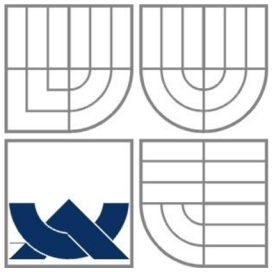
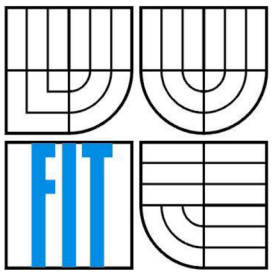


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM NÁVŠTĚVNICKÉHO CENTRA

INFORMATION SYSTEM OF THE VISITOR CENTER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. KAREL SOKL

VEDOUCÍ PRÁCE
SUPERVISOR

MGR. ROMAN TRCHALÍK, PH.D.

BRNO 2014

Abstrakt

Práce se zabývá tvorbou informačního systému pro návštěvnické centrum v Olomouci. Tento dokument popisuje celý životní cyklus tohoto rezervačního systému, včetně specifikace požadavků, návrhu systému, jeho implementace a testování. Výsledný systém bude nasazen do provozu muzea, kde se bude starat o ovládání turniketů, rezervaci a prodej vstupenek.

Klíčová slova

Informační systém, rezervační systém, návštěvnické centrum, muzeum, turnikety, vstupenky, prodej vstupenek, Java EE, Spring 4, Hibernate 4, REST API.

Abstract

Thesis is focused on creation of an information system for a visitor center in Olomouc. This document describes whole lifecycle of that reservation system, including requirements specification, system design, its implementation and testing. Final system will be deployed in museum, where it will handle the control of turnstiles, ticket reservations and sales.

Keywords

Information system, reservation system, visitor center, muzeum, turnstiles, tickets, ticket sales, Java EE, Spring 4, Hibernate 4, REST API.

Citace

Sokl Karel: Informační systém návštěvnického centra. Brno, 2014, diplomová práce, FIT VUT v Brně.

Informační systém návštěvnického centra

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Mgr. Romana Trchalíka, Ph.D a zadavatelelem projektu Mgr. Blankou Krausovou. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Děkuji svému vedoucímu práce, Mgr. Romanu Trchalíkovi, Ph.D, za jeho zájem o toto téma a také za dobré rady, které mi poskytl. Také děkuji projektovému týmu pevnůstky za to, že mi umožnili se podílet na vzniku Pevnosti poznání.

Obsah

Úvod

V Olomouci vzniká pod vedením Univerzity Palackého nové návštěvnické centrum, které dostalo název *Pevnost poznání*. Toto muzeum si klade za cíl hravou a zábavnou formou přilákat zejména děti a mládež k přírodovědným a společenským naukám. Pro veřejnost bude vystavovat různé exponáty z oblasti vědy, které budou v životní, někdy až v nadživotní velikosti. Například připravovaná expozice *Rozum v hrsti* se bude zabývat základním principem lidského vědění a návštěvníci se zde mimo jiné budou moci podívat dovnitř lidského mozku. Kromě místností pro expozice, bude mít pevnůstka tři multifunkční sály a malé planetárium, takže se zde také budou moci konat různé přednášky, koncerty, filmová promítání a podobné události.

Návštěva muzea samozřejmě nebude zdarma, na každou událost se budou prodávat vstupenky. Prostory pevnůstky jsou velké, místností je hodně a událostí zde bude probíhat spousta. Aby se v budoucnu ušetřila práce s organizací událostí, rezervací a prodejem vstupenek, kontrolou platnosti vstupenek nebo také s vedením účetnictví, bylo rozhodnuto, že bude nasazen informační systém, který bude tyto a mnoho dalších operací automatizovat.

Tato práce se zabývá specifikací požadavků, návrhem a realizací tohoto informačního systému. Nejprve je v první kapitole uvedeno zadání projektu, kde jsou neformálně popsány základní požadavky na systém. Také jsou zde shrnuty a přiblíženy všechny pojmy, které se budou v této práci vyskytovat. Druhá kapitola obsahuje specifikaci zadaných požadavků, dále případy jednání uživatelů systému a rizika. Obsahem třetí kapitoly je podrobný návrh systému. Tato kapitola systém rozděluje do několika samostatných modulů, které budou dále popisovány a pak také implementovány samostatně. Každému z nich je nejprve vybrána implementační technologie, následně je popsána jejich statická i dynamická struktura a nakonec jsou uvedeny náhledy na jejich grafická uživatelská rozhraní. Ve čtvrté kapitole jsou přiblíženy implementační detaily, které přímo nevyplývají z návrhu a v páté, poslední, kapitole je popsána metodika testování výsledného systému.

1 Zadání projektu

Následující popis slouží ke sjednocení pohledů zadavatele projektu a vývojáře na výsledný produkt. Zabývá se především vysvětlením pojmů, které přímo souvisí s provozem pevnůstky a se systémem samotným. Správné pochopení těchto pojmů je pro vývojáře nezbytným krokem pro vytvoření správného produktu, a proto byl sepsán na základě pravidelných konzultací zadavatele s vývojářem.

1.1 Základní popis systému

Jedná se o rezervační a objednávkový systém, do kterého bude možné zadávat události konané v rámci programu Pevnost poznání. Pomocí tohoto systému bude možné rezervovat a prodávat vstupenky na tyto události. Rezervaci i prodej vstupenek bude možné uskutečnit osobně na pokladně pevnůstky nebo vzdáleně přes webový portál pevnůstky. Systém bude spolupracovat s účetním softwarem, do kterého bude ukládat daňové doklady o prodaných vstupenkách.

Součástí systému budou také vstupní turnikety, které budou umístěné u vybraných místností pevnůstky. Systém bude tyto turnikety ovládat a bude kontrolovat platnost naskenovaných vstupenek podle jejich čárového, případně RFID, kódu. Systém bude dlouhodobě ukládat informace o prodaných vstupenkách a uskutečněných událostech.

1.2 Podrobný popis

V následujících odstavcích je uveden detailní popis pojmů a částí, které systém bude využívat a na kterých bude systém vystavěn. Popis je veden z pohledu uživatelů systému a jsou zde shrnuty všechny situace a operace, které bude možné se systémem provádět.

1.2.1 Místnost

Místnost je libovolný ohraničený objekt s omezenou kapacitou. Místnost má vstupní a výstupní turniket, pro kontrolu vstupenek, nebo nemá turniket žádný. V případě, že místnost není střežena turnikety, budou vstupenky kontrolovat zaměstnanci muzea. V místnosti se nebude udržovat zasedací pořádek pro účely prodeje vstupenek. Nebudou se tedy prodávat místenky, postačí omezení kapacitou místnosti.

1.2.2 Událost

Událost je vždy časově ohraničená a probíhá v nějaké místnosti (případně ve více místnostech). Událost má omezenou kapacitu účastníků. Každá událost se řídí centrálně zadanou otevírací dobou muzea, mimo tuto dobu události neprobíhají. Každé události je možné přiřadit nestandardní den a čas

(mimo pracovní dobu), ve kterém událost bude probíhat. Je tedy možné vytvářet události typu muzejní noc. Každá událost bude buď typu expozice anebo typu akce.

1.2.2.1 Expozice

Expozice je dlouhodobá událost, která nemá stanovený konec konání. Tato událost probíhá v místnostech, které jsou chráněné turnikety. Každý návštěvník této události bude mít svoji osobní vstupenku pro průchod turnikety.

1.2.2.2 Akce

Akce je krátkodobá událost, v řádu hodin nebo dní. Tato událost má přesně stanovený začátek i ukončení. Akce může probíhat v libovolné místnosti. Pokud bude akce určena pro veřejnost, bude mít vždy každý návštěvník svoji osobní vstupenku. Pokud bude akce soukromá (například pro konkrétní školu/třídu) a nebude probíhat v prostorách expozice, bude celé návštěvní skupině vytištěn jediný doklad s počtem návštěvníků.

Akce může být vytvořena i v prostorách, ve kterých probíhá expozice. V takovémto případě je průběh expozice v této místnosti pozastaven. Turnikety, které střeží místnost, propustí jen vstupenky platné na vytvořenou akci, vstupenky na expozici jsou zamítnuty.

Některé akce budou do systému registrovány na konkrétní datum a čas teprve tehdy, až si je zákazník rezervuje (týká se akcí pro školy).

1.2.3 Vstupenka

Vstupenka je časově ohraničený vstupní lístek, který platí na konkrétní zvolené události. Vstupenka bude mít vždy stanovenou dobu platnosti ve dnech (od - do), dále bude možné nastavit dobu platnosti vstupenky (v hodinách) od označení, také bude možné nastavit počet vstupů (pro dlouhodobé vstupenky). Kromě běžných jednorázových vstupenek s čárovým kódem se budou používat čipové karty.

1.2.3.1 Jednorázové vstupenky

Běžné jednorázové vstupenky budou tištěné na tiskárně a budou obsahovat přístupový čárový kód. Tyto vstupenky nebude možné recyklovat (znovu použít). Zákazníci si tyto vstupenky mohou vytisknout doma, pokud si je zakoupí přes internet.

1.2.3.2 Čipové karty

Dlouhodobé vstupenky budou řešeny formou čipových karet. Platnost těchto vstupenek bude omezena časově i počtem vstupů. Platnost bude možné prodlužovat. Tyto karty budou mít přidělené unikátní zákaznické číslo, které bude na kartě vyobrazené. Karty budou vydávány konkrétní osobě

a v systému budou o majiteli uchovávány základní informace (jméno, příjmení, e-mail, případně další). Čipové karty budou vydávány proti peněžité záloze.

Váženým zákazníkům bude možné vydat zlatou kartu. Zlatá karta bude mít volný přístup na veškeré expozice. Zákaznické číslo umožní majiteli karty čerpat slevy na akce.

Zaměstnanci budou mít své zaměstnanecké karty. Každé kartě bude možné nastavit dobu platnosti, bude také možné nastavit, do které místnosti s expozicí bude mít volný přístup. Zaměstnanecké karty budou do místnosti vpuštěny i v případě, že zákazník vstoupit nemůže (z důvodu zaplněnosti místnosti).

1.2.4 Statické turnikety

Turnikety budou schopné rozpoznat platnou vstupenku. Majiteli platné vstupenky umožní vstup na událost pouze v případě, že je v místnosti nějaké volné místo. Pokud volné místo v místnosti není, zobrazí se na displeji text "plná místnost". Pokud vstupenka není platná, zobrazí se na displeji text "neplatná vstupenka".

Turnikety mohou pracovat celkem ve třech módech: OPEN, CLOSE a TICKET. Pokud turniket pracuje v módu OPEN, tak se otevře vždy při naskenování libovolného čárového kódu (EAN13) nebo RFID kódu (125KHz), bez ohledu na to, zda se jedná o kód platné vstupenky nebo ne. Průchod turniketu v tomto režimu neovlivní platnost vstupenky – projít může kdokoliv. Režim CLOSE je opakem OPEN. Turniket, pracující v tomto režimu, zůstane vždy zavřený, tedy i v případě naskenování platné vstupenky. Výjimkou jsou zaměstnanecké karty, pro tyto se turniket otevře i v režimu CLOSE. Poslední režim TICKET je běžný pracovní režim turniketů. Turnikety, které pracují v tomto režimu, se otevřou pouze pro platnou vstupenku. Průchod turniketem bude adekvátně zpracován a zaznamenán: jednorázová vstupenka bude zneplatněna pro další vstupy a dlouhodobé vstupence bude snížen počet vstupů o jeden vstup.

1.2.5 Administrátorský klient

Administrátorský klient je počítačový program, případně webová aplikace, který bude umožňovat celkovou správu informačního systému a databáze událostí. Program bude podporovat uživatelské účty chráněné heslem, kde jednotliví uživatelé budou mít různá práva a budou moci provádět pouze operace, na které mají pověření. Pověření provádět jednotlivé operace bude uživatelům přidělovat hlavní administrátor, který bude pouze jeden.

Níže jsou vyjmenovány jednotlivé operace, které bude moci provádět administrátor s plným pověřením:

- Události (přidání, změna, smazání, vyhledávání)
- Místnosti (přidání, změna, smazání, zobrazení aktuální zaplněnosti)
- Turnikety (změna, zobrazení stavu)

- Vstupenky (vyhledávání zablokování, odblokování)
- SAP (export účetních dokladů)

1.2.6 Pokladna

Na každé pokladně poběží počítačový program, dále jen kasa, který bude umožňovat přímý prodej vstupenek zákazníkům. Kasa bude podporovat uživatelské účty chráněné heslem, z důvodu rozlišení obsluhujících pracovníků. Níže jsou uvedeny jednotlivé operace, které bude možné pomocí kasy provádět:

- Jednorázová vstupenka (prodej a tisk, zjištění platnosti)
- Dlouhodobá vstupenka (prodej, zjištění platnosti, prodloužení platnosti)
- Rezervace (vytvoření, zaplacení, zrušení)
- Platba (hotově, kartou)
- Aktuální zůstatek na kase (zobrazení)

Možnost vkládat a vybírat peníze z kasy má pouze pověřený úředník, tyto operace se budou provádět v účetním softwaru.

Vstupenky bude možné před jejich prodejem modifikovat, bude tedy možné vytvořit vstupenku, která bude platit na více událostí. Dále bude možné prodat na jednu prodejku více vstupenek a na pokladně bude možné vrátit platnou, nepoužitou vstupenku.

1.2.7 Web

Na webu Pevnosti poznání bude možné objednávat (kupovat) a rezervovat vstupenky na všechny nabízené události. Jednorázové vstupenky, zakoupené přes web, si výhradně bude zákazník tisknout sám. Tisk zakoupené vstupenky přes web nebude u pokladny možný. Dlouhodobé vstupenky, zakoupené přes web, si zákazník musí osobně vyzvednout na pokladně.

Při vytváření rezervace zákazník zadá své osobní údaje a při dokončení rezervace obdrží unikátní číslo této rezervace. Číslo rezervace bude sloužit pro její jednoznačnou identifikaci a bude po zákazníkovi požadováno v případě, že bude chtít svoji rezervaci zaplatit nebo zrušit. Zákazník může rezervaci zaplatit pouze na pokladně. Pokud bude zákazník chtít svoji rezervaci zrušit, může tak učinit přes web nebo osobně na pokladně. Všechny rezervace budou automaticky zrušeny 30 minut před začátkem konání události, ke které se vztahují. Podobně rezervace dlouhodobých vstupenek nezaplacené do 30 ti dnů od vytvoření budou zrušeny.

Nestandardní operace, například pronájmy místností, budou řešeny individuálně pověřenou osobou. Pomocí webu nebude možné objednat nebo rezervovat pronájem místnosti.

1.2.8 Ruční čtečky

Ruční čtečky, dále jen čtečky, budou používány pro události, které budou probíhat v místnostech bez statických turniketů. Čtečky budou schopné pracovat samostatně (off-line). Čtečky budou schopné rozpoznat platnou vstupenku. V případě naskenování platné vstupenky se na displeji zobrazí zpráva "Vstup povolen" a jednou zazní krátký zvukový signál (dle možností přístroje). Pokud je naskenována neplatná vstupenka, na displeji se zobrazí zpráva "Neplatná vstupenka" a dvakrát za sebou zazní krátký zvukový signál. Pokud je platná vstupenka naskenována podruhé, zobrazí se na displeji zpráva "Vícenásobný sken" a dvakrát za sebou zazní krátký zvukový signál.

Před začátkem práce s ruční čtečkou je nutné ji připojit k administrátorskému klientovi a nahrát do ní platné vstupenky. Po ukončení práce se čtečkou je nutné ji připojit k administrátorskému klientovi a naskenované vstupenky nahrát do systému. Ruční čtečka může obsahovat platné vstupenky pouze na jednu událost, nebude tedy možné jednou čtečkou kontrolovat více událostí najednou. Na jednu událost je však možné použít více ručních čteček. V takovémto případě ale není zajištěna ochrana proti vícenásobnému naskenování jedné vstupenky více čtečkami. Dodatečnou ochranu může zajišťovat obsluha znehodnocením naskenované vstupenky. Stále však zůstává riziko vytištění nebo zkopírování vstupenky a jejím násobném použití. Vícenásobné naskenování jedné vstupenky se zjistí až po nahrání naskenovaných vstupenek do systému.

Vstupenky, které se prodají až po nahrání platných vstupenek do čtečky (například na místě konání události), bude možné dodatečně (ručně) přidat do čtečky.

1.2.9 Dlouhodobě uchovávaná data

Databáze bude uchovávat veškeré (i ty, které jsou již neplatné) záznamy o událostech, vstupenkách, místnostech i turniketech minimálně po dobu dvou let. Veškeré informace týkající se objednávek a plateb budou uloženy v účetním systému.

1.2.10 Účetní software

Pro vedení účetnictví a uchovávání účetních dokladů bude použit účetní software. Do tohoto softwaru budou mít přístup autorizované osoby, které budou moci provádět veškeré běžné účetní operace, které tento účetní software bude nabízet. Program kasa bude automaticky vytvářet potřebné doklady a vkládat je přímo do účetního softwaru, podobně tomu bude v případě webu. Pokladní tedy nebude mít přímý přístup do účetního softwaru.

1.2.11 Profily uživatelů

Typickými uživateli, ze strany webu, budou osoby se základní znalostí práce na počítači, webové rozhraní proto bude mít jednoduché a intuitivní ovládání.

Zaměstnanci, kteří budou obsluhovat pokladnu, budou mít patřičné školení pro ovládání programu kasa. Pro zajištění svižné práce bude mít kasa jednoduché a přehledné uživatelské rozhraní.

Administrátorskou správu bude provádět školená osoba, uživatelské rozhraní administrátorského klienta proto může být tvořeno pomocí tabulek a formulářů. O celkovou údržbu systému se bude starat osoba s pokročilou znalostí práce na počítači.

2 Specifikace požadavků

Ve spolupráci se zadavatelem projektu byly sepsány požadavky na cílový systém. V této kapitole jsou nejprve uvedeny případové studie, dále jsou z těchto studií odvozeny případy užití, které by měl systém umožňovat. K těmto případům je také uveden diagram jednání. Nakonec jsou uvedena rizika sytému.

2.1 Případové studie

Uživatelé systému byli rozděleni celkem do tří skupin: zákazníci u pokladny, zákazníci na webu a správci systému. Zákazníci u pokladny jsou ti zákazníci, kteří osobně přijdou do pevnůstky a kteří chtějí využít některou službu pokladny, například si zakoupit vstupenku. Podle potřeb tohoto zákazníka budou utvářeny funkce, které by měla nabízet pokladna. Zákazník na webu je takový zákazník, který chce využít nějakou službu, nabízenou prostřednictvím webového portálu. Podobně jako u pokladny, funkce webu budou utvořeny tak, aby pokryly potřeby tohoto zákazníka. Poslední skupinu, správce systému, budou tvořit zaměstnanci pevnůstky, kteří budou mít na starosti správu a údržbu systému. Administrátorský klient by měl svými funkcemi pokrýt veškeré potřeby správce.

U každé z těchto tří skupin jsou uvedeny očekávané situace, které by měly v reálném běhu systému běžně nastávat. Všechny studie jsou pojaty z pohledu zástupců dané skupiny ve smyslu „co chce udělat“, například zákazník u pokladny si *chce* zakoupit vstupenku. Pokud je to vhodné, je také případ doplněn komentářem, který popisuje smysl požadavku nebo výsledek obsloužení požadavku.

2.1.1 Zákazník u pokladny

- Zakoupit vstupenku na událost
  - Vstupenka je rovnou zaplacena, vytisknuta a aktivována v systému.
- Zjistit platnost vstupenky
  - Zákazník si chce ověřit, zda jeho vstupenka platí, do kdy platí, kolik vstupů ještě zbývá nebo na kterou událost vstupenka platí.
- Vrátit platnou, nepoužitou vstupenku
  - Zákazníkovi jsou vráceny peníze, vstupenka je znovu nabídnuta do prodeje.
- Zjistit jaké události probíhají a zda jsou volná místa
- Vytisknout fakturu k zakoupené vstupence
  - Zákazník musí dopředu říct, že chce vstupenku na fakturu, tuto fakturu je pak možné tisknout i zpětně.
- Zaplatit objednanou rezervaci vstupenek

- Zákazník sdělí své jméno, název události, datum a čas konání události, číslo rezervace, případně variabilní číslo. Zaplacené vstupenky jsou vytisknuty a aktivovány v systému.

- Zrušit objednanou rezervaci vstupenek (nezaplacenou)
  - Zákazník sdělí své jméno, název události, datum a čas konání události, číslo rezervace, případně variabilní číslo. Rezervace je zrušena, vstupenky jsou uvolněny do dalšího prodeje.

2.1.2 Zákazník na webu

- Zakoupit vstupenku na událost
  - Vstupenka je rovnou zaplacena (platební bránou nebo převodem), vstupenka je nabídnuta ve formátu PDF ke stažení a vytisknutí. Vstupenka je aktivována v systému.
- Rezervovat vstupenku na událost
  - Vstupenka je blokována pro konkrétní osobu (na jméno). Nezaplacená rezervace bude automaticky zrušena 30 minut před začátkem události nebo dva týdny po jejím vytvoření. Vstupenka se rezervuje na konkrétní čas a událost. Zákazník obdrží variabilní symbol a číslo objednávky.
- Zrušit objednanou rezervaci vstupenek
  - Zákazník zadá do formuláře číslo objednávky a variabilní symbol. Objednávka je zrušena, vstupenky jsou uvolněny do dalšího prodeje.
- Zjistit platnost vstupenky
  - Zákazník si chce ověřit, jestli jeho vstupenka platí, do kdy platí, kolik vstupů ještě zbývá, na kterou událost vstupenka platí.
- Zjistit jaké události probíhají a zda jsou volná místa

2.1.3 Správce systému

- Vytvořit novou událost
  - Událost je vytvořena, jsou jí přiřazeny místnosti, a je zvolena cena, za kterou se budou prodávat vstupenky na tuto událost. Od okamžiku vytvoření je možné zakoupit vstupenky na tuto událost. Místnosti, ve kterých událost probíhá, jsou po dobu konání události blokovány.
- Vytvořit novou krátkodobou událost v obsazené místnosti
  - Bude možné vytvořit jednorázovou, krátkou událost v místnosti, ve které probíhá dlouhodobá událost (expozice). Dlouhodobá událost se pozastaví po dobu konání krátkodobé události, vstupenky na dlouhodobou událost nebudou platné po dobu

konání krátkodobé události. Po skončení krátkodobé události se obnoví platnost dlouhodobé události.

- Upravit událost
    - Bude možné upravit popis události a bude možné změnit místnosti, ve kterých bude událost probíhat.
- Zobrazit všechny události podle kritérií
    - Vyhledávací kritéria budou: datum konání události, místnost, ve které událost probíhá, název události, klíčová slova v popisu události.
- Zobrazit detail události
    - Zobrazí se veškeré informace o události.
- Zobrazit stav turniketů
    - Zobrazí se aktuální informace o turniketech.
- Upravit turniket
    - Bude možné upravit adresy, módy a popisy turniketů.
- Zobrazit přehled místností
    - Zobrazí se seznam registrovaných místností.
- Zobrazit detail místnosti
    - Zobrazí se veškeré informace o vybrané místnosti, včetně událostí, které v ní v blízké době budou probíhat.
- Přidat novou místnost
    - Zaregistrování nové místnosti do systému, místnost pak bude možné přiřadit událostem.
- Upravit místnost
    - Provedené změny se projeví s okamžitou platností – důležité brát v úvahu především pokud se změní vstupní nebo výstupní turniket.
- Zobrazit informace o konkrétní vstupence
    - Budou zobrazeny veškeré informace o vstupence, například název události, počet zbývajících vstupů, datum a čas provedených skenů, datum a čas zakoupení, datum ukončení platnosti a další.
- Zablokovat vstupenku
    - Konkrétní vstupenka bude v systému zablokována, zneplatněna.

2.2 Případy užití

V následujícím textu jsou všechny případové studie, uvedené v předchozí podkapitole, přeformulovány z pohledu zákazníka na pohled ze strany systému. Jak bylo naznačeno v předchozím textu, účastníci systému jsou rozděleni do tří skupin. V následujícím textu jsou tyto tři skupiny vhodně přejmenovány tak, aby jejich název byl stručnější a aby charakterizoval jak povahu a roli účastníka, tak jeho místo působiště. Zároveň jsou tímto operace přesunuty na ty účastníky, kteří je budou v systému skutečně provádět.

Zákazníka na pokladně bude představovat samotná *pokladna*. Pokud zákazník přijde na pokladnu pevnůstky a bude si chtít například zakoupit vstupenku, bude to právě pokladna, která uskuteční samotný prodej. Pokladnu je možné chápat jako osobu, která obsluhuje pokladní software, nebo jako samotný software, který na pokladním počítači běží. Veškeré operace, které může pokladní v systému provádět, bude moci provést pouze pomocí pokladního softwaru.

Zákazníka na webu bude představovat *web*. V tomto případě je web možné chápat buď jako osobu (zákazníka), který obsluhuje webový portál pevnůstky, nebo samotný webový portál pevnůstky. Podobně jako u pokladny, veškeré operace, které bude moci zákazník na webu provádět, bude možné provést pouze pomocí webového portálu.

Správce jako jediný si zachová své pojmenování. V tomto případě je správcem osoba, která se bude starat o plynulý běh systému pevnůstky. Na rozdíl od hlavního administrátora, který bude spravovat hardware a databázi na její zdrojové úrovni a jiné, bude tento správce pouze přidávat do systému nové události, přidávat nové místnosti, spravovat vstupenky a podobně.

Následuje výčet jednání pro jednotlivé aktéry v systému.

2.2.1 Pokladna

- Prodej a tisk vstupenky
- Vytvořit rezervaci
- Zaplatit rezervaci
- Zrušit rezervaci
- Tisk faktury
- Zobrazit informace o rezervaci
- Zobrazit informace o vstupence
- Zobrazit události
- Vrátit vstupenku
- Zobrazit aktuální zůstatek

2.2.2 Web

- Prodej a tisk vstupenky
- Vytvořit rezervaci
- Zrušit rezervaci
- Tisk faktury
- Zobrazit informace o rezervaci
- Zobrazit informace o vstupence
- Zobrazit události

2.2.3 Správce

- Zobrazit události
- Zobrazit turnikety
- Upravit turniket
- Zobrazit místnosti
- Přidat místnost
- Upravit místnost
- Smazat místnost
- Zobrazit vstupenky
- Zablokovat vstupenku
- Odblokovat vstupenku
- Přidat událost
- Upravit událost
- Smazat událost
- Vyhledat událost
- Export dokladů do SAPu

2.2.4 Diagram jednání

Obrázek 2.1: Celkový diagram jednání

2.3 Rizika

Zde jsou popsána známá rizika, která mohou reálně narušit zamýšlený provoz systému.

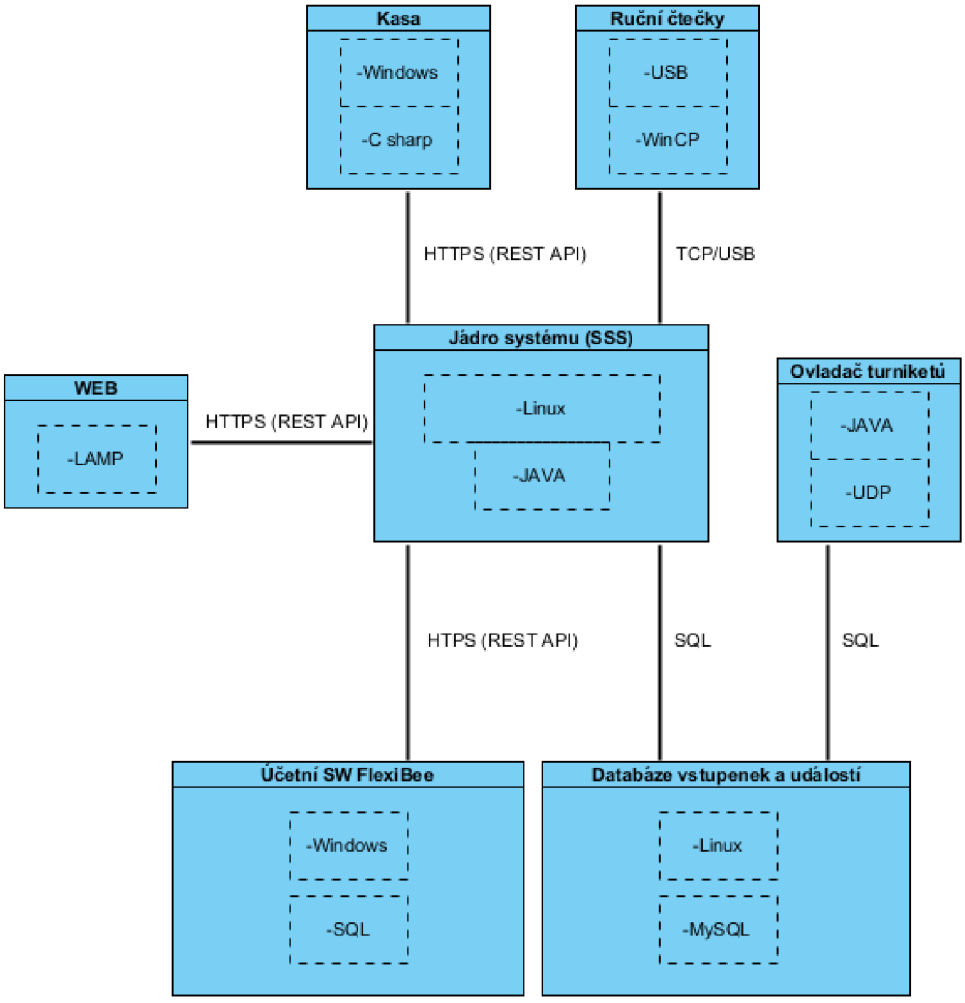
- Pokud zákazník zaplatí převodem v den konání akce, je velmi pravděpodobné, že platba nestihne dorazit na účet pevnůstky a koupená vstupenka tedy nebude platná. Platba se navíc nejprve musí spárovat s fakturou v účetním softwaru a teprve pak se vstupenka v systému aktivuje. Jako opatření bude zákazníkovi také nabídnuta možnost zaplatit vstupenky elektronickou kartou. V takovémto případě systém obdrží od banky potvrzení o platbě ihned.
- Zákazník naskenuje vstupenku na turniketu a nestihne projít. Jednorázovou vstupenku je možné použít pouze jednou, proto zákazník na událost nebude vpuštěn. Navíc dokud vstupenka nebude naskenována výstupním turniketem, systém bude věřit, že zákazník je stále v místnosti. Tato situace by měla nastávat zřídka kdy a pokud nastane, budou ji řešit zaměstnanci muzea například tak, že zákazníka pustí pomocí své zaměstnanecké karty.
- Zrušení události, na kterou už jsou prodané vstupenky. Zrušení události není běžná operace a v případě potřeby ji bude provádět správce pomocí administrátorského rozhraní. Vrácení prodaných vstupenek a peněz se bude muset provést ručně v účetním systému.
- Ruční čtečky budou pracovat off-line, proto bude nutné ručně vyřizovat prodeje vstupenek v den konání události. Kódy vstupenek, které se prodají před začátkem konání události, budou nahrané do ruční čtečky, proto bude možné provádět jejich plnou kontrolu. Kódy vstupenek, které se prodají později, nebude možné plně kontrolovat, pravděpodobně je bude nutné kontrolovat oproti vytištěnému seznamu.
- Pokud se použije více ručních čteček pro kontrolu vstupenek jediné události, čtečky mezi sebou nebudou komunikovat. Nebude tedy možné během skenování vstupenky ověřit, zda již vstupenka byla nebo nebyla naskenována jinou čtečkou.

3 Návrh systému

V této kapitole je sepsán návrh celého systému. V úvodu kapitoly je ukázáno rozčlenění systému do jednotlivých modulů a každý modul pak bude popisován zvlášť. U každého budou uvedeny vybrané implementační technologie a návrhové vzory. Další významnou částí návrhu je popis statické a dynamické struktury modulů systému a komunikačních rozhraní těchto modulů. Nakonec jsou uvedeny náhledy na grafické uživatelské rozhraní vybraných obrazovek.

3.1 Rozdělení systému do modulů

S ohledem na požadavky, které jsou na systém kladeny, byl tento rozdělen celkem na sedm částí: jádro systému (*Server Side System*, SSS), databázi vstupenek a událostí, účetní software, ovladač turniketů, ruční čtečky, kasu a web. Situaci vystihuje následující obrázek (Obrázek 3.1). Jednotlivé části jsou blíže popsány v následující podkapitole.



Obrázek 3.1: Moduly informačního systému

3.2 Implementační technologie

Na následujících řádcích jsou postupně popsány všechny moduly systému a u každého z nich jsou uvedeny zvolené implementační technologie. Tyto technologie byly vybírány s ohledem na požadovanou funkčnost jednotlivých modulů a také s ohledem na hostující hardware a jeho operační systém.

Na hlavním serveru pevnůstky bude nainstalovaný Ubuntu server 12.04.3 LTS, Apache, MySql, PHP server a Java Virtual Machine. Na počítači, který bude sloužit jako pokladna, bude nainstalovaný systém Windows 7 nebo 8. Počítače, které budou sloužit pro administrátorský přístup (pro správce), budou mít nainstalovaný systém Windows 7 nebo 8.

3.2.1 Jádro systému SSS

Tato serverová aplikace bude nejdůležitější částí systému. Jádro bude zpracovávat téměř všechny operace, které jsou se systémem spojené. Ostatní moduly budou implementovány jako klienti, kteří budou využívat služeb tohoto jádra. Výjimkou bude ovladač turniketů, který bude pracovat samostatně, nezávisle na jádře.

Toto jádro poběží na hlavním serveru pevnůstky a bude mít na starosti veškerou manipulaci s událostmi a se vstupenkami, včetně prodeje vstupenek a vytváření dokladů v účetním softwaru. Ke svému provozu bude využívat databázi a účetní software, do kterých bude mít přímý přístup.

SSS bude implementované jako webová aplikace napsaná v Javě, která své služby bude nabízet přes *REST API* pomocí *HTTP* protokolu. Pro vývoj budou použity rámce *Spring* [2 a 8] a *Hibernate* [3].

3.2.2 Administrátorský klient

Tento klient bude sloužit jako správce jádra SSS (a jeho databáze), které bude přímo ovládat pomocí *REST API*. Implementován bude jako webová aplikace (pomocí *JSP*, *HTML*, *CSS, JavaScriptu*) a bude vyvíjen spolu s jádrem SSS jako jeho součást. Z tohoto důvodu není tento klient uveden jako samostatný modul.

3.2.3 Kasa

Aplikace pro pokladnu (kasa) bude sloužit pro přímý prodej vstupenek v muzeu. Kasa bude implementována v jazyce *C#* jako samostatná desktopová aplikace. Tato aplikace bude představovat klienta a se serverem (jádrem SSS) bude komunikovat pomocí *REST API*. Tato aplikace bude implementována jako „tenký klient“ a veškeré požadavky, včetně autorizace obsluhy pokladny, budou delegovány serveru.

3.2.4 Web

Webový portál pro zákazníky bude sloužit především pro rezervaci a nákup vstupenek a dále také pro poskytnutí informací o událostech. Web bude součástí stávajícího portálu Pevnosti poznání. Pro komunikaci se serverem (jádrem SSS) bude použito *REST API*. Pro implementaci bude použit rámec *Nette* [9].

3.2.5 Ovladač turniketů

Tento ovladač bude sloužit pro obsluhu statických turniketů. Zda se turniket má nebo nemá otevřít, bude vyhodnoceno na základě údajů v databázi vstupenek a událostí. Ovladač bude implementován jako samostatná serverová aplikace v Javě a bude mít k databázi přímý přístup. Statické turnikety budou posílat všechny kódy vstupenek, které naskenují, přímo tomuto ovladači. Ovladač bude zpětně, jako odpověď, turniketům posílat příkazy pro otevření nebo vypsání textu na displej. Veškerá komunikace s turnikety bude probíhat po síti pomocí UDP protokolu.

3.2.6 Databáze vstupenek a turniketů

V této databázi budou uložené veškeré dlouhodobé informace, které budou potřebné pro plynulý běh systému. Především se zde budou ukládat informace o probíhajících událostech, o prodaných vstupenkách a také o uživatelích systému. Do této databáze budou mít přímý přístup jádro SSS a ovladač turniketů. Ovladač turniketů bude v databázi především vyhledávat. Změny, které bude ovladač v databázi provádět, budou minimální, a proto svým přímým přístupem nijak neovlivní práci jádra systému.

Celková podoba databáze je uvedena v kapitole 3.4. Pro implementaci bude použito *MySql*.

3.2.7 Ruční čtečky

Podle konkrétního vybraného modelu zařízení budou vybrány odpovídající technologie pro implementaci aplikace. Jedinou nutnou podmínkou pro výběr zařízení je povinná podpora čtení čárového kódu v kódování *EAN13*. Velmi pravděpodobně bude vybráno zařízení s operačním systémem *Windows CE*.

3.2.8 Účetní software

Účetní software by měl nabízet standardní funkce potřebné pro vedení účetnictví a také by měl umožnit automatizovat většinu operací (například vytvoření prodejky nebo faktury). Také by měl tento software umožňovat vzdálený přístup (po síti) a možnost spouštět operace pomocí skriptů.

Všem těmto podmínkám vyhovuje produkt *FlexiBee* [4]. Jedná se o účetní software s architekturou klient-server, který je možné ovládat pomocí *REST API* přes *HTTPS*. Oficiální stránky

výrobce poskytují relativně kvalitní programátorskou dokumentaci, ve které popisují jak s programem pracovat pomocí nabízeného *REST API* a *CHANGES API*. Díky tomuto rozhraní je možné implementovat vlastního klienta, který bude účetní software ovládat. Tento produkt je napsaný v Javě a je tedy možné jej nainstalovat na libovolný operační systém, který Javu podporuje.

Při výběru účetního softwaru byly zvažovány celkem čtyři různé produkty: *KelSQL* [5], *POHODA* [6], *Money S3* [7] a *FlexiBee*. První tři jmenované produkty, byly pro některé své vlastnosti a nedostatky zavrženy. Software *KelSQL* sice nabízí architekturu klient-server, ale je implementovaný nad databází *MS SQL*, a musí běžet pod operačním systémem Windows. Ze stejného důvodu byly odmítnuty také produkty *POHODA* a *Money S3*, které jsou také určeny výhradně pro stanice s operačním systémem Windows. *Money S3* navíc nenabízí žádné rozhraní, pomocí kterého by se daly vytvářet nebo mazat účetní doklady. Ovládání pomocí příkazové řádky, které tento produkt umožňuje, zdaleka nenabízí takové možnosti, jaké jsou pro tento informační systém vyžadovány.

3.3 Řízení přístupu

V systému bude rozlišováno několik uživatelských rolí podle pravomocí, které budou v rámci systému mít. Přihlašovací údaje všech uživatelů budou uloženy v databázi SSS. Uživatelé, kteří chtějí se systémem pracovat, se musí nejdříve přihlásit platným uživatelským jménem a heslem. Jednotlivé role v systému jsou: administrátor (správce), pokladní a zákazník (veřejnost). Všechny tyto role mohou v systému provádět právě ty operace, které jsou popsané ve specifikaci požadavků.

3.3.1 Profil uživatele

Aplikace rozlišuje dva uživatelské profily: *zákazník* a *zaměstnanec*. Uživatel kteréhokoli typu nemá možnost svůj profil si zobrazit ani změnit. Veškeré potřebné změny provádí autorizovaná osoba – správce.

Profil zaměstnance slouží pouze pro identifikaci uživatele v systému. Pokud zaměstnanec zapomene své přihlašovací heslo, administrátor mu nastaví nové heslo a odešle jej na jeho e-mail.

Podle specifikace požadavků je každá dlouhodobá vstupenka (RFID karta) vydávána na konkrétní jméno proti finanční záloze. Každý zákazník, který má zakoupenou dlouhodobou vstupenku, má v systému uloženy své osobní údaje.

3.3.2 Profil vstupenky

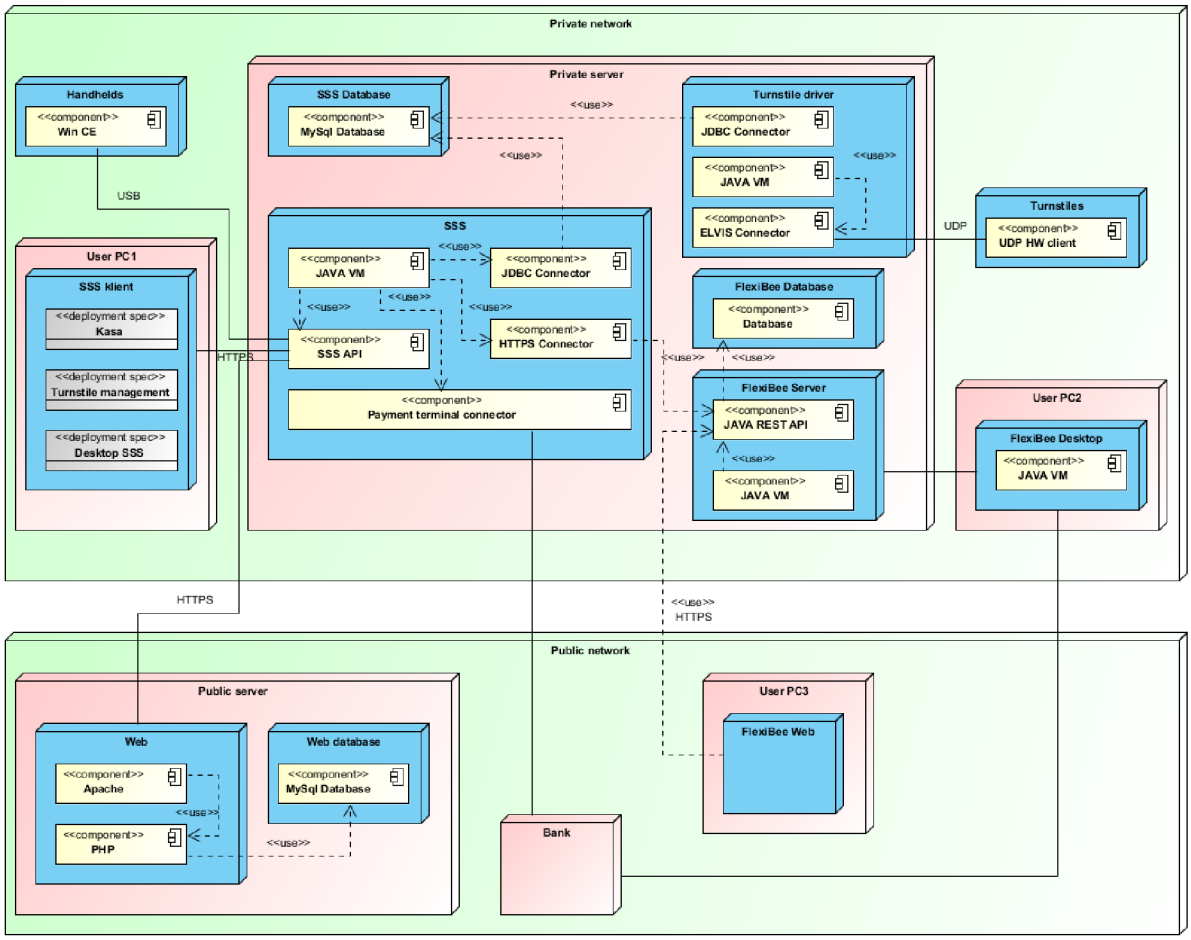
Každá dlouhodobá vstupenka má své zákaznické číslo a je vázána na konkrétního uživatele. Každý uživatel má možnost si prohlédnout aktuální stav vstupenky a zjistit tak zbývající počet vstupů nebo datum ukončení platnosti vstupenky. Profil karty je možné zobrazit přes web. Pro zobrazení je nutné znát číslo karty, jméno a příjmení držitele.

3.4 Statická struktura

V této kapitole bude popsána statická struktura systému. Nejprve je zde ukázán systém jako celek, dále je zde uvedena celková podoba databáze a nakonec jsou popsány jednotlivé moduly systému.

3.4.1 Model komponent systému

Na následujícím obrázku (Obrázek 3.2) je ukázán celkový pohled na výsledný systém. Jsou zde vidět všechny jeho části, včetně použitých implementačních technologií. Také je zde možné vidět, jak bude vypadat reálné nasazení systému, v jakých sítích budou provozovány jednotlivé moduly a způsob komunikace mezi jednotlivými částmi.



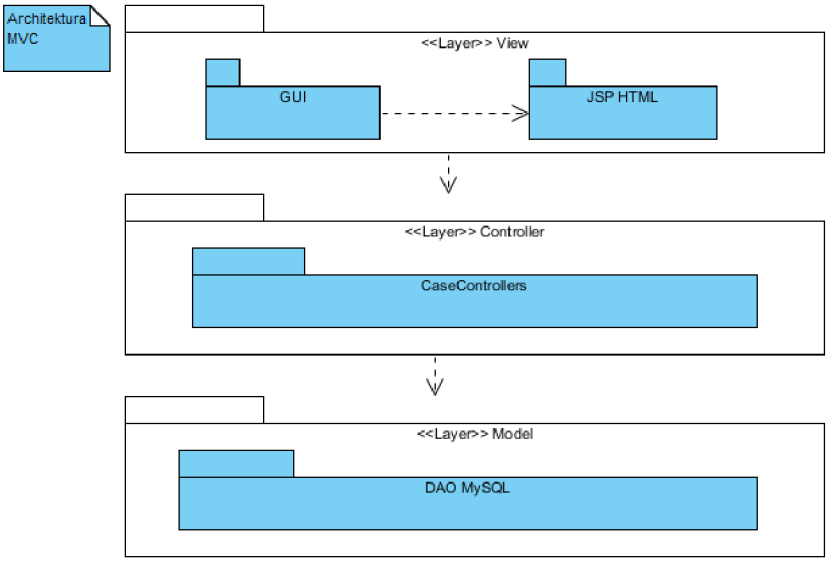
Obrázek 3.2: Diagram nasazení

Private Network představuje vnitřní síť pevnůstky, do této sítě nebudou mít běžní uživatelé přímý přístup. Do této sítě budou připojeny: privátní server pevnůstky, statické turnikety (*Turnstiles*), pokladny (kasy), správci systému (administrátorský klient) a také ruční čtečky (*Handhelds*).

Public network představuje veřejnou síť (internet). Do této sítě bude mít přístup kdokoliv, nejdůležitějšími částmi jsou: veřejný server, uživatelské počítače a banka.

3.4.2 Jádro SSS

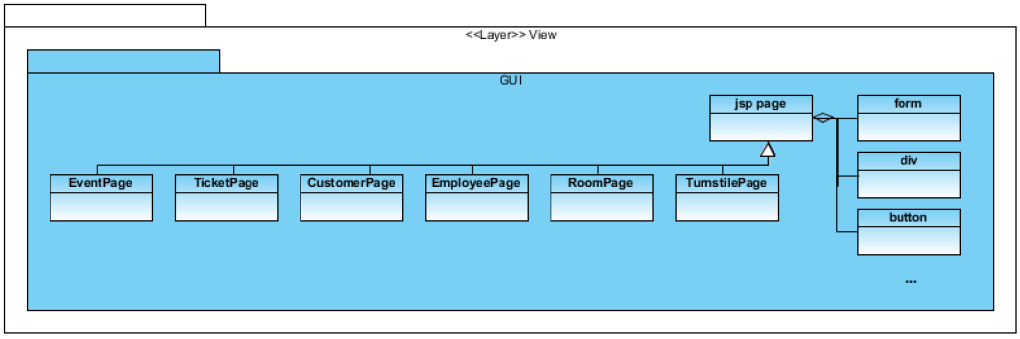
Pro implementaci bude použita architektura *Model – View – Controller* (*MVC*). Komponenta *Model* v této architektuře zajišťuje zpracování dat a manipulaci s nimi. Pro implementaci této komponenty bude využit rámec *Hibernate*, který automaticky dokáže mapovat třídy na relační tabulky a naopak. Komponenta *Controller* realizuje řídící (*business*) logiku aplikace a komponenta *View* se stará o prezentaci výstupu aplikace uživateli. Architekturu je možné vidět na následujícím obrázku (Obrázek 3.3). V následujících podkapitolách jsou uvedeny podrobnější pohledy na tyto tři komponenty.



Obrázek 3.3 Architektura systému (MVC)

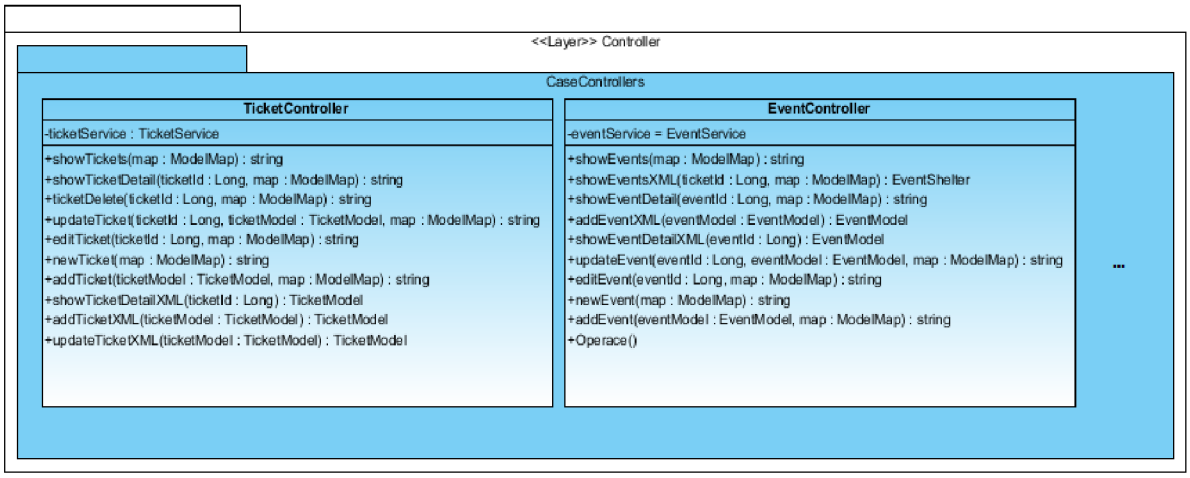
3.4.2.1 View

Přes toto grafické rozhraní bude možné plně ovládat celý systém, a proto bude sloužit jako administrátorský klient pro správce systému. Na následujícím obrázku (Obrázek 3.4) je uveden detailnější pohled na vrstvy *view* a *controller*.



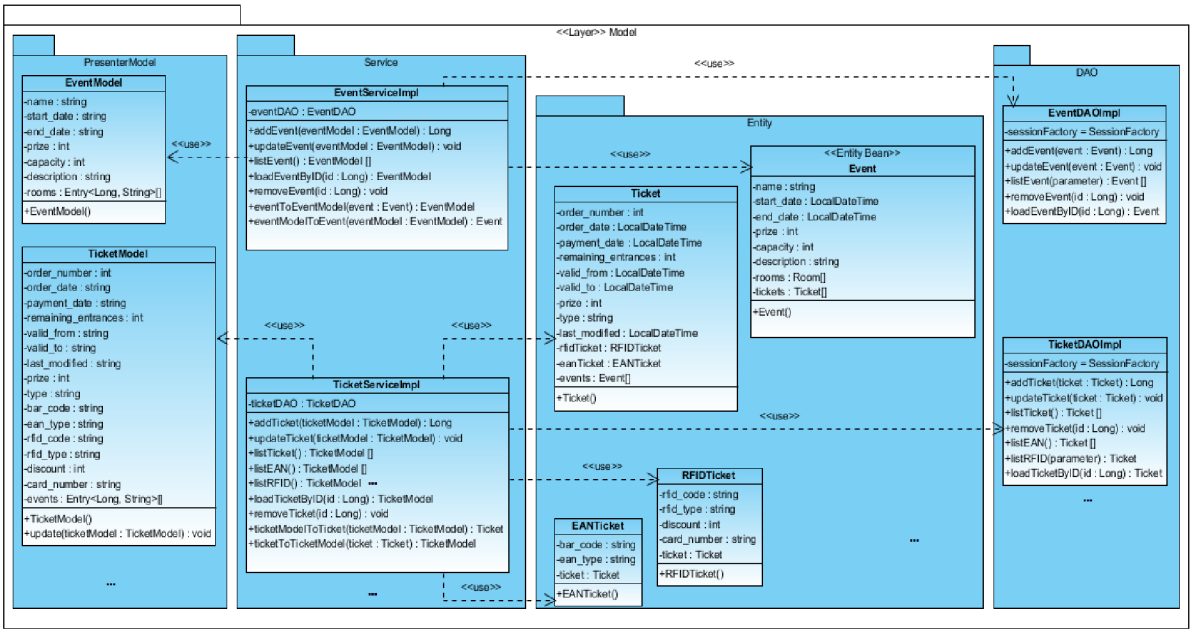
Obrázek 3.4 View

3.4.2.2 Controller



Obrázek 3.5 Controller

3.4.2.3 Model

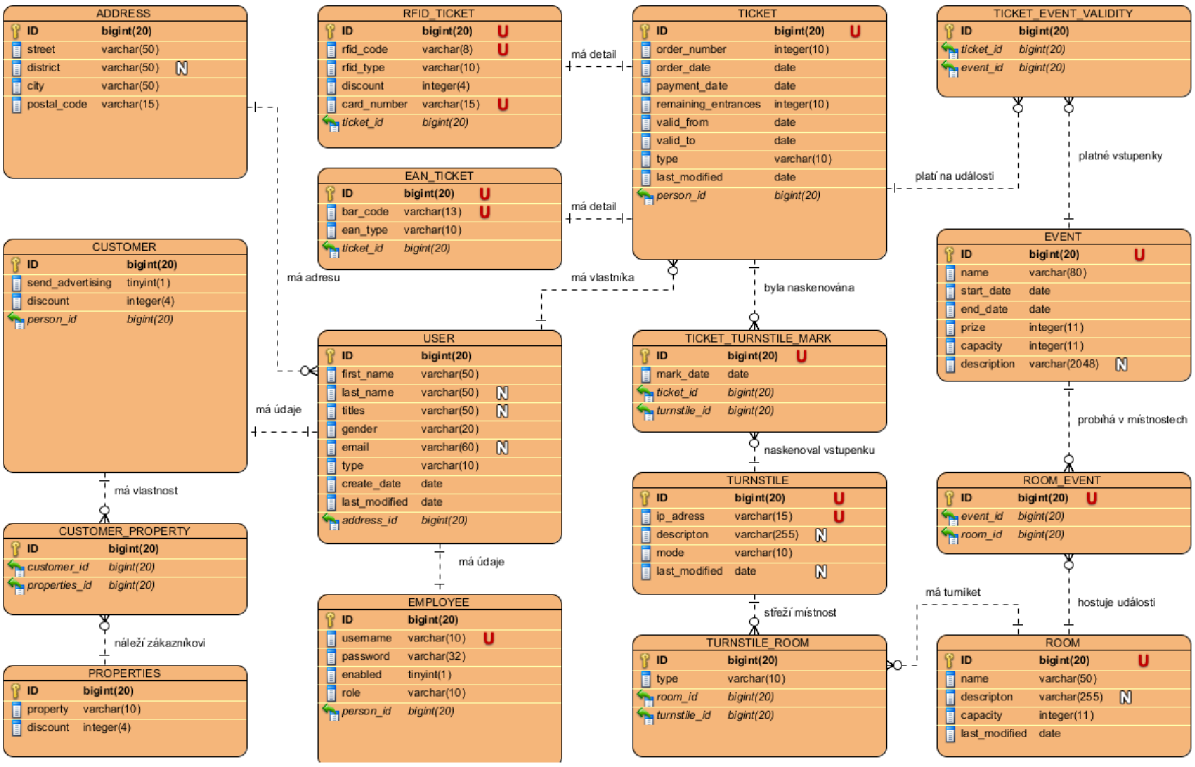


Obrázek 3.6 Model

Z důvodu úspory místa a udržení přehlednosti obrázku je vyobrazeno pouze několik hlavních zástupců z každého balíčku tříd. Také zde byly vynechány některé metody (konkrétně settery a gettery). Celkový diagram je možné vidět na přehledovém diagramu balíčků v příloze tohoto dokumentu (Obrázek II.1).

3.4.3 Relační databázový model

S ohledem na požadavky a na potřeby systému, bylo vytvořeno následující schéma databáze (Obrázek 3.7).



Obrázek 3.7: Schéma databáze vstupenek a událostí

V systému jsou rozlišovány dva typy vstupenek: dlouhodobá vstupenka (*RFID* karta) a jednorázová vstupenka (vstupenka s čárovým kódem *EAN13*). Obě tyto vstupenky mají mnoho společného, a proto jsou ukládány do společné tabulky *TICKET*. Některé společné vlastnosti jsou například:

- Číslo objednávky, které slouží pro spárování vstupenky s účetním dokladem,
- datum zaplacení, podle kterého se pozná, zda byla vstupenka zaplacena a kdy,
- zbývající počet vstupů, kde u jednorázových vstupenek je maximálně jeden vstup,
- začátek platnosti nebo
- konec platnosti vstupenky.

Vlastnosti, které jsou odlišné, jsou ukládány odděleně do tabulek *RFID_TICKET* a *EAN_TICKET*. U obou typů vstupenek jsou odlišné vlastní identifikační kódy. RFID kód má délku 8 znaků v šestnáctkové soustavě, zatímco čárový kód EAN13 má délku 13 číslic v desítkové soustavě. Další odlišností je, že u dlouhodobých vstupenek je potřeba také ukládat zákaznické číslo karty a procentuální slevu karty.

O každém uživateli jsou v systému ukládány základní osobní údaje, jako například jméno, příjmení nebo email. Tyto údaje jsou uloženy ve společné tabulce *USER*. Další údaje, specifické pro konkrétní roli, jsou ukládány zvlášť v tabulkách *EMPLOYEE* (pro zaměstnance) a *CUSTOMER*

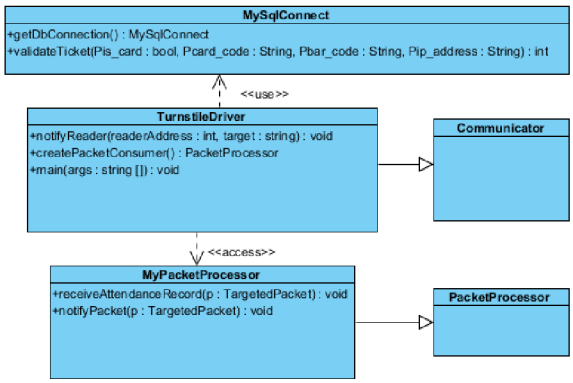
(pro zákazníka). O zaměstnancích jsou v databázi navíc ukládány údaje, které jsou nutné pro jeho identifikaci v systému, tedy uživatelské jméno, heslo a role. O zákaznících jsou navíc ukládány údaje spojené s marketingem, tedy například informace o tom, zda zákazníkovi posílat na email reklamu, nebo velikost jeho osobní procentuální slevy.

3.4.4 Kasa

Pro aplikaci kasa byla také zvolena architektura *MVC*. Modelová vrstva bude komunikovat se serverem, bude odesílat a získávat data, vrstva *Controller* bude implementovat obsluhu požadavků vrstvy *View* a bude předávat data mezi vrstvami *Model* a *View*. Pohledová vrstva bude vytvořena pomocí *WPF* (*Windows Presentation Foundation*) [18], vrstvy modelu a controlleru budou tvořeny *C#* kódem.

3.4.5 Ovladač turniketů

Tento ovladač bude vytvořen jako konzolová aplikace v Javě o celkovém rozsahu několika málo tříd. Názorný pohled na strukturu aplikace je uveden na následujícím obrázku (). Třída *TurnstileDriver* bude obsluhovat požadavky turniketů. Pro každý požadavek vytvoří novou instanci třídy *MyPacketProcessor*, která požadavek obslouží v novém vlákně. Třídy *Communicator* a *PacketProcessor* obsluhují nízko-úrovňovou komunikaci s turnikety a byly dodány v rámci kódu od dodavatele turniketů, od firmy *ELVIS*. Třída *MySqlConnect* bude statická, veřejná třída, která bude zprostředkovávat přístup do databáze.



Obrázek 3.8: Diagram tříd ovladače turniketů

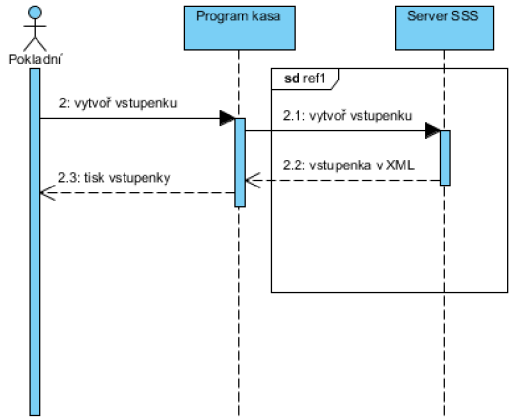
3.5 Dynamická struktura

Tato kapitola popisuje chování systému během zpracování vybraných důležitých operací. Byly vybrány operace, které budou během provozu běžně nastávat. U těchto operací jsou uvedeny sekvenční nebo komunikační diagramy, které pomohou lépe pochopit zpracování dané operace systémem.

3.5.1 Zakoupení vstupenky

Zakoupení (prodej) vstupenky je jednou z nejdůležitějších a nejčastějších operací, kterou bude systém provádět. Zákazník si může vstupenku zakoupit buď osobně na pokladně pevnůstky, anebo vzdáleně přes webový portál. Pro obě tyto možnosti je zde uveden sekvenční diagram, který názorně ukazuje způsob zpracování požadavku.

Nejprve je uveden prodej vstupenky na pokladně. Tento prodej začíná vytvořením nové objednávky na pokladně. Jakmile je objednávka připravena, pokladní klikne na tlačítko *Zaplatit*. Po stisku tohoto tlačítka se kontaktuje server SSS a předají se mu informace o požadované vstupence. Server nejprve vytvoří instanci vstupenky. Následně se vytvoří záznam vstupenky v databázi, což způsobí, že vstupenka bude od této chvíle v systému platná. Pokud byla vstupenka úspěšně uložena, vytvoří se v účetním systému prodejka. Nakonec server odešle zpět na pokladnu vytvořenou vstupenku, která je pak vytištěna na tiskárně.

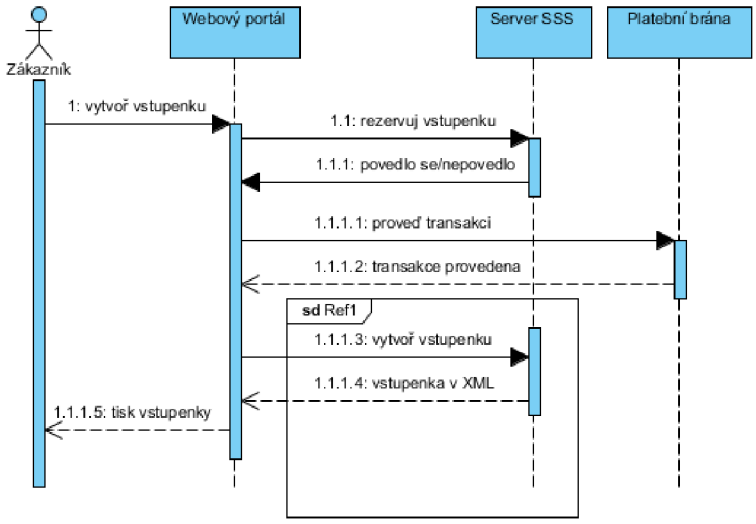


Obrázek 3.9: Zakoupení/prodej vstupenky na pokladně

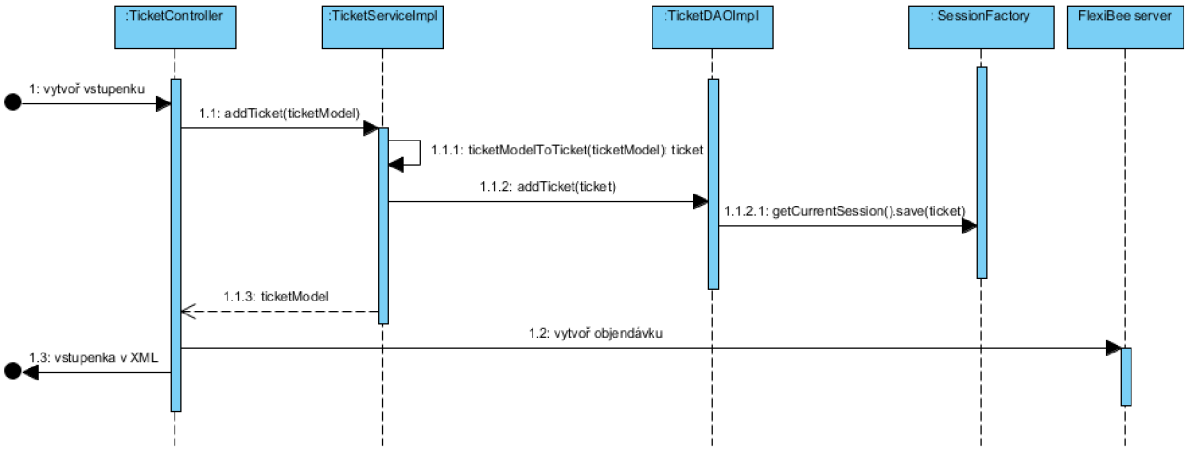
Na obrázku (Obrázek 3.9) je stručně ukázána celá operace. Naznačená komunikace mezi programem *kasa* a serverem probíhá přes rozhraní *REST API*, poskytované serverem. Detailnější popis zpracování operace na serveru je ukázáno na obrázku (Obrázek 3.11). Je možné si všimnout, že v diagramech není nikde zmínka o platbě vstupenky. Příjem platby za vstupenku má na starosti osoba, která obsluhuje pokladnu. Pokladní je zodpovědná za přijetí správné sumy peněz a systém s tímto počítá.

Další variantou je nákup vstupenky přes webový portál pevnůstky. Tento nákup se od toho na pokladně liší v několika skutečnostech. Protože komunikace se serverem bude probíhat přes internet, předpokládá se určité časové zpoždění mezi platbou a samotným vytvořením vstupenky. Mohla by nastat situace, že již zaplacenou vstupenku nebude možné vytvořit, nebo naopak, že již vytvořená vstupenka nebude zaplacena (pokud zákazník přeruší objednávku). Z tohoto důvodu se při potvrzení objednávky nejprve zkontroluje, zda je možné vstupenku vytvořit, a pokud ano, vstupenka se na určitou, krátkou, dobu zarezervuje. Následně je zákazník přesměrován na platební bránu banky, kde je vybídnut k zaplacení objednané vstupenky. Pokud platba proběhne v pořádku, je na server

poslán požadavek na vytvoření vstupenky. Díky rezervaci vstupenky se nemůže stát, že po zaplacení vstupenky nebude možné vstupenku vytvořit. Pokud zákazník transakci zruší a vstupenku nezaplatí, rezervace se po dané, krátké, době automaticky zruší. Samotné zpracování požadavku na serveru je pak shodné jako v případě s pokladnou. Situaci vystihuje obrázek (Obrázek 3.10).



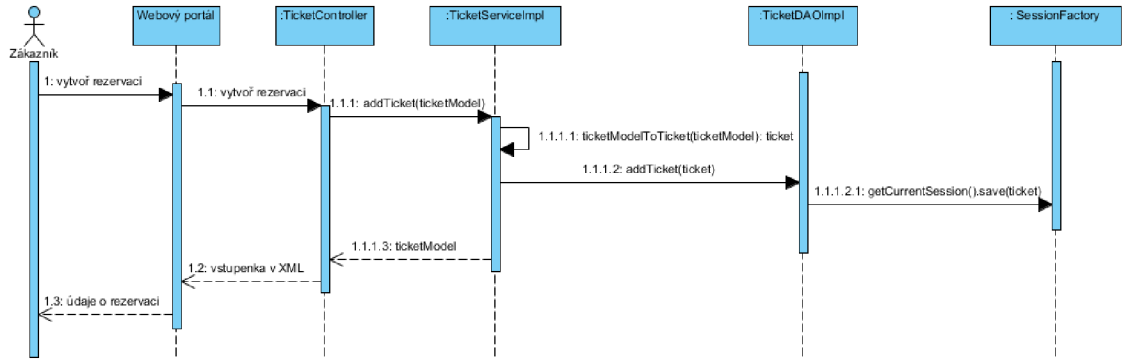
Obrázek 3.10: Zakoupení/prodej vstupenky přes web



Obrázek 3.11: Ref1, Zpracování zakoupení/prodeje vstupenky na serveru

3.5.2 Rezervace vstupenky

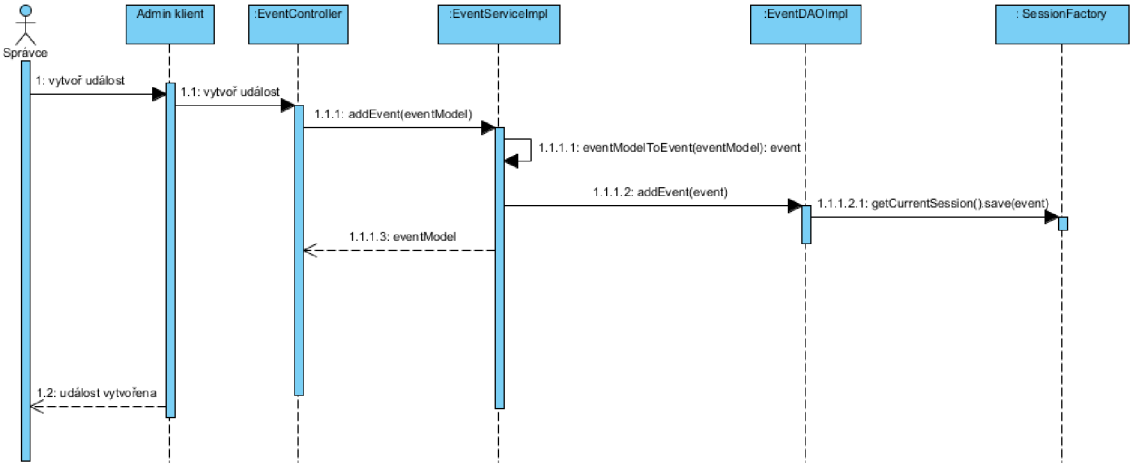
Rezervace vstupenky odpovídá vytvoření záznamu nezaplacené vstupenky v databázi. Každá nezaplacená rezervace bude automaticky zrušena půl hodiny před začátkem události, ke které se vztahuje. Následující diagram (Obrázek 3.12) znázorňuje zpracování požadavku na vytvoření rezervace přes webové rozhraní. Rezervaci je také možné vytvořit na pokladně, zpracování je pak úplně stejné jako v případě webu.



Obrázek 3.12: Rezervace vstupenky

3.5.3 Vytvoření události

Vytvoření nové události bude možné výhradně pomocí administrátorského klienta. Po zadání veškerých údajů o události do formuláře a stisku tlačítka *Vytvořit*, se odešle požadavek na server. Pro vytvoření události je nutné vybrat název události, čas začátku a konce konání události, místnost, ve které bude událost probíhat, a také cenu vstupenek. Průběh zpracování operace je možné vidět na obrázku (Obrázek 3.13).

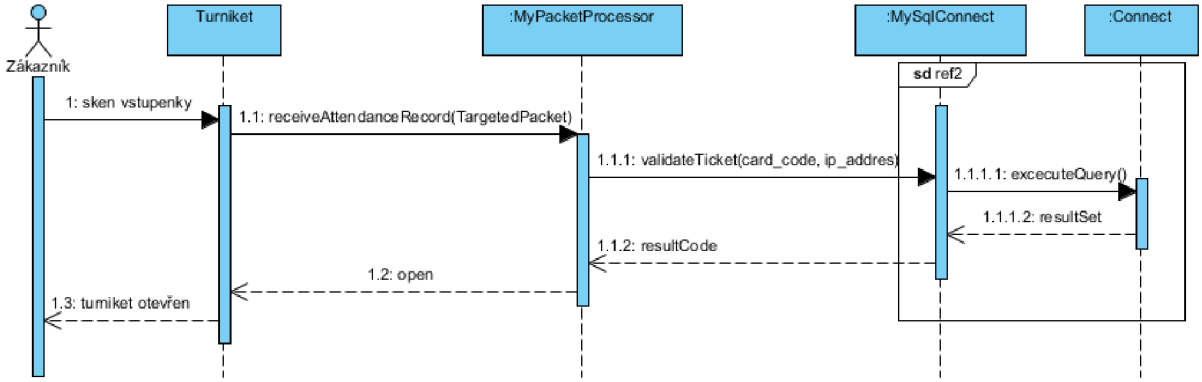


Obrázek 3.13: Vytvoření události

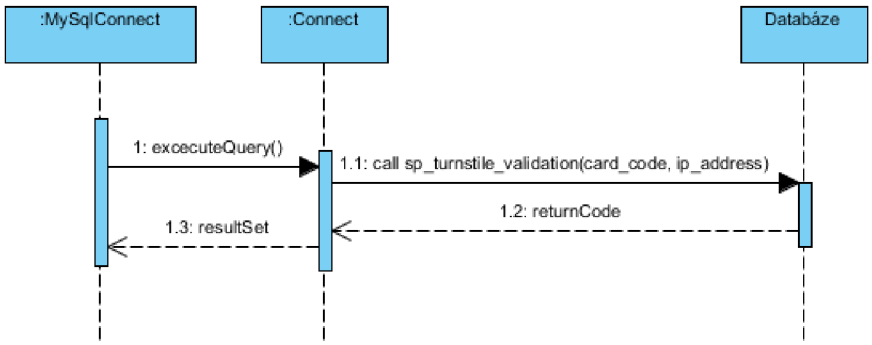
3.5.4 Ověření platnosti vstupenky turniketem

Ovládání turniketů má na starosti ovladač, což je samostatná aplikace, která běží na serveru pevnůstky. Samotné ověření platnosti vstupenky provádí databáze, konkrétně její uložená procedura. Pokud turniket naskenuje vstupenku, odešle její kód ovladači. Ovladač následně zavolá ověřovací uloženou proceduru a jako argumenty jí předá kód vstupenky a identifikátor turniketu, který vstupenku naskenoval. Podle identifikátoru turniketu je možné v databázi dohledat konkrétní událost, která v místnosti probíhá. Procedura tedy zkontroluje, zda vstupenka na tuto událost platí, zda je

vstupenka zaplacená a zda ještě nebyla použita. Pokud je vše v pořádku, procedura odečte vstupence jeden vstup (aktualizuje záznam v tabulce) a vrátí ovladači informaci, že může otevřít turniket. Ovladač následně otevře zákazníkovi turniket a ten může projít. Situaci ukazují následující obrázky (Obrázek 3.14) a (Obrázek 3.15).



Obrázek 3.14: Odeslání požadavku z turniketu do ovladače



Obrázek 3.15: Ref2, Ověření platnosti vstupenky v databázi

3.6 Komunikační rozhraní

V této kapitole je popsáno rozhraní jádra SSS, které využívají jednotlivé moduly pro vzájemnou komunikaci. Velká většina operací je zpracovávána samotným jádrem (serverem) a rozhraní je podle toho uzpůsobeno. Klientské moduly pomocí tohoto rozhraní přímo volají služby (metody) jádra se svými argumenty a získávají odpovídající výsledky. Veškerá komunikace odpovídá modelu z kapitoly 3, který je uveden na obrázku (Obrázek 3.1).

Jádro SSS, jakožto serverová aplikace, bude své služby nabízet přes *REST API*, implementovaného v Javě. Jedná se tedy o webovou aplikaci, přístupnou přes *HTTP* protokol. Přestože budou nabízené služby pro jednotlivé moduly velmi podobné, ne-li totožné, bude pro každý modul vytvořeno jiné rozhraní. Oddělená budou z důvodu možných budoucích změn nebo rozšíření.

Služby jsou nabízeny přes web, nabízené rozhraní je tedy webové. Každá služba má svůj vlastní identifikátor *URI* a každá služba má definovaný způsob volání. Pokud je služba určena pro získání dat ze serveru, je pro její zavolání potřeba použít metodu *GET*. Pokud má služba na starosti

změnu nebo vytvoření nových dat na serveru, použije se metoda *POST*. V následujících podkapitolách je vždy tabulkou popsáno rozhraní serveru s konkrétním modulem. Tabulka vždy obsahuje název služby, identifikátor služby *URI* a metodu, kterou je nutné použít pro její zavolání.

3.6.1 Rozhraní mezi SSS a kasou

Kasa je klientská aplikace, která využívá služeb jádra. Následující tabulka (Tabulka 1) obsahuje všechny podstatné služby, které jsou nezbytné pro provoz kasy. Kvůli zachování přehlednosti je v tabulce vynechána část adresy, která obsahuje adresu serveru, port a název aplikace (*http://adresa_serveru:8080/SSS*).

Název služby	URI	Metoda
Vytvoření rezervace	/kasa/ticket/reserve	POST
Zaplacení rezervace	/kasa/ticket/pay	POST
Zrušení rezervace	/kasa/ticket/deleteReservation	POST
Zobrazit vstupenku v XML	/kasa/ticket/{ticketId}.xml	GET
Zobrazit vstupenku v PDF	/kasa/ticket/{ticketId}.pdf	GET
Zobrazit události	/kasa/event.xml	GET
Zobrazit detail události	/kasa/event/{eventId}.xml	GET
Zobrazit zákazníky	/kasa/customer.xml	GET
Zobrazit zákazníka	/kasa/customer/{customerId}.xml	GET

Tabulka 1: Rozhraní mezi SSS a kasou

Pokud bude chtít klient využít službu s metodou GET, stačí se dotázat na uvedenou URI. Server na takovýto dotaz klientovi odešle XML, případně PDF, dokument, který bude obsahovat požadovaná data. U služeb s metodou POST tomu bude podobně. Klient odešle na uvedenou adresu XML dokument, který obsahuje potřebná data, server požadavek zpracuje (vytvoří požadovaný záznam) a klientovi zpět odešle potvrzení o provedení operace. Struktura předávaných XML dokumentů pro jednotlivé služby je popsána XSD schématem. Ukázka schématu pro vytvoření nové EAN vstupenky je uvedena na následujícím obrázku (Obrázek 3.16). Kompletní schémata všech předávaných dokumentů jsou i s příklady uvedena v příloze na konci tohoto dokumentu.

```
<xs:schema attributeFormDefault="unqualified"
           elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ticket">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="remaining_entrances"/>
        <xs:element type="xs:string" name="valid_from" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:string" name="valid_to" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:string" name="type"/>
        <xs:element type="xs:string" name="ean_type" minOccurs="0" maxOccurs="1"/>
        <xs:element name="events">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:byte" name="item" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek 3.16: XSD schéma nové EAN vstupenky

3.6.2 Rozhraní mezi SSS a webem

Web je další klientská aplikace, která pouze využívá služeb jádra SSS. V následující tabulce jsou uvedeny důležité služby, které web bude využívat. Podobně jako u kasy, komunikace probíhá výhradně prostřednictvím XML dokumentů. Struktura těchto dokumentů je stejná jako v případě kasy.

Název služby	URI	Metoda
Vytvoření rezervace	/web/ticket/reserve	POST
Zaplacení rezervace	/web/ticket/pay	POST
Zrušení rezervace	/web/ticket/deleteReservation	POST
Zobrazit vstupenku v XML	/web/ticket/{ticketId}.xml	GET
Zobrazit vstupenku v PDF	/web/ticket/{ticketId}.pdf	GET
Zobrazit všechny události	/web/event.xml	GET
Zobrazit detail události	/web/event/{eventId}.xml	GET
Zobrazit všechny zákazníky	/web/customer.xml	GET
Zobrazit detail zákazníka	/web/customer/{customerId}.xml	GET

Tabulka 2: Rozhraní mezi SSS a webem

3.6.3 Rozhraní mezi SSS a ručními čtečkami

Vstupenky budou mezi jádrem a čtečkou předávány pomocí XML souboru. Pro export vstupenek do čtečky se kódy vstupenek nejprve stáhnou z jádra do souboru, který se uloží na místní disk počítače správce. Tento soubor se pak nahraje přímo do zařízení ruční čtečky. Pokud mají být vstupenky naopak importovány ze čtečky do jádra SSS, postup je přesně opačný. Nejprve se stáhne soubor ze zařízení čtečky na disk a pak se tento soubor odešle na server (jádro SSS). V následující tabulce (Tabulka 3) jsou uvedeny služby, které umožňují stažení (a nahrání) vstupenek ze serveru (nebo na server). V příloze III je také uvedena struktura předávaných XML souborů i s příklady.

Název služby	URI	Metoda
Stáhnout vstupenky do čtečky	/handheld/ticket/{eventId}/download	GET
Nahrát vstupenky ze čtečky	/handheld/ticket/{eventId}/upload	POST

Tabulka 3: Rozhraní mezi SSS a ručními čtečkami

3.6.4 Rozhraní mezi SSS a FlexiBee

Z pohledu jádra SSS je *FlexiBee* jediný modul, který se chová jako server. Jádro SSS je v pozici klienta, který využívá služeb softwaru *FlexiBee*. Tento účetní software je napsán jako serverová aplikace v Javě a své služby také nabízí přes *REST API* pomocí protokolu *HTTPS*. Následující tabulka shrnuje všechny podstatné služby, které jádro ve svém běžném provozu bude využívat.

Název služby	URI	Metoda
Vytvořit fakturu	/c/pp/faktura-vydana.xml	POST
Zobrazit fakturu	/c/pp/faktura-vydana/{id}.xml	GET
Vytvořit prodejku	/c/pp/prodejka.xml	POST
Zobrazit prodejku	/c/pp/prodejka/{id}.xml	GET
Zobrazit sumu v pokladně	/c/pp/pokladni-pohyb/$sum	GET

Tabulka 4: Rozhraní mezi SSS a FlexiBee

3.7 Grafické uživatelské rozhraní

Součástí návrhu jsou také náhledy na obrazovky výsledného systému. Tyto náhledy jsou pouze předběžné a slouží pro získání hrubé představy o tom, jak bude systém vypadat. Výsledná grafická podoba systému se může od těchto ukázek mírně lišit. Z důvodu úspory místa jsou všechny náhledy přesunuty do přílohy I tohoto dokumentu (na straně 42).

3.7.1 Administrátorské rozhraní

Jako první je uvedena přihlašovací obrazovka (Obrázek I.1). Dále jsou uvedeny obrazovky klienta pro správce systému (Obrázek I.2 až Obrázek I.18). Jak je možné vidět, grafické rozhraní je tvořeno převážně pomocí tabulek a formulářů. Tabulky je možné řadit podle hodnot sloupců, u nichž je uveden symbol  . Zobrazená data je možné také filtrovat podle sloupců, u nichž je uveden symbol filtru (). Přehledové obrazovky, například "zobrazení všech zákazníků" (Obrázek I.14), obsahují pouze důležité informace. Stisknutím tlačítka *Detail*, se zobrazí veškeré detaily o konkrétním záznamu, například adresa, procentuální sleva atd. Pokud si uživatel zobrazí detail nějakého záznamu, může na této stránce záznam změnit nebo ho smazat.

3.7.2 Kasa

Předpokladem je, že aplikace bude provozována na dotykovém monitoru, proto je grafické rozhraní co možná nejjednodušší a proto jsou grafické ovládací prvky velké. Na prvním obrázku (Obrázek I.19) je možné vidět hlavní okno kasy. V levé části okna je seznam zvolených vstupenek. U každé z nich je vidět seznam akcí, na které budou platit, jejich počet a cena. Například první uvedená vstupenka bude platit jak na první, tak i na druhou expozici. Ostatní vstupenky budou platit vždy na jedinou událost a budou prodány po dvou kusech každá. Počet prodaných kusů bude možné nastavovat označením vstupenky a následným kliknutím na číslo na zobrazené číselné klávesnici nebo klikáním na tlačítka '+' a '-' také na této klávesnici. Na pravé straně okna jsou, kromě již zmíněné klávesnice, tlačítka pro přidání vstupenky na expozici, dále tlačítko pro výběr akcí a také je zde okno pro vyhledání zákazníka. Zákazníka bude možné vyhledat podle čísla jeho zákaznické karty a to buď zadáním tohoto čísla do vyhledávacího boxu pomocí klávesnice, nebo naskenováním jeho vstupenky pokladním skenerem.

Stisknutím tlačítka *Cash* nebo *Karta* se zahájí prodej vstupenek. V prvním případě se objeví dialogové okno, ve kterém bude vidět požadovaná částka, přijatá částka a částka pro navrácení zákazníkovi. Také bude otevřen šuplík pro bankovky a po potvrzení zaplacení budou vstupenky odeslány na tiskárnu. Ve druhém případě se objeví informační okno, které bude zobrazovat stav platby, a bude kontaktován terminál pro platbu kartou. Po úspěšném provedení platby se také vstupenky odešlou na tiskárnu.

Po stisknutí tlačítka pro výběr akce (tlačítko *Akce…*) se obsah okna aplikace změní na podobu uvedenou na dalším obrázku (Obrázek I.20). Cílem tohoto okna je umožnit rychle vyhledat konkrétní událost a stiskem tlačítka *Přidat* vytvořit vstupenku na tuto událost, aby bylo možné ji prodat. V levé části tohoto okna je seznam událostí, které probíhají nebo které probíhat budou. U každé události je zobrazeno datum a čas zahájení, počet volných míst a také cena. Pokud je jako počet volných míst zobrazena hodnota „-1“, pak tato událost není omezena co do počtu návštěvníků. Na takovouto událost je možné prodat libovolný počet vstupenek. Tyto události je možné filtrovat podle jejich charakteru a k tomuto účelu slouží tlačítka na pravé straně okna. Například po stisku tlačítka *Planetárium* se vlevo zobrazí pouze ty události, které budou probíhat v planetáriu.

Při každém přidání události (stisk tlačítka *Expozice* nebo *Přidat*) se vytvoří nová vstupenka. Pokud je požadováno, aby se událost přidala na některou, již existující, vstupenku jako další vstup, je nejdříve nutné tuto vstupenku v seznamu označit kliknutím. První ukázaná vstupenka (Obrázek I.19) se tedy například vytvoří tak, že se klikne na tlačítko Expozice 1, pak se klikne na nově vytvořenou vstupenku v levém okně a pak se klikne na tlačítko Expozice 2.

4 Implementace

Vzhledem k rozsáhlosti systému byla implementace rozdělena do několika částí, které bylo možné vyvíjet nezávisle na sobě. Toto rozdělení vycházelo přímo z návrhu a tedy ovladač turniketů, jádro systému a kasa byly vyvíjeny jako tři samostatné aplikace, které pak budou spolupracovat. Pro vývoj aplikací v *Javě* bylo zvoleno prostředí *Eclipse* [11] (*Kepler*) a sestavovací nástroj *Maven* [12]. Aplikace, psané v jazyce *C#*, byly vyvíjeny v prostředí *Microsoft Visual Studio 2013* [13].

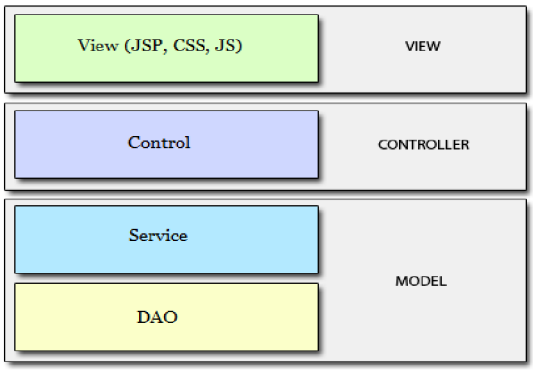
4.1 Ovladač turniketů

Nejprve bylo potřeba vytvořit společný základ, databázi, a tuto databázi pak naplnit vzorovými daty. Ovladač turniketů byl vytvářen jako *Java SE* aplikace, která obsahuje kód pro komunikaci s turnikety pomocí *UDP* protokolu a také kód pro komunikaci s databází. Dodavatel statických turniketů je firma *ELVIS* [10], která k nim také poskytla ovládací kód v Javě. Tento kód posloužil jako bytelný základ ovladače, k němuž postačilo přidat komunikaci s databází. Logika, která rozhoduje o tom, zda turniket otevřít nebo neotevřít byla přesunuta do databáze a byla implementována jako uložená procedura. Při obsloužení požadavku turniketu ovladač pouze zavolá uloženou proceduru v databázi, předá jí jako parametry IP adresu turniketu, adresu snímače turniketu (každý turniket má alespoň dva snímače) a kód naskenované vstupenky. Návratovou hodnotou procedury je buď číslo nula (otevřít turniket), nebo kladné číslo – kód chyby (vypsat chybovou hlášku na displej turniketu a turniket nechat uzavřený).

Tento ovladač a jádro systému jsou jediné dvě aplikace, které mají přímý přístup do databáze. Hlavním správcem databáze je jádro, ovladač turniketů bude pouze aktualizovat záznamy naskenovaných vstupenek. Konkrétně bude upravovat zbývající počet vstupů vstupenky (bude jej snižovat o jedna) a také bude vstupence nastavovat aktuální datum změny záznamu. Protože mají obě aplikace možnost měnit záznam vstupenky, budou si teoreticky navzájem konkurovat. Aby byla zajištěna konzistence databáze, je procedura implementována jako transakce.

4.2 Jádro systému SSS

Implementace vycházela z návrhu z předchozí kapitoly. Rozdělení do vrstev se při implementaci ukázalo jako velká výhoda, celá aplikace je rozdělena na jednotlivé logické části, které mezi sebou komunikují pouze pomocí svého rozhraní. Bylo tedy možné každou část implementovat a testovat zvlášť, aniž by byla ovlivněna část jiná. Detailní pohled na realizaci MVC vrstev je na následujícím obrázku (Obrázek 4.1).



Obrázek 4.1: Vrstvy aplikace

Implementace byla vedena od nejnižší vrstvy, od modelu, směrem k vrstvám vyšším. Podle databáze byly nejprve vytvořeny entitní třídy, které přímo odpovídají tabulkám v databázi. S využitím rámce *Hibernate* pak byla implementována vrstva pro komunikaci s databází, vrstva *DAO*. Následně byly souběžně vytvářeny ostatní vrstvy (*Service*, *Control*, *Wiew*). Každá pohledová stránka (*JSP*) pracuje s tzv. *managed bean*, tedy s instancí modelové třídy. Tato modelová třída obsahuje data, která je možné na stránce zobrazit nebo změnit. Každá stránka pracuje pouze s jedinou modelovou třídou. Pokud bychom uvažovali o tom, že jako *managed bean* budeme používat přímo entitní třídy, narazíme na problém. Například vstupenka je v databázi uložena vždy ve dvou tabulkách, v první jsou uloženy obecné informace a ve druhé jsou pak uloženy specifické informace podle typu vstupenky. Vstupenka se z databáze načte do dvou entitních tříd a nebylo by tedy možné ji jednoduše zobrazit nebo upravit na jediné *JSP* stránce. Z tohoto důvodu byly vytvořeny prezentační modely, které obsahují vždy kompletní data. Například třída *TicketModel* tedy obsahuje veškeré informace o vstupence. Tyto prezentační modely jsou pak použity jako *managed beany*.

Servisní vrstva má na starosti převod entitních tříd (získaných z databáze) na prezentační modely a zpět. Vrstva *Controller* zpracovává veškeré klientské požadavky, které jsou určeny jedinečným URI. Každý takový požadavek zpracovává jedna konkrétní metoda některého *controlleru* (instance třídy vrstvy *Controller*). Vrstva *View* je implementována jako webové (*JSP)* stránky, které jsou stylované pomocí *CSS* a podpořené *jQuery* skripty.

Pro přístup do systému je nutné mít uživatelský účet a být přihlášen. Pro zabezpečení byl použit rámec *spring security* [14]. Navíc oproti návrhu byly zavedeny role *WEB* a *KASA*, kde tyto role budou mít uživatelé, kteří budou chtít do systému přistupovat pomocí rozhraní pro web nebo kasu.

Vytvořené rozhraní pro komunikaci s klienty je založeno na vzájemném předávání XML, případně PDF, dokumentů. Pro práci s XML je použitý rámec *JAXB* [15]. Tento nástroj umožňuje téměř automaticky mapovat objekty do XML podoby a také naopak vytvářet objekty z poskytnutého XML dokumentu. U některých tříd převod nebyl tak přímočarý a bylo nutné implementovat pomocné třídy a převodní metody (*un/marshall*), veškerý související kód je umístěn v balíku *xmlAdapter*. Vstupenky bylo potřeba generovat také do PDF formátu pro tisk, včetně čárového kódu pro turnikety.

Pro převod vstupenek do PDF byl použit rámec *FOP* [16] a pro generování čárových kódů byl využit rámec *Barcode4j* [17]. Ukázka vytvořených vstupenek ve formátu PDF je uvedena na následujícím obrázku (Obrázek 4.2). Vstupenka uvedená vlevo platí celkem na tři události, pravá vstupenka je platná pouze na jednu událost.

Obrázek 4.2: PDF vstupenky pro tisk

4.3 Kasa

Na rozdíl od jádra, byl pro kasu zvolen opačný postup implementace. Nejprve bylo vytvořeno grafické uživatelské rozhraní (GUI), které bylo během jeho tvorby konzultováno se zadavatelem projektu. Teprve pak byly postupně oživovány jednotlivé ovládací prvky a byly vytvářeny nižší vrstvy aplikace. Tedy podle požadavků budoucího uživatele programu se vytvářela jeho logická funkčnost. Vzhled a způsob ovládání této aplikace je velmi důležitý, proto byl zvolen právě tento postup.

Kasa je rozdělena do stejných logických vrstev jako jádro. Servisní vrstva překládá entitní objekty (přijímané ze serveru) na prezentační modely (v tomto případě jsou jimi položky *ListView*) a zpět. Vrstva controlleru obsluhuje požadavky pohledové vrstvy a předává data mezi pohledovou a modelovou vrstvou. DAO vrstva se stará o komunikaci se serverem, přijímá data ve formátu XML a převádí je na odpovídající objekty – entity. Pro implementaci téměř veškeré funkcionality byly použity knihovny dostupné v *.NET Frameworku 4.5* a pro správný chod aplikace je tento také vyžadován. K implementaci komunikace se serverem byla použita knihovna *RestClient* [19], kterou její autor nabídl volně ke stažení.

5 Testování

Testování jednotlivých aplikací probíhalo průběžně, během jejich implementace. Veškeré testy probíhaly na počítačích s následujícími konfiguracemi:

Komponenta	Model
Základní deska	GIGABITE GM-ES2L
Procesor	Intel Core 2 Duo E6400
Grafická karta	Intel GMA 4500(M)(HD)
RAM	2x Transcend JM2GDDR2-6K (2GB celkem)
Disk	WDC WD800BEVS-22RST0 [80GB, 8MB]
Síťová karta	RealTek RTL8168C/8111C PCI-E Gigabit Ethernet Adapter
Operační systém	Ubuntu 12.04 LTS Precise Pangolin

Tabulka 5: Testovací sestava pro jádro a ovladač

Komponenta	Model
Základní deska	LENOVO 6885B8G
Procesor	Intel Core i5-3230M
Grafická karta	Intel HD Graphics 4000
RAM	SK Hynix HMT41GS6A FR8A-PB 8GB (2x)
Disk	WDC WD10JPVX-08JC3T2 [1000 GB, 8MB]
Síťová karta	RealTek RTL8168/8111 PCI-E Gigabit Ethernet NIC
Operační systém	MS Windows 8.1 (x64) Build 9600

Tabulka 6: Testovací sestava pro kasu

Zvolený testovací software je stejný jako ten, který bude k dispozici na cílových provozních stanicích, tedy na serveru muzea a na pokladním počítači. Provedené testy proto mají reálnou vypovídající hodnotu, co do softwarové kompatibility. Reálnou provozní zátěž serveru nebo ovladače na těchto zařízení otestovat nebylo možné. Zátěžové testy budou provedeny přímo v muzeu, jakmile bude nainstalován potřebný hardware.

5.1 Ovladač turniketů

Pro možnost průběžného testování implementovaného ovladače zapůjčila firma *ELVIS* elektroniku z jednoho statického turniketu. Elektronika je lehce upravená tak, aby ji bylo možné přenášet v relativně malé krabici. Pro simulaci otáčení ramen turniketu má napájená tlačítka na plošném spoji (zmáčknutí tlačítka odpovídá otočení ramene turniketu) a namísto čtečky čárových kódů, která je běžně umístěna v kostře turniketu, je připojena ruční čtečka. Pro představu je v příloze tohoto dokumentu přiložena fotografie této elektroniky (Obrázek IV.1).

Díky tomuto poskytnutému hardwaru bylo možné ověřit správnost implementace, tedy zda se ovladač chová dle požadavků. V tomto testovacím prostředí pracoval ovladač jak má, v reálném provozu bude ovšem ovladač obsluhovat turniketů hned několik. Reálnou zátěž bude možné vyzkoušet až v muzeu samotném, až budou nainstalovány servery a zapojeny turnikety. Technické zázemí (hardware serverů, síťová realizace) bude v muzeu dostatečně výkonné, takže bude ovladač, jakožto více vláknová aplikace, zcela jistě pracovat svižně, s prakticky nulovou odezvou.

5.2 Jádro SSS

Jakákoli manipulace nebo komunikace s jádrem systému je možná výhradně pomocí jeho rozhraní, které nabízí. Systém byl proto testován průběžně podle toho, jak toto rozhraní vznikalo a rozšiřovalo se. Jako hlavní testovací nástroj byla použita konzolová aplikace *cURL* [20], která umožňuje přijímat a odesílat data pomocí různých protokolů, včetně *HTTP* nebo *HTTPS*. Díky tomu, že se tento nástroj spouští v konzoli, je možné jej spouštět například v dávce pomocí *BAT* skriptů a je takto možné testy automatizovat a také je opakovat. Tímto způsobem byla testována především rozhraní pro web a kasu. Správná funkčnost administrátorského rozhraní byla testována ručně pomocí internetových prohlížečů (konkrétně Chrome, Firefox, Opera, Internet Explorer a Safari) a pomocí nástroje pro správu databáze *PhpMyAdmin*. Během svého provozu bude jádro obsluhovat velké množství požadavků v malém časovém měřítku. Bylo by tedy vhodné provést zátěžové testy, které by ukázaly reálné možnosti této aplikace. Vzhledem k výkonu hardware, který bude v muzeu instalován, by však odezva systému měla být dostatečně krátká i při velkém množství připojených klientů.

5.3 Kasa

Během testování této aplikace byl kladen důraz především na použitelnost grafického uživatelského rozhraní. Hlavním parametrem byla dostatečná rychlost odezvy, aby bylo možné vstupenky vytvářet svižně a bez čekání. U většiny událostí je omezený počet míst a proto je potřeba při každém prodeji vstupenky stahovat ze serveru aktuální informace o počtu volných míst. Navíc během vytváření vstupenky se tato nejdříve v systému rezervuje (aby ji před zaplacením nemohl odkoupit někdo jiný například přes internet) a teprve po zaplacení se vytvoří plnohodnotný záznam vstupenky. Pro vytvoření vstupenky jsou tedy potřeba celkem tři dotazy na server: dotaz zda je volné místo, dotaz pro vytvoření rezervace a dotaz pro vytvoření vstupenky. Bylo nutné otestovat, zda takovéto nároky na síťovou komunikaci neovlivní odezvu aplikace. Testy, které byly provedeny na výše uvedených počítačových sestavách, podaly uspokojující výsledy, reakce byly okamžité. Přesto bude ještě nutné, podobně jako u jádra a ovladače, kasu ještě řádně otestovat na cílovém hardware přímo v muzeu.

Závěr

Datum otevření Pevnosti poznání bylo původně stanovené na začátek července roku 2014. Bohužel, po řadě nešťastných příhod, byl tento termín posunut až na konec roku 2014. Hlavním důvodem odkladu bylo pozdržení samotné stavby objektu muzea, čímž došlo také k pozdržení nákupu a instalace technického vybavení interiéru muzea. Díky tomuto systém ještě nebyl uveden do reálného provozu a nebyl podroben zátěžovým testům, přestože je hotový, nachystaný a připravený k použití.

Během této práce byly postupně sbírány požadavky od zadavatele projektu a podle těchto požadavků byl vypracován návrh informačního systému. V rámci této práce byla také vytvořena implementace tohoto systému tak, aby splňovala nejen požadavky na chování (rezervace a prodej vstupenek, správa událostí apod.), ale také tak, aby výsledný systém fungoval s dodaným hardwarem, především s turnikety.

Struktura zadání této práce odpovídá vodopádovému modelu vývoje. V praxi se však tento model nedá striktně dodržet a během tvorby této práce dodržován nebyl. Průběh prací spíše odpovídal modelu spirálovému. V průběhu plnění práce se požadavky na výsledný systém, především na začátku, výrazně měnily. Podle nových požadavků se pak musel přizpůsobit návrh, podle nového návrhu se musela upravit implementace a následně bylo nutné implementovaný kód (nový i ten starý) znovu otestovat. Dokonce i nyní, když už je systém provozu schopný, přibyly požadavky nové. Například by nově mělo být možné prodávat pomocí aplikace kasy kromě vstupenek také drobné předměty nebo by mělo být možné otevřít libovolný turniket stisknutím jediného tlačítka na kase. Možností rozšíření tohoto systému a pokračování v této práci bude zcela jistě spousta. Kromě výše zmíněných úprav by bylo také možné například implementovat administrátorské rozhraní pro libovolnou platformu, například jako desktopovou nebo mobilní aplikaci, nebo jej přidat jako součást aplikace kasy. Také by bylo možné přidat do prostor muzea počítač se čtečkou čárových a RFID kódů, aby si zákazníci mohli zkontrolovat stav své vstupenky. Díky dobré struktuře, vrstvené architektuře a implementovanému *REST API* by mělo být jednoduché tento modulární systém jakkoli rozšířit.

Literatura

[1] WWW stránka: Pevnost poznání [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.pevnostpoznani.cz>

[2] WWW stránka: Spring framework [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.spring.io>

[3] WWW stránka: Hibernate Framework [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.hibernate.org>

[4] WWW stránka: FlexiBee: Internetové ekonomické systémy [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.flexibee.eu>

[5] WWW stránka: POHODA – Účetní program [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.stormware.cz/pohoda/>

[6] WWW stránka: KelSQL ekonomický systém [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.keloc-software.cz/produkty/kelsql>

[7] WWW stránka: Účetní program Money S3 pro menší společnosti a živnostníky [online]. [cit. 2014-01-10]. Dostupné na URL: <http://www.money.cz/money-s3/>

[8] WALLS, Craig. *Spring in action*. 3rd ed. Shelter Island: Manning, c2011, xxiii, 400 s. ISBN 19-351-8235-8.

[9] WWW stránka: Nette framework [online]. [cit. 2014-05-08]. Dostupné na URL: <http://nette.org/cs/ >

[10] WWW stránka: ELVIS [online]. [cit. 2014-05-19]. Dostupné na URL: < http://www.elvi.cz/ >

[11] WWW stránka: Eclipse [online]. [cit. 2014-05-19]. Dostupné na URL: <http://www.eclipse.org/ >

[12] WWW stránka: Apache Maven [online]. [cit. 2014-05-19]. Dostupné na URL: < http://maven.apache.org/ >

[13] WWW stránka: Visual Studio Professional 2013 [online]. [cit. 2014-05-19]. Dostupné na URL: <http://www.microsoftstore.com/store/mseea/cs_CZ/pdp/Visual-Studio-Professional-2013/productID.288483200>

[14] WWW stránka: Spring security [online]. [cit. 2014-05-19]. Dostupné na URL: <http://projects.spring.io/spring-security/>

[15] WWW stránka: Java Architecture for XML Binding (JAXB) [online]. [cit. 2014-05-19]. Dostupné na URL: <http://www.oracle.com/technetwork/articles/javase/index-140168.html>

[16] WWW stránka: Apache FOP [online]. [cit. 2014-05-19]. Dostupné na URL: <http://xmlgraphics.apache.org/fop/>

[17] WWW stránka: Barcode4j [online]. [cit. 2014-05-19]. Dostupné na URL: <http://barcode4j.sourceforge.net/index.html>

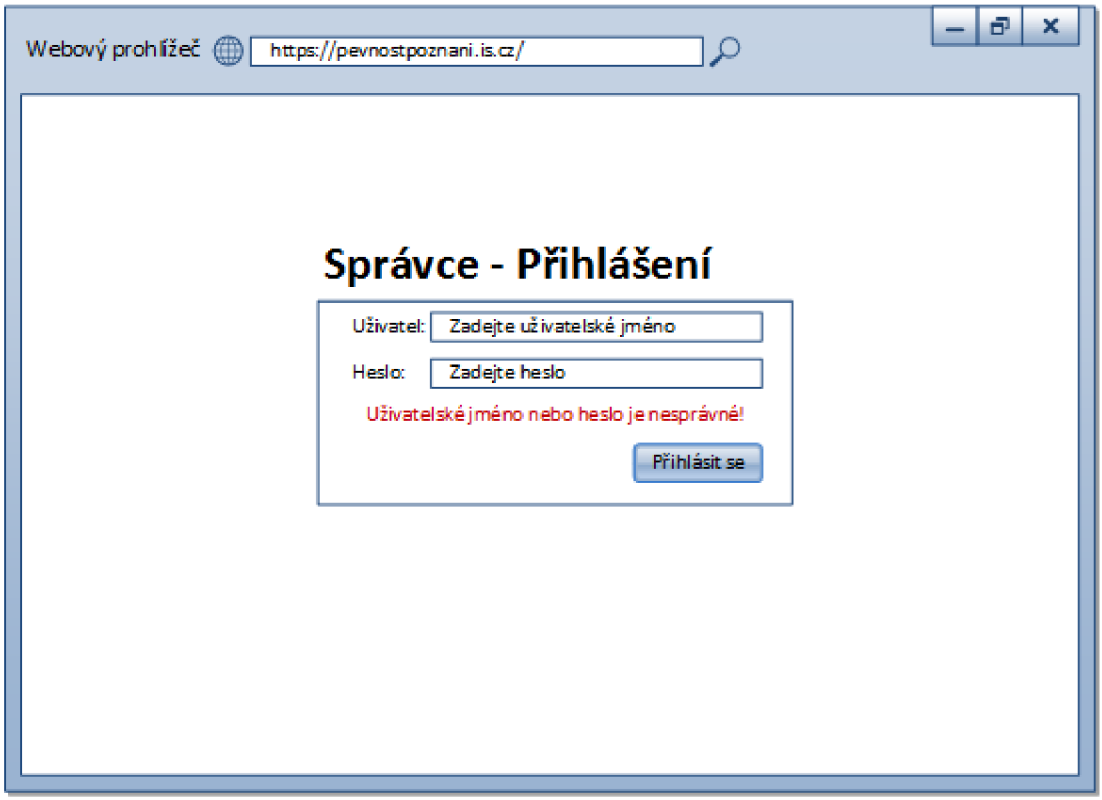
[18] WWW stránka: Windows Presentation Foundation [online]. [cit. 2014-05-19]. Dostupné na URL: <http://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx>

[19] WWW stránka: How to make REST requests with C# [online]. [cit. 2014-05-21]. Dostupné na URL: <http://www.codeproject.com/Tips/497123/How-to-make-REST-requests-with-Csharp>

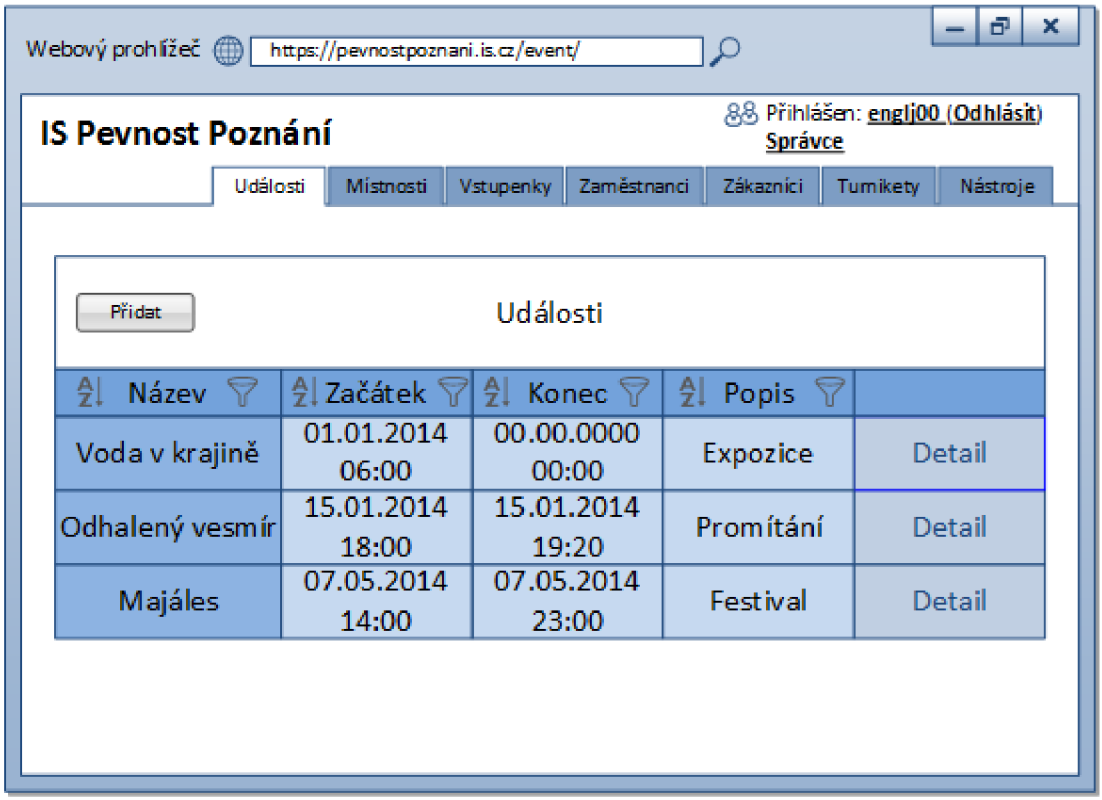
[20] WWW stránka: cURL [online]. [cit. 2014-05-21]. Dostupné na URL: <http://curl.haxx.se/>

Seznam příloh

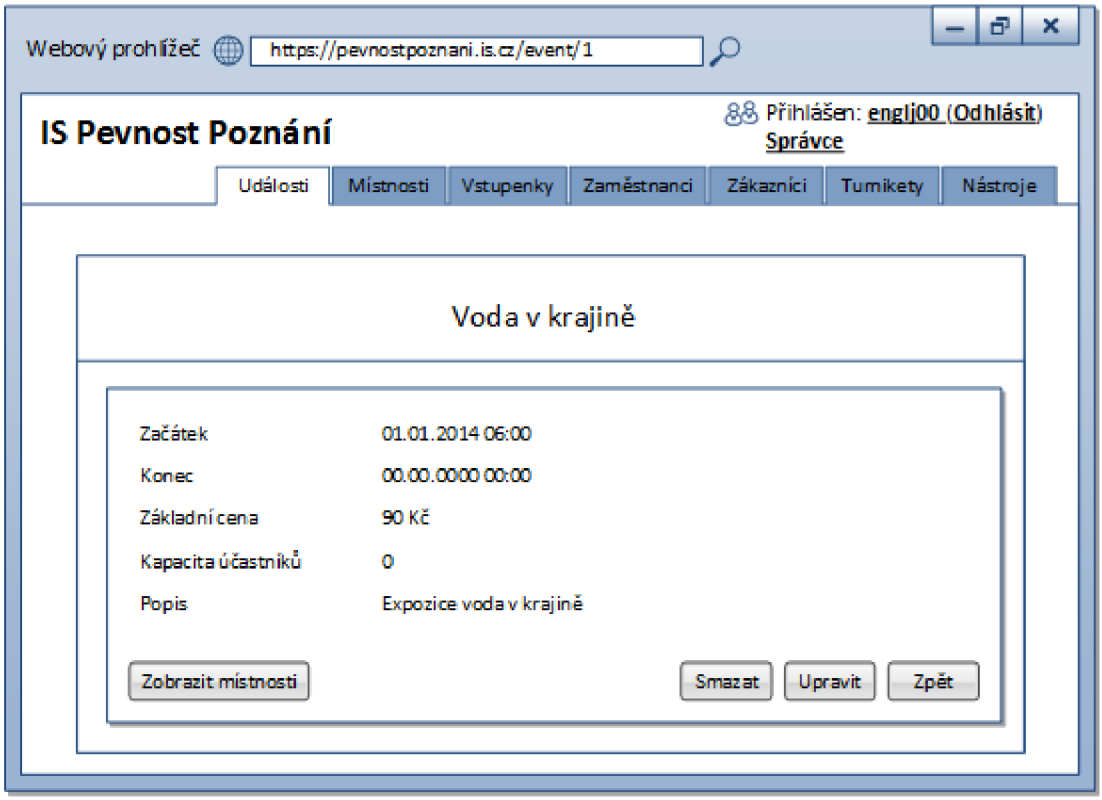
I. Grafické náhledy obrazovek



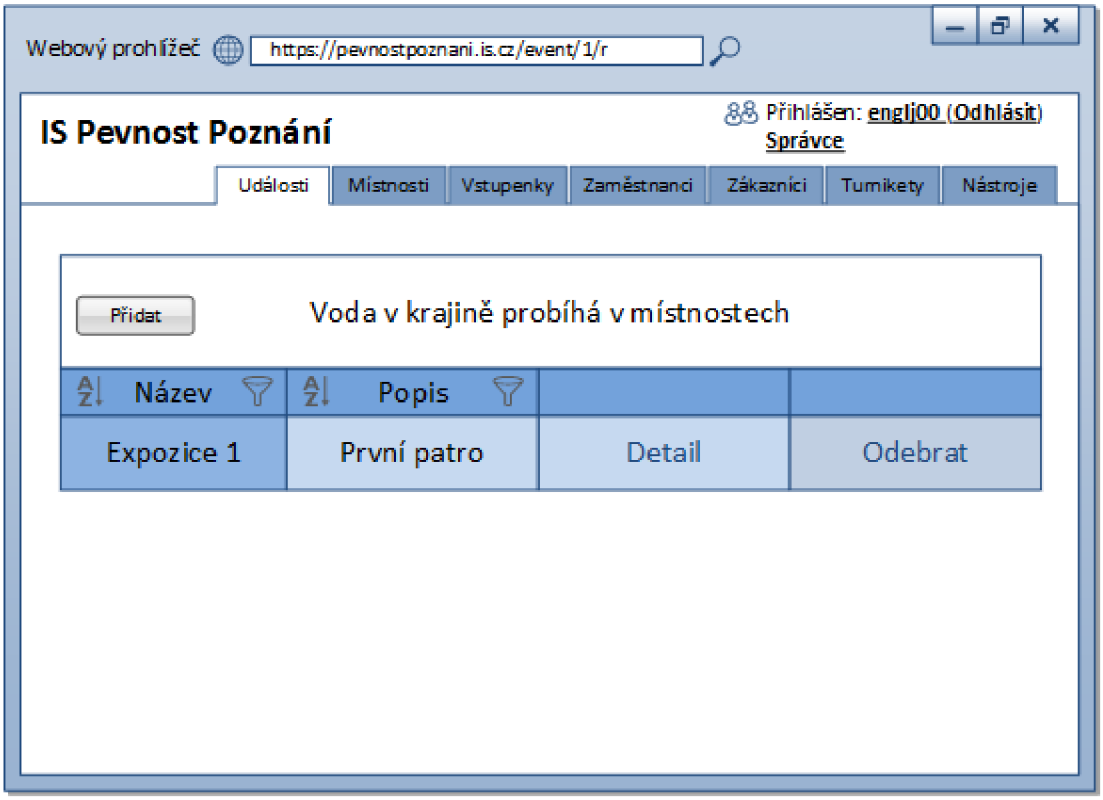
Obrázek I.1: Přihlášení uživatele



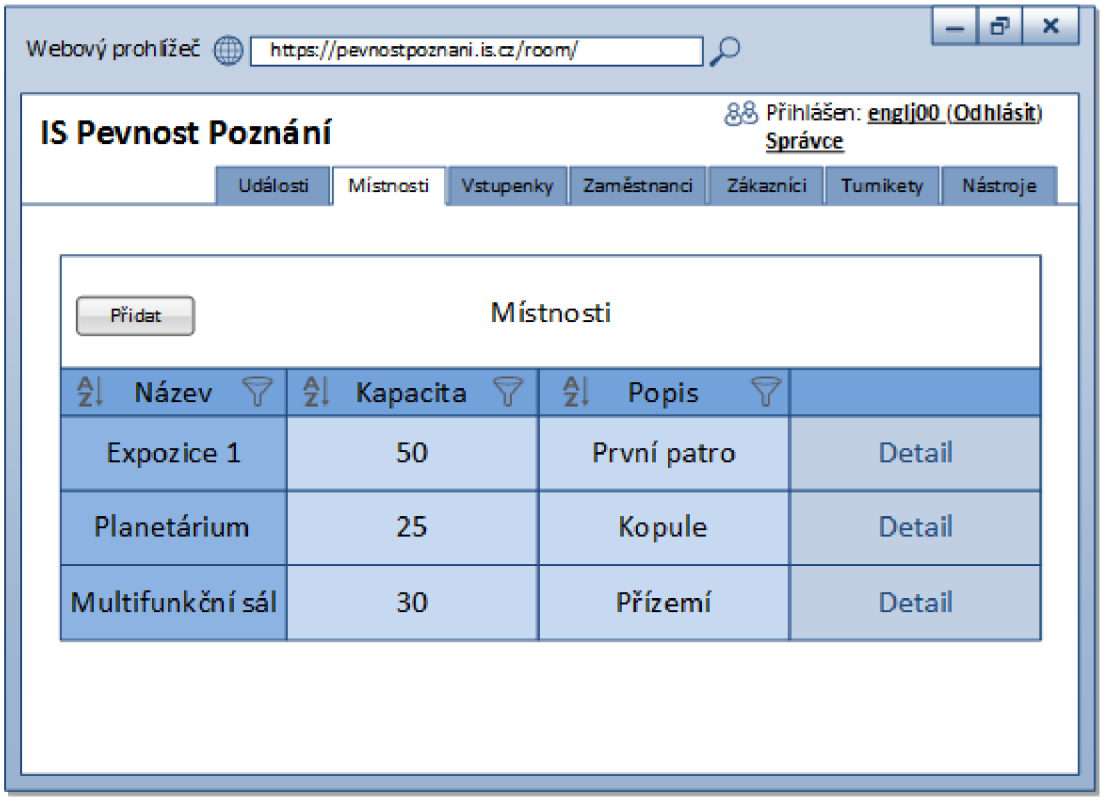
Obrázek I.2: Správce, zobrazení všech událostí



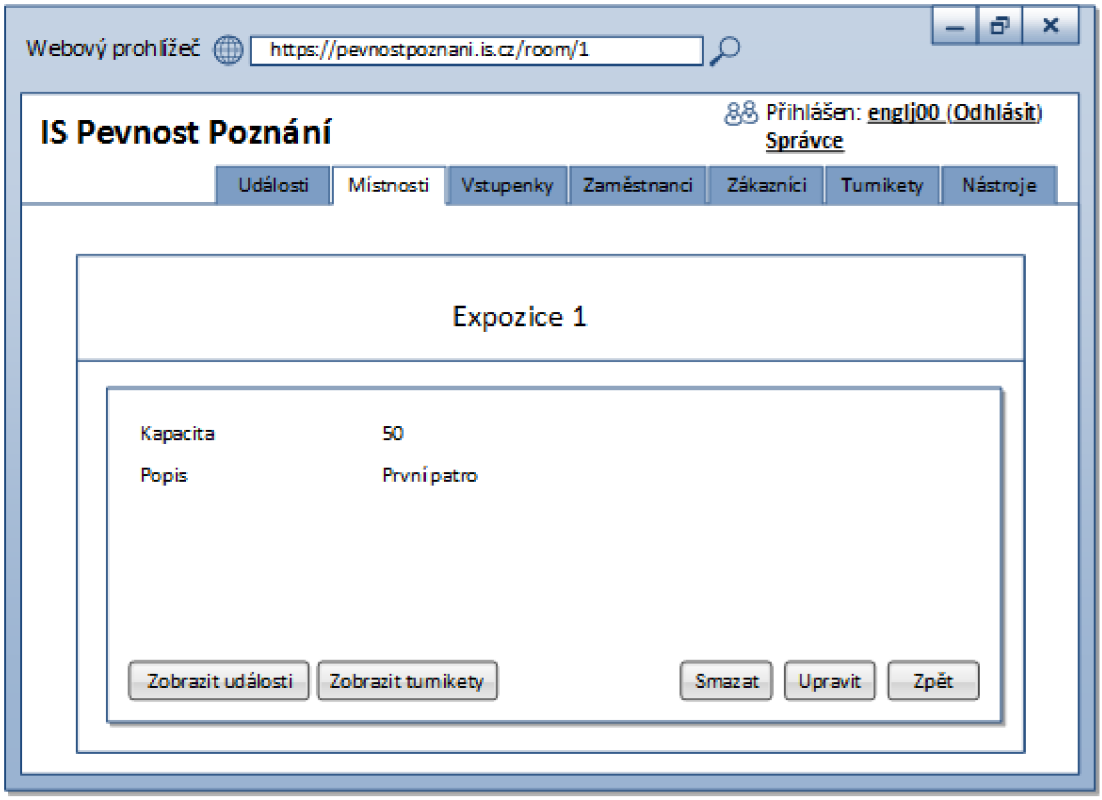
Obrázek I.3: Správce, detail události



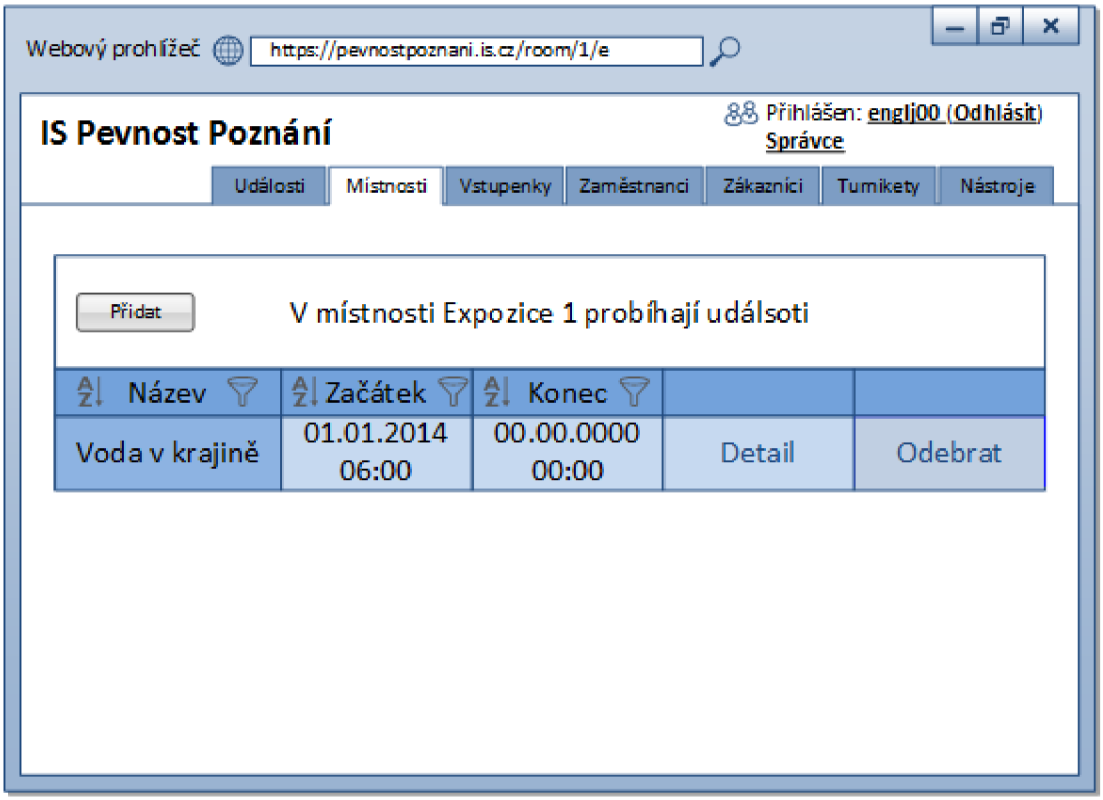
Obrázek I.4: Správce, detail události, seznam místností



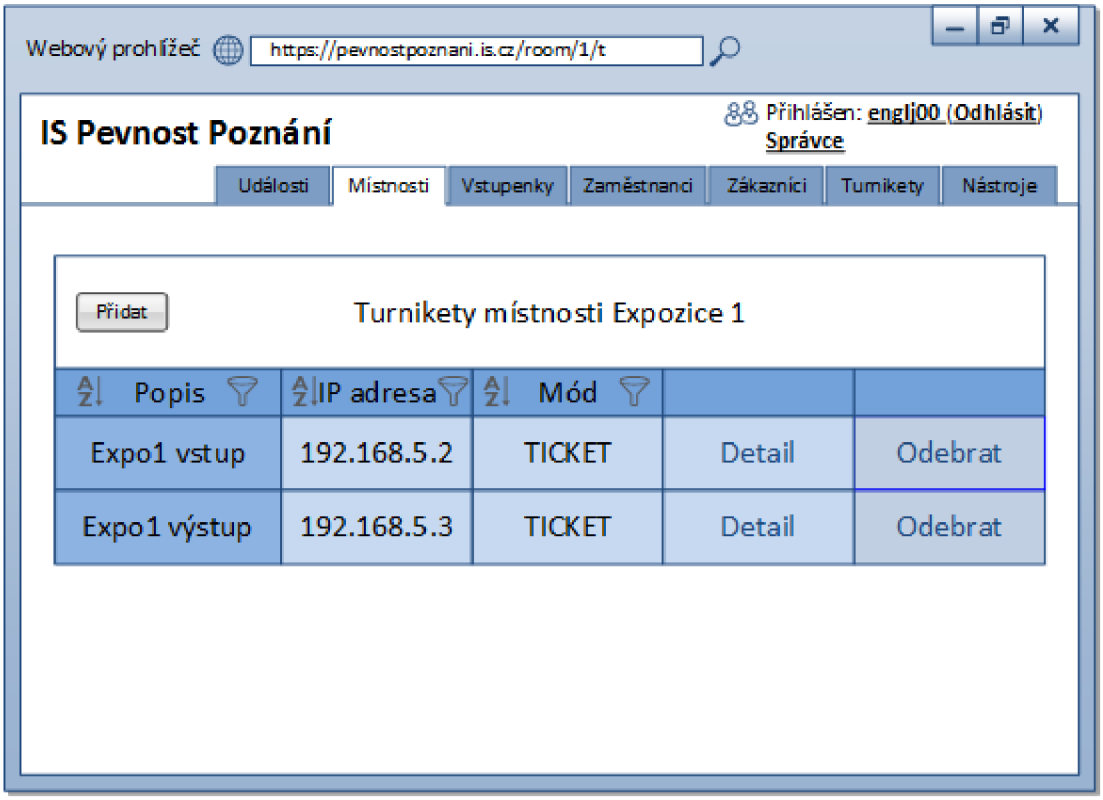
Obrázek I.5: Správce, zobrazení všech místností



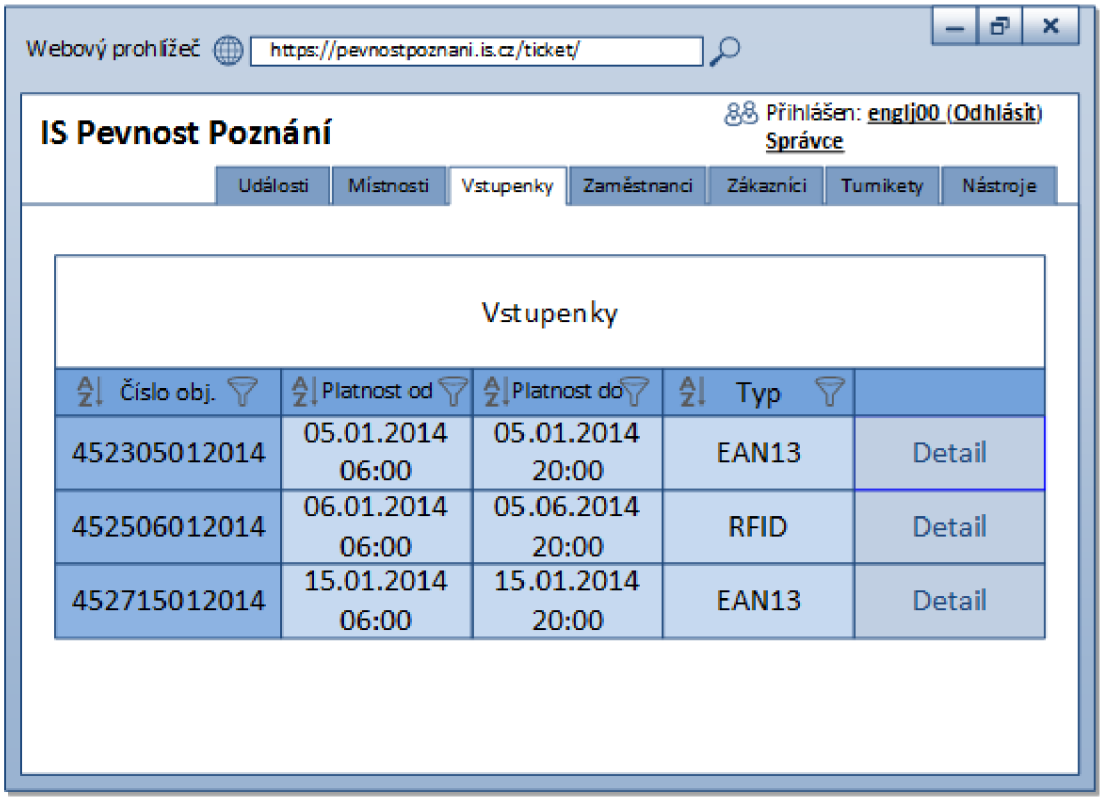
Obrázek I.6: Správce, detail místnosti



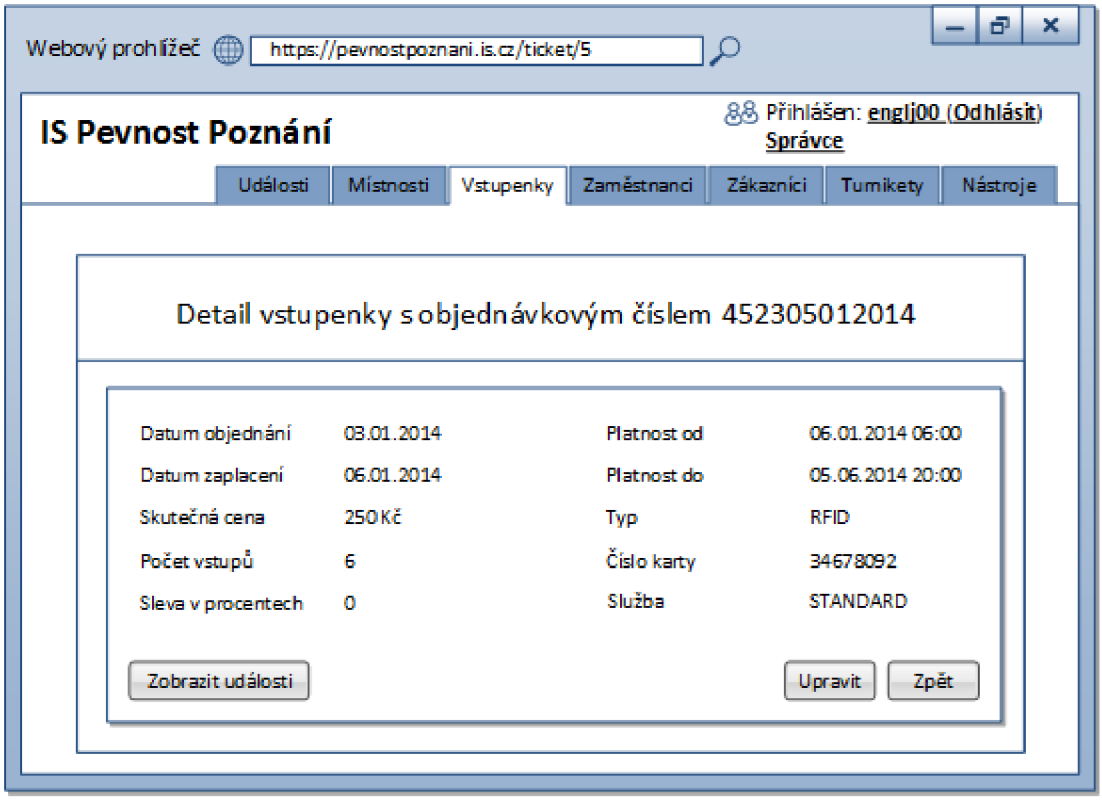
Obrázek I.7: Správce, detail místnosti, seznam událostí



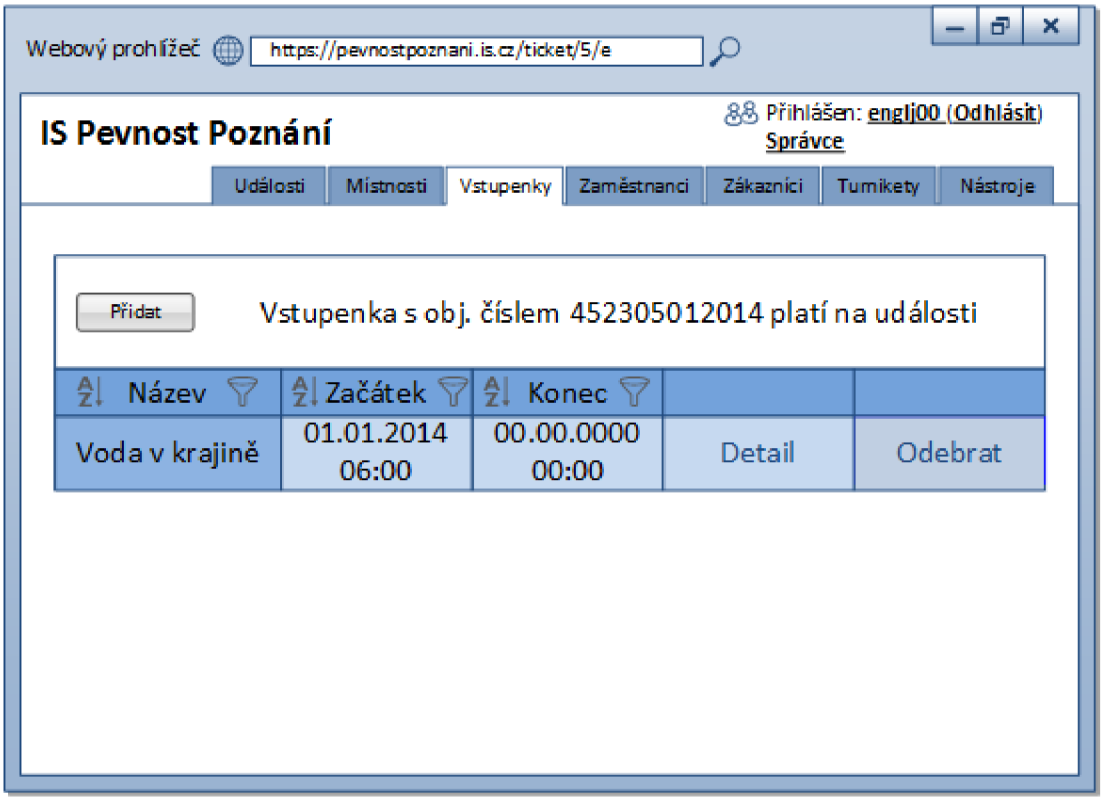
Obrázek I.8: Správce, detail místnosti, seznam turniketů



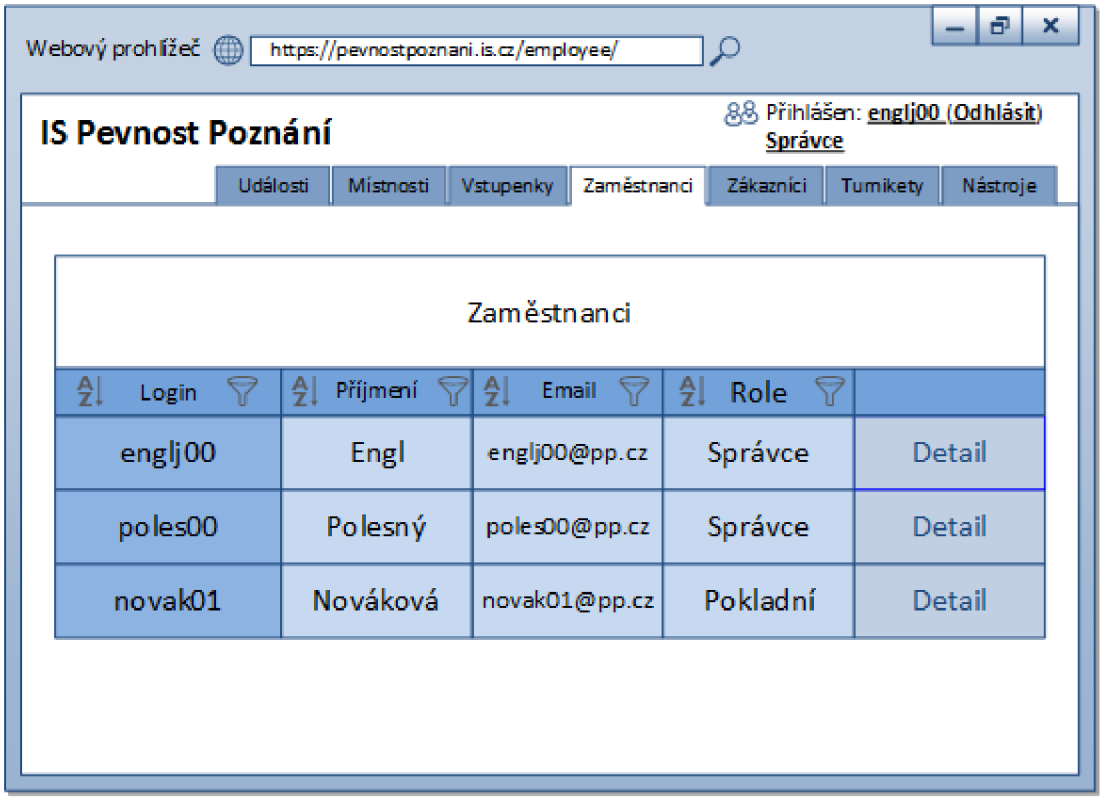
Obrázek I.9: Správce, zobrazení všech vstupenek



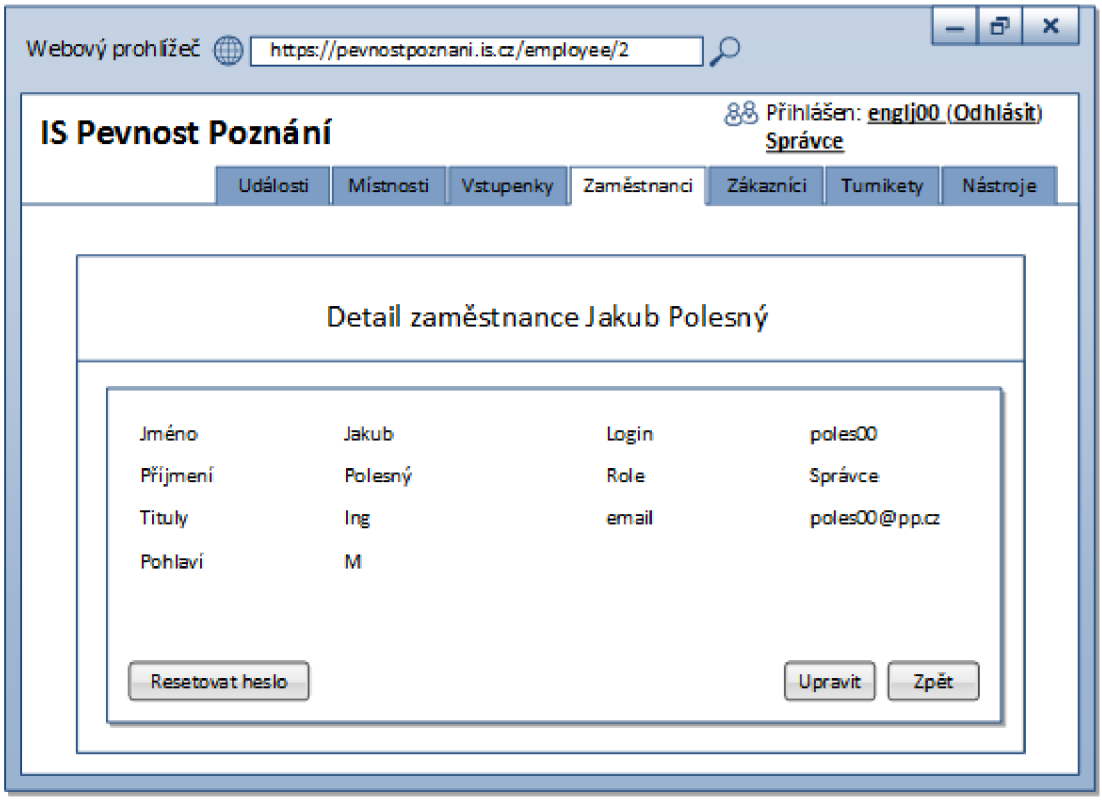
Obrázek I.10: Správce, detail vstupenky



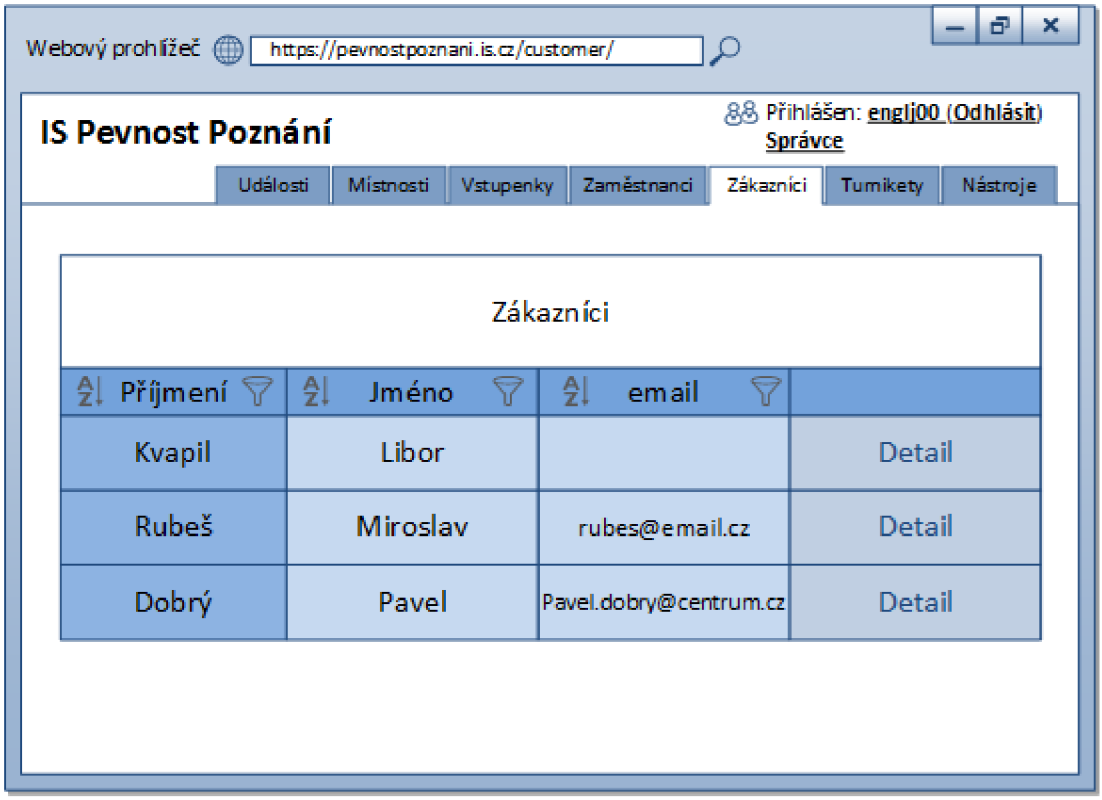
Obrázek I.11: Správce, detail vstupenky, seznam událostí



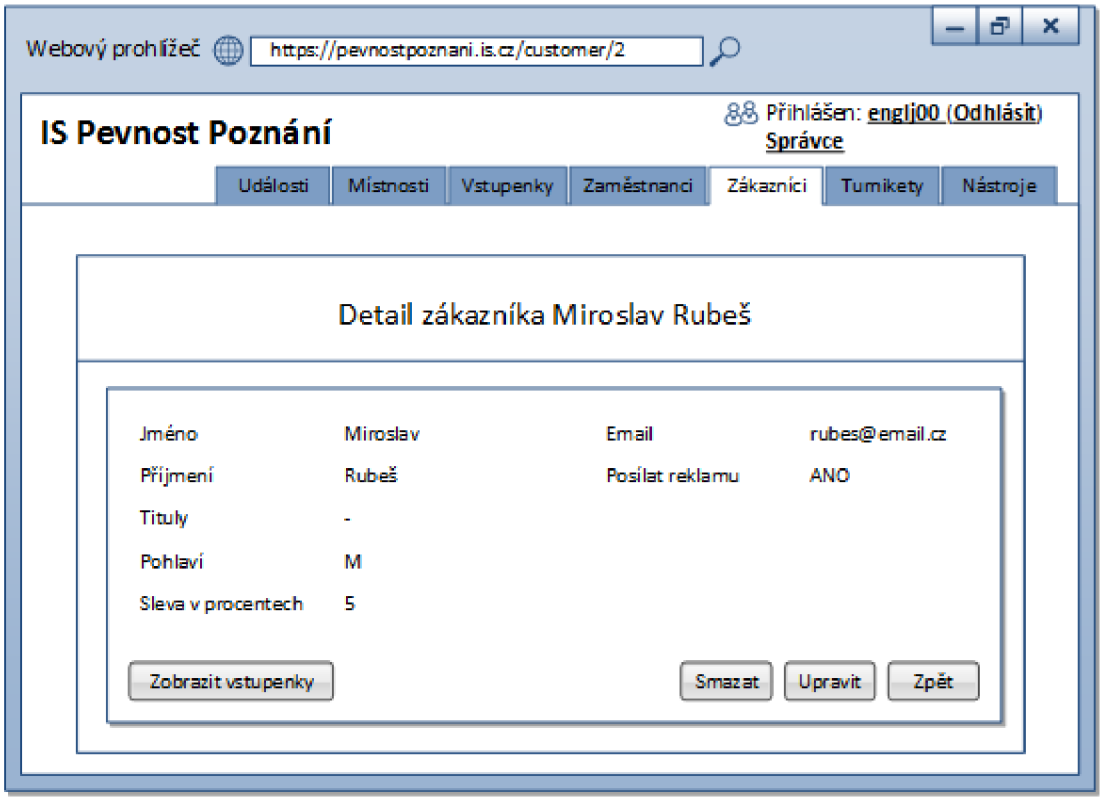
Obrázek I.12: Správce, zobrazení všech zaměstnanců



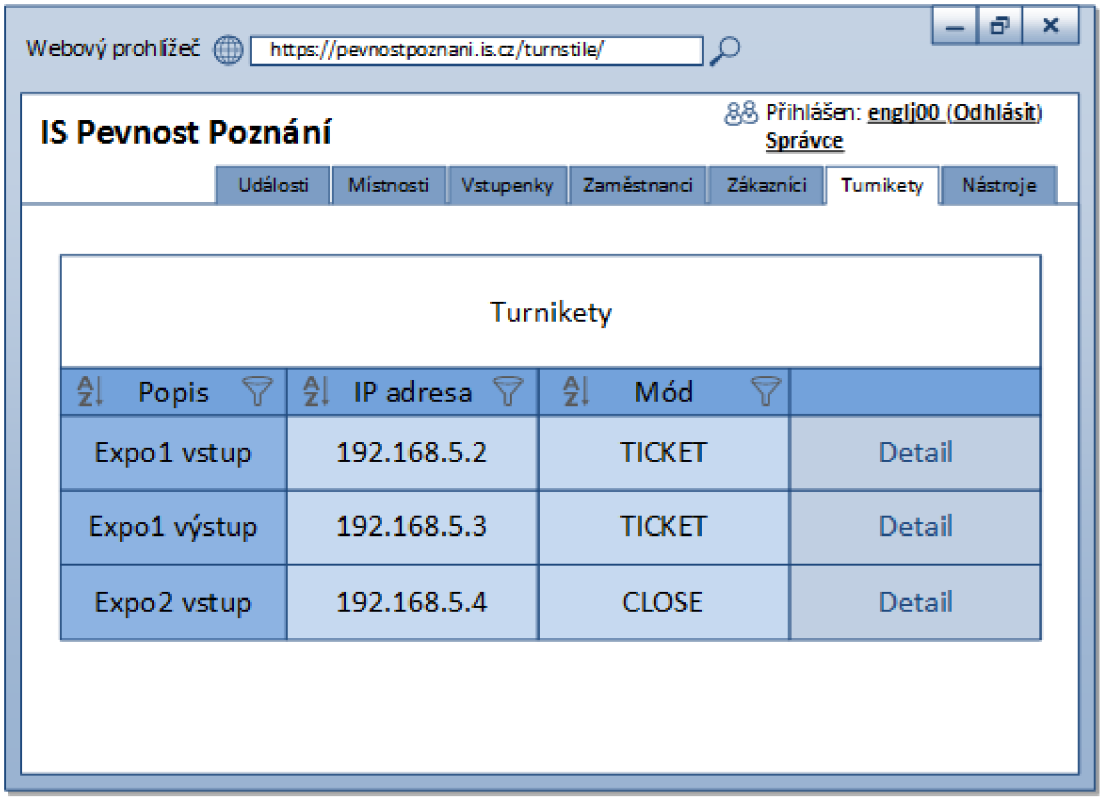
Obrázek I.13: Správce, detail zaměstnance



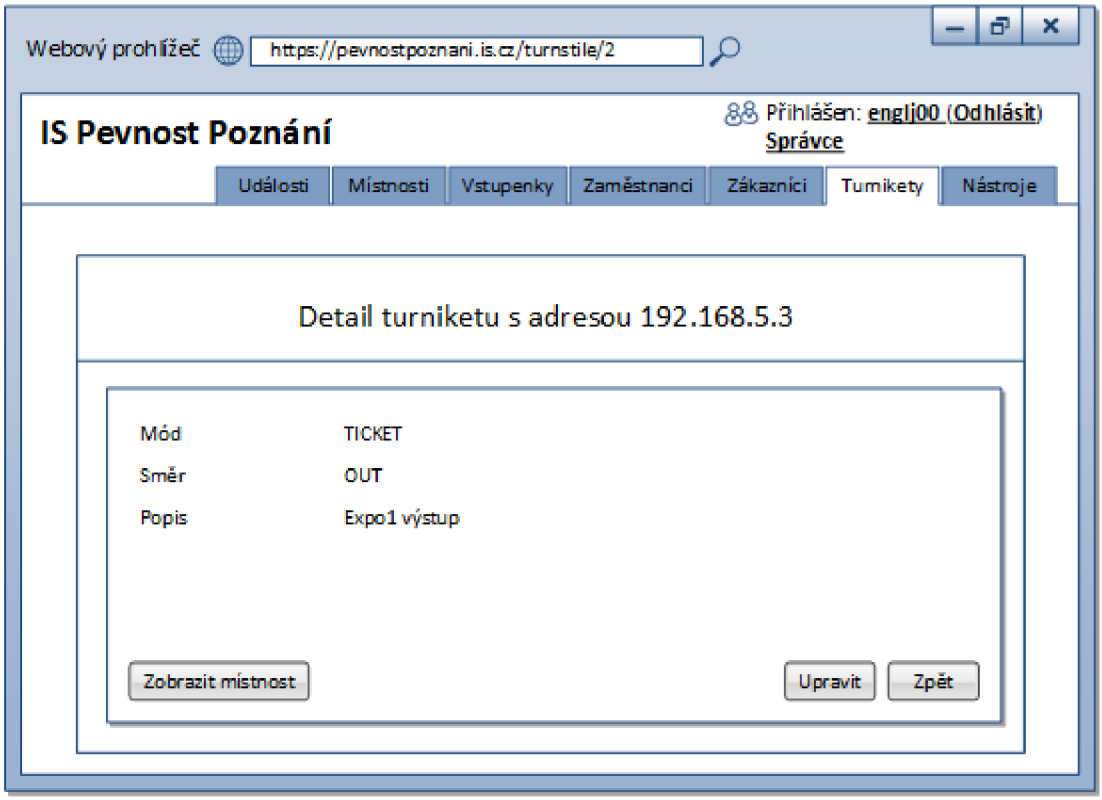
Obrázek I.14: Správce, zobrazení všech zákazníků



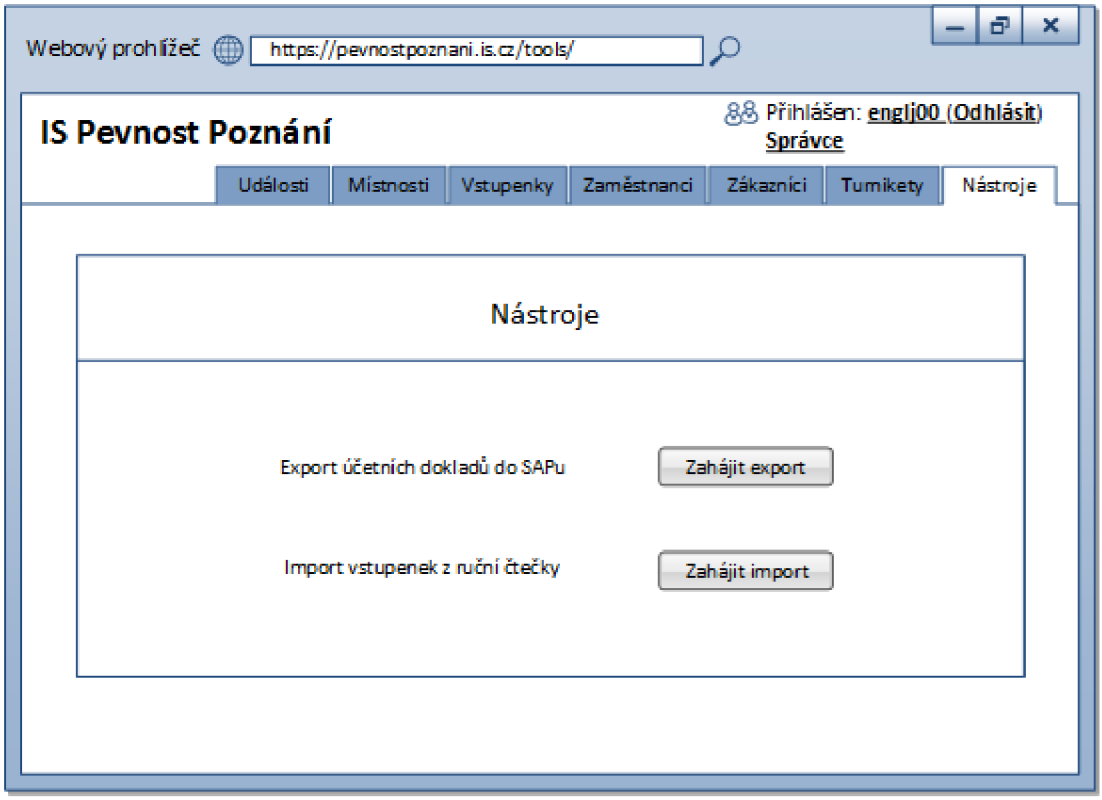
Obrázek I.15: Správce, detail zákazníka



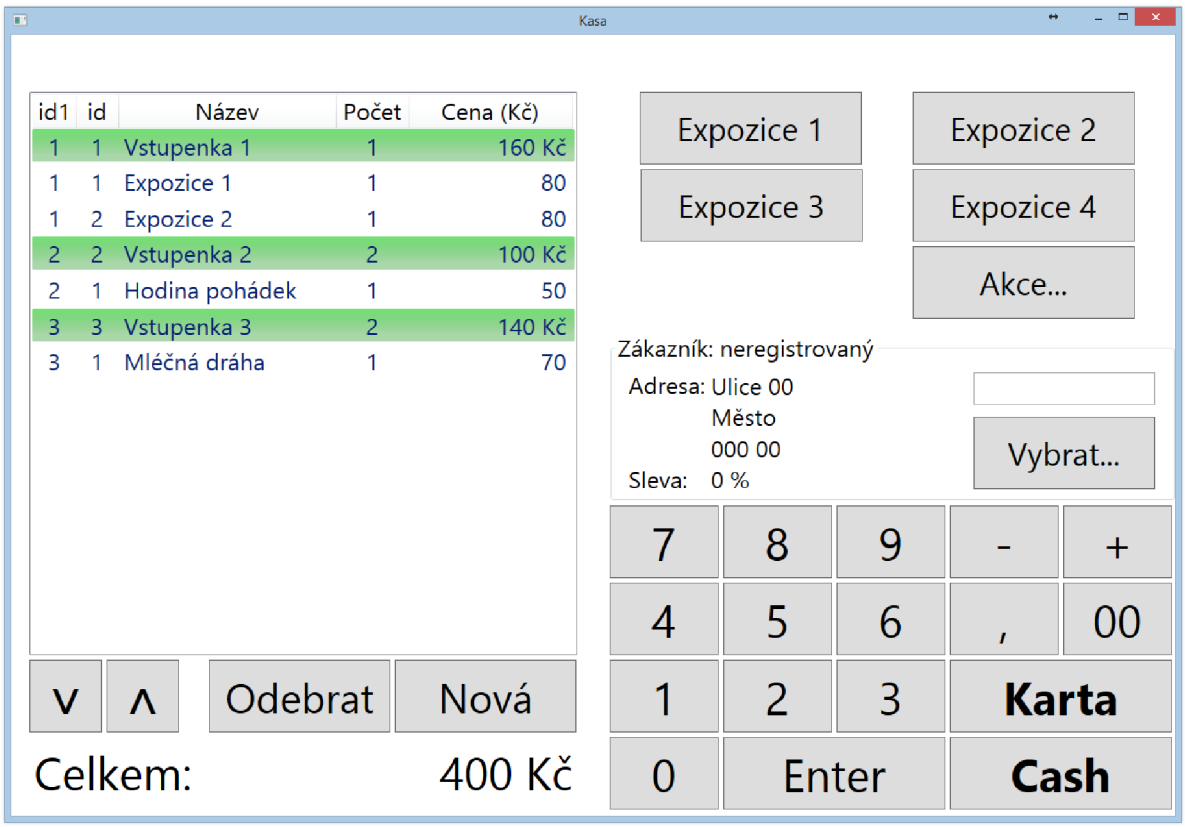
Obrázek I.16: Správce, zobrazení všech turniketů



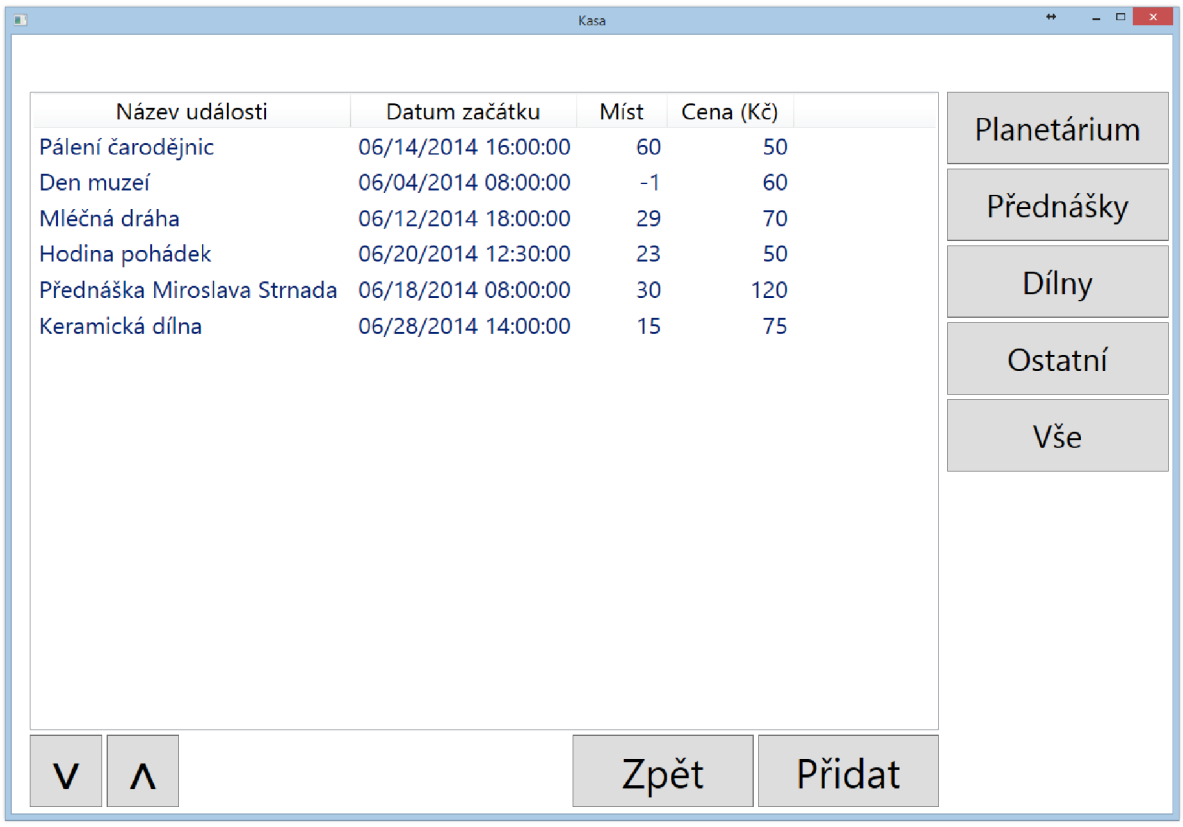
Obrázek I.17: Správce, detail turniketu



Obrázek I.18: Správce, zobrazení nástrojů

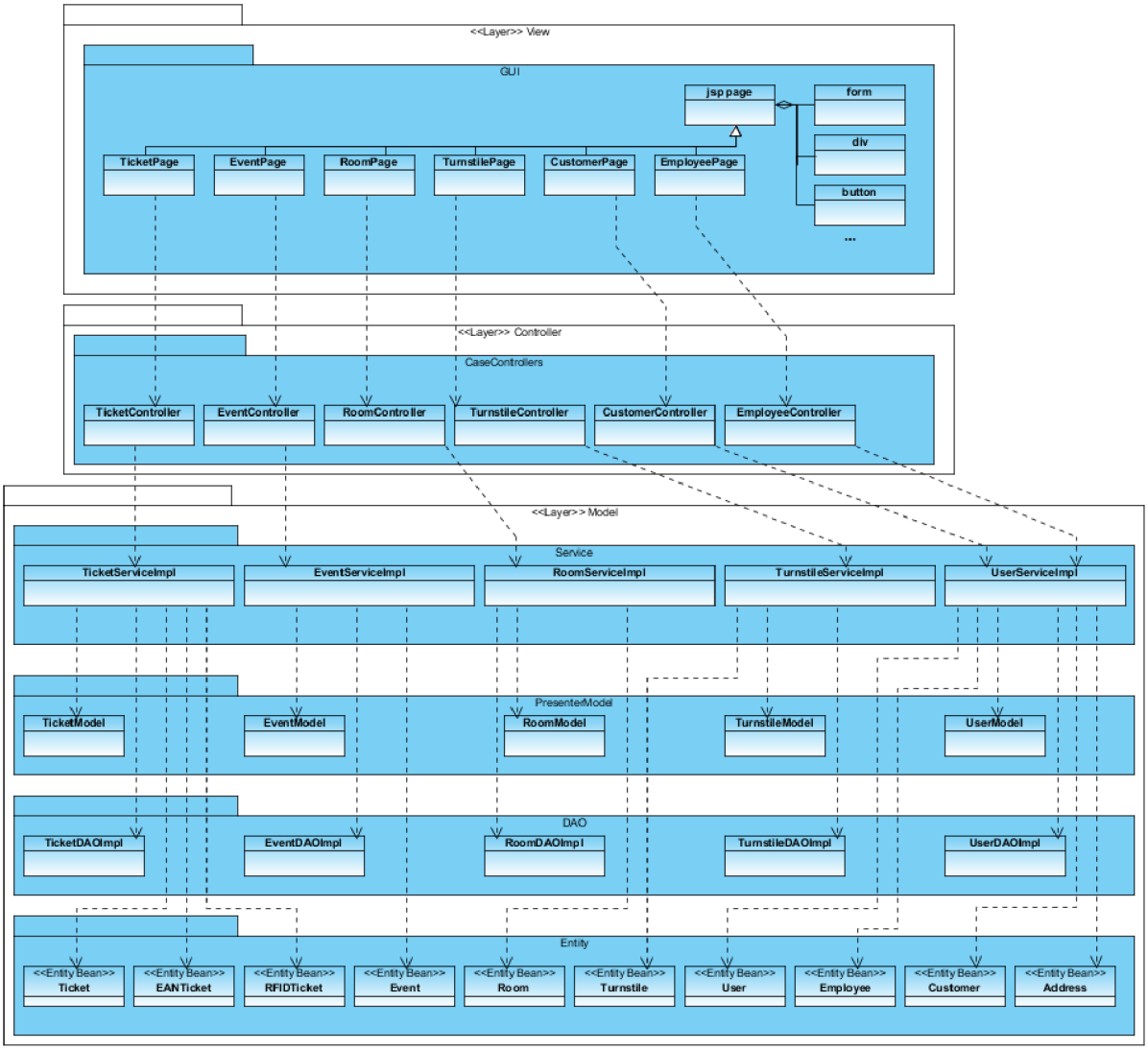


Obrázek I.19: Kasa, hlavní obrazovka



Obrázek I.20: Kasa, výběr akcí

II. Přehledový diagram balíčků



Obrázek II.1Přehledový diagram balíčků

III. XSD schémata dokumentů

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ticket">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="id"/>                    <!-- unikátní identifikátor v systému -->
        <xs:element type="xs:int" name="order_number"/>           <!-- číslo objednávky, generuje systém -->
        <xs:element type="xs:string" name="order_date"/>          <!-- datum objednání vstupenky, přiřazuje systém -->
        <xs:element type="xs:string" name="payment_date"/>        <!-- datum zaplacení vstuepnky, přiřazuje systém -->
        <xs:element type="xs:byte" name="remaining_entrances"/>   <!-- zbývající počet vstupů -->
        <xs:element type="xs:string" name="valid_from"/>          <!-- datum a čas začátku platnosti vstupenky -->
        <xs:element type="xs:string" name="valid_to"/>            <!-- datum a čas ukončení platnosti vstupenky -->
        <xs:element type="xs:string" name="last_modified"/>       <!-- datum a čas poslední modifikace vstupenky -->
        <xs:element type="xs:byte" name="prize"/>                 <!-- celková cena vstupenky, vypočte systém-->
        <xs:element type="xs:string" name="type"/>                <!-- typ vstupenky (EAN) -->
        <xs:element type="xs:byte" name="ean_id"/>                <!-- identifikace EAN v systému -->
        <xs:element type="xs:long" name="bar_code"/>              <!-- EAN kód vstupenky -->
        <xs:element type="xs:string" name="ean_type"/>            <!-- typ vstupeky (STANDARD, SERVICE) -->
        <xs:element name="events">
          <xs:complexType>
            <xs:sequence>  <!-- seznam id událostí, na které vstupenka platí -->
              <xs:element type="xs:byte" name="item"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.1: XSD schéma EAN vstupenky

```
<?xml version="1.0" encoding="UTF-8"?>
<ticket>
    <id>2</id>
    <order_number>548780</order_number>
    <order_date>03/02/2014 14:21:00</order_date>
    <payment_date>03/02/2014 14:21:30</payment_date>
    <remaining_entrances>1</remaining_entrances>
    <valid_from>03/02/2014 14:21:31</valid_from>
    <valid_to>09/02/2016 15:09:00</valid_to>
    <last_modified>04/26/2014 21:42:49</last_modified>
    <prize>100</prize>
    <type>EAN</type>
    <ean_id>2</ean_id>
    <bar_code>123456789012</bar_code>
    <ean_type>STANDARD</ean_type>
    <events>
        <item>3</item>
    </events>
</ticket>
```

Obrázek III.2: Příklad EAN vstupenky

```
<xs:schema attributeFormDefault="unqualified"
           elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ticket">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="remaining_entrances"/>
        <xs:element type="xs:string" name="valid_from" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:string" name="valid_to" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:string" name="type"/>
        <xs:element type="xs:string" name="ean_type" minOccurs="0" maxOccurs="1"/>
        <xs:element name="events">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:byte" name="item" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.3: XSD schéma pro vytvoření nové EAN vstupenky

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ticket>
    <remaining_entrances>1</remaining_entrances>
    <valid_from>05/09/2014 14:00:00</valid_from>
    <valid_to>05/09/2015 14:00:00</valid_to>
    <prize>120</prize>
    <type>EAN</type>
    <ean_type>STANDARD</ean_type>
    <events>
        <item>5</item>
        <item>2</item>
    </events>
</ticket>
```

Obrázek III.4: Příklad XML pro vytvoření EAN vstupenky

```xml
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ticket">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="id"/>                    <!-- unikátní identifikátor v systému -->
        <xs:element type="xs:int" name="order_number"/>           <!-- číslo objednávky, generuje systém -->
        <xs:element type="xs:string" name="order_date"/>          <!-- datum objednání vstupenky, přiřazuje systém -->
        <xs:element type="xs:string" name="payment_date"/>        <!-- datum zaplacení vstuepnky, přiřazuje systém -->
        <xs:element type="xs:byte" name="remaining_entrances"/>   <!-- zbývající počet vstupů -->
        <xs:element type="xs:string" name="valid_from"/>          <!-- datum a čas začátku platnosti vstupenky -->
        <xs:element type="xs:string" name="valid_to"/>            <!-- datum a čas ukončení platnosti vstupenky -->
        <xs:element type="xs:string" name="last_modified"/>       <!-- datum a čas poslední modifikace vstupenky -->
        <xs:element type="xs:byte" name="prize"/>                 <!-- celková cena vstupenky, vypočte systém-->
        <xs:element type="xs:string" name="type"/>                <!-- typ vstupenky (RFID) -->
        <xs:element type="xs:byte" name="rfid_id"/>               <!-- identifikace RFID v systému -->
        <xs:element type="xs:int" name="rfid_code"/>              <!-- RFID kód vstupenky -->
        <xs:element type="xs:string" name="rfid_type"/>           <!-- typ vstupenky (STANDARD, GOLD, EMPLOYEE) -->
        <xs:element type="xs:byte" name="discount"/>              <!-- sleva v procentech -->
        <xs:element type="xs:int" name="card_number"/>            <!-- zákaznické číslo karty, přiděluje systém -->
        <xs:element name="events">
          <xs:complexType>
            <xs:sequence>  <!-- seznam id událostí, na které vstupenka platí -->
              <xs:element type="xs:byte" name="item" maxOccurs="unbounded" minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.5: XSD schéma RFID vstupenky

```xml
<xs:schema attributeFormDefault="unqualified"
           elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ticket">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="remaining_entrances"/>
        <xs:element type="xs:string" name="valid_from" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:string" name="valid_to" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:string" name="type"/>
        <xs:element type="xs:string" name="rfid_type" minOccurs="0" maxOccurs="1"/>
        <xs:element type="xs:byte" name="discount"/>
        <xs:element name="events">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:byte" name="item" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.6: XSD schéma pro vytvoření nové RFID vstupenky

```xml
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="event">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="id"/>              <!-- unikátní identifikátor v sytému -->
        <xs:element type="xs:string" name="name"/>          <!-- název události -->
        <xs:element type="xs:string" name="start_date"/>    <!-- datum a čas zahájení události -->
        <xs:element type="xs:string" name="end_date"/>      <!-- datum a čas ukončení události -->
        <xs:element type="xs:byte" name="prize"/>           <!-- základní cena událsoti v Kč -->
        <xs:element type="xs:byte" name="capacity"/>        <!-- počet vstupenk k prodeji -->
        <xs:element type="xs:string" name="type"/>          <!-- typ (kategorie) událsoti -->
        <xs:element type="xs:string" name="description"/>   <!-- popis událsoti -->
        <xs:element type="xs:byte" name="remaining_seats"/> <!-- počet volných míst -->
        <xs:element name="rooms">
          <xs:complexType>
            <xs:sequence> <!-- seznam místností, ve kterých událost probíhá -->
              <xs:element type="xs:byte" name="item" maxOccurs="unbounded" minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element> <!-- seznam prodaných vstupenek -->
        <xs:element type="xs:string" name="tickets"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.7: XSD schéma události

```xml
<?xml version="1.0" encoding="UTF-8"?>
<event>
    <id>4</id>
    <name>Den muzeí</name>
    <start_date>06/04/2014 08:00:00</start_date>
    <end_date>06/04/2014 20:30:32</end_date>
    <prize>60</prize>
    <capacity>-1</capacity>
    <type>OTHERS</type>
    <description>Den muzeí, prohlídka expozicí</description>
    <remaining_seats>-1</remaining_seats>
    <rooms>
        <item>5</item>
        <item>1</item>
        <item>2</item>
    </rooms>
    <tickets />
</event>
```

Obrázek III.8: Příklad události

```xml
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="user">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:byte" name="id"/>                  <!-- unikátní identifikátor v systému -->
        <xs:element type="xs:string" name="first_name"/>        <!-- jméno zákazníka -->
        <xs:element type="xs:string" name="last_name"/>         <!-- příjmení zákazníka -->
        <xs:element type="xs:string" name="titles"/>            <!-- tituly zákazníka -->
        <xs:element type="xs:string" name="gender"/>            <!-- pohlaví zákazníka -->
        <xs:element type="xs:string" name="email"/>             <!-- email zákazníka -->
        <xs:element type="xs:string" name="type"/>              <!-- typ zákazníka (CUSTOMER) -->
        <xs:element type="xs:string" name="create_date"/>       <!-- datum vytvoření záznamu zákazníka -->
        <xs:element type="xs:string" name="last_modified"/>     <!-- datum poslení změny záznamu zákazníka -->
        <xs:element type="xs:byte" name="address_id"/>          <!-- identifikátor adresy -->
        <xs:element type="xs:string" name="street"/>            <!-- ulice -->
        <xs:element type="xs:string" name="district"/>          <!-- čtvrť -->
        <xs:element type="xs:string" name="city"/>              <!-- město -->
        <xs:element type="xs:int" name="postal_code"/>          <!-- PSČ -->
        <xs:element type="xs:byte" name="customer_id"/>         <!-- identifikátor zákazníka -->
        <xs:element type="xs:string" name="send_advertising"/> <!-- zda posílat reklamní nabídky (true, false) -->
        <xs:element type="xs:byte" name="discount"/>            <!-- osobní procentuální sleva zákazníka -->
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.9: XSD schéma detailu zákazníka

```xml
<?xml version="1.0" encoding="UTF-8"?>
<user>
    <id>4</id>
    <first_name>Petr</first_name>
    <last_name>Fohler</last_name>
    <titles>Ing.</titles>
    <gender>MALE</gender>
    <email>fohler@mail.cz</email>
    <type>CUSTOMER</type>
    <create_date>03/02/2014 00:00:00</create_date>
    <last_modified>05/09/2014 20:52:38</last_modified>
    <address_id>4</address_id>
    <street>Oblá 20</street>
    <district />
    <city>Kolorov</city>
    <postal_code>34678</postal_code>
    <customer_id>4</customer_id>
    <send_advertising>true</send_advertising>
    <discount>0</discount>
</user>
```

Obrázek III.10: Příklad detailu zákazníka

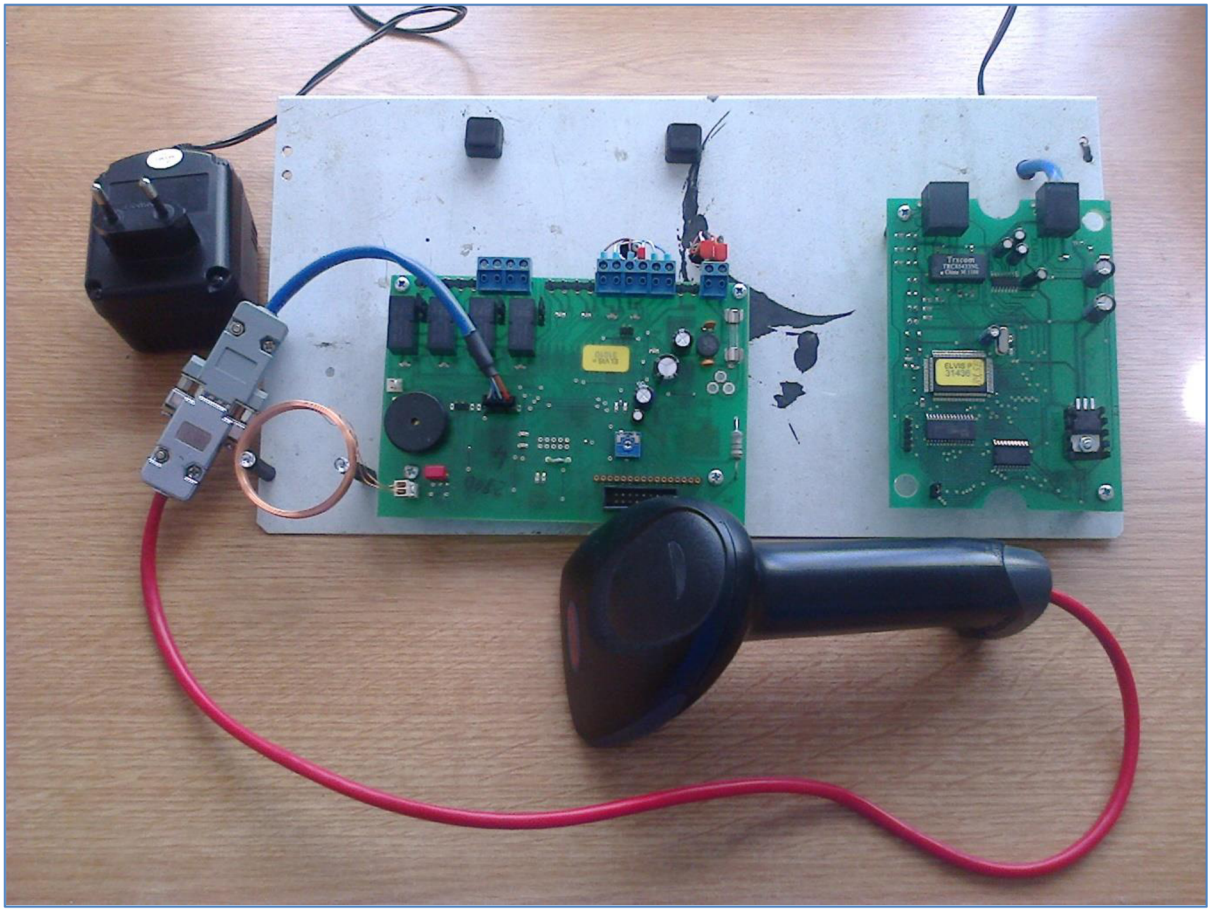
```xml
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="tickets">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ticket" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:byte" name="id"/>
              <xs:element type="xs:int" name="order_number"/>
              <xs:element type="xs:string" name="order_date"/>
              <xs:element type="xs:string" name="payment_date"/>
              <xs:element type="xs:byte" name="remaining_entrances"/>
              <xs:element type="xs:string" name="valid_from"/>
              <xs:element type="xs:string" name="valid_to"/>
              <xs:element type="xs:string" name="last_modified"/>
              <xs:element type="xs:short" name="prize"/>
              <xs:element type="xs:string" name="type"/>
              <xs:element type="xs:byte" name="rfid_id"/>
              <xs:element type="xs:int" name="rfid_code"/>
              <xs:element type="xs:string" name="rfid_type"/>
              <xs:element type="xs:byte" name="discount"/>
              <xs:element type="xs:int" name="card_number"/>
              <xs:element name="events">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:byte" name="item"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázek III.11: XSD schéma souboru vstupenek pro ruční čtečku

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tickets>
    <ticket>
        <id>1</id>
        <order_number>548799</order_number>
        <order_date>03/03/2014 00:21:31</order_date>
        <payment_date>03/03/2014 15:45:50</payment_date>
        <remaining_entrances>6</remaining_entrances>
        <valid_from>04/27/2014 15:46:34</valid_from>
        <valid_to>04/27/2015 15:51:28</valid_to>
        <last_modified>04/27/2014 15:53:05</last_modified>
        <prize>120</prize>
        <type>RFID</type>
        <rfid_id>1</rfid_id>
        <rfid_code>22334455</rfid_code>
        <rfid_type>GOLD</rfid_type>
        <discount>10</discount>
        <card_number>1234567</card_number>
        <events>
            <item>5</item>
        </events>
    </ticket>
    <ticket>
        <id>6</id>
        <order_number>123456</order_number>
        <order_date>04/15/2014 12:21:31</order_date>
        <payment_date>04/15/2014 12:30:00</payment_date>
        <remaining_entrances>5</remaining_entrances>
        <valid_from>04/15/2014 12:30:00</valid_from>
        <valid_to>04/15/2015 12:30:00</valid_to>
        <last_modified>04/28/2014 09:57:04</last_modified>
        <prize>300</prize>
        <type>RFID</type>
        <rfid_id>6</rfid_id>
        <rfid_code>22334455</rfid_code>
        <rfid_type>GOLD</rfid_type>
        <discount>7</discount>
        <card_number>1234567</card_number>
        <events>
            <item>5</item>
        </events>
    </ticket>
</tickets>
```

Obrázek III.12: Příklad souboru vstupenek pro ruční čtečku

IV. Fotografie elektroniky turniketu



Obrázek IV.1: Testovací elektronika turniketu