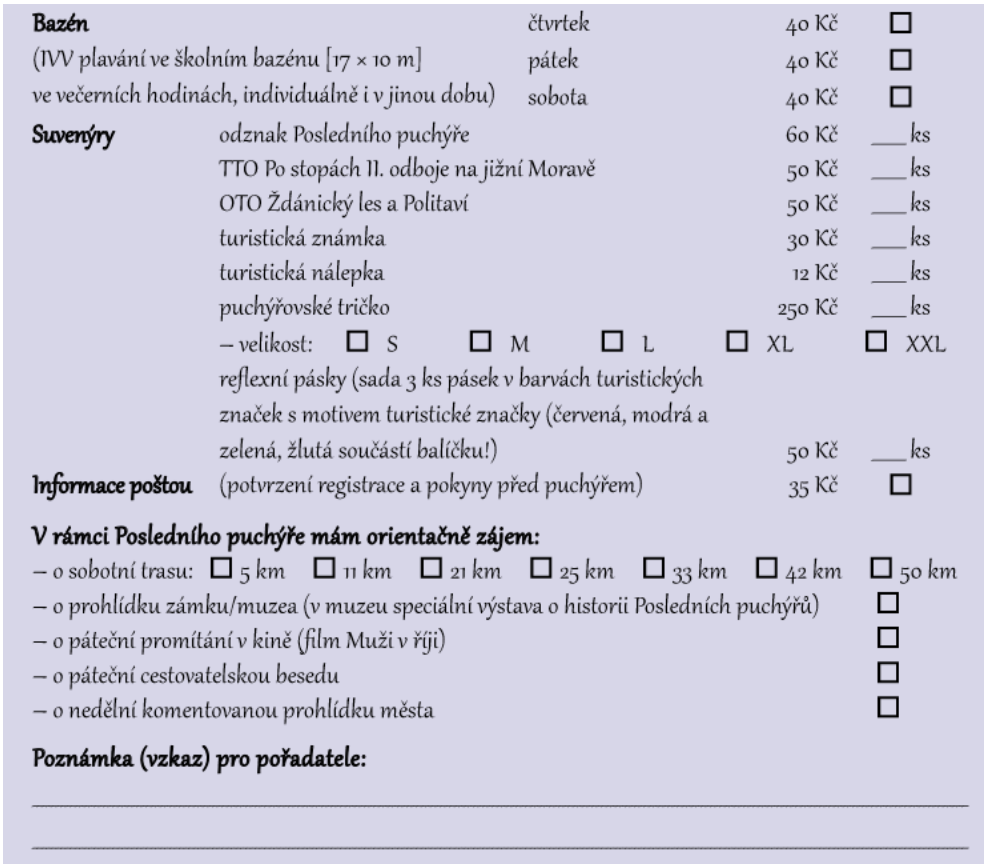
Účastnický poplatek	člen KČT	80 Kč	☐
	nečlen KČT	100 Kč	☐
Ubytování	čtvrtek/pátek	60 Kč	☐
(tělocvičny a třídy)	pátek/sobota	60 Kč	☐
	sobota/neděle	60 Kč	☐
Stravování	čtvrtek – večeře	90 Kč	☐
	pátek – snídaně	50 Kč	☐
	pátek – večeře	90 Kč	☐
	sobota – snídaně	50 Kč	☐
	sobota – večeře	90 Kč	☐
	neděle – snídaně	50 Kč	☐
Zájezdy	1 – Slavkovské bojiště (Mohyla míru, Žuráň, Slavkov)	250 Kč	☐
(pátek 17. 11. 2017)	2 – Chřiby (Bunč, Brdo, Modrá, Velehrad)	250 Kč	☐
	3 – Slovácko 1 – Strážnicko (Tvarožná Lhota, Plže)	250 Kč	☐
	4 – Hanácké Slovácko (Klobouky, Židlochovice, Těšany, v ceně závěrečné posezení ve sklípku)	350 Kč	☐
	5 – Slovácko 2 – Čejkovicko (Vrbice, Modré hory) (v ceně vstup do Templářských sklepů a degustace)	400 Kč	☐
	V případě obsazenosti vybraného zájezdu mám zájem o zájezd č.:		___
Puchýřovská zábava	(country skupina Pantofláči, kapacita sálu omezena!!!)	150 Kč	☐

Obrázek 2.1: Za posledním puchýřem 2017 – přihláška strana 1 [1]

- Náročnost manuálního zpracování přihlášek na větší akce.
- Není využito výhod strojové čitelnosti údajů na průkazkách členů.
- Pomalé odbavování účastníků při příchodu na akci a kontrolních stanovištích.
- Vysoká nákladovost RFID technologie.

2.2 Analýza požadavků

V této sekci jsou analyzovány funkční a nefunkční požadavky na výsledný backend systému.



Bazén		čtvrtek	40 Kč	☐
(IVV plavání ve školním bazénu [17 × 10 m]		pátek	40 Kč	☐
ve večerních hodinách, individuálně i v jinou dobu)		sobota	40 Kč	☐
Suvenýry	odznak Posledního puchýře		60 Kč	___ ks
	TTO Po stopách II. odboje na jižní Moravě		50 Kč	___ ks
	OTO Ždánický les a Politaví		50 Kč	___ ks
	turistická známka		30 Kč	___ ks
	turistická nálepka		12 Kč	___ ks
	puchýřovské tričko		250 Kč	___ ks
	– velikost: ☐ S ☐ M ☐ L ☐ XL ☐ XXL			
	reflexní pásky (sada 3 ks pásek v barvách turistických značek s motivem turistické značky (červená, modrá a zelená, žlutá součástí balíčku!)		50 Kč	___ ks
Informace poštou	(potvrzení registrace a pokyny před puchýřem)		35 Kč	☐

V rámci Posledního puchýře mám orientačně zájem:

– o sobotní trasu: ☐ 5 km ☐ 11 km ☐ 21 km ☐ 25 km ☐ 33 km ☐ 42 km ☐ 50 km
– o prohlídku zámku/muzea (v muzeu speciální výstava o historii Posledních puchýřů) ☐
– o páteční promítání v kině (film Muži v říji) ☐
– o páteční cestovatelskou besedu ☐
– o nedělní komentovanou prohlídku města ☐

Poznámka (vzkaz) pro pořadatele:

Obrázek 2.2: Za posledním puchýřem 2017 – přihláška strana 2 [1]

2.2.1 Funkční požadavky

1. **Správa akcí** – systém umožní pořadatelům vytvářet a upravovat akce, včetně kontrolních stanovišť, tras a poskytovaných služeb.

2. **Správa tras jednotlivých akcí** – Akce typu pochodů mohou mít jednu nebo více tras, které je potřeba evidovat v systému. Každá trasa má svůj název a seznam kontrolních stanovišť z nichž jedno je start a jedno cíl, navíc je potřeba znát jejich pořadí v rámci trasy a pro budoucí využití v systému i jejich GPS souřadnice.

3. **Správa uživatelů aplikace** – Uživateli aplikace jsou organizátoři z řad KČT. Je potřeba jim vytvořit uživatelské účty pro zabezpečený přístup do systému. Tyto přístupy spravují administrátoři, kteří mají právo přidávat, mazat a upravovat uživatele.

4. **Evidence služeb v rámci akce** – U každé akce je potřeba evidovat služby pro účastníky, které si mohou vybrat při registraci, přičemž

některé jsou povinné – např. vstupné. Dále si mohou vybrat např. ubytování nebo účast na kulturním vystoupení. U každé takovéto služby je potřeba evidovat její název, cenu a dostupnou kapacitu, některé mohou mít na výběr z více možností.

5. **Registrace účastníka akce** – Uživatelé jsou na akci registrování z externích aplikací (např. webové stránky dané akce), které uživatele registrují pomocí API. Při registraci účastníka je potřeba evidovat jeho osobní údaje, které poslouží k jeho identifikaci při příchodu, doplňkové služby, které si zvolil a případně trasu, kterou chce jít.

6. **Odbavení účastníka při příchodu na akci** – Při příchodu účastníka na akci je účastník odbaven organizátory akce. Při odbavení je potřeba vyhledat příslušnou přihlášku, případně účastníka registrovat. Po vyhledání přihlášky se určí cena za objednané služby a zjistí zda již není přihláška zaplacená.

7. **Evidence průchodu kontrolním stanovištěm** – Je potřeba zaznamenat průchody kontrolním stanovištěm a časy průchodu zanést do systému. Na základě těchto údajů se může určit třeba pořadí, nebo kolik lidí je ještě na trase.

8. **Export dat z akce** – Je potřeba mít možnost exportovat data z akce, která zahrnují jmenný seznam účastníků, který obsahuje co nejvíce informací z přihlášky a údaj, zda vyrazili na trasu (kvůli tisku diplomů).

9. **Přiřazení platby k přihlášce** – Účastníci mohou uhradit poplatek za účast a služby na akci předem a to převodem na účet. Takto provedenou platbu je potřeba manuálně přiřadit ke správné přihlášce v systému.

2.2.2 Nefunkční požadavky

1. **Přijímání dat načtených z QR kódu** – backend musí být schopný přijímat data načtená z QR průkazek klubu KČT nebo karet účastníka.

2. **Možnost nasazení na operačním systému Linux** – Servery KČT běží na OS Linux, proto je potřeba zvolit takovou platformu, která půjde nasadit i na tomto systému.

3. **Bezpečnost aplikace** – je třeba zajistit dobrou bezpečnost za použití prověřených bezpečnostních standardů.

4. **Rozšířitelnost** – aplikace musí být snadno rozšířitelná.

5. **Dostupnost administračního rozhraní přes WWW** – Administrační rozhraní by mělo být přístupné z webového prohlížeče a optimalizované pro zobrazení na mobilních zařízeních.

Kapitola 3

Návrh

V této kapitole je popsán návrh aplikace, který vychází z analýzy (viz. kapitola 2). Je zde specifikován seznam účastníků a předpokládané případy užití systému. Další úlohou je volba vhodných technologii a návrh datových tříd. Na základě návrhu bude možné provést realizaci systému.

3.1 Seznam účastníků

V této sekci jsou rozebrání jednotliví uživatelé a jejich role v systému.

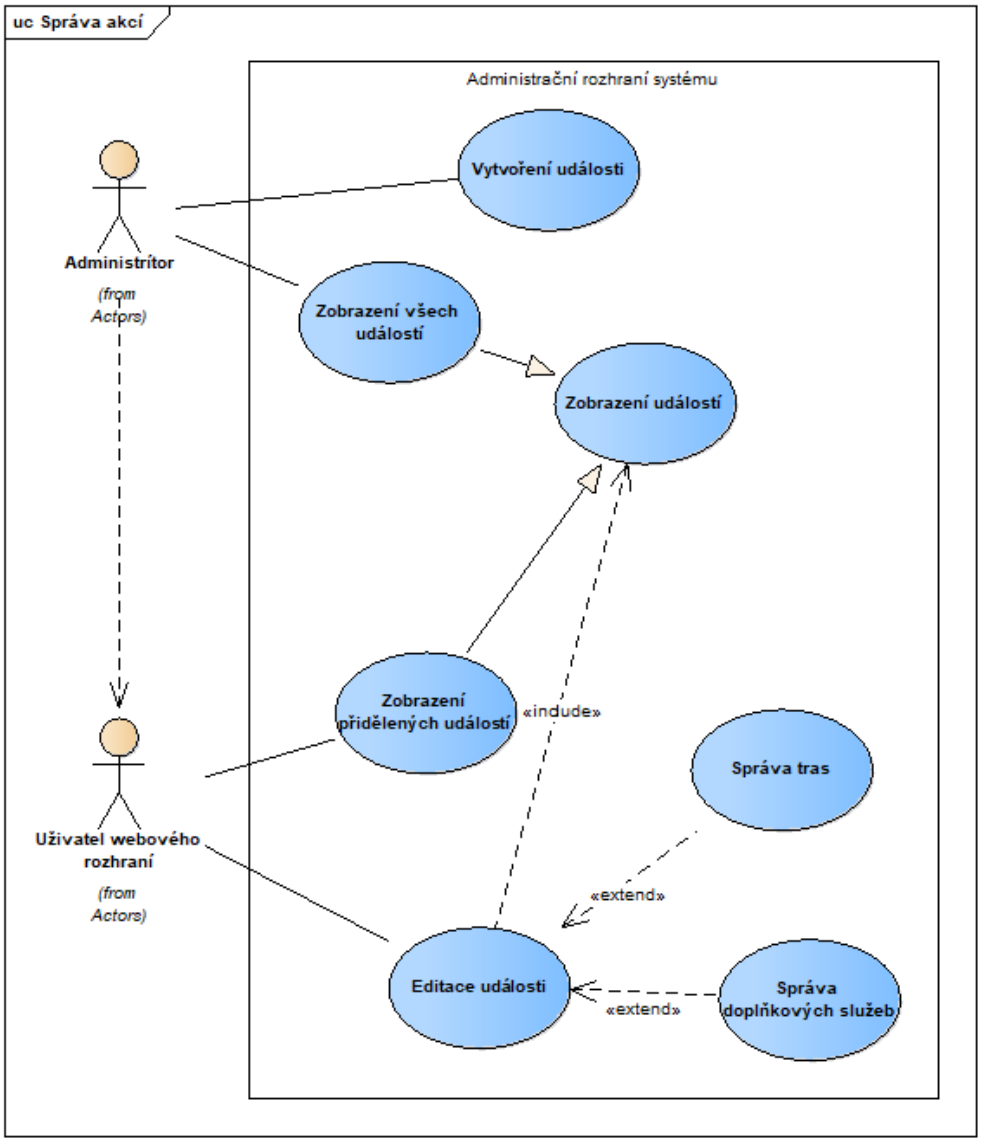
1. **Uživatel webového rozhraní (Web user)** – Uživatel, který má právo využívat webové rozhraní ke správě akcí, ke kterým má přístup.

2. **API Klient (API User)** – Klient, který má právo přistupovat k API a upravovat údaje akcí, ke kterým má povolený přístup. Může registrovat nové účastníky na akce, odbavovat je při příchodu na akci, zaznamenávat průchod účastníků kontrolními stanovišti. Také má právo získat data potřebná k nastavení klientské aplikace (seznam akcí, kontrolních bodů, tras, ...).

3. **Administrátor systému (Admin)** – Má všechna oprávnění, může spravovat uživatele aplikace, vytvářet akce a následně je přidělovat uživatelům systému.

3.2 Případy užití

V této části jsou popsané případy použití (use case) pro účastníky definované v části 3.1. Pro přehlednost byly případy užití rozděleny do tří balíčků, které sjednocují podobné typy činností.



Obrázek 3.1: Model případů užití – Správa akcí

3.2.1 Správa akcí

V tomto balíčku jsou popsány činnosti spojené se zavedením akcí do systému. Model případů užití naleznete na obrázku 3.1.

1. **Vytvoření události** – Vložení nové události do systému s povinnými atributy:
   - **Název** – Název.

- **Datum konání** – Datum začátku a konce akce.
- **Typ** – Informace o tom, jestli se jedná o pochod, organizační akci nebo jiný typ akce.

a vložit následující nepovinné údaje:

- **Popis události**
- **Místo konání**
- **Kontrolní stanoviště** – Stanoviště, kde budou probíhat kontroly účastníků.
- **Trasy** – Jednotlivé trasy závodu, které mohou účastníci absolvovat.
- **Doplňkové služby** – Služby, které je možné zvolit v rámci akce.
- **Správci akce** – Uživatelé (kromě administrátorů), kteří mohou akci spravovat.
- **Web** – Webové stránky dané akce.
- **Pořadatel** – Informace o pořadateli a kontakt na něj.
- **Externí odkazy** – Odkazy na stránky související s akcí např. mapa nebo dopravní spojení. U každého je třeba evidovat URL a popis, který se zobrazí v systému.

2. **Zobrazení seznamu akcí** – Uživateli musí mít možnost zobrazit seznam akcí, jejichž je správcem.

3. **Úprava akce** – Úprava údajů akce, ke které má přihlášený uživatel přístup. Údaje jsou stejné jako při vytváření nové akce. Správci události mohou také editovat seznam správců dané akce.

4. **Správa kontrolních stanovišť** – Ké každé akci je potřeba evidovat kontrolní stanoviště, která mohou být součástí více tras. Je potřeba evidovat následující atributy:

   - **Název**

   A nepovinné údaje:

   - **Popis** – Stručný popis, například informace o tom, kde přesně se stanoviště nachází.
   - **GPS souřadnice** – Souřadnice pro případné zakreslení do mapy.

5. **Správa tras** – Systém musí umožnit spravovat trasy jednotlivých akcí. Akce může mít více tras nebo nemusí mít žádnou. Trasy jsou určeny pořadím stanovišť, kde první je start a poslední je cíl. Každá trasa má následující údaje:

- **Název**
- **Kontrolní body** – Kontrolní body na trase, u každého je potřeba evidovat jeho pořadí v rámci dané trasy.

A nepovinné údaje jsou:

- **Délka trasy**
- **Popis trasy**

6. **Správa služeb akce** – Systém musí umožnit evidenci služeb akce – vstupné, ubytování, stravování, zájezdy, kulturní akce, suvenýry, atd. Existují 2 typy takovýchto služeb – normální a výčtové. U normálních jsou možnosti ano/ne, u výčtových existuje seznam položek, ze kterých je možno vybrat. Některé služby jsou povinné a nelze je neobjednat (např. vstupné). U každé služby se evidují následující atributy:

   - **Název**
   - **Povinná** – Příznak, jestli je daná služba nutná pro účast na akci.
   - **Počitatelná** – Příznak, zda je možno objednat více jednotek.

   A nepovinné atributy:

   - **Popis**
   - **Volby** – Možnosti, ze kterých je možno vybrat v případě, že je služba výčtová. U možností evidujeme:
     - **Název**
     - **Cena**
     - **Kapacita** – Některé volby mohou mít omezenou kapacitu.

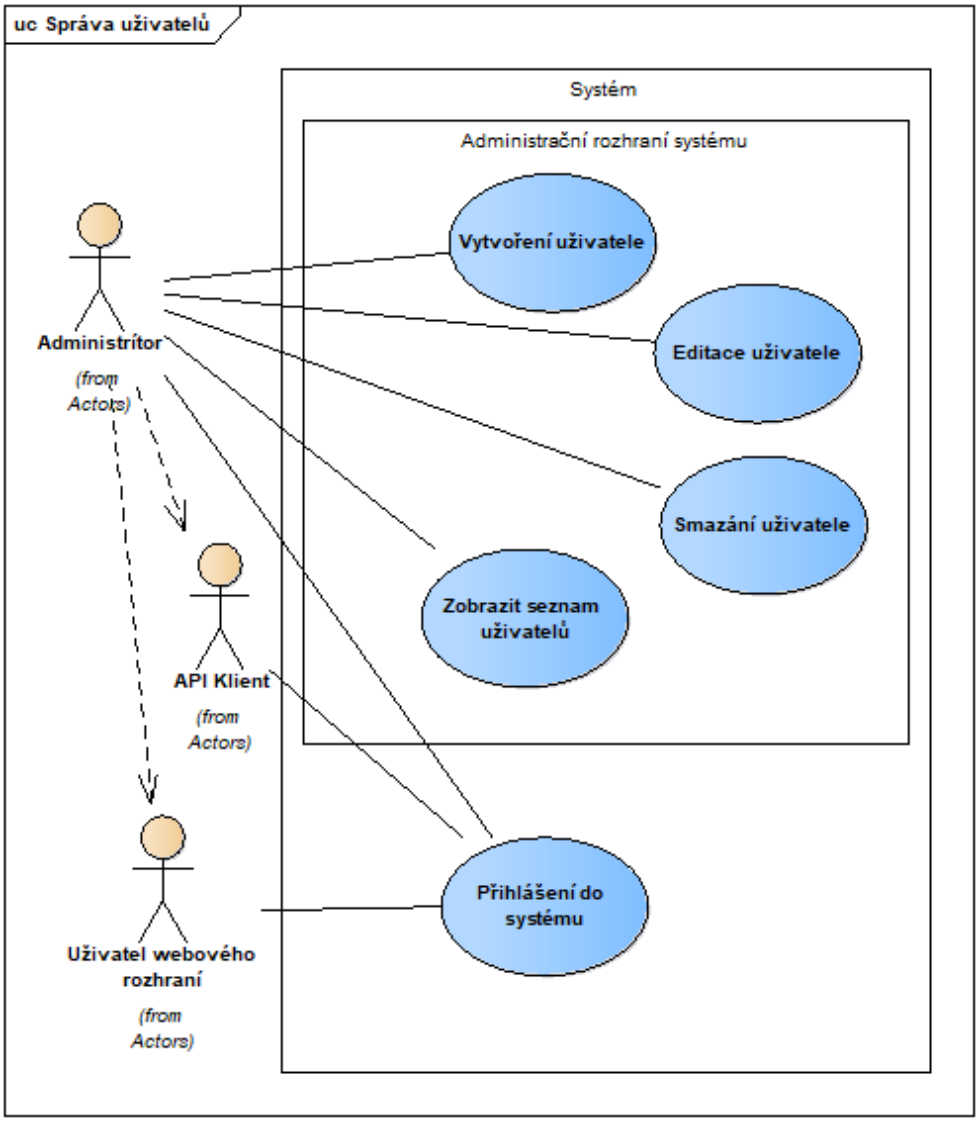
3.2.2 Správa uživatelů a klientů

V tomto balíčku je rozebrána správa uživatelů aplikace. Model případů užití je na obrázku 3.2.

1. **Vytvoření uživatele** – Administrátoři systému potřebují vytvářet další uživatele. Při jejich tvorbě jsou vyžadovány následující údaje:

   - **Uživatelské jméno**
   - **Heslo**
   - **Role** – role, které uživatel v systému zastává a které určují, ke kterým zdrojům bude mít uživatel přístup.

   A nepovinné atributy:



Obrázek 3.2: Model případů užití – Správa uživatelů

- **E-mail** – Může být využit v případě rozšíření systému pro posílání hromadných e-mailů správcům akce.

2. **Přihlášení do systému** – Systém musí umožnit nepřihlášeným osobám přihlášení za pomoci uživatelského jména a hesla.
3. **Editace uživatele** – Systém musí umožnit editaci uživatele, údaje jsou stejné jako při jeho vytváření, ale bez možnosti měnit uživatelské jméno.
4. **Smazání uživatele** – Administrátoři mohou mazat uživatele systému.
5. **Zobrazit seznam uživatelů** – Systém musí umožnit zobrazení seznamu uživatelů administrátorům.

3.2.3 Evidence účastníků akcí

V tomto balíčku jsou rozepsané akce, které jsou požadované v souvislosti s registrací účastníků akcí a správou takto vzniklých přihlášek. Model případů užití naleznete na obrázku 3.3.

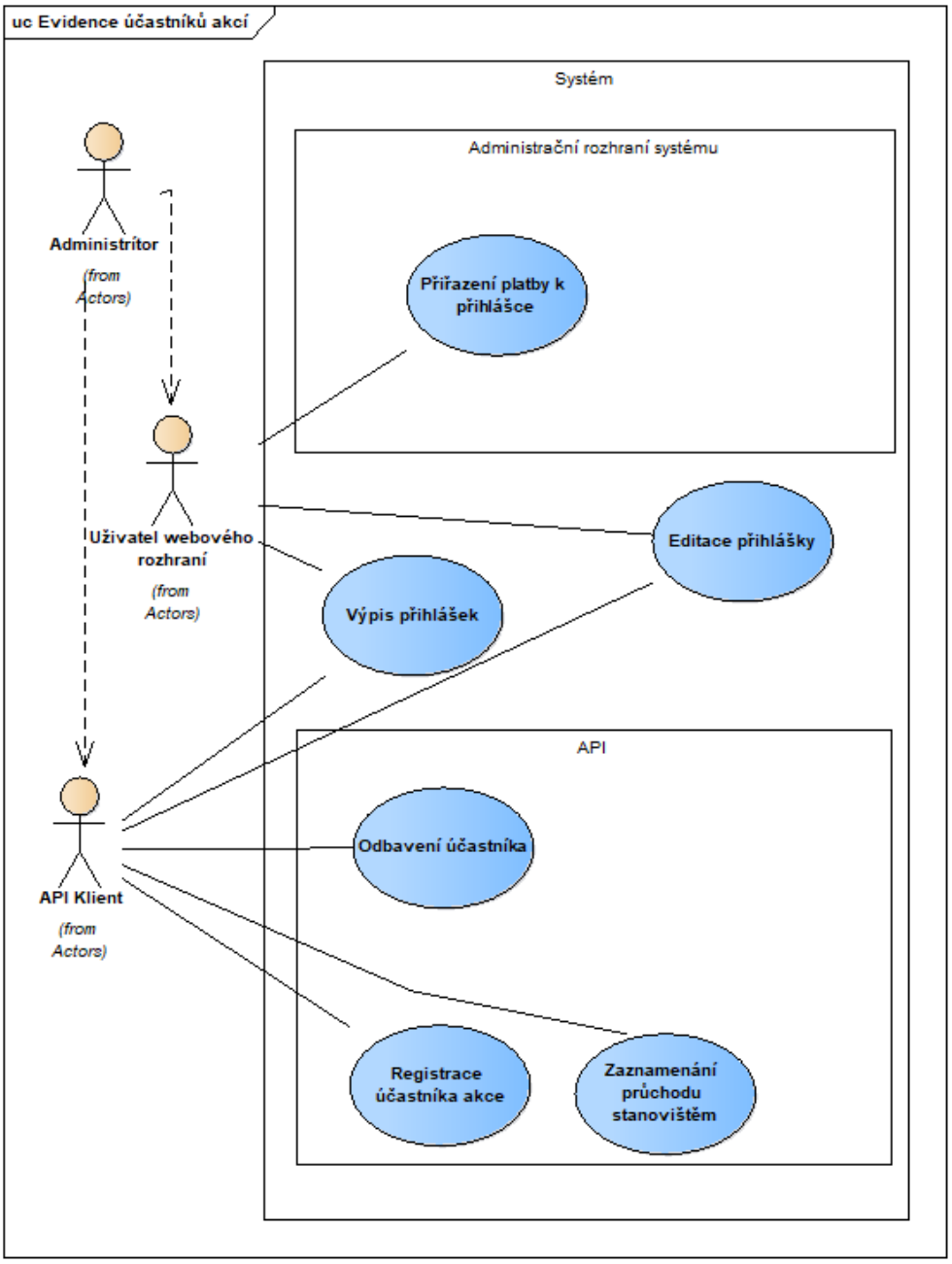
1. **Registrace účastníka akce** – Systém musí umožnit registrovat osoby na akce. Uživatelé budou registrování pomocí klientských aplikací, např. webové formuláře nebo plnohodnotné aplikace (registrace v místě konání). Každý takto registrovaný uživatel obdrží unikátní identifikační číslo platné v rámci dané akce. Při registraci jsou povinné následující údaje:

   - **Jméno a příjmení**
   - **Datum registrace**
   - **Trasa** – Je nutné zvolit trasu (pokud daná akce trasy obsahuje).
   - **Zvolené služby** – Doplňkové služby, které si registrovaný vybral. U každé se eviduje počet jednotek, případně varianta služby.

   A volitelné údaje:

   - **Číslo průkazu** – Číslo průkazu registrovaného (případně jiné unikátní číslo v rámci dané akce)
   - **E-mail** – Unikátní identifikátor.

2. **Odbavení účastníka** – Systém musí umožnit odbavení účastníka při příchodu na akci. Klient předá id akce a identifikátor příchozího, čas registrace a případně částku, kterou uhradil za objednané služby.
3. **Zaznamenání průchodu stanovištěm** – Systém musí umožnit zaznamenání průchodu stanovištěm po zaslání následujících údajů, podle kterých lze účastníka v systému jednoznačně identifikovat.



Obrázek 3.3: Model případů užití – Evidence účastníků akcí

4. **Získání dat o průchodu stanovišti** – Aplikace by měla umožnit zobrazení přehledu průchodů kontrolními stanovišti (aktuální pořadí).

5. **Výpis přihlášek** – Systém musí umožnit získání seznamu přihlášek pro jednotlivé akce.

6. **Editace přihlášky** – Je potřeba editovat přihlášku po jejím odeslání – změna objednaných služeb

7. **Přiřazení platby k přihlášce** – Je potřeba přiřadit platbu převodem k přihlášce. Jako variabilní symbol pro platbu může být použito přidělené id v rámci akce. Konstantním symbolem je id akce.

3.2.4 Export dat z aplikace

V tomto balíčku je rozebrán export dat z aplikace.

1. **Export jmenného seznamu účastníků** – Aplikace by měla umožnit export všech přihlášek na danou akci ve formátu Microsoft Excel (.xlsx).

2. **Export přihlášky** – aplikace by měla umět exportovat detaily přihlášky ve formátu PDF. V takovýchto detailech by měly být dostupné všechny informace jako v prvním bodě z 3.2.3 a navíc data o průchodu kontrolními stanovišti.

3.3 Volba technologii

Z požadavků vyplývá, že aplikace by měla běžet jako webová služba s administračním rozhraním dostupným přes webový prohlížeč. Klub Českých Turistů má k dispozici server s OS Linux, proto je možné využít stávající infrastrukturu. V současné době je nejpoužívanějším jazykem na serverové straně webových aplikací PHP, druhý nejrozšířenější je .NET[4].

Pro tvorbu serverové části jsem zvažoval právě použití PHP v kombinaci s nějakým hojně používaným frameworkem, např. Nette[5] nebo Symfony2[6] kvůli jejich dobré podpoře, nakonec jsem se rozhodl PHP nepoužít, protože s ním nemám žádné zkušenosti. Oproti tomu s platformou .NET mám dostatek vesměs pozitivních zkušeností a jsem se pro využití platformy C#/.NET. Podle požadavků má systém běžet na OS Linux a klasický .NET Framework nemá podporu pro tento systém. Proto jsem se odhodlal vyzkoušet novou technologii .NET Core, která má navazovat na klasický .NET Framework, ale má podporovat více platforem, včetně Linuxu. [7]. Další jeho výhodou je licence – open-source. Hlavní nevýhodou je určitě to, že v době psaní této práce se jedná o novinku a hrozí, že ještě nebude úplně odladěný.

3.3.1 Architektura uživatelského rozhraní

V obyčejném ASP.NET Frameworku jsou k dispozici dva hlavní způsoby tvorby webových aplikací. Jsou to starší WebForms a modernější MVC. Použití multiplatformního .NET Core značně usnadňuje výběr, protože podporuje pouze modernější z těchto způsobů – MVC. To rozděluje aplikaci na tři hlavní komponenty – model, view a controller. Hlavní motivací stojící za tímto rozdělením je oddělení aplikační logiky od výstupu [8]. Architektura je tvořena třemi komponentami:

- **Model** – Obsahuje logiku aplikace a vše související, jako jsou výpočty, komunikace s databází a validace dat. Jeho hlavní funkcí je vydání dat po přijetí parametrů (neví, odkud data v parametrech přišla, ani jak budou ve výsledku formátována). [8]
- **View** – Stará se o zobrazení výstupu a jeho správné formátování. [8]
- **Controller** – Propojuje ostatní komponenty (view, model) a komunikuje s uživatelem. Jedná se tedy o prostředníka, který drží celý systém pohromadě. [8]

3.3.2 Architektura API

Volba architektury pro API je jednoduchá, protože ASP.NET Core má integrovanou podporu pro architekturu REST a bylo by zbytečně komplikované jít jinou cestou. REST byl představen v roce 2000 v disertační práci Roye Fieldinga, který je jedním ze spoluautorů protokolu HTTP. Proto jsou oba protokoly hodně podobné. Umožňuje jednotný a snadný přístup ke zdrojům, z nichž každý má vlastní identifikátor URI. Rest implementuje čtyři metody přístupu ke zdrojům, také známé jako CRUD, které se mapují na HTTP metody. [9]

- **Create (POST)** – Vytváření nového zdroje, případně podřízeného zdroje. V případě úspěšného vytvoření je návratová hodnota HTTP 200 nebo 201 a případně URI nově vzniklého zdroje v hlavičce.[10]
- **Read (GET)** – Základní požadavek na zdroj. Při jeho zavolání nedochází ke žádný změnám na straně serveru. V případě nalezení požadovaného zdroje vrací HTTP 200 a jeho reprezentaci ve formátu JSON nebo XML. [10]
- **Update (PUT)** – Slouží k aktualizaci zdroje. V těle PUT dotazu na příslušnou URI je zaslána nově upravená reprezentace původního zdroje. Při úspěchu operace vrací HTTP 200 nebo 204 a případně reprezentaci upraveného zdroje. [10]

- **Delete (DELETE)** – Slouží ke smazaní zdroje na dané URI. Po úspěšném smazání vrací HTTP 200 s reprezentací smazaného zdroje v těle odpovědi nebo HTTP 204 s prázdným tělem. [10]

3.3.3 Databáze

Pro usnadnění a urychlení vývoje aplikace je vhodné použít relační databázi s nějakým ORM frameworkem. ORM redukuje množství kódu nutného pro přístup k datům, který musí vývojář napsat. Rozhodl jsem se pro použití Microsoft Entity Frameworku[11], protože je to v současné době nejpoužívanější objektově relační mapper pro .NET Core, má silnou uživatelskou základnu a data získává přes ADO.NET, který podporuje naprostou většinou dnešních databází. Existují tři přístupy tvorby výsledných objektů – code-first, model-first a database-first. Pro aplikaci jsem zvolil přístup code-first, abych se zbavil nutnosti znát cílovou databázi nebo vizuální editace modelu a program byl konzistentní, protože IdentityFramework4 ze sekce 3.3.4.4 využívá právě tento přístup. [12]

Vzhledem k použití code-first přístupu příliš nezáleží na zvoleném databázovém serveru, struktura databáze se vytvoří za pomoci migračních nástrojů. Rozhodoval jsem se mezi MariaDB [13] a PostgreSQL [14], kvůli tomu, že jsou jednoduché na ovládání a open-source. Nakonec jsem na doporučení vedoucího práce zvolil PostgreSQL v kombinaci s odpovídajícím ovladačem pro ADO.NET – NpgSQL. [15]

3.3.4 Zabezpečení

Pro zabezpečení webového API je vhodné použít bezstavové ověřování, které zlepší škálovatelnost aplikace [16], která je součástí nefunkčních požadavků na systém ze sekce 2.2.2. Na doporučení mého vedoucího jsem se rozhodl využít protokol OAuth2 [17]. Webové administrační rozhraní využije stejné uživatele, ale bude využívat protokol OpenID Connect [18] založený na OAuth2, který se vedle ověřování uživatelů spravuje i jejich identitu.

3.3.4.1 JWT

Token sloužící pro přenos, který uchovává ve formátu JSON. Je plně samostatný (obsahuje všechny potřebné informace), funguje napříč různými programovacími jazyky a snadno se přenáší (např. v hlavičce HTTP požadavku). Token je zakódovaný jako Base64 a skládá se ze tří částí: [19]

1. **Hlavička** – Obsahuje informaci o typu objektu (JWT) a použitý hashovací algoritmus. [19]
2. **Data** – Obsahuje informace o uživateli, emitentovi, čas vydání a expirace. [19]

3. **Podpis** – Obsahuje digitální podpis, slouží k ověření pravosti. [19]

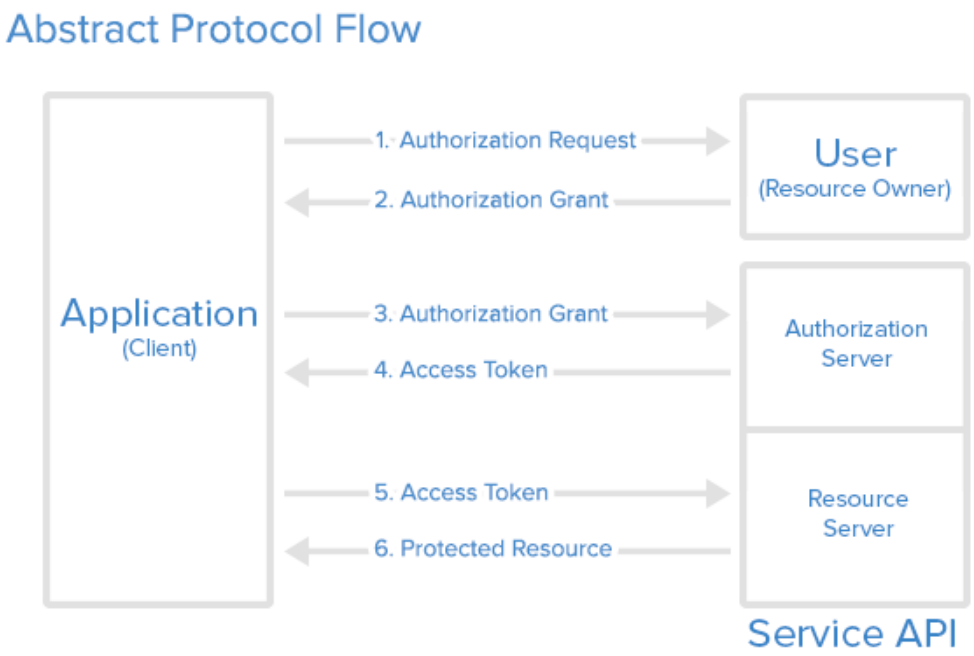
3.3.4.2 OAuth2

OAuth 2 je framework sloužící k ověřování uživatelů, který umožňuje klientským aplikacím omezený přístup k uživatelským účtům na serveru. Funguje tak, že ověření uživatele přenechá službě, která uživatelské účty spravuje a pouze autorizuje ostatní aplikace, které získávají (omezený) přístup k těmto uživatelským účtům.[2]. V rámci tohoto protokolu existují 4 role:

- **Vlastník zdroje** – Samotný uživatel, který autorizuje aplikaci k (omezenému) použití jeho účtu.
- **Server se zdrojem dat** – Služba, která spravuje samotný zdroj.
- **Autorizační server** – Ověřuje identitu uživatele. Vydává přístupové tokeny.
- **Klient** – Aplikace, která vyžaduje přístup ke zdroji vlastněného uživatelem. Každý klient je registrovaný na autorizačním serveru, kde jsou vedle jména klienta uložené také jeho přihlašovací údaje (id a secret) a povolené adresy pro přesměrování (redirect url). [2]

Pro různé případy užití existuje několik typů grantů, pomocí kterých si může aplikace vyžádat autorizaci – viz. obrázek 3.4 . Každý je vhodný pro použití v různých případech:

- **Autorizační kód** – Nejvíce používaný u serverových aplikací, kde je možné zajistit bezpečné uložení klientových přihlašovacích údajů (client secret). Nutností je možnost interakce koncového uživatele, který je přesměrován na přihlašovací stránku autorizačního serveru, kde se přihlásí a případně klientovi povolí přístup ke zdrojům. Klient pak obdrží autorizační kód, pomocí kterého může získat přístupový token. [2]
- **Implicitní** – používaný u mobilních a desktopových aplikací, kde nemůže být zajištěno bezpečné uložení tajemství (secret). Uživatel je opět přesměrován na přihlašovací stránku autorizační služby, ale uživatel rovnou obdrží token, který je předán aplikaci za pomoci uživatelova prohlížeče (User-agent). [2]
- **Přístupové údaje vlastníka zdroje** – Uživatel zadá přihlašovací údaje přímo v klientovi, který pomocí nich získá přístupový token od autorizační služby. [2]
- **Přístupové údaje klienta** – Nejsou vůbec vyžadovány přihlašovací údaje uživatele. Přihlášení proběhne pouze pomocí přihlašovacích údajů klienta (id a secret). Na základě těchto obdrží přístupový token k vlastnímu servisnímu účtu. [2]



Obrázek 3.4: OAuth2 – Abstract protocol flow [2]

3.3.4.3 OpenID Connect

OpenID Connect je standard z roku 2014 pro poskytování identity v rámci internetu. Využívá jednoduché JWT tokeny, které jsou zprostředkovány pomocí protokolu OAuth 2.0. Známým případem poskytovatele, který OpenId Connect využívá je služba MojeId. [20] Token, který klient obdrží obsahuje následující údaje:

- Identitu uživatele
- Autoritu, která token vydala
- Cílovou audienci
- Datum vydání
- Datum expirace
- Dodatečné informace o uživateli (e-mail, telefon, . . .)[21]

3.3.4.4 Volba poskytovatele identity

Při volbě poskytovatele identity jsem se rozhodoval mezi dvěma možnostmi. První bylo využít službu třetí strany, která se o správu uživatelů postará.

Bohužel jsem nenašel žádnou bezplatnou a proto jsem se nakonec rozhodl pro vytvoření vlastního poskytovatele identity. K tomuto účelu se výborně hodí open-source knihovna IdentityServer4. Ten zajišťuje server pro identifikaci uživatelů pro externí systémy a podporuje protokoly OAuth2 i OpenID Connect. [22] Další výhodou je možnost jeho napojení na systém správy uživatelů ASP.NET Core Identity, který zajistí bezpečné přihlašování, uložení přihlašovacích údajů do databáze a umožňuje použít systém rolí, pomocí kterého lze snadno namapovat účastníky ze seznamu účastníků (dostupný v sekci 3.1 na jednotlivé role. [23]

3.3.5 Volba frontend frameworku

Z analýzy nefunkčních požadavků 2.2.2 vzniká potřeba optimalizace výsledného administračního rozhraní ve formě webové stránky i pro mobilní zařízení. Vzhledem k tomu, že nemám žádné zkušenosti s tvorbou webových stránek, tak jsem se rozhodl pro využití jednoduchého HTML, CSS a JavaScript frameworku Bootstrap ve verzi 3, který umožní rozdělení stránky do mřížky a postará se o správné zobrazení na mobilních zařízeních a moderní vzhled systému. [24]

3.4 Návrh datových tříd

Pro uložení dat bude použita relační databáze, proto je vhodné využít datové třídy, které budou mapovat data na jednotlivé tabulky v databázi. Tato část je věnována jejich návrhu. Pro mapování tříd do databáze je použito ORM EntityFramework[11]. Pro návrh tříd jsem zvolil metodu code-first – první vytvořím třídy v jazyce C#, ze kterých je potom automaticky vytvořena databáze. IdentityServer4 napojený na .NET Core Identity si generuje vlastní databázi, která bude nezávislá na systému pro správu akcí, proto se jí v této sekci nebudu věnovat.

Pro všechny textové atributy jsem zvolil možnost nespecifikovat jejich délku, takže se ve výsledné databázi namapují na sloupec typu Text, který má dostatečnou kapacitu pro všechny naše potřeby a je celkově dobře optimalizovaný – práce s ním je přinejmenším stejně rychlá jako s ostatními datovými typy s omezenou délkou(varchar, char). [25] Dále je vhodné specifikovat cizí klíče, nejen reference. Ty by se sice vygenerovaly automaticky, ale je dobré mít přístup i k hodnotám těchto klíčů. Názvy výsledných tabulek odpovídají názvům tříd, ale jsou pluralizované (Event -> Events).

3.4.1 Event

Třída reprezentující jednotlivé události obsahuje následující atributy:

- **EventId(int)** – Primární klíč generovaný databází.
- **Name(string)** – Název události.
- **Start(DateTime)** – Začátek.
- **End(DateTime)** – Konec.
- **Type(EventType)** – Typ události. Jedná se o výčtový datový typ – enum namapovaný na byte. Nabývá hodnot Walk (pochod) = 0, Organisational (organizační) = 1, Other (jiný) = 2.
- **Description(string)** – Popis. Nepovinný atribut.
- **Venue(string)** – Místo konání. Nepovinný atribut.
- **Webpage** – Webová stránka události – systém se stará centralizovanou správu přihlášek, každá událost může mít vlastní webovou stránku. Nepovinný atribut.
- **ContactName(string)** – Jméno a příjmení pořadatele. Nepovinný atribut.
- **ContactEmail(string)** – E-mail pořadatele. Nepovinný atribut.
- **ContactPhone(string)** – Telefon na pořadatele. Nepovinný atribut.
- **ContactOther** – Doplňující info o pořadateli. Například skype, ICQ... Nepovinný atribut.
- **Checkpoints(ICollection<Checkpoint>)** – Kolekce všech kontrolních stanovišť dané akce.
- **Routes(ICollection<Route>)** – Kolekce všech tras dané akce.
- **Services(ICollection<Service>)** – Kolekce všech doplňkových služeb.
- **Administrators(ICollection<EventAdministrator>)** – Kolekce všech administrátorů dané akce.
- **ExternalLinks(ICollection<ExternalLink>)** – Kolekce externích odkazů akce
- **Registrations(ICollection<Registration>)** – Kolekce všech registrací

3.4.2 EventAdministrator

Uchovává informace o uživateli, který může přistupovat k akci. Uživatelé jsou evidování v jiné databázi a náš systém už jen dostane JWT token, který mimo jiné obsahuje Id subjektu, které používáme pro identifikaci jednotlivých uživatelů. Proto třída obsahuje následující parametry:

- **EventId (int)** – Součást primárního klíče a cizí klíč pro tabulku Events.
- **Event (Event)** – Reference na třídu Event.
- **UserId (string)** – Součást primárního klíče, Id uživatele, který má k akci přístup.

3.4.3 ExternalLink

Uchovává externí odkazy pro jednotlivé akce.

- **ExternalLinkId (int)** – Primární klíč generovaný databází.
- **EventId (int)** – Cizí klíč pro tabulku Events.
- **Event (Event)** – Reference na třídu Event.
- **Description (string)** – Popisek odkazu.
- **URL (string)** – Url na kterou odkazuje.

3.4.4 Checkpoint

Třída reprezentující kontrolní stanoviště a potřebné údaje o něm.

- **CheckpointId (int)** – Primární klíč generovaný databází.
- **EventId (int)** – Cizí klíč pro tabulku Events.
- **Event (Event)** – Reference na třídu Event.
- **Name (string)** – Název stanoviště.
- **Description (string)** – Popis stanoviště. Nepovinný atribut.
- **Latitude (double?)** – Zeměpisná šířka. Nepovinný atribut. Předpokládá se použití dekadického formátu v intervalu $[-90, 90]$.
- **Longitude (double?)** – Zeměpisná délka. Nepovinný atribut. Předpokládá se použití dekadického formátu v intervalu $[-180, 180]$.

3.4.5 CheckpointPass

Třída reprezentující průchod kontrolním stanovištěm. Vedle referencí na provázané třídy obsahuje ještě údaj o času průchodu. Primární klíč tvoří dvojice cizích klíčů – registrace, kontrolní stanoviště. Vedle atributů vyplývajících z analýzy případů užití 3.2.3 jsem se rozhodl přidat navíc atributy sloužící ke snadnému vyhledání konkrétní přihlášky – datum narození, telefon a adresa.

- **RegistrationId (int)** – Součást primárního klíče, součást cizího klíče do tabulky. Registrations.
- **Registration (Registration)** – Reference na registraci.
- **EventId (int)** – Součást primárního klíče, součást cizího klíče do tabulky Registrace.
- **Event (Event)** – Reference na událost.
- **CheckpointId (int)** – Součást primárního klíče, cizí klíč do tabulky Checkpoints.
- **Checkpoint (Checkpoint)** – Reference na kontrolní stanoviště.
- **Time (DateTime)** – Čas průchodu kontrolním stanovištěm.

3.4.6 Registration

Třída reprezentující registraci. Primární klíč tvoří id události a id registrace – to je generované automaticky od 1 pro každou událost tak, aby byla čísla pokud možno co nejmenší a dobře se zadávala do systému (např. při odbavování osoby bez členského průkazu).

- **EventId (int)** – Součást primárního klíče, cizí klíč do tabulky Events.
- **Event (Event)** – Reference na událost.
- **RegistrationId (int)** – Součást primárního klíče. Id registrace, generuje se od 1 pro každou událost při jejím vytvoření. Může tak sloužit např. jako identifikátor účastníka, který lze manuálně zadat např. do mobilního telefonu, nebo jako startovní číslo.
- **Name (string)** – Jméno registrovaného.
- **Surname (string)** – Příjmení registrovaného.
- **RegistrationDate (DateTime)** – Datum a čas registrace. Index.
- **RouteId (int?)** – Id trasy, cizí klíč do tabulky Routes. Nepovinný atribut.

- **Route (Route)** – Reference na objekt trasy.
- **ItentificationNumber (string)** – Identifikační číslo účastníka. Unikátní v rámci akce – může to být číslo členské průkazky, nebo např. číslo načtené z QR kódu (pokud účastník nebude vlastníkem průkazky). Nepovinný atribut. Index.
- **Email (string)** – E-mail účastníka akce. Unikátní v rámci akce. Nepovinný atribut. Index.
- **Phone (string)** – Telefonní číslo účastníka. Unikátní v rámci akce. Nepovinný atribut. Index.
- **Address (string)** – Adresa účastníka. Nepovinný atribut.
- **BirthDate (DateTime?)** – Datum narození. Nepovinný atribut.
- **AmountPaid (decimal)** – Suma, kterou již účastník zaplatil za objednané služby.
- **CheckInDate (DateTime?)** – Datum a čas odbavení účastníka. NULL v případě, že ještě nebyl odbaven.
- **SelectedServices (ICollection<SelectedService>)** – Kolekce zvolených služeb.
- **CheckpointPasses (ICollection<CheckpointPass>)** – Kolekce průchodů kontrolními stanovišti.

3.4.7 Route

Třída reprezentující trasu akce.

- **RouteId (int)** – primární klíč generovaný databází.
- **Name (string)** – Název trasy.
- **Description (string)** – Popis. Nepovinný atribut.
- **Length (double?)** – Přibližná délka trasy v km. Na základě tohoto atributu se může počítat průměrná rychlost účastníka. Nepovinný atribut.
- **EventId (int)** – Cizí klíč do tabulky Events.
- **Event (Event)** – Reference na akci.
- **RouteCheckpoints (ICollection<RouteCheckpoint>)** – Kolekce kontrolních stanovišť na trase.

3.4.8 RouteCheckpoint

Třída reprezentující kontrolní stanoviště na trase a jeho pořadí v rámci trasy. Jedním stanovištěm může procházet více tras.

- **RouteId (int?)** – Součást primárního klíče, cizí klíč do tabulky Routes.
- **Route (Route)** – Reference na trasu.
- **Order (int)** – Pořadí kontrolního stanoviště na trase. Stanoviště s nejnižším pořadím je považováno za start, s nejvyšším za cíl.
- **CheckpointId (int)** – Součást primárního klíče, cizí klíč do tabulky Checkpoints.

3.4.9 SelectedService

Třída reprezentující uživatelem zvolenou službu a její množství. Vytváří se smyčka v databázi, protože je potřeba omezit, aby zvolená služba nemohla být v přihlášce vybrána vícekrát v různých variantách (např. zároveň vybráno dětské a normální vstupné).

- **EventId (int)** – Součást primárního klíče, část cizího klíče do tabulky Registrations.
- **Event (Event)** – Reference na akci.
- **RegistrationId (int)** – Součást primárního klíče, část cizího klíče do tabulky Registrations.
- **Registration (Registration)** – Reference na registraci.
- **ServiceId (int)** – Součást primárního klíče, cizí klíč do tabulky Services.
- **Service (Service)** – Reference na službu.
- **Count (int)** – Objednaný počet. 1 v případě, že se jedná o službu bez možnosti volby množství.

3.4.10 Service

Třída reprezentující službu, kterou si může uživatel zvolit v rámci přihlášky na akci.

- **ServiceId (int)** – Primární klíč generovaný databází.
- **EventId (int)** – Cizí klíč do tabulky Events.
- **Event (Event)** – Reference na službu.

- **Name (string)** – Název služby.
- **Description (string)** – Popis služby. Nepovinný atribut.
- **Required (bool)** – Příznak, zda je služba nezbytně nutná k úspěšnému přijetí přihlášky (např. vstupné na akci).
- **Countable (bool)** – Příznak, zda je možné v přihlášce vybrat více kusů služby (např. pamětní předměty).
- **ServiceOptions (ICollection<ServiceOption>)** – Kolekce všech možností, ze kterých je možné vybírat. Pokud neobsahuje alespoň 1, tak je služba neaktivní.
- **SelectedServices (ICollection<SelectedService>)** – Kolekce všech přihlášek, ve kterých byla daná služba vybrána.

3.4.11 ServiceOption

Třída reprezentující volbu u služby. Pokud služba nemá žádnou možnou variantu, potom je neaktivní.

- **ServiceOptionId (int)** – Primární klíč automaticky generovaný databází.
- **ServiceId (int)** – Cizí klíč do tabulky Services.
- **Service (Service)** – Reference na službu.
- **Name (string)** – Název. Nepovinný atribut pro případ, že služba má pouze jednu možnost.
- **Price (decimal)** – Cena volby, kde 0 znamená, že služba je zdarma.
- **Capacity (int?)** – Kapacita možnosti – kolik jednotek systém umožní registrovat. V kombinaci s povinným atributem lze omezit kapacitu celé akce.
- **SelectedServices (ICollection<SelectedService>)** – Kolekce všech přihlášek, které mají zvolenou tuto možnost.

Kapitola 4

Realizace

Tato kapitola se zabývá realizací výsledného systému na základě návrhu. Jsou zde rozebrané použité vývojářské nástroje, výsledná databáze a diagram nasazení. Protože se však jedná o backendové řešení, tak největší důraz je kladen na API Referenci, která poslouží jako základ pro tvorbu navazujících aplikací.

4.1 Volba nástrojů

Před samotným začátkem vývoje je vhodné zvolit vývojové prostředí a další podpůrné nástroje pro samotný vývoj, jako jsou verzovací systém nebo testovací nástroje.

4.1.1 Vývojové prostředí (IDE)

Volba vývojového prostředí pro .NET Core je poměrně přímočará. Jasnou volbou je Visual Studio 2017 Community Edition [26], protože zatím neexistuje jiná alternativa pro vývoj na takto nové platformě. Konkurenční IDE Rider od JetBrains [27] je v době psaní této práce stále ještě ve vývoji a navíc se mi špatně pracuje ve vývojových prostředích od JetBrains. Visual studio obsahuje kompletní sadu vývojářských nástrojů, dokáže se propojit s Google Chrome a debugovat JavaScript, podporuje snadné stahování knihoven třetích stran pomocí systémů NuGet a Bower.

Hlavní výhodou Visual Studia je jeho přívětivost k novým uživatelům – všechno je již přednastavené a funguje tak, jak se očekává – tedy ve většině případů. Jediný problém, na který jsem za celou dobu vývoje narazil bylo vytváření nových souborů typu .cshtml (pohledů). Ty postrádaly po vytvoření našeptávání a bylo potřeba je otevřít znovu. Takto otevřené soubory ztratily kódování UTF-8 a špatně vykreslovaly české znaky, proto bylo potřeba změnit jejich kódování v jiném programu.

4.1.2 Verzovací systém

Jako verzovací systém byl díky dobrým předchozím zkušenostem zvolen open-source software git. Jako hostitele git repositáře jsem se rozhodl využít Bitbucket, který umožňuje vytvářet git repozitáře s přístupem až pro 5 uživatelů zdarma, což je pro mě jako jedince plně dostačující. [28] Navíc grafický git klient, na kterého jsem zvyklý – Sourcetree podporuje integraci Bitbucket účtu. [29]

4.1.3 Testovací nástroje

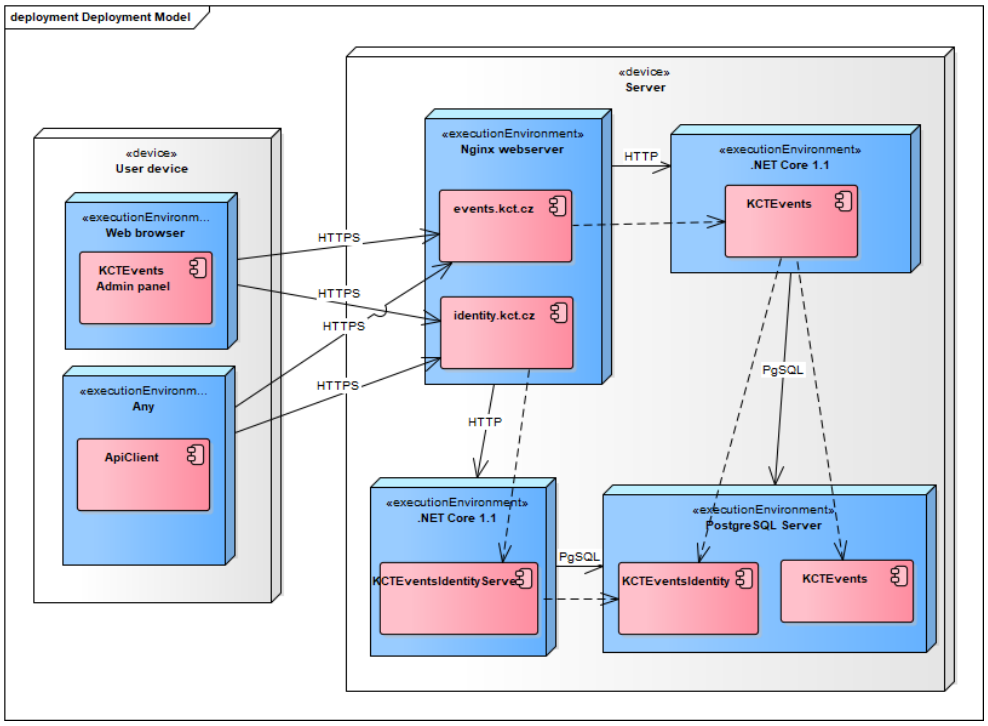
Pro testování REST API existuje spousta nástrojů, já jsem zvolil nástroj Postman, protože je jednoduchý na použití, podporuje ověřování pomocí OAuth2 v režimu autorizačního kódu a snadné vkládání získaného tokenu do hlavičky požadavku. Také podporuje psaní automatických testů a běží na všech hlavních desktopových platformách. [30]

4.2 Implementace

Systém byl implementován na základě návrhu ze sekce 3. Systém je rozdělený na dva projekty, protože je vhodné oddělit server, který se stará o správu přístupů a vydávání tokenů od zbytku aplikace. Samotné API společně s administračním rozhraním jsou v projektu druhém. Každý projekt používá vlastní databázi. Vzhledem k tomu, že se jedná o novou technologii a integrovaný webserver (Kestrel) nemá tak prověřenou bezpečnost, tak je vhodné použít reverse proxy server například Nginx, který zajistí šifrování spojení a dodatečnou ochranu před DoS/DDoS útoky. Výsledný model nasazení je na obrázku 4.1.

4.2.1 KCTEventsIdentityServer

Autorizační server je naprogramován s využitím knihovny IdentityServer4 [22], který je napojený na systém pro správu uživatelů .NET Core Identity. Obě tyto knihovny si samy vytvoří databázi pomocí migrací ORM knihovny EntityFramework [11] a jsou schopné pracovat na jedné databázi. Takto vzniklá databáze má 25 tabulek a v případě potřeby je uživatelsky rozšiřitelná. Bylo potřeba napsat administrační rozhraní pro základní správu povolených klientských aplikací a uživatelů. Také jsou zde k dispozici základní přihlašovací formulář (na obrázku 4.3) a stránka, která se uživatele ptá na udělení přístupu konkrétnímu klientovi. Tato implementace podporuje všechny typy grantů popsané v sekci 3.3.4.2 , takže je na vývojářích navazujících aplikací, jaký typ si zvolí.



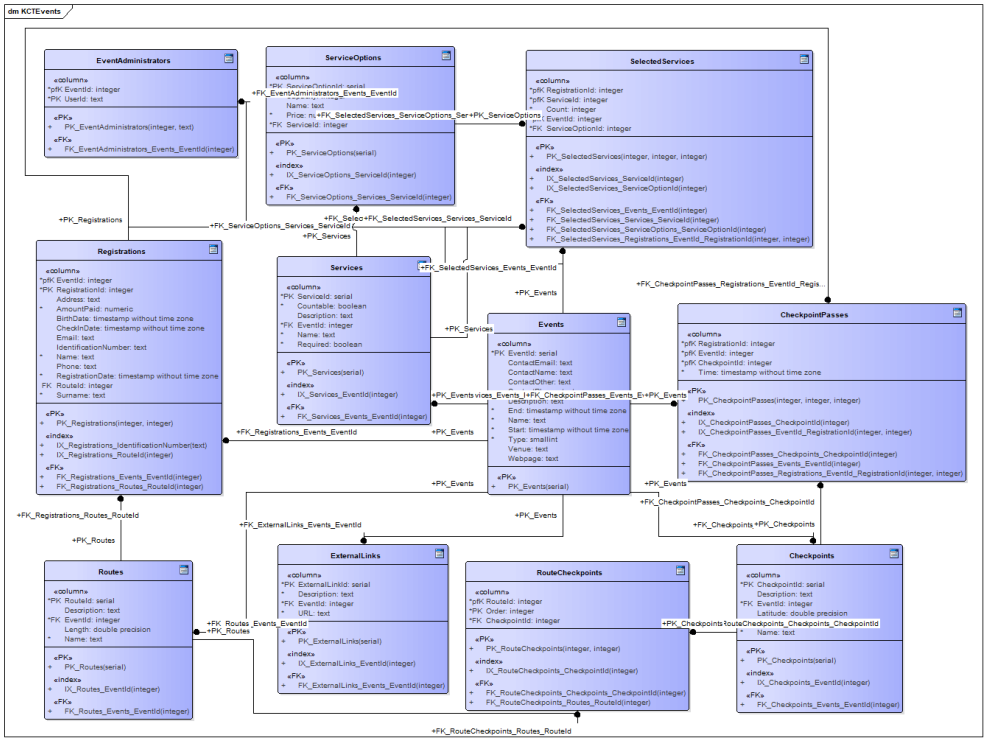
Obrázek 4.1: Model nasazení systému

4.2.2 KCTEvents

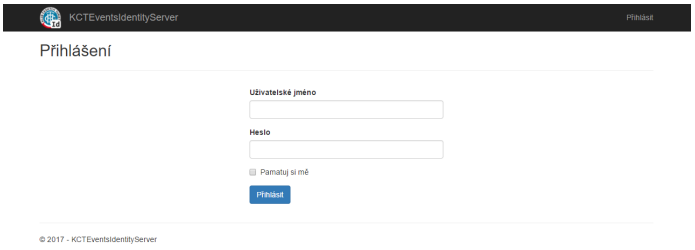
Jedná se o klasickou .NET MVC aplikaci s API, která podporuje více typů ověřování uživatelů – Jednak pomocí OpenId Connect v kombinaci s cookies pro uživatele, kteří se připojují z prohlížeče. Uživatelé přistupující k API jsou ověřování pomocí OAuth2. Ukázka vzhledu administračního rozhraní výsledné aplikace je k dispozici na obrázku 4.4 a kompletní popis všech funkcí tohoto API naleznete v sekci 4.3. Schéma výsledné databáze pro tuto aplikaci je na obrázku 4.2.

4.3 API Reference

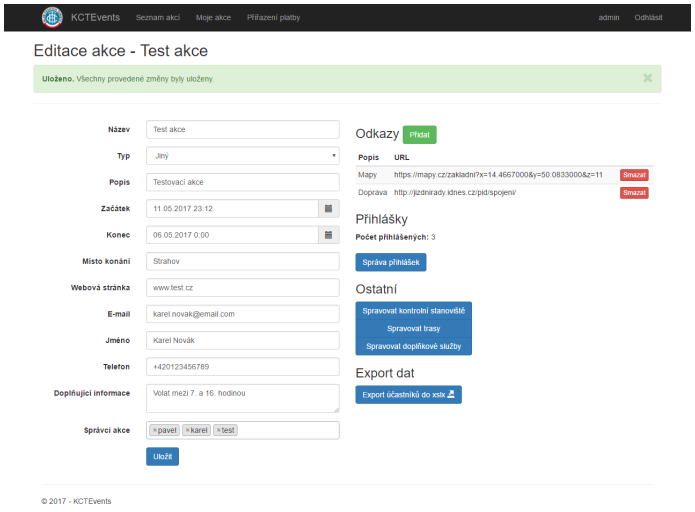
V této sekci jsou popsané metody volání API, jejich parametry a návratové hodnoty. Všechny url jsou relativní k **https://events.kct.cz/api/v1.0/**, pokud není specifikováno jinak. Informace o konfiguraci serveru poskytujícího identitu včetně všech jeho potřebných koncových bodů naleznete na adrese **https://identity.kct.cz/.well-known/openid-configuration**. Pro komunikaci používám objekty ve formátu JSON. Odpověď serveru může nabývat následujících hodnot:



Obrázek 4.2: Diagram databáze KCTEvents



Obrázek 4.3: KCTEventsIdentityServer – přihlašovací obrazovka



Obrázek 4.4: KCTEvents – stránka akce

HTTP Kód	**Název**	**Popis**
200	OK	Požadavek proběhnul v pořádku.
400	BadRequest	Server odmítl zpracovat požadavek, protože selhala jeho validace.
401	Unauthorized	Přístup k požadovanému zdroji vyžaduje přihlášení.
403	Forbidden	Přihlášený uživatel nemá přístup k požadovanému zdroji.
404	NotFound	Požadovaná stránka nebo zdroj s daným id neexistuje.
409	Forbidden	Přihlášený uživatel nemá přístup k požadovanému zdroji.
500	Internal server error	Nastala nespecifikovaná chyba při zpracování požadavku.

V těle chybové odpovědi je k dispozici objekt se specifikací chyby. Obsahuje následující atributy:

Název vlastnosti	**Hodnota**	**Popis**
code	int	Kód chyby.
message	string	Stručný popis chyby v angličtině.

4.3.1 Eventy

Zde jsou rozebrané všechna volání sovisející se získáváním informací o událostech. Seznam všech dostupných metod:

Metoda	HTTP Požadavek
getEvent	GET /event/{eventId}
list	GET /event/list

4.3.1.1 getEvent (GET /event/{eventId})

Získá podrobné informace o události, například pokud obsluha kontrolního stanoviště potřebuje kontakt na organizátora akce .Parametry požadavku z URL jsou následující:

Parametr	Hodnota	Popis
eventId	int	Id požadované události.

V těle odpovědi, pokud požadavek neskončí chybou, nalezneme následující strukturu:

Název vlastnosti	Hodnota	Popis
eventId	int	Id události.
name	string	Název.
start	DateTime	Datum a čas začátku.
end	DateTime	Datum a čas konce.
type	byte	Typ události. 0 – pochod, 1 – organizační, 2 – jiný
description	string	Popis.
venue	string	Místo konání.
webpage	string	Webová stránka.
contactName	string	Jméno pořadatele.
contactEmail	string	Email na pořadatele.
contactPhone	string	Telefonní číslo pořadatele.
contactOther	string	Další informace o pořadateli.

4.3.1.2 list (GET /event/list)

Získá seznam všech akcí dostupných přihlášenému uživateli (všech, pokud se jedná o administrátora). Slouží primárně k získání Id akce pro její další obsluhu. Výsledky jsou stránkované a seřazené sestupně podle id) Nemá žádné parametry v URL, ale v dotazu (query) lze specifikovat následující:

Parametr	Hodnota	Popis
count	int	Počet výsledků na stránku. Nejmenší možná hodnota: 1.
page	int	Číslo požadované stránky. Nejmenší možná hodnota: 1.
searchString	string	Vyhledávací řetězec. Dotaz vrátí jen výsledky, které obsahují zadaný výraz. Nerozlišuje velikost písmen.

V těle odpovědi, nalezneme seznam výsledků, každý výsledek je v následujícím tvaru:

Název vlastnosti	Hodnota	Popis
eventId	int	Id události.
name	string	Název.
description	string	Popis.
start	DateTime	Datum a čas začátku.
end	DateTime	Datum a čas konce.

4.3.2 Registrace

V této části je rozebrán přístup ke zdrojům souvisejících se správou registrací. Seznam všech metod:

Metoda	HTTP Požadavek
form	GET /registration/{eventId}/form
getRegistration	GET /registration/{eventId}/{registrationId}
postRegistration	POST /registration/{eventId}
putRegistration	PUT /registration/{eventId}/{registrationId}

4.3.2.1 form (GET registration/{eventId}/form)

Získá prázdný registrační formulář pro zvolenou službu. Obsahuje všechny nutné údaje pro zobrazení plnohodnotného formuláře uživateli. Po vyplnění údajů je zaslán na server pomocí metody „postRegistration“, která je zdokumentovaná v sekci 4.3.2.2. Dotaz vyžaduje následující parametry v URL:

Parametr	Hodnota	Popis
eventId	int	Id události.

V těle odpovědi je objekt s následujícími vlastnostmi a pomocnými objekty:

Obsah těla odpovědi metody **form**		
Název vlastnosti	**Hodnota**	**Popis**
eventId	int	Id události. Pouze informativní a nemá vliv na výsledek žádného dotazu.
registrationId	int	Id registrace. Pouze informativní a nemá vliv na výsledek žádného dotazu.
eventName	string	Název události.
name	string	Jméno účastníka.
surname	string	Příjmení účastníka.
registrationDate	DateTime	Datum a čas registrace.
identificationNumber	string	Číslo průkazu/jiného identifikátoru. Může nabývat hodnoty null.
email	string	E-mail. Může nabývat hodnoty null.
phone	string	Telefon. Může nabývat hodnoty null.
address	string	Adresa. Může nabývat hodnoty null.
birthDate	DateTime	Datum narození. Může nabývat hodnoty null.
amountPaid	decimal	Částka, která již byla uhrazena.
selectedRouteId	int	Id zvolené trasy. Může nabývat hodnoty null.
availableRoutes	AvailableRoute[]	Seznam všech dostupných tras.
services	ServiceSelector[]	Seznam selektorů všech dostupných služeb.
skipCapacityCheck	bool	Příznak, zda má server přeskočit kontrolu dostupné kapacity.

Třída **AvailableRoute**		
Název vlastnosti	**Hodnota**	**Popis**
routeId	int	Id trasy.
name	string	Název trasy.
description	string	Popis trasy.
length	double	Délka trasy v km.

Třída **ServiceSelector**		
Název vlastnosti	**Hodnota**	**Popis**
serviceId	int	Id služby.
name	string	Název služby.
description	string	Popis služby.
required	bool	Příznak, zda je služba vyžadovaná – musí být vybrána možnost a vybraný počet musí být větší než nula.
countable	bool	Příznak, zda je možné si dané služby objednat více než jednu jednotku.
selectedCount	int	Počet objednaných jednotek.
selectedIptionId	int	Zvolená možnost ze seznamu.
availableOptions	AvailableServiceOption[]	Seznam všech dostupných možností.

Třída **AvailableServiceOption**		
Název vlastnosti	**Hodnota**	**Popis**
optionId	int	Id varianty.
name	string	Název varianty – může být null.
price	decimal	Cena.
remainingCapacity	int	Zbývající kapacita volby. null pokud není kapacitně omezená.

4.3.2.2 postRegistration (POST registration/{eventId})

Vytvoří novou registraci v systému. Využívá objekt, který server zašle při dotaz na registrační formulář „form“. Definici tohoto objektu lze najít v sekci 4.3.2.1. Dotaz vyžaduje následující parametry v URL:

Parametr	**Hodnota**	**Popis**
eventId	int	Id události.

Po odeslání požadavku na server proběhne kontrola následujícího:

- Vlastnosti „email“, „phone“ a „identificationNumber“ jsou unikátní v rámci akce (pokud nejsou null).
- Služby označené jako required jsou zvolené (vybraná varianta a počet je větší než nula).

- Kapacita žádné služby nebyla překročena (tuto kontrolu lze přeskočit pomocí vlastnosti „skipCapacityCheck“).

- Formát všech zadaných dat je platný.

V případě, že všechny kontroly proběhnou a nezjistí žádnou chybu, server odpoví zprávou s kódem 200 a v těle odešle následující objekt s klíčem nově vytvořené registrace:

Název vlastnosti	**Hodnota**	**Popis**
eventId	int	Id události.
registrationId	int	Id nově vytvořené registrace.

4.3.2.3 getRegistration (GET registration/{eventId}/{registrationId})

Získá registrační formulář pro zvolenou přihlášku. Objekt v odpovědi je stejný jako u dotazu „form“, ale již jsou v něm vyplněná data konkrétní přihlášky. Definice objektu je dostupná v sekci 4.3.2.1. Dotaz očekává následující parametry v URL:

Parametr	**Hodnota**	**Popis**
eventId	int	Id události.
registrationId	int	Id registrace.

4.3.2.4 putRegistration (POST registration/{eventId}/{registrationId})

Dotaz na úpravu registrace. V těle dotazu je očekáván objekt, který se získá pomocí dotazu „getRegistration“. Tento dotaz je popsaný v sekci 4.3.2.3. Při zpracování dotazu probíhá stejná kontrola správnosti, jako u dorazu „postRegistration“, která je popsaná v sekci 4.3.2.2. Dotaz očekává následující parametry v URL:

Parametr	**Hodnota**	**Popis**
eventId	int	Id události.
registrationId	int	Id registrace.

4.3.3 Kontrolní stanoviště

V sekci jsou popsané všechny metody související s evidencí průchodů kontrolními stanovišti. Přehled všech metod:

Metoda	HTTP Požadavek
list	GET /checkpoint/list/{eventId}
passList	GET /checkpoint/pass/list/{eventId}
postPass	POST /checkpoint/pass
postPassBatch	POST /checkpoint/pass/batch

4.3.3.1 list (GET /checkpoint/list/{eventId})

Získá seznam všech kontrolních stanovišť pro požadovanou akci. Jsou vyžadované následující URL parametry:

Parametr	Hodnota	Popis
eventId	int	Id události.

V dotazu (query) žádné parametry přijímány nejsou. Tělo odpovědi obsahuje seznam objektů, které mají následující tvar:

Název vlastnosti	Hodnota	Popis
checkpointId	int	Id kontrolního stanoviště.
name	string	Název.
description	string	Popis.
latitude	double	Zeměpisná šířka v desítkové soustavě z intervalu $[-90, 90]$
longitude	double	Zeměpisná délka v desítkové soustavě z intervalu $[-180, 180]$

4.3.3.2 passList (GET /checkpoint/pass/list/{eventId})

Získá seznam všech průchodů kontrolními stanovišti pro zvolenou akci. Povinné parametry z URL jsou:

Parametr	Hodnota	Popis
eventId	int	Id události.

Dále je možné získat pouze podmnožinu výsledků specifikací následujících nepovinných parametrů v dotazu(query):

Parametr	Hodnota	Popis
checkpointId	int	Id kontrolního stanoviště.
registrationId	int	Id registrace.

V těle odpovědi na dotaz je seznam objektů, které mají následující atributy:

Název vlastnosti	Hodnota	Popis
checkpointId	int	Id kontrolního stanoviště.
name	string	Název.
description	string	Popis.
latitude	double	Zeměpisná šířka v desítkové soustavě z intervalu $[-90, 90]$
longitude	double	Zeměpisná délka v desítkové soustavě z intervalu $[-180, 180]$

4.3.3.3 postPass (POST /checkpoint/pass)

Slouží k odeslání informace o průchodu stanovištěm na server. Nepřijímá žádné parametry, pouze objekt v těle požadavku. V případě neúspěchu vrátí chybový stav a objekt chyby popsaný v sekci 4.3. Na straně serveru proběhne kontrola, zda požadovaná registrace/kontrolní stanoviště existují a zda se nejedná o duplicitní průchod. Pokud dotaz proběhne v pořádku, vrátí odpověď s kódem 200 (OK) a prázdné tělo. V těle požadavku je požadován objekt s následujícími parametry:

Název vlastnosti	Hodnota	Popis
checkpointId	int	Id kontrolního stanoviště.
time	DateTime	Čas průchodu.
idType	byte	Typ identifikátoru, podle kterého se má vyhledat registrace. 0 – Id registrace, 1 – Identifikační číslo průkazu.
identification	string	Identifikátor odpovídající zvolenému typu ve vlastnosti „idType“.

4.3.3.4 postPassBatch (POST /checkpoint/pass/batch)

Slouží k hromadnému odeslání průchodů kontrolními stanovišti. V těle požadavku je očekáván seznam objektů stejného typu, jako při odesílání jednoho průchodu popsaného v sekci 4.3.3.3. V případě úspěšného načtení seznamu z požadavku vždy vrací návratový kód 200 (OK) a v případě, že se nepovedlo zpracovat některé záznamy vrátí v těle odpovědi seznam chyb ve formátu:

Název vlastnosti	Hodnota	Popis
code	int	Chybový HTTP kód, který by nastal v případě funkce pro odeslání jednoho průchodu.
message	string	Stručný popis chyby v angličtině.
request	object	Objekt, jehož zpracování skončilo chybou. Je stejný jako v požadavku.

4.3.4 CheckIn

Zde jsou popsané všechny dotazy spojené s odbavováním účastníků na akcích. Předpokládá se nárok na co nejnižší datovou náročnost a klienti by si měli umět vytvořit vlastní lokální databázi, ve které budou vyhledávat příchozí, proto jsou metody koncipovány pro snadné vytvoření lokální databáze v klientské aplikaci. Seznam všech dostupných metod:

Metoda	**HTTP Požadavek**
list	GET /checkin/{eventId}/list
serviceList	GET /checkin/{eventId}/services
routeList	GET /checkin/{eventId}/routes
checkIn	POST /checkpoint/{eventId}/{registrationId}

4.3.4.1 list (GET /checkin/{eventId}/list)

Slouží k získání úplného nebo inkrementálního seznamu přihlášek v kompaktní formě – je uveden pouze identifikátor u služeb a trasy. Požadavek musí obsahovat následující parametry v URL:

Parametr	**Hodnota**	**Popis**
eventId	int	Id akce.

Požadavek zároveň slouží k vyhledávání přihlášek podle osobních údajů. Kompletní informace k přihlášce V dorazu (query) lze zadat následující parametry:

Parametr	**Hodnota**	**Popis**
from	DateTime	Při vyplnění tohoto parametru vrátí dotaz pouze přihlášky s datem registrace novějším nebo stejným.
name	string	Pouze přihlášky u kterých jméno účastníka začíná na zadaný řetězec. Nerozlišuje velikost písmen.
surname	string	Pouze přihlášky u kterých příjmení účastníka začíná na zadaný řetězec. Nerozlišuje velikost písmen.
identificationNumber	string	Pouze přihlášky, u kterých identifikační číslo začíná zadaným řetězcem. Nerozlišuje velikost písmen.
email	string	Pouze přihlášky, u kterých e-mail začíná zadaným řetězcem. Nerozlišuje velikost písmen.
phone	string	Pouze přihlášky, u kterých telefonní číslo končí zadaným řetězcem (může ignorovat předčíslí). Nerozlišuje velikost písmen a ignoruje bílé znaky.

V odpovědi nalezneme seznam objektů, které odpovídají parametrům, případně všech objektů. Parametry a jejich hodnoty v jednom vráceném objektu a jeho podobjektech jsou následující:

Obsah těla odpovědi metody **list**		
Název vlastnosti	**Hodnota**	**Popis**
registrationId	int	Id registrace.
name	string	Jméno
surname	string	Příjmení.
registrationDate	DateTime	Datum registrace.
routeId	int	Id trasy. Může být null.
identificationNumber	string	Identifikační číslo. Může být null.
email	string	E-mail. Může být null.
phone	string	Telefon. Může být null.
address	string	Adresa. Může být null.
birthDate	Date	Datum narození.
checkInDate	DateTime	Datum a čas odbavení. Může být null – účastník ještě nebyl odbaven.
services	ServiceCompact[]	Kolekce všech objednaných služeb.

Trída **ServiceCompact**		
Název vlastnosti	**Hodnota**	**Popis**
serviceOptionId	int	Id varianty služby.
count	int	Počet objednaných jednotek.

4.3.4.2 serviceList (GET /checkin/{eventId}/services)

Vrátí seznam všech možných voleb všech služeb dané akce. V URL požadavku jsou očekávány následující atributy:

Parametr	**Hodnota**	**Popis**
eventId	int	Id akce.

Odpověď serveru obsahuje seznam objektů ve tvaru:

Název vlastnosti	**Hodnota**	**Popis**
serviceOptionId	int	Id varianty služby.
serviceName	string	Název služby.
optionName	string	Název varianty. Může být null.
price	decimal	Cena za jednotku.

4.3.4.3 routeList (GET /checkin/{eventId}/routes)

Vrátí seznam všech tras dané akce. V URL požadavku jsou očekávány následující parametry:

Parametr	**Hodnota**	**Popis**
eventId	int	Id akce.

Odpověď serveru obsahuje seznam objektů ve tvaru:

Název vlastnosti	**Hodnota**	**Popis**
routeId	int	Id trasy.
optionName	string	Název trasy.
length	double	Délka trasy v km.

4.3.4.4 checkIn (POST /checkpoint/{eventId}/{registrationId})

Odbavení zvolené registrace. Uloží datum odbavení, případně částku, která byla uhrazena. Pokud proběhne požadavek bez chyby, je návratový kód odpovědi 200 (OK) a tělo je prázdné. Odeslání duplicitního záznamu skončí chybou 409 (Conflict). V URL požadavku jsou očekávány následující parametry:

Parametr	**Hodnota**	**Popis**
eventId	int	Id akce.
registrationId	int	Id registrace.

V těle požadavku je požadován objekt s následujícími atributy:

Název vlastnosti	**Hodnota**	**Popis**
time	DateTime	Datum a čas odbavení.
moneyPaid	decimal	Částka uhrazená u odbavení. Nula v případě, že nebyl uhrazen žádný obnos.

Kapitola 5

Testování a možnosti dalšího vývoje

Testování jsem provedl pouze manuální, podle scénářů plynoucích z případů užití ze sekce 3.2. Testování administračního rozhraní proběhlo v prohlížeči Google Chrome pří různých velikostech oken prohlížeče tak, aby se nasimuloval vzhled stránky na všech zařízeních – desktop, tablet, mobilní telefon. Testování API proběhlo pomocí aplikace Postman [30].

5.1 Testovací scénáře

5.1.1 Správa uživatelských účtů

Bylo ověřeno, že na autorizačním serveru lze vytvořit nového uživatele a přidělit mu požadované role v systému. Také byla otestováno, že lze editovat již vytvořeného uživatele, případně ho smazat.

5.1.2 Správa klientů

Bylo ověřeno, že lze vytvořit nového klienta a provést jeho editaci, případně ho smazat. Při testování tohoto scénáře byla objevena chyba při vkládání povolených adres pro přesměrování, která byla záhy opravena.

5.1.3 Přihlášení do systému

Bylo otestováno, že se lze bez potíží přihlásit do administračního rozhraní a že autorizační server vydává tokeny, pomocí kterých pak lze přistupovat k API.

5.1.4 Správa akcí

Bylo otestováno, že v systému lze vytvořit akci a následně ji editovat. Bylo ověřeno, že lze korektně spravovat (přidávat, upravovat, mazat) kontrolní sta-

noviště, trasy a dostupné služby včetně jejich voleb. Tento test ukázal, že nefunguje žádný formulář s metodou POST, pokud má nějaké parametry v dotazu (query) a selže validace jeho modelu – parametry se po odeslání formuláře smazaly. Tato chyba byla opravena. Dále bylo zjištěno, že v některých formulářích se korektně nepředávají některé proměnné v modelu a bylo třeba je přidat do formuláře.

5.1.5 API

Všechny metody API byly otestovány včetně všech jejich parametrů, jestli plně odpovídají API referenci ze sekce 4.3. Při těchto testech nebyly zjištěny žádné problémy a sytém by měl fungovat tak, jak je popsáno v referenci.

5.1.6 Zabezpečení

Bylo otestováno, že se nepřihlášený uživatel nedostane k žádným zdrojům, ke kterým nemá přístup. Dále bylo otestováno, že se přihlášený uživatel nedostane ke zdrojům, které ke kterým mu jeho role v systému nedává přístup, případně že nemůže zasahovat do akcí, u kterých není evidován jako správce. Byl také proveden test, jestli korektně funguje validace anti-forgery tokenů – zda je tedy systém nenapadnutelný pomocí CSRF útoků.

5.2 Závěry testování

Testování ukázalo, že systém funguje dobře jako celek a objevilo se při něm několik chyb, které by znepříjemnily jeho používání. Tyto chyby byly již opraveny. Také bylo ověřeno, že se webové stránky zobrazují korektně na všech zařízeních. Předpokládá se, že systém bude nasazen v testovacím režimu pro vývoj navazujících klientských aplikací, během kterého budou případně odhaleny další chyby předtím, než se systém nasadí do ostrého provozu.

5.3 Možnosti dalšího vývoje

V současné době je realizované pouze backendové řešení systému a předpokládá se jeho návaznost hned na několik aplikací. Použití QR kódu pouze z průkazek členů mi nepřišlo dostačující, proto je systém připraven na použití libovolného identifikačního čísla, která by se nechala vytisknout na samolepky. Ty by se potom jen přilepily na kartu účastníka a zaevidovaly do systému k příslušné přihlášce. Další příležitostí je využití identifikačního serveru, který vznikl jako součást této práce pro budoucí projekty KČT. Při vývoji této aplikace jsem předpokládal vznik dalších tří klientských aplikací, které budou zasílat data na server.

5.3.1 Mobilní aplikace pro kontrolní stanoviště

Mobilní aplikace pro Android, která je určena organizátorům na kontrolních stanovištích. Jejím úkolem je pouze evidovat průchody daným kontrolním stanovištěm. Měla by být schopná přečíst QR kód z průkazky člena nebo karty účastníka, případně obsahovat možnost manuálně zadat id registrace a tato data odeslat na server. Měla by být schopná pracovat i v offline režimu pro případ, že kontrolní stanoviště bude v místě se špatným signálem.

5.3.2 Desktopová aplikace pro odbavení účastníků

Aplikace, která se bude starat o odbavování účastníků při příchodu na akci, případně na jejím konci. Měla by umět vytvářet nové registrace pro ty, kteří přišli neregistrovaní a editovat přihlášky. Mohla by obsahovat i možnost připojení tiskárny, aby se mohly karty účastníka tisknout přímo na místě. Měla by být schopná si stáhnout všechny přihlášky do vlastní databáze a pak fungovat i offline.

5.3.3 Javascriptový registrační formulář

Registrační formulář napsaný v JavaScriptu, který se dokáže sám vykreslit podle poskytnutého JSON formuláře a po svém vyplnění odeslat vyplněný formulář zpět na server. Neměl by sám nijak manipulovat s JWT tokeny, protože bude veřejně přístupný a pouze předávat data na server, který je potom bezpečně pošle na do systému pomocí API. Formulář by se měl dát vložit na existující stránky akcí, případně integrovat přímo do administračního rozhraní KCTEvents.

Závěr

Cílem práce bylo navrhnout a vytvořit backendové řešení systému pro evidenci účastníků akcí KČT. Po analýze problému a požadavků vznikl návrh výsledného systému. Na základě tohoto návrhu byly implementovány dvě aplikace, které tvoří základ celého systému.

Cíle práce se podařilo naplnit v podobě webového API a jeho administračního rozhraní. Povedlo se splnit všechny body zadání a navíc vznikl systém pro správu identit, který podporuje protokoly OAuth2 a OpenID Connect. Tento systém může být využit pro případné další projekty KČT.

Hlavním přínosem práce pro mě bylo získání nových zkušeností s novým frameworkem .NET Core, jehož vývoj sleduji už delší dobu a toto byla moje první příležitost jej vyzkoušet na reálném projektu. Také jsem si vyzkoušel práci s PostgreSQL databází, se kterou jsem neměl předešlé zkušenosti.

Literatura

[1] Poslední puchýř 2017 - Bučovice. [online], [cit. 2017-05-9]. Dostupné z: http://www.puchyr-bucovice.cz/prihlasky.php

[2] An Introduction to OAuth 2 | DigitalOcean. [online], červenec 2015, [cit. 2017-05-9]. Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[3] Formuláře Google. [online], [cit. 2017-04-24]. Dostupné z: https://docs.google.com/forms/u/0/

[4] Millares, G.: Top 5 Programming Languages Used In Web Development. listopad 2015, [cit. 2017-04-22]. Dostupné z: http://blog.stoneriverelearning.com/top-5-programming-languages-used-in-web-development/

[5] Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework. [online], [cit. 2017-04-22]. Dostupné z: https://nette.org/cs/

[6] Symfony, High Performance PHP Framework for Web Development. [online], [cit. 2017-04-22]. Dostupné z: https://symfony.com/

[7] .NET - Powerful Open Source Development. [online], [cit. 2017-04-22]. Dostupné z: https://www.microsoft.com/net/core#windowsvs2017

[8] Čápka, D.: MVC architektura. [online], [cit. 2017-04-22]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor

[9] Malý, M.: REST: architektura pro webové API - Zdroják. [online], srpen 2009, [cit. 2017-04-22]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[10] HTTP Methods for RESTful Services. [online], [cit. 2017-04-22]. Dostupné z: http://www.restapitutorial.com/lessons/httpmethods.html

[11] Entity Framework. [online], [cit. 2017-04-24]. Dostupné z: https://docs.microsoft.com/en-us/ef/

[12] Database First development with Entity Framework. [online], [cit. 2017-04-24]. Dostupné z: http://www.entityframeworktutorial.net/choosing-development-approach-with-entity-framework.aspx

[13] The MariaDB Foundation – Ensuring continuity and open collaboration. [online], [cit. 2017-04-24]. Dostupné z: https://mariadb.org/

[14] PostgreSQL: The world's most advanced open source database. [online], [cit. 2017-04-24]. Dostupné z: https://www.postgresql.org/

[15] Npgsql - .NET Access to PostgreSQL | Npgsql Documentation. [online], [cit. 2017-04-24]. Dostupné z: http://www.npgsql.org/

[16] Sancheti, S.: Scalability Design Principles - Internal Session. [online], červenec 2013, [cit. 2017-05-15]. Dostupné z: https://www.slideshare.net/ssachin7/scalability-design-principles-internal-session

[17] OAuth 2.0 – OAuth. [online], [cit. 2017-04-25]. Dostupné z: https://oauth.net/2/

[18] OpenID Foundation website. [online], [cit. 2017-05-7]. Dostupné z: http://openid.net/

[19] Sevilleja, C.: The Anatomy of a JSON Web Token | Scotch. [online], leden 2015, [cit. 2017-04-25]. Dostupné z: https://scotch.io/tutorials/the-anatomy-of-a-json-web-token

[20] MojeID - mainpage. [online], [cit. 2017-05-9]. Dostupné z: https://www.mojeid.cz/

[21] OpenID Connect explained | Connect2id. [online], [cit. 2017-05-9]. Dostupné z: https://connect2id.com/learn/openid-connect

[22] Open Source OpenID Connect and OAuth 2.0 framework for .NET. [online], [cit. 2017-05-9]. Dostupné z: http://identityserver.io/

[23] ASP.NET Core MVC - Authentication And Role Based Authorization With ASP.NET Core Identity. leden 2017, [cit. 2017-05-9]. Dostupné z: http://www.c-sharpcorner.com/article/asp-net-core-mvc-authentication-and-role-based-authorization-with-asp-net-core/

[24] Bootstrap The world's most popular mobile-first and responsive front-end framework. [online], [cit. 2017-05-9]. Dostupné z: http://getbootstrap.com/

[25] select * from depesz; » Blog Archive » CHAR(x) vs. VARCHAR(x) vs. VARCHAR vs. TEXT – UPDATED 2010-03-03. březen 2010, [cit. 2017-05-9]. Dostupné z: https://www.depesz.com/2010/03/02/charx-vs-varcharx-vs-varchar-vs-text/

[26] Novinky v sadě Visual Studio 2017. [online], [cit. 2017-05-9]. Dostupné z: https://www.visualstudio.com/cs/vs/whatsnew/

[27] A cross-platform .NET IDE based on the IntelliJ platform and ReSharper. [online], [cit. 2017-05-9]. Dostupné z: https://www.jetbrains.com/rider/

[28] Bitbucket | The Git solution for professional teams. [online], [cit. 2017-05-9]. Dostupné z: https://bitbucket.org/product

[29] SourceTree | Free Git and Hg Client for Mac and Windows. [online], [cit. 2017-05-9]. Dostupné z: https://www.sourcetreeapp.com/

[30] Postman | Supercharge your API workflow. [online], [cit. 2017-05-9]. Dostupné z: https://www.getpostman.com/

Příloha **A**

Seznam použitých zkratek

JSON JavaScript Object Notation

API Application Programming Interface

GUI Graphical user interface

KČT Klub českých turistů

OS Operační systém

WWW World Wide Web

GPS Global Positioning System

REST Representational state transfer

MVC Model-view-controller

HTTP Hypertext Transfer Protocol

URI Uniform Resource Identifier

CRUD Create, read, update and delete

XML eXtensible Markup Language

RFID Radio Frequency Identification

ORM Objektově relační mapování

JWT JSON web token

URL Uniform Resource Locator

IDE Integrated Development Environment

CSRF Cross-site request forgery

DOS Denial of service

DDoS Distributed denial of service

PDF Portable document format

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `ea` ..... soubor projektu Enterprise Architect
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF