Sem vložte zadání Vaší práce.

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

Diplomová práce

Systém pro centrální správu a integraci rezervačních systémů sportovních zařízení

Bc. Jan Fránek

Vedoucí práce: doc. Ing. Tomáš Vitvar, Ph.D.

18. ledna 2016

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 18. ledna 2016 .....

Odkaz na tuto práci

Fránek, Jan. *Systém pro centrální správu a integraci rezervačních systémů sportovních zařízení.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá integrací rezervačních systémů sportovních zařízení. Na začátku je proveden průzkum a analýza existujících rezervačních systémů pro sportoviště. Dále je provedena analýza způsobu integrace rezervačních systémů pro hotely. Poté jsou navrženy funkční požadavky, datový model, architektura systému a rozhraní pro integraci. V další části je popsána implementace systému a v závěru provedeno testování.

Klíčová slova integrace, rezervační systém, webová služba, webová aplikace

Abstract

This thesis is focused on the integration of reservation systems for sport facilities. The work starts with a survey and analysis of the existing reservation systems for sport facilities. Next there is an analysis of integration of reservation systems for hotels. Afterthat I proposed functional requirements, data model, system architecture and interfaces for integration. The next sections describe the system implementation and system testing.

Keywords integration, reservation system, web service, web application

Obsah

Seznam obrázků

Úvod

V této praci se chci zaměřit na řešení problému hledání sportoviště na internetu jako je například sportovní hala, squashový kurt nebo fitness centrum apod.

Například pokud chci najít volnou sportovní halu pro trénování florbalu, tak musím nejprve jít na katalog se sportovníma halama, tam se prokliknout na webovou stránku haly a napsat nebo zavolat, jestli nemají nějaký volný termín. Některé sportovní haly používají rezervační systém, takže hned vidím, kdy mají volno a kdy obsazeno, případně si mohu termín rezervovat. Nevýhodou ale je, že tato informace, zda je volno nebo ne, není někde přístupná z jednoho místa.

Cílem práce je proto vytvořit systém, který by zákazníkům umožňoval vyhledávat a rezervovat sportoviště z jednoho přístupového místa. Data o dostupnosti sportoviště by se buď získávala tím, že by se stávající systém sportoviště do vyvíjeného systému integroval, anebo by sportoviště zadávalo data přes administrátorské rozhraní přímo do systému. Pro tento problém se mohu inspirovat v doméně cestování jako je například hledání hotelu ve známem rezervačním systému Booking.com [1] nebo Expedia [2].

Práce je strukturována následovně:

- V kapitole 1 jsem provedl průzkum existujících rezervačních systémů sportovišť a analýzu integrace rezervačních systémů v doméně hotelů.
- V kapitole 2 jsem definoval požadavky na vyvíjený systém, navrh datový model a popsal rozhraní pro integraci. Dále jsem navrh softwarovou architekturu a model pro nasazení systému.
- V kapitole 3 jsem popsal strukturu a funkce implementovaného systému.

[1] http://www.booking.com
[2] https://www.expedia.com

- Na závěr jsem provedl v kapitole 4 integrační, akceptační a systémové testování implementovaného systému.

Kapitola 1

Průzkum rezervačních systémů

V této kapitole jsem provedl průzkum existujících rezervačních systémů pro sportoviště. Poté jsem provedl analýzu integrování rezervačních systémů v doméně hotelů. Struktura kapitoly je následující:

- 1.1 Průzkum rezervačních systémů pro sportoviště
- 1.2 Integrace rezervačních systémů v doméně hotelů
- 1.3 Shrnutí

1.1 Průzkum rezervačních systémů pro sportoviště

V této části jsem provedl průzkum systémů pro rezervaci a správu sportovišť jako jsou sportovní centra, sportovní haly, fitness kluby apod. Rezervační systémy, na které jsem při průzkumu narazil, jsem rozdělil do tří kategorií:

- **Systém používaný jedním sportovištěm.** Tyto systémy jsou používány jedním sportovištěm nebo skupinou sportovišť, které vlastní jedna firma. Příkladem je rezervační systém areálu Hamr [3] nebo fitness klub Holmes Place [4]
- **Systém jako SaaS.** Tyto rezervační systémy jsou provozovány třetí stranou. Každému sportovišti je poskytnuto webové rozhraní pro správu sportoviště a webové rozhraní pro zákazníky. Mezi takové systémy patří

[3] http://hodiny.hamrsport.cz
[4] http://holmesplace.cz

například systém Bizzy [5], Fitsystem [6], iSport system [7], Bookitme [8], Bookingbug [9] nebo Mindbody [10].

- **Systém s vyhledáváním.** Tyto rezervační systémy poskytují zákazníkům jednotné webové rozhraní, přes které mohou vyhledávat mezi více sportovištěma a rezervovat si u nich hřistě nebo cvičení. Mezi tyto systémy patří například ActiveSG [11], EasySport [12], OpenPlay [13], Jdemenato [14] nebo FitReserve [15].

1.1.1 Analýza vybraných systémů pro sportoviště

U rezervačních systémů mne zajímaly následující funkce a vlastnosti:

- Jaké služby se rezervují a co mají za atributy.
- Granularita termínů rezervovatelného hříště.
- Jestli systém používá cenová pásma a cenové kategorie.
- Kolik lze zarezervovat termínů při jedné rezervaci.
- Jakým způsobem může zákazník rezervaci uhradit.
- Jaká lze v systému nastavit omezení pro rezervování nebo stornování

1.1.1.1 Rezervační systém areálu Hamr

Rezervační systém používají tři sportovní areály v Praze. Zákaznící k rezervačnímu systému přistupují přes webovou adresu `http://hodiny.hamrsport.cz`. Uživatel může v systému procházet rozvrhy hřišť, ale rezervovat může až po přihlášení. Příhlášený uživatel může dále vstupovat do svého účtu, kde má přehled o svých rezervacích, platbách apod. Shrnutí systému je následujicí:

- **Co se rezervuje:**
  Rezervují se hřiště a skupinová svičení. Každý termín má atributy: čas začátku, čas konce, místo a cenu. Termín cvičení má navíc atributy: instruktor, kapacita osob, počet náhradníků a poznámka.

[5] http://www.e-rezervace.cz
[6] http://www.fitsystem.cz
[7] http://www.isportsystem.cz
[8] http://www.bookitme.com
[9] http://bookingbug.com
[10] https://www.mindbodyonline.com
[11] https://www.myactivesg.com/
[12] http://www.easysport.de/easysport
[13] https://www.openplay.co.uk/
[14] https://jdemenato.cz
[15] https://www.fitreserve.com/

- **Granularita termínů:**
  Termíny pro rezervování jsou pevně nastaveny.
- **Rezervace termínů:**
  V jedné rezervaci lze zarezervovat nejvýše tři termíny jdoucí zasebou.
- **Cenová pásma:**
  Rezervační systém používá cenová pásma i cenové kategorie.
- **Způsob platby:**
  Rezervace lze hradit z kreditového účtu, permanentkou nebo hotově na místě.
- **Omezení na rezervace:**
  - Rezervaci lze provést pouze po přihlášení.
  - Rezervovat lze určitou dobu dopředu.
  - Pokud je u cvičení vyčerpaná kapacita, lze se přihlásit jako náhradník.
- **Ostatní:**
  Zákazník se může registrovat online. Jeho účet je aktivní až po ověření jeho telefonního čísla a e-mailu .

1.1.1.2 Rezervační systém Isportsystem

Sportoviště tento rezervační systém používají jako Saas. Systém je rozdělen na dvě sekce, na sekci pro zákazníky a sekci pro správu sportoviště. Webové stránky systému jsou `http://www.isportsystem.cz`.

- **Co se rezervuje:**
  Rezervují se hřiště (kurty, haly, apod.) a skupinová cvičení. Každý termín má atributy: čas začátku, čas konce, místo a cenu. Termín cvičení má navíc atributy: instruktor, fotka instruktora, kapacita osob a počet náhradníků.
- **Granularita termínů:**
  Termíny pro rezervování jsou pevně nastaveny.
- **Rezervace termínů:**
  Zákazník může zarezervovat několik termínů najednou v jedné rezervaci.
- **Cenová pásma:**
  Rezervační systém používá cenová pásma i cenové kategorie.
- **Způsob platby:**
  Rezervace lze hradit přes PayPal, z kreditového účtu, permanentkou nebo hotově na místě.

- **Omezení na rezervace:**
  - Pokud je u cvičení vyčerpaná kapacita, lze se přihlásit jako náhradník.
  - Správce může nastavit, na jak dlouho dopředu může zákazník hřiště nebo cvičení rezervovat.
  - Správce může nastavit, jestli musí být zákazník před rezervací přihlášen.
  - Správce může nastavit, do kdy může rezervaci zákazník stornovat.

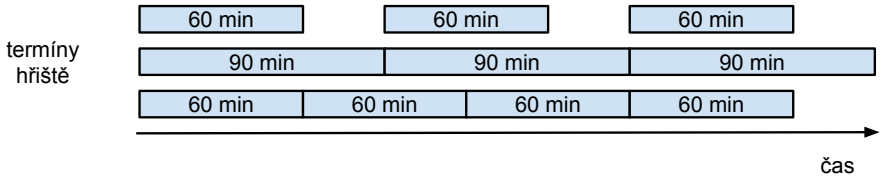
1.1.1.3 Rezervační systém Fitsystem

Sportoviště tento rezervační systém používají také jako Saas. Systém je také rozdělen na dvě sekce, na sekci pro zákazníky a sekci pro správu sportoviště. Webové stránky systému jsou `http://www.fitsystem.cz`.

- **Co se rezervuje:**
  Rezervují se hřiště (kurty, haly, apod.), fitness stroje a skupinová cvičení. Každý termín má atributy: čas začátku, čas konce, místo a cenu. Termín cvičení má navíc atributy: instruktor, kapacita osob, počet náhradníků a poznámka.
- **Granularita termínů:**
  Termíny pro rezervování se mohou v čase překrývat a mohou být různě dlouhé (viz obrázek 1.1). Při rezervaci fitness strojů lze zvolit trenéra.
- **Rezervace termínů:**
  Zarezervovat lze pouze jeden termín v jedné rezervaci
- **Cenová pásma:**
  Rezervační systém nepoužívá cenová pásma ani cenové kategorie.
- **Způsob platby:**
  Rezervace lze hradit z kreditového účtu, permanentkou nebo hotově na místě. Kreditový účet lze dobíjet platební kartou.
- **Omezení na rezervace:**
  Správce může nastavit, na jak dlouho dopředu může zákazník hřiště nebo cvičení rezervovat

1.1.1.4 Rezervační systém Jdemenato

Sportoviště tento rezervační systém používají také jako Saas. Systém je také rozdělen na dvě sekce, na sekci pro správu sportoviště a na sekci pro zákazníky. Zákazníci mohou pod jedním účtem rezervovat více sportovišť a dále mohou



Obrázek 1.1: Příklad možného rozvržení termínů hřiště.

sportoviště hledat podle aktivity, času a místa. Oficiální stránky systému jsou na `https://www.jdemenato.cz`.

- **Co se rezervuje:**
  Rezervují se hřiště (kurty, haly, apod.). Každý termín má atributy: čas začátku, čas konce, místo a cenu.
- **Granularita termínů:**
  Termíny pro rezervování jsou pevně nastaveny.
- **Rezervace termínů:**
  Zarezervovat lze pouze jeden termín v jedné rezervaci
- **Cenová pásma:**
  Rezervační systém používá cenová pásma.
- **Způsob platby:**
  Rezervace lze hradit z kreditového účtu, permanentkou nebo hotově na místě.
- **Omezení na rezervace:**
  - Rezervaci lze provést pouze po přihlášení
  - Správce může nastavit, na jak dlouho dopředu může zákazník hřiště nebo cvičení rezervovat

Výhodou tohoto systému je, že v něm lze vyhledávat z několika sportovišť podle dostupnosti z jednoho místa. Nevýhodou je, že systém neumožňuje integrovat stávající systém sportoviště.

1.1.1.5 Rezervační systém Mindbody

Tento rezervační systém je zaměřený na fitness centra v USA, Kanadě, Velké Británii a Austrálii. Systém je rozdělen na dvě sekce, na sekci pro zákazníky a sekci pro správu fitness centra. Systém je komplexní, dají se v něm rezervovat služby, prodávat produkty, spravovat prodeje, rezervace, zaměstnance, zákazníky a marketingové nástroje. Rezervační systém dále poskytuje za poplatek

webovou službu [16], pomocí níž lze přistupovat k byznys logice a datům systému. Webové stránky projektu jsou na `https://www.mindbodyonline.com`.

- **Co se rezervuje:**
  - *Trenéři.* Termín pro rezervaci trenéra má atributy: čas začátku, čas konce, trenér a cena.
  - *Skupinová cvičení.* Termín pro rezervaci cvičení má atributy: čas začátku, čas konce, cena, místo, instruktor, úroveň obtížnosti, poznámka.
- **Granularita termínů:**
  Termíny pro rezervování jsou pevně nastaveny.
- **Rezervace termínů:**
  Zarezervovat lze pouze jeden termín v jedné rezervaci.
- **Cenová pásma:**
  Rezervační systém používá cenová pásma i cenové kategorie.
- **Způsob platby:**
  Rezervace lze hradit platební kartou nebo permanentkou.
- **Ostatní:**
  - Rezervační systém má propravován systém permanentek.
  - Permanentky lze online nakupovat ve speciální sekci rezervačního systému

1.1.1.6 Rezervační systém FitReserve

Během provádění průzkumu se systém jmenoval GoRecces a umožňoval zákazníkům vyhledávat fitness centra a skupinové lekce podle času, místa a druhu aktivity. Systém využíval webové rozhraní systému Mindbody 1.1.1.5. V současné době se systém jmenuje FitReserve [17]. Jako rezervační systém se již nejeví a funkci hledání fitness center jsem nenalezl.

1.2 Integrace rezervačních systémů v doméně hotelů

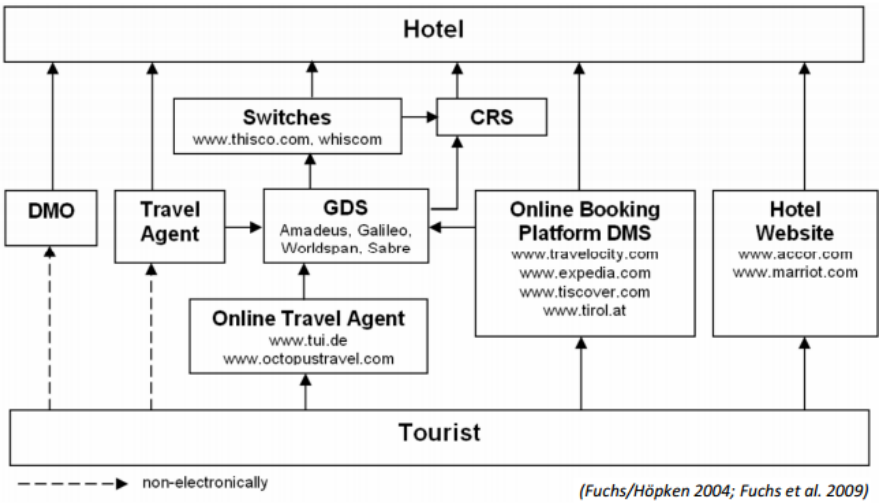
V této části jsem provedl analýzu toho, jakým způsobem může být systém hotelu integrován do velkých rezervačních systému jako jsou Bookng.com nebo Expedia. Tato sekce je strukturována následovně:

[16] https://developers.mindbodyonline.com/
[17] https://www.fitreserve.com

- 1.2.1 Přehled typů systémů
- 1.2.3 Integrace do systému Expedia
- 1.2.2 Integrace do systému ChannelRush

1.2.1 Přehled typů systémů

V této podsekci popíšu typy systémů, které hotel používá za účelem propagace svých služeb. Na obrázku 1.2 jsou znázorněny možné způsoby rezervace z pohledu zákazníka.



Obrázek 1.2: Možné způsoby rezervace hotelu z pohledu zákazníka

1.2.1.1 Property management system

Je to systém, který umožňuje správu hotelu, zákazníků, rezervací, dostupností a mnoho dalších funkcí. Kromé toho poskytuje rozhraní pro integraci s dalšími systémy.

1.2.1.2 Online booking platform

V různých zdrojích se lze setkat s označením *Internet booking engine* (IBE), *Online travel agency* (OTA). Jedná se o webovou aplikaci umožňujicí přímé rezervace na on-line bázi nebo na bázi poptávka–potvrzení rezervace. Příkladem jsou např. Booking.com, Orbitz [18], Expedia, Travelocity [19] nebo LateRooms [20]. Pokud hotel používá více těchto aplikací, může používat systém

[18] http://www.orbitz.com
[19] https://www.travelocity.com
[20] http://www.laterooms.com

označovaný jako *Channel manager*, který je s nimi integrován a udržuje v nich aktuální data o dostupnostech a cenách hotelu.

1.2.1.3 Global Distribution System

Global Distribution System (GDS) je síť provozovaná určitou společností. Umožňuje provádět automatické transakce mezi poskytovateli turistických služeb (letecké společnosti, hotely, půjčovny aut) a prodejci (cestovní kanceláře). Hlavními funkcemi systému GDS jsou rezervace letenek, ubytování, aut, plaveb, zájezdů apod. Dále umožňuje zpracovávat platby, spravovat letenky, reportování a učetní funkce. Mezi hlavními provozovateli tohoto typu systému patří například Amadeus [21], Travelport [22] a Sabre[23].

1.2.1.4 Channel manager

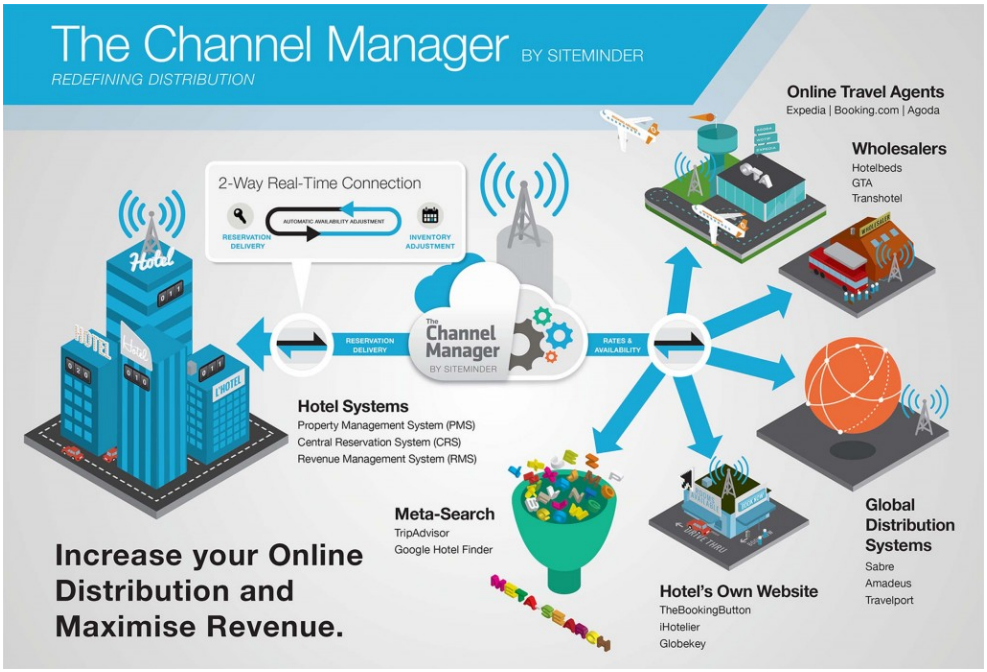
Systém umožňující udržovat aktuální data (rezervace, dostupnost a ceny pokojů, omezení na rezervace) mezi více systémy (Online booking platform, GDS, Property management system apod.). Výhodou používání systému je úspora času při udžování dat mezi systémy a minimalizování možného přerezervování. Poskytovatelů tohoto typu systému jsou desítky. Patří mezi ně např. Vertical Booking [24], Allotz [25], Siteminder [26] nebo ChannelRush [27]. Na obrázku 1.3 je znázorněn systém Siteminder.

1.2.2 Integrace do systému ChannelRush

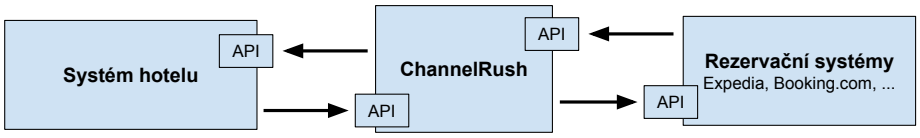
Systém ChannelRush je systémem typu *channel manager*. Systém propojuje systém pro správu hotelu s rezervačními systémy nebo se systémy typu GDS. ChannelRush poskytuje pro systém pro správu hotelu SOAP webovou službu, jejíž rozhraní umožňuje:

- Dotázat se na seznam pokojů, cenových plánů a kanálů. Kanálem se myslí konkrétní propojení systému ChannelRush a nějakého rezervačního systému (např. Expedia nebo Booking.com).
- Aktualizovat dostupnost pokojů, ceny a omezení. Dostupnost pokoje pro hotel znamená počet volných pokojů určitého typu.
- Dotázat se na rezervace a potvrzovat rezervace

[21] http://www.amadeus.com
[22] https://www.travelport.com
[23] http://www.sabre.com
[24] http://www.verticalbooking.com/
[25] http://www.allotz.com/
[26] http://www.siteminder.com/
[27] https://www.channelrush.com/



Obrázek 1.3: Schéma channel manager systému Siteminder



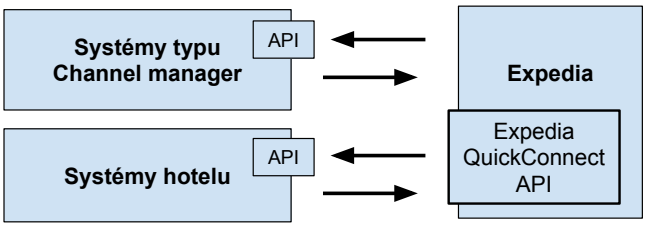
Obrázek 1.4: Schéma integrace do channel manager systému ChannelRush

Při výměně zpráv se používají identifikátory pokojů a rezervací. Tyto identifikátory si musí hotel namapovat na své identifikátory, které používá ve svém systému. Mápování se provádí pro každý použitý kanál.

Pro informování systému pro správu hotelu o nové rezervaci, změně nebo stornu systém ChannelRush nabízí dva způsoby, které nazývají:

- *Reservation Pickup and Confirm*. Systém hotelu se periodicky dotazuje, jestli není nová rezervace.
- *Reservation Delivery*. Systém hotelu poskytne URL, kam systém ChannelRush pošle informace o rezervaci.

V rámci rezervace se posílají tyto údaje: rezervované pokoje, seznam hostů a informace o platbě a ceně.



Obrázek 1.5: Schéma integrace do systému Expedia

1.2.3 Integrace do systému Expedia

Systému Expedia poskytuje rozhraní Expedia QuickConnect (EQC) [28], které je určené pro systémy hotelu a systémy typu channel manager. Rozhraní EQC umožňuje:

- Dotázat se na dostupnost pokojů, cen a omezení
- Aktualizovat dostupnost pokojů, ceny a omezení
- Dotázat se na rezervace
- Potvrzovat rezervace.

Na stránkách popisující rozhraní ECQ je uvedeno, že po provedení nové rezervace v systému Expedia je systém hotelu o rezervaci informován, ale v technické dokumentaci ECQ jsem detailnější informaci nenalezl.

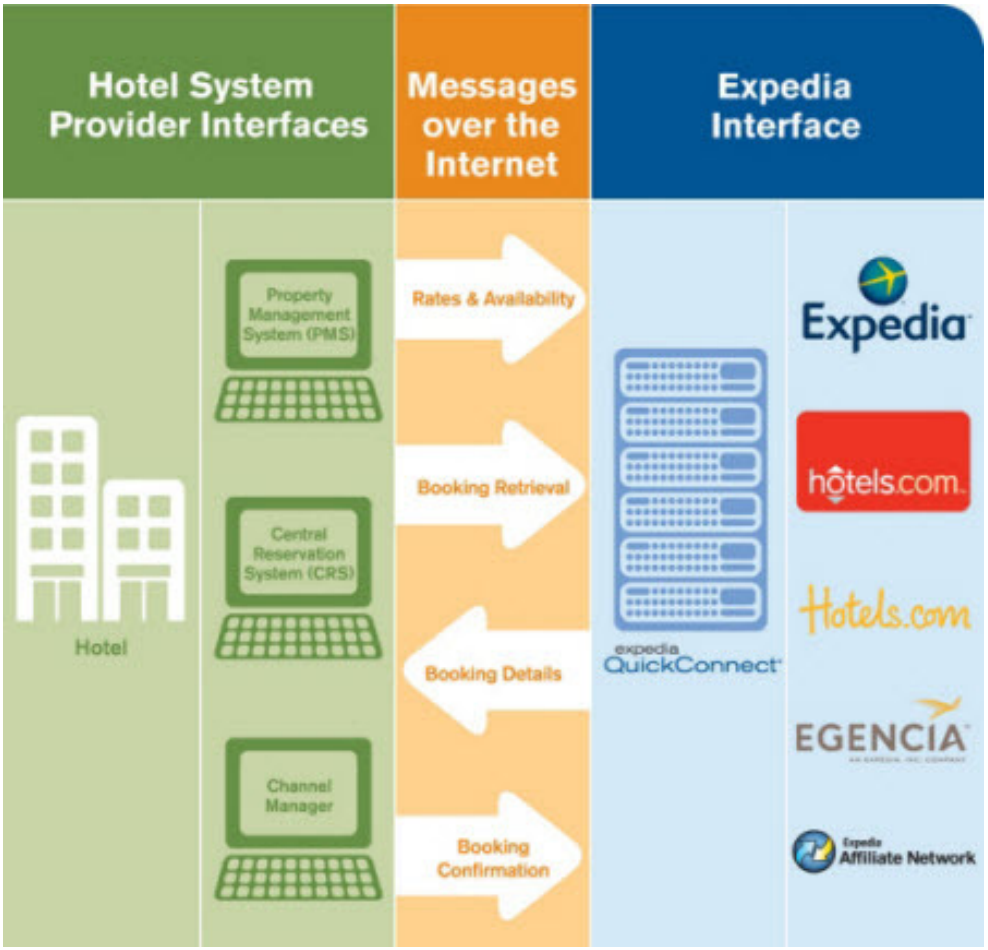
Z technického hlediska jsou zprávy vyměňované mezi EQC a partnerem kódované v jazyce XML a posílány přes zabezpěčený HTTP protokol. Informace pro autentizaci je zakódována v XML dokumentu. Komunikace je synchronní, rozhraní přijme požadavek a následně odpoví kladnou nebo zápornou odpovědí.

1.3 Shrnutí

1.3.1 Shrnutí systémů z domény hotelů

Z domény hotelů se můžu inspirovat principem integrace do systému Expedia přes rozhraní ECQ popsané v sekci 1.2.3. Vyvíjený systém by si nechal od systému sportoviště posílat dostupnost a ceny a pokud by ve vyvíjeném systému vznikla rezervace, tak by byl o tom systém sportoviště přes jeho rozhraní informován. Oproti systému Expedia jsou v rezervačním systému pro sportoviště tyto rozdíly:

[28] https://www.expediaquickconnect.com



Obrázek 1.6: Schéma rozhraní Expedia QuickConnect

- Rezervují hřiště, treneři nebo cvičení místo pokojů.
- Délka rezervovatelného termínu je v řádu minut, kdežto pokoje se rezervují na dny.
- K zaplacení rezervace lze použít kreditový účet nebo permanentku vázanou na zákazníka anebo neplatit při rezervaci vůbec a zaplatit až na místě. V systému Expedia je potřeba pro prevedení rezervace zaplatit platební kartou.

1.3.2 Shrnutí systémů pro sportoviště

Rezervační systémy pro sportoviště bych shrnul v těchto bodech:

- **Co se rezervuje:**
  Abstraktně se rezervují *služby* (hřiště, treneři nebo cvičení), které mají termíny. Termíny mají atributy *čas začátku*, *čas konce* a *cenu*. Termíny skupinového cvičení mají navíc atributy *název cvičení*, *instruktor*, *sál*, *obtížnost cvičení*, *maximální počet účastníků (kapacita)* a *počet možných náhradníků*.

- **Granularita termínů:**
  U cvičení jsou termíny pevně dány. U rezervace hřiště mohou termíny být různě dlouhé (například 30 min, 60 min, 90 min) a mohou se v čase překrývat (viz Obrázek 1.1).

- **Omezení na rezervace:**
  - Rezervaci lze provést pouze po přihlášení.
  - Správce může nastavit, na jak dlouho dopředu může zákazník hřiště nebo cvičení rezervovat.
  - Správce může nastavit, do jaké doby může zákazník ještě hřiště nebo cvičení rezervovat.
  - Správce může nastavit, do jaké doby může zákazník ještě hřiště nebo cvičení stornovat.
  - Pokud je u cvičení vyčerpaná kapacita cvičení, lze se přihlásit jako náhradník.

- **Rezervace termínů:**
  V jedné rezervaci lze rezervovat jeden nebo více termínů.

- **Ceny:**
  Hřiště mohou mít cenová pásma a cena může mít i více kategorií, například cena pro dospělé, pro studenty, apod.

- **Způsob platby:**
  Rezervace lze hradit z kreditového účtu, permanentkou nebo hotově na místě.

Kapitola 2

Návrh architektury systému

V této kapitole definuju požadavky na systém, navrhnu datový model, rozhraní pro intrgraci a softwarovou architekturu systému. Struktura kapitoly je následující:

- 2.1 Požadavky na vyvíjený systém
- 2.2 Konceptuální datový model
- 2.3 Návrh rozhraní pro integraci
- 2.4 Návrh softwarové architektury
- 2.5 Nasazení systému

2.1 Požadavky na vyvíjený systém

2.1.1 Funkční požadavky

Systém umožní tyto hlavní funkce:

- Vyhledávat sportoviště podle aktivity, času a místa.
- Rezervovat sportoviště.
- Spravovat rezervace.
- Spravovat sportoviště přes webovou službu.

Systém pro jednoduchost umožní rezervovat sportoviště bez nutnosti placení. Neumožní tak hradit rezervaci pomocí platební karty, permanentky nebo kreditového účtu, jak je tomu u systémů sportovišť.

2.1.2 Požadavky na vlastnosti systému

Systém by měl mít tyto vlastnosti:

- Systém bude pro uživatele přístupný jako webová aplikace.
- Systém bude schopen obsluhovat několik uživatelů současně.
- Systém bude škálovatelný a bude připravený na případnou zvětšující se zátěž uživatelů.

2.1.3 Případy užití systému

Se systémem budou interagovat následující skupiny uživatelů:

- Nepřihlášený zákazník.
  Jakýkoliv uživatel, který navštíví webové stránky systému.
- Přihlášený zákazník.
  Uživatel přihlášený ke svému uživatelskému účtu.
- Správce sportoviště.
  Uživatel přihlášený do administrátorské sekce systému pro správu sportoviště.
- Systém sportoviště.
  Stávající rezervační systém sportoviště nebo systém pro jeho správu, který se bude do vyvíjeného systému integrovat přes webovou službu.

Nepřihlášený zákazník Systém pro uživatele umožní:

- UC01. Přihlásit se ke svému účtu
- UC02. Vyhledat sportoviště
- UC03. Zobrazit profil sportoviště
- UC04. Zobrazit rozvrh sportoviště

Přihlášený zákazník Systém pro uživatele umožní:

- UC11. Rezervovat sportoviště
- UC12. Zobrazit seznam rezervací
- UC13. Stornovat rezervaci
- UC14. Odhlásit se

Správce sportoviště Systém pro uživatele umožní:

- UC21. Přihlásit se
- UC22. Změnit profil sportoviště
- UC23. Zobrazit seznam služeb.
- UC24. Přidat službu.
- UC25. Změnit službu.
- UC26. Odhlásit se

2.1.4 Popis vybraných případů užití

UC02. Zákazník vyhledá sportoviště

1. Zákazník přejde na hlavní stranu, kde se nachází formulář pro hledání sportoviště.
2. Zákazník vyplní ve formuláři druh aktivity, místo a čas a klikne na tlačítko hledat sportoviště.
3. Systém vrátí seznam nalezených sportovišť. U položky v seznamu je název sportoviště, adresa a tlačítko pro zobrazení profilu sportoviště.

UC03. Zákazník zobrazí profil sportoviště

1. Zákazník vyhledá sportoviště a systém mu vrátí seznam nalezených sportovišť.
2. Zákazník v seznamu nalezených sportovišt klikne na tlačítko zobrazit profil sportoviště.
3. Systém vrátí stranu s profilem sportoviště, kde je popis sportoviště, kontaktní údaje (adresa, telefon, e-mail), provozní doba a formulář pro zobrazení rozvrhu sportoviště.

UC04. Zákazník zobrazí rozvrh sportoviště

1. Zákazník přejde na stranu s profilem sportoviště, kde se nachází formulář pro zobrazení rozvrhu.
2. Zákazník vyplní ve formuláři druh aktivity a čas a klikne na tlačítko zobrazit rozvrh.
3. Systém vrátí seznam nalezených termínů. U položky v seznamu je název hřiště nebo cvičení, čas, cena a tlačítko pro rezervaci.

UC11. Zákazník rezervuje sportoviště

1. Přihlášený zákazník si zobrazí rozvrh sportoviště a systém mu vrátí seznam termínů.

2. Zákazník klikne na tlačítko rezervovat termín.

3. Systém vrátí stranu pro potvrzení rezervace, kde jsou shrnující údaje o termínu a formulář pro poznámku a potvrzení rezervace.

4. Zákazník klikne na tlačítko potvrdit rezervaci.

5. Systém provede rezervaci a vrátí stranu oznamující úspěšné provedení rezervace.

UC13. Zákazník stornuje rezervaci

1. Zákazník se příhlásí a na hlavní straně v pravém horním rohu klikne na odkaz přehled rezervací.

2. Systém vratí seznam provedených rezervací. U položky v seznamu je název sportoviště, název hřiště nebo cvičení, čas a tlačítko stornovat rezervaci.

3. Zákazník klikne na tlačítko stornovat rezervaci.

4. Systém provede storno rezervace a zobrazí hlášku oznamující úspěšné provedení storna rezervace.

UC24. Správce přidá službu

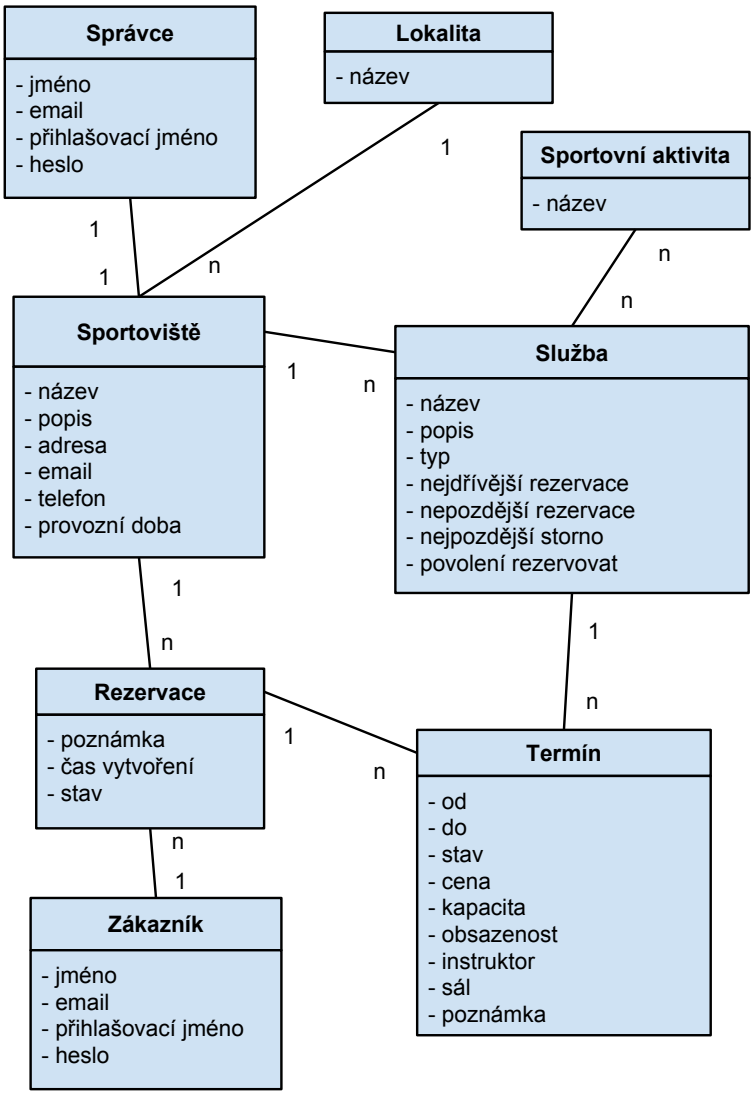
1. Správce přejde na stranu s formulářem pro přidání služby.

2. Správce vyplní typ (hřiště nebo cvičení), název, druh aktivity a parametry pro omezení rezervace a storna.

3. Systém vytvoří službu a přesměruje správce na stranu se seznamem služeb.

2.2 Konceptuální datový model

V této sekci je popsán konceptuální datový model systému pomocí diagramu tříd, který je na obrázku 2.1



Obrázek 2.1: Diagram tříd konceptuálního datového modelu systému.

2.2.1 Třída sportoviště

Třída modeluje sportoviště (sportovní centrum, sportovní hala, fitness klub apod.). Třída obsahuje atributy:

- *Název*
- *Popis*
- *Adresa*
- *E-mail*
- *Telefon*
- *Provozní doba*

2.2.2 Třída Služba

Každá instance této třídy bude modelovat prostor pro hraní (hřiště, kurt apod.) nebo skupinové cvičení. Atributy třídy jsou

- *Název* - Označení hříště nebo druhu cvičení.
- *Popis*
- *Typ* - Rozlišení, zda se jedná o hřiště nebo cvičení.
- *Popis*
- *Nejdřívější rezervace* - Čas před začátkem termínu, od kdy může zákazník termín rezervovat.
- *Nepozdější rezervace* - Čas před začátkem termínu, od kdy už nemůže zákazník termín rezervovat.
- *Nejpozdější storno* - Čas před začátkem termínu, od kdy už nemůže zákazník termín rezervovat.
- *Povolení rezervovat* - Službu půjde rezervovat v případě, že je to bude povolené.

2.2.3 Třída Termín

Každá instance této třídy bude modelovat termín služby. Atributy třídy jsou

- *Od* - Čas začátku termínu.
- *Do* - Čas konce termínu.
- *Stav* - Stav termínu může být: *dostupný*, *zrušený*.

- *Cena* - Cena, kterou zákazník zaplatí na místě. Pro jednoduchost nebudu uvažovat cenové kategorie.
- *Kapacita* - Maximální počet míst.
- *Obsazenost* - Aktuální počet volných míst.
- *Instruktor* - Jméno instruktora cvičení.
- *Sál* - Název místa, kde se cvičení koná.
- *Poznámka* - Doplňující informace týkající se daného termínu.

2.2.4 Třída Rezervace

Třída modeluje rezervaci zákazníkem. Atributy třídy jsou

- *Čas vytvoření* - Čas, kdy zákazník provedl rezervaci.
- *Poznámka* - Doplňující informace od zákazníka.
- *Stav* - Stav rezervace. Rezervace může mít stav: *rezervováno* anebo *stornováno*.

2.2.5 Třída Zákazník a třída Správce

Třídy modelují uživatele systému. Atributy třídy jsou

- *Jméno*
- *E-mail*
- *Přihlašovací jméno*
- *Heslo*

2.2.6 Třida Lokalita a třída Sportovní aktivita

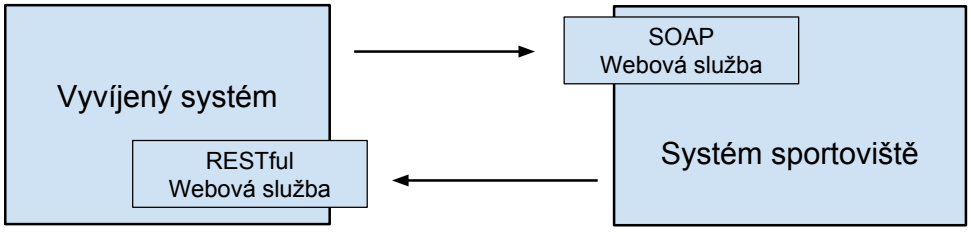
Třídy jsou v modelu za účelem filtrování sportovišť a služeb podle místa a druhu aktivity.

2.3 Návrh rozhraní pro integraci

Pří návrhu jsem se inspiroval principem integrace do rezervačního systému Expedia popsaného v sekci 1.2.3. K integraci bude potřeba navrhnout dvě rozhraní webových služeb:

- rozhraní vyvíjeného rezervačního systému

- rozhraní systému sportoviště



Obrázek 2.2: Schéma integrace systému sportoviště a vyvíjeného systému

V oblasti webových služeb existují dva typy:

- SOAP služby.
  Orientují na zprávy a ke komunikaci používají protokol SOAP. Taková služba se popisuje pomocí jazyka WSDL.
- REST služby.
  Orientují na zdroje představující entity datového modelu. Služba se implementuje pomocí protokolu HTTP a její popis je většinou v textový.

Z cvičných důvodů chci vyzkoušet implementovat oba způsoby. Webovou službu vyvíjeného systému jsem se rozhodl implementovat jako RESTful službu a popis jejího rozhraní je níže v sekci 2.3.1. Webová služba systému sportoviště bude používat SOAP protokol. Rozhraní této služby jsem popsal v sekci 2.3.2.

2.3.1 Popis RESTful služby vyvíjeného systému

Turo službu bude poskytovat vyvíjený systém a klientem služby bude systém sportoviště. Služba bude postavená na architektuře REST.

2.3.1.1 Autentizace

K autentizaci klienta ke službě použiju protokol OAuth 2.0. Tento protokol není jediným možným řešením, autentizovat by šlo například pomocí HTTP autentizace. K implementaci protokolu OAuth 2.0 bude zapotřebí vytvořit koncový bod pro autorizaci a pro získání *access tokenu*.

2.3.1.2 Identifikace zdrojů

Webová služba bude mít následující zdroje a URI:

- Sportoviště
  `/sportoviste`

- Seznam služeb
  `/sportoviste/{sportoviste-id}/sluzby`
- Služba s id `{sluzba-id}`
  `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}`
- Seznam termínů
  `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy`
- Termín s id `{termin-id}`
  `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy/{termin-id}`
- Seznam rezervací
  `/sportoviste/{sportoviste-id}/rezervace`
- Rezervace s id `{rezervace-id}`
  `/sportoviste/{sportoviste-id}/rezervace/{rezervace-id}`

2.3.1.3 Popis operací

Služba bude poskytovat následující operace:

- Vrátit detail sportoviště
- Vrátit seznam služeb, detail služby
- Změnit službu
- Vrátit seznam termínů, detail termínu
- Vytvořit, smazat termín, přesunout termín, zrušit termín, změnit termín
- Vrátit seznam rezervací, detail rezervace
- Stornovat rezervaci, přesunout rezervaci

Službou označuju hřiště nebo cvičení. Každá operace požaduje povinný parametr *access_token* sloužící pro autentizaci sportoviště. V případě že tento parametr bude chybný, vrátí služba chybovou odpověď s kódem 401 Unauthorized. Pokud klient pošle požadavek na neexistující zdroj, služba vrátí chybovou odpověď s kódem 404 Not found. Pokud klient pošle chybný požadavek na existující zdroj, služba vrátí chybovou odpověď s kódem 400 Bad request.

Níže jsou detailněji popsány jednotlivé operace, jejich možné kódy odpovědí a příklady reprezentace zdroje ve formátu JSON.

Operace Vrátit detail sportoviště

- Zdroj: `/sportoviste`
- HTTP metoda: GET
- Kód odpovědi: 200
- Kód chybové odpovědi: 401
- Příklad vrácených dat:

```
{ id: 1, nazev: "Sportcentrum Radlice" }
```

Operace Vrátit seznam služeb

- Zdroj: `/sportoviste/{sportoviste-id}/sluzby`
- HTTP metoda: GET
- Nepovinné parametry: offset, limit
- Kód odpovědi: 200
- Kód chybové odpovědi: 401, 404
- Příklad vrácených dat:

```
[ { id: 1, nazev: "Sportovni hala" },
  { id: 2, nazev: "Joga" }, ... ]
```

Operace Vrátit detail služby

- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}`
- HTTP metoda: GET
- Kód odpovědi: 200
- Kód chybové odpovědi: 401, 404
- Příklad vrácených dat:

```
{
 id: 2,
 nazev: "Joga",
 nejpozdejsi-rezervace: 5, // 5 minut pred zacatkem terminu
 nejdrivejsi-rezerace: 14,  // 14 dni pred zacatkem terminu
 nejpozdejsi-storno: 30,  // 30 minut pred zacatkem terminu
 povoleni-rezervovat: true
}
```

Operace Změnit službu

- Popis: Pomocí této operace půjdou změnit atributy *nejpozdejsi-rezervace*, *nejdrivejsi-rezerace*, *nejpozdejsi-storno*, *povoleni-rezervovat*
- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}`
- HTTP metoda: PUT
- Kód odpovědi: 204
- Kódy chybových odpovědi: 401, 400, 404

Operace Vrátit seznam termínů

- Popis: Vrátí seznam termínů, které leží v intervalu zadaného pomocí parametrů *start* a *end*
- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy`
- HTTP metoda: GET
- Povinné parametry: *start*, *end*
- Kód odpovědi: 200,
- Kód chybové odpovědi: 401, 400, 404
- Příklad vrácených dat:

```
[ { id: 23, od: "3.11.2015 13:00", do: "3.11.2015 14:30" },
  { id: 24, od: "3.11.2015 14:30", do: "3.11.2015 15:00", ... ]
```

Operace Vytvořit termín

- Popis: Operace po vytvoření vrátí identifikátor vytvořeného termínu.
- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy`
- HTTP metoda: POST
- Kód odpovědi: 200
- Kód chybové odpovědi: 401, 400, 404
- Příklad posílaných dat:

```
{
 od: "4.11.2015 13:00",
 do: "4.11.2015 14:30"
 cena: "250 Kc",
 kapacita: 10,
 obsazeno-mist: 0,
}
```

Operace Vrátit detail termínu

- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy/{termin-id}`
- HTTP metoda: GET
- Kód odpovědi: 200
- Kód chybové odpovědi: 401, 404
- Příklad vrácených dat:

```
{
 id: 23,
 od: "3.11.2015 13:00",
 do: "3.11.2015 14:30"
 cena: "150 Kc",
 kapacita: 10,
 obsazeno-mist: 2,
}
```

Operace Přesunout termín, zrušit termín, změnit termín

- Popis: Pro zrušení termínu pošlu atribut *stav* nastavený na zrušeno. Pro přesunutí termínu na jiný čas pošlu atributy *od*, *do*. Dále půjde změnit atributy *cena*, *kapacita*, *obsazeno-mist*, *sal*, *instruktor* a *poznamka*.
- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy/{termin-id}`
- HTTP metoda: PUT
- Kód odpovědi: 204
- Kódy chybových odpovědi: 401, 400, 404

Operace Smazat termín

- Popis: Termín půjde smazat, pokud je zrušený.
- Zdroj: `/sportoviste/{sportoviste-id}/sluzby/{sluzba-id}/terminy/{termin-id}`
- HTTP metoda: DELETE
- Kód odpovědi: 204
- Kódy chybových odpovědi: 401, 400

Operace Vrátit seznam rezervací

- Zdroj: `/sportoviste/{sportoviste-id}/rezervace`
- HTTP metoda: GET
- Nepovinné parametry pro stránkování: *offset*, *limit*
- Nepovinné parametry pro filtrování: *stav*, *zakaznik-id*, *sluzba-id*
- Kód odpovědi: 200,
- Kód chybové odpovědi: 401, 400, 404
- Příklad vrácených dat:

```
[ { id: 193 }, ... ]
```

Operace Vrátit detail rezervace

- Zdroj: `/sportoviste/{sportoviste-id}/rezervace/{rezervace-id}`
- HTTP metoda: GET
- Kód odpovědi: 200
- Kód chybové odpovědi: 401, 404
- Příklad vrácených dat:

```
{
 id: 193,
 stav: "rezervovano",
 termin-id: 23
 zakaznik-id: 12,
 zakaznik_jmeno: "Jan Novak",
 zakaznik_email: "jan@novak.cz",
 zakaznik_telefon: "+420 775398432"
}
```

Operace Stornovat rezervaci, přesunout rezervaci

- Popis: Pro stornování rezervace pošlu atribut *stav* nastavený na stornováno. Pro přesunutí rezervace na jiný termín pošlu identifikátor nového termínů *termin-id*.
- Zdroj: `/sportoviste/{sportoviste-id}/rezervace/{rezervace-id}`
- HTTP metoda: PUT
- Kód odpovědi: 204
- Kódy chybových odpovědi: 401, 400, 404
- Příklad posílaných dat:

```
{ stav: "stornovano" }
```

2.3.2 Popis SOAP služby systému sportoviště

Tuto službu bude poskytovat systém sportoviště a klientem služby bude vyvíjený systém. Službu implementuju pomocí SOAP protokolu. Pro klienta bude služba popsána pomocí dokumentu WSDL. Služba bude poskytovat dvě operace

- Vytvořit rezervaci
- Stornovat rezervaci

2.3.2.1 Autentizace

Klient služby se bude autentizovat pomocí HTTP autentizace.

2.3.2.2 Operace - Vytvořit rezervaci

Parametry opearace budou: ID sportoviště, ID rezervace, ID služby, ID termínu, začátek termínu, konec termínu, ID zákazníka, jméno zákazníka, email zákazníka a telefon zákazníka. Výstupem bude informace, jestli byla rezervace v systému sportoviště vytvořena nebo ne. Pro tuto operaci v dokumentu WSDL definuji dvě zprávy pro požadavek a odpověď. Níže je příklad reprezentace obou zpráv ve formátu XML

```
<ns:VytvoritRezervaci xmlns:ns="...">
    <sportovisteId>1</sportovisteId>
    <rezervaceId>2</rezervaceId>
    <sluzbaId>2</sluzbaId>
    <terminId>3409</terminId>
```

```
    <terminOd>3.11.2015 13:00</terminOd>
    <terminDo>3.11.2015 14:00</terminDo>
    <zakaznikId>4</zakaznikId>
    <zakaznikJmeno>Jan Novak</zakaznikJmeno>
    <zakaznikEmail>jan@novak.cz</zakaznikEmail>
    <zakaznikTelefon>+420 775398432</zakaznikTelefon>
</ns:VytvoritRezervaci>

<ns:VytvoritRezervaciResponse xmlns:ns="...">
    <vytvoreno>true</vytvoreno>
</ns:VytvoritRezervaciResponse>
```

2.3.2.3 Operace - Stornovat rezervaci

Parametry operace budou: ID sportoviště, ID rezervace. Výstupem bude informace, jestli byla rezervace stornována nebo ne. Pro tuto operaci v dokumentu WSDL definuji dvě zprávy pro požadavek a odpověď. Níže je příklad reprezentace obou zpráv ve formátu XML

```
<ns:StornovatRezervaci xmlns:ns="...">
    <sportovisteId>1</sportovisteId>
    <rezervaceId>2</rezervaceId>
</ns:StornovatRezervaci>

<ns:StornovatRezervaciResponse xmlns:ns="...">
    <stornovano>true</stornovano>
</ns:StornovatRezervaciResponse>
```

2.4 Návrh softwarové architektury

Vyvíjený systém budu vyvíjet jako webovou aplikaci, která bude zahrnovat

- Sekci pro zákazníka
- Sekci pro správce sportoviště
- RESTful webovou službu pro systém sportoviště

Aplikace z pohledu typu softwarové architektury bude centralizovanou aplikací typu klient-server, kde klientem je systém sportoviště a internetový prohlížeč zákazníka a správce sportoviště. Dále bude aplikace založena na vícevrstvé architektuře obsahující tyto vrstvy:

- Datová vrstva.
  Vrstvu představuje databáze a je umístěna na straně serveru.

- Aplikační vrstva.
  Vrstva představuje logiku aplikace a může být umístěna na straně serveru nebo na straně klienta

- Prezentační vrstva.
  Vrstvu představuje uživatelské rozhraní aplikace. Vrstva je umístěná na straně klienta.

2.4.1 Volba technologií datovou vrstvu

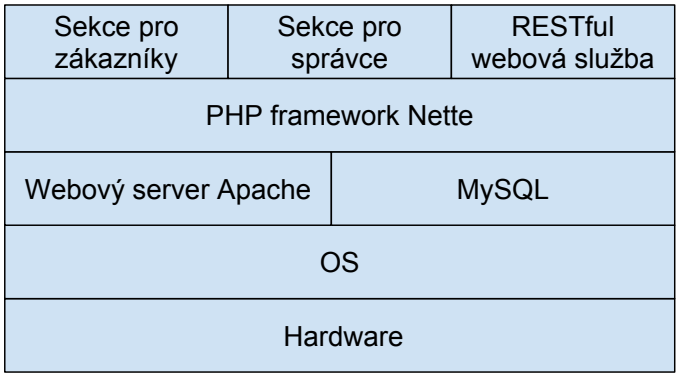
Pro uložení dat použiju relační databázi MySQL [29]. Důvodem volby je, že je databáze volně dostupná ke stažení a je podporovaná frameworkem, který chci použít.

2.4.2 Volba technologií pro aplikační vrstvu

Pro aplikace na straně serveru si zvolím framework Nette [30], který je psaný v jazyku PHP. Důvodem volby je, že jazyk a framework znám a používám. Pro vývoj webových aplikací lze použít i jiné programovací jazyky a technologie jako je např. Java, Node.js nebo Ruby. Aplikaci budu vyvíjet na webové serveru Apache.

2.4.3 Volba technologií pro prezentační vrstvu

Pro usnadnění tvorby uživatelského rozhraní a jeho vzhledu použiju framework Twitter Bootstrap [31].



Obrázek 2.3: Softwarová architektura vyvíjeného systému

[29] https://www.mysql.com/
[30] https://nette.org
[31] http://getbootstrap.com

2.5 Nasazení systému

Aby systém zvládal obsluhovat více klientů najednou a zároveň byl připraven na případnou zvětšující se zátěž, bude potřeba, aby byl systém dobře škálovatelný. Možným řešením je:

- Vytvořit vlastní infrastrukturu
- Využít služby IaaS nebo PaaS

2.5.1 Vlastní infrastruktura

Pro vypořádání se s výkonem systému existují dva hlavní způsoby škálování:

- Vertikální škálování - Zvyšuje se výkon jednoho fyzického serveru přídáváním více zdrojů jako je CPU, paměť, síťové adaptéry, řadiče disků nebo disky.
- Horizontální škálování - Výkon systému se zvýší přidáváním dalších fyzických serverů.

Nejprve bude potřeba oddělit databazový server a umístit ho na samostaný fyzický server. Databázový server by šlo škálovat buď vertikálně anebo horizontálně využitím MySQL Cluster [32]. Pro běh samotné aplikace bych použil nějaký aplikační server jako např. Zend Server [33] nebo Appserver.iohttp://appserver.io, které umožňují běh aplikace v klastru. Toto řešení má nevýhodu vysokých počátešních nákladů a proto raději využiju služeb IaaS nebo PaaS.

2.5.2 Služby IaaS a PaaS

Protože dopředu nevím, kolik uživatelů bude systém používat, nabízí se řešení použít služeb *Infrastructure as s service* (IaaS) nebo *Platform as s service* (PaaS). Ve službě IaaS se manuálně konfiguruje celá infrastruktura, kdežto u služby PaaS se konfiguruje na úrovni běhového prostředí, databáze a webového serveru. Někteří poskytovatelé PaaS mají infrastrukturu postavenou tak, že umožňuje automatické škálování a rozložení zátěže. Příkladem jsou služby Google App Engine [34] nebo AWS Elastic Beanstalk [35], které umožňujé nasadit aplikace napsané v PHP. Vyvíjený systém nasadím na App Engine nebo Elastic Beanstalk.

[32] https://www.mysql.com/products/cluster/
[33] http://www.zend.com/en/products/zend_server
[34] https://cloud.google.com/appengine/
[35] https://aws.amazon.com/documentation/elastic-beanstalk/

Kapitola 3

Implementace

V této kapitole popíšu implementaci rezervačního systému, který jsem vyvíjel jako webovou aplikaci. Aplikace se skládá z následujících sekcí

- Sekce pro zákazníky
- Sekce pro správce sportoviště
- RESTful webová služba

Struktura kapitoly je následující:

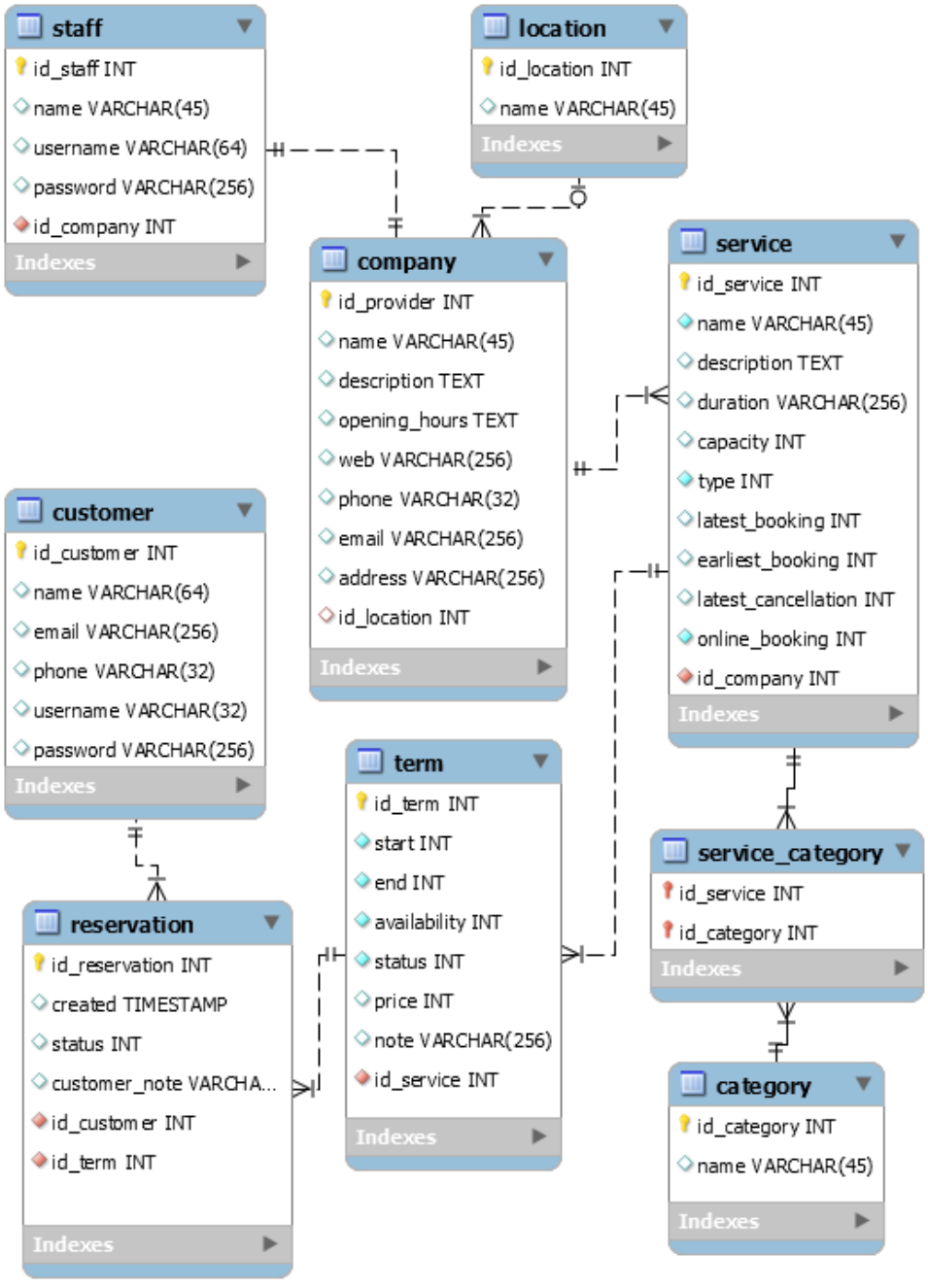
- 3.1 Struktura aplikace
- 3.2 Sekce pro zákazníky
- 3.3 Sekce pro správce
- 3.4 RESTful Webová služba
- 3.5 Vývojové prostředí
- 3.6 Nasazeni aplikace

3.1 Struktura aplikace

Pro vývoj aplikace jsem zvolil webový aplikační framework Nette, který je volně ke stažení pod licencí GNU GPL. Framework je objektově orientovaný a je napsaný v jazyku PHP.

3.1.1 Datová vrstva

Pro uložení dat jsem použil relační databázový systém MySQL. Před použitím jsem nejprve navrhl ER model (viz obrázek 3.1) podle datového modelu popsaného v sekci 2.2 na straně 18. ER model jsem následně převedl na jednotlivé tabulky v databázi. Jako formát uložiště dat jsem použil formát InnoDB.



Obrázek 3.1: ER model datového modelu

3.1.2 Aplikační vrstva a prezentační vrstva

Při vytváření aplikační logiky jsem využil návrhového vzoru Facade a vytvořil třídu *Facade*, která poskytuje jednotné rozhraní pro vyšší vrstvu. Třída fasády má v aplikaci jedinou instanci, což jsem zařídil pomocí návrhového vzoru Singleton. Vytváření uživatelského rozhraní ve frameworku Nette je založeno na návrhovém vzoru Model-view-presenter (MVP). Presentery v aplikaci zpracovávají požadavky od uživatele a definují se v nich akce. Aby se určilo, který požadavek má zpracovat který presenter, musí se tato informace nakonfigutovat pomocí třídy *RouteList*. Pro generování obsahu stránek framework používá šablonovací systém Latte. Pro vytváření uživatelského prostředí jsem použil framework Twitter Bootstrap.

Jednotlivé sekce aplikace jsem ve frameworku vytvořil pomocí modulů. Celkem má aplikace tyto moduly

- *FrontendModule* - sekce pro zákazníky
- *BackendModule* - sekce pro správce sportoviště
- *WebServiceModule* - RESTful webová služba

3.2 Sekce pro zákazníky

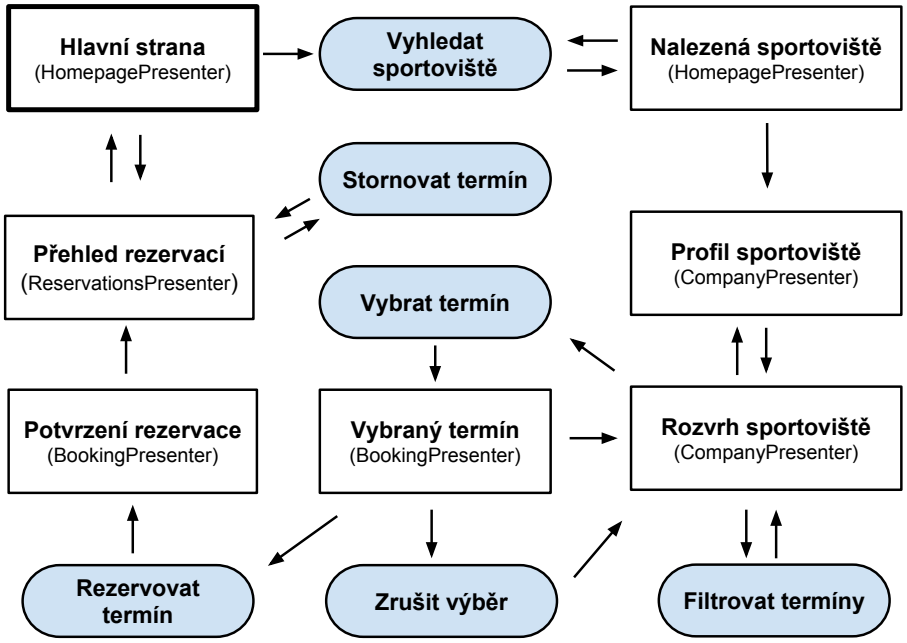
Tuto sekci webové aplikace jsem ve frameworku Nette implementoval jako modul *FrontendModule*. Sekce je obsluhována pomocí těchto presenterů:

- *HomepagePresenter* - Obsluhuje hlavní stranu s formulářem pro hledání sportovišť.
- *SignPresenter* - Obsluhuje přihlášení zákazníka.
- *CompanyPresenter* - Obsluhuje stranu s profilem s profilem sporotiště a jeho rozvrhem.
- *BookingPresenter* - Obsluhuje rezervaci termínu.
- *ReservationsPresenter* - Obsluhuje stranu s přehledem rezervací.

Struktura uživatelského rozhraní včetně vyznačení jednotlivých presenterů je na obrázku 3.2.

3.2.1 Hledání sportoviště

Hledání sportoviště se provádí pomocí formuláře umístěného na hlavní straně. Uživatel zadá druh aktivity, místo, datum a čas a klikne na tlačítko Vyhledat sportoviště. Zpracování formuláře provádí presenter *HomepagePresenter*. Presenter ve třídě *Facade* zavolá metodu *findCompanies*, která vyhledá sportoviště pomocí tohoto SQL dotazu:



Obrázek 3.2: Struktura uživatelského rozhraní sekce pro zákazníky

```
SELECT com.* FROM company AS com
JOIN service AS se ON (se.id_company = com.id_company)
JOIN service_category AS sec ON (sec.id_service = se.id_service)
JOIN term AS te ON (te.id_service = se.id_service)
WHERE com.id_location = {form.location} AND
        sec.id_category = {form.category} AND
        te.start BETWEEN {form.from} AND {form.to}
GROUP BY com.id_company;
```

3.2.2 Profil sportoviště a výpis termínů

Na stránce profilu sportoviště se nachází informace o sportovišti (popis, adresa, e-mail, telefon, webové stránky a provozní doba) a formulář pro výpis termínů. Ve formuláři uživatel zadá druh aktivity, datum a klikne na tlačítko Vypsat termíny. Poté se mu zobrazí jednotlivé termíny seskupené podle hřiště nebo druhu cvičení. U termínu se nachází informace o času, ceně a tlačítko Rezervovat termín.

Zpracování formuláře provádí presenter *CompanyPresenter*. Presenter ve třídě *Facade* zavolá metodu *findTerms*, která vyhledá termíny pomocí tohoto SQL dotazu:

```
SELECT te.* FROM term AS te
JOIN service AS se ON (se.id_service = te.id_service)
JOIN service_category AS sec ON (sec.id_service = se.id_service)
WHERE se.id_company = {form.company} AND
        sec.id_category = {form.category} AND
        te.start BETWEEN {form.from} AND {form.to};
```

Následně se každý termín zkontroluje, jestli lze rezervovat. Pro zkontrolování slouží pomocná funkce *canBookTerm*.

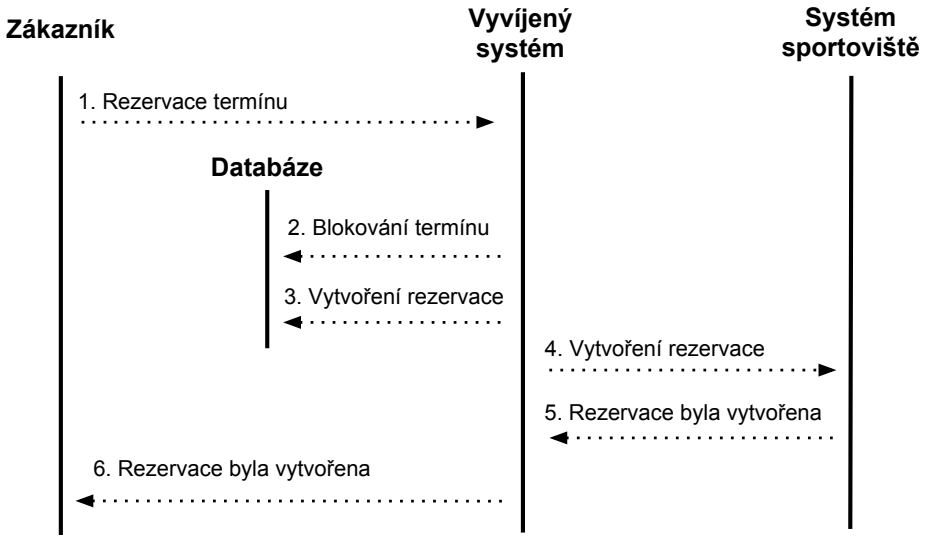
```
function canBookTerm(term, service)
    tn = time(); //cas nyni
    if  service.online_booking | tern.status == 'cancelled'
        return false;
    elseif term.availability < 1
        return false; //obsazen
    elseif term.start <= tn
        return false; //jiz probiha nebo probehl
    elseif (term.start - service.earliestBooking) > tn
        return false; //prilis brzo
    elseif (term.start - service.latestBooking) < tn
        return false; //prilis pozde
    else
        return true;
```

3.2.3 Rezervace termínu

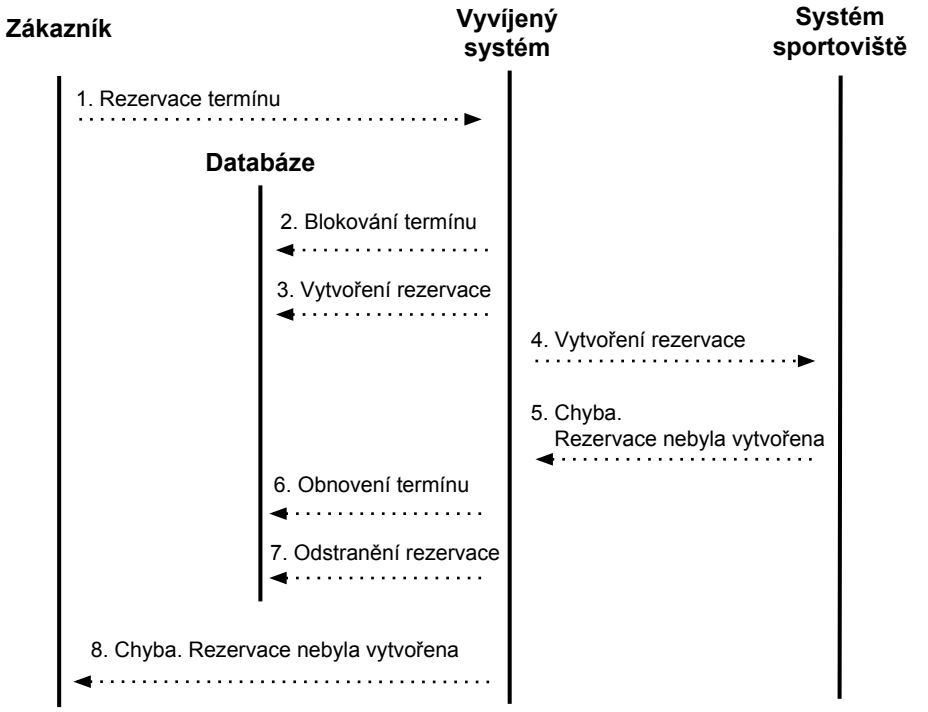
Aby zákazník mohl provést rezervaci, musí si nejprve nechat vypsat termíny pomocí formuláře na straně s profilem sportoviště a poté vybrat termín a kliknout na tlačítko Rezervovat termín. Následně se mu zobrazí stránka, kde jsoun informace o vybraném termínu a formulář pro potvrzení rezervace. Ve formuláři uživatel může zadat poznámku k rezervaci a poté klikne na tlačítko Potvrdit rezervaci. Pokud rezervace proběhne úspěšně, zobrazí se mu hláška o úspěšném provedení.

Provedení rezervace provádí presenter *BookingPresenter*, který ve třídě *Facade* zavolá metodu *bookTerm*. Metoda *bookTerm* nejprve zablokuje termín pomocí metody *blockTerm*, poté vytvoří rezervaci v databázi a následně zavolá operaci *CreateReservation* na SOAP službě, kterou poskytuje systém sportoviště. Proces je znázorněn na obrázku 3.3 a na obrázku 3.4.

Níže je ukázka metody *blockTerm* pomocí pseudokódu. Pokud rezervovaný termín překrývá jiné termíny, budou tyto termíny také blokovány.



Obrázek 3.3: Proces úspěšné rezervace termínu zákazníkem



Obrázek 3.4: Proces rezervace při zamítnutí sportovištěm.

```
function blockTerm(term)
    if term.type == TYPE_EXERCISE //typ cviceni
        term.availability--;
        update(term);
    else //typ hriste
        term.availability = 0;
        update(term);
        ovTerms = getOverlappedTerms(term);
        foreach ovTerms as ovTerm
            ovTerm.availability = 0;
            update(ovTerm);
```

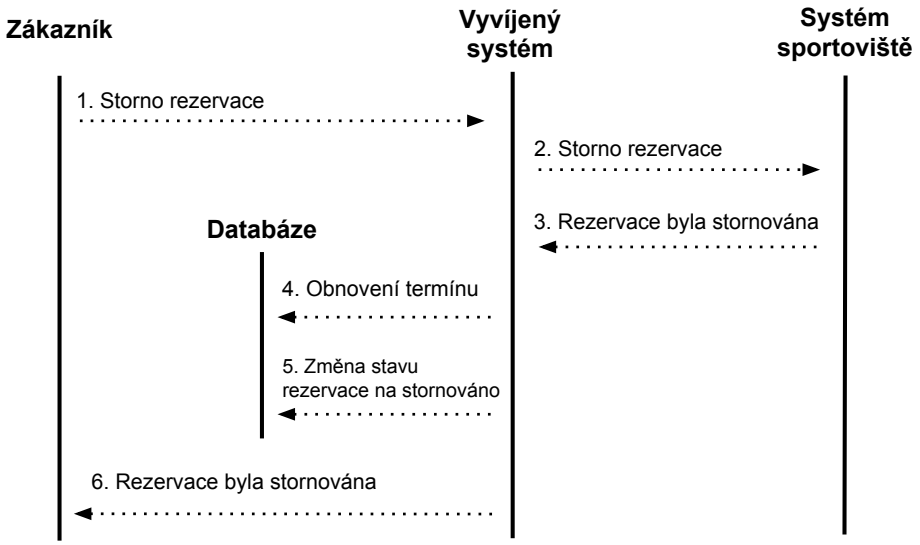
Pomocná funkce *getOverlappedTerms* vrátí pomocí následujicího SQL dotazu pro zadaný termín termíny, se kterými se zadaný termín překrývá.

```
SELECT * FROM term
WHERE id_service = {term.service} AND
        id_term != {term.id} AND
        ({term.start} <= start AND start < {term.end}) OR
        (start <= {term.start} AND {term.start} < end)
```

3.2.4 Přehled rezervací a storno rezervace

Po přihlášení se může zákazník podívat na své rezervace kliknutím na odkaz Moje rezervace v pravém horním rohu. U jednotlivé položky seznamu rezervací, které jsou seskupené podle sportovišť, se nachází informace o času, místu, druhu služby a ceně termínu. Pokud lze rezervaci stornovat, zobrazí se u rezervace tlačítko, kterým může zákazník rezervaci stornovat.

Zobrazení rezervací a provední storna provádí presenter *ReservationsPresenter*. Informaci, zda lze rezervaci stornovat, vrátí pomocná funkce *canBookTerm*.

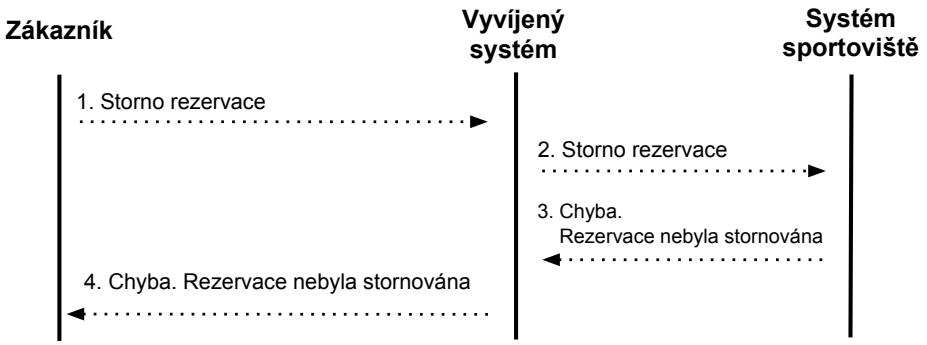


Obrázek 3.5: Proces storna rezervace zákazníkem

```
function canCancelReservation(reservation, service)
    tn = time();
    if reservation.status == CANCELLED
        return false; //jiz stornovano
    elseif reservation.start <= tn
        return false; //termin jiz probiha nebo probehl
    elseif (reservation.start - service.latest_cancellation) < tn
        return false; //prilis pozde
    else
        return true;
```

Pokud zákazník rezervaci stornuje, zavolá presenter metodu *cancelReservation* ve třídě *Facade*. Metoda *cancelReservation* nejprve zavolá operaci *CancelReservation* na SOAP službě, kterou poskytuje systém sportoviště. Poté obnoví termín pomocí metody *recoverTerm* a změní stav rezervace na stornováno. Proces je znázorněn na obrázku 3.5 a na obrázku 3.6.

Níže je ukázka metody *recoverTerm* pomocí pseudokódu. Pokud rezervovaný termín překrývá jiné termíny, budou tyto termíny také obnoveny.



Obrázek 3.6: Proces storna při zamítnutí sportovištěm.

```
function recoverTerm(term)
    if term.type == TYPE_EXERCISE //typ cviceni
        term.availability++;
        update(term);
    else //typ hriste
        term.availability = 1;
        update(term);
        recoverOverlappedTerms(term);

function recoverOverlappedTerms(term)
    ovTerms = getOverlappedTerms(term);
    foreach ovTerms as ovTerm
        recoverTerm = true;
        ovTerms2 = getOverlappedTerms(ovTerm);
        foreach ovTerms2 as ovTerm2
            if ovTerm2.availability == 0 & !ovTerms.contains(ovTerm2)
                recoverTerm = false;
        if recoverTerm
            ovTerm.availability = 1;
            update(ovTerm);
```

3.3 Sekce pro správce

Tuto sekci webové aplikace jsem ve frameworku Nette implementoval jako modul *BackendModule*. Sekce je obsluhována pomocí těchto presenterů:

- *HomepagePresenter* - Obsluhuje hlavní stranu.
- *SignPresenter* - Obsluhuje přihlášení správce sportoviště.

- *CompanyPresenter* - Obsluhuje stranu s editací profilu sporotiště.
- *ServicesPresenter* - Obsluhuje správu hřišť a cvičení.

V modulu je také implementován autorizační server pro protokol OAuth 2.0, který je blíže popsán v sekci 3.4.1 na straně 44.

3.3.1 Editace profilu sportoviště

Editaci profilu sportoviště obsluhuje presenter *CompanyPresenter*. Editaci správce provádí pomocí formuláře, ve kterém zadá název, popis, adresu, telefon, e-mail a provozní dobu.

3.3.2 Správa služeb

Správu hřišť a cvičení obsluhuje presenter *ServicesPresenter*. Presenter umožňuje vypsat seznam služeb, vytvořit novou službu a editovat službu. Službou označuju hřiště nebo cvičení. Vytváření a editaci služby správce provádí pomocí formuláře, ve kterém zadá název, popis, typ služby (cvičení nebo hřiště) a omezení (nejdřívější rezervace, nejpozdější rezervace, nejpozdější storno, povolení rezervace). Dále správce zařadí službu do kategorie (druh aktivity), aby zákazník mohl službu najít.

3.4 RESTful Webová služba

Službu jsem ve frameworku Nette implementoval jako modul *WebServiceModule*. Při implementaci jsem použil dopňek Nette REST API[36]. Jednotlivé zdroje služby jsem implementoval pomocí presenterů, které jsou potomkem presenteru *ResourcePresenter* z doplňku Nette REST API. Služba je obsluhována pomocí těchto presenterů

- *CompanyResourcePresenter* - Obsluhuje operace: *Vrátit detail sportoviště*
- *ServiceResourcePresenter* - Obsluhuje operace: *Vrátit seznam služeb*, *Vrátit detail služby*, *Změnit službu*. Službou označuju hřiště nebo cvičení.
- *TermResourcePresenter* - Obsluhuje operace: *Vrátit seznam termínů*, *Vrátit detail termínu*, *Vytvořit termín*, *Změnit termín*, *Smazat termín*,
- *ReservationResourcePresenter* - Obsluhuje operace: *Vrátit seznam rezervací*, *Vrátit detail rezervace*, *Stornovat rezervaci*

[36] https://github.com/drahak/Restful

URL adresy zdrojů jsem namapoval na jednotlivé presentery pomocí třídy *ResourceRoute*. Příklad namapování URL zdroje *Termín* je zde:

```
$url = '/api/v1/company/<company-id>'
  . '/services/<service-id>/terms/<term-id>';

$router[] = new ResourceRoute($url , array(
  'presenter' => 'TermResource',
  'action' => array(
    ResourceRoute::GET => 'readTerm',
    ResourceRoute::PUT => 'updateTerm',
    ResourceRoute::DELETE => 'deleteTerm')
));
```

Pro vykonání operace je potřeba poslat v adrese požadavku autentizační parametr access token, který se před vykonáním operace se ověřuje. Zde je zjednodušená ukázka kódu pro vykonání operace *Vrátit seznam služeb* v presenteru *ServiceResourcePresenter*:

```
public function actionReadServices() {

  if(!$this->isValidAccessToken($_GET['access_token'])) {
        $this->sendErrorResource("Unauthorized", 401);
        return;
  }

  $filter = array('id_company' => $this->id_company);
  $services = \Facade::getInstance()->getServicesWhere($filter);
  $data = array();

  foreach($services as $service) {
    $data[] = array(
      'id_service' => $service->id_service,
      'name' => $service->name
    );
  }

  $this->resource->data = $data;
  $this->sendResource(IResource::JSON);
}
```

3.4.0.1 Přesunutí a zrušení termínu

Obsluhu operace změny a smazání termínu obsluhuje presenter *TermResourcePresenter*. U termínu lze pomocí metody HTTP PUT změnit

- *Čas termínu* - Změnit čas termínu lze pouze u cvičení. Nový čas se pošle v atributech *od* a *do*. Změna času proběhne v pořádku, pokud termín v novém čase nepřekrývá jiný termín.
- *Stav termínu* - Změnit stav na zrušeno. Poté, co se změní stav, už nejde stav měnit. U rezervací, které se vážou k tomuto termínu, se změní stav na stornováno. Dostupnost termínu se neobnovuje. Poté, co je termín zrušený, je možné ho smazat pomocí metody HTTP DELETE.
- Atributy *dostupnost* a *cena*

3.4.0.2 Stornování rezervace

Stornování rezervace obsluhuje presenter *ReservationResourcePresenter*. Tato operace se provede pomocí HTTP medoty PUT posláním atributu stav nastaveného na stornováno. Při této operaci se také obnoví dostupnost rezervovaného termínu.

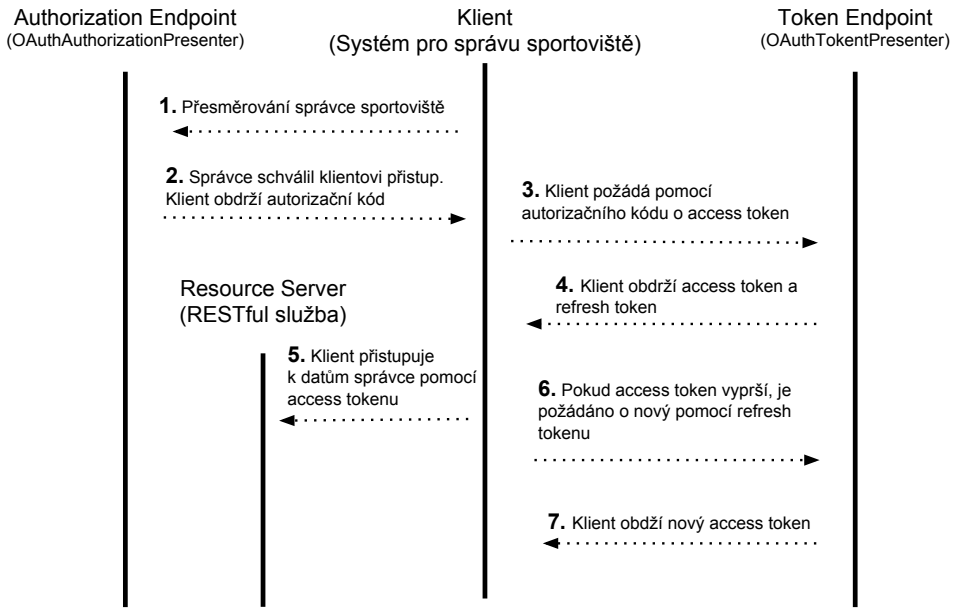
3.4.1 Implementace OAuth 2.0 protokolu

Protokolu OAuth 2.0 se účastní následující entity

- *Uživatel* - Správce, který upravuje data sportoviště ve vyvíjeném systému.
- *Klient* - Aplikace, která chce přistupovat k datům sportoviště ve vyvíjeném systému)
- *Resource server* - RESTful webová služba, která zpřístupňuje klientům data.
- *Authorization endpoint* - Zde se uživatel rozhodne, jestli klientovi povolí přístup k datům. Pokud uživatel povolí klientovi přístup, klient obdrží autorizační kód.
- *Token endpoint* - Zde klient žádá o access token, kterým se autentizuje při přístupu k webové službě.

Princip fungování protokolu je znázorněno na obrázku. Koncový bod pro autorizaci a koncový bod pro získání access tokenu jsem implementoval v modulu *BackendModule* pomocí presenterů:

- *OAuthAuthorizationPresenter* - Zastupuje roli koncového bodu pro autorizaci



Obrázek 3.7: Protokol OAuth 2.0

- *OAuthTokentPresenter* - Zastupuje roli koncového bodu pro získání access tokenu

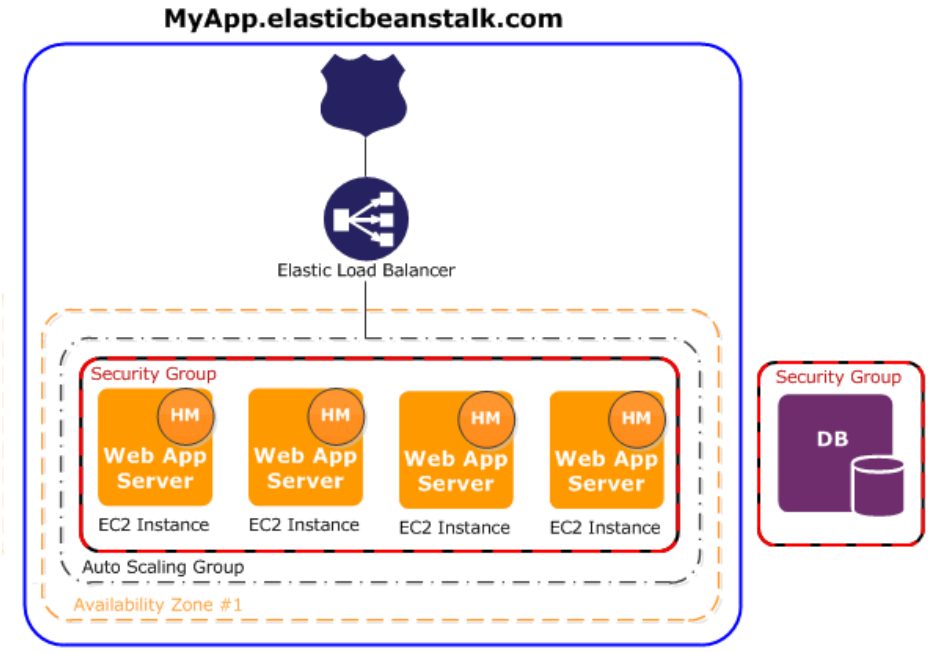
Při implementaci jsem použil knihovnu *OAuth 2.0 Server PHP* [37]. Knihovna vyžadovala vytvoření pěti tabulek v databázi:

- *oauth_access_tokens*
- *oauth_authorization_codes*
- *oauth_clients*
- *oauth_refresh_tokens*
- *oauth_scopes*

3.5 Vývojové prostředí

Pro spuštění a průběžné testování aplikace jsem použil program WampServer [38], který mi v operačním systému Windows nainstaloval webový server Apache, jazyk PHP a MySQL databázi. Aby aplikace běžela v pořádku, bylo potřeba na webovém serveru povolit modul *mod_rewrite*.

[37] https://github.com/bshaffer/oauth2-server-php
[38] http://www.wampserver.com/



Obrázek 3.8: Schéma prostředí služby AWS Elastic Beanstalk

Pro vývoj zdrojového kódu aplikace jsem používal program NetBeans IDE a verzovací systém Git a webou službu Bitbucket [39]. Pro vytvoření ER modelu jsem použil program MySQL Workbench [40].

3.6 Nasazeni aplikace

Vyvíjenou aplikaci jsem se nejprve pokusil nasadit na službu Google App Engine, kde jsem narazil na problém se souborovým systém, který je ve službě pouze pro čtení. Framework Nette ale vyžaduje do souborového systému zapisovat kvůli kešování. Aplikaci jsem proto nasadil na AWS Elastic Beanstalk, kde tento problém nebyl.

Služba Amazon Web Services [41] nabízí zdarma roční využívání jejich služeb v rámci programu AWS Free Usage Tier. Abych program mohl využívat, musel jsem si u nich vytvořit účet. Po vytvoření jsem se přihlásil do konzole a nakonfiguroval prostředí pro běh aplikace ve službě Elastic Beanstalk. Prostředí používá jednu až čtyři instance aplikačního serveru a jednu instanci databázového serveru. Prostředí je znázorněno na obrázku 3.8.

[39] https://bitbucket.org
[40] https://www.mysql.com/products/workbench/
[41] https://aws.amazon.com/

Kapitola 4

Testování

Před samotným testováním jsem vytvořil testovací aplikaci, která hraje roli systému sportoviště. Poté jsem provedl tato testování

- Integrační testování
- Systémové testování
- Akceptační testování

Integrační a akceptační testování jsem provedl formou automatizovaných testů napsaných v testovacím frameworku Codeception [42] používající jazyk PHP. V rámci systémového testování jsem aplikaci nasadil na AWS Elastic Beanstalk a experimentoval se zátěží pomocí nástroje Apache JMeter [43].

4.1 Popis testovací aplikace

Aplikaci jsem implementoval jako webovou a stejně jako vyvíjený systém v jazyku PHP ve frameworku Nette. Datovou vrstvu aplikace představuje RESTful služba vyvíjeného systému. V aplikaci je implementováno

- SOAP služba pro vyvíjený systém, který bude v roli klienta
- Protokol OAuth 2.0 pro roli klienta k RESTful službě vyvíjeného systému
- Uživatelské rozhraní pro správce sportoviště

[42] http://codeception.com/
[43] http://jmeter.apache.org/

4.1.1 SOAP služba

Službu jsem implementoval pomocí vestavěné třídy *SoapServer* v PHP. Té v konstruktoru předávám odkaz na popis služby v dokumentu WSDL, který je v aplikaci umístěn v adresáři *www*. Tento dokument je též přístupný pro vyvíjený systém v případě potřeby se službou komunikovat. Služba poskytuje operace pro vytvoření rezervace a stornování rezervace. Detailnější popis rozhraní služby je popsán v sekci 2.3.2 na straně 28.

4.1.2 Popis implementace protokolu OAuth 2.0

Ke komunikaci tímto protokolem jsem použil knihovnu OAuth 2.0 Client [44]. Před použitím bylo potřeba, aby testovací aplikace byla zaregistrována u vyvíjeného systému, na kterém běží autorizační server a RESTful služba. Registraci jsem provedl manuálně vložením záznamu do databáze vyvíjeného systému (tabukla `oauth_clients`). Dále bylo potřeba u knihovny nastavit tyto parametry:

- *clientId* - Identifikátor testovací aplikace
- *clientSecret* - Tajný kód
- *redirectUri* - Adresa, kam se má přesměrovat uživatel po autorizaci.
- *authorizationUrl* - Adresa autorizačního serveru
- *tokenUrl* - Adresa pro získání access tokenu

4.2 Popis integračního testování

Při tomto testování jsem formou automatizovaných testů ve frameworku Codeception otestoval:

- RESTful službu vyvíjeného systému
- SOAP službu testovací aplikace

4.2.1 Popis testování RESTful služby

Při testování jsem použil ve frameworku Codeception modul REST a modul Db pro vkládání testovacích dat a jejich kontrolu. U služby jsem otestoval tyto operace

- Vrácení dat sportoviště
- Vrácení seznamu služeb, detailu služby, změna omezení služby

[44] https://github.com/thephpleague/oauth2-client/

- Vrácení seznamu termínů, detailu termínu, změna času termínu, zrušení termínu, změna dostupnosti termínu
- Vrácení seznamu rezervací, detailu rezervace, stornování rezervace

Vytvořené testy mají tento sled kroků

1. Inicializace, při které se vytvoří v databázi testovací data.
2. Zavolání operace služby
3. Kontrola kódu odpovědi a vracených dat a dat v databázi.

4.2.2 Popis testování SOAP služby

Při testování jsem použil ve frameworku Codeception modul SOAP určený pro testování tohoto typu služby. U služby jsem otestoval operaci pro vytvoření rezervace a operaci pro stornování rezervace. V testu jsem zkontroloval, jestli vrací správné odpovědi.

4.2.3 Vyhodnocení integračního testování

Spuštění vytvořených testů proběhlo v pořádku.

4.3 Popis akceptačního testování

Při tomto testování jsem též formou automatizovaných testů ve frameworku Codeception otestoval funkční požadavky vyvíjeného systému. Testy jsem vytvořil jsem pro následující scénáře:

- Zákazník vyhledá sportoviště
- Zákazník si nechá vypsat termíny služeb
- Zákazník se přihlásí
- Zákazník rezervuje termín
- Zákazníkovi se zobrazí provedené rezervace v přehledu rezervací
- Zákazník stornuje rezervaci

V testovacím scénaři pro rezervaci termínu jsem také otestoval, jestli se správně zablokují termíny, které se překrývají. Podobně ve scénáři pro stornování rezervace jsem kontroloval, jestli se správně obnoví obsazenost termínů.

4.3.1 Vyhodnocení akceptačních testů

Spuštění vytvořených testů proběhlo v pořádku.

4.3.2 Zátěžový test

V rámci systémového testování jsem provedl zázěžový test s nástrojem Apache Jmeter. Nejprve jsem si pro aplikaci nageneroval data. Data obsahovala: 30 000 zákaznických účtů, 1 000 sportovišť a správců, 7 800 služeb (každé sportoviště mělo 1 až 10 služeb), 1 500 000 termínů a 320 000 rezervací.

V nástroji JMeter jsem vytvořil testovací plán s dvěma vlákny. Jedno vlákno obsahovalo kroky simulujicí zákazníka a druhé vlákno simulovalo klienta RESTful služby.

4.3.2.1 Popis testovacího plánu

V jedné iteraci vlákna zákazníka se provádí tyto kroky:

1. Zobrazení hlavní strany.
2. Zobrazení strany pro příhlášení a následné příhlášení náhodným uživatelským účtem. Tuhle akci provedu v rámci vlákna pouze jednou.
3. Zobrazení hlavní strany a vyhledání sportoviště podle náhodně zvolených parametrů.
4. Pokud byla nalezena nějaká sportoviště, náhodně nějaké vyberu a zobrazím jeho profil. Pokud ne, skončím.
5. Vypsání termínů podle náhodně zvolených parametrů.
6. Pokud byl nalezen termín, termín vyberu a zarezervuju. Pokud ne, skončím.
7. Zobrazení seznamu rezervací zákazníka a stornování rezervovaného termínu.

Pro testování klienta jsem vyvtořil jednoduchý skript, který vrátí access token podle poslaného parametru v adrese, kde parametrem je číslo vlákna. Pro stejné číslo mi vrací skript stejný access token. V jedné iteraci vlákna klienta se provádí tyto kroky:

1. Získání access tokenu. Tuto akci provedu v rámci vlákna pouze jednou.
2. Vrácení seznamu služeb.
3. Vrácení detailu náhodně vybrané služby.
4. Vrácení seznamu termínů služby podle náhodně zvoleného časového rozmenzí.
5. Vrácení detailu náhodně vybraného termínu.

6. Pokud není termín zcela obsazený, snížím obsazenost o jedno místo. Pokud termín nemá žádné volné místo, skončím.

7. Zvýšení obsazenosti o jedno místo

4.3.2.2 Průběh zátěžový testu

Před spuštěním testu jsem aplikaci měl nasazebou na službě Elastic Beanstalk, ve které používám jednu až čtyři instance aplikačního serveru a jednu instanci databázového serveru. Každá instance aplikačního severu byla typu t1.micro [45] a instance databázového serveru byla typu db.t1.micro [46]. Škálování jsem nastavil tak, aby se škálovalo podle toho, jaká je průměrná latence. Přidání nové instance webového serveru proběhne v případě, že je průměrná latence v intervalu 5 minut vyšší než 20s.

V nástroji Jmeter jsem nastavil, aby prodleva mezi kroky vlákna zákazníka byla v rozmezí 3 - 5 sekund. U vlákna klienta jsem nastavil prodlevu mezi kroky od 300 do 700 ms. Dále jsem nastavil aby každé vlákno provedlo 3 iterací a před startováním nového vlákna se počká 3 sekundy.

U testu mě zajímala průměrná doba odezvy systému. Test začínal při jedné puštěné instanci a testování jsem pustil pro

- 10 vláken zákazníka a 10 vláken klienta - průměrná odezva byla 500ms u obou vláken
- 20 vláken zákazníka a 20 vláken klienta - průměrná odezva u vlákna klienta byla 1200ms a 800ms u vlákna zákazníka.
- 40 vláken zákazníka a 40 vláken klienta - průměrná odezva u vlákna klienta byla 4000ms a 1600ms u vlákna zákazníka.

Při zvyšování počtu vláken začala průměrná odezva prudce stoupat a aplikace začala odpovídat chybovým kódem 504. Pro další experimentování už bohužel nezbyl čas.

[45] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts_micro_instances.html
[46] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html

Závěr

Výsledkem práce je webová aplikace, která zákazníkům umožňuje hledat sportoviště a rezervovat si u nich sportovní aktivity. Aplikaci tvoří sekce pro zákazníky, sekce pro správce sportoviště a webová služba pro integraci systému sportoviště.

Integrace je založena na dvou webových službách. Jedné, kterou poskytuje webová aplikace pro systém sportoviště a druhé, kterou poskytuje systém sportoviště pro webovou aplikaci. Při návrhu rozhraní webových služeb jsem se inspiroval způsobem integrace do rezervačního systému Expedia.

Pro webovou aplikaci jsem vytvořil integrační a akceptační testy, kterými aplikace prošla v pořádku. Poté jsem aplikaci nasadil na službu AWS Elastic Beanstalk a provedl experimentování se zátěží.

Literatura

[1] McKinnon, C.: *Linux, Apache, MySQL, PHP Performance End to End:*. Colin McKinnon, 2015, ISBN 9781311044747.

[2] Richardson, L.; Amundsen, M.; Ruby, S.: *RESTful Web APIs*. O'Reilly Media, 2013, ISBN 9781449359744.

[3] Golden, B.: *Amazon Web Services For Dummies*. –For dummies, Wiley, 2013, ISBN 9781118651988.

[4] Felix, K.; Josef, N.: *Moderní hotelový management: 2., aktualizované a rozšířené vydání*. Grada Publishing, a.s., 2014, ISBN 9788024789835.

Příloha **A**

Seznam použitých zkratek

GDS Global Distribution System

CRS Computer Reservation System

PMS Property management system

OTA Online travel agency

SaaS Software as a service

XML Extensible Markup Language

HTTP Hypertext Transfer Protocol

URI Uniform Resource Identifier

JSON JavaScript Object Notation

REST Representational state transfer

SOAP Simple Object Access Protocol

WSDL Web Services Description Language

API Application Programming Interface

SOAP Simple Object Access Protocol

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF