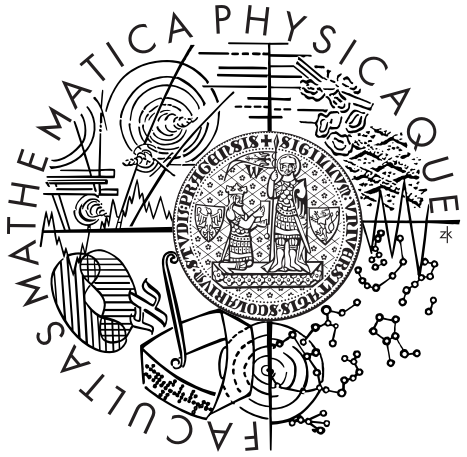
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jakub Zíka

Distribuovaný systém správy objednávek pro restaurační zařízení

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D.
Studijní program: Informatika
Studijní obor: programování

Praha 2014

Poděkování

Děkuji RNDr. Janu Kofroňovi, Ph.D. za vedení této práce, za jeho cenné připomínky a za pomoc s jejím vznikem. Dále děkuji své rodině a svým přátelům za jejich podporu. Nakonec patří můj dík autorům těch softwarových děl, bez jejichž existence by tato práce nemohla vzniknout.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 19. května 2014 .....

Název práce: Distribuovaný systém správy objednávek pro restaurační zařízení

Autor: Jakub Zíka

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D.
Katedra distribuovaných a spolehlivých systémů

Abstrakt: Pro restaurace a podobné podniky je velice výhodné používat software pro správu objednávek kvůli udržení přehledu o financích podniku a požadavcích zákazníků. Na českém trhu již existuje celá řada produktů vytvořených pro tento účel. Existující softwarová řešení je ovšem často složité rozšiřovat nebo propojovat s dalšími systémy. V rámci této práce je navržen a vyvinut systém pro správu objednávek EasyPub fungující na operačních systémech GNU/Linux a Microsoft Windows. EasyPub umožňuje externím programům přistupovat k informacím uloženým v systému prostřednictvím otevřeného protokolu. Vznik systému je přínosný a přispívá k inovaci v oblasti restauračního software.

Klíčová slova: restaurace, objednávky, síť, rozšiřitelnost

Title: Distributed order management for restaurants

Author: Jakub Zíka

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D.
Department of Distributed and Dependable Systems

Abstract: Order management software is vital to restaurants and other order-based businesses to keep track of their finances and the demands of their customers. Many such products are available in the Czech market; however, it is often difficult to modify these existing solutions or interface them with new or different systems. In this thesis, we design and implement a new order management system called EasyPub. EasyPub works on GNU/Linux and Microsoft Windows operating systems, and allows external programs to access information stored within the system through an open communication protocol. Its creation is beneficial and contributes to innovation in the area of restaurant software.

Keywords: restaurants, orders, network, extensibility

Obsah

1. Úvod

Při provozu restaurací, kaváren, čajoven a dalších podobných podniků je nutné mít dokonalý přehled o objednávkách jednotlivých hostů, jinak by obsluha neměla možnost zjistit, které objednávky ještě nebyly vyřízeny, ke kterému stolu daná objednávka patří a co mají naúčtovat zákazníkovi, který se rozhodne zaplatit.

Je samozřejmě možné udržovat informace o objednávkách za pomoci tužky a papíru. Tento přístup má ovšem velké množství na první pohled zřejmých nevýhod. Patří mezi ně větší nároky na personál, který musí udržovat tyto informace aktuální, složitější vyhledávání informací, nebezpečí numerických chyb a obecně nebezpečí chyb způsobených nepozorností. Navíc je takový přístup prakticky nepoužitelný pro větší podniky s velkým počtem zákazníků.

Jako vhodné řešení všech těchto nedostatků se nabízí využít služeb počítačového programu, který by zjednodušil práci personálu a zabránil lidským chybám. Kromě toho by mohl poskytnout i další funkcionalitu, jako například správu databáze nabízeného zboží, sledování popularity jednotlivých druhů zboží, automatické generování a tisk účtenek pro zákazníky, zlepšení přehledu o finanční situaci podniku, správu databáze zákazníků a další. Množství takových systémů již existuje a v běžném provozu jsou čím dál tím rozšířenější.

1.1 Cíle práce

Cílem této práce je shromáždit informace o existujících systémech, získat požadavky na systém tohoto typu, provést analýzu a specifikovat a implementovat systém (pracovně označený EasyPub), který by potenciálně mohl být konkurenceschopný existujícím systémům a v ideálním případě by poskytl některá inovativní vylepšení.

EasyPub je vytvářen na zakázku podle potřeb jednoho konkrétního podniku, ovšem jedním z cílů je, aby po případných drobných úpravách byl použitelný i pro další zákazníky. Proto je v průběhu analýzy kladen důraz na dostatečnou obecnost systému a jeho případnou snadnou budoucí rozšiřitelnost.

Systém je primárně určen k používaní v České republice, tedy uživatelské rozhraní bude v českém jazyce a určité implementační detaily budou předpokládat práci s českými korunami. Tyto aspekty mohou být změněny v rámci budoucí práce, ovšem tato práce si to neklade za svůj cíl.

Vzhledem k tomu, že ztráta dat by v případě systému tohoto typu mohla mít katastrofální následky, je první prioritou implementace zabránit jakékoliv ztrátě. Uživatelská data musí zůstat v konzistentním stavu i v případě nestandardního ukončení aplikace nebo nečekaného vypnutí počítače, například z důvodu přerušení dodávky elektrického proudu.

V následujících kapitolách je možné nalézt analýzu zadání obsahující uživatelské požadavky a informace o podobných existujících systémech, návrh systému čerpající z poznatků analýzy, uživatelskou dokumentaci popisující práci s programem, programátorskou dokumentaci sloužící k potřebám vývojářů, kteří by v budoucnu program upravovali nebo rozšiřovali, a závěr shrnující vlastnosti vzniklé implementace a celkový přínos práce z pohledu autora.

2. Specifikace práce

V této kapitole jsou shrnuty okolnosti vzniku systému EasyPub. Jsou rozděleny na dvě hlavní části: požadavky zákazníka a jejich posouzení s přihlédnutím k již existujícím systémům s podobnou funkcionalitou. Na konci kapitoly je krátce shrnuto, čím bude nově vytvářený systém inovativní a jaké bude mít unikátní vlastnosti.

2.1 Požadavky zákazníka

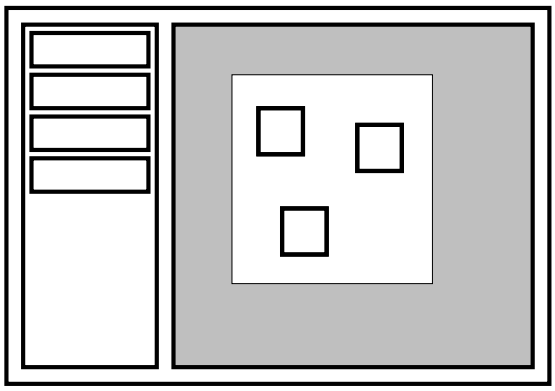
V prvních fázích vývoje systému byla se zákazníkem dohodnuta základní požadovaná funkcionalita systému a bylo hrubě načrtnuto uživatelské rozhraní. Tyto požadavky jsou popsány zde.

2.1.1 Prostředí aplikace

Dílo bude primárně vytvářeno pro běh v Unixovém prostředí, avšak, pokud se v průběhu implementaci nevyskytnou nečekané překážky, bude dílo tvořeno takovým způsobem, aby podporovalo co nejvíce různých platforem, především operační systém Microsoft Windows, který je mezi potenciálními uživateli asi nejrozšířenější.

2.1.2 Uživatelské rozhraní

Hlavní obrazovka

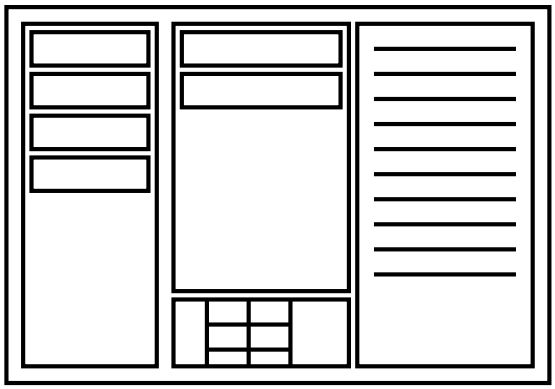


V levé části je seznam všech objednávek z aktuálního dne. U každé objednávky je uvedeno, co a kdy bylo objednáno, cena a aktuální stav. Při kliknutí na tlačítko objednávky je možné měnit stav objednávek (popsáno dále v sekci požadavků na funkci).

V pravé části se nachází tlačítka znázorňující jednotlivé stoly v podniku. Stoly, u kterých jsou nezaplacené nebo nezpracované objednávky budou barevně zvýrazněné. Po kliknutí na tlačítko stolu bude hlavní obrazovka nahrazena obrazovkou detailu tohoto stolu.

V horní části obrazovky se dále bude nacházet aplikační menu, jehož prostřednictvím bude mít uživatel přístup k méně často používaným funkcím programu, jako je přihlášení, změna stavu hotovosti v kase a další.

Obrazovka detailu stolu



V levé části je seznam všech objednávek z aktuálního dne. Tento seznam je vzhledem i chováním identický seznamu v levé části hlavní obrazovky popsané dříve.

V prostřední části je seznam nezaplacených objednávek u stolu, jehož detail je zobrazován, který kliknutím umožňuje vybrat některé nebo všechny objednávky. Ve spodní části seznamu se nachází tlačítka usnadňující práci s výběrem objednávek („Vybrat vše“, „Nevybrat nic“, „Invertovat výběr“) umožňující platbu nebo zrušení objednávek („Zrušit vybrané“, „Zaplatit vybrané“, „Zaplatit vše“), umožňující přesun objednávek mezi stoly („Přesunout vybrané“) a tlačítko „Zpět“ pro zavření obrazovky detailu stolu a zobrazení hlavní obrazovky.

Po kliknutí na jedno z tlačítek určených ke zrušení nebo zaplacení objednávky dojde k odpovídající akci popsané dále.

V pravé části obrazovky je seznam aktuálního sortimentu. Kliknutím na položku sortimentu dojde k vytvoření objednávky této položky u daného stolu. Ve spodní části seznamu se nachází textové pole, které je možné využít pro filtrování a vyhledávání seznamu položek sortimentu.

Tato obrazovka bude v horní části obsahovat stejné aplikační menu jako hlavní obrazovka.

Dialog vytvoření objednávky

Tento dialog bude obsahovat alespoň následující prvky:

- Informace o názvu a kusové ceně
- Editovatelné pole pro nastavení počtu kusů
- Tlačítka pro potvrzení a zrušení akce

Dialog vytvoření platby

Tento dialog bude obsahovat alespoň následující prvky:

- Informace o počtu kusů a celkové ceně
- Tlačítko pro tisk účetního dokladu
- Tlačítka pro potvrzení a zrušení akce

Dialog změny hotovosti

Tento dialog bude obsahovat alespoň následující prvky:

- Informace o současném stavu kasy
- Editovatelné pole pro zadání vložené nebo vybrané částky
- Tlačítka pro potvrzení a zrušení akce

2.1.3 Definice pojmů

Objednávka je označována jako *aktivní*, pokud byla vytvořena a zatím nebyla ani zaplacena ani zrušena.

Den je pro potřeby systému definován jako období, na jehož začátku ani konci neexistují žádné aktivní objednávky. Zpravidla zhruba odpovídá kalendářnímu dni, ale není to nutné pravidlo (má-li například podnik zavírací dobu až po půlnoci, může jeden den zasahovat do více kalendářních dnů).

2.1.4 Zahájení a ukončení dne

V jeden okamžik může být aktivní nejvýše jeden den, tedy zahájení dne je možné pouze v případě, že jiný den není aktivní.

Ukončení dne je možné pouze v případě, že všechny objednávky byly buď zaplaceny nebo zrušeny. Na konci dne program vypíše aktuální stav kasy, aby mohl být tento očekávaný stav porovnán se skutečným stavem.

Není-li žádný den aktivní, není možné vytvářet nové objednávky.

2.1.5 Přihlášení obsluhy

Systém bude možné nakonfigurovat tak, aby některé akce mohli provádět pouze oprávnění uživatelé.

Přihlášení obsluhy je řešeno zadáním uživatelského jména a hesla. Jméno aktuálně přihlášeného uživatele je uvedeno na hlavní obrazovce programu.

2.1.6 Vyhledávání položek sortimentu

Na obrazovce detailu stolu je seznam všech položek sortimentu. Protože těchto položek může být velké množství, pro efektivní práci s programem je nutné mít možnost mezi nimi rychle vyhledávat.

K tomu bude sloužit textové pole umístěné pod seznamem. Bude-li do něj zadán text, budou ze seznamu odstraněny všechny položky, které tento text neobsahují ve svém názvu. Navíc bude učiněna snaha seřadit položky podle relevantnosti k vyhledávanému výrazu a jejich popularity. Bude-li ve vyhledávacím poli stisknuta klávesa Enter, bude se tato akce považovat za ekvivalentní kliknutí na první zobrazenou položku, umožňující její rychlé objednání pouze za použití klávesnice.

2.1.7 Správa objednávek

Vytvoření objednávky

Objednávku je možné vytvořit pouze je-li aktivní den.

Proces vytvoření objednávky je následující:

- Na hlavní obrazovce se vybere stůl, k němuž by měla být objednávka vytvořena. Zobrazí se obrazovka detailu stolu.
- Na obrazovce detailu stolu se v části s dostupným sortimentem vybere ta položka, která je objednávána. Zobrazí se dialog vytvoření objednávky.
- Na dialogu vytvoření objednávky se zadá požadované množství a zkontroluje se cena. Dialog se potvrdí a operace vytvoření objednávky je hotova.

Změna stavu objednávky

Každá z objednávek může být v jednom z následujících stavů:

- nezpracovaná a nezaplacená
- zpracovaná a nezaplacená
- zaplacená
- zrušená

Je-li objednávka ve stavu zaplacená nebo zrušená, není možné měnit její stav.

Je-li objednávka v jednom z nezaplacených stavů, je možné volně měnit její stav mezi nezpracovaným a zpracovaným (změna ze zpracovaného na nezpracovaný je pouze pro opravení chybného označení objednávky jako zpracované) nebo objednávku zrušit. Tyto změny je možné provést kliknutím na tlačítko objednávky v seznamu všech objednávek a v zobrazeném kontextovém menu vybrat odpovídající možnost.

Změnu stavu na zaplacený není možné provést přímo, pouze prostřednictvím vytvoření platby zahrnující tuto objednávku.

Přesun objednávek mezi stoly

Na obrazovce detailu stolu je možné vybrat některé objednávky, které jsou u daného stolu objednané a zatím nezaplacené, a přesunout je k jinému stolu. Bude k tomu sloužit tlačítko „Přesunout vybrané“, po jehož stisku se zobrazí dialog umožňující výběr stolu, k němuž mají být objednávky přesunuty.

Tato funkcionalita je určena k opravení chyby obsluhy programu nebo ošetření situace, kdy se zákazník přesune k jinému stolu, než kde byl původně.

Zaplacení objednávek

Platbu je možné vytvořit na obrazovce detailu stolu dvěma způsoby: buď vybráním alespoň jedné objednávky a stiskem tlačítka „Zaplatit vybrané“ nebo přímo stiskem tlačítka „Zaplatit vše“.

Kromě vzniku nové platby tento proces nastaví všem objednávkám přiřazeným do platby status „zaplaceno“ a zvýší aktuální stav hotovosti v kase o zaplacenou částku.

2.1.8 Tisk účetního dokladu

Při vytváření platby dojde k tisku účetního dokladu pro zákazníka. Tento doklad bude obsahovat alespoň následující údaje:

- Název restauračního zařízení
- Další informace o zařízení (adresa, identifikační číslo a tak dále)
- Aktuální datum a čas
- Číslo účetního dokladu
- Seznam koupeného zboží obsahující názvy, počty a kusové a celkové ceny
- Celkovou cenu za všechno účtované zboží

2.1.9 Manipulace s hotovostí v kase

Do kasy je možné vkládat hotovost nebo ji vybírat. Změna hotovosti se provede výběrem požadované operace v aplikačním menu a zadáním velikosti vybrané nebo vložené částky do dialogu změny hotovosti, který se zobrazí.

Historie hodnot hotovosti uložených v kase se ukládá spolu s informací o tom, který uživatel byl přihlášený v okamžiku změn hotovosti.

2.1.10 Editování sortimentu

Provozovatel restauračního zařízení a další uživatelé s dostatečným oprávněním budou mít možnost měnit názvy, ceny a další vlastnosti jednotlivých položek sortimentu. K tomu bude sloužit speciální uživatelské rozhraní určené pro správce systému.

2.1.11 Rozšiřitelnost a začlenitelnost systému

Zákazník vyslovil přání mít možnost systém propojit s dalším softwarem, který bude používat, především možnost integrovat data systému do webových stránek podniku. Toto je hlavní funkcionalita, která mu chyběla u obdobných dostupných systémů, a jeden z hlavních důvodů vzniku nového systému. Důležitým úkolem při návrhu systému proto bude vytvoření softwarového rozhraní vhodného pro tento účel.

2.2 Podobné existující systémy

Jak bylo zmíněno v úvodu, systémů pro restaurace existuje celá řada. V rámci práce bylo provedeno podrobné porovnání čtrnácti nejpopulárnějších systémů dostupných pro český trh. Jeho výsledky jsou k dispozici v příloze A na straně 42 (do této přílohy byly zpětně přidány i informace o systému EasyPub implementovaném v rámci této práce).

Z průzkumu vyplynulo, že některé funkce jsou implementovány prakticky ve všech systémech (základní funkcionalita, podpora dotykové obrazovky) a že množiny funkcí podporovaných všemi pokročilejšími systémy jsou prakticky totožné. Tyto systémy se od sebe liší především svým uživatelským rozhraním, cenou a kompatibilitou s dalším softwarem a hardwarem.

S výjimkou ceny je tyto odlišnosti obtížné porovnat. Preference grafického rozhraní je subjektivní a názor se snadno může změnit v závislosti na zkušenostech a znalostech uživatele. Význam kompatibility s externím softwarem nebo hardwarem závisí především na tom, v jakém prostředí plánuje uživatel systém používat, případně který další software nebo hardware již používá nebo má k dispozici. Například některé systémy podporují mobilního číšníka v podobě hardwarového produktu Orderman, zatímco jiné poskytují aplikace pro tablety nebo mobilní telefony. Uživatelé uvažující o nákupu systému potom musí vzít v úvahu, které zařízení pro ně bude výhodnější pořídit a spravovat.

Prodejci systémů podporujících další hardware nebo software potom také často nabízejí zvýhodněné balení obsahující kromě samotného systému i tyto externí součásti.

Některé systémy jsou součástí většího celku, který kromě modulu pro restaurace obsahuje i další moduly, například pro hotel (relativně častá varianta) nebo i jiné. Někdy je správa skladu tvořena samostatným modulem, který není přímou částí restauračního modulu. Relativně častá je také podpora nějakého druhu komunikace s účetním softwarem.

Funkcionalita chybějící prakticky ve všech posuzovaných systémech je možnost programátorského přístupu k datům uloženým v systému či dokonce jejich modifikace. Pravděpodobným důvodem toho je komerční podstata těchto systémů.

Vytvoření systému s otevřeným komunikačním protokolem by umožnilo například vytvořit mobilní aplikaci, modul pro zpracování statistik o provozu restaurace nebo modul pro správu skladu, případně integrovat objednávkový systém do webových stránek podniku. Takový systém by měl zjednodušit experimentování s vývojem modulů, což by mohlo mít za následek vznik zajímavých a inovativních rozšíření systému.

3. Analýza

3.1 Rozdělení na vrstvy

Součástí zadání je přání zákazníka, aby bylo možné systém propojit s externími programy. Systém tedy bude muset takovýmto potenciálním programům poskytnout cestu, kterou s ním budou moci komunikovat. Ze všech možných druhů meziprocesové komunikace byl výběr omezen na ty používající síťové spojení. Prostřednictvím síťového spojení spolu mohou totiž komunikovat i procesy běžící na různých strojích, což umožní například implementaci mobilních aplikací komunikujících se systémem.

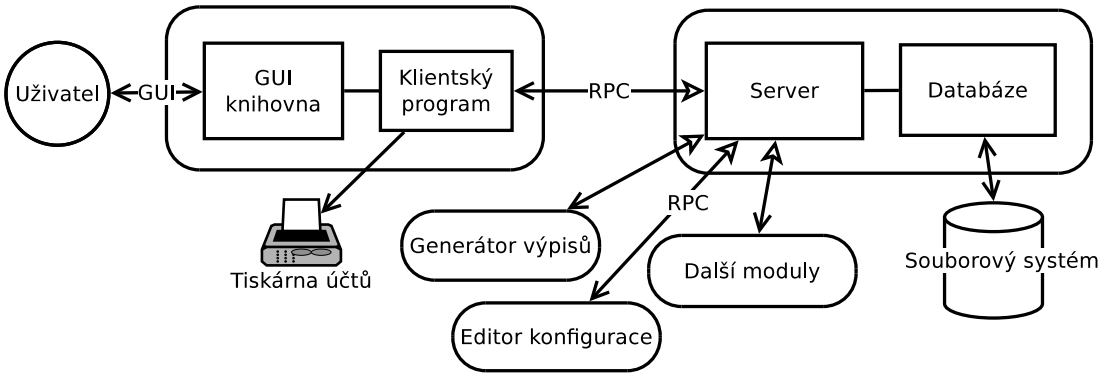
Systém tedy bude obsahovat síťový server, ke kterému se budou moci připojovat klientské programy a který bude používat protokol umožňující získávat a modifikovat data v systému uložená. Vzhledem k tomu, že cílem je navrhnout tento protokol tak, aby byl co nejúplnější (aby bylo možné vytvořit plnohodnotný klientský program), dává smysl, aby i uživatelské rozhraní vytvořené v rámci této práce používalo stejný protokol. Tím bude zajištěno, že externí klientský program bude mít možnost provádět stejné operace, které bude moci provádět uživatel z grafického rozhraní.

Proto se systém bude skládat z alespoň dvou programů, které spolu budou komunikovat po síti. Jeden z nich bude server, který se bude starat o uchování a správu dat systému a poskytnutí softwarového rozhraní klientským programům. Klientské programy potom budou toto rozhraní využívat pro získání dat, která budou zobrazovat uživatelům, a pro jejich případnou modifikaci, pokud si to uživatel bude přát.

Mezi výhody tohoto rozdělení patří snížení komplexnosti jednotlivých vrstev, vznik explicitního rozhraní mezi vrstvami (to zlepší přehlednost kódu), a částečné omezení důsledků programátorských chyb, které se v určitých případech nebudou moci propagovat za hranici vrstvy, kde vznikly, protože ostatní vrstvy budou samy ověřovat smysluplnost předaných dat.

Použití síťového spojení navíc umožňuje měnit implementaci jednotlivých částí bez nutnosti upravovat ostatní části nebo přerušovat jejich činnost.

Podle závěrů předcházejících úvah je tedy možné nastínit přibližnou architekturu systému. Diagram tohoto návrhu je k dispozici na obrázku 3.1.



Obrázek 3.1: Diagram návrhu systému

Klientské programy v tomto modelu tvoří převážně prezentační vrstvu systému a server poskytuje aplikační a datovou vrstvu. Datová vrstva je při tom v serveru tvořena převážně databází a jí používaným souborovým systémem, tedy server sám o sobě implementuje pouze aplikační vrstvu.

3.2 Použité technologie

Nyní se podíváme na konkrétní technologie používané systémem. U každé bude popsán její účel, budou zmíněny případné alternativy s obdobnou funkcionalitou a bude zdůvodněno, proč byla vybrána tato konkrétní technologie místo dostupných alternativ.

3.2.1 Programovací jazyk

Jedním z cílů je vytvořit systém, který zabrání ztrátě a korupci dat. K tomu je vhodné použít jazyk se statickými typy podporující objektově orientované programování, neboť tyto jazyky jsou většinou kompilované a jejich kompilátory umožňují odhalit chyby v programu ještě před jeho vlastním spuštěním. Zároveň je žádoucí použít dostatečně populární jazyk, pro který jsou k dispozici různé nástroje usnadňující práci s ním.

S ohledem na tato kritéria byli předběžně vybráni tři kandidáti: C++, C# a Java. C# byl poměrně brzy zavržen z důvodu obtížné přenositelnosti mezi operačními systémy, protože, ačkoliv podpora na Unixových systémech existuje, často pokulhává za oficiální implementací pro Windows. Rozhodování mezi C++ a Javou bylo o něco složitější, ale nakonec bylo rozhodnuto ve prospěch C++. Důvody pro toto rozhodnutí byla možnost nativní kompilace, která zjednodušuje distribuci výsledného programu, a osobní preference. Argumenty ve prospěch Javy byly především možnost vytvoření univerzálního balíku pro všechny platformy a spravovaná paměť, která by mohla pomoci při hledání a opravování chyb v programu.

3.2.2 RPC framework

Pro umožnění přístupu externích programů do systému bude použit RPC[1] framework komunikující přes síťové spojení. Od tohoto frameworku byla vyžadována podpora pro C++ i další jazyky (pro umožnění vytvoření klientských programů v jiných jazycích), možnost specifikovat statické datové typy v rozhraní serveru, možnost provést typovou kontrolu při kompilaci programu a schopnost obstarání síťového spojení mezi klientem a serverem.

Mezi možné kandidáty byly zahrnuty knihovny Apache Thrift, Protocol Buffers, různé implementace standardů JSON-RPC, SOAP[2] a další, ale nakonec byl vybrán framework Apache Thrift nejlépe splňující zadaná kritéria.

Při použití tohoto frameworku vývojář nejdříve napíše specifikaci síťové služby se speciální syntaxí podporovanou Thriftem a potom použije kompilátor Thriftu

[1] *RPC* – Remote procedure call. Způsob meziprocesové komunikace, kde volané metody se mohou vykonávat v jiném než volajícím procesu.

[2] *SOAP* – Simple Object Access Protocol. Protokol meziplatformní výměny dat založený na standardu XML.

pro vygenerování zdrojových kódů v cílovém jazyce implementujících specifikované rozhraní. Tyto zdrojové kódy je potom možné používat z ostatních částí programu a jsou překládány spolu se zbytkem programu. Thrift podporuje velké množství různých programovacích jazyků a poskytuje abstrakci nad síťovým spojením mezi klientem a serverem. (Slee et al. [3])

Jednou z nevýhod Thriftu odhalených v průběhu práce je obtížnost kompilace frameworku pomocí GCC[3] pro Windows, kde bylo potřeba provést drobné úpravy jeho zdrojového kódu. Po provedení úprav ovšem spolehlivě fungoval i v prostředí Windows. Změny, které bylo nutné provést, jsou k dispozici v elektronické příloze práce.

3.2.3 Databáze

Server potřebuje spolehlivý způsob uchování dat, který mu umožní zajistit jejich konzistenci a zabránit jejich ztrátě a poškození. K tomuto účelu je vhodné použít SQL[4] databázi s podporou transakcí a s dalšími zárukami, které by bylo velice obtížné dostat při přímé práci se souborovým systémem.

Ačkoliv byly brány v úvahu plnohodnotné SQL databáze jako například PostgreSQL, MySQL a další, nakonec byla místo nich vybrána knihovna SQLite3, která má ke správě databáze diametrálně odlišný přístup. Hlavní rozdíl spočívá v tom, že SQLite nevyužívá pro správu databáze samostatný proces a všechny přístupy k databázovému souboru na disku jsou prováděny přímo z procesu volajícího funkce knihovny SQLite.

Tento přístup má své výhody, ale také některé nedostatky. Mezi výhody patří větší kontrola programu nad samotnou databází, zbavení se nutnosti existence externího procesu a komunikace s ním a fakt, že uživatel se nemusí zabývat konfigurací databáze, protože program to může automaticky udělat za něj. Mezi nevýhody na druhou stranu patří horší výkon databáze, neboť veškerou údržbu je nutné provádět v rámci volání knihovních funkcí, což může tato volání pozdržet, zatímco v tradiční databázi by se tato údržba mohla provádět například v samostatném vlákně, i když zrovna k databázi není jinak přistupováno.

SQLite3 podporuje plnohodnotné ACID[5] transakce, indexování dat v tabulkách a většinu dalších funkcí specifikovaných ve standardu SQL-92.[6] Množství dat uložených v databázi s velkou pravděpodobností nebude tak velké, aby s ním SQLite měla jakékoliv problémy. Knihovna podporuje veškerou funkcionalitu, kterou systém bude potřebovat, a proto nemá smysl místo ní používat komplexnější databázové systémy.

3.2.4 Grafické rozhraní

Od knihovny grafického rozhraní byla především požadována přenositelnost na různé platformy. Z knihoven dostupných pro C++ byli vybráni tři hlavní kandidáti: Qt, GTK+ a wxWidgets. Nakonec byla zvolena knihovna wxWidgets díky

[3] *GCC* – GNU Compiler Collection. Kolekce softwarových nástrojů umožňující mimo jiné kompilaci C a C++ programů.

[4] *SQL* – Structured Query Language. Jazyk určený pro práci s daty v relačních databázích.

[5] *ACID* – Atomicity, Consistency, Isolation, Durability. Vlastnosti databázových transakcí zaručující jejich spolehlivost.

[6] `http://sqlite.org/about.html`

své kompatibilitě s mnoha různými systémy a možnosti použití nativních ovládacích prvků platformy, na které běží.[7] Za výhodu wxWidgets je také možné považovat fakt, že zdrojové kódy se píší v C++ a není potřeba zdrojové kódy nejdříve zpracovávat speciálním preprocesorem, jak tomu je například u Qt.

Na druhou stranu knihovna wxWidgets má i mnohé negativní vlastnosti. Hlavní z nich je pravděpodobně nestandardní správa paměti při používání grafických prvků. Knihovna totiž ve většině případů vyžaduje, aby tyto objekty byly dynamicky alokovány na haldě, ale sama se stará o jejich destrukci. To vyžaduje používání ukazatelů a dynamických alokací v situacích, kde by byly jinak zbytečné nebo by šly nahradit nějakým typem chytrých ukazatelů, které by byly v moderním kódu preferované. Na knihovně se navíc projevuje její stáří, protože na mnoha místech příliš nespolupracuje se standardní knihovnou a často duplikuje její funkcionalitu.

Při zpětném pohledu je potřeba říci, že ačkoliv wxWidgets při vývoji programu netvořily výraznou překážku, použití modernějšího frameworku by pravděpodobně umožnilo psaní elegantnějšího kódu využívajícího nejnovější standardy a doporučení.

3.2.5 Další softwarové knihovny

Při psaní kódu pro tisk účtenek bylo zjištěno, že ačkoliv knihovna wxWidgets obsahuje různé funkce pro grafickou manipulaci s obrázky, které by bylo možné využít pro vykreslování účetních dokladů, mají tyto funkce neočekávané problémy na systémech s malým množstvím volné paměti. Proto bylo rozhodnuto, že pro tuto úlohu bude použita jiná knihovna specializovaná na daný úkol. Pro tento účel byla vybrána knihovna Cairo umožňující práci s 2D grafikou a textem, která splňovala všechny na ni kladené požadavky.

3.3 Zpracování uživatelského požadavku

V této části je popsáno, jak budou zpracovávány uživatelské požadavky za použití vrstevnatého modelu popsaného na začátku kapitoly a jednotlivých vybraných technologií. Postup zpracování požadavků je následující:

- Interakce uživatele s grafickým rozhraním vyvolá událost operačního systému.
- Knihovna wxWidgets zpracuje událost a vyvolá příslušný obslužný kód, který je součástí klientského programu.
- Podle typu události kód programu může zavolat některou metodu na proxy objektu vygenerovaném Thriftem pro komunikaci se serverem.
- Framework Apache Thrift zajistí přenesení informací o provedeném volání na server.
- Na serveru Thrift přijme a dekóduje tyto informace a zavolá obslužnou metodu v kódu serveru.

[7] `http://www.wxwidgets.org/`

- Kód serveru vyhodnotí toto volání. Je-li k tomu potřeba číst nebo upravit nějaká uložená data, využije k tomu knihovnu SQLite.
- Knihovna SQLite přečte nebo zapíše data v souborovém systému podle požadavků serveru.

Po zpracování požadavku serverem putují informace o jeho vyhodnocení zpět těmito vrstvami až k uživateli, kterému je zobrazen výsledek prostřednictvím grafického rozhraní.

V některých případech nemusí požadavek projít všemi popsanými vrstvami. Například, má-li klientský program uložené informace, na které se uživatel ptá, není potřeba provádět vzdálené volání na server a všechny vrstvy po něm následující. Naopak v některých případech může být požadavek na server vyvolán automaticky bez zásahu uživatele. V tom případě jsou zase přeskočeny vrstvy zajišťující interakci s uživatelem.

3.4 Komunikační protokoly a formáty

Protokol síťové komunikace je ponechán jako odpovědnost frameworku Thrift. Bližší diskuze je k dispozici v programátorské dokumentaci na straně 15.

4. Programátorská dokumentace

4.1 Organizace kódu

Kód systému je rozdělen do několika adresářů: `admin`, `client`, `common`, `gui`, `model`, `server`, `thrift` a `user`. Toto rozdělení odpovídá rozdělení kódu do jmenných prostorů jazyka C++. Složka `thrift` je jediná, ve které se nenachází zdrojový kód v jazyce C++. Místo toho jsou v ní soubory, které kompilátor frameworku Apache Thrift používá k vygenerování zdrojových kódů implementujících protokol síťové komunikace.

Systém EasyPub obsahuje tři spustitelné programy: server, uživatelské rozhraní a administrační rozhraní. Pro každý z nich existuje složka obsahující zdrojové kódy specifické pro daný program a implementaci jeho funkce `main`. Pro server je to složka `server`, pro uživatelské rozhraní složka `user` a pro administrační rozhraní složka `admin`. Kód ve složkách `client` a `gui` je používán uživatelským a administračním rozhraním. Kód v ostatních složkách (`common`, `model` a `thrift`) je používán ve všech třech programech.

Většina tříd jazyka C++ je deklarována v souboru s příponou `.h`, jehož jméno je shodné se jménem třídy. Implementace takových tříd se nachází ve stejnojmenném souboru s příponou `.cpp`. Výjimkou jsou soubory, které obsahují pouze deklarace nebo implementace funkcí, a někdy třídy, které jsou velice jednoduché a jsou si navzájem podobné, takže jsou umístěny společně do jednoho souboru.

4.2 Síťový model

Systém byl navržen podle síťového modelu klient-server. Při použití systému bude v provozu právě jeden server, k němuž se budou moci připojovat jednotliví klienti. Server sám je pasivní, tedy pouze odpovídá na požadavky klientů a sám nezahajuje žádnou komunikaci. Aby si klienti mohli předávat informace mezi sebou, musí být připojeni ke stejnému serveru předávat si informace jeho prostřednictvím. V rámci systému je tedy možné použít pouze jednu implementaci serveru. Naopak implementací klientů může být více, pokud všechny dodržují předepsaný protokol.

Jednotliví klienti se mohou podstatně lišit v mnoha aspektech svého fungování a očekává se, že většina klientů bude využívat pouze podmnožinu funkcí definovaných rozhraním serveru. V rámci práce byl vytvořen klient určený ke každodennímu použití personálem restauračního zařízení a klient určený k administraci systému. Průnik funkcí použitých těmito dvěma klienty je malý a tvořený především funkcemi sloužícími k přihlášení uživatele a k získání informací o objektech uložených v systému.

4.3 Protokol síťové komunikace

Síťovou komunikaci zprostředkovává framework Apache Thrift. Tento framework sestává z generátoru kódu pro vývojářem specifikované rozhraní a knihovny umožňující vygenerovaný kód použít k síťové komunikaci.

Apache Thrift podporuje několik formátů serializace dat. Patří mezi ně například JSON[1] a různé formáty specifické pro Thrift, především Binary a Compact. EasyPub používá Binary formát, který je z hlediska množství přenášených dat úspornější než JSON a je implementovaný ve všech jazycích podporovaných Thriftem. Compact formát by s velkou pravděpodobností poskytl ještě větší úspory, ale v současné verzi jej podporuje pouze malé množství vybraných jazyků, což by mohlo ztížit budoucí rozšiřování systému. Pokud bude v budoucnosti v Thriftu implementována podpora pro Compact formát i pro ostatní jazyky, může to být důvod ke změně tohoto rozhodnutí.

Je-li klient připojen k serveru, Apache Thrift poskytuje objekt implementující specifikované rozhraní serveru sloužící ke vzdálenému volání funkcí. Pokud jsou volány funkce na tomto objektu, Thrift zajistí serializaci předaných parametrů, předá je spolu se jménem volané funkce serveru pomocí síťového TCP/IP[2] spojení, na serveru dekóduje serializovaná data, vyvolá kód obsluhující vybranou funkci a výsledek výpočtu obdobným procesem předá prostřednictvím síťového spojení zpět klientovi, který dekódovaná data vrátí volajícímu procesu. Umožňuje tak relativně jednoduchou implementaci vzdálených volání se zachováním iluze lokálního volání. Abstrakce samozřejmě z technických důvodů není zcela dokonalá, hlavními rozdíly oproti lokálnímu volání jsou vyšší a nepředvídatelná latence a možnost neúspěchu volání z důvodu přerušení spojení.

Ke specifikaci rozhraní serveru slouží soubory s příponou `.thrift` se syntaxí specifickou pro Apache Thrift. Tyto soubory jsou určeny ke zpracování generátorem kódu z frameworku Apache Thrift, který vygeneruje zdrojové kódy v cílovém jazyce obsahující specifikované třídy a rozhraní. Tyto vygenerované soubory je potom možné přeložit spolu se zbytkem zdrojových kódů aplikace.

V případě systému EasyPub jsou zdrojové kódy pro Apache Thrift uloženy v podadresáři `src/thrift`. Jde o dva soubory, `EasyPub.thrift` a `Types.thrift`. V souboru `EasyPub.thrift` je definované rozhraní serveru ve formě funkcí, jejich argumentů, návratových typů a výjimek, ke kterým může dojít. V souboru `Types.thrift` jsou potom definované jednotlivé datové struktury použité jako argumenty nebo návratové hodnoty volání z prvního souboru.

V následujících sekcích je uvedeno stručné shrnutí významu a obsahu jednotlivých struktur a funkcí definovaných ve dříve zmíněných souborech. Přesnější specifikace je k dispozici ve dříve zmíněných souborech.

4.3.1 Datové struktury

- `User`

  Obsahuje informace o uživateli, například jeho ID a přihlašovací jméno.

- `Role`

  Obsahuje informace o uživatelské roli, jako je její ID a její název. Každý uživatel je asociován s množinou rolí, z nichž každá mu dává určitá oprávnění.

[1] *JSON* – JavaScript Object Notation. Datový formát vhodný pro meziplatformní sdílení dat.

[2] *TCP/IP* – Transmission Control Protocol/Internet Protocol. Rodina protokolů síťové komunikace.

- `Time`

  Obsahuje reprezentaci absolutního času nějaké události. Tato struktura je použita především jako člen ostatních struktur. V budoucnu se plánuje rozšíření o ukládání časové zóny. Informace obsažené v této struktuře nejsou používány pro žádnou logiku programu, slouží pouze jako informační údaj pro uživatele, protože údaje o absolutním čase na počítači nemusí být spolehlivé.

- `Day`

  Obsahuje informace o dni, kdy došlo k nějaké události. Koncept dne je vysvětlen v části Definice pojmů na straně 6. Struktura obsahuje informace o času zahájení a ukončení dne, jeho ID a další.

- `Table`

  Obsahuje informace o stolu jako jeho ID, jméno, pozici a rozměry. Pozice a rozměry se používají pro nakreslení plánku stolů.

- `Product`

  Obsahuje informace o prodávaném druhu zboží jako například jeho ID, název, cenu, je-li v současné nabídce a další příznaky. Při změně zboží se informace o předchozí verzi nemažou, pouze se označí jako neaktuální a vytvoří se nová verze. Proto tato struktura obsahuje i informace o své aktuálnosti a verzi.

- `Order`

  Obsahuje informace o objednávce jednoho kusu zboží, například její ID, dobu vytvoření, informace o uživateli, který ji vytvořil, druh objednávaného zboží, stůl, ke kterému objednávka patří, a informace o případné provedené platbě.

- `Payment`

  Obsahuje informace o platbě zahrnující její ID, dobu platby, typ platby (hotovostí nebo kartou), celkovou cenu a seznam zaplacených objednávek.

- `Customer`

  Obsahuje informace o zákazníkovi jako například jeho ID, zákaznický kód a jeho nárok na slevu.

- `CashChange`

  Obsahuje informace o změně množství hotovosti v kase, jako jsou její ID, čas změny, vložené nebo vybrané množství, nový celkový obnos v kase a informace o uživateli, který změnu provedl.

Navíc je definována výjimka `EasyPubException`, kterou mohou volané funkce vyhodit v případě chyby, a různé chybové kódy blíže identifikující důvod vzniku chyby.

4.3.2 Rozhraní serveru

Server poskytuje klientům rozhraní umožňující zjišťovat stav objektů uložených v databázi a modifikovat jej. Toto rozhraní zahrnuje i funkce zajišťující autentizaci uživatelů. Použití některých metod může být umožněno pouze uživatelům s patřičným oprávněním.

Metody pro zjišťování stavu objektů v databázi:

- `getDays`, `getOrders`, `getOrdersByPaymentId`, `getUsers`, `getUserRoles`, `getAllRoles`, `getPrivileges`, `getTables`, `getProducts`, `getCustomers`, `getCashChanges`, `getPayments`

  Vrací seznam všech objektů daného typu v databázi. Některé metody umožňují filtrovat seznam vrácených objektů podle jejich stavu nebo podle asociace s konkrétním stolem, dnem nebo platbou. Když žádný takový objekt neexistuje, vrátí prázdný seznam.

- `getDayById`, `getOrderById`, `getUserById`, `getTableById`, `getProductById`, `getCustomerById`

  Jako parametr dostane ID hledaného objektu v databázi, a pokud tento objekt existuje, vrátí jej volajícímu. Pokud se objekt nepodaří najít, vyhazuje výjimku.

- `getCurrentDay`, `getLastDay`, `getCurrentUser`, `getUserByName`, `getCustomerByCode`

  Podobně jako metody z předchozího odstavce se snaží najít v databázi a vrátit jeden hledaný objekt, na rozdíl od nich ho ovšem nehledají podle jeho ID ale podle jiného unikátního kritéria. `getCurrentDay` vrací den, který je v době volání aktivní. Pokud není žádný den aktivní, jde o chybu. `getLastDay` se chová stejně, pokud je nějaký den aktivní. Pokud není žádný den aktivní, vrací poslední aktivní den. `getCurrentUser` vrací aktuálně přihlášeného uživatele. `getCustomerByCode` vrací zákazníka podle jeho zákaznického kódu.

- `getCurrentCash`

  Vrací částku, která by měla být v současném okamžiku v kase.

- `getTableUrgencies`

  Vrací mapu přiřazující jednotlivým stolům informaci o tom, v jakém stavu jsou aktivní objednávky u těchto stolů. Slouží k rychlému zjištění, zda u některých stolů jsou nevyřízené nebo nezaplacené objednávky.

- `getProductPopularities`

  Vrací mapu přiřazující jednotlivým položkám sortimentu jejich popularitu. Popularita není v žádných konkrétních jednotkách, je určena pouze k porovnávání s popularitou ostatních položek. Tato funkce je používána k řazení položek v seznamu sloužícím k vytváření objednávek pro zjednodušení hledání oblíbeného zboží.

Metody modifikující stav objektů v databázi:

- `login`

  Umožňuje uživateli přihlásit se do systému. Její návratová hodnota je objekt třídy `SessionToken` (ve skutečnosti řetězec), který se předává jako jeden z parametrů při volání všech ostatních metod. Není-li tato funkce zavolána, nebude pro uživatele vytvořeno sezení a může přistupovat pouze k metodám, ke kterým má oprávnění anonymní uživatel.

- `changePassword`, `resetUserPassword`

  Nastaví heslo buď aktuálního uživatele (první metoda) nebo libovolného uživatele podle jeho ID (druhá metoda). První metoda vyžaduje od uživatele zadat i jeho aktuální heslo. Druhá metoda je zamýšlena pro použití správcem systému v případě, že jiný uživatel své heslo zapomene.

- `beginDay`, `endDay`

  Slouží k zahájení nebo ukončení dne. Před zahájením dne nesmí být žádný den aktivní. Před ukončením dne musí být nějaký den aktivní a všechny objednávky musí být vyřízené (buď zaplacené nebo zrušené).

- `setLastDayTips`

  Nastaví výši spropitného za poslední aktivní den. Žádný den nesmí být aktivní při volání této metody. Zároveň není možné volat tuto metodu, pokud spropitné za poslední den již bylo nastaveno dříve. Tato metoda je zamýšlena pro kontrolu obsahu kasy po ukončení dne s možností evidovat rozdíl oproti očekávané částce.

- `updateCash`

  Změní evidovaný obsah kasy. Umožňuje výběr nebo vložení hotovosti a uvedení komentáře k dané akci. Všechny výběry a vložení se logují v databázi.

- `updateTable`

  Vytvoří nový nebo upraví existující stůl. Umožňuje změnit jméno a umístění stolu.

- `createOrder`

  Očekává jako parametry druh zboží a stůl a vytvoří podle nich objednávku tohoto zboží. Při vytvoření objednávky musí být aktivní den.

- `updateOrderStates`, `moveOrders`

  Parametry obou metod jsou seznam ID objednávek, které mají být upraveny, a požadované modifikace. `updateOrderState` změní stav objednávky (například při vyřízení objednávky nebo zrušení). Není možné použít tuto funkci na změnu stavu na zaplacený, k tomu je nutné vytvořit a potvrdit platbu. `moveOrders` přesune zadané objednávky k jinému stolu. Obě tyto metody je možné použít pouze k modifikaci objednávek, které ještě nebyly zaplaceny ani zrušeny.

- `updateCustomer`

  Vytvoří nového zákazníka nebo upraví informace o existujícím zákazníkovi. Umožňuje editovat zákaznické kódy a slevu, na kterou mají zákazníci nárok.

- `preparePayment`, `confirmPayment`

  Vytvoří a potvrdí platbu. `preparePayment` dostane jako parametry seznam ID objednávek k zaplacení a ID zákazníka a vytvoří odpovídající platbu. Pokud jsou všechny údaje platby v pořádku, `confirmPayment` platbu potvrdí. Pokud nedojde k potvrzení platby a bude vytvořena jiná platba obsahující některou z objednávek ve staré platbě, bude stará platba automaticky zrušena.

- `updateUser`

  Vytvoří nového uživatele nebo uloží změněné informace o existujícím uživateli. Umožňuje měnit jména uživatelů a zamykat a povolovat uživatelské účty.

- `updateRole`

  Vytvoří novou nebo upraví existující uživatelskou roli. Umožňuje měnit názvy rolí.

- `setUserRoles`

  Přepíše seznam rolí asociovaných s daným uživatelem.

- `setRolePrivileges`

  Přepíše seznam oprávnění, které daná role poskytuje uživatelům s ní asociovaným.

4.4 Společné zdrojové kódy

Některé zdrojové kódy jsou sdíleny mezi serverem a klienty, protože jsou použity v obou částech. Patří mezi ně třídy modelující objekty uložené v databázi, třída a makro pro logování a funkce pro bezpečnou konverzi mezi celočíselnými typy.

4.4.1 Model databázových objektů

Deklarace tříd modelujících objekty uložené v databázi se nachází v souboru `src/model/Model.h`. Vznikly proto, že do struktur generovaných Thriftem není možné přidávat funkce, a tvoří tak jejich obal, který je obohacuje o novou funkcionalitu. Třídy Thriftu a modelové třídy je možné explicitně konvertovat z jedné na druhou a zpět pomocí funkcí `fromThrift` a `getThrift` implementovaných na modelových třídách.

Modelové třídy většinou pouze poskytují přístup k atributům struktur generovaných Thriftem, u tříd `Money` a `Percentage` přidávají přetěžování operátorů pro usnadnění provádění aritmetických operací a jejich porovnávání a zabránění přetečení a v případech, kde to dává smysl, umožňují formátovat objekty jako řetězce nebo jejich hodnoty z řetězců číst.

4.4.2 Logování

Soubor `src/common/Log.h` obsahuje deklaraci třídy `Log`, která slouží k logování událostí na několika různých úrovních závažnosti.

Samotná třída `Log` obsahuje pouze statické funkce umožňující nastavit, které úrovně událostí se logují a které se ignorují a do kterého souboru se zprávy ukládají. O samotné logování se stará třída `Log::Instance`. Když je vytvořen objekt této třídy, uloží si do svého bufferu aktuální čas, nastavenou úroveň logování a informace o souboru a řádce (předané jako parametry konstruktoru). Na tento objekt je potom možné použít operátor `<<`, který přidá další data do bufferu. Před zničením objektu je obsah bufferu zapsán na standardní chybový výstup a do logovacího souboru.

Protože přímé použití třídy `Log::Instance` by bylo těžkopádné, je zde také definované makro `LOG`, které umožňuje provádět logování následujícím způsobem:

```
LOG(INFO) << "Informace o události: " << x;
```

Makro `LOG` vytvoří objekt třídy `Log::Instance` a předá mu informace o souboru a řádku, kde bylo makro použito, takže programátor se těmito detaily nemusí zabývat. Po zapsání hodnot do vnitřního bufferu objektu dojde k jeho destrukci a vypsání bufferu na výstup.

4.4.3 Konfigurace

Třída `Configuration` se stará o načtení konfiguračního souboru `easypub.conf` a předání informací v něm obsažených ostatním částem aplikace. Většinu této třídy tvoří kód pro čtení informací z konfiguračního souboru. Více informací o konfiguračním souboru, datech v něm obsažených a jeho syntaxi je možné najít v uživatelské dokumentaci v sekci Konfigurace na straně 37.

4.4.4 Bezpečná konverze celočíselných typů

V souboru `src/common/SafeCast.h` je k dispozici funkce `safe_cast`, která umožňuje provádět bezpečné konverze z jednoho celočíselného typu na jiný. Jde o šablonovou funkci, která podle typu parametru zkontroluje, zda zadaná hodnota nepřekračuje rozsah cílového typu. Pokud je hodnota mimo rozsah, funkce vyhodí výjimku.

Díky použití šablon a optimalizacím prováděným kompilátorem je často možné jednu nebo obě kontroly vynechat a tak má použití funkce minimální dopad na rychlost běhu programu.

Funkci je možné použít následujícím způsobem:

```
int x = 42;
unsigned char y = safe_cast<unsigned char>(x);
```

Smyslem funkce je zabránit ztrátě informace při konverzi mezi typy a to i v případě, kdy programátor neví, jakou má typ, se kterým pracuje, velikost (například velikosti typů `int` nebo `size_t` jsou závislé na použité platformě).

4.5 Server

V této kapitole jsou popsány jednotlivé třídy, které dohromady implementují server systému EasyPub. Tyto třídy je možné najít v repositáři zdrojového kódu v podsložce `src/server`.

4.5.1 Obsluha síťových připojení

O obsluhu síťových připojení se stará framework Apache Thrift. Ten poskytuje několik různých implementací serveru, které mají shodné klientské rozhraní, ale liší se svým vnitřním fungováním. Pro EasyPub byl zvolen vícevláknový server, který pro každé aktivní spojení vytváří nové vlákno zajišťující komunikaci na tomto spojení a nový objekt `HandlerWrapper` zprostředkující přístup k databázi.

4.5.2 `HandlerWrapper`

Tato třída implementuje rozhraní serveru definované Thriftem. Jde o první bod, kde se obsluhují všechna vzdálená volání po jejich přijetí a zpracování Thriftem. V této třídě se nenachází přímo implementace logiky těchto volání, pouze se ověří, že uživatel má patřičná oprávnění, a volání se deleguje na odpovídající funkci ve třídě `HandlerEngine`. Tato třída navíc zajišťuje, že v průběhu vyhodnocování volání třídou `HandlerEngine` je aktivní SQL transakce, aby všechny provedené úpravy proběhly atomicky a v případě chyby je bylo možné zrušit a vrátit databázi do původního stavu.

4.5.3 `HandlerEngine`

V této třídě je implementována logika všech metod serveru. Protože jde o velké množství různorodých funkcí, je implementace této třídy rozdělena do více zdrojových souborů podle účelu jednotlivých funkcí (například funkce sloužící k přihlašování a zjišťování stavu uživatelů jsou implementovány v souboru `HandlerEngineUsers.cpp` zatímco funkce k vytváření, modifikaci a zjišťování stavu objednávek v souboru `HandlerEngineOrders.cpp`).

Funkce implementované v této třídě buď pracují přímo s databází za použití objektu třídy `SqlConn` a SQL kódu, nebo používají funkce třídy `SqlAccess` k načítání objektů z databáze.

Kvůli možnosti existence více objektů této třídy a jejich současného použití z více vláken je důležité, aby si každý z nich udržoval své vlastní „spojení“ s databází a aby nepoužívaly žádné globální nebo sdílené proměnné.

V souboru `HandlerEngine.cpp` jsou navíc implementovány některé funkce kontrolující stav objektů v databázi, které jsou používané více než jednou další funkcí, například funkce kontrolující, zda je v daném okamžiku aktivní den, což je nutná podmínka k provedení mnoha akcí, nebo funkce kontrolující, zda objekt se zadaným ID je skutečně v databázi.

Dále se zde nachází funkce `internalException` a `usageException`. Obě způsobí vyhození výjimky `EasyPubException` s patřičně nastaveným chybovým kódem. `internalException` se liší od `usageException` v tom, že při jejím zavolání je chyba zalogována, neboť se předpokládá, že jde o druh chyby, ke kterému by nemělo v běžném provozu dojít (například nesprávný předpoklad vývojáře, chyba

při zápisu na disk a tak dále). `usageException` je naopak zamýšlena jako reakce na chyby uživatele, například když je parametr funkce ID neexistujícího objektu nebo je objekt v takovém stavu, že s ním danou operaci nelze provést. Pokud není výjimka zachycena v kódu třídy `HandlerEngine`, třída `HandlerWrapper` nejdříve způsobí zrušení transakce a návrat dat v databázi do stavu před voláním a poté je výjimka dále propagována do kódu frameworku Thrift, kde je zachycena a přenesena jako výsledek volání do klientského procesu.

4.5.4 `SqlConn` a `SqlStmt`

Tyto dvě třídy poskytují nízkoúrovňový přístup k SQLite databázi. Jde o jediné dvě třídy, které přímo pracují s funkcemi knihovny SQLite, všechny ostatní části kódu přistupují k databázi prostřednictvím těchto tříd. Jde tak vlastně o obal funkcí knihovny SQLite využívající vlastností jazyka C++.

Mezi výhody používání těchto tříd oproti funkcím knihovny SQLite patří automatické uvolnění alokovaných prostředků při smazání objektu třídy `SqlStmt`, automatické použití správné funkce podle typu parametru při použití volání `bind` a vyhození výjimky v případě, že volaná funkce skončí s chybovým návratovým kódem.

Hlavním cílem těchto tříd je zamezení chybám způsobeným nepozorností programátora, ke kterým by mohlo dojít při používání SQLite funkcí. Druhotný pozitivní efekt je, že kdyby bylo učiněno rozhodnutí použít jinou SQL databázi, stačilo by změnit kód těchto dvou tříd (pokud nebereme v potaz, že by mohlo být nutné přepsat části SQL kódu). A nakonec, díky variadickým šablonám ze standardu C++11, je často možné napsat kratší a přehlednější kód, než kdyby byly používány přímo nízkoúrovňové funkce knihovny SQLite.

Nevýhodou používání těchto tříd je, že nepodporují veškerou funkcionalitu, kterou je možné nalézt v knihovně SQLite. V případě potřeby by ovšem neměl být problém tyto třídy o chybějící funkcionalitu rozšířit.

Za účelem zabránění potenciálnímu útočníkovi ve vložení škodlivého kódu do SQL dotazů (SQL-injection, blíže popisuje Halfond et al. [1]) jsou použity připravené výrazy (prepared statements), do kterých se bezpečnými `bind` funkcemi dosazují konkrétní hodnoty proměnných. Dotazy jsou předávány jako řetězcové literály. Tím je zajištěno, že proměnné budou před vložením do výrazu řádně ošetřeny a nebudou interpretovány jako kód a žádný škodlivý kód se do výrazů nemůže dostat.

Jediná výjimka z tohoto pravidla je třída `SqlAccess`, která používá dynamicky generované SQL dotazy. Ta se ovšem pečlivě vyhýbá použití uživatelského vstupu v samotných dotazech. Pokud je potřeba výsledky filtrovat podle hodnot pocházejících od uživatele, je i v ní možné použít parametry, které jsou dosazeny do výrazu prostřednictvím `bind` funkcí.

4.5.5 `SqlTrans`

Tato třída usnadňuje správu SQL transakcí. Jejím hlavním přínosem je omezení životnosti transakce po dobu, kdy existuje objekt této třídy. Při vytvoření objektu `SqlTrans` je zahájena SQL transakce. Pokud není tato transakce explicitně potvrzena před smazáním objektu, je zrušena a všechny změny jsou vráceny zpět.

Tedy například při vyhození výjimky není nutné zajišťovat zrušení databázové transakce, neboť se o to postará destruktor objektu této třídy.

Při použití této třídy je nutné, aby se programátor vyvaroval práce s transakcemi jinak než prostřednictvím této třídy. Například, když použije SQL příkaz `COMMIT` místo funkce `SqlTrans::commit()`, třída se to nedozví a při mazání objektu se pokusí provést zrušení transakce, která ovšem už není platná.

Zároveň je nutné se vyvarovat současné existence dvou objektů této třídy nad jedním připojením k databázi, protože jazyk SQL zakazuje vytvářet nové transakce, je-li jiná transakce již aktivní. Při pokusu o vytvoření druhého objektu třídy nad stejnou databází tedy dojde k chybě.

4.5.6 `SqlAccess`

Tato třída poskytuje přístup k objektům v databázi na vyšší úrovni, než `SqlConn` a `SqlStmt`, ale umožňuje objekty z databáze pouze číst, není možné je prostřednictvím této třídy modifikovat. Třída umožňuje použít například následující příkaz k načtení všech objednávek v databázi:

```
// db je objekt třídy SqlConn
vector<Order> orders = SqlAccess<Order>::load(db, {});
```

Druhý parametr funkce `load` je možné použít k filtrování výsledků (předaný seznam je spojen logickými operátory `AND` a vložen do `WHERE` podmínky SQL dotazu). Pokud by v druhém parametru byl předán kód z nedůvěryhodného zdroje (například od uživatele), mohlo by dojít k vložení škodlivého kódu do SQL dotazu, proto jsou k dispozici další volitelné parametry, které slouží k bezpečnému předání hodnot proměnných odkazovaných v druhém parametru.

Smysl této třídy je především zjednodušení získávání objektů z databáze. Vzhledem k tomu, že některé objekty mohou obsahovat jiné objekty (například objednávky obsahují informace o uživateli, který objednávku vytvořil, stolu, ke kterému objednávka patří, zboží, které bylo objednáno, a další). Důsledkem toho pro načtení jednoho objektu může být potřeba provést dotaz na několik tabulek a několik desítek sloupců, které je potřeba správně načíst. To je velké množství kódu, který by bylo nutné opakovat na více místech a bylo by snadné v něm udělat chybu. Použitím této třídy je zajištěno, že proměnné budou načteny ze správných sloupců a zároveň se minimalizuje duplikace kódu.

Pro ilustraci komplexity dotazů, o nichž je řeč, je zde uveden SQL dotaz vygenerovaný příkazem ze začátku této sekce (odsazení doplněno pro lepší čitelnost):

```
SELECT
    a.id b,a.time c,a.cost d,a.no_discount e,a.name f,a.state g,
    ua.id va,ua.name wa,ua.locked xa,
    z.id aa,z.begin ba,z.end ca,z.tips da,
    ea.id fa,ea.time ga,ea.type ha,ea.cost_before_discount ia,
        ea.discount ja,ea.cost ka,
    qa.id ra,qa.identifier sa,qa.discount ta,
    la.id ma,la.begin na,la.end oa,la.tips pa,
    o.id p,o.version q,o.time r,o.current s,o.active t,o.name u,
        o.cost v,o.over18 w,o.modifiable x,o.no_discount y,
    h.id i,h.name j,h.pos_x k,h.pos_y l,h.size_x m,h.size_y n
FROM
    orders a
    LEFT JOIN users ua ON a.staff_id = ua.id
    LEFT JOIN days z ON a.day_id = z.id
    LEFT JOIN payments ea ON a.payment_id = ea.id
    LEFT JOIN customers qa ON ea.customer_id = qa.id
    LEFT JOIN days la ON ea.day_id = la.id
    LEFT JOIN products o ON a.product_id = o.id
                            AND a.product_version = o.version
    LEFT JOIN tables h ON a.table_id = h.id
```

Pro činnost metody `load` je nutné mít informaci, ve kterých tabulkách hledat data objektu a jakého jsou typu. Tato informace se sděluje prostřednictvím specializace funkce `initialize` v souboru `SqlAccessLoaders.cpp`, kde se nastaví proměnná `tableName` a pomocí definovaných maker přidají datová pole a podobjekty, které objekty daného typu obsahují.

Zavolání funkce `load` způsobí rekurzivní zjištění jmen tabulek a sloupců, ze kterých je potřeba přečíst data objektu a jeho podobjektů. Tyto jsou potom zkombinovány do jednoho SQL dotazu, který je následně spuštěn na databázi. Každé tabulce a každému sloupci je přiřazen jednoznačný alias, aby bylo zabráněno konfliktu mezi sloupci různých tabulek se stejným jménem a tabulkami, do kterých se přistupuje více než jednou v rámci jednoho dotazu (těchto aliasů je možné si všimnout jako ve dříve uvedené ukázce, jde o jedno- nebo dvoupísmenné názvy).

Všechny tabulky podobjektů jsou připojeny příkazem `LEFT JOIN`, protože se předpokládá, že bude existovat nejvýše jeden takový podobjekt. Podmínky použité pro připojení tabulek podobjektů jsou specifikované ve funkci `initialize`. Kvůli použití aliasů v dotazech musí být jména tabulek v těchto podmínkách být nahrazeny jejich aliasem. K tomu je použita speciální syntaxe, která nahradí výrazy typu `$sloupec` nebo `$tabulka.sloupec` aliasem použitým pro daný sloupec. Stejná nahrazení jsou provedena na filtry předané v druhém parametru metody `load`.

4.5.7 Schéma databáze

Server používá SQLite databázi pro uložení všech dynamických dat. SQLite je knihovna implementující podmnožinu SQL funkcionality, která nepoužívá samostatný proces pro SQL server, tedy všechny přístupy k souboru databáze jsou prováděny přímo z procesu, který volá funkce SQLite knihovny. To dává serveru větší kontrolu nad databází, mimo jiné si ji může sám vytvořit a nakonfigurovat při prvním spuštění a není ji třeba nijak samostatně konfigurovat předem. Na druhou stranu to může mít negativní dopad na výkonnost databáze a znemožňuje to implementaci určité funkcionality. Například implementace uživatelských oprávnění v SQLite by nedávala smysl, protože celá databáze je uložena v lokálním souboru, a pokud má proces oprávnění k jeho čtení nebo zápisu do něj, může číst nebo upravovat veškerá data v databázi.

Schéma databáze většinou odráží definici tříd v rozhraní serveru, kde každá tabulka obsahuje data objektů jedné třídy. Téměř všechny tabulky objektů mají jako primární klíč sloupec `id`, který ostatní tabulky používají jako cizí klíč, když chtějí na objekty daného typu odkazovat. Výjimkou je tabulka s informacemi o zboží `products`, která má jako primární klíč sloupce `id` a `version`, protože jsou v ní uložené i všechny minulé verze daného druhu zboží.

Navíc jsou v databázi některé pomocné tabulky umožňující reprezentovat `n:n` relace mezi určitými třídami objektů. Konkrétně se jedná o tabulky `user_roles` a `role_privileges`. Tyto tabulky obsahují pouze dva sloupce, totiž cizí klíče odkazující do tabulek, které spojují. Jejich dva sloupce zároveň tvoří jejich primární klíč.

V příloze B je k dispozici ER[3] diagram dokumentující všechny typy objektů uložených v databázi a vztahy mezi nimi.

SQLite nepodporuje typ reprezentující logickou pravdu a nepravdu ani výčtové typy, takže k uložení pravdivostních hodnot jsou použity celočíselné typy, kde `0` reprezentuje nepravdu a `1` pravdu.

SQLite je transakční databáze, takže při ukončení transakce se nejdříve čeká na potvrzení, že byla všechna data fyzicky zapsána na disk, než je navrácena kontrola procesu. To může trvat relativně dlouho, především u pevných disků, kde může být potřeba čekat na fyzický pohyb disku (řádově jednotky až desítky milisekund). Proto se pro tabulky, kde se předpokládá časté zapisování a není vyžadována persistence, používá databáze uložená v paměti, která je výrazně rychlejší. V současnosti je použita pro ukládání informace o poslední aktivitě uživatelského sezení, která je aktualizována při každém uživatelském dotazu. Vedlejší efekt toho je ukončení všech uživatelských sezení při restartu serveru.

4.5.8 Automatické aktualizace databáze

V průběhu vývoje systému došlo k několika úpravám schéma databáze podle změn a upřesňování požadavků zákazníka a dolaďování implementačních detailů. Tyto úpravy nebyly vždy zpětně kompatibilní se staršími verzemi systému, takže před použitím databáze je nutné ověřit, že schéma databáze má takovou verzi, jakou systém očekává. Proto se při spuštění serveru před zahájením obsluhy dotazů klientů nejprve zkontroluje verze databáze. Pokud je databáze novější, než s jakou

[3] *ER* – Entity-relationship. ER diagramy jsou grafy znázorňující vztahy mezi třídami objektů.

systém umí pracovat, jde o chybu a server se okamžitě ukončí, aby nezpůsobil korupci dat. Pokud je starší, než jakou systém očekává, provedou se postupně jednotlivé aktualizace a databáze se uvede do očekávané podoby. V průběhu aktualizace je aktivní SQL transakce, tedy dojde-li k chybě, jsou všechny dosud provedené změny zrušeny a databáze se vrátí do původního stavu.

Informace o aktuální verzi databáze jsou uloženy v tabulce `version`. Pokud tato tabulka neexistuje, předpokládá se, že jde o prázdnou databázi, databáze je inicializována na první verzi a pak se postupně provedou všechny aktualizace.

Samotná funkce `Update<n>::apply` zkontroluje aktuální verzi databáze, a je-li potřeba, zavolá funkce `Update<n>::applyDependencies` a `Update<n>::update`. První z nich pouze zavolá `Update<n-1>::apply`, která zajistí aplikaci všech předchozích aktualizací (s výjimkou pro $n = 1$, která nemá žádné závislosti). Druhá je potom specializovaná pro každou verzi a provádí samotnou aktualizaci z verze $n - 1$ na verzi n.

Navíc, pokud jde o první spuštění programu (to server určí podle chybějící tabulky `version`), třída `HandlerEngine` se postará o umístění ukázkových dat do databáze, aby byla zajištěna existence alespoň jednoho uživatelského účtu a aby uživatel nemusel program pracně nastavovat před tím, než ho bude moci vyzkoušet.

4.5.9 Kryptografické funkce

Server obsahuje několik kryptografických funkcí sloužících především k bezpečnému ukládání hesel uživatelů. Kdyby byla samotná hesla ukládána v databázi, pak kdokoliv, kdo k ní má přístup by mohl zjistit hesla všech uživatelů a potenciálně je zneužít. Proto jsou v databázi uloženy pouze otisky uživatelských hesel získané hašovací funkcí. Pro ztížení nalezení vzoru otisku jsou použity výpočetně náročné hašovací funkce s vysokým počtem iterací. Navíc je heslo každého uživatele spojeno s kryptografickou solí (náhodné číslo, které je uloženo spolu s otiskem hesla) aby heslo každého uživatele muselo být zjišťováno samostatně.

Tento způsob ukládání hesel, ačkoliv znesnadňuje útočníkovi zjistit hesla uživatelů, stále umožňuje relativně snadné ověření, zda otisk patří heslu, které uživatel zadal.

Server používá implementaci hašovací funkce a funkce pro získání náhodných čísel z knihovny OpenSSL. Použitou hašovací funkci specifikuje Kaliski [2] jako PBKDF2[4] používající HMAC[5] se SHA-1[6] jako pseudonáhodnou funkci. Tuto kombinaci doporučuje Turan et al. [4]

Každý otisk je uložený v databázi jako řetězec sestávající ze tří částí oddělených od sebe dvojtečkami. První dvě části jsou parametry pro PBKDF2, konkrétně počet iterací hašovací funkce zapsaný v desítkové soustavě a kryptografická sůl zapsaná v šestnáctkové soustavě. Třetí část je výstup PBKDF2 zapsaný v šestnáctkové soustavě. Kontrola hesla probíhá tak, že se přečtou první dvě části a spolu s heslem zadaným uživatelem se předají hašovací funkci. Heslo je potom

[4] *PBKDF2* – Password-Based Key Derivation Function 2. Algoritmus umožňující vygenerovat kryptografický klíč z hesla.

[5] *HMAC* – Hash-based message authentication code. Algoritmus sloužící k autentizaci zpráv s použitím sdíleného klíče.

[6] *SHA* – Secure Hash Algorithm. Hašovací funkce sloužící k vytvoření otisku dat.

považováno za správné, pokud se výstup hašovací funkce shoduje s třetí částí řetězce uloženého v databázi.

4.6 Klientské programy

V rámci práce byly vytvořeny dva různé programy schopné komunikovat se serverem. Jde o uživatelské a administrační rozhraní systému. Oba tyto programy jsou napsány v jazyce C++ a ke komunikaci s uživatelem používají grafické rozhraní vytvořené za použití knihovny wxWidgets. Ačkoliv většina kódu obou programů je specifická pro daný program, některé části kódu jsou mezi programy sdílené.

4.6.1 Sdílený kód

Kód, který klientské programy sdílí, se z většiny nachází v repositáři zdrojových kódů ve složce `src/gui`. Výjimkou je třída `Session`, kterou je možné najít ve složce `src/client`.

Správa spojení se serverem

O spojení se serverem se stará třída `Session`. Její rozhraní je velice podobné samotnému rozhraní serveru, ovšem s tím rozdílem, že jejím funkcím není potřeba při každém volání předávat kód sezení (`SesionToken`). Objekt třídy `Session` si totiž při přihlášení tento kód zapamatuje a sám se stará o jeho předávání. Další odpovědností této třídy je vytvoření připojení k serveru, pokud není aktivní. V některých případech navíc zjednodušuje práci programátora použitím přetěžování metod a volitelných parametrů, které Thrift sám o sobě nepodporuje.

Asynchronní dotazy

Je dobrým zvykem, že aplikace zpracovávají události grafického rozhraní a reagují na ně, i když zrovna v pozadí provádějí nějakou dlouhotrvající operaci. To mimo jiné zajišťuje, že je operační systém neoznačí jako neodpovídající.

Volání na objektu, který Thrift poskytuje ke komunikaci se serverem, jsou synchronní a mohou trvat relativně dlouhou dobu, tedy není možné je používat z hlavního vlákna a zároveň v něm pravidelně obsluhovat události grafického prostředí. Z toho důvodu byla vytvořena třída `BackgroundThread`, jejímž úkolem je provádět takováto volání na pozadí v jiném vlákně, zatímco hlavní vlákno se dále stará o zpracování událostí. Když úkol běžící na pozadí dokončí svou činnost, třída vygeneruje událost pro hlavní vlákno, které provede kód zpracovávající výsledky dokončeného úkolu.

Jedním z omezení Thriftu je, že objekt komunikující se serverem je možné používat pouze z jednoho vlákna a navíc to musí být to vlákno, ve kterém byl tento objekt vytvořen. To samé omezení platí i pro grafické rozhraní a většinu tříd knihovny wxWidgets, pokud přístup k nim není jinak synchronizován. Proto je nutné vzdálené volání metod na serveru (respektive použití objektu třídy `Session`) důsledně provádět pouze ve vláknu běžícím na pozadí a návratové hodnoty zpracovávat až v kódu běžícím v hlavním vlákně.

Třída `BackgroundThread` umožňuje splnění těchto omezení prostřednictvím své metody `enqueue`. Ta dostává tři parametry. První z nich je funkce, která bude spuštěna ve vláknu na pozadí a dostane jako parametr objekt třídy `Session`. Tím je omezena dostupnost objektu třídy `Session` a do určité míry zamezeno jeho použití z hlavního vlákna. Druhý a třetí parametr jsou funkce, z nichž jedna se vyvolá v hlavním vlákně po dokončení úkolu běžícího na pozadí. V případě úspěšného dokončení se zavolá funkce z druhého parametru, která může dále zpracovat návratové hodnoty. V případě, že úkol na pozadí skončí výjimkou, zavolá se funkce ze třetího parametru, která jako parametr dostane chybovou hlášku. Očekává se od ní, že ošetří vzniklý chybový stav.

V kombinaci se syntaxí pro anonymní funkce podporovanou jazykem C++11 toto rozhraní poskytuje relativně přístupnou cestu, jak asynchronně provádět vzdálená volání funkcí na serveru. Samozřejmě je stále potřeba dbát na synchronizaci mezi vlákny a korektní správu životnosti objektů, ke kterým může být přistupováno z více vláken. Pokud je potřeba, aby byl ve vláknu na pozadí dostupný dočasný objekt vytvořený ve hlavním vláknu, je možné buď zachytit jeho kopii nebo použít chytrý ukazatel `shared_ptr`, který zajistí jeho sdílené vlastnictví a jeho destrukci, až ho všichni vlastníci přestanou používat.

Různé pomocné funkce a třídy

Dále se v adresáři `src/gui` nachází kód zjednodušující práci s knihovnou wxWidgets. V souboru `StringConversion.h` jsou k dispozici funkce pro převod mezi třídami `std::string` a `wxString`. Třída `RedirectEvents` umožňuje přijímat události grafického rozhraní a přesměrovávat je na jiný objekt. Soubor `WxProxy.h` zajišťuje import hlavičkového souboru `wx/wx.h` (na Windows má tento soubor konflikty s makry definovanými ve `windows.h`, které jsou ve `WxProxy.h` ošetřeny). V souboru `GuiErrors.h` jsou k dispozici různé funkce pro zobrazení chybových zpráv a získání chybové zprávy z objektu výjimky.

Dále jsou zde definovány různé grafické prvky, které mohou být použité v obou grafických rozhraních, jako například dialog sloužící k přihlášení uživatele.

4.6.2 Uživatelské rozhraní

Zdrojové kódy specifické pro uživatelské rozhraní se nacházejí ve složce `src/user`. Jde z velké většiny o rozšíření tříd knihovny wxWidgets, která implementují grafické rozhraní tak, jak je popsáno ve specifikaci.

Samotná aplikace je tvořena třídou `ClientApp`. Její hlavní okno představuje třída `MainFrame`. Ta používá ostatní třídy pro zobrazení jednotlivých částí grafického rozhraní (seznam objednávek, mapu stolů, detaily stolu) a stará se o zpracování událostí nástrojové lišty.

Seznam objednávek v levé části hlavního okna implementuje třída `OrderLog`, která je spolu se třídou `OrderLogControls` (tlačítko pro seřazení seznamu) obalena třídou `OrderLogWrapper`. Jednotlivé objednávky v tomto seznamu jsou potom představovány instancemi třídy `LogItem`.

Grafická mapa stolů v pravé části hlavního okna je implementována v třídě `Tables`. Jednotlivá tlačítka stolů jsou instance třídy `TableButton`.

Detail stolu, který se zobrazí po kliknutí na tlačítko stolu, je implementován v třídě `TableDetail`. Obsahuje dvě části: seznam objednávek u tohoto stolu (třída

`TableDetailOrderList`, jednotlivé objednávky jsou představovány instancemi třídy `TableDetailOrder`) a seznam nabízeného zboží (třída `ProductSelection`, jednotlivé typy zboží představovány instancemi třídy `ProductItem`).

Jednoduché dialogy (žádost o potvrzení, dialog pro zadání jediné hodnoty a tak dále) jsou implementovány jako objekty třídy `wxDialog` vytvořené na zásobníku. Složitější dialogy jsou potom realizovány třídami, které dědí od `wxDialog` (například `OrderDialog` pro vytvoření objednávky nebo `PaymentDialog` pro vytvoření a potvrzení platby).

Třídy `OrderLog` (seznam v levé části okna zobrazující všechny objednávky z aktuálního dne), `TableDetailOrderList` (seznam objednávek při zobrazení detailu stolu) a `ProductSelection` (seznam nabízeného zboží v pravé části detailu stolu) dědí od třídy `gui::ListOfPanels`, která jim zjednodušuje zobrazování seznamu položek a jeho procházení. Tyto třídy se tak díky svému společnému předkovi mohou vyhnout duplikaci kódu.

Třída `Printer` se stará o vykreslování a tisk účtenek. V současné verzi neumožňuje multiplatformní podporu tisku, tisk byl testován pouze na Linuxu a pouze s tiskárnou Citizen CT-S280 USB, s níž program komunikuje prostřednictvím sériového rozhraní (při vývoji systému došlo k problémům s instalací ovladačů této tiskárny, proto byla jako alternativa zvolena přímá komunikace s tiskárnou bez použití ovladačů). Podpora tisku na více platformách bude předmětem další práce, současné řešení je pouze provizorní.

Pro testování (nebo ověření funkcionality) vykreslování jsou obrázky účtenek před odesláním na tiskárnu uloženy do souboru `receipt.png`.

Části vzhledu účtenky je možné měnit z konfiguračního souboru, takže při tisku je k dispozici objekt třídy `common::Configuration`, jehož obsahem se řídí kód zajišťující vykreslování.

4.6.3 Administrační rozhraní

Zdrojové kódy administračního rozhraní se nacházejí ve složce `src/admin`. Aplikace je implementována třídou `AdminApp`, hlavní okno třídou `AdminFrame`, která se podobně jako `MainFrame` v uživatelském rozhraní stará o zpracování událostí nástrojové lišty a zobrazení dalších částí grafického rozhraní.

Nejdůležitější třídou administračního rozhraní je `ObjectGrid`. Ta umožňuje zobrazovat data systému v tabulce. Data, která má zobrazovat, jsou jí předávána implementací abstraktní třídy `ObjectGrid::DataSource`. Tato abstraktní třída (či spíše rozhraní) má několik různých implementací, jednu pro každý z typů tabulek, které je možné v administračním rozhraní zobrazit.

Některé implementace `DataSource` umožňují editování zobrazených dat. K tomu slouží šablonová třída `CellType`, která je schopna v závislosti na typu proměnné vytvořit správný grafický ovládací prvek pro její zobrazení a případnou editaci a později převést stav tohoto grafického prvku zpět na hodnotu proměnné. Takto vytvořené prvky jsou při žádosti o editaci dat umístěny do dialogu, kde má uživatel příležitost upravovat jejich hodnoty. Po potvrzení dialogu jsou z nich vyčteny hodnoty, které jsou potom předány serveru k uložení do databáze.

Všechny ostatní třídy jsou pouze specifické implementace `DataSource` nebo pomocné třídy používané k zobrazení komplexnějších ovládacích prvků.

4.7 Budoucí práce

4.7.1 Multiplatformní podpora tisku

Jak bylo zmíněno dříve, tisk je v současnosti podporován pouze na Linuxu na jednom konkrétním typu tiskárny. Pro budoucí vývoj systému je nutné implementovat multiplatformní podporu tisku využívající služby poskytované operačním systémem.

4.7.2 Zabezpečení

Ačkoliv současná architektura umožňuje serveru ověřit identitu klientů prostřednictvím jména a hesla, neexistuje žádný způsob, jak by mohli klienti ověřit identitu serveru. Síťové spojení mezi klienty a serverem navíc není nijak šifrované.

Nepřítomnost autentizace serveru způsobuje, že systém není chráněn proti aktivním útokům (man-in-the-middle attack). Kvůli chybějícímu šifrování je navíc zranitelný i pasivními útoky (odposlouchávání spojení). Proto je z hlediska bezpečnosti nutné, aby byl systém provozován jedním z následujících způsobů:

- Server i klienti běží na stejném stroji, komunikují prostřednictvím loopback rozhraní (vůbec nepoužívají síť).
- Server a klienti komunikují prostřednictvím soukromé sítě s omezeným přístupem, kde všechna zařízení jsou důvěryhodná.
- Server a klienti komunikují prostřednictvím virtuální soukromé sítě nebo je spojení mezi nimi jiným způsobem tunelováno skrz zabezpečený protokol.

Toto omezení protokolu by mělo být možné odstranit, neboť Thrift má podporu pro vzájemnou autentizaci obou komunikujících stran a šifrování spojení, k čemuž využívá knihovnu OpenSSL. Bude ovšem potřeba vymyslet způsob generování, distribuce a ověřování platnosti certifikátů a další detaily zabezpečení.

4.7.3 Nové klientské programy

Díky použití Thriftu při vývoji serveru je možné vytvořit nové klientské programy v libovolném jazyce podporovaném Thriftem. Programy napsané v C++ mohou využít modelové třídy a třídy pro správu sezení implementované v rámci této práce. Ostatní programy budou muset obsahovat vlastní implementaci těchto tříd nebo jejich funkcionalitu nahradit jiným způsobem.

Příklady námětů pro budoucí klientské programy zahrnují aplikaci pro mobilní telefony a webové rozhraní systému.

5. Uživatelská dokumentace

5.1 Instalace

5.1.1 Microsoft Windows

Distribuce systému EasyPub pro operační systém Microsoft Windows používá instalátor vytvořený programem NSIS (Nullsoft Scriptable Install System[1]). Pro instalaci systému tímto instalátorem spusťte program `EasyPub_Setup.exe` a dále se řiďte jeho pokyny. Jako cíl instalace je potřeba vybrat složku, do níž bude mít uživatel systému mít právo zápisu. Před samotnou instalací je možné si zvolit části systému, které chcete nainstalovat:

- Server

  Je nutný pro běh systému, ale pro systém operující na více počítačích stačí, když serverová část bude nainstalován pouze na jednom z nich.

- Client

  Standardní rozhraní systému pro každodenní použití. Poskytuje přehled o aktivních objednávkách, umožňuje s objednávkami manipulovat a vytvářet platby.

- Admin

  Rozhraní určené pro správce systému. Umožňuje upravovat nastavení systému, editovat seznamy zboží, rozložení stolů, uživatelské účty a jejich oprávnění.

Instalátor také umožňuje vytvoření zástupců v nabídce Start pro spuštění jednotlivých částí systému.

5.1.2 Ostatní operační systémy

Pro ostatní operační systémy není k dispozici žádný instalátor ani balík a systém je nutné zkompilovat ze zdrojových kódů. Pravděpodobně bude nutné zkompilovat i některé jeho závislosti. Bližší instrukce jsou k dispozici v elektronické příloze práce.

5.2 Spuštění systému

Systém, jak bylo popsáno v kapitole zabývající se jeho návrhem, sestává z několika samostatných částí. To znamená, že k jeho použití nestačí spustit pouze jeden program. Pro vyzkoušení systému v režimu běhu na jednom počítači je potřeba nejdříve spustit server a teprve poté spustit klienta. Klient by se ve výchozí konfiguraci měl automaticky pokusit připojit k serveru běžícím na stejném počítači a komunikovat s ním.

[1] `http://nsis.sourceforge.net`

Ke spuštění uživatelské části je tedy potřeba provést následující kroky:

- Pokud již neběží, spusťte program `easypub-server`.
- Spusťte program `easypub-client`.

Postup spuštění administračního rozhraní je obdobný:

- Pokud již neběží, spusťte program `easypub-server`.
- Spusťte program `easypub-admin`.

5.3 Uživatelské rozhraní

Po spuštění uživatelského rozhraní se zobrazí hlavní obrazovka programu[2]. Ta je svisle rozdělena na dvě části. V levé části programu je vidět seznam objednávek, v pravé části mapa stolů. Za povšimnutí stojí také nástrojová lišta na horní části obrazovky, přes kterou je možné přistupovat k méně často používaným funkcím programu.

Před dalším použitím programu je potřeba se přihlásit. To je možné udělat zvolením tlačítka „Přihlášení“ v nabídce „Obsluha“ na nástrojové liště v horní části obrazovky, jak je popsáno v části 5.3.4. Při prvním spuštění systému je automaticky vytvořen uživatelský účet `user` s prázdným heslem. Tento účet má administrátorská oprávnění. Další účty je možné vytvořit v administračním rozhraní.

5.3.1 Seznam objednávek

Objednávky v seznamu v levé části hlavní obrazovky obsahují informace o objednávkách jako například čas jejich vzniku, stůl, ke kterému patří, a název a cenu objednaného zboží. Navíc jsou barevně kódované podle svého stavu. Objednávky mohou být označeny následujícími barvami:

- oranžová – nevyřízená objednávka
- zelená – vyřízená, ale nezaplacená objednávka
- modrá – zaplacená objednávka
- černá – zrušená objednávka

Při vytváření objednávek jsou nové objednávky přidávány na horní konec seznamu. Objednávky je možné seřadit podle jejich stavu a času vzniku kliknutím na tlačítko „Seřadit podle stavu“.

U aktivních objednávek je možné měnit jejich stav. Kliknutím levým tlačítkem myši na barevné tlačítko u objednávky se změní stav na vyřízený a nezaplacený. Změnu na ostatní stavy (například na nevyřízený stav v případě chybného označení objednávky jako vyřízené) je možné provést z kontextové nabídky, která se zobrazí po kliknutí pravým tlačítkem myši kamkoliv na informace o objednávce.

[2] Její podobu je možné vidět na obrázku C.1

5.3.2 Mapa stolů

V pravé části hlavní obrazovky se nachází grafická mapa stolů. Každý stůl má přiřazenou barvu podle nejurgentnější aktivní objednávky, která k němu patří. Je tak rychle možné zjistit, u kterých stolů jsou nevyřízené nebo nezaplacené objednávky.

Po kliknutí na stůl se místo mapy stolů zobrazí detail vybraného stolu.

5.3.3 Detail stolu

Tato obrazovka je svisle rozdělena na tři části. Vlevo je seznam všech objednávek (stejný jako seznam v levé části hlavní obrazovky). V prostřední části je seznam aktivních objednávek přiřazených k vybranému stolu. V pravé části je seznam všech nabízených druhů zboží.

Pro návrat k mapě stolů slouží tlačítko „Zpět“ umístěné v prostřední části u spodního okraje obrazovky.

V prostřední části je možné vybírat a rušit objednávky, přesouvat je k jiným stolům a vytvářet platby. Vybrání objednávky nebo zrušení vybrání je možné uskutečnit kliknutím levým tlačítkem myši na danou objednávku. Alternativně je možné vybrat všechny objednávky u daného stolu, zrušit vybrání objednávek nebo vybrat právě ty objednávky, které zrovna nejsou vybrané pomocí tlačítek „Vybrat vše“, „Nevybrat nic“ a „Invertovat výběr“. Aktuálně vybrané objednávky jsou zvýrazněny modrou barvou a mají u sebe zaškrtnutou kolonku.

Přesunutí objednávek k jinému stolu je možné provést vybráním objednávek, které mají být přesunuty, stiskem tlačítka „Přesunout“ a vybráním cílového stolu.

Vybrané objednávky je možné zrušit stiskem tlačítka „Zrušit vybrané“.

K vytvoření platby slouží tlačítka „Zaplatit vše“ a „Zaplatit vybrané“. První z nich připraví platbu za všechny aktivní objednávky u tohoto stolu, druhé připraví platbu obsahující pouze vybrané objednávky. Tak je možné účtovat útratu za celý stůl dohromady nebo ji rozdělit do více plateb. Po připravení platby se zobrazí dialog s informacemi o platbě, kde je možné zadat číslo uživatelského účtu pro uplatnění slevy, nechat vytisknout účtenku a platbu potvrdit. Po potvrzení platby jsou všechny objednávky do ní patřící automaticky označeny jako zaplacené.

K vytváření objednávek slouží seznam zboží v pravé části obrazovky detailu stolu. Vytvoření objednávky se provede kliknutím na název zboží, zadáním počtu kusů a potvrzením. Ve spodní části seznamu zboží se nachází textové pole sloužící k vyhledávání zboží. Po zadání části názvu zboží do tohoto pole se ze seznamu odstraní všechny druhy zboží, které tento text neobsahují ve svém názvu. Při vyhledávání není kladen důraz na velikost písmen, českou diakritiku ani pořadí zadaných slov.

Místo používání myši je možné se v seznamu zboží orientovat šipkami nahoru a dolů na klávesnici a vytvoření objednávky potvrdit klávesou Enter.

5.3.4 Další funkce

Vybírání a vkládání hotovosti

Program si pamatuje informace o množství hotovosti v kase. Při vytvoření platby je toto množství automaticky zvýšeno o celkovou hodnotu platby. Někdy je potřeba hotovost z kasy vybrat nebo ji do ní vložit. K tomu slouží odpovídající tlačítka v nabídce „Kasa“ na nástrojové liště v horní části obrazovky. Po jejich stisku se zobrazí dialog informující o aktuální částce evidované v kase, kam je možné zadat částku k výběru nebo vložení spolu s komentářem a evidovaná hodnota bude o tuto částku upravena. Při této operaci se ukládá i jméno právě přihlášeného uživatele.

Přihlášení a odhlášení uživatele

Přihlášení uživatele je možné provést stiskem tlačítka „Přihlášení“ v nabídce „Obsluha“ na nástrojové liště v horní části obrazovky. Zobrazí se dialog, do něhož uživatel může zadat své přihlašovací údaje a tak se přihlásit. Odhlášení právě přihlášeného uživatele se provede stiskem tlačítka „Odhlášení“ v té samé nabídce.

V závislosti na konfiguraci systému se mohou oprávnění jednotlivých uživatelů lišit. Uživatelská oprávnění je možné nastavit v administračním rozhraní.

Změna uživatelského hesla

Změnu hesla je možné provést po přihlášení stiskem tlačítka „Změna hesla“ v nabídce „Obsluha“ na nástrojové liště v horní části obrazovky. Zobrazí se dialog, do něhož je potřeba zadat staré i nové heslo a potvrdit jej.

Zahájení a ukončení dne

K zahájení a ukončení dne (koncept dne je vysvětlen v části Definice pojmů na straně 6) slouží odpovídající tlačítka v nabídce „Dny“ na nástrojové liště v horní části obrazovky. Před zahájením každého dne je vždy potřeba nejdříve zkontrolovat obsah kasy. Pokud částka v kase nesouhlasí s evidovanou částkou, je rozdíl přidán do kasy jako spropitné.

Registrace zákaznického účtu

Registrace zákaznického účtu se provádí stiskem tlačítka „Zaregistrovat zákazníka“ v nabídce „Nástroje“ na nástrojové liště v horní části obrazovky. Každý nově vytvořený zákaznický účet má při použití výchozího nastavení nárok na slevu 10%. Tuto hodnotu je možné editovat prostřednictvím administračního rozhraní.

5.4 Administrační rozhraní

Hlavní část administračního rozhraní je tvořena tabulkou obsahující data systému[3]. Na horním okraji obrazovky se nachází lišta umožňující přihlášení obsluhy

[3]Podobu administračního rozhraní je možné vidět na obrázku C.5

(nabídka „Obsluha“, funguje stejně jako v uživatelském rozhraní, více informací v sekci 5.3.4) a výběr zobrazené tabulky (nabídky „Přehledy“ a „Nastavení“).

Tabulky v nabídce „Přehledy“ jsou určeny pouze pro čtení, zatímco ty v nabídce „Nastavení“ je možné editovat. Editovatelné tabulky obsahují ve spodní části obrazovky tlačítko „Přidat“ pro vložení nového záznamu. Editace existujících záznamů se provede dvojitým kliknutím levým tlačítkem myši na položku, kterou chce uživatel editovat. Zobrazí se dialog obsahující informace o vybrané položce, které je možné upravit. Potvrzením tohoto dialogu budou upravená data uložena do systému.

5.4.1 Přehledy

V přehledech jsou vidět různé informace o prodaném zboží v minulých dnech a manipulaci s hotovostí v kase. Tyto tabulky není možné editovat.

5.4.2 Nastavení

Zboží

V této tabulce jsou vidět jednotlivé typy zboží uložené v systému. Při úpravě nějakého typu zboží se informace o předchozí verzi nepřepíšou, místo toho se vytvoří nová verze. Tedy sloupec „Verze“ říká, kolikrát bylo dané zboží upravováno. Zboží není možné mazat, ale je možné podobného efektu dosáhnout deaktivací zboží. Zboží, které není aktivní, se nezobrazuje v seznamu zboží v uživatelském rozhraní a není možné jej objednávat (neaktivní položky jsou v této tabulce zvýrazněny červenou barvou).

Sloupec „DPH“ určuje sazbu daně z přidané hodnoty. Podniky, které nejsou plátci DPH, by měly tento sloupec nastavit na 0%.

„Věkový limit“ slouží pouze pro možnost filtrování zboží v potenciálním programu, který by vytvářel výpisy. Na chování žádné části systému implementované v rámci této práce nemá žádný zvláštní vliv.

Hodnota „Editovatelné“ určuje, zda je možné měnit název a cenu zboží při vytváření platby. To není třeba u většiny zboží, ale může se hodit v některých výjimečných případech.

Hodnota „Sleva možná“ určuje, zda se na tento typ zboží vztahují zákaznické slevy.

Zákazníci

V této tabulce je možné přiřadit jednotlivým zákaznickým kódům různé slevy.

Stoly

V této tabulce je možné nastavit názvy, pozice a rozměry stolů.

Uživatelé a role

V těchto tabulkách je vidět seznam uživatelů a seznam rolí. S každým uživatelem mohou být asociované různé role a s každou rolí je pak asociovaná množina oprávnění. Oprávnění odpovídají jednotlivým vzdáleným voláním na serveru. Uživatel

má povolení použít určité vzdálené volání pouze v případě, že je asociován s nějakou rolí, která je asociována s tímto oprávněním. Jinak pokus o provedení volání skončí chybou.

Uživatelské účty je možné zamknout. Na zamknutý uživatelský účet se není možné přihlásit.

5.5 Konfigurace

Konfigurace určitých vlastností systému, které nejsou uložené na serveru (například proto, že je potřeba je znát ještě před připojením k serveru), může být provedena prostřednictvím konfiguračního souboru `easypub.conf`. Tento soubor se nachází ve složce instalace systému.

Každá řádka tohoto souboru je buď prázdná řádka, komentář (pokud začíná znakem `#`) nebo přiřazení konfigurační proměnné. Řádek obsahující přiřazení konfigurační proměnné se skládá ze tří částí: názvu proměnné, dvojtečky a obsahu proměnné. Mezery a tabulátory na začátku a konci řádky a mezi těmito třemi částmi řádky se ignorují.

Systém rozpoznává následující konfigurační proměnné:

- `SERVER_ADDRESS`, `SERVER_PORT`

  Určují adresu a port, na který se mají připojit uživatelská a administrační rozhraní systému, když chtějí komunikovat se serverem. Server používá pouze proměnnou `SERVER_PORT`, podle níž se rozhoduje, na kterém portu bude očekávat příchozí spojení. Výchozí hodnoty jsou `localhost` a `38139`.

- `LOG_LEVEL`

  Určuje, na jaké úrovni bude systém logovat zprávy. Přípustné hodnoty jsou `trace`, `debug`, `info`, `warning`, `error` a `fatal` (pořadí od nejméně závažné úrovně po nejzávažnější). Logování na vybrané úrovni úrovni zahrnuje i výpis zpráv na všech závažnějších úrovních. Výchozí hodnota je `info`.

- `MAIN_FRAME_TITLE`

  Titulek hlavního okna uživatelské části systému.

  Výchozí hodnota je `EasyPub`.

- `ASK_FOR_CUSTOMER_CODES`

  Může být nastavena na `yes` nebo `no`. Je-li nastavena na `yes`, systém se při každém vytváření platby automaticky zeptá na zákaznický kód. Zákaznické kódy je možné použít, i když je tato proměnná nastavena na `no`, ale je potom před zadáním kódu potřeba stisknout příslušné tlačítko na dialogu s informacemi o platbě. Výchozí hodnota je `no`.

- `RECEIPT_WIDTH`

  Šířka v pixelech, která má být použita při vykreslování účtenek.

- `RECEIPT_FONT`, `RECEIPT_FONT_SIZE`

  Písmo a jeho velikost, které má být použité při vykreslování účtenek.

- `RECEIPT_HEADER`, `RECEIPT_FOOTER`

  Umožňuje měnit obsah úvodní a závěrečné části každé účtenky. Do těchto proměnných je možné zapisovat opakovaně, dané elementy se na účtence potom zobrazí v tom pořadí, v jakém jsou uvedené v konfiguračním souboru. Mohou sloužit například k přidání informací o podniku na účtenku.

  Přípustné hodnoty mají jednu z následujících podob (špičaté závorky vyhrazují části, které je možné měnit, a nejsou součástí syntaxe):

  - `TEXT <text1>|<text2>|<text3>`

    Zadané texty se zobrazí v jednom řádku na účtence. `<text1>` bude zarovnán vlevo, `<text2>` doprostřed a `<text3>` vpravo.

  - `LINE`

    Na účtenku se vykreslí horizontální proužek.

  - `PNG <soubor.png>`

    Zadaný obrázek ve formátu PNG[4] se zobrazí na účtence.

  - `VSPACE <vyska>`

    Na účtence vznikne vertikální odsazení, jehož velikost v pixelech je určena parametrem `<vyska>`.

[4] *PNG* – Portable Network Graphics. Rastrový grafický formát podporující bezeztrátovou kompresi.

6. Závěr

V rámci práce byly zanalyzovány požadavky zákazníka na restaurační software, proveden průzkum existujících řešení a navržen a implementován systém EasyPub. Implementovaný systém podporuje podstatnou část funkcionality existujících systémů, na rozdíl od nich ovšem poskytuje vývojářům rozsáhlé možnosti rozšiřování systému a snadnou propojitelnost s jinými systémy.

Systém splňuje všechny požadavky na něj kladené, které jsou uvedeny ve specifikaci zadání. Také je již používán v provozu u zákazníka, podle jehož požadavků vznikl. Je tedy provozuschopný, ačkoliv samozřejmě stále existují možná vylepšení, z nichž mnohá jsou nutná k tomu, aby byl systém plnohodnotným konkurentem komerčně dostupných řešení.

Serverová část systému je plně funkční a jde o dobrý stavební kámen vhodný pro další rozšiřování. Klientská část má určité problémy, především co se týče rozhraní s hardwarem. Zvlášť palčivým nedostatkem je chybějící víceplatformní podpora tisku účtenek. Tento nedostatek bude potřeba v budoucnu napravit, aby byl systém konkurenceschopný, neboť je pravděpodobné, že většina potenciálních uživatelů jej bude chtít provozovat na operačním systému Windows, kde tisk zatím není systémem podporován.

Také by bylo vhodné zjednodušit a zpřehlednit konfiguraci systému, která je v současnosti poněkud těžkopádná.

I přes tyto nedostatky byla implementace systému EasyPub přínosná, systém ve své současné podobě je užitečný a jeho další vývoj a především vývoj jeho modulů může poskytnout zajímavá a inovativní softwarová řešení potenciálně prospěšná mnoha restauracím a podobným podnikům.

Seznam použité literatury

[1] Halfond, W. – Viegas, J. – Orso, A. A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering, Arlington, VA, USA*, s. 13–15, 2006.

[2] Kaliski, B. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898 (Informational), September 2000. Dostupné z: <http://www.ietf.org/rfc/rfc2898.txt>.

[3] Slee, M. – Agarwal, A. – Kwiatkowski, M. Thrift: Scalable Cross-Language Services Implementation. Dostupné z: <https://thrift.apache.org/static/files/thrift-20070401.pdf>.

[4] Turan, M. S. et al. Recommendation for Password-Based Key Derivation. *NIST Special Publication*. 2010, 800, s. 132.

Seznam použitých zkratek

- *ACID* – Atomicity, Consistency, Isolation, Durability. Vlastnosti databázových transakcí zaručující jejich spolehlivost.
- *ER* – Entity-relationship. ER diagramy jsou grafy znázorňující vztahy mezi třídami objektů.
- *GCC* – GNU Compiler Collection. Kolekce softwarových nástrojů umožňující mimo jiné kompilaci C a C++ programů.
- *HMAC* – Hash-based message authentication code. Algoritmus sloužící k autentizaci zpráv s použitím sdíleného klíče.
- *JSON* – JavaScript Object Notation. Datový formát vhodný pro meziplatformní sdílení dat.
- *PBKDF2* – Password-Based Key Derivation Function 2. Algoritmus umožňující vygenerovat kryptografický klíč z hesla.
- *PNG* – Portable Network Graphics. Rastrový grafický formát podporující bezeztrátovou kompresi.
- *RPC* – Remote procedure call. Způsob meziprocesové komunikace, kde volané metody se mohou vykonávat v jiném než volajícím procesu.
- *SHA* – Secure Hash Algorithm. Hašovací funkce sloužící k vytvoření otisku dat.
- *SOAP* – Simple Object Access Protocol. Protokol meziplatformní výměny dat založený na standardu XML.
- *SQL* – Structured Query Language. Jazyk určený pro práci s daty v relačních databázích.
- *TCP/IP* – Transmission Control Protocol/Internet Protocol. Rodina protokolů síťové komunikace.

A. Porovnání existujících systémů pro restaurace

Systémy byly porovnávány na základě kategorií uvedených v následující tabulce. Informace pocházejí z webových stránek výrobců. V případech, kde to bylo možné, byla stažena a vyzkoušena demoverze programu. V některých případech jsou kolonky označeny otazníkem, což znamená, že se nepodařilo najít potvrzení, že program danou funkcionalitu podporuje, ale kvůli nedostatku informací není možné prokázat opak.

Pokud existuje více verzí jednoho systému, jsou v tabulce uvedeny funkce podporované nejpokročilejší verzí. Uvedený cenový rozsah je potom rozsah cen jednotlivých verzí, tedy k dosažení plné funkcionality je potřeba brát v úvahu horní hranici rozsahu.

Seznam porovnávaného softwaru spolu s odkazy na webové stránky výrobců:

- Agnis Pokladna
  `http://www.agnis.cz/2_cestina/17_programy/3_texty/20_pokladni-systemy,-registracni-pokladna/`
- AWIS GASTRO
  `http://www.kasa-pokladna.cz/en/pokladni-systemy-pro-restaurace-kluby-bary`
- Conto
  `http://www.money.cz/pos/pokladni-software/conto/`
- DeCe RESTAURACE
  `http://www.dece.cz/dece-software-menu/dece-restaurace`
- FINTA
  `http://www.finta.cz/restauracni-system/`
- Gastro-Easy
  `http://www.micros.cz/produkty/pokladni-systemy/gastro-easy.html`
- StaeGurmán G4
  `http://gurman.parent.cz/`
- HARSYS 6
  `http://www.ab-x.cz/42-software_pro_restaurace_-_srovnani_verzi_programu`
- Jazz Restaurant
  `http://www.jazzware.cz/restaurant.html`
- KelEXPRESS
  `http://www.keloc-software.cz/produkty/kelexpress/balicky/restaurace/`

- Menu55
  `http://www.menu55.cz/`
- POS Expert
  `http://www.posexperts.cz/Produkty/Restaurace.aspx`
- POS NET
  `http://www.supportsystem.cz/index.php/produkty/pos-net`
- TRELL Restaurace
  `http://www.trell.cz/restaur.htm`

Na posledním řádku je v tabulce navíc možné najít porovnání se systémem EasyPub, který byl vytvořený v rámci této práce.

Software	Správa sortimentu a objednávek	Síťové propojení více počítačů	Správa skladu	Ovládání z mobilního zařízení	Podpora dotykové obrazovky	Správa zákaznických účtů	Konfigurovatelná grafická mapa stolů	Cena (s DPH)
Agnis Pokladna	✓	?	✓	✓	✓	✓	✓	9 075 Kč
AWIS GASTRO	✓	✓	✓	✓	✓	✓	✓	19 239 Kč
Conto	✓	✓	✓	✓	✓	✓	✓	8 349–13 189 Kč
DeCe RESTAURACE	✓	✓	✓	✓	✓	✗	✗	3 035–14 399 Kč[1]
FINTA	✓	?	✓	✓	✓	✓	✓	na požádání
Gastro-Easy	✓	?	✓	?	✓	✓	?	?
StarGurmán G4	✓	✓	✓	✗	✓	✓	✗	6 038–9 656 Kč
HARSYS 6	✓	✓	✓	✓	✓	✓	✓	5 929–22 966 Kč[2]
Jazz Restaurant	✓	✓	✓	✓	✓	✓	✓	10 890–24 503 Kč[3]
KelEXPRESS	✓	?	✓	?	✓	?	✓	109–363 Kč měsíčně
Menu55	✓	✓	?	✓	✓	?	?	424–581 Kč měsíčně
POS Expert	✓	✓	✓	✓	✓	✓	✓	12 088–25 398 Kč
POS NET	✓	✓	✓	✓	✓	✓	✓	9 559 Kč
TRELL Restaurace	✓	✓	✓	✗	✓	✓	✓[4]	4 828–14 520 Kč
EasyPub	✓	✓	✗[5]	✗[6]	✓[7]	✓	✓	- - -

[1]Modul pro sledování stavu skladu se prodává zvlášť, stojí 7 260–9 680 Kč.

[2]Cena verze pro mobilní zařízení na požádání.

[3]Modul pro podporu mobilních zařízení se prodává zvlášť a stojí 3 630–5 445 Kč.

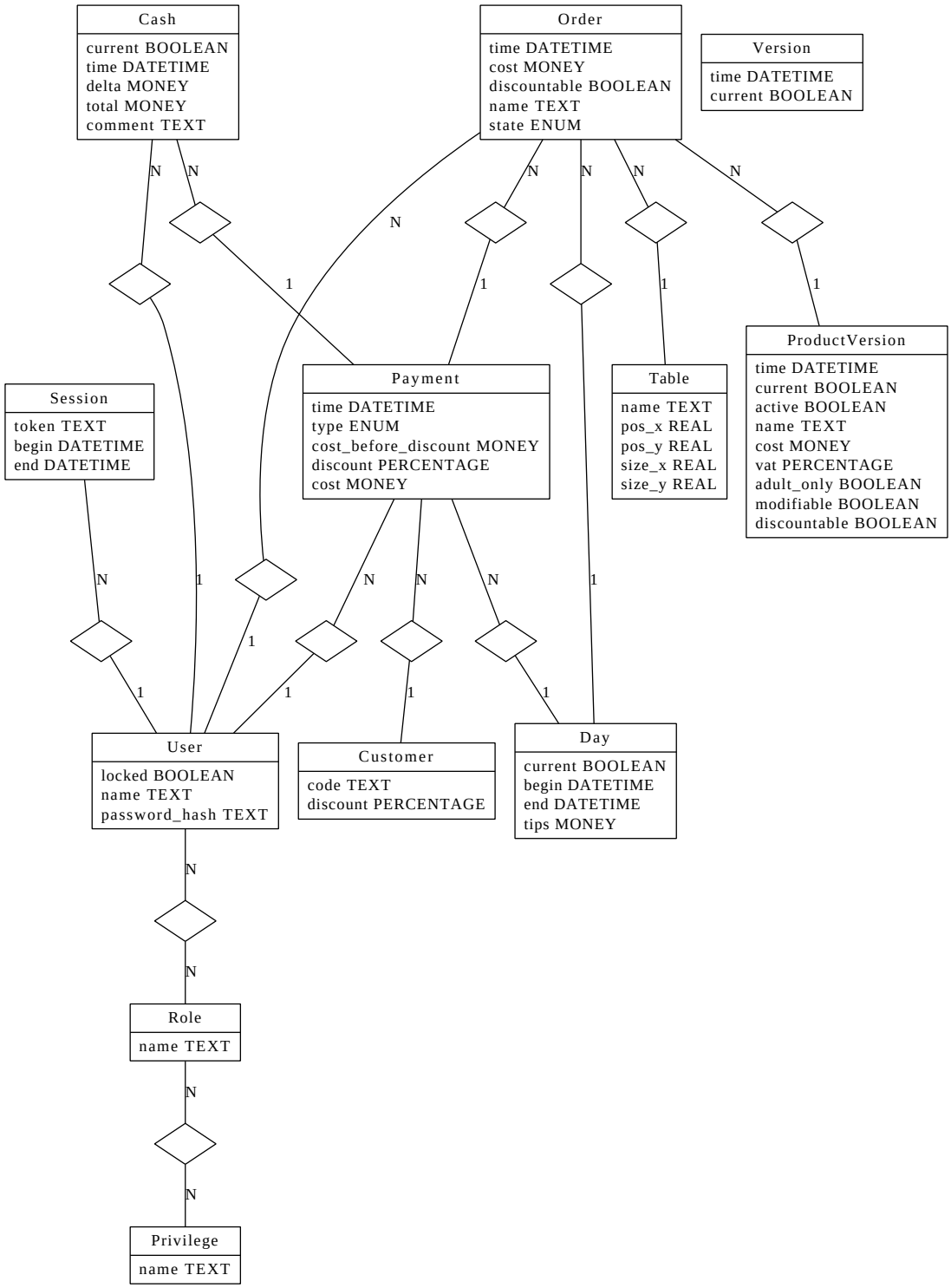
[4]Pozice stolů jsou přednastavené a je možné si z nich pouze vybírat.

[5]EasyPub sleduje pouze počet prodaných kusů zboží.

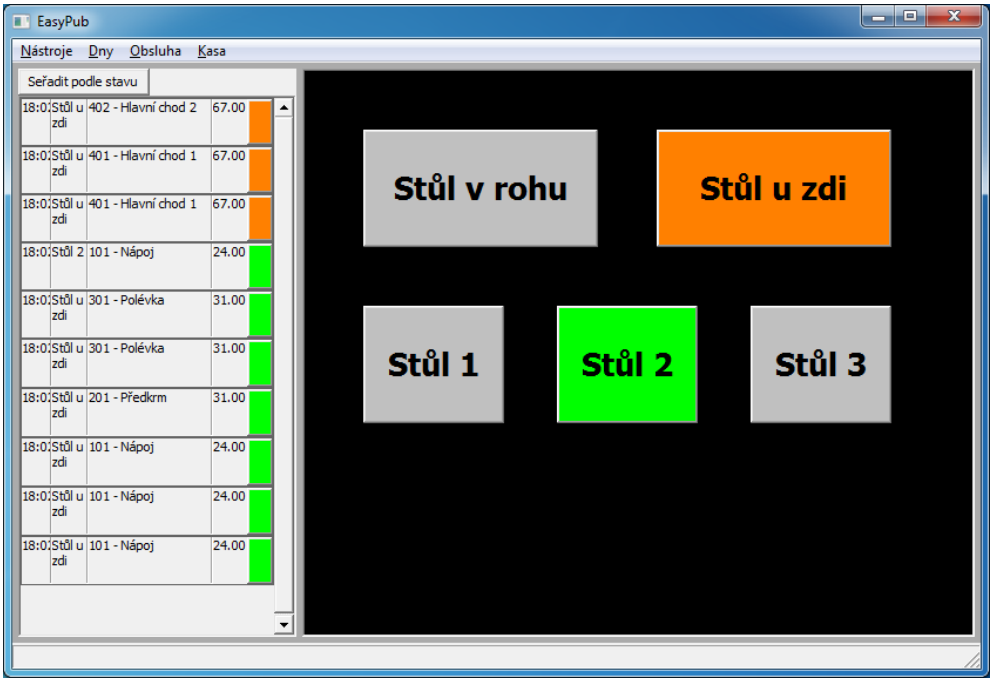
[6]Aplikaci pro mobilní zařízení by bylo možné implementovat beze změn kódu serveru.

[7]Události z dotykové obrazovky jsou většinou systémem překládány na ekvivalentní události myši. EasyPub používá pouze tyto události myši (tedy například nepodporuje žádná dotyková gesta více prsty). Zdá se, že všechny ostatní vyzkoušené systémy fungují stejným způsobem.

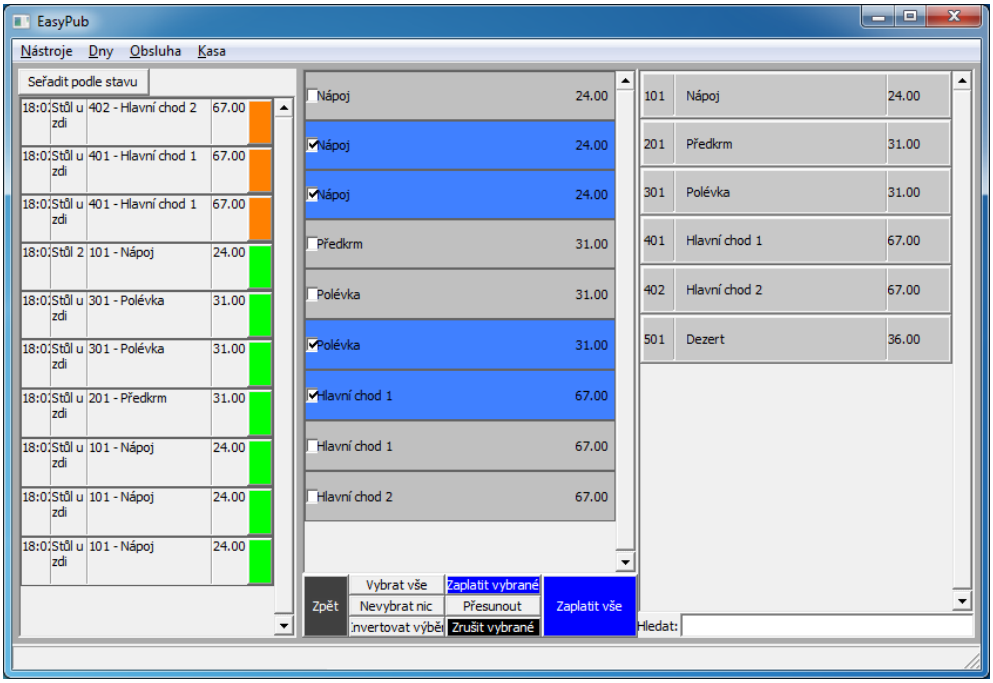
B. ER diagram databáze



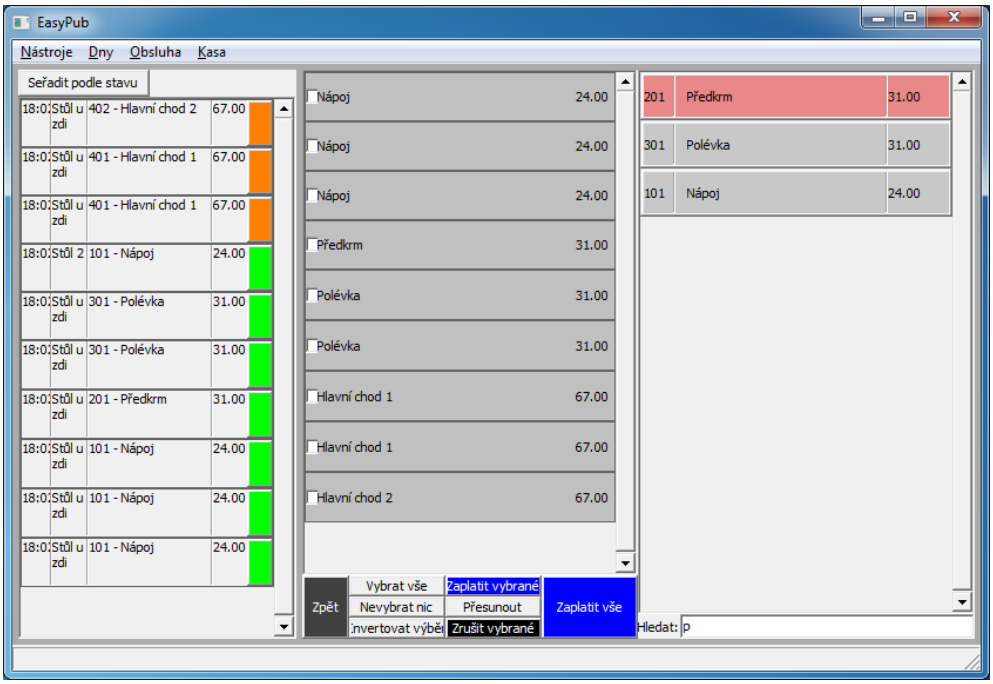
C. Snímky uživatelského rozhraní



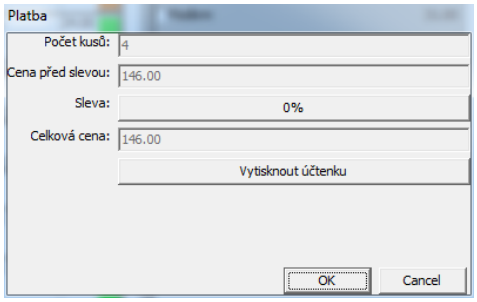
Obrázek C.1: Hlavní okno uživatelského rozhraní. V levé části je vidět seznam objednávek z aktuálního dne, v pravé části grafický přehled stolů. Barvy u objednávek a na tlačítkách stolů reprezentují stav objednávek.



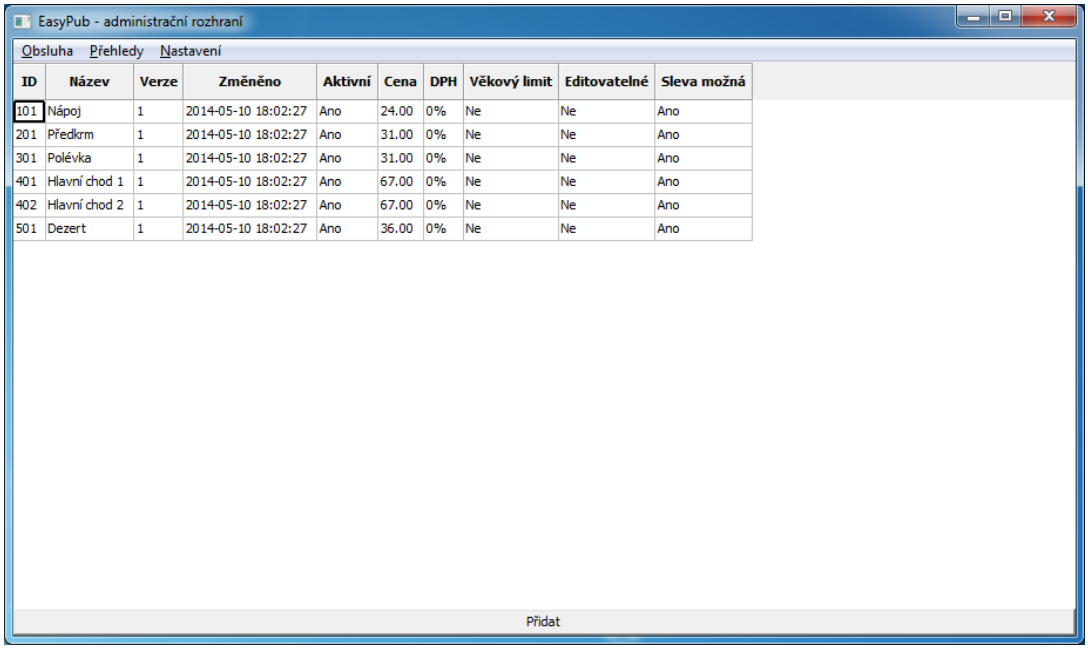
Obrázek C.2: Detail stolu v uživatelském rozhraní. V levé části je stejný seznam objednávek, který je vidět i v hlavním okně, v prostřední části je seznam aktivních objednávek u vybraného stolu, z nichž některé jsou vybrané, v pravé části je seznam zboží umožňující vytváření nových objednávek.



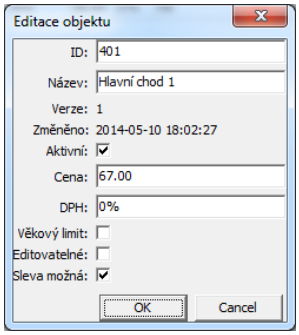
Obrázek C.3: Stejný pohled jako na obrázku C.2. Zde je navíc předvedena funkce vyhledávání zboží. V seznamu zboží jsou zobrazeny pouze ty položky, jejichž název obsahuje text zadaný do pole pod seznamem.



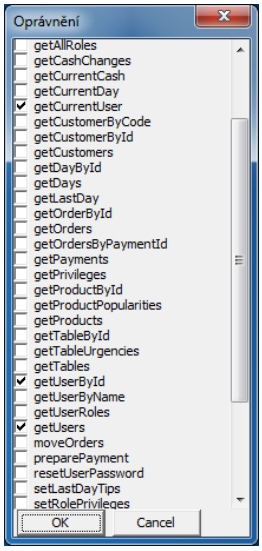
Obrázek C.4: Dialog uživatelského rozhraní pro vytvoření a potvrzení platby.



Obrázek C.5: Hlavní okno administračního rozhraní. Toto rozhraní umožňuje prohlížet a upravovat různá data a nastavení systému. Aktuálně zobrazená tabulka obsahuje informace o typech zboží uložených v systému.



Obrázek C.6: Dialog administračního rozhraní umožňující editovat vybraný typ zboží.



Obrázek C.7: Dialog administračního rozhraní umožňující přiřadit oprávnění vybrané uživatelské roli.