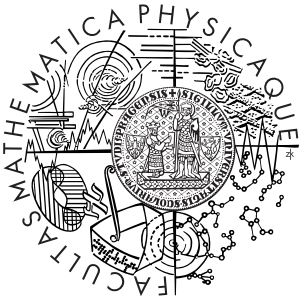
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



David Šenkeřík

Rezervační systém vstupenek

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Josef Zlomek, Katedra teoretické informatiky a matematické logiky

Studijní program: informatika, obor programování

2006

Na tomto místě bych chtěl poděkovat RNDr. Josefu Zlomkovi za vedení a odbornou pomoc při tvorbě bakalářského projektu.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 27. května 2006 David Šenkeřík

Obsah

Název práce: Rezervační system vstupenek
Autor: David Šenkeřík
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Josef Zlomek
e-mail vedoucího: Josef.Zlomek@mff.cuni.cz

Abstrakt: Cílem bakalářské práce bylo vytvořit Rezervační systém vstupenek. Rezervační systém je primárně určen pro rezervaci a prodej vstupenek na příležitostné kulturně-společenské akce. Rezervace a prodej vstupenek se provádí nad centrálními daty z vícero prodejních míst. Zákazníci si mohou před návštěvou prodejního místa rezervovat vstupenky pomocí webového rozhraní. Místa pro rezervaci a prodej se vybírají na "mapě" prostor, ve kterých se akce bude konat.

Administrátor systému má k dispozici uživatelsky přívětivé rozhraní pro nadefinování rozložení prostor, ve kterých se bude akce konat, tj. rozhraní pro definování tvaru místností, rozložení stolů a židlí nebo řad sedadel apod.

Klíčová slova: rezervace vstupenek, sdílená data, MySQL, bezpečnost

Title: Ticket Reservation System
Author: David Šenkeřík
Department: Department of Software Engineering
Supervisor: RNDr. Josef Zlomek
Supervisor's e-mail address: Josef.Zlomek@mff.cuni.cz

Abstract: The aim of the bachelor thesis was to design and implement a Ticket Reservation System. The reservation system is primarily focused on ticket distribution for occasional culture events. During reservation and selling of tickets it is necessary to work with centralized shared data from many selling places. The customers may book the tickets via the web interface before they buy the tickets from the distributor.

A user-friendly interface for designing the room models (including chairs, tables, rows, walls, etc.) is available for the system administrator. The tickets to book or sell can be chosen using the model of the room.

Keywords: tickets reservation, MySQL, shared data, security

Kapitola 1

Úvod

S rozvojem počítačových technologií a jejich rozšířením do světa obchodu a služeb vznikla v posledních letech řada systémů pro distribuci vstupenek. Ty s postupem času nacházejí stále větší uplatnění a setkává se s nimi každý z nás, kdykoli plánuje jít do divadla, do restaurace nebo letět na dovolenou.

Tyto systémy, fungující často jako webové aplikace na internetu a tedy dostupné prakticky odkudkoli, se často označují jako rezervační systémy.

Ve strategii firem, které se snaží získat zákazníka, hraje dnes významnou roli nejen kvalita nabízených služeb, ale i zprostředkování a přiblížení služeb cílovému zákazníkovi. Zprostředkování služeb pro širokou veřejnost je významnou úlohou právě rezervačních systémů, pomocí nichž si zákazník může vstupenky jen s nepatrným úsilím zajistit.

Pod pojmem rezervační systém (rezervační rozhraní) tedy chápeme aplikaci, kterou bude obsluhovat zákazník, který si chce něco rezervovat, a tedy zřejmě vzdáleně.

Druhou skupinu aplikací, které používají distributoři při přijetí platby a zaevidování, že rezervace byla naplněna, budeme nazývat prodejní systémy (prodejní rozhraní aplikace). Prodejní systémy, často provozované v kombinaci s rezervačními, poskytují prodejci větší kontrolu nad prodejem. Elektronický prodej je v dnešní době velmi často nutností, příkladem je existence sítě několika distribučních center, která spolu komunikují a sdílí data o prodeji.

1.1 Cíl

Cílem bakalářského projektu bylo navrhnout a implementovat komplexní informační systém pro rezervaci a prodej vstupenek určený pro distribuci vstupenek na příležitostné akce převážně kulturně-společenského, ale také sportovního a vzdělávacího charakteru. Základním požadavkem projektu je schopnost pracovat s centrálně sdílenými daty z vícero prodejních míst. Software by měl být navržen tak, aby cílový zákazník provozovatele měl před zakoupením možnost vybrat a zarezervovat vstupenky pomocí webového rozhraní, k jejich koupi pak musí navštívit oprávněnou osobu, která mu vstupenky prodá pomocí prodejního rozhraní. Platba a předání vstupenek neprobíhá elektronicky, ale při prodeji u provozovatelem oprávněné osoby.

Uživatel software by měl mít k dispozici uživatelsky přívětivý editor pro modelování prostor a událostí v nich. Editor by měl umožnit vytvářet sedadla, stoly, řady a další objekty tak, aby si zákazník mohl vybrat umístění vstupenek přímo v modelu prostor. Cílem editoru je zejména usnadnit zákazníkovi rozhodování při výběru umístění vstupenek, ale také zpříjemnit distributorům práci s aplikací.

1.2 Motivace

Motivací pro vytvoření software je nedostupnost freeware systémů, které by mohly být využity pro prodej vstupenek na příležitostné kulturní a společenské události. Software by měl být tedy co nejvíce založen na open-source technologiích, aby jeho provoz nekladl na uživatele příliš velké finanční nároky.

Následující motivační příklad demonstruje možnost využití systému.

Příklad nasazení Představme si situaci, kdy větší firma pořádá kulturní událost pro své zaměstnance a potřebuje k tomuto účelu distribuovat vstupenky. Je přitom určeno několik oprávněných osob, které mají prodej na starost (tedy distributorů).

Protože mohou být prostory pro pořádání kulturních akcí značně různorodé, vytvoří nejdříve správce systému podle reálného rozložení prostor model. Do tohoto modelu přidá jednotlivé objekty, se kterými se asociují vstupenky. Příkladem takových objektů mohou být stoly, řady nebo sedadla. Účelem vytvoření modelu je co nejvíce usnadnit zákazníkům výběr konkrétního umístění, ale také usnadnit a zpřehlednit distributorům prodej. Model prostor bude tedy posléze používán jednak distributory při prodeji, jednak zákazníky při rezervaci.

Zaměstnanec společnosti, v roli cílového zákazníka, si před zakoupením zarezervuje vstupenky přes webový prohlížeč. Pokud je událost do jisté míry veřejná, bude rozhraní pro rezervaci přístupné přímo z internetu. Pokud je požadováno, aby k rezervacím měli přístup pouze zaměstnanci společnosti, bude rezervační

rozhraní dostupné pouze na firemním intranetu. Poté, co si zaměstnanec zarezervoval konkrétní místa, zakoupí si před událostí vstupenky přes prodejní rozhraní u pověřeného distributora.

Administrátor systému si bude v průběhu přípravy na událost zobrazovat statistiky o prodeji. Ty využije k řešení organizačních záležitostí, například pro předběžné oznámení velikosti zájmu. Podle statistik může také odměňovat jednotlivé distributory.

Můžeme si také představit situaci, kdy chce zakoupit vstupenky větší skupina osob, například celé oddělení. Logickým požadavkem je přitom umístění míst co nejblíže k sobě, tak aby členové skupiny seděli co nejvíce spolu.

Příkladem využití, kvůli kterému byl software navržen a implementován, je distribuce vstupenek na plesy Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

1.3 Požadované vlastnosti

Z motivačního příkladu plynou následující vlastnosti, které by měl software splňovat:

Široká možnost nasazení Cílem návrhu je, aby aplikace byla díky obecnosti snadno použitelná pro široké rozpětí druhu událostí. Software sice bude primárně určen pro příležitostné akce kulturně-společenského charakteru, může být ale v principu nasazen i ve sportovních, dopravních či jiných společnostech.

Grafický editor Důležitou součástí software by měl být zejména propracovaný grafický editor, který by umožnil uživateli snadno a přehledně vytvářet mapy rozložení prostor, ve kterých probíhají události, obdobně pak grafické rozhraní pro modelování událostí. Základním požadavkem na grafický editor prostor je zejména jednoduchá práce uživatele v přehledném, uživatelsky přívětivém prostředí.

Prohledávání prostor Další vlastností, která byla požadována po rozhraní pro prodej, je možnost prohledávání prostor a s tím spojené automatické přidělování vstupenek podle velikosti požadavku. Prohledávání prostor musí být inteligentní, mělo by co nejblíže simulovat požadavky skutečného zákazníka. Uživatel aplikace může využít funkci vyhledávání pro hospodárné rozdělování požadavků tam, kde zákazník nepožaduje žádné konkrétní umístění. Příkladem využití je hospodárné a plně automatické přidělování stolů.

Využití této vlastnosti plyne také z motivačního příkladu, funkce může být aplikována pro hromadný prodej, kde nejlepší umístění není zřejmé pouze z mapy obsazení sedadel.

Statistiky prodeje Důležitou a zároveň velmi praktickou vlastností software je možnost zobrazit přehledně statistická data o prodeji, nejlépe formou grafů. Tato vlastnost slouží provozovateli aplikace k celkovému hodnocení prodeje, jednotlivých prodejců nebo k vytváření dalších statistik.

Bezpečnost aplikace Rezervace vstupenek má z principu veřejný charakter, aplikace umožňující rezervace musí být tedy dostupná na síti (obvykle přímo na internetu). Problémem nejen rezervačních systémů, ale obecně všech aplikací, které přistupují k datům umístěným na síti, je zajištění bezpečnosti a integrity uložených dat. K datům by měly mít možnost přistupovat pouze ty osoby, které k tomu mají náležité oprávnění od provozovatele software, každá osoba by měla být při přístupu autentizována a její dotaz autorizován. Obdobně je nutné zamezit všem útokům na databázi (útokům pomocí SQL Injection popsaných v kapitole 3.6.3).

Problematika bezpečnosti je u rezervačních systémů velmi citlivá, neautorizovaný zásah do databáze by mohl vést jednak k finanční ztrátě provozovatele software, jednak k poškození cílového zákazníka. Proto by měl být v projektu na téma bezpečnosti kladen mimořádný důraz.

1.4 Struktura práce

Kapitola 2 je analýzou návrhu rezervačního a prodejního systému, jsou zde popsány jednotlivé aspekty návrhu včetně informací o existujícím příbuzném software.

Kapitola 3 popisuje návrh a implementaci vytvořeného systému RTRS. Součástí kapitoly je popis řešení problémů, které vyplynuly z analýzy, popis architektury programu (včetně rozdělení funkcí mezi jednotlivé části aplikace), popis použitých technologií a některých implementačních detailů.

Kapitola 4 je instalační a konfigurační příručkou programu. Jsou zde popsány požadavky na zprovoznění software, instalace a možnosti konfigurace jednotlivých komponent aplikace.

Kapitola 5 je stručnou uživatelskou příručkou prodejního rozhraní aplikace. Uspořádání kapitoly zachovává pořadí kroků, ve kterých uživatel pracuje s aplikací. Kapitola popisuje přihlášení uživatele, modelování prostor v editoru, postup vytváření událostí v definovaném modelu a samotný prodej vstupenek.

Kapitola 2

Analýza

V této kapitole jsou analyzovány jednotlivé aspekty návrhu a implementace rezervačních systémů. Kapitola ukazuje, v čem dostupné systémy nevyhovují našim požadavkům.

Z analýzy a rozboru základních otázek návrhu pak vyplynuly problémy, které bylo při návrhu a implementaci nutno vyřešit.

2.1 Rezervační systémy

Důležitou součástí analýzy problému a návrhu řešení bylo studium příbuzného software, který je k dispozici na trhu.

Většina projektů příbuzného charakteru na trhu je vytvořena na komerční bázi, jejich instalace a používání jsou podmíněny zakoupením licence. Velmi často je zpoplatněn také samotný provoz systému (obvyklé jsou měsíční poplatky nebo poplatky za správu systému). Provozování těchto systémů může klást na provozovatele neúměrné finanční nároky, což je obvykle nejsilnějším impulsem k zamítnutí použití těchto programů. Problémem je také to, že dostupný software až na výjimky není opensource.

Komerční software, který se zabývá distribucí vstupenek, se většinou snaží co nejvíce přiblížit požadavkům konkrétního typu provozovatele, uchyluje se tak od obecnosti ke konkretizaci řešení. Důsledkem je vznik rezervačních systémů, které jsou cíleně zaměřeny například na prodej vstupenek pro dopravní společnosti.

2.1.1 Bizzy

Rezervační systém Bizzy je příkladem komerčního rezervačního systému, je zaměřen převážně na provozovatele sportovních center. Jedná se o nástroj, který pomáhá firmám provozujícím fit centra, tenisové kurty nebo solária organizovat rezervace a objednávky. Popis systému a demoverze jsou k nahlédnutí na adrese[8].

Systém podporuje online rezervace, je ho tedy možno provozovat jako internetovou aplikaci, popřípadě jako aplikaci na firemním intranetu. Každý zákazník se musí v systému registrovat, po autentizaci má pak k dispozici rozhraní pro rezervace.

Nejzajímavější vlastností tohoto systému, kterou se odlišuje od dalších rezervačních systémů zaměřených na tuto oblast, je možnost integrace se systémem pro ovládání vybavení sportovního centra. Aplikace tak dokáže rozsvěcovat osvětlení na tenisových kurtech nebo spouštět bowlingové dráhy na základě dat o jejich obsazení.

Další zajímavou vlastností systému Bizzy je možnost automatických, periodických rezervací. Tato vlastnost je pro sportovní centra poměrně zajímavá, pro příležitostné akce je však nepoužitelná. V námi navrhovaném systému je tedy zbytečné ji implementovat.

Rezervační systém Bizzy je logicky určen pro jiný typ rezervací, než je řešen v tomto projektu. Alternativu jeho využití pro naše účely můžeme tedy vyloučit. Obsahuje však některé zajímavé vlastnosti, které by měl implementovat obecný rezervační systém. Jedná se hlavně o statistické výstupy, ukládání zákazníků pro budoucí marketingové účely a možnost propojení systému s emailovým klientem. Propojení s emailovým klientem může být využito pro odesílání upozornění na končící rezervace nebo pro rozesílání nabídek stálým zákazníkům.

2.1.2 PHPMyTicket

Jako studijní materiál sloužil zejména online rezervačním systém PHPMyTicket, který se zabývá prodejem vstupenek na kulturně-společenské akce. Tento systém je freeware a opensource, zdrojové kódy programu, plná verze i online demoverze jsou k nahlédnutí a ke stažení na adrese [9].

Systém je kompletně navržen v jazyce PHP, k uchovávání dat využívá MySQL databázi. Data jsou v databázi ukládána do tabulek typu InnoDB, které umožňují využívání transakčního zpracování dotazů. Systém je logicky rozdělen na čtyři samostatné části:

- Admin Interface
- Online Shop
- Sales Point Interface
- Tickets Control Point Interface

Admin interface Rozhraní pro administrátora slouží k přidávání a editaci událostí v databázi. Pro práci s tímto rozhraním je nutné autentizovat se uživatelským jménem a heslem. Administrátor pak může vytvářet jednoduché modely

prostor. Jeden prostor může být rozčleněn na několik podprostorů, každý podprostor pak na jednotlivé zóny. Do těchto zón jsou přidělovány cenové kategorie vstupenek, jednotlivé kategorie se odlišují barvou sedadla. Po nadefinování prostoru a cenových kategorií vstupenek může administrátor vytvořit událost. Po vytvoření musí událost publikovat, ta se bude od té chvíle bude zobrazovat v prodejním rozhraní jako aktivní.

Administrátorské rozhraní nabízí také možnost vyhledat vstupenku podle číselného kódu, lze tak zobrazit informace o jejím vlastníkovi. Kromě toho poskytuje rozhraní také statistické informace o prodeji, textovou i velmi jednoduchou grafickou formou.

Data, která jsou primárně uložena v databázi, mohou být přes administrátorské rozhraní exportována do XML, možný je poté i zpětný import dat do databáze.

Sales point interface Druhou částí systému PHPMyTicket je webové rozhraní, které umožňuje sledovat stav rezervací, rušit objednávky a podobně. Rozhraní je tedy určeno pro cílové zákazníky, pro vstup do tohoto rozhraní je nutná registrace a autentizace zákazníka.

Online Shop Další částí systému je online rozhraní pro zakoupení vstupenek. Autoři systému PHPMyTicket pojali také předání vstupenek online metodou. Zákazník obdrží po elektronické platbě na zadaný email dokument ve formátu PDF, tento dokument obsahuje unikátní číselný a také čárový kód, který jednoznačně charakterizuje konkrétní vstupenku. Vytištěním se dokument stává dokladem o zakoupení vstupenky. Tato metoda je pro provozovatele velmi efektivní, je pohodlná také pro cílového zákazníka, nevýhodou je ale nutnost kontrolovat vstupenky v místě události elektronicky pomocí speciálního rozhraní. Elektronickou kontrolu na vstupu totiž nemusí některá centra podporovat.

Tickets control point interface Poslední částí systému PHPMyTicket je rozhraní pro kontrolu vstupenek při příchodu zákazníka. Toto rozhraní musí být provozováno přímo v místě konání události. Cílem rozhraní je kontrolovat interaktivně s přístupem do databáze číselné kódy jednotlivých vstupenek. Pokud číselný kód projde úspěšně kontrolou, změní se v databázi stav příslušné vstupenky, další pokus o použití vstupenky s tímto kódem nahlásí správci chybu. Chyba je nahlášena také při pokusu udat vstupenku s číselným kódem, který není veden v databázi. Nutnost implementace tohoto rozhraní plyne z předávání vstupenek online metodou, je tak omezena možnost zneužití software vytištěním neexistující vstupenky.

Nevýhodou kompletního návrhu systému PHPMyTicket v jazyce PHP, který příliš neumožňuje vytvářet grafické uživatelské rozhraní, je možnost vytvářet a editovat události pouze v textovém formátu. Chybí zde propracovanější grafický editor prostor, který by uživateli umožnil definovat stoly, řady nebo stěny, rozvr-

žení prostor se týká pouze samotných sedadel, což nevyhovuje našemu požadavku na aplikaci. Součástí software není také žádné rozhraní pro prohledávání prostor a s tím spojené automatické přidělování požadavků. Další nevýhodou je také nepříliš obsáhlé zobrazení statistik prodeje. Aplikace PHPMyTicket tedy nesplňuje naše požadavky na funkčnost rezervačního systému.

2.2 Diskuze návrhu systému

Při návrhu rezervačního systému byly analyzovány následující body:

Rezervace přes síť Při návrhu funkcí každého systému hraje rozhodující roli fakt, pro koho bude aplikace určena. S tím pak souvisí i umístění a konkrétní implementace programu. Rozhraní pro rezervaci vstupenek musí být svým umístěním přístupné pro všechny potenciální zákazníky. Prakticky všechny rezervační systémy proto implementují toto rozhraní jako webovou aplikaci. Webová aplikace může být podle požadavků na ochranu a přístup umístěna na internetu nebo firemním intranetu. To s sebou nese problém zabezpečení přenášených dat, na který je potřeba při návrhu řešení klást důraz.

Prodej přes síť Charakter prodeje vstupenek vyžaduje existenci distribuční sítě. Základním problémem je, zda má mít tato distribuční síť veřejný charakter. Prodej vstupenek by pak probíhal přímo přes internet, k prodejnímu rozhraní by měl přístup každý potenciální zákazník. Druhou možností je realizace distribuční sítě pomocí několika obchodních míst, která jsou navzájem propojena privátní nebo veřejnou sítí.

Výhodou návrhu distribuční sítě pomocí veřejného rozhraní je zejména vysoká dostupnost pro zákazníky. Tato metoda navíc zcela odstiňuje prodej od fyzických distributorů, je tedy velmi efektivní a šetří prostředky provozovatele. Problémem je však nutnost elektronické evidence plateb a také fyzické předávání vstupenek.

I přesto, že téměř všechny platební ústavy dnes poskytují pro své klienty službu elektronického bankovnictví, velká část klientů nemá tuto službu u své banky zřízenu. Hlavními důvody tohoto jednání, se kterými se dnes velmi často setkáváme, jsou zejména nedůvěra v elektronické bankovnictví, strach z možného zneužití a strach z prozrazení citlivých údajů při bankovní transakci.

Systém, který by chtěl realizovat možnost elektronické platby přes internet, by musel implementovat rozhraní pro komunikaci s bankou. Zákazník by zaplatil požadovanou částku přes online aplikaci své banky, popřípadě přímo v bance. Rozhraní pro komunikaci s bankou by pouze zjišťovalo, zda platba za vstupenky byla v bance provozovatele zaevidována, tedy zda na účet provozovatele přišla daná částka společně s údaji o zákazníkovi (například ve formě specifického symbolu ve

tvaru rodného čísla). Pro každý platební ústav by toto rozhraní pravděpodobně bylo jiné. Mohlo by se pak velmi lehce stát, že by systém podporoval zjišťování zaevidovaných plateb pouze u některých bank. Pokud by systém podporoval většinu velkých bank, nemuselo by toto být na škodu. Omezovalo by to však některé provozovatele, ti by si kvůli systému museli zřídit konto v jedné z podporovaných bank.

Problémem veřejné prodejní sítě je také elektronické předávání vstupenek. Vstupenka v elektronické podobě by měla obsahovat ochranné prvky, které by zabránily jejímu padělání. Ochrana vstupenky by mohla být realizována například pomocí číselných kódů, dostatečně dlouhý náhodný kód by minimalizoval riziko udání padělané vstupenky. Číselné kódy by pak musely být kontrolovány v místě prodeje, kontrola by probíhala nejspíše díky přístupu do databáze. Alternativou pro číselné kódy je použití čárových kódů, jejich použití by ulehčilo kontrolu na vstupu. Ke kontrole čárových kódů je však nutné mít speciální zařízení, které by je dokázalo číst.

Realizace prodeje pomocí sítě obchodních míst je dnes nejpoužívanějším druhem distribuce vstupenek. Výhodou tohoto prodeje je zejména větší kontrola uživatele nad prodejem. Pokud prodej probíhá pouze v několika prodejních centrech, která jsou umístěna na jednom místě (například jednotlivé brány stadionu), jsou data o prodeji šířena pouze po soukromé síti, což znemožňuje vnější útoky na aplikaci. Nevýhodou prodeje přes provázanou síť prodejních center je ale nutnost zapojení distributorů do prodeje.

Při prodeji vstupenek přes distributory může systém podporovat bezhotovostní platby. Zákazník by v takovém případě mohl uhradit částku platební kartou. Tento druh platby za vstupenky by byl realizován prodejním terminálem, který by provedl finanční transakci mezi bankou zákazníka a bankou provozovatele. Nastává zde podobný problém jako u evidence plateb, terminál by musel umožňovat převody mezi všemi bankami.

Částečného odstranění nedostatků těchto dvou metod prodeje je možno dosáhnout implementací obou rozhraní, která by fungovala paralelně. To by umožnilo jednak elektronickou online koupi vstupenky, jednak koupi přes distributora – systém by tak byl variabilnější a přizpůsoboval by se požadavkům obou typů zákazníků.

Projekt implementuje pouze rozhraní pro prodej přes distributory, které je nutné vzhledem k definovaným požadavkům. Jednotlivá distribuční centra komunikují přes internet, což s sebou nese problém zabezpečení přenášených dat. Implementace rozhraní podporující přímý prodej a evidenci elektronických plateb by mohla být v budoucnu dalším rozšířením programu.

Designování systému Zatímco rezervační rozhraní je prakticky ve všech projektech navrženo jako webová aplikace, prodejní rozhraní pro distributory

může být navrženo několika způsoby. První možností je navrhnout toto rozhraní jako aplikaci pro web, druhou možností je implementovat rozhraní pro některou z rozšířených platforem, například pro MS Windows nebo pro Linux.

Hlavní výhodou vytvoření prodejního rozhraní jako webové aplikace je přenositelnost, aplikace může být používána na libovolné softwarové platformě. Tato implementace poskytuje také větší dostupnost rozhraní pro distributory, ti mohou prodávat vstupenky z libovolného místa s připojením k síti, na které jsou umístěna data. Problémem vytvoření rozhraní jako webové aplikace jsou však malé možnosti pro vytváření grafického uživatelského prostředí. Toto kritérium bylo hlavním důvodem, proč byla zvolena implementace prodejní rozhraní jako aplikace pro platformu MS Windows.

Návrh prodejního rozhraní pro jednu cílovou platformu s sebou nese problém složitější implementace. Aplikace, která je umístěna u uživatele, musí přistupovat vzdáleně ke sdíleným datům. Výhodou je ale možnost definovat přehlednější a uživatelsky přívětivější grafické prostředí.

Výběr vstupenek Rezervační systémy mohou uživateli v zásadě nabídnout dva způsoby výběru vstupenek. Prvním způsobem je výběr vstupenky podle cenové kategorie, čísla sedadla nebo čísla řady, uživatel nemá možnost vybírat vstupenky podle umístění v místnosti. Druhým způsobem je nabídnout zákazníkovi grafický model prostor, ze kterého si může interaktivně místa vybrat. Výhoda výběru vstupenky podle čísla je zejména velmi jednoduchá implementace. Nevýhodou je ale poměrně nepohodlný výběr bez možnosti orientovat se u umístění sedadel.

Grafický model prostor s sebou nese problém jeho vytváření, k němu je zapotřebí implementovat speciální rozhraní - editor. Toto rozhraní lze implementovat buď textově, vstupem editoru je text v definovaném formátu (například XML), nebo graficky. Výhoda grafického editoru je jednodušší ovládání pro administrátora, který si nemusí pamatovat formát textového vstupu. Grafický editor, který by byl uživatelsky přívětivý, je přitom hlavní vlastností, která chybí u většiny dostupných rezervačních systémů.

S vytvářením modelu prostor je spojeno také rozhodnutí, jaké druhy objektů by měl editor umožnit vytvářet. To samozřejmě značně závisí na typu událostí, pro které je software primárně určen.

Aby měl uživatel při rezervaci a koupi vstupenek možnost výběru jejich umístění, byl do rezervačního systému navržen editor prostor. Editace může probíhat oběma zmiňovanými způsoby (textově i graficky), výběr konkrétní metody je odsunut na uživatele software.

Granularita prodeje Rezervační systém pro kulturní akce může realizovat prodej komplikovanějších objektů, kde je s jedním objektem asociováno několik vstupenek. Příkladem takového objektu je stůl. Otázkou je, zda umožnit prodej stolu pouze v celku, nebo po částech (po jednotlivých sedadlech). Obdob-

nou situací, kterou musí řešit rezervační systémy pro hotely, je prodej pouze celých pokojů nebo jednotlivých lůžek. Z tohoto rozhodnutí plynou různé požadavky na implementaci editoru události.

Při rezervaci a prodeji jsem se rozhodl pro granularitu objektů na úrovni jednotlivých vstupenek. Tato metoda umožňuje prodat pouze část stolu, stoly mohou být ale prodávány také v celku. Při částečném prodeji může být administrátorem uměle změněn stav zbývajících sedadel na nedostupné. Tento postup je obecnější, je aplikovatelný například i na řady.

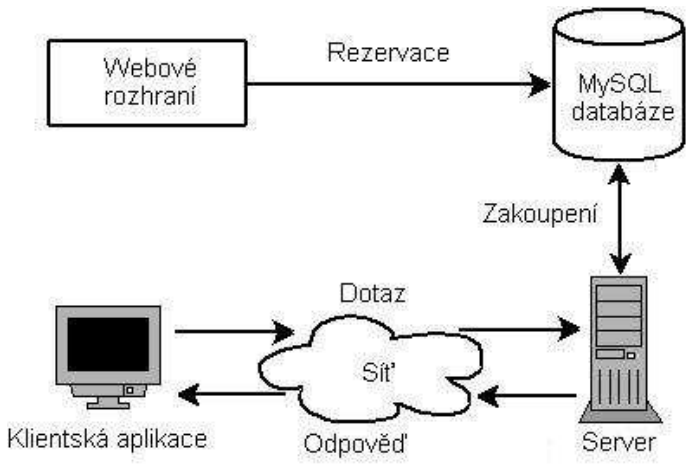
Kapitola 3

Popis řešení

3.1 Architektura

Z analýzy požadavků na vytvoření rezervačního systému plyne nutnost vytvoření dvou prostředí: webového rozhraní pro rezervaci vstupenek cílovým zákazníkem a aplikaci pro prodej vstupenek oprávněnými distributory. Rezervační a prodejní rozhraní jsou logicky oddělené části aplikace, každé je určeno pro jiný typ uživatele, obě rozhraní navíc poskytují disjunktní množinu nabízených funkcí. Pokud by aplikace byla tvořena pouze jednou monolitickou částí, která by sloužila k rezervaci i k prodeji, došlo by ke zbytečnému znepřehlednění celého návrhu.

Proto je nejlepším řešením navrhnout a implementovat části samostatně, společným prvkem je pouze jednotné rozhraní pro přístup do datových struktur, které uchovávají data sdílená oběma částmi aplikace. Schéma rozvržení a komunikace jednotlivých částí aplikace je namodelováno na obrázku 3.1.



Obrázek 3.1: Schéma aplikace

3.1.1 Uchovávání dat

Jednou z nejdůležitějších součástí návrhu aplikace uchovávající data je návrh datových struktur pro jejich uložení. Protože aplikace pracuje s centrálně uloženými sdílenými daty, ke kterým může přistupovat vícero uživatelů najednou, byla zvolena pro uložení dat databáze. Ta uchovává data strukturovaně, což poskytuje největší možnosti pro přístup, pohodlnou manipulaci s daty a usnadňuje tudíž správu datových záznamů.

Pokud by byla sdílená data uložena v datových souborech, bylo by nutné tyto soubory při každém přístupu korektně uzamykat a kontrolovat tím konzistenci jejich obsahu, což by vedlo k přílišným nárokům a zbytečně složité implementaci. Databáze nám navíc zprostředkovává možnost velmi rychle data třídit a filtrovat podle zadaného požadavku.

3.1.2 Návrh prodejního rozhraní

Aplikace sloužící pro prodej vstupenek oprávněnou osobou provozovatele je stěžejní částí celého projektu. Měla by být nainstalována v místech prodeje, požadavek na grafické rozhraní totiž neumožňuje provoz prodejního rozhraní jako webové aplikace. Jelikož aplikace umístěná u distributora zároveň zprostředkovává uživateli centrálně uložená data, byla navržena architekturou klient – server.

Klient – Server

Vztah klient – server můžeme definovat jako vztah, kdy jedna entita (server) poskytuje určité služby jiné entitě (klientovi). Hlavní výhoda implementace výpočetním modelem klient – server je zejména minimalizace množství přenášených dat po síti. Centrálně umístěná část aplikace, která přistupuje ke sdíleným datům, existuje pouze v jednom exempláři, může přitom komunikovat s více klientskými částmi najednou a v případě nutnosti může být velmi lehce aktualizováno její chování. Model klient – server navíc zcela odstiňuje klientskou aplikaci od fyzického uložení dat na serveru, což umožňuje velmi snadnou a efektivní kontrolu práv uživatelů.

Server aplikace může být umístěn na libovolném webovém serveru s podporou jazyka PHP, komunikuje s klientem pomocí protokolu HTTP systémem dotaz-odpověď. Úkolem serveru je tedy:

- Zprostředkování dat klientským částem
- Zajištění ochrany a integrity uložených dat

Klientská část aplikace byla naopak navržena jako aplikace určená pro operační systém MS Windows, toto řešení umožňuje vytvářet na klientovi rozsáhlé

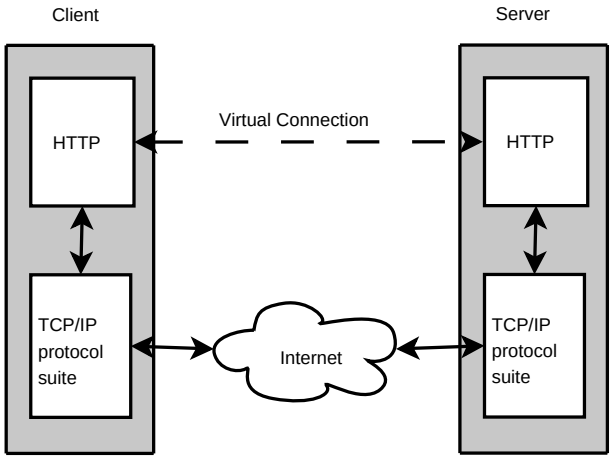
uživatelsky přívětivé grafické rozhraní.

Analýza požadavků na funkce klientského rozhraní ukázala, že je lze rozdělit do tří logicky oddělených, na sobě téměř nezávislých skupin. Jsou to:

- Modelování prostor
- Modelování událostí
- Prodej vstupenek

Aplikace implementuje tyto funkce tak, aby části byly pro uživatele co možná nejvíce odděleny, společný je pouze formát dat, která se mezi jednotlivými částmi předávají. Důsledkem toho jsou v databázi odděleně uložena data o definovaných prostorách a konkrétní parametry vstupenek.

Monolitický návrh aplikace, kde by všechny funkce splývaly, by značně snižoval přehlednost celého řešení. Pro několik událostí ve stejných prostorách by musel být model místnosti definován vícekrát, což není ani uživatelsky příjemné, ani logicky správné. Zvýšila by se tím navíc také redundance dat v databázi, což by přinášelo aktualizační problémy.



Obrázek 3.2: Komunikace přes HTTP protokol

3.1.3 Návrh rozhraní pro rezervaci

Pro rozhraní pro rezervaci vstupenek byla zvolena přímočará implementace, rozhraní bylo vytvořeno jako webová aplikace. Ta je svým umístěním na internetu

lehce zprostředkovatelná všem zákazníkům, má navíc pomocí PHP funkcí přímou možnost přístupu do MySQL databáze. Úkolem rozhraní je:

- Rezervace vstupenek zákazníkem
- Kontrola vstupů od zákazníka

3.2 Databáze

Protože aplikace pracuje s centrálními daty, ke kterým může přistupovat vícero uživatelů najednou, byla zvolena metoda ukládání dat do databáze. Výhodou této implementace oproti uložení do datových souborů na serveru je zejména pohodlnější manipulace s přehledně uloženými daty a odstranění problému nekonzistence při sdíleném přístupu více uživatelů.

Jádrem databázového přístupu je odtržení definic a údržby dat od uživatelských programů. Data již nejsou organizována v izolovaných souborech, ale v komplikovanější centrálně zpracovávané struktuře dat zvané databáze. Databázová technologie se tedy zabývá řízením velkého množství perzistentních, spolehlivých a sdílených dat[3]. Principiálně databáze zahrnuje čtyři komponenty: datové prvky (slouží k zachycení elementárních, dále nedělitelných hodnot), vztahy mezi prvky dat, integritní omezení (explicitní podmínky pro data v databázi) a schéma databáze (popis dat na úrovni uživatele databáze).

Data aplikace jsou ukládána do MySQL databáze, což je opensource relační databázový systém, který má plnou podporu v jazyce PHP, nabízí vysoký výkon a silnou ochranu dat[10]. Program byl testována na databázovém serveru s verzí MySQL 4.1. Dotazy do databáze jsou formulovány v jazyce SQL (Structured Query Language), což je neprocedurální strukturovaný dotazovací jazyk pro práci s daty v relačních databázích.

Primárním cílem návrhu databáze, podle kterého jsou vytvořeny všechny tabulky, byla minimální redundance a tudíž odstranění aktualizačních anomálií a snadná údržba dat.

Transakce / Uzamykání Protože kontrola a změna stavu vstupenek neprobíhá při prodeji atomicky, je potřeba zajistit v každém okamžiku konzistentní stav databáze. Jinak by mohla nastat situace, kdy dvě klientské aplikace úspěšně prodají stejnou vstupenku. Potřebujeme tedy, aby se určitá posloupnost dotazů na databázi jevila navenek atomicky.

Jednou z možností, jak korektně řídit posloupnosti dotazů do databáze je transakční zpracování. Transakce přitom reprezentuje logickou jednotku práce, při chybě v některém kroku transakce může být celá transakce zrušena a původní data obnovena. Problémem je, že transakce nejsou podporovány v MySQL tabulkách typu MyISAM, databázový server projektu by musel proto vytvářet tabulky

typu InnoDB, které již plně podporují transakční zpracování. Protože transakce mohou výrazně zpomalovat rychlost databáze, není tento typ tabulek standardně na freehostingu podporován. Zavedení transakcí do aplikace by tedy značně omezilo možnosti umístění databáze a serveru. Z tohoto důvodu byla zvolena jiná metoda řízení posloupnosti dotazů do databáze. Touto metodou je uzamykání. Uzamčením databáze je dosaženo vzájemného vyloučení. V kritické sekci, kde se mění stavy vstupenek, může být v daný okamžik maximálně jeden klient.

Při koupi je implementována následující posloupnost příkazů (příkazy jsou napsány v pseudokódu):

```
if uzamkni_databázi then
  {začátek kritické sekce}

  for  vybrané_vstupenky  do
    if  nelze_prodat  then
      odemkni databázi a ukonči skript chybou
    end if
  end for

  for vybrané_vstupenky do
    aktualizuj stav vstupenek a data o prodeji
  end for

  {konec kritické sekce}
  odemkni databázi
else
  vrať chybu, databáze nelze uzamknout
end if
```

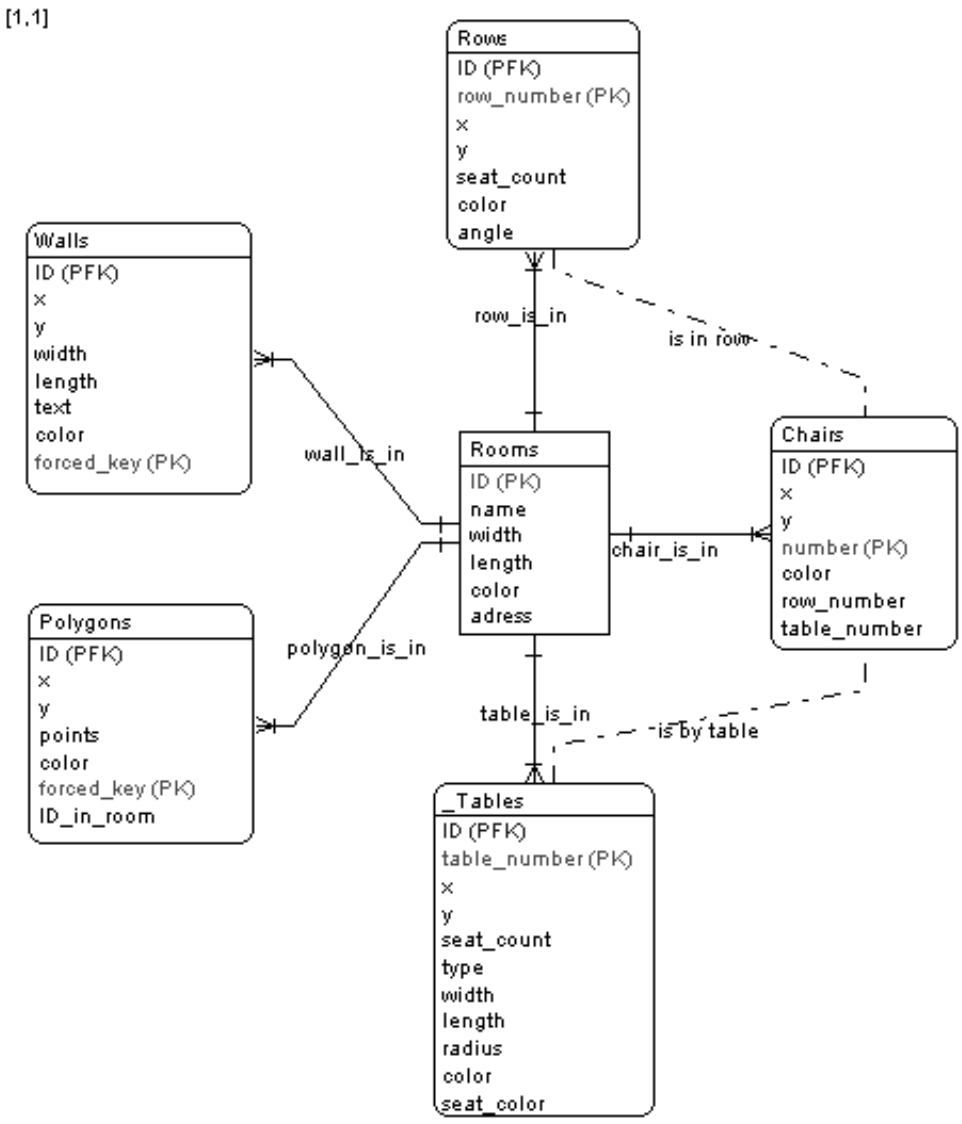
Nejdříve se server pokusí uzamknout databázi. Parametrem serveru je doba, po kterém se neúspěšné pokusy o uzamčení databáze ukončí. V případě úspěšného uzamčení se postupně zkontroluje stav všech vstupenek, pokud mezi nimi není žádná prodaná, provede se aktualizace dat. Nakonec server odemkne databázi, ukončí tak kritickou sekci a umožní vstup pro požadavky ostatních uživatelů.

Model databáze Databáze byla namodelována v programu Case Studio 2, skript RTRSdbs.sql, sloužící k vytvoření tabulek databáze, je exportem tohoto modelu do SQL. Do exportovaného skriptu je navíc přidán SQL příkaz na vytvo-

ření uživatele s loginem root a heslem root, který má přidělena maximální práva na práci s klientskou aplikací. Tento uživatel může po vytvoření databáze přes speciální rozhraní klientské aplikace přidávat nové uživatele.

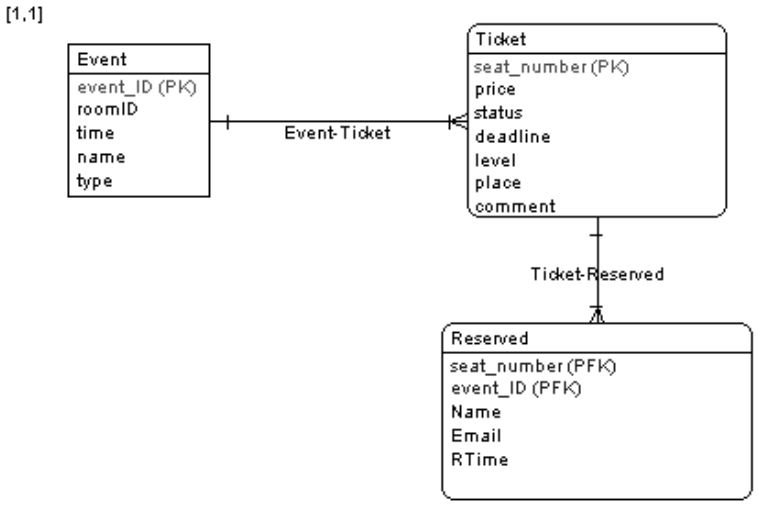
Grafický export tabulek databáze je zobrazen na obrázcích 4.1 (model tabulek prostor), 4.2 (model tabulek událostí) a 4.3 (model tabulek uživatelů).

Hlavní informace o nadefinovaných místnostech jsou uloženy v tabulce Rooms, jejímž primárním klíčem je ID místnosti. Pro každý druh objektu v místnosti je pak definována zvláštní tabulka, jejím klíčem je unikátní dvojice ID místnosti (v roli cizího převzatého klíče) a ID objektu v rámci místnosti (u objektů nevyžadujících identifikátor v rámci místnosti je tento klíč vygenerován uměle). Rozdělením objektů do tabulek podle jejich typu je snížena velikost záznamu v databázi, pokud by existovala pouze jedna obecná tabulka objektů, byly by pro některé záznamy některé sloupce nadbytečné a objem dat v databázi by zbytečně narůstal.



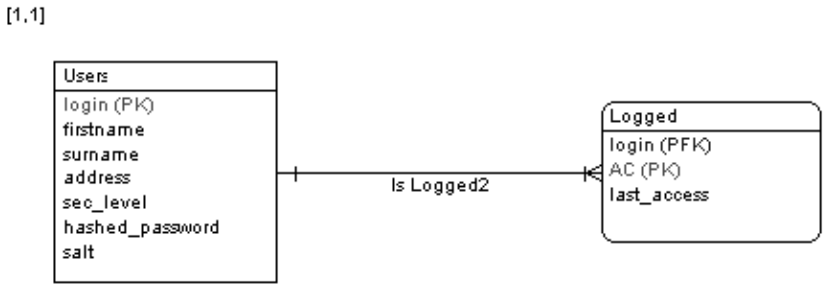
Obrázek 3.3: Tabulky reprezentující model prostor

Události nadefinované v databázi jsou uloženy v tabulce Event. Klíčem v této tabulce je unikátní identifikátor události, ten společně s číslem sedadla tvoří primární klíč tabulky Ticket, která obsahuje datové záznamy vstupenek. Aby nedocházelo ke zbytečnému zvětšení tabulky Ticket přidáním atributu s informací o zákazníkovi, který rezervoval vstupenku pomocí webového rozhraní (velká část záznamů by tuto položku měla nastavenu na NULL), jsou tyto informace uloženy v tabulce Reserved, která přebírá kompletní primární klíč z tabulky Ticket.



Obrázek 3.4: Model tabulek událostí

V tabulce Users jsou uchovávány informace o uživatelích klientské aplikace, kteří mohou přistupovat do databáze přes PHP server. Klíčem tabulky je login uživatele, ten tedy musí být unikátní v rámci celé databáze. Záznamy o přihlášených uživatelích se ukládají do tabulky Logged, jejím primárním klíčem je dvojice Login (cizí klíč) a AC (autentizační kód). Vztah mezi tabulkami User a Logged má kardinalitu 1:N, pro jednoho uživatele klientské aplikace může existovat současně více záznamů o nalogování (uživatel může být přihlášen přes několik klientských aplikací). Dvojice Login,AC je však vždy unikátní a jednoznačně identifikuje konkrétní session uživatele.



Obrázek 3.5: Model tabulek uživatelů

3.3 Klientská aplikace

Klientská aplikace je určena pro operační systém MS Windows XP, byla vytvořena v jazyce C/C++, v prostředí Microsoft Visual Studia 6.0. Pro práci s okny využívá rozhraní WIN API a také jeho nadstavbu třídy MFC (Microsoft Foundation Classes). Jako datové struktury jsou použity převážně třídy STL (Standard Template Library) knihovny.

Cílovými uživateli klientské aplikace jsou distributoři pověření provozovatelem software. Aplikace poskytuje zejména funkce pro modelování prostor, modelování událostí a pro prodej vstupenek.

3.3.1 Objektový model

Aplikace je kvůli přehlednosti, čitelnosti a bezpečnosti navržena objektově, pro místnosti, vstupenky, uživatele a jiné jsou navrženy a implementovány třídy. Objektově orientované programování umožňuje popisovat problémy způsobem blízkým přirozenému jazyku a neomezovat se pouze na strohé počítačové vyjadřování[1], přispívá tak k výraznému zjednodušení řešení široké škály problémů. Nevýhodou objektového návrhu může být menší hospodárnost a rychlost. Tyto vlastnosti jsou sice v dnešní době stále důležitými kritérii, v zásadě byl však zájem o ně odsunut na druhé místo kritérii, jako je čitelnost kódu, rozšiřitelnost a elegance návrhu. Ty jsou hlavními výhodami objektově orientovaného programování.

3.3.2 Komunikační rozhraní aplikace

Klientská aplikace je navržena tak, aby mohla pracovat ve dvou komunikačních rozhraních.

První rozhraní je určeno pro lokální komunikaci, data se načítají a ukládají do lokálních textových souborů. Motivací pro toto rozhraní je situace, kdy uživatel nepotřebuje centralizovat data. Aplikace tak může plnit roli jediného terminálu pro prodej vstupenek přímo v místě konání akce. Co se týká nabízených funkcí, je lokální rozhraní rovnocenné s rozhraním síťové komunikace, není v něm k dispozici pouze autentizace uživatelů, protože data v souborech nejsou sdílená.

Druhé rozhraní je určeno pro práci s centralizovanými daty ze serveru. Protože je v tomto rozhraní potřeba kontrolovat práva uživatelů a řídit přístup k datům v databázi, je nutné se před prací autentizovat přihlašovacím jménem a heslem. Server poté kontroluje oprávněnost každého jemu zaslaného dotazu.

Výhodou implementace dvou rozhraní je možnost provozu klientské aplikace v prostředí bez síťové komunikace, aniž by bylo nutné instalovat lokální databázový server.

3.3.3 Práva a role uživatelů

Vzhledem k tomu, že databáze musí být přístupná přes rozhraní pro rezervaci libovolnému potenciálnímu zákazníkovi, jsou data klasicky umístěna na veřejné síti. To s sebou nese nebezpečí útoku na utajení a konzistenci dat, je tedy nutné důsledně řídit a kontrolovat přistup do databáze.

Uživatelé klientské aplikace se musí před prací autentizovat svým loginem a heslem. Kontrola práv uživatele probíhá jednak v klientské aplikaci, jednak na serveru při každém přístupu do databáze. Podle požadavků na provozování aplikace a využívání nabídnutých funkcí má uživatel přiřazenu roli, což je ucelená pojmenovaná množina práv, která odpovídá svým obsahem okruhu jeho práce. Toto rozdělení plyne z bezpečnostního principu co nejmenších práv. Podle tohoto principu by každý subjekt měl mít pouze nejmenší možná oprávnění nutná ke korektnímu plnění jeho úkolu. Přiřazení rolí uživatelům snižuje možnost průniku v případě selhání části ochranného mechanismu.

Analýzou postupu vytváření událostí a prodeje vstupenek byly navrženy a implementovány 3 uživatelské role: Administrator, Creater a Reserver. Role Reserver obsahuje souhrn práv nutných k prodeji vstupenek. Je to role určená pro běžné distributory. Vlastní nadmnožinu těchto práv pak tvoří role Creater, uživatel s touto rolí má právo navrhovat a editovat prostory akcí, vytvářet a upravovat události v databázi. Absolutní práva na aplikaci má pak uživatel s rolí Administrator, kromě již zmíněných práv garantuje role právo přes specifikované rozhraní aplikace přidávat nové uživatele do databáze a přidělovat jim roli.

Hierarchie uživatelů se kromě kontroly oprávnění využívají také při prodeji, systém umožňuje nadefinovat „prioritní“ vstupenky. Každá vstupenka má přiřazenu úroveň ochrany, která je při prodeji porovnávána s úrovní oprávnění uživatele. Tento návrh tak umožňuje definovat například vstupenky, které smí prodat pouze uživatel s rolí administrátora. Tím je zvětšena granularita ochrany vstupenek, pro povolení prodeje už nestačí pouze samotný záznam o uživateli v databázi, prioritním prvkem kontroly se stává úroveň jeho oprávnění.

3.3.4 Algoritmy vyhledávání

Klientská aplikace obsahuje implementaci algoritmů na prohledávání místnosti. Tyto algoritmy umožňují automaticky přidělovat požadavky zákazníků. Při vývoji bylo navrženo několik algoritmů pro procházení datových struktur prostor, některé z nich se však díky své velké časové složitosti ukázaly jako v praxi nepoužitelné.

Časová složitost „Časovou složitostí algoritmu rozumíme závislost jeho časových nároků na velikosti konkrétního řešeného problému nebo konkrétních vstupních dat. Časovou složitost určujeme pouze počtem elementárních operací

(kroků algoritmu), které budou provedeny při výpočtu programu s danými vstupními daty. Je to tedy funkce, která každé hodnotě N udávající velikost konkrétního řešeného problému přiřazuje počet operací vykonaných při výpočtu podle daného algoritmu.“[4]

Vzhledem k tomu, že nadefinované prostory mohou obsahovat řádově stovky až tisíce sedadel, která se při vyhledávání procházejí, je pro nás časová složitost hlavní kvalitativní charakteristikou algoritmu. Při práci s velkým množstvím dat může totiž i malé zefektivnění prováděných algoritmů přinést výrazný časový a ekonomický efekt.

Chování vyhledávacích algoritmů se snaží co nejblíže simulovat požadavky reálného zákazníka. Strategií automatického přidělování je umístit požadavek na mapě prostor co nejblíže k sobě a rozdělit jej přitom mezi co nejméně objektů. Schéma prohledávání prostor je následující: nejdříve se algoritmus pokusí umístit požadavek k jednomu stolu, poté do jedné řady, pokud ani jedna ze zmíněných možností neexistuje, volá funkci na kombinované umístění požadavku. Uživatel má mimoto k dispozici i speciální způsob vyhledávání, kterým je výběr „ideálního“ stolu.

Best Table Best Table je algoritmus navržený pro výběr „ideálního“ stolu pro danou velikost požadavku. Při návrhu algoritmu byly zvažovány výhody a nevýhody implementace pomocí jednotlivých metod pro přidělování zdrojů. Porovnávány byly zejména metody FIRST FIT, BEST FIT a WORST FIT.

Po zvážení aspektů všech metod algoritmus implementuje přidělování zdrojů metodou BEST FIT, vrací tedy vždy nejmenší stůl dostatečné velikosti. Výhodou tohoto návrhu je zejména fakt, že takto nedochází ke zbytečné fragmentaci velkých stolů kvůli malým požadavkům. Nevýhodou je naopak rychlost, algoritmus musí projít vždy všechny stoly ve struktuře. Další negativní vlastností chování algoritmu je vznik malých fragmentů, které zůstanou pravděpodobně neobsazeny. Při výběru algoritmu bylo přihlédnuto k předpokladu, že malý počet nevyužitých sedadel u jinak obsazeného stolu je menším problémem než celková fragmentace prostoru. Pokud by stoly byly přidělovány neúsporně a prostor by se tím fragmentoval, nastávala by často situace, kdy požadavek zákazníka nelze umístit pouze k jednomu stolu a musí být tudíž rozdělen. Časová složitost algoritmu je lineární vzhledem k počtu stolů v místnosti.

Search Closest Algoritmus nejdříve pevně zvolí první, centrální sedadlo, postupně prochází zbylá sedadla a k vybraným přidá v každém kroku to, které není vybráno a je k centrálnímu sedadlu nejblíže. Tento postup je aplikován n krát, kde n je počet sedadel v místnosti. V každém kroku je vybráno jiné centrální sedadlo. Výsledkem algoritmu je umístění požadavku do kruhu s nejmenším možným poloměrem vzdálenosti od centrálního sedadla. Časová složitost algoritmu je polynomiální řádu n^3.

Matrix Algorithm Algoritmus využívá k prohledávání prostor speciálně navrženou nepolymorfní datovou strukturu, čtvercovou matici záznamů o straně velikosti počtu volných sedadel. Záznam na pozici (i, j) obsahuje datové položky číslo vrcholu j a vzdálenost vrcholů i,j. Nejprve se setřídí každý řádek matice vzestupně podle vzdálenosti, čímž pro každý vrchol získáme setříděnou posloupnost jemu nejbližších vrcholů. Výsledek pak tvoří prvních k sedadel z řádku s minimální hodnotou vzdálenosti na pozici $k - 1$. Hlavní předností algoritmu je jeho menší časová složitost, ta je řádově rovna $n^2 * \log n$. Nevýhodou je pak, že třídění dat probíhá ve speciálně vytvořené struktuře, což zvyšuje paměťovou složitost algoritmu řádově na n^2.

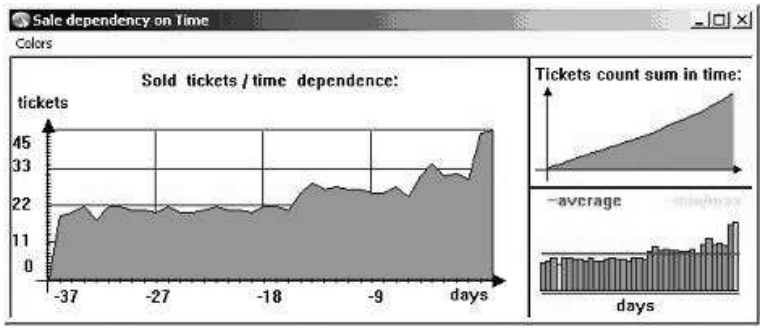
Z teoretické časové složitosti algoritmů plyne, že se podstatněji liší až při prohledávání velkých vstupních dat (z asymptotické časové složitosti). Můžeme si všimnout také faktu, že rychlost maticového algoritmu závisí pouze na velikosti prohledávaných prostor, nezávisí na samotné velikosti požadavku, zatímco u druhého algoritmu tato závislost existuje.

Jak bylo zmíněno již dříve, je pro nás hlavním požadavkem pro výběr algoritmu jeho časová složitost. Maticový algoritmus se tedy podle našich kritérií rozhodování ukázal jako efektivnější. Není sice optimální z hlediska paměťové složitosti, nevede však k příliš extrémním nárokům na paměť, a proto je v aplikaci používán implicitně. Protože výběr nejlepšího algoritmu není jednoznačný a záleží vždy na podmínkách, jaké pro řešení máme, je možnost změny algoritmu odsunuta na uživatele software, ten má k dispozici volbu algoritmus změnit a přizpůsobit se tak svým výpočetním možnostem (rychlosti procesoru nebo velikosti hlavní paměti).

3.3.5 Vizualizace statistických dat

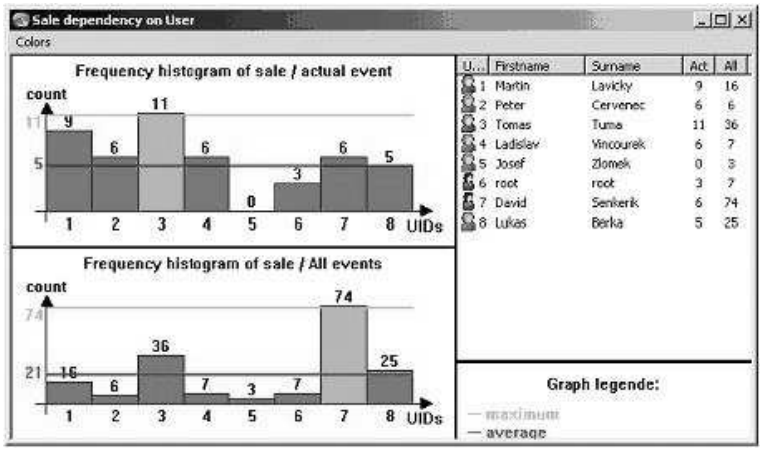
Při vývoji software byly zvažovány dvě metody zobrazení získaných dat: pomocí tabulek a pomocí grafů. Primárně byly za zobrazovací metodu zvoleny grafy, jejich výhodou oproti zobrazení dat do tabulek je zejména to, že uživateli vizuálně usnadňují čtení informace. Každý přihlášený uživatel má tak k dispozici názorná statistická data o prodeji. Při vytváření statistik se sledují dvě základní závislosti: závislost prodeje na čase (SDT) a závislost prodeje na distributorech (SDU).

Statistika SDT (Sale Dependency on Time) obsahuje informace o vývoji prodeje v čase. Hlavní graf je tvořen polygonem četností prodaných vstupenek v jednotlivých dnech. Okno kromě toho zobrazuje také četnostní histogram prodeje včetně zvýraznění průměru, maxima a minima. Příklad zobrazení dat je na obrázku 3.6.



Obrázek 3.6: Graf vývoje prodeje v čase

Statistika SDU (Sale Dependency on User) zobrazuje z databáze úspěšnost jednotlivých prodejců. Ze statistického souboru dat se vytváří dva četnostní histogramy. První je vytvořen z dat pro aktuálně zvolenou událost, druhý ze souhrnu dat pro všechny události. Na osu x jsou vyneseny intervaly, reprezentující jednotlivé distributory, na osu y pak absolutní četnosti prodaných vstupenek. Data jsou kromě sloupcových grafů zobrazena také textově ve formě tabulky.



Obrázek 3.7: Histogramy prodeje podle uživatelů

3.3.6 Vícejazyčná podpora

Klientská aplikace je navržena tak, aby poskytovala uživateli vícejazyčnou podporu. Důraz byl kladen zejména na obecnost jazykové podpory, řetězcové konstanty nejsou v aplikaci zabudovány pevně, ale ve formě dynamicky linkovaných knihoven.

Pokud uživatel přidá do adresáře *./Languages* novou dll knihovnu, aplikace nabídne uživateli jako položku hlavního menu volbu tento jazyk použít. Vybraná

knihovna je načtena pomocí funkce LoadLibraryEx, funkce obdrží na vstupu flag LOAD_LIBRARY_AS_DATAFILE, dynamická knihovna je namapována tak, jakoby neobsahovala žádný kód, slouží pouze k získávání řetězců ze String Table.

Návrh umožňuje v případě potřeby velmi lehce přidávat do aplikace podporu pro další jazyky, stačí pouze překladem řetězců ze String Table již existující knihovny vytvořit novou dynamicky linkovanou knihovnu. Základními jazyky jsou čeština a angličtina, knihovny s těmito jazyky (*Czech.dll*, *English.dll*) byly vytvořeny při vývoji software a jsou šířeny společně s aplikací.

3.3.7 Vykreslování do paměti

Hlavním problémem při implementaci grafického uživatelského rozhraní klientské aplikace byla rychlost zobrazování dat. Protože definice místnosti může obsahovat řádově stovky objektů, je nutné, aby vykreslování mapy prostor probíhalo co nejefektivněji. Neustálé překreslování místnosti a přímé vykreslování do bufferu grafické karty způsobují nepříjemné a rušivé blikání. Byla proto zvolena metoda vykreslování objektů do paměti.

Princip metody je založen na faktu, že zprávy na překreslení okna dochází většinou bez změny zobrazovaných dat. Příkladem je překrytí části okna oknem jiné aplikace nebo vlastním dialogem. Po obnovení této části okna je operačním systémem poslána zpráva WM_PAINT na překreslení zneplatněné části okna. Dalším příkladem je minimalizace a následné obnovení okna. V tomto případě je za zneplatněnou část označeno celé okno.

Při vykreslování do bufferu grafické karty dochází k pevné změně pixelů, předchozí stav pixelů není zachován. Obsah okna po odkrytí nemůže být tedy obnoven pouze z momentálně pamatovaných dat.

Řešení problému, které je implementováno v klientské aplikaci, je následující. Stačí si ve speciální struktuře v paměti pamatovat aktuální stav zobrazované bitmapy. Pokud uživatel změní vykreslovaná data, aplikace překreslí data do této struktury. Zprávy na překreslení okna, posílané operačním systémem, jsou pak velmi efektivně ošetřeny pouhým kopírováním příslušné oblasti z paměti.

Je zřejmé, že výhodou této metody je zejména její rychlost, kopírování úseku paměti je zcela jistě rychlejší než volání GDI funkcí pro kreslení 2D objektů. Klientská aplikace touto metodou zobrazuje všechny grafické výstupy.

3.4 Server

Klientská aplikace v síťovém rozhraní komunikuje vzdáleně přes HTTP protokol se serverovou částí. Začlenění serveru do projektu je vynuceno tím, že u některých webhostingů je databáze MySQL přístupná pouze lokálně (prostřednictvím

PHP skriptů). Není tedy možné přistupovat do takové databáze vzdáleně přímo z klientské aplikace, která je umístěna u distributora.

Server je navržen jako PHP aplikace, díky tomu může být velmi snadno provozován na většině veřejných webových serverů (je požadována pouze podpora jazyka PHP). Skript byl testován na webovém serveru s verzí PHP 4.3.4.

Hlavní úloha serveru je formou dotaz-odpověď zprostředkovávat klientovi přístup k datům v databázi. To, že samotná klientská aplikace je fyzicky odstíněna od uložených dat, poskytuje poměrně snadnou a efektivní možnost kontroly uživatelských práv. Server při zpracování každého dotazu od klienta volá bezpečnostní funkce, které ověří identitu uživatele, jeho přihlášení a zda jemu přidělená role garantuje dostatečná práva na provedení požadovaného dotazu.

Data se na server posílají přes požadavek typu POST. Klient nastaví při každém dotazu na server proměnnou *query*, hodnota této proměnné určuje typ požadovaných dat. Pokud proměnná není na serveru nastavena, ukončí se skript okamžitě chybou. Předdefinované hodnoty dotazů jsou uloženy v souboru *QueryConst.h*, jejich hodnoty musí odpovídat konstantám nadefinovaným pro dotazy na serveru.

Významnou úlohou serveru je kromě zprostředkovávání dat z databáze také kontrola uživatelských vstupů. Server každou externí proměnnou, která se má stát součástí dynamicky konstruovaného SQL dotazu, explicitně přečte a zkontroluje, zda její obsah není potenciálně nebezpečný. Tímto server zamezuje útokům typu SQL Injection (princip útoku a kontrola uživatelských dat je popsána podrobně v kapitole 3.6.3).

3.5 Webové rozhraní

Webové rozhraní pro rezervaci je naprogramováno a zvalidováno podle normy XHTML 1.0 Transitional, pro definici vzhledu používá kaskádových stylů, vzhled aplikace je tak separován od zobrazovaného obsahu a může být změněn nadefinováním nového stylového souboru.

Účelem vytvoření webového rozhraní je možnost zákazníka rezervovat si vstupenku před zakoupením u distributora, zákazník má tak větší přehled o míře obsazenosti události, kategoriích a cenových relacích vstupenek.

Zákazníkovi jsou posloupností několika kroků nabídnuty postupně: výběr cenové kategorie vstupenky, náhled na mapu prostor (klientská aplikace umožňuje ukládat náhledy, možná je ale libovolná mapa rozložení prostor), dynamicky z databáze generovaný model aktivních sedadel v dané cenové kategorii, přehled vybraných vstupenek a formulář na zadání osobních údajů.

Po potvrzení požadavku na rezervaci se informace o zákazníkovi uloží do databáze, rezervovaná místa jsou pak ve webové aplikaci pro další zákazníky neaktivní, jejich stav může být změněn pouze vlastníkem při zrušení rezervace přes webové

rozhraní nebo použitím klientské aplikace (prodejem vstupenky nebo vynuceným zrušením rezervace).

Parametrem webového rozhraní je také časový interval, po kterém je rezervace automaticky zrušena. Při každém přístupu do databáze jsou smazány starší záznamy o rezervaci. Provozovatel může tuto dobu nastavit například na 5 dnů, po tuto dobu bude rezervace platná, pak bude automaticky zrušena a vstupenka bude dále vedena jako volná. Poté ji tedy může rezervovat jiný zákazník.

3.6 Bezpečnost

Z charakteru rezervačního systému plyne nutnost umístit data tak, aby byla přístupná přes rozhraní pro rezervaci všem potenciálním zákazníkům. Data budou tedy klasicky umístěna přímo na internetu. Toto uchovávání dat zvyšuje možnost útoku na aplikaci. Bylo tedy nutné stanovit si kritéria pro zabezpečení aplikace.

Při návrhu prodejní části aplikace byla definována následující bezpečnostní kritéria, která se snaží co nejvíce minimalizovat nebezpečí útoku:

- Autentizace uživatele
- Autorizace každého dotazu
- Kontrola uživatelských dat
- Utajení hesla při komunikaci
- Utajení záznamu hesla v databázi

Klasifikace útoků podle útočníka V principu mohou být útoky podle útočníka rozděleny na dvě skupiny: útočníkem může být cizí osoba, ale také vlastní uživatel software. Útok tedy můžeme chápat v zásadě jako vnitřní nebo vnější.

Vnitřní útok Před vnitřním útokem, tedy před zneužitím samotným uživatelem, se aplikace brání definicí a důslednou kontrolou uživatelských práv. Uživatelům jsou přiděleny podle jejich požadavků na aplikaci role, které garantují přidělení nejmenších potřebných práv pro danou oblast činnosti. Jednotlivé role jsou popsány v kapitole 3.3.3. Práva jsou kontrolována jednak na serveru před přístupem do databáze, jednak v klientské aplikaci. Zdvojením kontroly je snížena možnost získání přístupu k datům přes klientskou aplikaci při selhání jedné z těchto částí ochrany.

Vnější útok Hlavní bezpečnostní princip, který brání vnějšímu útoku, je autentizace každého uživatele klientské aplikace, který přistupuje k centrálním

datům. S tím je spojen problém utajení hesla při komunikaci. Heslo musí být také bezpečně uloženo na serveru, to brání ztrátě citlivých informací při prolomení bezpečnosti databáze.

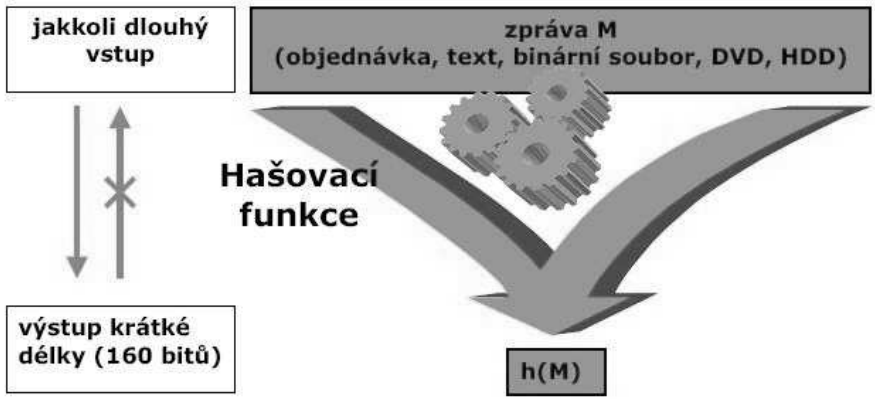
Další bezpečnostní úlohou je rozpoznávat nebezpečné uživatelské vstupy, které by mohly způsobit ztrátu nebo neautorizovanou změnu dat v databázi. Nebezpečí tohoto útoku je, že může přijít od obou typů útočníků. Je tedy nutné kontrolovat jednak vstupy z rozhraní pro rezervaci, jednak vstupy z klientské aplikace.

3.6.1 Přenos hesla po síti

Klientská aplikace pracuje v rozhraní síťové komunikace se sdílenými daty, vyžaduje tedy autentizaci uživatelským jménem a heslem. Vzhledem k tomu, že heslo je primární autentizační prvek uživatele a uživatelé používají velmi často stejná hesla ve více aplikacích, je základním požadavkem bezpečnosti aplikace utajení hesla při síťové komunikaci. Heslo se proto nikdy nevyskytuje na síti jako otevřený text, což znemožňuje jeho získání monitorováním komunikace. K tomuto účelu používá klientská aplikace implementaci kryptografické hašovací funkce MD5.

Hašovací funkce je používána nejen při autentizaci, ale také při registraci nového uživatele klientské aplikace či při změně hesla - v databázi je obdobně uložen pouze otisk hesla.

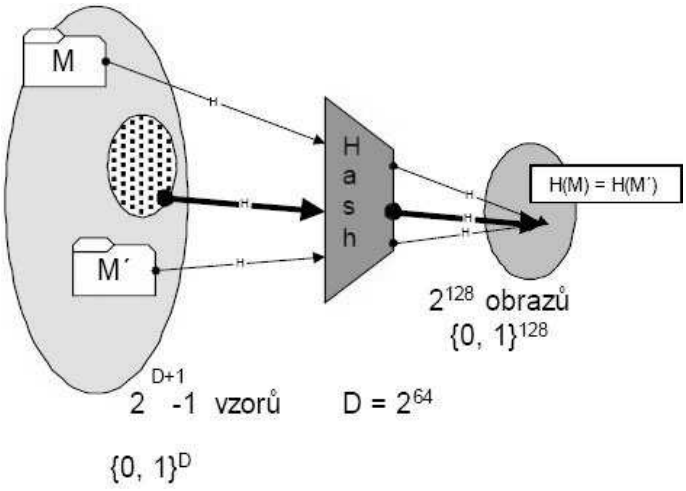
Definice: Kryptografická hašovací funkce je definována jako jednosměrná (one-way), bezkolizní (collission-free) funkce, která nevratným procesem přiřazuje libovolně velkému vstupu krátký hašový kód pevně definované délky (otisk nebo message digest).



Obrázek 3.8: Schéma hašovací funkce [2]

Vlastnost bezkoliznosti je přitom velmi často chápána špatně. Funkci $f : X \rightarrow Y$ nazveme bezkolizní, jestliže je výpočetně velmi složité nalézt různá x, y z X, že $f(x) = f(y)$. Z definice hašovací funkce je zřejmé, že tato není prostá, existuje tedy velké množství zpráv vedoucích na tentýž hašový kód, nalezení takové kolize však musí být nad momentální výpočetní možnosti.

Pro zabezpečování hesel je důležitá zejména vlastnost jednosměrnosti, která zaručuje poměrně jednoduchý výpočet otisku z otevřeného textu hesla, opačná transformace je však výpočetně neproveditelná.



Obrázek 3.9: Schéma kolize dvou zpráv při použití hašovací funkce [2]

Z hlediska bezpečnosti bychom byli rádi, kdyby se hašovací funkce chovala jako náhodné orákulum. Orákulum nazýváme libovolný stroj, který na základě vstupu odpovídá nějakým výstupem. Má pouze vlastnost, že na tentýž vstup odpovídá tímtéž výstupem. Náhodné orákulum je pak orákulum, které na nový vstup odpovídá náhodným výběrem výstupu z množiny možných výstupů[2].

U prakticky používaných hašovacích funkcí není prokázána výpočetní složitost nalezení kolize nebo druhého vzoru. Jejich bezpečnost tak u obou vlastností (jednosměrnost, bezkoliznost) závisí pouze na stavu vědy v oblasti kryptografie a kryptoanalýzy[5].

Autentizační kód Heslo slouží v aplikaci k primární autentizaci, po přihlášení je uživateli serverem přidělen náhodný autentizační kód, který zastupuje roli hesla při autorizaci dotazů do databáze. Takto je snížena frekvence odesílání otisku hesla a tím i nebezpečí jeho zachycení útočníkem. Autentizační kód je uložen v databázi pouze po dobu přihlášení uživatele, při novém přihlášení je vygenerován nový náhodný autentizační kód. Autentizační kód se tedy chová jako s časem proměnné heslo, po prozrazení ho lze používat pouze relativně krátkou

dobu. Výhodou této implementace je taktéž možnost současného přihlášení několika uživatelů pod jedním přihlašovacím jménem (například host), každé instanci klientské aplikace je přidělen její vlastní autentizační kód, což umožňuje separovat dotazy jednotlivých instancí a monitorovat tak chování uživatele.

Aby se při nestandardním ukončení klientské aplikace (výpadek proudu) nehromadily v databázi staré záznamy o nalogování uživatelů, jsou tyto po určitém čase od posledního dotazu uživatele smazány. Časový interval, po kterém je záznam o nalogování neaktivního uživatele smazán, je nastavitelný konfigurační konstantou serveru.

3.6.2 Bezpečné uložení hesla v databázi

V aplikacích, které vyžadují autentizaci uživatele, je jedním ze základních problémů bezpečnosti uchovávání hesel. Ta se typicky ukládají do databáze. Poměrně velkou chybou v bezpečnosti aplikace je uchovávat heslo bez kódování, tedy přímo ve formě zadání uživatelem. Výhodou této přímočaré implementace je sice možnost odesílání hesla vlastníkovi při jeho zapomenutí, skrývá však riziko získání všech autentizačních údajů ať už správcem nebo útočníkem při prolomení databáze. Proto je výhodnější uchovávat v databázi pouze otisk hesla, který získáme použitím hašovací funkce. Při autentizaci se pak stejnou hašovací funkcí vytvoří otisk zadaného hesla, ten se porovnává s otiskem z databáze.

Solení hesla Použití hašovací funkce (definice 3.6.1), která se chová z definice jako náhodné orákulum - na stejný vstup odpovídá stejným náhodným výstupem, s sebou nese problém kolize otisků stejných hesel. Pokud mají dva uživatelé stejná hesla, shoduje se poté i jejich otisk v databázi. Informace o shodnosti dvou otisků, pro cizího útočníka poměrně bezcenná, se stává hrozbou bezpečnosti v momentu, kdy je útočníkem jeden ze zmíněných dvou uživatelů. Ten se v takovém případě může autentizovat cizím přihlašovacím jménem a svým heslem. Této situaci se v aplikaci předchází vygenerováním náhodného řetězce *salt*, který se připojí k heslu před vstupem do hašovací funkce. Tento řetězec musí být samozřejmě uložen společně s otiskem v databázi. Při autentizaci nejdříve klient zažádá do databáze o jemu přidělený řetězec *salt*, tento připojí k heslu a spočítá jeho otisk. Otisk se poté porovnává se záznamem v databázi.

Samotná znalost řetězce *salt*, což je dotaz do databáze, který nesmí vyžadovat autentizaci uživatele (je totiž částí autentizačního procesu), je pro útočníka bez znalosti hesla bezcenná. Nebylo navíc prokázáno, že by pouhé zřetězení hesla s náhodným řetězcem snižovalo bezpečnost celého řešení[6].

Změna hesla Další problém v aplikaci s registrací a autentizací uživatelů je změna hesla. Změna hesla je podmíněna zadáním starého platného hesla a to

i v případě, že uživatel je momentálně tímto heslem autentizován. Předchází se tak situacím, kdy útočník dostane aplikaci k dispozici ve chvilkové nepřítomnosti přihlášeného uživatele.

3.6.3 SQL Injection Attack

Cíl útoku Pro potenciálního útočníka bývá často nejzajímavějším cílem útoku databáze a data v ní uložená. Útočník dokáže pomocí vložení nekorektních dat do jemu dostupné části aplikace (klientské části, webového rozhraní) způsobit škody v té části, ke které přístup nemá (v databázovém serveru). U rezervačních systémů je tento problém obzvláště citlivý, změna stavu vstupenky v databázi při útoku by mohla vést k duplicitě jejího prodeje a tímto k finančnímu poškození cílového zákazníka. Cílem útoku však nemusí být pouze změna dat v databázi. Útokem lze také získat z databáze citlivé údaje, útočník by tak mohl získat například tabulku uživatelů klientské části aplikace.

Princip útoku Takovýmto druhem útoku je SQL Injection, jeho podstatou je snaha vložením nekorektních uživatelských dat změnit sémantický význam SQL dotazu. Příkladem nekorektního vstupu jsou data, která obsahují klíčová slova SQL, znaky se speciálním významem v SQL (apostrof, středník) nebo konstrukci SQL komentáře. Vložení takového vstupu do SQL dotazu může změnit jeho význam a znamená hrozbu pro bezpečnost databázového serveru.

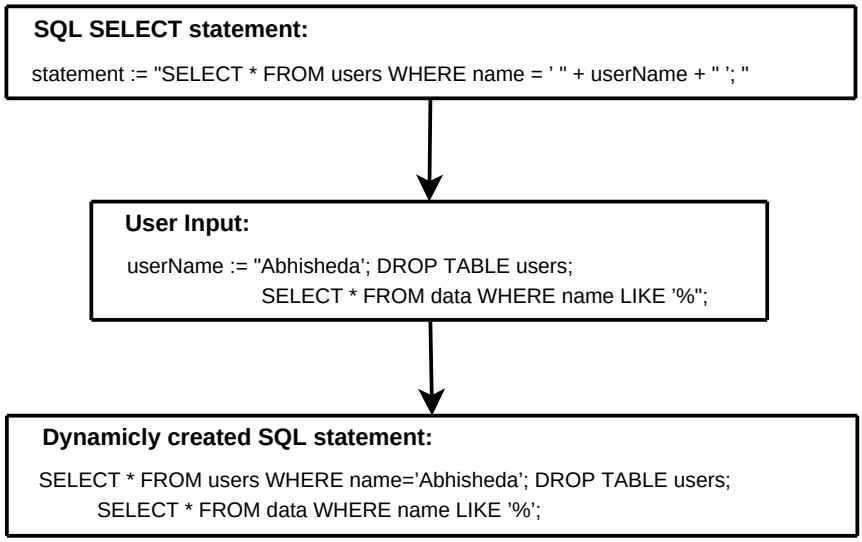
Waiting for delay Trikem waiting for delay může útočník pomocí SQL Injection vytvořit tzv. *orákulum*, které je schopné vyhodnotit v databázi libovolný logický výraz dávající výsledek *ano* nebo *ne*. Pokud je výraz pravdivý (vrací hodnotu true), počká databázový server s odpovědí po zadaný časový interval, jinak vrací výsledek okamžitě. Opakovanými dotazy a metodou půlení intervalů lze pak z těchto výrazů zjistit obsah libovolné tabulky databáze. Proto je i přes dobré zabezpečení přístupu do databáze důležité, aby se citlivé údaje (zejména pak hesla uživatelů) nevyskytovala v databázi jako otevřený text.

Ochrana dat K odstranění této slabiny je nutné identifikovat její příčinu, jíž je příliš velká důvěra a s ní spojená nedostatečná kontrola uživatelských dat, která se vkládají do SQL dotazů. Řešením, které implementuje PHP server i webové rozhraní projektu, je použití PHP funkce addslashes, která escapuje znaky se speciálním významem v jazyce SQL.

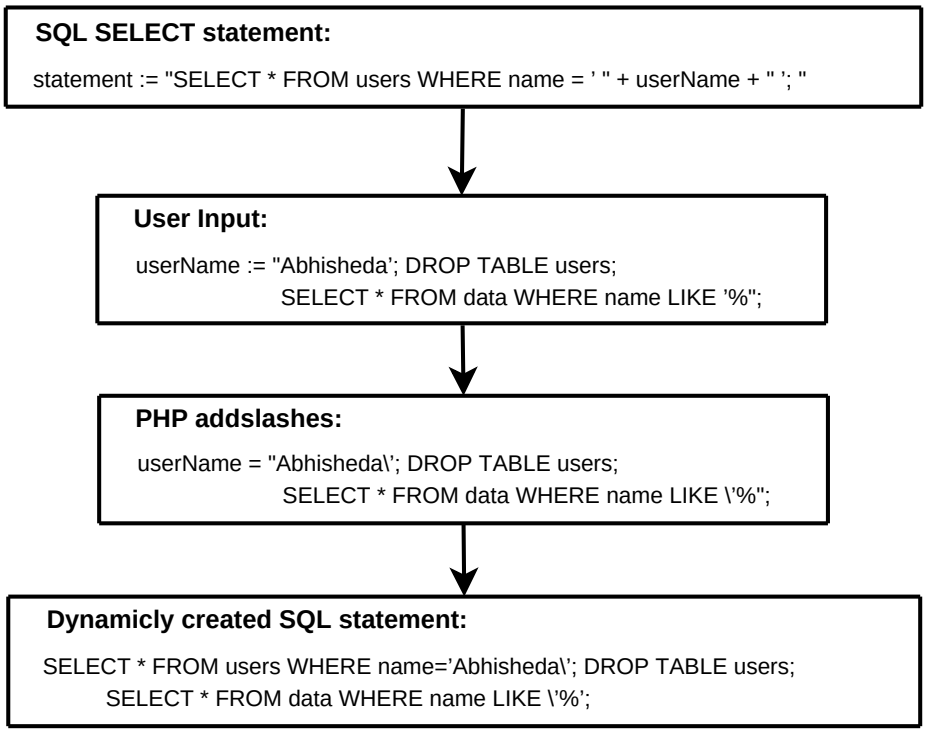
Reakce na útok V mnoha případech může aplikace útočníkovi značně ušetřit práci tím, že uživateli podrobně vrací popis chyby, pokud tato nastane v průběhu zpracování SQL dotazu. Popis chyby by neměl obsahovat jména tabu-

lek, sloupců v tabulkách, kolize datových typů a podobu samotných SQL dotazů, do kterých uživatel vkládá svá data. U víceřádkových dotazů je dokonce informace čísla řádku, na kterém se vyskytuje chyba, potenciálně nebezpečná.

Z tohoto důvodu vrací server při chybě pouze její stručný popis bez jakýchkoliv citlivých informací a detailů o její příčině, což je nejbezpečnější reakcí. Jinou možností by bylo, pokusit se chyby opravit pomocí escape sekvencí a poté dotazy provést, což by však nemuselo vést ke spolehlivým výsledkům.



Obrázek 3.10: Příklad útoku použitím SQL injection[11]



Obrázek 3.11: Ochrana před útokem - použití PHP funkce addslashes

Kapitola 4

Instalační a konfigurační příručka

Pro instalaci systému předpokládáme následující prostředí. Prodejní klientská aplikace předpokládá instalaci na PC s operačním systém MS Windows XP. Pro zprovoznění databáze, serveru a webového rozhraní může administrátor zřídit účet na některém z webových hostingů. Hosting musí podporovat jazyk PHP a umožňovat vytvoření MySQL databáze. Tyto funkce jsou v dnešní době podporovány na velké většině českých freehostingů.

Instalaci systému můžeme rozdělit na dvě části: instalaci klientské aplikace a instalaci zbylých komponent systému.

Klientská aplikace se instaluje lokálně (v místech prodeje) pomocí standardního průvodce pro instalaci programů pod operačním systémem MS Windows.

Zbylé komponenty systému (tedy databáze, server skript a webové rozhraní) by měly být umístěny fyzicky na jednom místě, důvodem je zejména problém s podporou vzdáleného přístupu do MySQL databáze. Nejlepším řešením je proto umístit tyto komponenty na jeden webový hosting s podporou jazyka PHP a s možností vytvoření MySQL databáze.

Tabulky databáze vytvoří správce systému pomocí SQL skriptu RTRSdbs.sql, který je součástí distribuce systému. Skript po vytvoření tabulek databáze přidá do tabulky uživatelů administrátora s loginem *root* a heslem *root*. Aby uživatel s tímto heslem nebyl v systému oprávněn trvale, měl by správce systému okamžitě heslo přes klientskou aplikaci změnit. Uživatel root, již s tajným heslem správce, slouží dále k vytváření dalších uživatelských účtů v systému. Uživatel s loginem root je automaticky přidán proto, aby nebylo nutné přidávat prvního uživatele do databáze přes pomocnou aplikaci (například přes aplikaci PHPMyAdmin).

Před prvním spuštěním aplikace musí uživatel provést konfiguraci jednotlivých komponent systému, ta je popsána níže.

4.1 Konfigurace klientské aplikace

Při prvním spuštění klientské aplikace musí uživatel v hlavním menu nakonfigurovat její vlastnosti. Vlastnosti se nastavují převážně pomocí položky hlavního menu *Nastavení*. Nejdůležitější konfigurační vlastností je způsob komunikace, ta je implicitně nastavena na rozhraní síťové komunikace. Uživatel, který chce pracovat v tomto rozhraní, musí specifikovat v záložce hlavního okna *Nastavení serveru* adresu, na které běží server skript. Adresa je rozdělena na adresu stroje a adresu skriptu *RTRSscript.php* na tomto stroji. Při nastavování adresy je k dispozici možnost vyzkoušet přístupnost serveru, pokud druhá strana odpoví ve správném formátu, vypíše se uživateli celkový čas komunikace. Kromě rozhraní komunikace uživatel nastavuje dále jazyk aplikace a algoritmus vyhledávání vstupenek. Implicitním jazykem je angličtina, konkrétní nabídka jazyků v menu je však závislá na přítomnosti dynamických knihoven.

Ukládání aktuální konfigurace Nastavení klientské aplikace se kvůli zjednodušení práce uživatele ukládá do konfiguračního souboru. Konfigurační soubor je vytvořen vždy při ukončení běhu aplikace, obsahuje tedy poslední konfiguraci programu. Cílem uchovávání konfigurace je, aby uživatel při každém dalším spuštění aplikace nemusel vlastnosti znova nastavovat.

Implicitní konfigurace Pokud konfigurační soubor není nalezen, je aplikace spuštěna v implicitní konfiguraci. Aplikace je v implicitním nastavení spuštěna v rozhraní síťové komunikace, zvoleným jazykem je angličtina. Adresa serveru a login uživatele nejsou automaticky předvyplněny. Implicitní konfigurace je předdefinována hodnotami konstant tak, aby bylo možno ji lehce změnit.

4.2 Konfigurace serveru

Před připojením prvního klienta by měl správce systému provést konfiguraci serveru. Konfigurace se týká zejména nastavení přístupu do databáze, v souboru *RTRSsettings.php* musí být specifikována adresa databáze, její název, login a heslo uživatele s právem přístupu. Server pak převádí dotazy klienta na SQL dotazy do této databáze.

V souboru *ConfigConstants.php* má správce systému k dispozici další konfigurační nástroje. Může nastavit například čas, po kterém dojde k automatickému odhlášení nekomunikujícího uživatele, délku generovaného řetězce *salt* (ta však musí odpovídat velikosti záznamu v databázi), časový limit pro pokus o uzamčení databáze a jiné.

4.3 Konfigurace webového rozhraní

Konfigurace webového rozhraní probíhá obdobně jako konfigurace server skriptu, správce systému musí nastavit v souboru settings.php parametry připojení k databázi. Je zde možné také specifikovat dobu, po které je rezervace vstupenky automaticky zrušena.

Kapitola 5

Uživatelská příručka

Tato kapitola popisuje stručně jednotlivé kroky postupu při vytváření událostí v klientské aplikaci. Struktura kapitoly zachovává postup, jak bude s aplikací pracovat uživatel. Podrobné informace o ovládání klientské aplikace jsou uvedeny v HTML nápovědě a v uživatelské dokumentaci projektu.

5.1 Přihlášení uživatele

Pokud chce uživatel pracovat v rozhraní síťové komunikace (nastavení rozhraní komunikace je blíže popsáno v konfiguraci klientské aplikace 4.1), musí se před prací autentizovat. Přihlášení uživatele probíhá pomocí záložky *Uživatel* v hlavním okně aplikace. Po uživateli je požadováno uživatelské jméno (může být předvyplněno jménem posledního uživatele) a platné heslo. Pokud zadá uživatel volbu *LogIn*, odešlou se tyto údaje na server, kde jsou porovnány s informacemi o uživatelích v databázi. Poté se zobrazí dialog s výsledkem autentizace.

Pokud autentizace proběhla bez chyb, zobrazí se v záložce *Uživatel* informace z databáze o právě přihlášeném uživateli. Jedná se o informace, které byly zadány při vytváření uživatelského účtu administrátorem systému. Záložka obsahuje také informaci o počtu prodaných vstupenek od přihlášení uživatele. Tento údaj je automaticky obnoven při každém prodeji.

Po ukončení práce s aplikací by se měl uživatel odhlásit ze systému. Odhlášení je opět realizováno pomocí záložky *Uživatel*, přihlášený uživatel je automaticky odhlášen také při ukončení aplikace (při zavření hlavního okna aplikace pomocí systémového menu). Pokud klientská aplikace delší dobu nekomunikuje (doba je konfigurovatelná na serveru), smaže server automaticky záznam o přihlášení. Uživatel se v takovém případě musí přihlásit znova.

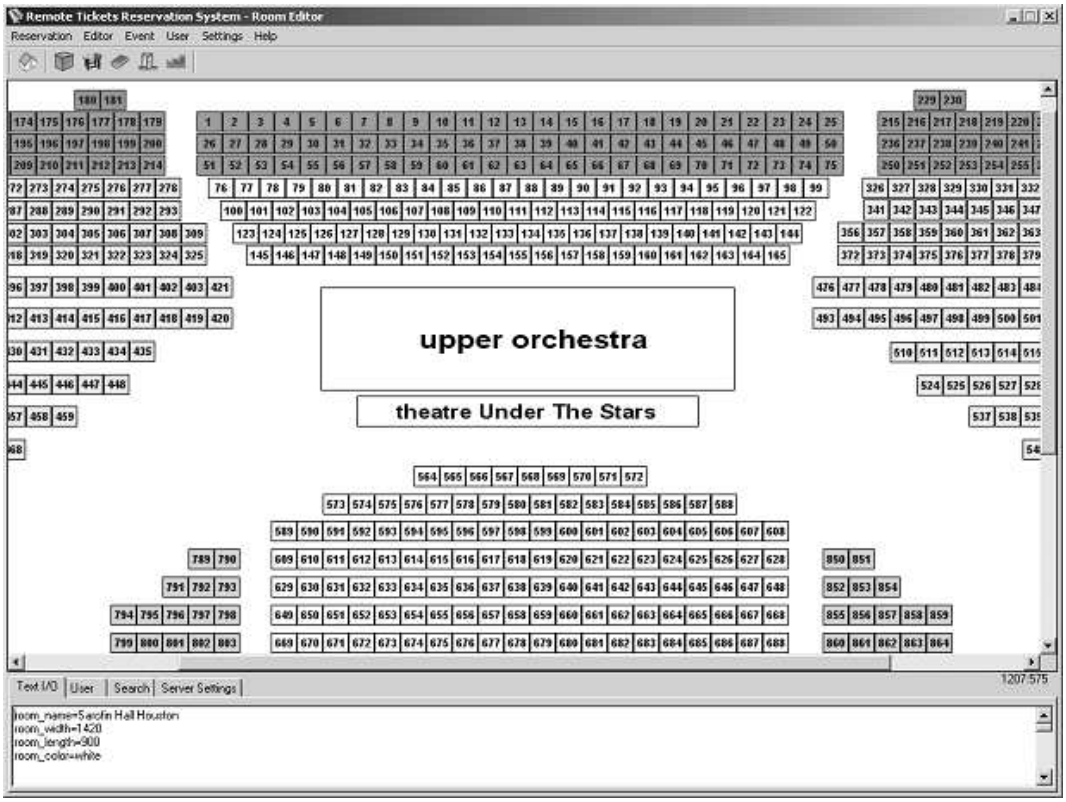
5.2 Modelování prostor

K vytvoření modelu rozložení prostor má uživatel k dispozici rozsáhlý grafický editor. V tomto editoru je možné vytvářet nové a měnit stávající prostory. Prioritou jeho návrhu je přitom možnost vytvářet co nejefektivněji a nejvěrněji model zobrazované reality.

Editor prostor se spouští položkou hlavního menu *Editor→Nová místnost* nebo *Editor→Načíst místnost*. K otevření editoru v síťovém rozhraní jsou nutná práva garantovaná rolí Creater nebo Administrator.

Editace vlastností místnosti a objektů probíhá dvěma způsoby: textově nebo graficky. Způsoby jsou přitom rovnocenné, oba nabízejí nastavení stejné množiny vlastností. Aplikace umožňuje asociovat s místností sedadla, řady, stoly, stěny a polygony. Přesný formát textového i grafického vstupu pro jednotlivé druhy objektů je uveden v nápovědě klientské aplikace a v uživatelské dokumentaci projektu. Editor má také vlastnost „drag-and-drop“, tzn. že objekt může být uchopen pomocí myši a tažen po mapě místnosti, mění se tím jeho logické souřadnice.

Pokud uživatel zvolí uložení modelu (položka hlavního menu *Editor→Uložit místnost*), jsou data podle zvoleného komunikačního rozhraní v textovém formátu odeslána na server nebo uložena do lokálního souboru.



Obrázek 5.1: Hlavní okno aplikace při editaci prostor, v horní části okna je toolbar pro grafické přidávání objektů. Ve spodní části okna je aktivní záložka s textovým výstupem objektů, pomocí níž jde prostor taktéž editovat.

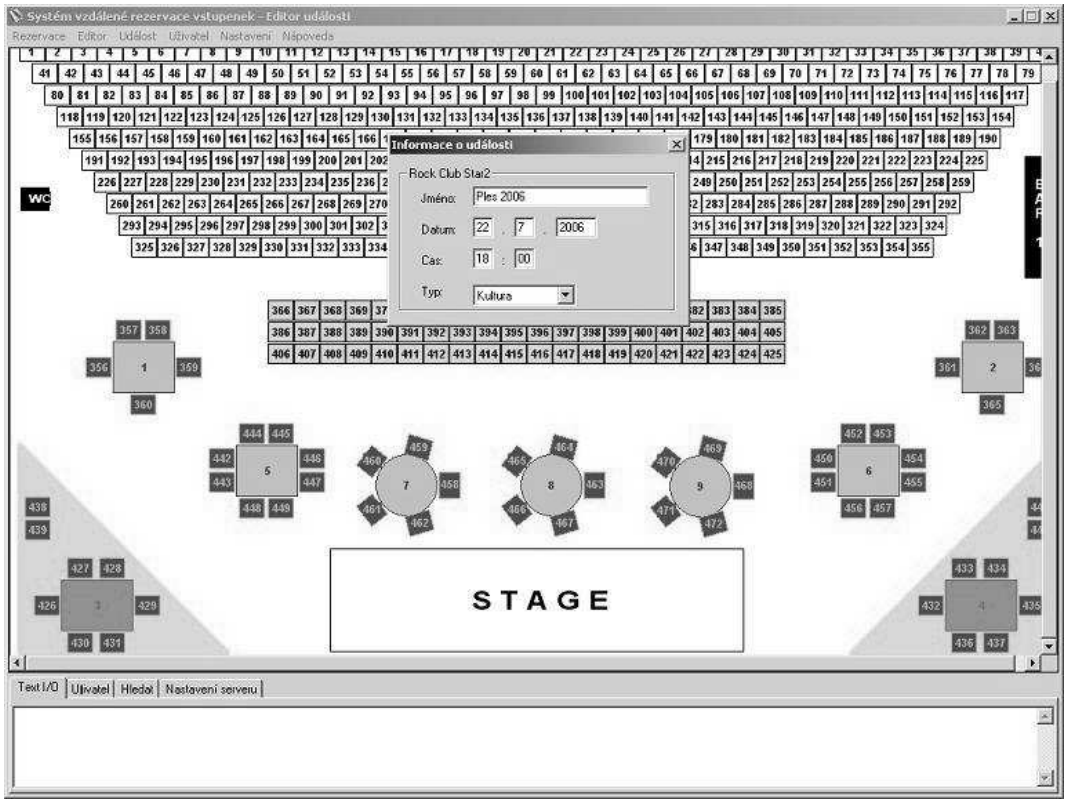
5.3 Modelování událostí

Dalším logickým krokem po namodelování mapy prostor je vytvoření události, to se odehrává v okně Event Editor. K jeho otevření v rozhraní síťové komunikace jsou obdobně jako u editoru místností vyžadována práva garantovaná rolemi Creater nebo Administrator.

Editor události se otevírá položkou hlavního menu *Událost→Vytvoř událost* nebo *Editor→Načti událost* pro editaci dříve definované události.

Uživatel musí při vytváření nové události nejdříve ve speciálním dialogu specifikovat její vlastnosti: název, datum, čas konání a typ, pod kterým se bude událost zobrazovat ve webovém rozhraní. Poté uživatel přiřazuje vlastnosti jednotlivým vstupenkám. Vstupenkám může uživatel přiřazovat cenu, termín uzavření jejího prodeje, úroveň ochrany, komentář a popis, kde je vstupenka k zakoupení. Na počátku jsou všechny vstupenky neinicializované, inicializace probíhá obdobně jako v editoru místností textově nebo graficky. Podrobný popis vytváření vstupenek je popsán v nápovědě klientské aplikace a v uživatelské dokumentaci projektu.

V momentu, kdy uživatel nastaví potřebné vlastnosti a zvolí uložení události (položka hlavního menu *Událost→Uložit událost*), jsou data odeslána s požadavkem na uložení na server nebo uložena přímo do lokálního textového souboru (podle zvoleného rozhraní pro komunikaci).



Obrázek 5.2: Aplikace při grafickém nastavování vlastností události. Model prostor obsahuje jako objekty řady, stoly (obdélníkové i kulaté), stěny s textovým popisem (například STAGE) a polygony.

5.4 Prodej vstupenek

Po nadefinování události může uživatel klientské aplikace využít funkcí pro prodej vstupenek. Toto rozhraní je obdobně jako oba editory implementováno graficky. Uživatel, který pracuje v rozhraní síťové komunikace, se musí před prodejem autentizovat. Jsou vyžadována alespoň práva garantovaná rolí Reserver, ta zajišťuje minimální potřebnou množinu práv pro prodej.

Rozhraní pro prodej vstupenek se spouští výběrem položky hlavního menu *Rezervace→Otevřít*.

Uživatel musí nejdříve specifikovat konkrétní událost z databáze, k dispo-

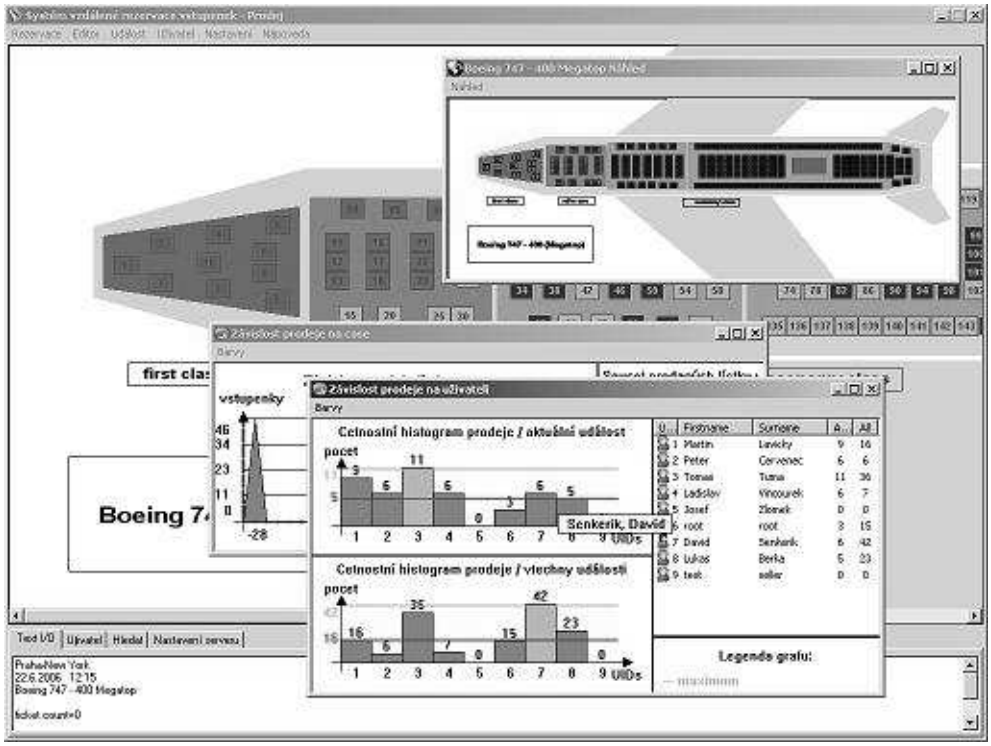
zici jsou mu přitom všechny nadefinované události, jejichž časové razítko je větší než razítko aktuální. Zamezí se tak zbytečnému výpisu starých událostí. Sedadla v okně se graficky zobrazují podle svého stavu, existují přitom tři možné stavy: aktivní, neaktivní (již prodaná) a vybraná sedadla.

Výběr vstupenky, která je zákazníkem rezervována z webového rozhraní, nahlásí varování. Prodejci se zobrazí data o zákazníkovi, který si lístek rezervoval. Podle těchto informací se distributor rozhodne, zda přidá lístek k vybraným či ne. Politika respektování rezervací zákazníků je odsunuta na distributory, ti musí sami podle zobrazených údajů rozhodnout o prodeji rezervované vstupenky jinému zákazníkovi.

Poté, co uživatel označí požadované vstupenky (výběr více vstupenek je realizován pomocí klávesy CTRL), může odeslat požadavek na koupi na server. Požadavek se odesílá položkou hlavního menu *Rezervace→Odeslat*, kontextovým menu při kliknutí mimo objekty místnosti, nebo klávesovou zkratkou CTRL+S.

Protože se předpokládá, že je zákazníkem požadována právě daná množina vstupenek, proběhne úspěšně změna dat buď u všech vstupenek, nebo u žádné. Alternativou by bylo prodat ty vstupenky, které jsou dostupné, u ostatních pak nahlásit chybu. Tato implementace by však mohla způsobovat při prodeji problémy. Představme si situaci, kdy požaduje zákazník jednu ze dvou variant umístění. Pokud distributor zvolí v naší implementaci prodeje první variantu a některá z požadovaných míst jsou již obsazena, dojde k chybě. Distributor se tak může pokusit zakoupit druhou požadovanou množinu vstupenek. Pokud by však požadavky byly uspokojovány po částech, musel by zákazník k dostupným místům z prvního požadavku vybrat zbylé vstupenky odjinud. Tato implementace prodeje by zřejmě nebyla příliš vhodná.

Při každém prodeji se obnovují v klientské aplikaci data o události, uživatel má mimoto možnost získat aktuální stav prodeje a rezervací také výběrem položky hlavního menu *Rezervace→Aktualizovat*.



Obrázek 5.3: Na obrázku je zobrazena možnost využití aplikace pro dopravní (leteckou) společnost. Je zde zachyceno hlavní okno aplikace při prodeji, okna s grafy a okno náhledu na vytvořený model prostor.

5.5 Vyhledávání

Uživatel klientské aplikace má k dispozici funkce pro prohledávání prostor. Pomocí těchto funkcí může automaticky přidělovat vstupenky podle počtu zákazníkem požadovaných míst. Funkce pro vyhledávání jsou k dispozici v záložce *Hledat* hlavního okna aplikace. Záložka je aktivní pouze pokud má uživatel otevřené okno pro prodej.

Vyhledává se pouze mezi volnými sedadly, rezervovaných a prodaných sedadel se vyhledávání netýká. Implementované algoritmy vyhledávání jsou popsány v kapitole 3.3.4, algoritmus vyhledávání je možno vybrat v hlavním menu programu.

Nejprve musí uživatel specifikovat velikost požadavku. Zadává tedy počet vstupenek, které chce zákazník umístit. Poté je k dispozici funkce pro hledání nejlepšího stolu a funkce pro obecné umístění požadavku (tlačítka *Nejlepší stůl* a *Hledat*).

Pokud zvolí uživatel funkci na obecné umístění, mohou nastat dvě možnosti:

První možností je, že požadavek lze umístit k jednomu stolu nebo do jedné

řady. Všechna taková umístění jsou uživateli nabídnuta. Jednotlivé možnosti umístění se zobrazují graficky do modelu, distributor se v nich může pohybovat pomocí tlačítek posuvníků. Výpis alternativ umístění je obsažen také v textovém formátu v pomocném okně.

Druhou možností je, že požadavek nelze umístit k jednomu stolu či do jedné řady, v takovém případě se volá funkce na kombinované umístění požadavku. Uživateli je nabídnuta pouze jedna možnost, posuvníky nejsou aktivní.

Kapitola 6

Závěr

Cílem bakalářského projektu bylo navrhnout a implementovat komplexní systém pro rezervaci a prodej vstupenek na příležitostné kulturně-společenské akce. Požadavkem bylo, aby projekt nezůstal po dokončení bez využití, ale aby mohl být nasazen na prodej vstupenek na plesy Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

Všechny předem splněné cíle práce byly splněny. Analýzou požadavků na vytvářený software byla navržena dvě samostatná rozhraní: pro rezervaci a pro prodej. Ta přistupují do společných datových struktur. Výsledkem práce na bakalářském projektu je návrh struktury databáze a plně funkční implementace obou navržených rozhraní.

V budoucnosti by projekt mohl být dále rozšiřován a zrobustněn, možností jeho dalšího rozšíření je například návrh a implementace abstraktní databázové vrstvy, která by umožňovala využití jiných databázové serverů (například Oracle).

Dalším rozšířením aplikace by mohla být implementace prodejního webového rozhraní, které by podporovalo elektronické platby a předávání vstupenek přes internet.

Literatura

[1] Eckel B.: *Myslíme v jazyku C++*, Grada Publishing, Praha, 2000.

[2] Klíma V.: Základy moderní kryptografie - Symetrická kryptografie III.

[3] Prof. RNDr. Pokorný J., CSc., Ing. Halaška I.: *Databázové systémy*, ČVUT, Praha 1999.

[4] Töpfer P.: *Algoritmy a programovací techniky*, PROMETHEUS, Praha 1995, 21-28.

[5] Mgr. Vondruška P.: *Crypto World 4/2005*.

[6] Mgr. Vrána J.: Ukládání hesel, http://php.vrana.cz/ukladani-hesel.php.

[7] PHP Documentation, http://www.php.net/docs.php.

[8] Smarcoms: Rezervační systém Bizzy, http://www.buybizzy.biz.

[9] Webtigo Web Development: phpMyTicket, http://www.PHPMyTicket.org

[10] MySQL Reference Manual, http://dev.mysql.com.

[11] Wikipedia contributors: SQL Injection. Wikipedia, The Free Encyclopedia. http://en.wikipedia.org, 2006.

Příloha A

Kompaktní disk

K této práci je přiložen disk CD, jehož obsah je následující:

A.1 Zdrojové kódy

Na přiloženém CD jsou v adresáři */src/* uloženy zdrojové kódy jednotlivých komponent systému. Projekt klientské aplikace z Visual Studia 6.0 je uložen v adresáři */src/client*, zdrojové kódy serveru v adresáři */src/server*, zdrojové kódy webového rozhraní v adresáři */src/web* a sql skript na vytvoření databáze v adresáři */src/dbs*. V adresáři */src/help* jsou uloženy zdrojové kódy nápovědy ke klientské aplikaci.

A.2 Instalace klientské aplikace

Instalační program klientské aplikace je k dispozici v adresáři */inst*.

A.3 Dokumentace

V adresáři */doc* je uložena uživatelská a programátorská dokumentace projektu ve formátu PDF. V adresáři */doc/bc* jsou uloženy zdrojové kódy této práce v systému LaTeX, včetně použitých obrázků. Adresář */doc/hlp* obsahuje zkompilovanou HTML nápovědu klientské aplikace ve formátu chm.