Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Vývoj internetových aplikací typu klient-server
s využitím moderních přístupů

Plzeň, 2013 **Roman GERHÁT**

P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne

.....

vlastnoruční podpis

Poděkování

Tímto bych rád poděkoval Ing. Pavlu Baldovi, Ph. D. za odborné vedení této diplomové práce.

Anotace

Tato diplomová práce se zabývá analýzou, návrhem a vývojem internetových aplikací typu klient-server s využitím moderních technologií a přístupů. Nejprve jsou popsány různé moderní způsoby vytváření těchto aplikací a jejich vlastnosti. Následuje analýza celkového systému, nástin jeho funkčnosti, návrh architektury programu a rozbor jednotlivých částí s popisem všech technologií použitých pro tvorbu tohoto systému.

Tato práce se také věnuje praktickému užití této aplikace, její vyzkoušení na reálném příkladu víceúčelové komunikace a správy dat samovýčepních restaurací, kde je takto vytvořená aplikace použita.

Klíčová slova: klient, server, klient-server, internet, správa dat, komunikace, xml, html5, CSS3, jquery, SSL, python, twisted, postgresql

Annotation

This thesis concerns the analysis, design and development of the internet client–server applications using modern techniques and approaches. Firstly, the modern ways of creating these applications and their characteristics are derived. This is followed by analysis of the whole system, outline of its functionality, design of the program architecture and delination of individual components with the detailed description of all technologies used for creation of this system.

This work also describes the practical use of this application, its testing on the real example of multipurpose communication and data management of self-serving beer restaurants, where the program with this designed structure is being used.

Key words: client, server, client-server, internet, data management, communication, xml, html5, CSS3, jquery, SSL, python, twisted, postgresql

Obsah

Úvod

Cílem této práce je návrh vhodné multiplatformní architektury aplikace typu klient—server použitím moderních technik a technologií a následná implementace takto vytvořené architektury v konkrétní aplikaci pro internetovou komunikaci samovýčepních restaurací. Tato aplikace sestává z rozšiřitelné serverové části, která zprostředkuje přenesená a filtrovaná data jednotlivých klientů, a klientské části, která je určena k vizualizaci lokálně získaných dat a také dat přenesených ze serveru. Návrh této aplikace by měl být natolik univerzální, aby mohl být aplikován i na podobné úlohy zabývající se internetovým přenosem a zobrazením dat.

Důvodem zadání této práce je aktualizace a modernizace technického i programového vybavení samovýčepních restaurací, dodávaných firmou MCAT Automation s.r.o., a zlepšení vizualizačních možností, které slouží k zatraktivnění návštěvy restaurace zákazníkem a informativním, propagačním či jiným komerčním účelům. Cílem je implementovat jednoduše aktualizovatelnou a rozšiřitelnou aplikaci, která může být konfigurovatelná podle budoucích žádostí a potřeby. Z tohoto důvodu jsou zohledněny a zkoumány moderní přístupy k problematice aplikací klient—server, které by zajistily tyto funkce i stálost softwarové podpory v dalších letech.

Klientská část aplikace by měla fungovat na nejvíce používaných operačních systémech a zajišťovat zobrazení lokálních i vzdálených dat v reálném čase. Lokální data mohou být shromažďována z programovatelného logického automatu nebo osobního počítače, vzdálená data musí být získávána ze serverové části aplikace.

Vizualizaci takto získaných dat musí být možné zobrazit na libovolné připojené obrazovce počítače (resp. projektoru, televizi, apod.). Současně se také tato vizualizace musí skládat z různých vizualizačních módů, které jsou dvojího charakteru – informativního s použitím získaných dat a propagačního s použitím reklamních obrázků a videí. Informativní vizualizace budou programovány ke svému specifickému účelu (např. sloupcové zobrazení restaurací s nejvyšší výtočí), ale bude u nich možné uživatelsky měnit grafické prvky. Propagační vizualizace je nutné měnit podle aktuálních potřeb restaurace, tedy jediným požadavkem je schopnost zobrazení obrázku a videa.

Úvodní část této práce obsahuje seznámení s použitými základními termíny, následuje popis architektury klient—server a moderní přístupy k této problematice. Další kapitola se zabývá analýzou požadavků k návrhu žádané aplikace a porovnává technologie k jejich docílení. Dále se tato práce zabývá návrhem architektury a dílčích komponent s použitím vybraných nástrojů. Obsahem poslední kapitoly je prezentace konkrétní aplikace, určené k nasazení do samovýčepních restaurací, vytvořené dle obecného návrhu.

1 Moderní aplikace typu klient-server

Tato kapitola podrobně popisuje možné moderní přístupy ke tvorbě aplikací typu server–klient, jejich předpoklady, použití a vlastnosti, ale také nástroje nezbytné k jejich zprovoznění. Zabývá se také objasněním základní terminologie užívané v souvislosti s těmito aplikacemi a elementárním popisem tříd klient-server.

1.1 Základní pojmy

Zde jsou vysvětleny základní pojmy nezbytné k pochopení této práce a principu aplikace, která je zde následně navržena.

1.1.1 Server

Nejběžněji je termínem server označován počítač nebo část počítačového hardwaru určená k běhu jedné či více aplikací – služeb. Tyto aplikace slouží jednotlivým uživatelům k nejrůznějším účelům. Tyto účely se rozvíjely spolu s internetem a v dnešní době se používají například tyto typy serverů [1]:

- Aplikační server – výpočetní nástavba mezi klienta (většinou prohlížeč) a databázový server
- Audio/Video server – zajišťují přístup k multimédiím v reálném čase
- Souborový server – nejčastěji FTP – nejstarší internetová služba, k bezpečnému přenosu souborů
- Komunikační server – Chat, IRC, Mail, News, Fax servery
- Groupware server – pro společnou práci více klientů ve stejném čase
- Telnet server – ke vzdálenému připojení k počítači
- Webový server – k zobrazení internetových stránek pomocí webového prohlížeče

Kromě těchto používaných typů existuje ještě množství jiných serverů, často přizpůsobených pro konkrétní službu, kterou má server obsluhovat [2].

Nehledě na typ služby má ale každý server dvě možné varianty:

- dedikovaný – slouží výhradně jako server pro specifické účely
- nededikovaný – kromě speciálních služeb slouží i jako klasický počítač

V případě modelu klient/server je serverem myšlena služba (program) obsluhující požadavky klientů – poskytující vyžádaná data, provádějící výpočty, úkony s databází, do které může přistupovat více klientů najednou, apod. Všechny tyto služby musí být dostupné všem klientům v rámci počítače nebo v rámci sítě, a to nepřetržitě – pro správnou funkčnost musí obsluhovat žádosti klientů 24 hodin denně, 7 dní v týdnu [3][2].

1.1.2 Klient

Klientem se stává každý systém nebo software, který začne přistupovat k serveru a odesílat mu požadavky. K serveru přistupuje většinou prostřednictvím sítě, i když může být umístěn i na stejném počítači. Po odeslání požadavku klient vyčkává na odpověď serveru, kterou následně v nějaké formě zobrazuje [4][3][5].

Běžně se klienti rozdělují na:

- Tenkého klienta
- Tlustého (obsáhlého) klienta
- Hybridního klienta
- Chytrý klient

Více informací o jednotlivých typech klientů je obsahem následujících částí kapitoly.

1.1.3 Tenký klient

Tenkým klientem je program, popřípadně počítač, který spoléhá výhradně na informace a výpočty provedené serverem. Tento typ klienta je určen jen k samotnému zobrazení obdržených dat.

Tenký klient může být [6][7]:

- hardwarový – zařízení (terminál), které je závislé na síťovém připojení k serveru; nepříliš rozšířená varianta
- softwarový – aplikace spouštěná v rámci operačního systému, závislá na síťové konektivitě; velice rozšířená varianta

Nejrozšířenějším tenkým klientem dnešní doby je internetový prohlížeč. Ten komunikuje se serverem pomocí bezestavového protokolu HTTP a není v něm prováděna žádná rozhodovací logika ani ukládání dat, pouze samotné zobrazování dat [6].

Tenký klient takového typu má pak mnoho předností, například[5] [8]:

- nízká hardwarová náročnost
- snadné aktualizace – pouze na straně serveru
- přenositelnost a multiplatformové využití
- snadné nasazení

Nevýhodou je například strohé uživatelské rozhraní a nemožnost ovlivňovat, jaký způsobem se budou přenesená data zobrazovat. Rychlost odezvy je obecně nižší než u jiných typů klientů a tenký klient také není schopen pracovat bez síťového připojení k serveru [5].

1.1.4 Tlustý klient

Na rozdíl od tenkého klienta obsahuje tlustý (obsáhlý) klient i část provádějící nezbytné výpočty a ze serveru dostává jen základní data, často je také zasílá zpět na server, ovšem kromě občasné výměny dat nemusí na server samotný spoléhat vůbec – většinou potřebuje alespoň cyklické připojení k serveru, ale je také schopen provádět spoustu úkonů i bez síťové konektivity [9][5].

Aktualizace neprobíhají na straně serveru jako u tenkého klienta, ale musí být provedeny v klientu samotném. K aktualizacím se tedy používají dva přístupy [5]:

- Push – server tlačí nové verze klientovi
- Pull – klient si nové verze sám stahuje

Mezi výhody tlustého klienta patří [9][5][2]:

- Možnost navrhnout vlastní uživatelské rozhraní, na míru aplikaci
- Možnost pracovat offline (bez síťového připojení)
- Snazší vývoj
- Klient má možnost ovlivnit zobrazování dat

Nevýhodami jsou pak vyšší hardwarová náročnost klienta, obtížnější aktualizace, klient musí být vyvíjen na míru různým operačním systémům (hůře získávaná multiplatformovost) a také obtížnější instalace a konfigurace, musí být provedena pro každého klienta zvlášť [5][8].

1.1.5 Hybridní klient

Hybridní klient je kombinací tenkého a tlustého klienta. Stejně jako tlustý klient zpracovává určitá data sám, ale stejně jako tenký klient nepoužívá lokální úložiště pro ukládání dat a software, stejně tak jako data je získáván ze serveru, tedy umožňuje centrální správu. Běh samotného programu je pak na klientském počítači, kde může používat jak lokální hardware, tak síťové připojení k výpočetnímu výkonu serveru [10][5].

Speciálním případem hybridního klienta je pak chytrý klient. To je aplikace, kterou poskytuje server stejně jako v případě tenkého klienta, ovšem ta je následně spuštěna na klientském počítači, kde může používat lokální hardware a využívat síťové připojení jen občas. Z těchto vlastností pak plynou následující výhody [10][5][7]:

- Vlastnosti tenkého klienta – aktuální stažená verze, snadné nasazení, multiplatformovost
- Vlastnosti tlustého klienta – možný offline mód, uživatelské rozhraní na míru, použití lokálního hardwaru, rychlejší odezvy

1.1.6 Síť

Počítačová síť je soubor dílčích hardwarových komponent a počítačů propojených komunikačními kanály, které umožňují sdílení a výměnu informací mezi jednotlivými počítači. Už dva počítače splňující tyto podmínky můžeme označit za počítačovou síť [11].

Počítačové sítě se dají rozlišit podle mnoha faktorů, například podle [11]:

- komunikačního protokolu
- topologie
- rozlehlosti
- účelu
- postavení uzlů – Klient/Server, Peer–to–Peer

V tomto textu se termín síť (počítačová síť) užívá ve významu blíže nespecifikovaného množství počítačů a směrovačů zajišťujících spojení mezi klientem a serverem [8] .

1.1.7 Soket

Sokety jsou koncové body meziprocesních komunikací napříč počítačovou sítí. Většina komunikací dnes je na bázi protokolu Internet Protocol (IP), tedy většina soketů jsou tzv. internetové sokety. Pro navázání spojení si musí každý proces, který chce komunikovat, vytvořit a nastavit soket tak, aby byl schopen vysílat a přijímat informace k a od jiného soketu na jiném počítači [12].

Sokety jsou nejpoužívanější TCP/IP rozhraní (API – Application Programming Interface – rozhraní pro programování aplikací), proto má pro vytváření soketů a práci s nimi většina programovacích jazyků vlastní metody a knihovny [13].

Pro adresování soketu se používá kombinace IP adresy a čísla portu [11].

1.1.8 Zabezpečené spojení

Spojení dvou počítačů na síti je zabezpečené, jestliže nepřenáší data v podobě prostého textu, ale jsou–li tato data nějakým stylem šifrována, aby byla zabezpečena proti odposlouchání či podvržení případným třetím počítačem v síti. Často je součástí zabezpečeného připojení taktéž schopnost ověřit identitu

Oba počítače musí být předem seznámeni se šifrovacím protokolem, aby byly schopni tuto komunikaci vést [14].

Nejpoužívanějšími protokoly dnes jsou:

- SSL – Secure Sockets Layer (Vrstva bezpečných soketů)
- TLS – Transport Layer Security (Zabezpečení transportní vrstvy)

1.1.9 Databáze

Databáze je logicky ucelený soubor smysluplných dat, příslušných k nějakým prvkům reálného světa [15]. V dnešní době a obzvlášť v tomto kontextu se jedná o digitalizovanou podobu takovéhoto souboru dat, uložených na nějakém paměťovém médiu. Často je tak ovšem nazýván i software, který je schopen s těmito daty dále pracovat – číst, editovat, zapisovat; ten se ovšem oficiálně nazývá Systém řízení báze dat (Database management system). Tento systém je ovšem schopen nejen změny položek, ale také například [15]:

- řídit bezpečnost
- zachovávat integritu dat
- poskytnout možnost pro zálohu a obnovu
- řidit nadbytečnost
- povolit nezávislost dat
- automaticky optimalizovat dotazování na prvky

Databázových systémů je poměrně velké množství zařazených pod nejrůznější licence, ale nejpoužívanějšími jsou v dnešní době [16]:

- MySQL – často používaná spolu s PHP, populární pro webové aplikace, použitá i pro velké webové aplikace – např. Wikipedia, Facebook a Twitter [17]
- Microsoft Office Access – používaná především v malých aplikacích, často použitá s Visual Basic for Applications pro přímou podporu kódu a prvků uživ. rozhraní
- Microsoft SQL Server – pro mnohem větší objem dat než Access, podpora multimédií, několik verzí pro různě strukturované účely
- Oracle – komerční systém řízení báze dat, široká oficiální i neoficiální podpora, pestrá škála nadstandardních funkcí
- PostgreSQL – globální open–source podpora, používána pro středně velké i velké aplikace, flexibilní a jednoduchá konfigurace
- SQLite – malá knihovna jazyka C užívaná v případech, kdy je pro vývojáře jednoduchost administrace a údržby důležitější než komplexní funkce nabízené jinými systémy

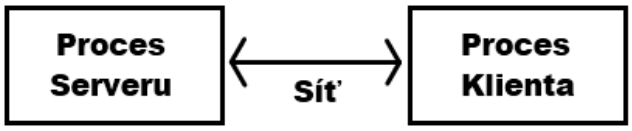
V tomto textu je také často zmíněn SQL (Structured Query Language), což je dotazovací jazyk určený pro práci s daty v relačních databázích [18].

1.2 Aplikace typu klient–server

V této kapitole je popsána funkčnost tohoto sítového komunikačního modelu – základy na kterých pracuje, průběh jeho komunikace, jeho vlastnosti, způsob použití i případné alternativy podobného principu včetně porovnání těchto používaných architektur – jejich předností i nevýhod a nejčastějšího použití na konkrétní aplikace.

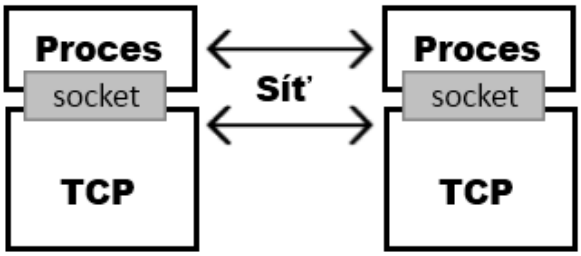
1.2.1 Model klient/server

Model klient/server je založen na distribuci funkcí mezi dvěma typy autonomních procesů – serveru a klienta. Klientem je jakýkoliv proces, který žádá specifické služby od procesu serveru. Server je naopak proces, který tyto vyžádané služby poskytuje klientovi. Oba tyto procesy mohou být jak na stejném počítači, tak na několika počítačích propojených sítí, která funguje jako médium, skrze které mohou tyto procesy komunikovat. Obrázek 1 ukazuje základní výpočetní model klient/server [2]:



Obrázek 1: Základní schéma modelu klient/server

Toto schéma jde ještě dále zpřesnit, protože většina systémů tohoto typu používá sokety. Jediný případ, ve kterém není použití soketů nutné, je pokud oba procesy běží na stejném počítači, ale i v tomto případě je použití soketů vhodná a efektivní varianta. Přesnější schéma systému používajícího sokety je znázorněno na obrázku 2 [13]:



Obrázek 2: Schéma klient/server s použitím soketů

V tomto případě je proces a vytvoření soketu řízen samotnou aplikací, ale samotný přenos dat zajišťuje TCP (Transmission Control Protocol – transportní vrstva internetu), čímž je garantováno spolehlivé doručování ve správném pořadí – TCP se chová jako potrubí (viz obrázek 3) přenášející data, sokety jsou jeho konce [13]:

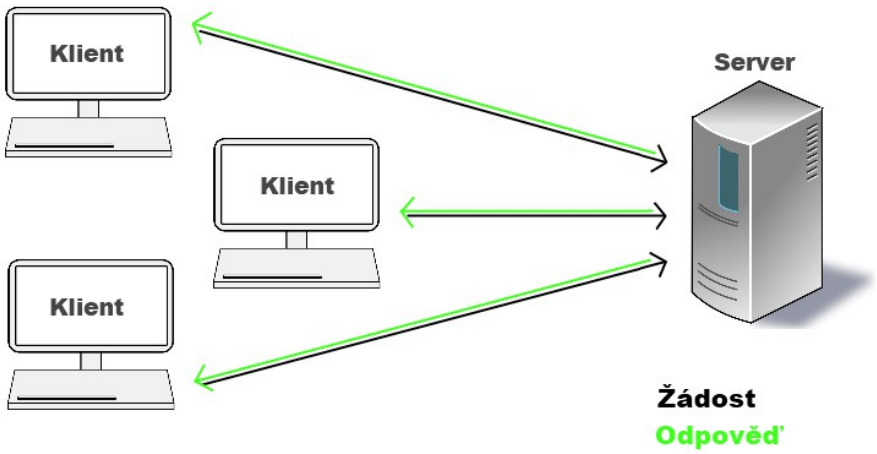
Obrázek 3: Schéma funkce soketu v modelu klient/server

TCP část je navíc řízena samotným operačním systémem, tedy samotná aplikace nemusí řešit nic víc než samotné vytvoření a konfiguraci soketu.

Model klient/server funguje běžně v několika základních variantách [2]:

- jeden klient, jeden server
- více klientů, jeden server
- více klientů, více serverů

Pro každého klienta může sice existovat právě jeden server, ovšem běžně se tyto aplikace vytváří například pro přístup více klientů k jednomu serveru (ke stejným datům), tedy takovéto řešení není obvyklé. Mnohem častější varianta je použití jednoho serveru (viz obrázek 4) pro více klientů [2]:



Obrázek 4: Více klientů připojených k jednomu serveru

Jak už je zmíněno v kapitole 1.1.1, server musí fungovat neustále, právě proto, aby mohl obsloužit kterýkoliv počítač, který se stane klientem, tj. takový počítač, který se připojí k serveru a odešle mu nějaký požadavek [2].

Podle způsobu jakým server přistupuje k jednotlivým žádostem dělíme dále architekturu klient/server podle tohoto aspektu na tzv. bezstavové a stavové servery [2]:

- Bezstavový server – Každou žádost zpracovává jako nezávislou transakci nijak nesouvisející s předchozími žádostmi. Výhodou tohoto přístupu je fakt, že si server nemusí nikde vyhrazovat paměť na předchozí žádosti a vytvořená (případně ukončená) připojení, jen pracuje s žádostmi, které přijdou, a zapomíná je hned po zpracování. Nevýhodou je, že tento postup vyžaduje větší množství informací obsažených v žádosti. Typickým příkladem je kterýkoliv webový server. S výjimkou cookies je každá žádost (URL) plnou specifikací dotazovaného dokumentu a nevyžaduje od serveru žádnou paměť minulých žádostí.
- Stavový server – Stav klienta je průběžně ukládán na serveru a ten si pamatuje co žádal klient předtím. Tento přístup pomáhá zpracovat žádosti efektivněji a žádosti mohou být obsahově menší než u bezstavového serveru. Nevýhodou je velké zahlcení serverové paměti, které může nastat například pokud klient z jakéhokoliv důvodu často rozpojuje připojení. Příkladem stavového serveru jsou například síťové souborové systémy.

Výhody modelu klient/server [2][13]

- Zvýšení výkonu a snížení zátěže programu – Zpracování jednotlivých úkonů je rozděleno mezi klienta a server a na rozdíl od tradiční počítačové databáze není rychlost řízení systému báze dat závislá na nepoužitém výkonu operačního systému, kde běží jako proces na pozadí. Tímto se také snížila zátěž sítě, kde se nemusí posílat celý soubor databáze když je zrovna používán, ale po počítačové síti probíhají jen dotazy a odpovědi na ně. Lépe zpracované servery redukují množství poslaných dat ještě více tím, že se databázové dotazy skládají na straně serveru a od klienta přicházejí jen krátké žádosti na specifická data.
- Nezávislost na typu stanice či OS – Uživatelé nejsou vázáni na žádný konkrétní hardware nebo operační systém, protože stanice nepotřebují žádný konkrétní software pro běh řízení systému báze dat, ke kterému pouze vzdáleně přistupují. Vše je tak závislé pouze na vývojáři, který může navrhnout klientský software nepřeberným množstvím způsobů.
- Rozšiřitelnost – K serveru se mohou připojovat další a další uživatelé, aniž by to nějak ovlivnilo způsob, jakým systém pracuje, nebo aniž by bylo nutné software měnit. Také je možné změnit hardware serveru (například přesun serverového procesu na jinou, výkonnější stanici), aniž by to nějak ovlivnilo způsob, jakým klient přistupuje k datům, tedy pro uživatele se přitom nic nezmění.
- Integrita dat – Přenos dat prostřednictvím soketů zajišťuje doručení dat na server, řízení systému báze dat pak poskytuje mnoho služeb pro správné ukládání, zálohu a kontrolu dat.
- Přístup k datům – Datové úložiště je centralizované, tedy i více uživatelů může přistupovat ke stejným souborům či databázi ve stejném okamžiku, nezávisle na počítači či místa ze kterého chtějí data prohlížet či editovat.
- Snadná administrace – Prostředí modelu klient/server je velice ovladatelné, díky rozdělení systému na jednotlivé části. Úkony jako záloha dat, autentizace uživatelů, kontrola integrity, aktualizace softwaru i hardwaru, vše jde provádět centralizovaně na jednom místě.
- Interoperabilita – Všechny dílčí systémy (klient, server, počítačová síť) spolu bez problémů spolupracují a každý z nich může být bez problému vyměněn, aniž by to narušilo chod celku – při změně či vytvoření nového klienta bude fungovat i ten původní, při změně serveru se nic nezmění pro klienty a dokonce i médium mezi klientem a serverem může být také nahrazeno jiným (lokální síť, metropolitní síť, rozlehlá síť, internet).
- Zpracování dat nezávislé na geografickém umístění – Uživatelé mohou přistupovat k datům odkudkoliv, jakmile mají klientský software, který je schopen navázat spojení se serverem, což je zvlášť v moderním světě přenosných a kapesních počítačů velká výhoda. Nejvýraznějším příkladem by v dnešní době byly webové prohlížeče, které se dokáží spojit s kterýmkoliv webovým serverem odkudkoliv, z jakéhokoliv operačního systému i zařízení – počítače, telefonu i např. televize.
- Redukování nákladů – Namísto vývoje drahých velkých systémů se může vytvořit více menších systémů sesíťovaných dohromady, což navíc dále sníží výdaje při případném rozšiřování. Stejně tak cena hardwaru je nižší, protože pouze server musí být vyloženě výkonný, náklady za klientské stanice jsou o poznání nižší.

Nevýhody modelu klient/server [2][13]

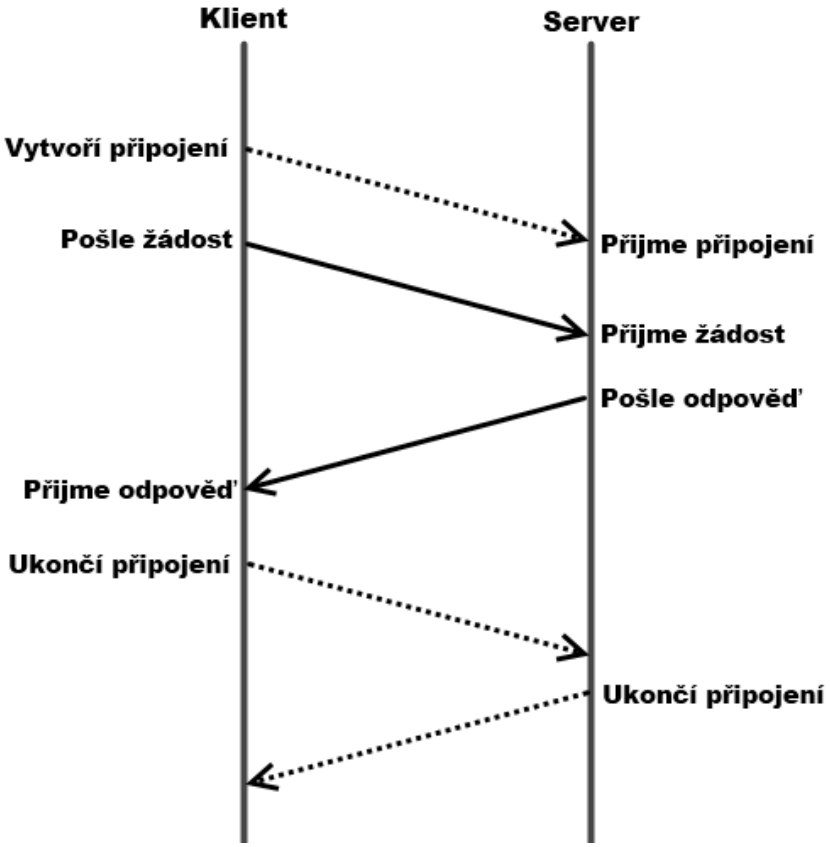
- Přetěžování sítě – Počet požadavků ve stejném okamžiku se zvyšuje s počtem připojených klientů, tedy server se může, zvlášť při větším počtu klientů, snadno přetížit
- Malá robustnost – V případě výpadku serveru nemohou proběhnout odpovědi na žádosti klientů, tedy server neplní svou funkci a není možné jeho funkce nahradit dokud není opět zprovozněn.
- Cena údržby – Je potřebný podpůrný personál pro administraci a údržbu serveru. V menších systémech tuto práci dokáže většinou zastat správce sítě, ovšem u komplexnějších systémů může být údržba a podpra všech klientů časově náročnější.
- Cena serverového hardwaru – V dnešní době není většinou u modelu klient/server předem známo, jaké množství klientů bude muset server obsloužit (příklad – webservery), tedy jsou potřebná co možná nejvýkonnější zařízení. Navíc čím menší aplikace, tím je investice do serveru znatelnější. Při větším množství klientů převýší příjmy ušetřené levnějším klientským hardwarem náklady na pořízení výkonného serveru, ovšem u menší aplikací mohou být náklady serveru vyšší.
- Složitost – Z čím více částí systém sestává, tím větší počet věcí, které se mohou poškodit nebo přestat fungovat. Dříve byl tento problém výraznější kvůli malým zkušenost programátorů a administrátorů s těmito technologiemi, protože v případě poruchy nebylo jednoduché nalézt problém, ale v dnešní době je zaznaménávání a zasílání chybových zpráv standardem díky němuž je i v komplexním systému relativně jednoduché nalézt a opravit problémové místo.

1.2.2 Průběh komunikace

Průběh komunikace [13]	
Klient	**Server**
Inicializace procesu: • vytvoření TCP Soketu • start Uživatelského rozhraní, apod.	Inicializace procesu: • vytvoření TCP Soketu • Navázání soketu na port
	Nastavení soketu na poslouchání • čeká na příchozí žádosti
Vytvoření připojení	Přijetí nového připojení
Komunikace – může opakovat: • posle žádost serveru • přijme odpověď od serveru	Komunikace – může opakovat: • přijme žádost od klienta • pošle odpověď klientovi
Ukončí připojení	Ukončí připojení

Tabulka 1: Průběh komunikace modelu klient–server

Často se interakce mezi klientem a serverem zobrazují pomocí sekvenčních diagramů (viz obrázek 5):



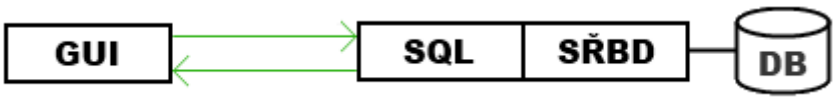
Obrázek 5: Sekvenční diagram interakce klienta a serveru

1.2.3 Varianty

Další dělení modelu klient–server je podle množství použitých zařízení (počítačů nebo částí softwaru) v architektuře systému:

- **Dvouvrstvá architektura** (2–tier Client/Server Model) – Dnes je používána méně. Obsahuje pouze prezenční vrstvu (uživatelské rozhraní – GUI) a datovou vrstvu (systém řízení báze dat – SŘBD, databáze, soubory, přístup k nim – SQL) **[2][13]**.

Obrázek 6: Dvouvrstvá architektura s tlustým klientem



Obrázek 7: Dvouvrstvá architektura s tenkým klientem

Z obrázků 6 a 7 je patrné že ve dvouvrstvé architektuře komunikuje klient přímo se serverem který má požadované služby/informace. Tento model je vhodné použít v malých prostředích (desítky, až sto klientských stanic) pro jeho flexibilitu sdílení dat při minimálních nákladech, jednoduchém vývoji a snadné údržbě aplikace.

- **Vícevrstvá architektura** (N–tier Client/Server Model) – Modely navržené v tomto stylu mohou mít strukturu přesně dle požadované funkcionality, rozvržené na jednotlivé provázané entity s vlastními rolemi (úlohami) v rámci celkového systému. Jednotlivé funkce mohou být různorodě rozmístěny na různých softwarových i hardwarových částech (tj. v různých procesech i na různých zařízeních), což umožňuje lepší sdílení těchto prostředků a komponent i pro obsluhu několika velkých aplikací ve stejném okamžiku. Takto rozvržený systém je pak i lépe rozšiřovatelný, ať už softwarově pro nové části systému, tak i hardwarově, když některé zařízení zastarává nebo by potřebovalo vylepšit – lépe se najdou problematické body, které systém zatěžují. Speciálním a nejužívanějším případem je níže zmíněný model třívrstvé architektury **[2][13]**.

- **Třívrstvá architektura** (3–tier Client/Server Model) – Vznikla vývojem z dvouvrstvé. Mezi klientem a databázovým serverem je umístěn druhý, aplikační server (viz obrázek 8), zajišťující například předávání dat klientovi, ukládání zpráv do front, spouštění různých aplikací, přesměrování požadavků na několik různých specifických databázových serverů nebo zaštiťovat jinou podobnou síťovou logiku **[2][13]**.

Obrázek 8: Třívrstvá architektura

Důvod vzniku třívrstvé architektury bylo eliminování nevýhod architektury dvouvrstvé – rozšiřitelnost pouze pro menší počet stanic, zátěž sítě, častá změna síťových pravidel znamená změny serveru i klientské části, problémy s aktualizacemi klienta a podobně.

Tyto problémy se ve třívrstvém modelu vyřešily přidáním aplikačního serveru, na který se přesunula většina softwarové logiky a při změně systému bylo tedy nutné místo opravy dvou míst (klienta i serveru) změnit jen aplikační server. Tento třetí server funguje jako prostředník, který koordinuje všechny klientské požadavky ke správným datovým serverům, čímž zároveň pro data zajišťuje i bezpečnější prostředí.

1.2.4 Použití

Architektura klient/server má všestranné použití a v moderní informatice je to jedna z hlavních vývojových myšlenek, kterou najdeme úplně všude. Většina světových obchodních a firemních aplikací používá tento model, ale jeho uplatnění je efektivní i v aplikacích pro domácí využití.

Nejvyužívanějším příkladem modelu klient/server dnešní doby je nepochybně používání webových prohlížečů k zobrazování internetových stránek. V tomto případě je prohlížeč klientem, který zobrazuje data poskytnuté web serverem (internetová stránka). Stejně tak je tato technologie použitá v emailových poštovních klientech a i další protokoly (kromě HTTP, IMAP a POP ho používají například také SMTP, DNS, Telnet a podobně) [13][19].

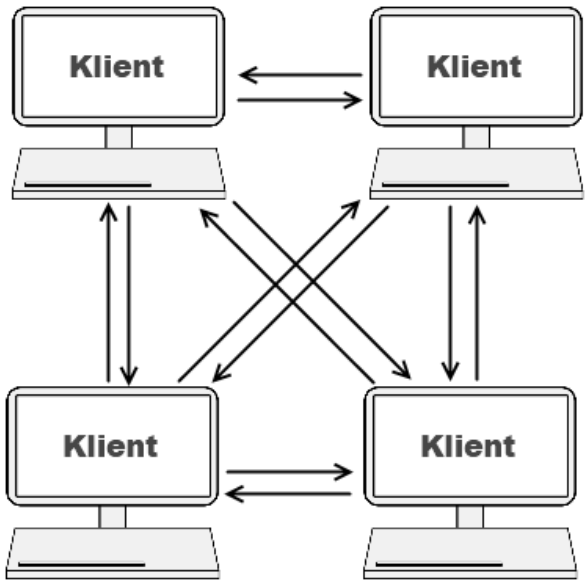
Stejný princip používá také většina počítačových her určených ke hře po počítačové síti (lokální, internetu).

Příklady aplikací tohoto typu najdeme dnes v téměř každém zařízení se síťovou kartou. S rozmachem užívání výpočetní techniky se i tento model dostal do téměř všech typů elektroniky, najdeme ho nejen v počítačích (stolních i přenosných), ale i v televizích, rádiích či mobilních telefonech a tento trend bude pravděpodobně dále pokračovat.

1.2.5 Alternativy

Peer–to–peer

Nejvýraznější alternativou k modelu klient/server je užití architektury Peer–to–peer (zkracováno P2P). Tato je také často nazývána klient–klient pro způsob její funkčnosti, jelikož je založena na přímé komunikaci klientů (uživatelů). Tato architektura vůbec nepoužívá server (ve smyslu centrálního úložiště, které je hierarchicky nadřazené), všechny klientské procesy jsou si rovny (viz obrázek 9) a působí zároveň jako klient (přijímání dat, vysílání požadavku) i jako server (zpracování požadavku, odesílání dat) [20].



Obrázek 9: Architektura Peer–to–peer

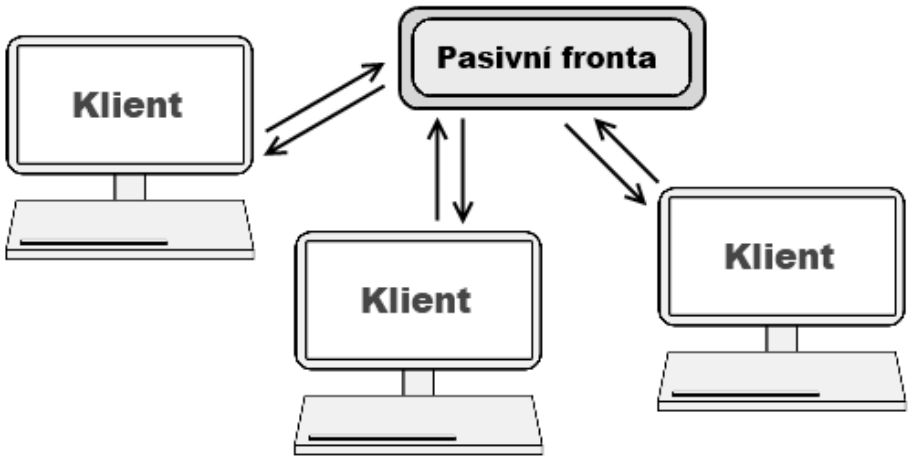
Největší výhoda Peer–to–peer architektury je relativní bezpečnost proti kompletnímu selhání – při selhání serveru v klient/server síti nemůže být žádný klient obsloužen a celkový systém tedy nefunguje. Naopak v architektuře Peer–to–peer se selhání jednoho článku projeví pouze menší šířkou pásma pro ostatní klienty, ale samotná funkčnost zůstává zachována. Další výhodou Peer–to–peer sítí oproti modelu klient/server se projevuje při rozšiřování klientské sítě. Zatímco při připojení klienta k serveru se zmenší šířka pásma, protože se o celkovou šířku pásma serveru musí dělit všichni klienti, u Peer–to–Peer se šířka pásma zvyšuje, protože každý klient má zároveň odesílací funkce serveru [20].

Největší nevýhodou je ale fakt, že zabezpečení peer–to–peer sítí je velice složité. Přestože existují moderní způsoby zabezpečení těchto modelů, bez centralizovaného uzlu, který autentizuje klienty a spravuje zabezpečovací pravidla, je komplexní bezpečnost jen těžko zaručitelná. Právě z tohoto důvodu se po těchto sítích šíří množství nelegálního obsahu a existují různá sdružení prosazujicí kompletní zakázání této síťové architektury [20].

Client–queue–client

Další alternativou k modelu klient/server je architektura nazývaná Client–queue–client (občas také passive queue). Její základy spočívají jak v Peer–to–peer tak v architektuře klient/server [19].

Namísto serveru je zde zavedena tzv. pasivní fronta. Jedná se o server, který ovšem nereaguje na žádosti klientů, funguje mezi nimi jen jako centrální uzel předávající zprávy (viz obrázek 10). Tento uzel je také rozdílem mezi Client–queue–client a Peer–to–peer, protože klienti si nepředávají zprávy přímo, ale jeho prostřednictvím [21].



Obrázek 10: Client–queue–client

Použití této architektury se nabízí v jednodušších aplikacích, které mají podobné předpoklady jako architektura klient/server, ovšem kde není důraz na komplikovanou řídící logiku pro práci s daty a kde tedy bude takovýto model nejen schopný fungovat, ale bude také velice vhodný, protože stejně tak jako u architektura klient/server budou hardwarové nároky klientů nízké, ale odpadnou také vysoké náklady na server – postačí pouze méně výkonný stroj (může běžet jako program na pozadí u některého z klientů) a také není potřeba specializovaný personál na údržbu, která zde není většinou vůbec třeba.

1.3 Moderní přístupy k této problematice

Přestože se tento výpočetní model používá intenzivně už desítky let, neustále se také vyvíjí. Změny se odehrávaly jak na úrovni klientů (používání tenkých i tlustých, vývoj hybridního a následně i chytrého klienta), tak na úrovni serverů (Transformace mnoha aplikací z dvouvrstvé na třívrstvou architekturu). Většinu z těchto změn s sebou přinesl rozvoj internetu, pro jehož potřeby a zdokonalení byly vynalézány různé pomůcky a metody. V dnešní době má téměř každý programovací jazyk vlastní knihovny usnadňující rozvoj takovýchto aplikací v nejrůznějších formách, od serverů pro ukládání dat, přes webové servery až po komplexní serverové aplikace o mnoha službách. V této kapitole budou nastíněny nejčastěji používané přístupy v dnešní době.

1.3.1 Vlastní klient/individuální server

Nejobecnější možná varianta je použití vlastnoručně vyrobeného klienta i serveru. V dnešní době standardizovaných programovacích knihoven není potřeba programovat komunikace síťových architektur od základu, ale s využitím mnoha podpůrných metod a funkcí má každý vývojář možnost navrhnout a vytvořit aplikace přesně podle své potřeby.

Klient

Klientem je aplikace vytvořená v libovolném programovacím jazyku. V tomto případě musí programátor kromě vytvoření aplikace také navrhnout grafické uživatelské rozhraní (GUI).

Server

Server je proces vytvořený také v libovolném programovacím jazyku. Tento proces může běžet na kterémkoliv operačním systému a na jakémkoliv hardwaru, pro který je zkompilován. Specifikace operačního systému, hardwaru ani programovacího jazyka nemusí mít nic společného s těmi, na kterých je postavena klientská část.

Komunikace

Nejpoužívanějším způsobem připojení klienta a serveru jsou i v dnešní době sokety, mnoho programátorů ale volí také komunikaci prostřednictvím vzdáleného volání procedur (RPC – Remote procedure call). Tato metoda byla používaná zvláště dříve, protože tak nebylo nutné vytvářet vlastní sokety pro navázání komunikace, ale při dnešních možnostech se k tomuto faktoru už většinou nepřihlíží. Ovšem i dnes má RPC význam – sokety jsou efektivnější a rychlejší pro přenos menšího až velkého obsahu dat, ovšem při přenosu extrémě velkých dat dosahuje rychlejší odezvy RPC. V dnešní doby je nejpoužívanější RPC variantou XML–RPC (RPC využívající XML extensive markup language – s přenosem po HTTP protokolu) [17].

Funkce soketů je popsána v kapitolách 1.1.7 a 1.2.1. V případě RPC, jak už název napovídá, volá klient metodu na serverové straně, její návratová hodnota se zabalí a je přenesena zpět do klientské části, kde se rozbalí a je podle potřeby zpracována. RPC může v závislosti na své formě fungovat buď samostatně, nebo také prostřednictvím soketů [17].

Výhody

Takto strukturovaná aplikace má nepřeberné možnosti. Každá její část může být uzpůsobena jakémukoliv účelu a může být taktéž do jakékoliv míry rozšířena, případně upravena a programátorovi se v tomto směru žádné meze nekladou. Další výhodou je možnost tvorby efektivního a přehledného GUI přesně dle potřeby a pro daný účel.

Nevýhody

Aspekty, které jsou výhodami tohoto konceptu, jsou zároveň i jeho největšími nevýhodami. I v případě použití standardizovaných knihoven neexistuje žádný nejlepší způsob návrhu a vytvoření takovéto aplikace a vše zde tedy záleží na programátorovi (i když existuje množství knih a návodů a skriptů pro kontrolu kódu, které mohou v tomto pomoci). V případě jeho pochybení, špatného návrhu aplikace, nepřehledného GUI, neošetřených výjimek nebo jakýchkoliv jiných nedokonalostí nemusí být zajištěna správná či dostatečně rychlá funkčnost aplikace.

Tato metoda je i v dnešní době často používána, a to pro široké spektrum účelů. Většinou se jedná o komerční aplikace – specializované firemní programy (často real–time), počítačové hry, centralizované databázové systémy a podobně.

1.3.2 Prohlížeč/individuální webserver

Speciálním případem výše zmíněné architektury je pak použití webového serveru a internetového prohlížeče k zobrazování dat. Tato metoda je dnes nejrozšířenějším typem modelu klient/server vůbec a její popularita neustále roste.

Klient

Klientem je kterýkoliv webový prohlížeč. Nejpoužívanějšími prohlížeči v dnešní době (k 1.5.2013) jsou [22]:

- Google Chrome – 52.7 %
- Mozilla Firefox – 27.9 %
- Internet Explorer – 12.7 %
- Safari – 4.0 %
- Opera – 1.7%

Existuje však ještě velká spousta dalších. Ovládací prvky prohlížečů se liší, ovšem uživatelské rozhraní je pro všechny stejné (většinou, až na malé výjimky, případně neschopnosti některého prohlížeče zobrazovat určité prvky). Webové prohlížeče jsou často přenositelné, není nutná žádná konfigurace k jejich základní funkcionalitě a navíc se vyskytují ve všech zařízeních s připojením k počítačové síti.

Server

Proces vyčkávající na připojení klienta. Server nejprve vytvoří soket pro požadované připojení (služba, protokol, port). Na tomto soketu následně naslouchá a čeká na příchozí připojení. Po připojení čeká na žádosti (ve formě URL) a odesílá zpět webové stránky pro zobrazení v klientské aplikaci (prohlížeči). Stejně jako u předchozí, obecnější možnosti, je i zde nepřeberné množství nástrojů a knihoven pro tvorbu webového serveru v nejrůznějších programovacích jazycích. Nehledě na vybraný jazyk, ve kterém je napsán server, odpovědi na požadavky klienta jsou psány v jazyce webových stránek (zobrazovací technologie). Mezi nejpoužívanější patří [23]:

- DHTML – Kombinace skriptovacího jazyka HTML, Kaskádových stylů (CSS) a JavaScriptu. Dříve byl problém s kompatiblitou zobrazování v jednotlivých prohlížečích, ovšem s příchodem HTML5 tento dříve velmi nepříjemný problém odpadl. Moderní weby používají, nebo alespoň přecházejí na kombinaci HTML5, CSS3 a JavaScriptu (nebo skriptů na JavaScriptu postavených, mezi nejvýznamější patří například JQuery). Pomocí této kombinace se dají vytvořit jednoduché i středně složité animace, odpadá tak nutnost používat Flash, který má vyšší paměťové nároky a také je méně stabilní [20].
- Macromedia Flash – Dynamické multimediální prezentace. Pro správný běh ve všech prohlížečích je nutný plugin. Flashové animace jsou jedny z paměťově nejnáročnějších procesů probíhajících ve webových prohlížečích, ovšem donedávna (před příchodem HTML5 a jeho tagu canvas) byly nejjednodušší a nejpoužívanější variantou pro animace a dynamické objekty o malé velikosti. Ovšem s příchodem výše zmíněného DHTML používajícího HTML5 už zájem o Flash pro jeho nevýhody ustává, například firma Adobe, jeden z hlavních dodavatelů vývojového softwaru pro Flash, zastavila podporu této technologie pro mobilní zařízení [24].
- Java applety – Softwarové součásti webových prohlížečů v jazyce Java. Při návštěvě webové stránky je natažen do prohlížeče a spouštěn ve Virtual Java Machine, prostředí, které musí být součástí prohlížeče, alespoň jako plugin, pokud už není obsaženo nativně. Tyto applety vznikly v roce 1995 a byly hojně používány pro běh rychlejší než JavaScript, kompatibilitu ve Windows, Linuxu i Mac OS a také pro podporu 3D hardwarové akcelerace, která je dostupná v Javě. Zhruba v roce 2011 už byla jejich rychlost srovnatelná, ne–li pomalejší než rychlost JavaScriptu, ale i v dnešní době se stále používají [23].

Komunikace

I tato komunikace funguje opět na bázi soketů. Server vytvoří soket naslouchající na kýženém portu, webové servery užívají běžně dva různé standardizované porty:

- 80 – HTTP – nezabezpečená komunikace se serverem
- 443 – HTTPS – komunikace zabezpečená protokolem SSL nebo TLS [23]

Webový prohlížeč, do kterého je zadána URL se nejprve dotáže DNS serveru na IP adresu, která náleží k internetové adrese URL. K serveru s touto IP se následně připojí a vyžádá si data na dané URL. Server je odešle zpět pro zobrazení v prohlížeči.

Výhody

Největší výhodou je fakt, že ve chvíli, kdy je webový server funkční, úplně odpadá nutnost instalace a konfigurace klienta. Pro každé síťové zařízení a operační systém existuje množství internetových prohlížečů, které fungují nativně bez jakéhokoliv zásahu administrátora serveru. Hardwarové nároky prohlížeče jsou naprosto minimální, proto funguje i na mobilních telefonech s minimální rychlostí procesoru a pamětí. Tato multiplatformovost je spolu s jednoduchostí obsluhy, nízkými nároky a podporou, kterou distribuované internetové architektury mají, důležitým aspektem, ke kterému vývojáři často přihlížejí.

Nevýhody

V případě potřeby komplexnějšího zobrazování je použití této metody velice náročné. Moderní internetové prostředky sice umožňují jednoduché, a v případě Flashe i složitější, ovšem paměťové náročné animace, ale tyto možnosti jsou horší a hůře implementovatelné než použití knihoven programovacích jazyků určených k vizualizaci. Ne každý počítač (zvláště v průmyslovém či velmi zabezpečeném prostředí) také disponuje připojením k serverum přes porty 80 a 443, které webové servery používají. V případě použití JavaScriptu musí být vývojář opatrný, neboť dřívější studie W3C ukazují, že 5–10% uživatelů internetu má JavaScript zakázán.

Tato metoda je základním stavebním kamenem internetu. Většinou je používána k interakci uživatelů s webovými servery, ovšem její použití je dobré nejen k zobrazování obsahu dat jednotlivých webových serverů, ale také k jejich editaci a dalšímu zpracování, vše záleží jen na samotném formátu serveru.

1.3.3 Další varianty

Existují samozřejmě ještě další používané varianty, ale většina používaných aplikací a programů vznikajících v dnešní době má právě jednu ze dvou výše zmíněných struktur.

Kombinace výše zmíněných variant

Další nejčastější výjimky by pak byly systémy, kde server zvládá obě úlohy, tj. funguje jednak jako server reagující na specifické požadavky vlastních specializovaných klientů, ale také jako webový server se schopností zobrazovat nabízená data jakožto webové stránky, dostupné z kteréhokoliv prohlížeče.

Tyto varianty bývají používány u průmyslových aplikací, například kontroly výroby, kde jsou k takovýmto serverům připojeny jak komplexní klienti s mnoha nadstandardními funkcemi, tak i jednodušší klienti ve formě prohlížečů pro kontrolu základních parametrů pro přístup z mimofiremních stanic.

2 Analýza

V této kapitole je popsána situace, pro kterou je daná aplikace vyvíjena, její jednotlivé aspekty a jsou probrány výhody i nevýhody jednotlivých síťových architektur, možných přístupů, programovacích jazyků i náročnost použití těchto metod pro dané účely.

2.1 Požadavky

Cílená aplikace by měla být robustní, postavená na open–source technologiích a splňovat množství požadavků sepsaných níže. Ty jsou směrovány jak na kompletní systém, tak i na jeho jednotlivé součásti.

2.1.1 Klient

Základní požadavky na klientskou aplikaci:

- mulitplatformovost – funkční na mnoha operačních systémech, hlavně systémech Microsoft Windows a distribucích OS Linux
- Zobrazení dat v reálném čase

Pro komplexní zobrazovací možnosti musí být klient schopen zobrazován:

- přijatá data cyklicky, v okamžiku kdy přijdou, bez nutnosti je jakkoliv ukládat či zpracovávat
- přijatá data v několika různých módech lišící se vzhledem, podle přání uživatele (mód – zobrazení různých skupin dat různými způsoby)
- data pomocí animací (od jednodušších po složitější)
- data nejen ze serveru, ale i z případného lokálního zdroje, například z místního úložiště, počítače v lokální síti nebo připojeného logického automatu
- data přes celou obrazovku na kterémkoliv připojeném displeji klientského počítače

Další vlastnosti klienta, které už nejsou nezbytně nutné, ale které by byly vhodné a ke kterým bude přihlíženo při výběru struktury programu, jsou:

- schopnost měnit vizualizační prvky uživatelem v grafickém uživatelském rozhraní
- schopnost používat v animacích 3D modely z pokročilých grafických editorů
- jednoduchá konfigurovatelnost uživatelských i superuživatelských funkcí
- schopnost aktualizace bez uživatelského zásahu
- schopnost zobrazení webových kamer uživatelů jednotlivých klientů
- rozšiřitelnost klienta – snadnost přidání nových módů vizualizace či nových funkcí

2.1.2 Server

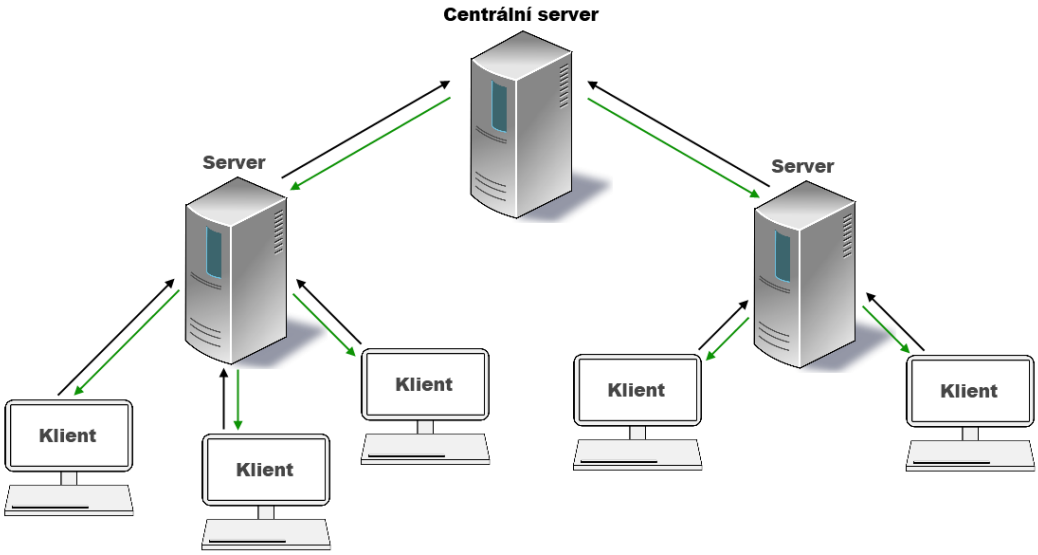
Proces na serveru musí samozřejmě splňovat všechny předpoklady serverové aplikace – odpovědi na přijaté žádosti klientů, nepřetržitý provoz, apod. Musí vytvořit soket, přiřadit mu předem určený port, naslouchat na něm a čekat na příchozí připojení.

Server ale také musíme přizpůsobit následujícím potřebám. Server musí být schopen:

- obsloužit řádově desítky klientů
- autentizovat příchozí připojení, rozhodnout zda–li je klient způsobilý k příjmu vyžádaných dat (není na černé listině) nebo odmítnout jeho obsloužení
- shromažďovat data od klientů, vybírat jejich části podle klientských žádostí a posílat je zpět, vše v reálném čase
- zajišťovat komunikaci uživatelů jednotlivých klientů
- odesílat klientům informace z:
    - lokálního úložiště
    - paměti
    - databáze – kromě čtení také zápis, dle potřeby

Vzhledem k tomu, že cílem této práce je navrhnout globální systém s rychlými odezvami (v řádově desítkách či stovkách milisekund), server by měl být geograficky lokalizován poblíž umístění klientské báze, aby kabelová trasa mezi serverem a klientem byla co nejmenší. Země je ovšem rozlehlá, proto je adekvátním řešením více serverů, obsluhující klienty ve svém okolí.

Tyto servery musí být mezi sebou také propojeny, z bezpečnostního hlediska raději také architekturou klient/server (i když architektura Peer–to–peer se taky jeví jako rozumné řešení). Struktura celého systému je znázorněna na obrázku 11.



Obrázek 11: Architektura s centrálním serverem

To znamená, že každý server musí mít zároveň svojí „klientskou část“, která odesílá svá data a posílá žádosti pro příjem dat ostatních serverů centrálnímu serveru. To je dalším požadavkem serverové aplikace. Variantou by také bylo zasílání požadavku klienta všem ostatním serverům, ale v tomto případě by musel klient před samotnou vizualizací přijatá data dále zpracovávat (vybírat potřebná, hledat případné duplicitní záznamy, apod.) a to je v rozporu s požadavky na klientskou část).

Server by měl být také jednoduše rozšiřitelný pro případné další funkce.

Tento serverový proces ovšem musí běžet na nějakém konkrétním počítači. Z tohoto důvodu se nabízí dva přístupy, jakým způsobem tento hardware opatřit:

- Koupě počítače nebo specializovaného dedikovaného serveru – S tímto se ovšem pojí další povinnosti, které musí administrátor tohoto serveru vykonat:
    - umístit na vhodné místo (ideální teplota a relativní vlhkost vzduchu)
    - konfigurace stroje – instalace OS, nastavení síťového připojení, spuštění služeb
    - zajištění zálohování na jiné lokální/síťové místo
    - zajištění napájení, rychlého internetového připojení, dostupnosti
    - zajištění pravidelné údržby pro správnou funkčnost zařízení, apod.

  Nebo může tento stroj svěřit nějaké společnosti, která tyto funkce za poplatek provádí
- Pronajmutí takového počítače od společnosti, která výše zmíněné služby dodává v ceně měsíčního paušálu serveru, tj. zajišťuje pravidelné zálohování, připojení k páteřní síti internetu přes optické kabely, údržbu, změny hardwaru, zaručení dostupnosti na internetu, použití záložních generátorů v případě výpadku energie a podobně.

2.1.3 Komunikace

Dále jsou kladeny mírné požadavky i na samotnou komunikaci mezi jednotlivými uzly:

- Klientská a serverová (případně serverová a serverová) část musejí být přes internet propojeny prostřednictvím soketů pro zajištění spolehlivé komunikace.
- Přenášená data musí být zašifrována (hlavně kvůli zasílání přihlašovacích údajů – ochrana proti případnému odposlechu).

2.1.4 Databáze

Požadavky na databázi jsou o poznání volnější než na ostatní části aplikace – databáze pouze musí být přítomna, pro občasné zápisy hodnot (statistické účely) a pro ukládání důležitých informací.

Struktura databáze, její typ a prostředky kterými je vyvíjena nejsou důležité, musí být ovšem zajištěny základní databázové vlastnosti – stabilita, integrita dat, možnost zálohy a obnovy.

2.2 Porovnání klientských variant

Tato kapitola porovnává výhody a nevýhody jednotlivých typů klientů pro aplikaci, která byla popsána podmínkami výše.

2.2.1 Tenký klient – webový prohlížeč

Webový prohlížeč je právem nejpoužívanějším klientem dnešní doby.

Při jeho použití není vůbec třeba tvořit klientskou část, protože prohlížeč samotný je nativní součástí všech nejpoužívanějších operačních systémů:

- V počítačích – Microsoft Windows, Mac OS X, GNU Linux, apod.
- V mobilních zařízeních – Windows Mobile OS, Mac iOS, Google Android, Symbian, MeeGo, BlackBerry OS, Bada OS a další
- V chytrých televizích – Většinou používány operační systémy pro mobilní zařízení používané výše.

Prohlížení internetu je dnes zkrátka základní činností pro každé zařízení se síťovou kartou. Použití tohoto tenkého klienta je tedy z hlediska dostupnosti služeb rozumným řešením.

Analýza výhod

- Multiplatformní použití, přenositelnost – splnění základní podmínky
- Schopnost přenášet a zobrazovat data v reálném čase – základní podmínka
- Cyklické zobrazení dat bez dalšího zpracování
- Aktualizace bez uživatelského zásahu – pouze na straně serveru, klient se aktualizuje automaticky, s plnou podporou vývojářů daného prohlížeče
- Rozšiřitelnost klienta – v rámci serveru, jen s další URL
- Snadná administrace – nasazení, správa, uživatelé
- Možnost jednoduchých animací, přímé komunikace uživatelů,..

Analýza nevýhod

- Obtížný offline mód, vyžaduje další soubory na klientském počítači
- Omezené vizualizační možnosti
- Obtížné změny vizualizačních prvků (náročnější serverová část)
- Zobrazení v celoobrazovkovém režimu závislé na typu prohlížeče
- Potřeba výkonného webového serveru

2.2.2 Tenký klient na míru

Dalším řešením je použití vlastního tenkého klienta.

Na rozdíl od tenkého klienta v podobě webového prohlížeče, zde by byla potřeba klientskou část naprogramovat.

Analýza výhod

- Schopnost přenášet a zobrazovat data v reálném čase – základní podmínka
- Cyklické zobrazení dat bez dalšího zpracování
- Rozšiřitelnost klienta – možnosti větší než u prohlížeče
- Pokročilé možnosti animací (závislé na programovacím jazyku)
- Možnost měnit vizualizační prvky uživatelsky
- Schopnost přímé komunikace uživatelů
- Možné použití 3D modelů ve vizualizaci dat
- Zobrazení celoobrazovkové vizualizace na vybraný displej
- Jednoduché přepínání zobrazovacích módů uživatelem
- Jednoduchá konfigurovatelnost uživatelských funkcí

Analýza nevýhod

- Multiplatformovost možná, ale musí být docílena programátorem
- Obtížný offline mód, vyžaduje lokální server
- Aktualizace musí být naprogramované ručně
- Hardwarově náročnější než prohlížeč, náročnost roste úměrně s pokročilostí použitých animací a vizualizačních prvků

Z těchto bodů je patrné, že oproti webovému prohlížeči má tento typ tenkého klienta širší možnosti zobrazení, závislé jen na vizualizační schopnost vybraného programovacího jazyka nebo jeho knihoven, ovšem na druhou stranu musí být pro multiplatformní použití naprogramován a možnosti jeho využití na různých operačních systémech či zařízeních nebudou nikdy tak široké jako v případě prohlížeče. Pro jeho použití stále stačí méně výkonný počítač (či jiné zařízení), stále je ale toto řešení náročnější než samostatný webový prohlížeč.

2.2.3 Tlustý klient

Nyní se zaměříme na analýzu použití tlustého klienta.

Stejně jako u tenkého klienta na míru je i zde potřeba vytvořet současně se serverovou částí i část klientskou.

Analýza výhod

- Schopnost přenášet a zobrazovat data v reálném čase – základní podmínka
- Snadné docílení offline funkčnosti, nejsou potřeba lokální servery ani další soubory
- Cyklické zobrazení dat bez dalšího zpracování, ale je možná i jejich úprava
- Rozšiřitelnost klienta – možnosti větší než u tenkého klienta
- Pokročilé možnosti animací (závislé na programovacím jazyku)
- Možnost měnit vizualizační prvky uživatelsky
- Schopnost přímé komunikace uživatelů
- Možné použití 3D modelů ve vizualizaci dat
- Zobrazení celoobrazovkové vizualizace na vybraný displej
- Jednoduché přepínání zobrazovacích módů uživatelem
- Jednoduchá konfigurovatelnost uživatelských a dalších pokročilých funkcí

Analýza nevýhod

- Multiplatformovost možná, ale musí být docílena programátorem
- Aktualizace musí být naprogramované ručně
- Hardwarově náročnější než tenký klient
- Nutnost větší pozornosti při programování, více možných chybových stavů
- Větší náročnost na šířku pásma při komunikaci se serverem

Z analýzy výhod a nevýhod je jasné, že tlustý klient přebírá všechny výhody tenkého klienta naprogramovaného na míru a k nim přidává ještě několik dalších, například možnost fungování (s omezenými funkcemi) i bez přístupu k serveru. Další podstatnou výhodou jsou bohaté vizualizační schopnosti, které o mnoho převyšují možnosti tenkých klientů. Proto jsou tlustí klienti většinou používáni jako prostředek pro tvorbu síťových počítačových her. Jeho nevýhody jsou také podobné, nejvýraznější nevýhodou je fakt, že multiplatformovost musí být zajištěna programátorem.

2.3 Porovnání databázových systémů

V kapitole 1.1.9 jsme si představili nejpoužívanější systémy řízení báze dat, ovšem pouze několik z nich je open–source (tzv. otevřený software – otevřený zdrojový kód, licence podporující editaci a volné používání zdrojového kódu, často i pro komerční účely – záleží na typu a přesném znění dané licence):

- MySQL – viz kapitola 1.1.9
- PostgreSQL – viz kapitola 1.1.9
- SQLite – viz kapitola 1.1.9

Tato práce si klade za cíl vytvoření robustní globálně použitelné aplikace, proto se použití SQLite (kompaktního systému řízení báze dat, který může mít problémy při práci více uživatelů současně) nejeví jako příliš vhodné. Zaměříme se tedy pouze na analýzu systémů PostgreSQL a SQLite.

Dlouhou dobu se předpokládalo, že MySQL je rychlejší z těchto dvou systémů a PostgreSQL je naopak systém s bohatší výbavou, ovšem v dnešní době se funkcionalita MySQL mnohonásobně rozrostla a stejně tak se zvýšila rychlost systému PostgreSQL. Ovšem i dnes většinou platí, že MySQL je rychlejší v jednoduchých dotazech a PostgreSQL spolehlivější a rychlejší v komplexních případech. Oba systémy jsou funkční na všech hlavních operačních systémech i na počítačích s více procesory.

2.3.1 PostgreSQL

Tato kapitola zanalyzuje vhodnost použití systému řízení báze dat PostgreSQL pro použití ve výše popsané aplikaci.

PostgreSQL bylo vyvinuto v roce 1985 na univerzitě v Berkley jako nástupce Ingres databáze. Podle vývojářů tohoto systému se jedná o nejpokročilejší open–source databázový systém, který má širokou celosvětovou komunitu fungující zároveň jako podpora.

Analýza vlastností

- Účinná komprese a dekomprese za běhu pro úsporu místa a rychlejší práci s daty
- Nastavitelnost pro co největší efektivitu podle prostředí na kterém běží
- Škála funkcí zajišťujících integritu, funkčnost, jednoduchost a výkon
- Nativní synchronní i asynchronní replikace dat – pro zálohu
- Plná podpora funkce „join“ pro dotazy do hloubky
- chybí zabudovaná funkce pro nastavení maximální velikosti databáze
- Podpora asynchronního čtení/zápisu, lepší podpora multiprocesorů než u MySQL

2.3.2 MySQL – InnoDB

MySQL je nejpoužívanější systém řízení báze dat. Byl vytvoře v roce 1995 a už od začátku byl koncipován jako systém zaměřený na rychlost čtení a zápisu.

MySQL se skládá ze dvou vrstev:

- vrchní – SQL vrstvy – obsahuje všechny funkce a nadstandardní výbavu
- úložiště dat – nativní podpora devíti formátů, InnoDB je dnes nejpoužívanější

Analýza vlastností

- Škála funkcí zajišťujících integritu, funkčnost, jednoduchost a výkon
- Nativní asynchronní replikace dat – pro zálohu, chybí synchronní replikace dat pro klasické hvězdicové topologie – zde nepotřebné
- Podpora funkce „join“ pro dotazy do hloubky, ovšem nedosahuje funkčnosti PostgreSQL
- Mechanismus pro nastavení maximální velikosti databáze – z tohoto důvodu je často užívána hostingovými firmami, které tak mohou lépe kontrolovat prostor, který uživatel dostává
- Používaná více než PostgreSQL – větší uživatelská podpora i více psaných materiálů

2.3.3 Porovnání možností pokročilé indexace

Porovnání možností indexace MySQL a PostgreSQL		
Typ Indexu	**MySQL – InnoDB**	**PostgreSQL**
Hash index	Ano	Ano
Vícenásobné indexy	Ano	Ano
Fulltextové indexy	Ano	Ano
Částečné indexy	Ne	Ano
Prefixy	Ano	Jako součást výrazových
Vícesloupcové indexy	Maximálně 16 sloupců	Maximálně 32 sloupců
Bitmapové indexy	Ano, ale ne nativně	Ano
Výrazové indexy	Mohou být emulovány	Ano
Neblokované CREATE INDEX	Ano	Ano
Krycí indexy	Ano	Ano

Tabulka 2: Porovnání možností indexace databázových systémů

3 Návrh aplikace

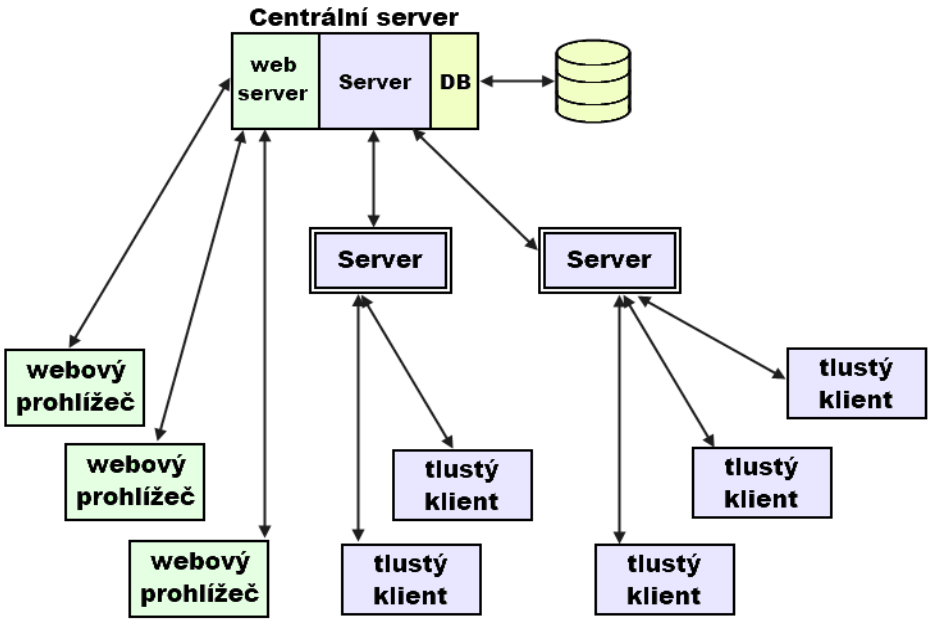
Tato kapitola se zabývá konečným návrhem aplikace – vybráním vhodné architektury, prostředků, pomocí kterých bude aplikace vyvíjena a popisem jejich použití. Dále bude také znázorněna funkčnost jednotlivých složek včetně použití diagramů znázorňujících běh jednotlivých částí.

3.1 Výběr architektury

Přestože většina dnešních modelů klient/server je stavěna ve smyslu třívrstvé architektury (viz kapitola 1.2.3), rozhodl jsem se použít architekturu dvouvrstvou, která má i dnes své využití, zvláště pro účely podobným cílům této aplikace. Třívrstvá architektura je nejčastěji používána pro potřeby internetu, to jest zobrazování většinou jednoduchého obsahu s častými datovými zápisy/čtením.

Jak už bylo nastíněno v kapitole 2.1.2, architektura tohoto systému je hierarchická – k jednomu nebo více serverům se připojuje předem neznámé množství klientů. V případě, že je těchto serverů víc, jsou tyto připojeny k centrálnímu serveru, který zajišťuje přenos dat mezi nimi. Tento model zajišťuje rychlé odezvy v rámci blízkého geografického okolí.

Jádrem systému je serverový proces, který je jednak určen ke komunikaci se specializovanými tlustými klienty pro pokročilé vizualizační prostředky a zachovanou funkčnost i bez připojení na internet. Zároveň má ale centrální server i webserverovou část, která je uzpůsobená pro zobrazení dat ve formě webových stránek. Tímto způsobem zůstanou schopnosti nadstandardního zobrazení dat v téměř libovolné formě, ale zároveň je zachována multiplatformovost s použitím tenkého klienta (webového prohlížeče). Tento model (viz obrázek 12) je dostatečně obecný pro použití v mnoha situacích podle konkrétní potřeby a pro konkrétní účel.



Obrázek 12: Výsledný návrh architektury

3.2 Výběr programových prostředků

Podle TIOBE Indexu (pořadí podle míry popularity – záznamů z nejpoužívanějších vyhledávačů) jsou v dnešní době používáno nejvíc těchto 10 programovacích jazyků [25]:

1.	2.	3.	4.	5.
C	Java	Objective–C	C++	C#

6.	7.	8.	9.	10.
PHP	Visual Basic	Python	Perl	Ruby

Tabulka 3: Srovnání programovacích jazyků

Šedivě označené položky v tabulce 3 jsou ty, které jsou zároveň v první desítce seznamu Craigslist jako jazyky, o něž je největší zájem při hledání programátorů (k dubnu 2013). [26]

Pro účely této práce jsem z výše nabízených vybral jazyk Python. Tato kapitola se zaměří na popis technologií použitých k návrhu architektury popsaného výše.

3.2.1 Python

Programovací jazyk Python je open–source, což je jeden z předpokladů výběru.

Jedná se o dynamicky interpretovaný skriptovací jazyk, jehož možnosti jsou ale srovnatelné s klasickými programovacími jazyky, například jazykem C, na kterém je Python postaven. Je multiplatformě podporovaný a některé operační systémy, jako například Red Hat Linux, v tomto jazyce implementují základní části operačního systému [27]. V posledních letech je tento jazyk velice oblíben pro svou jednoduchost tvoření zdrojového kódu, jelikož jednoduchá syntaxe, velké množství užitečných funkcí a široká škála podpůrných knihoven z něj dělají efektivní nástroj pro rychlou tvorbu aplikací (Rapid Application Development) a v tomto ohledu se mu může málokterý programovací jazyk vyrovnat. Studie ukazují že čas pro tvoření aplikací v Pythonu zabere programátorům méně než polovinu času (často mnohem méně), který je potřeba pro vytvoření stejné aplikace v jazycích C++, C# nebo Java [28]. Kvůli velmi přehlednému a čitelnému kódu je také doporučován jako programovací jazyk pro začátečníky, ovšem stejně tak poslouží i zkušeným programátorům. Kromě výše zmíněných výhod je také Python snadno integrovatelný s jinými jazyky a jeho výkon, ač většinou nedosahuje hodnot C++, je ale obecně 3–5x rychlejší než výkon PHP [27]. Disponuje také snadným testováním kódu a širokou uživatelskou podporou.

Kromě uvedených výhod má také Python velké množství open–source knihoven, které jsou jednak vhodné pro aplikace podobného typu, a navíc jsou také pod licencí LGPL, která umožňuje vývoj jak svobodných (open–source), tak nesvobodných (closed–source) aplikací. Obě tyto licence mohou být použity i pro komerční aplikace, ale LGPL povoluje dodávání aplikace bez zdrojového kódu, což je obecně vhodnější, protože ne každý, kdo by používal tuto architekturu musí souhlasit s otevřeným kódem.

Twisted

Twisted je síťový framework pro Python. Tato knihovna je v poslední době velice oblíbená a používaná společnostmi různě ve světě, ale k jejímu používání se hlásí i známé organizace jako NASA či filmařská společnost LucasFilm [29].Jedná se o obsáhlou a ucelenou knihovnu používanou pro asynchronní (událostmi řízené) komunikace. Podporuje všechny běžné protokoly a obsahuje všechny potřebné třídy pro vytvoření nejrůznějších typů klientů i serverů. Zároveň má také širokou komunitu složenou z jedinců i různých organizací, takže má i podporu v blízké budoucnosti zajištěnou [30].

Pygame

Jedná se o set modulů pro Python navržený pro tvorbu her. Je určen pro tvorbu plnohodnotných počítačových her a multimediálních programů, s podporou počítačů s více procesory a různými video ovladači. Pygame je schopen tvorby nejen pokročilých animací, ale také 3D prostředí s mnoha ovládanými prvky. Současně je také tento modul schopen pracovat se zvukem, zvukovými i multimediálními soubory a také obsluhovat různá vstupní zařízení a reagovat na ně, aniž by musel vývojář používat nízko–úrovňové programování.

PySide

Tato knihovna je určená pro provázání Pythonu s frameworkem Qt. Qt je multiplatformní nástroj pro tvorbu grafických uživatelských rozhraní. Spolu s GTK+ patří k nejpoužívanějším knihovnám používaných k interakci s uživatelem. Modul PySide je schopen zprostředkovat grafické prvky Qt jednoduchým a funkčním způsobem pro snadnou tvorbu jakkoliv náročného uživatelského prostředí.

Storm

Storm je modul jazyka Python pro mapování relační databáze na objekty. Je schopný pracovat s malými typy systémů řízení báze dat jako SQLite stejně jako s obsáhlejšími systémy typu PostgreSQL a MySQL. Stejně jako Twisted je i tento modul založen na asynchronním zpracování požadavků, je rychlý, efektivní a jednoduše použitelný i pro náročné žádosti či nestandardní funkce.

lxml

Knihovna jazyka Python pro jednoduché vytvoření XML (Extensible Markup Language). Bývá označována za funkcionálně nejbohatší a nejjednodušeji používanou knihovnu pro zpracování XML a HTML v Pythonu. Výhodou přenosu dat ve tvaru XML je jednoduchá čitelnost člověkem i jednoduché zpracování počítačem díky přehledným značkám.

PyYaml

je knihovna pro snadné čtení a ukládání dat v YAML formátu. YAML je formát pro serializaci strukturovaných dat. Stejně jako XML je také snadno čitelný člověkem.

json

Enkodér a dekodér pro formát JSON (JavaScript Object Notation) – je multiplatformí datový formát. Díky využití jednotných konvencí mnoha programovacích jazyků je JSON vhodný pro výměnu dat nejen mezi různými operačními systémy a jazyky.

Tyto knihovny tvoří základ celého systému a jejich použití bude objasněno u jednotlivých dílčích subsystémů.

3.2.2 HTML5

HTML5 je HTML (HyperText Markup Language) nové generace. Kromě unifikování mnoha funkcí, které se v různých webových prohlížečích lišily, přidává funkce nové, nezbytné v dnešním světě pro tvorbu moderní webových aplikací [20] :

- Canvas – dvourozměrné plátno pro kreslení a animace s využitím JavaScriptu
- Video – pro zobrazení videa na webových stránkách bez nutnosti použití jakýchkoliv pluginů
- Možnost geolokace – zjištění fyzické lokality ze které uživatelé k serveru přistupují
- lokální klientské úložiště pro ukládání dočasných souborů
- Vylepšení webových formulářů
- Zpětná kompatibilita se staršími prohlížeči

Mnoho vývojářů se tomuto standardu stále vyhýbá, ale přesto už je HTML5 rozšířeno a podporováno většinou webových prohlížečů. Všechny hlavní prohlížeče (viz 2.3.2) už podporují plátno canvas, vnořené video, geolokaci i lokální úložiště. Dokonce i Microsoft Internet Explorer, často kritizovaný pro nedostatečnou podporu nových standardů, dokáže většinu HTML5 prvků zobrazit.

Navíc je HTML5 uznávano jako standard a podle W3C (organizace fungující jako základna webových standardů založená Tim Berners–Leem – rovněž zakladatel World Wide Webu) je v dnešním době jediným standardem s neustávajícími aktualizacemi a podporou.

3.2.3 CSS3

CSS je označení pro kaskádové styly (Cascade Style Sheet), což je mechanismus pro přidávání stylů do webových dokumentů. CSS3 je na rozdíl od předchozích verzí rozčleněna do modulů pro jednoduché přidávání nových funkcí při zachování zpětné kompatibility [31].

CSS umožňují jednoduché a přenositelné nastavení vzhledu prvků nejen samotných webových stránek, ale i například prvků grafických uživatelských rozhraní.

3.2.4 Qt

Qt je multilplaformní prostředí pro tvorbu aplikací a grafických uživatelských prostředí. Spolu s GTK+ patří k nejpoužívanějším nástrojům pro tvorbu GUI, ale na rozdíl od GTK+ se nativně snaží přizpůsobit vzhled grafických prvků systému, na kterém je používáno. V současné době je ve vývoji společnosti Nokia se zaměřením na mobilní telefony této značky, nicméně je podporován i ve všech hlavních operačních systémech s knihovnami pro nejpoužívanější programovací jazyky dnešní doby.

3.2.5 Gettext

GNU Gettext je sada nástrojů umožňující jednoduché vytváření vícejazyčných aplikací. Gettext je určen k použití s velkým množstvím programovacích jazyků, mimo jiné i s jazykem Python. Jednoduchost tohoto rozhraní je v automatickém procházení zdrojovými soubory k nalezení všech použitých názvů a hlášek a automatickém vygenerování katalogů, do kterých se následně vepisují překlady. Tímto způsobem je možné i průběžně doplňovat další jazykové lokalizace bez nutnosti měnit části zdrojového kódu.

3.3 Server

Serverová část centrálního i necentrálního serveru jsou funkčně naprosto totožné. Vytvářejí sokety pro naslouchání na vybraném portu a zpracovávají došlé požadavky.

3.3.1 Požadavky

Přestože je server o následující struktuře multiplatformní, je pro jeho běh nutné splnit několik základních požadavků:

- Operační systém s podporou jazyka Python
- programovací jazyk Python
- knihovny Twisted, pyYaml, lxml, Storm a json

3.3.2 Výběr programovacích prostředků

Pro tento účel byl zvolen programovací jazyk Python s komunikacemi založenými na knihovně Twisted. (viz 4.2.1). Tyto prostředky umožňují jednoduché vytvoření základní části serveru obsahující vytvoření a obsluhu soketu, asynchronní komunikaci s klienty, ale i vytváření vláken v případě potřeby pro efektivní správu spojení s klienty. Knihovna Twisted zároveň umožňuje používání vygenerovaných SSL certifikátů pro zabezpečené spojení, vytváření webových serverů, apod. Zároveň je tento jazyk multiplatformní, je tedy možné spustit skript v tomto jazyce na velkém množství operačních systémů, ať už určených pro serverové či domácí účely.

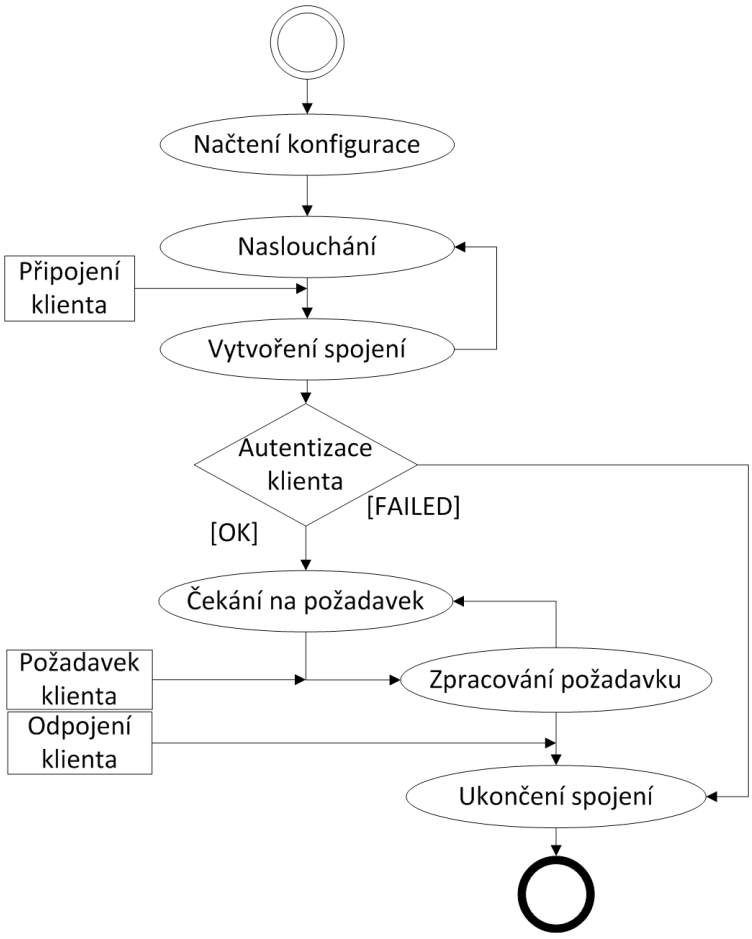
3.3.3 Funkce serveru

Při startu serveru se nejprve načte konfigurační soubor, který server použije ke správné inicializaci. Tento soubor obsahuje všechny potřebné informace, např.:

- Port pro naslouchání příchozích komunikací tlustých klientů
- Port pro naslouchání modulu webového serveru
- lokální či vzdálenou adresu databáze
- přihlašovací informace do databáze
- černou listinu obsahující odmítané klienty
- autentizační údaje klientů
- údaje o dostupných aktualizacích pro jednotlivé klienty

Tyto informace pak používá ke správnému běhu. Tento konfigurační soubor má tu výhodu, že v případě jakékoliv změny lokace serveru, databáze či jeho jiných součástí, není třeba provádět žádné změny v serverovém kódu.

Po načtení konfigurace začne server naslouchat na vybraném portu, připojuje klienty, zpracovává jejich požadavky a při odpojení klienta ukončuje toto spojení (viz obrázek 13)



Obrázek 13: Funkce serveru a průběh komunikace s klientem

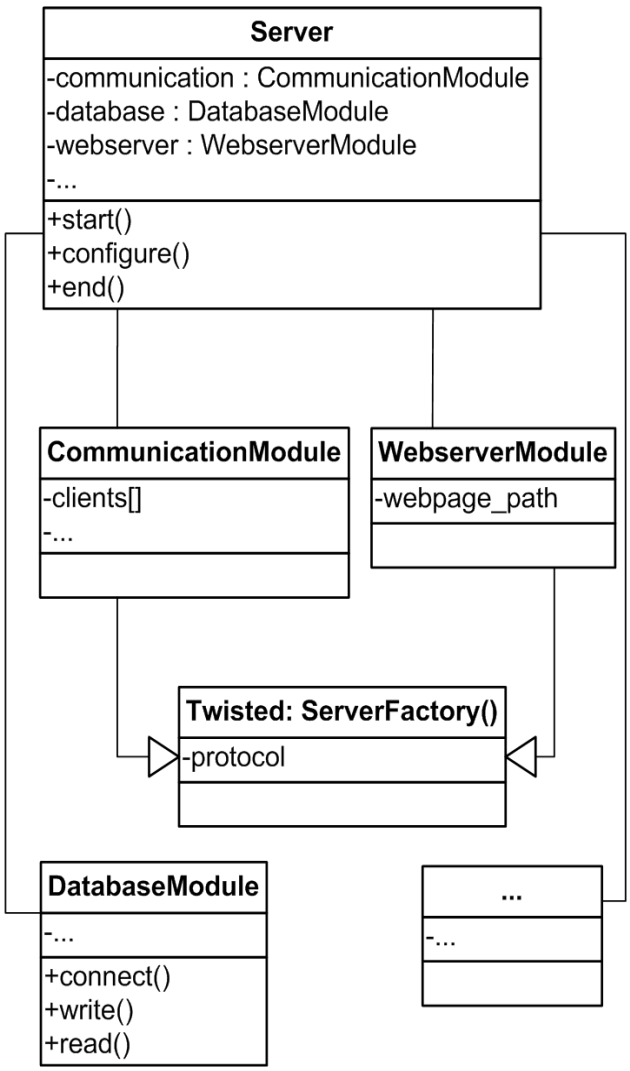
3.3.4 Struktura serveru

Server sestává ze základní třídy a různých připojitelných modulů pro konkrétní funkce, např.:

- komunikace s klienty
- automatické aktualizace klientů
- práce s daty
- práce s databází

Tyto moduly se iniciují při jeho startu, nastaví podle dostupné konfigurace a plní své funkce, dokud není serverová aplikace ukončena. Všechny komunikační moduly jsou odvozené od tříd knihovny Twisted. Výhody tohoto odvození jsou popsány v kapitole 3.3.2.

Struktura serveru znázorněna pomocí UML diagramu je ukázána na obrázku 14.



Obrázek 14: UML Diagram Serveru se vzorovými moduly

3.4 Webový server

Součástí serverové části by kromě samostatné serverové části zajišťující TCP/IP komunikaci také část zprostředkující klientská data skrze webové rozhraní.

Komunikační modul je zde odvozen od třídy websocket knihovny Twisted, která umožňuje jednoduché využití webových soketů. Ty jsou součástí standardu HTML5 a v dnešní době jsou velmi populární díky možnosti současné obousměrné komunikace, která se jinak v případě webových technologií zajišťuje složitě.

Webový server je inicializován při startu serveru spolu s ostatními moduly a ihned začne naslouchat na vybraném portu.

Součástí webového serveru je i třída pro vytváření HTML souborů, která je volána pokaždé, když od klienta přijdou změněná data. Struktura HTML souboru závisí na množství dat, které mají být zobrazeny a může být upravena pro konkrétní případy.

HTML soubory jsou separovány podle skupin a restaurací do stromové struktury, lze tedy získat informace o konkrétní restauraci bez procházení celého stromu.

3.5 Tenký klient – browser

Při dnešních možnostech prohlížečů, standardizaci HTML a rozšířeném používání JavaScriptu se jeví použití tenkých klientů ve formě internetových browserů jako velice praktické řešení. Pro animace a hry, kde byl dříve zapotřebí Flash, stačí dnes již HTML5 canvas a JavaScript, nejsou potřeba žádné nadstandardní moduly a aktualizace a proto je tato technologie dnes tolik ceněna.

Navíc s použitím HTML5 pro tvorbu webových stránek se zmenšují rozdíly způsobované odlišnostmi jednotlivých internetových prohlížečů, které jsou dnes nativní součástí téměř každého zařízení obsahujícím síťovou kartu.

3.6 Databáze

Tento systém je navrhován jakožto univerzální a poměrně robustní řešení pro širokou škálu aplikací, proto nelze předem jasně stanovit, jak obsáhlá databáze bude třeba. Vzhledem k tomuto faktu a požadavkům na tento systém mohou být z možných databázových systémů zmíněných v kapitole 1.1.9 použity jen MySQL a PostgreSQL, analyzované v kapitole 2.4

Pro tento systém byl vybrán systém řízení báze dat PostgreSQL a to z následujících důvodů:

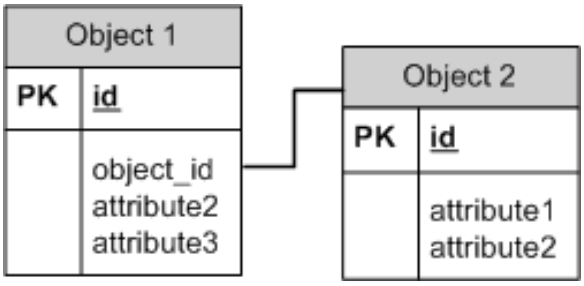
- Rychlost – už dlouho neplatí, že je PostgreSQL ve srovnání s MySQL pomalý; v posledních letech se rychlost tohoto systému výrazně zvýšila a je MySQL dobrým konkurentem
- Podpora jakkoliv náročných aplikací – PostgreSQL je výhodný jak k tvorbě malých databází, tak k databází pro velké aplikace, je používaný i nadnárodními společnostmi či výzkumnými centry jako NASA [29] pro nejrůznější projekty.
- Spolehlivost a funkčnost – Přestože se spolehlivost a možnosti MySQL s použitím InnoDB zvýšily, PostgreSQL je pořád brán za mnohem robustnější systém s větším množstvím možností. Pro velké množství pokročilých funkcí je často přirovnávám k Open–source Oracle.
- Rozšiřitelnost – jak bylo zmíněno v úvodu této kapitoly, obsáhlost databáze není předem známa a jsou možné průběžné změny, případná rozšiřitelnost databáze je také důležitým prvkem ke kterému je přihlíženo. Pro tyto účely je PostgreSQL také výhodnější.

3.6.1 Spojení s databází

K vytvoření spojení serverové aplikace s databází je použita knihovna Storm popsaná v kapitole 3.2.1. Pomocí této knihovny je možné databázi vytvářet, editovat i mazat.

Pro používání této knihovny k práci s databází je nejprve třeba vytvořit třídy reflektující objekty v databázi včetně spojení mezi nimi.

Příkladem může být databáze definovaná modelem na obrázku 15:



Obrázek 15: Model vzorové databáze

Pro takovou databázi by vytvořené třídy měly tuto podobu:

```
class Object1(object):
__storm_table__ = 'Object1'
id = Int(primary=True)
object_id = Reference(Object2, Object2.id)
attribute2 = Unicode()
attribute3 = Unicode()
```

```
class Object2(object):
__storm_table__ = 'Object2'
id = Int(primary=True)
attribute1 = Unicode()
attribute2 = Unicode()
```

3.7 Tlustý klient

I přes dnešní možnosti tenkých klientů se občas použití tlustých klientů jeví jako dobré a efektivní řešení. V dnešní době mohou i tencí klienti fungovat do jisté míry offline, ale pro účely lokálního fungování s pokročilými zobrazovacími možnostmi je pořád jednoduší vytvoření tlustého klienta.

3.7.1 Výběr programovacích prostředků

Stejně jako pro serverovou část byl i zde použit programovací jazyk Python a to ze stejných důvodů:

- Jednoduchost a čitelnost řešení
- Multiplatformovost – možnost běhu na různých operačních systémech i kompilace do binárních souborů pro ně určené
- Rychlost vývoje aplikace, apod.

Pro vytvoření tohoto tlustého klienta jsou použity následující knihovny (kromě standardních knihoven jazyka Python):

- Twisted – komunikace se serverem i s případnými lokálními zdroji dat
- PySide – pro návrh a zpracování grafického uživatelského rozhraní
- Pygame – pro zobrazení dat v podobě grafických prvků a animací
- lxml – pro vytváření XML zpráv v jejichž formě jsou data přenášena mezi zařízeními
- pyYaml – pro serializaci a práci s lokálními konfiguračními soubory

Všechny tyto knihovny jsou detailněji popsány v kapitole 3.2.1. Jejich výhodou je opět multiplatformost a také licence umožňující i komerční použití, tedy takto postavená aplikace najde použití jak v open–source, tak komerční sféře.

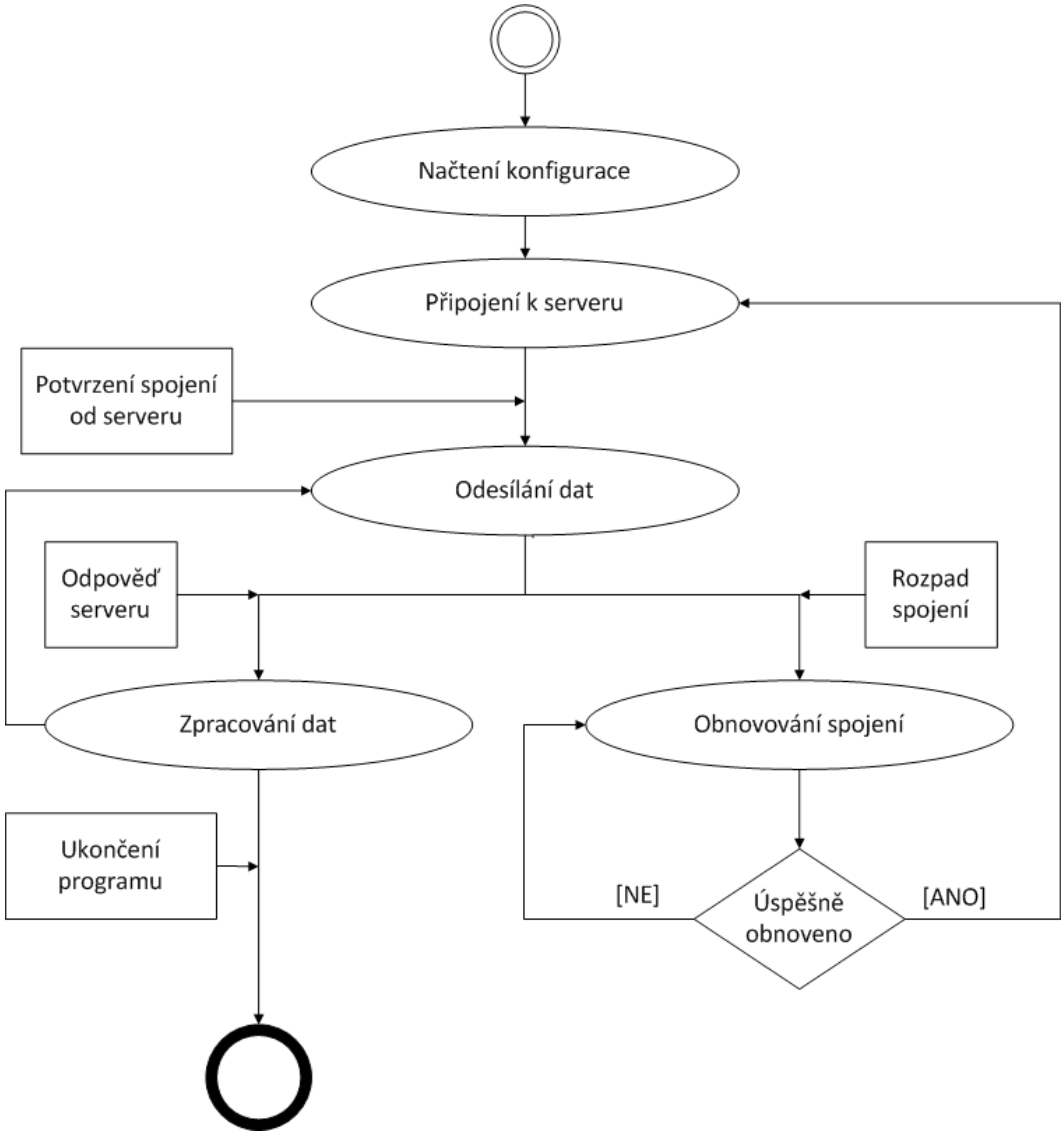
Podrobný popis vybraných programovacích prostředků je detailně popsán v následujících kapitolách se zaměřením na jejich konkrétní využití pro specifické účely – grafické uživatelské rozhraní, komunikace, vizualizace dat.

3.7.2 Jádro a komunikace

Základem klientské části aplikace je Qt4reactor, mezispojení knihoven Twisted a PySide, které umožňuje hlavní třídě aplikace odvozené od třídy ReconnectingClientFactory pro tvorbu klientů z Twisted spojit svoji vlastní komunikační smyčku se smyčkou grafického uživatelského rozhraní z PySide.

Použití tohoto Qt4reactoru tedy umožňuje jedné třídě uživatelskou interakci v grafickém prostředí, stejně jako využívání nativních funkcí ReconnectingClientFactory (viz obrázek 16)

- inicializace komunikací na vybrané síťové adresy a porty
- automatické pokusy o opětovné připojení v případě výpadku, oznámení důvodu výpadku
- možnost spojení a přenosu dat v samostatných vláknech
- použití vlastních časovačů pro vytváření cyklického odesílání dat, apod.



Obrázek 16: Základní model funkčnosti komunikace tlustého klienta

3.7.3 Uživatelské rozhraní

K interakci s uživatelem je vytvořeno grafické uživatelské rozhraní pomocí knihovny PySide, popsané v kapitole 3.2.1, používající prvky frameworku Qt, o kterém pojednává kapitola 3.2.5.

PySide rozlišuje tři základní prvky uživatelského prostředí:

- QMainWindow – Pro vytvoření základního uživatelského okna. Poskytuje framework pro vytvoření prvků jako menu, nástrojových a stavových lišt (ať už pevných či plovoucích) a dalších prvků častých pro základní okno aplikací.
- QDialog – Pro vytvoření jakýchkoliv samostatných oken – základní stavební kámen všech odvozených prvků typu MessageBox pro notifikaci uživatelů, FileDialog pro výběr souborů a podobně.
- QWidget – Seskupení prvků, které lze použít nejen jako samostatné okno, ale i jako dynamická součást jiného okna typu QDialog, QMainWindow či QWidget.

Vytváření uživatelského prostředí je možné jednak čistě pomocí kódu, nebo také pomocí programů určených k vytvoření Qt uživatelského rozhraní, například Qt Creatoru či Qt Designeru. Tato prostředí nabízejí všechny standardní prvky pro uživatelskou interakci:

- Rozvržení – horizontální i vertikální zarovnávače, mřížky, apod.
- Tlačítka – klasická, přepínače, zaškrtávací pole, boxy pro dynamicky propojená tlačítka, atd.
- Seznamy položek – stromové struktury, tabulky, seznamy, sloupcové přehledy.
- Seznamy ovládacích prvků – stejné jako seznamy položek, ale pro specifické položky.
- Oblasti a rámce – skupinový box, oblast s posuvníkem, oblast se záložkami, panely nástrojů, ohraničené oblasti a podobně.
- Prvky pro vstupy dat – kombinovaná pole, textové rámce, číselníky, prvky pro výběr času/data, posuvníky, šoupátka, atd.
- Zobrazovací prvky – Popisky, grafické rámce, video rámce, dělící čáry, kalendář, LCD panel, ukazatele průběhu, webové prohlížeče a další.

Tyto prvky ovšem nejsou vše, co může framework Qt skrze knihovnu PySide nabídnout. Mimo tyto existuje množství dalších předdefinovaných dialogů pro specifické funkce, jako je otevírání, výběr a ukládání souborů, tisk souborů či textu, výběr barev, ale také pop–up okna a menu a další podobné.

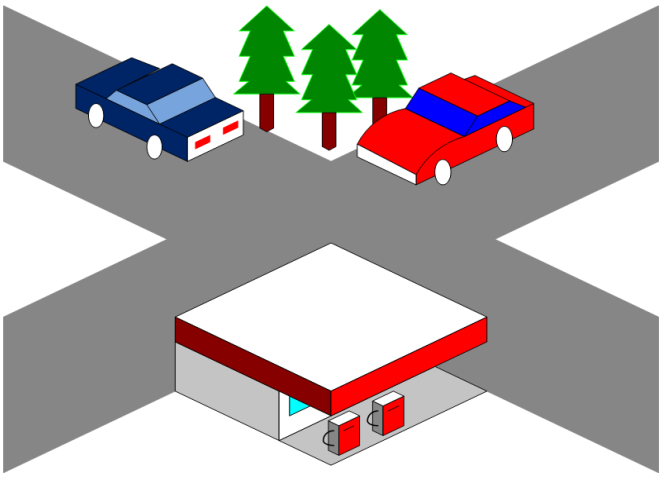
Z těchto prvků lze sestavit téměř jakékoliv grafické rozhraní pro všechny typy aplikací a přizpůsobit ho konkrétní potřebě – od databázových klientů po počítačové hry. Navíc velkou výhodou je také schopnost Qt přiřadit každému prvku list stylu CSS, tedy veškeré prvky mohou být upraveny (uskupeny, barveny, tvarovány, apod.) stejně jako prvky zobrazené na internetových stránkách pomocí CSS stylů.

3.7.4 Vizualizace

Pro účely zobrazování dat je používána knihovna pygame, popsaná v kapitole 3.2.1. Tato knihovna je uzpůsobena k tvorbě animací a počítačových her, má tedy poměrně obsáhlé možnosti.

Základní prvky pro vytvoření jakékoliv vizualizace:

- display – Součást pygame vytvářející obrazovku, do které se všechny animace a prvky vykreslují. Této pak mohou být specifikovány další módy – hardwarová akcelerace, openGL, dvojité bufferování, celoobrazovkový režim, možnost dynamické změny velikosti, apod.
- Surface – Povrch pro vykreslování obrázků, barev a textu.
- Sprite – Jakákoliv aktivní položka vizualizace – od této třídy se mohou odvodit vlastní třídy pro vykreslování prvků. Pro efektivní funkčnost by měl mít každý Sprite vnitřní proměnnou image, ve které je uchováván objekt Surface, který znázorňuje vizualizovaný obrázek tohoto prvku Sprite, a proměnnou position, která uchovává jeho aktuální pozici. Navíc se mohou objekty typu Sprite sdružovat do skupin, které pak usnadňují jejich aktualizaci a pohyb a také umožňují uchovávat různé skupiny v různých vrstvách zobrazení pro pokročilejší zobrazovací techniky.



Obrázek 17: Model pro objasnění funkčnosti prvků pygame

Například pro dynamické zobrazení modelu na obrázku 17 by bylo třeba:

- Surface pro vykreslování s pozadím silnice
- 3xSprite typu Strom
- 2xSprite typu Auto
- 1xSprite typu budova

S knihovnou Pygame lze docílit nejen vrstvených 2D vizualizací, ale i mnohem pokročilejších prostorových zobrazení, proto je tento způsob vhodný pro širokou škálu účelů.

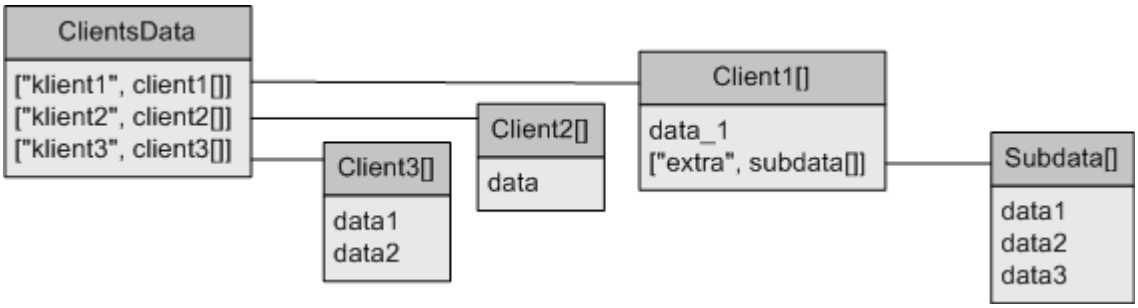
3.8 Komunikace

Ke komunikaci mezi klientem a serverem jsou používány zprávy používající strukturu XML a také běžné datové typy seznamy a mapy. K vytvoření XML struktury je použita knihovna lxml, která má mimo jiné i tu výhodu, že dokáže sama převádět xml na textové řetězce a zpět, usnadňuje tedy i přenášení zprávy mezi jednotlivými zařízeními.

3.8.1 Struktura přenášených dat

Tato struktura je navržena především k přenosu dat, zvláště pak polí reprezentujících klienty a jejich víceúrovňová data, adresářovou hierarchii pro získávání souborů ze serveru, atd.

Tedy před vytvořením zprávy se očekávají data ve formě pole či mapy nebo jejich kombinace (v pythonu seznam-list a slovník-dictionary) o libovolné hloubce, strukturovaná například ve formátu ukázaném na obrázku 18.



Obrázek 18: Model vstupních dat pro vytvoření zprávy

3.8.2 Princip vytvoření zprávy

- Vstupní datová struktura je jednoduše serializována pomocí knihovny pyyaml do jednoho textového řetězce čitelného jak strojově tak člověkem
- struktura YAML je převedena na datový formát Base64, aby případné nestandardní znaky nebránily vytvoření XML souboru; tato struktura je jednolitý blok ASCII znaků, nečitelný člověkem
- K textovému řetězci je přidělena hlavička a případné další údaje a je vytvořena XML zpráva, obsahující překódovanou YAML strukturu jako textový atribut. Tato zpráva může mít více podob, od jednoduchého XML s jedním tagem po víceúrovňové XML s množstvím nadstandardních informací (viz obrázek 19).

```
<?xml version=“1.0“ encoding=“ASCII“ ?>
<header>
   Base64 encoded YAML structure
</header>
```

Obrázek 19: Nejjednodušší možná odeslaná zpráva

3.8.3 Princip odeslání a příjmu zprávy

- Vytvoření zprávy metodou popsanou v kapitole 3.8.2.
- Zkontrolování, jestli není zpráva příliš dlouhá
- Pokud je zpráva příliš dlouhá, je před odesláním rozdělena na menší, kde každá z těchto menších zpráv je označena číslem určujícím pořadí v kompletní zprávě. Poslední část zprávy dostává mimo pořadí také nečíselnou značku určující konec zprávy.
- Zprávy (zpráva) jsou odeslány pomocí vybraného protokolu.
- Po příchodu je zkontrolována hlavička a značky zprávy, v případě rozdělených zpráv je první došlý text udržován v bufferu a další příchozí zprávy jsou k této přidávány na příslušná místa daná značkou pořadí. Po příchodu poslední části je zpráva zkompletována.
- XML struktura zkompletované zprávy je parsována.
- XML text je překódován z base 64 na strukturu typu YAML, z které je následně načtena odpovídající struktura
- Podle XML hlavičky je provedena odpovídající akce, zpracovávající data z datové struktury

3.8.4 Zabezpečená komunikace

Veškerá probíhající komunikace mezi klienty a serverem musí být zabezpečená, aby nemohlo dojít k připojování neoprávněných klientů v případě ztráty hesel, apod. Zabezpečení zpráv je prováděno použitím vygenerovaného SSL certifikátu, který se musí shodovat pro serverovou i klientskou část. SSL certifikát slouží k asymetrickému šifrování zpráv. Při automatických aktualizacích se pak bere v úvahu i případné změna certifikátu, který je zabudován v tlustém klientovi, aby nemohl být použit odděleně.

4 Praktická aplikace

Tento systém byl otestován jako součást softwarové výbavy samovýčepních restaurací pro sdílení výtočí jednotlivých stolů a restaurací mezi ostatními restauracemi v řetězci. Požadavky na tuto aplikaci vyhovují požadavkům sepsaným v kapitole 2.1.

4.1 Popis aplikace

Každá z restaurací v řetězci je vybavena jedním klientem, který komunikuje se vzdáleným serverem, odesílá lokální data a přijímá data potřebná k právě zobrazovanému módu. Tyto data následně zobrazuje v režimech, které jsou předem definované uživatelem a které se cyklicky opakují. Tyto projekce dat jsou vysílány na vybrané televize či projektory v restauraci.

4.2 Serverová část

Na dedikovaném serveru je nainstalován operační systém Debian 6 Squeeze, na kterém je jako démon spuštěna serverová aplikace navržená a vytvořená podle obecného modelu popsaného v kapitole 3.3.

4.2.1 Popis serverové aplikace

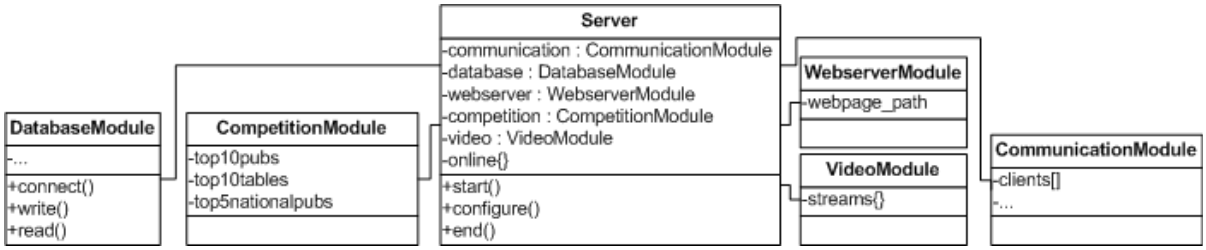
Při spuštění počítače je automaticky spuštěna serverová část aplikace. Ta po svém startu inicializuje všechny používané moduly (viz obrázek 20) a informace z konfiguračního souboru a současně vytvoří struktury pro lokální data.

Používané moduly:

- Komunikace – spravuje komunikaci s klienty a přesměrování dat v rámci serveru
- Soutěž – pro uložení příchozích dat klienta do připravených struktur a odeslání dat klientu podle módu, který klient právě používá
- Databáze – pro přístup do databáze, načítání a ukládání dat
- Webserver – pro zobrazení soutěžních dat prostřednictvím webových prohlížečů
- Video – pro zprostředkování video přenosu z jednotlivých restaurací

Každý modul je samostatně fungující celek bez závislosti na ostatních modulech, je tedy možné moduly zapínat a vypínat, odstraňovat nebo přidávat nové a to bez ztráty funkčnosti zbylých částí serveru.

Po inicializaci aktuálně používaných modulů je serverová aplikace připravena k činnosti a k obsluze klientských požadavků.

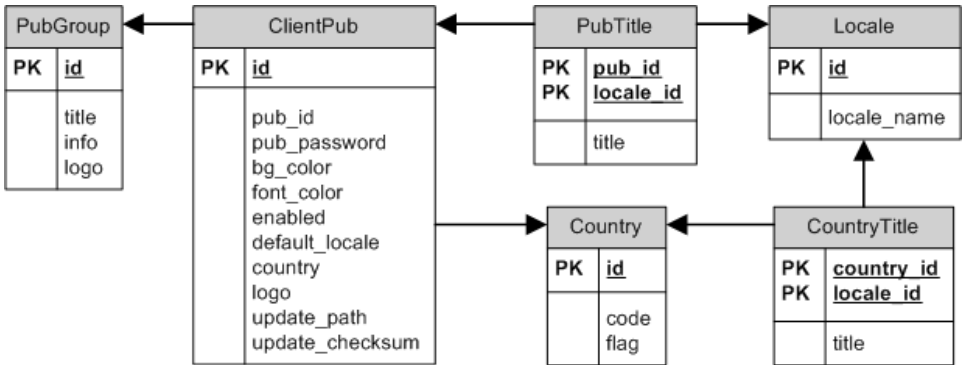


Obrázek 20: Struktura modulů serveru

4.2.2 Struktura databáze a klientských dat

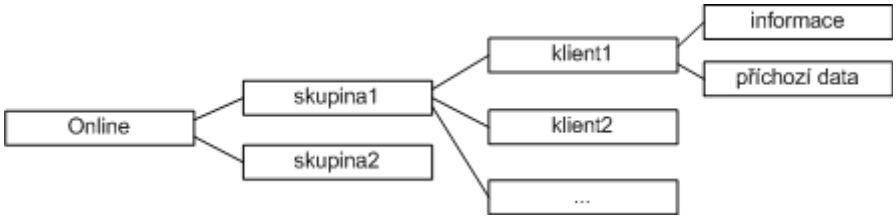
Nejobecnějším prvkem databáze je skupina, která obsahuje klientské restaurace. Skupina formálně reprezentuje řetězec restaurací nebo restaurace propojené kvůli vzájemnému soutěžení. Jednotlivé restaurace jsou pak reprezentovány záznamy v tabulce ClientPub, obsahující přihlašovací informace klienta, informace o jeho vizuální podobě v ostatních restauracích, možných aktualizacích a podobně (viz obrázek 21).

Je zde také použita tabulka obsahující potřebné lokalizace a tabulky s lokalizacemi názvů restaurací a zemí, aby mohli jednotliví klienti zobrazovat názvy podle svých jazykových nastavení.



Obrázek 21: Struktura podpůrné databáze

Mimo přednastavených klientských dat je také použita stromová struktura **online** (struktura Slovník jazyka Python – sdružuje dvojice typu klíč a hodnota) udržující informace o aktuálně připojených klientech a jejich lokálních výtočích (viz obrázek 22).

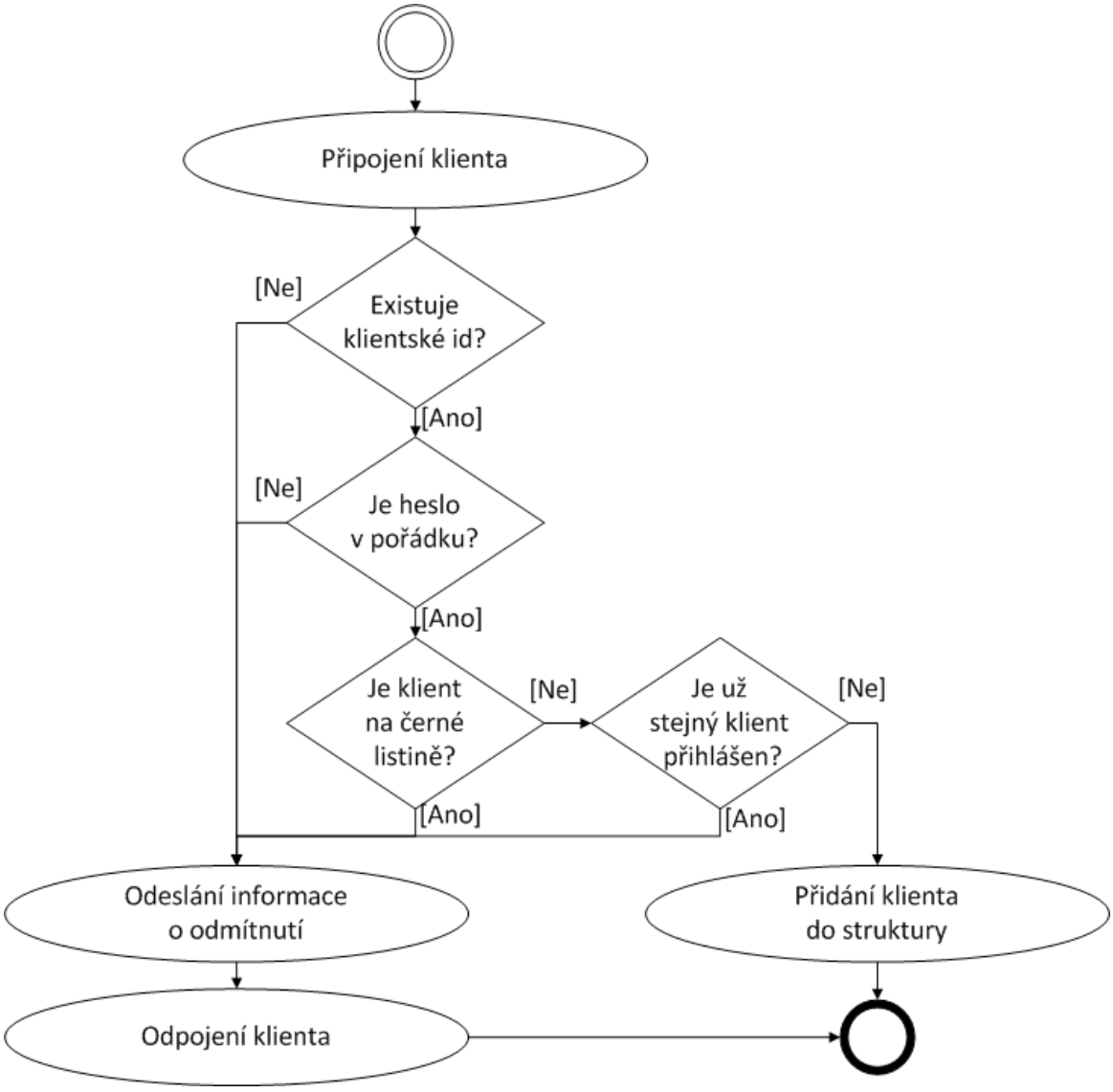


Obrázek 22: Struktura obsahující aktuální informace

4.2.3 Přihlášení klienta

Po připojení klienta k serveru prověřuje server množství informací (viz obrázek 23):

- zda existuje klient s tímto přihlašovacím jménem
- zda klient odesílá správné heslo
- zda není klient na černé listině
- zda už není přihlášen stejný klient



Obrázek 23: Proces přihlášení klienta k serveru

4.2.4 Webový server

Při inicializaci serverové části aplikace je automaticky spuštěn i modul webového serveru. Jeho jádro také využívá reaktor knihovny Twisted, tedy zatímco serverová část pro komunikaci s tlustými klienty naslouchá na vybraném portu, webový server naslouchá na druhém, administrátorsky vybraném portu, aby mohl obsloužit požadavky tenkých klientů (webových prohlížečů).

Základem webového serveru je knihovna websocket, která zprostředkuje použití webových soketů, což je součást standardu HTML5.

HTML stránky jsou na webovém úložišti uchovávány jednotlivě pro každou restauraci a skupinu ve stromové struktuře kopírující strukturu proměnné online (viz kapitola 4.2.2), tedy na lokálním úložišti jsou složky dělené do skupin a ty se dále dělí na složky podle restaurací.

Při zápisu klientských dat do serverových struktur dochází současně také k přepsání jednoduchého HTML souboru, který je při požadavku webového prohlížeče odeslán (viz obrázek 24).

Všechny vygenerované HTML soubory používají stejný CSS soubor pro zobrazení stylů.

```
<!DOCTYPE html>
<html>
 <header>Text záhlaví
  <link rel="stylesheet" href="style.css" type="text/css"/>
 </header>
 <body>
  <div id="layout">
   <div class="title">Název restaurace</div>
   <div class="floor">Patro
    <ul>
     <li class="table">
      <b class="number">1</b><a class="drain">6.3</a>
     </li>
     <li class="table">
      <b class="number">4</b><a class="drain">5.1</a>
     </li>
    </ul>
   </div>
  </div>
 </body>
 <footer>Text zápatí</footer>
</html>
```

Obrázek 24: Struktura vygenerovaného HTML souboru

4.3 Tlustý klient

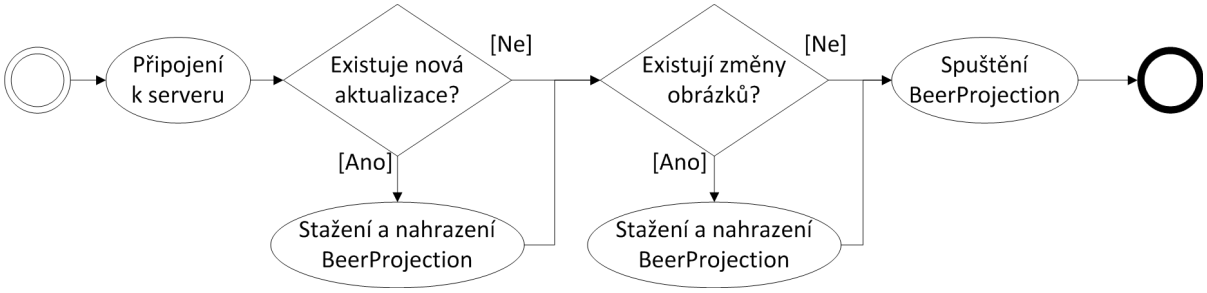
Tlustý klient je zkompilován pro operační systém Windows, který je nainstalován na počítači používaném i pro jiné účely restaurace.

4.3.1 Popis klienta

Tlustým klientem je myšlena dvojice spustitelných (.exe z *angl.* executable) souborů *updater.exe* a *BeerProjection.exe* (viz obrázek 25) obsahující všechny potřebné knihovny a základní obrázky potřebné ke svému běhu.

Obrázek 25: Ikona spustitelných souborů

Program je spouštěn souborem updater.exe, který se připojuje k serveru, zkontroluje dostupné aktualizace a případně nahradí soubor BeerProjection.exe novou verzí. Současně také kontroluje aktuálnost obrázků používaných k vizualizacím (loga měst, vlajky států) a nahrazuje zastaralé obrázky či stahuje nové. Po těchto kontrolách spouští druhý soubor – *BeerProjection.exe* (viz obrázek 26).



Obrázek 26: Model funkce programu updater

Po spuštění BeerProjection dochází k načtení zašifrované lokální konfigurace obsahující přihlašovací jméno a heslo a lokální nastavení vizualizací.

Tato část klienta zajišťuje několik základních funkcí:

- sběr a odesílání lokálních dat
- příjem vizualizačních dat ze serveru
- vizualizace lokálních i vzdálených dat na libovolnou připojenou obrazovku
- psaní automaticky mizejících zpráv do právě spuštěného vizualizačního módu
- promítání obrázků na libovolnou připojenou obrazovku (včetně automatické funkce promítání)
- přehrávání videí na libovolnou připojenou obrazovku (použitím externího přehrávače)

4.3.2 Smyčka programu

Jádrem programu je smyčka poskytovaná knihovnou Qt4reactor, jak už bylo zmíněno v návrhu aplikace v kapitole 3.7.2. Qt4reactor zajišťuje splynutí zobrazovací smyčky uživatelského rozhraní knihovny PySide i reaktoru knihovny Twisted, který je jádrem spravujícím smyčku událostí a používaná vlákna všech komunikačních částí. Zároveň je tento reaktor používán i k obnovování vizualizační obrazovky, jelikož by bylo zavádění nové smyčky knihovny pygame zbytečné (viz obrázek 27).

Používané funkce a třídy v rámci reaktoru:

- connectTCP() – lokální připojení pro sběr dat
- connectSSL() – připojení k serveru využívající zabezpečenou komunikaci
- callInThread(funkce) – spustí vybranou funkci v samostatném vlákně
- LoopingCall(funkce) – vytvoří třídu, která cyklicky spouští vybranou funkci

Obrázek 27: Použití knihovny Qt4reactor pro jednotlivé části klienta

4.3.3 Komunikace se serverem

Klientská komunikace je podobná komunikaci serverové částí. Při spuštění programu inicializuje zabezpečenou komunikaci s použitím zabudovaného certifikátu. Při úspěšném spojení odešle svoje přihlašovací informace a je serverem přijat nebo odmítnut v závislosti na odeslaných informacích jak je popsáno v kapitole 4.2.3.

V případě úspěšného připojení získá od serveru informace o všech ostatních klientech ve skupině – o barvách, lokalizovaných názvech a logu každého klienta, aby ho mohl zobrazit v případě, že bude tento klient součástí přijatých vizualizačních dat.

Dále už tlustý klient pouze odesílá zprávy obsahující aktuální data. Tyto zprávy mají různé nadpisy XML dokumentu, které určují, jaká data bude server klientu zpět odesílat. Tyto nadpisy reprezentují jednotlivé soutěžní módy, které také vyžadují data ostatních klientů ze stejné skupiny připojených k serveru. Zprávy mají tvar popsaný v kapitole 3.8.

4.3.4 Sběr lokálních dat

Lokální data mohou být získávána ze dvou různých hardwarových zařízení:

- z osobního počítače – několik možných programů s různou strukturou dat
- z programovatelného logického automatu

Z důvodu rozmanitosti přijímaných dat je nutné nastavit používaný zdroj dat, podle kterého je vybrána třída zajišťující lokální komunikaci. Všechny tyto komunikační třídy jsou opět založeny na knihovně Twisted a zabezpečují přenos a převod dat do požadované datové struktury určené k vizualizaci.

4.3.5 Datové struktury

Klient má speciální datovou strukturu pro každý vizualizační mód. Tyto individuální struktury obsahují jen ty informace, které jsou nutné k vizualizaci daného módu. Toto rozdělení zmenšuje délku zprávy přenášené ze serveru, protože zpráva vždy obsahuje jen data potřebná k zobrazení aktuálního vizualizačního módu.

Klient obsahuje tyto hlavní datové struktury (viz tabulka 4):

- top10tables – Nejlepších deset stolů ve skupině
- top10pubs – Nejlepších deset restaurací ve skupině
- top10local – Nejlepších deset lokálních stolů
- top5nationalpubs – Nejlepší restaurace v pěti zemích

Všechny struktury jsou typu seznam (pole) obsahující strukturu slovník (mapa) reprezentující jednotlivé stoly a restaurace.

Modely použitých datových struktur			
název	**typ struktury**	**typ vnitřní položky**	**klíče a hodnoty**
top10tables	seznam (pole)	slovník (mapa)	**"title"** = název restaurace **"custom_title"** = individuální název stolu **"drain"** = celková výtoč stolu **"pub_id"** = ID restaurace **"number"** = číslo stolu
top10local	seznam (pole)		
top10pubs	seznam (pole)	slovník (mapa)	**"title"** = název restaurace **"drain"** = celková výtoč restaurace
top5nationalpubs	seznam (pole)	slovník (mapa)	**"title"** = název restaurace **"drain"** = celková výtoč restaurace **"country"** = ID země

Tabulka 4: Struktury dat použitých pro vizualizaci

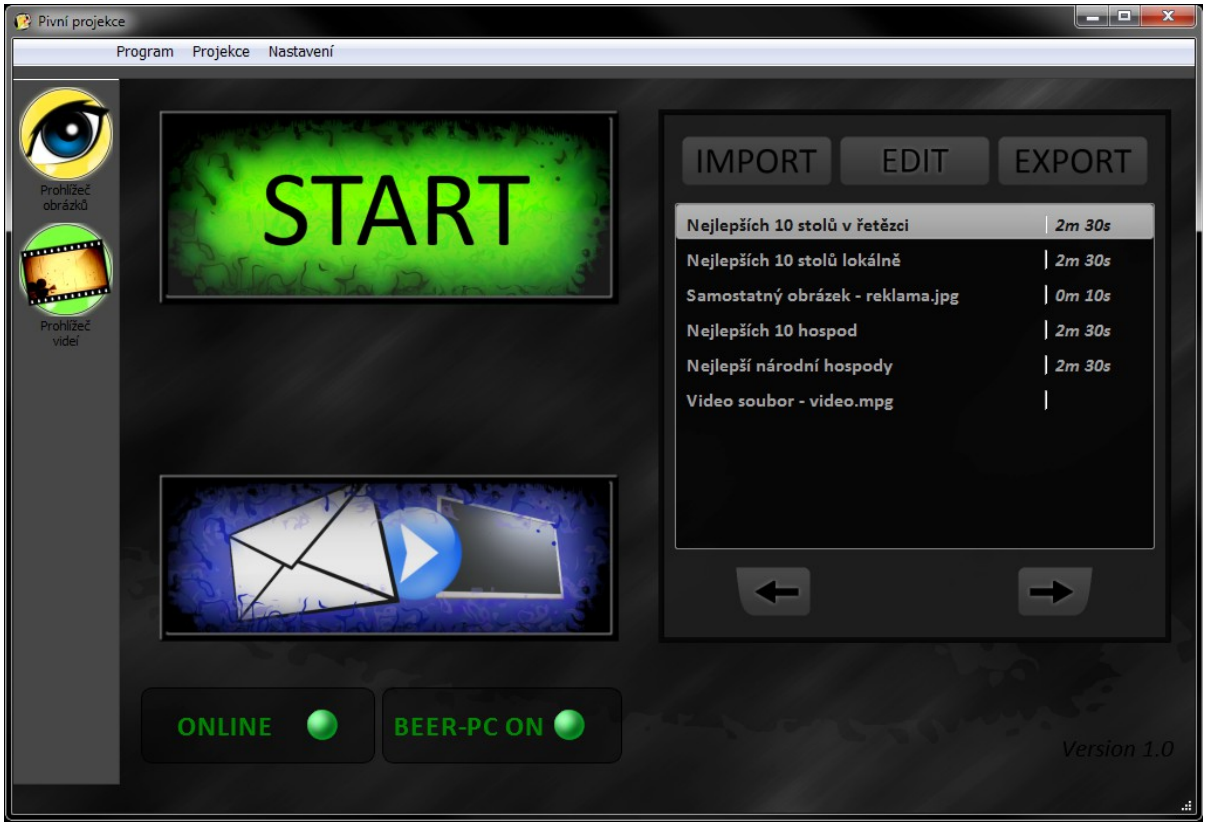
4.4 Uživatelské rozhraní

Základní prvky pro ovládání spouštění, vypínání a přepínání vizualizačních módů jsou umístěny přímo na hlavní obrazovce. Nastavení programu a úpravy vizualizace se dále provádějí z menu v horním panelu programu. Speciální funkce pro promítání složky obrázků a samostatných video souborů jsou umístěny v nástrojovém panelu v levé části hlavní obrazovky.

4.4.1 Hlavní obrazovka

Hlavní obrazovka je rozčleněna na logické celky (viz obrázek 28):

- horní panel – menu pro nastavení programu a vizualizací
- levý nástrojový panel – speciální funkce promítání
- dolní panel – stavový řádek oznamující problémy
- centrální oblast – slouží ke standardní vizualizaci:
  - levá část – spouštění/zastavení vizualizace a posílání zpráv
  - pravá část – seznam módů k přehrání (playlist), práce s ním a přepínání módu



Obrázek 28: Hlavní obrazovka tlustého klienta

4.4.2 Menu

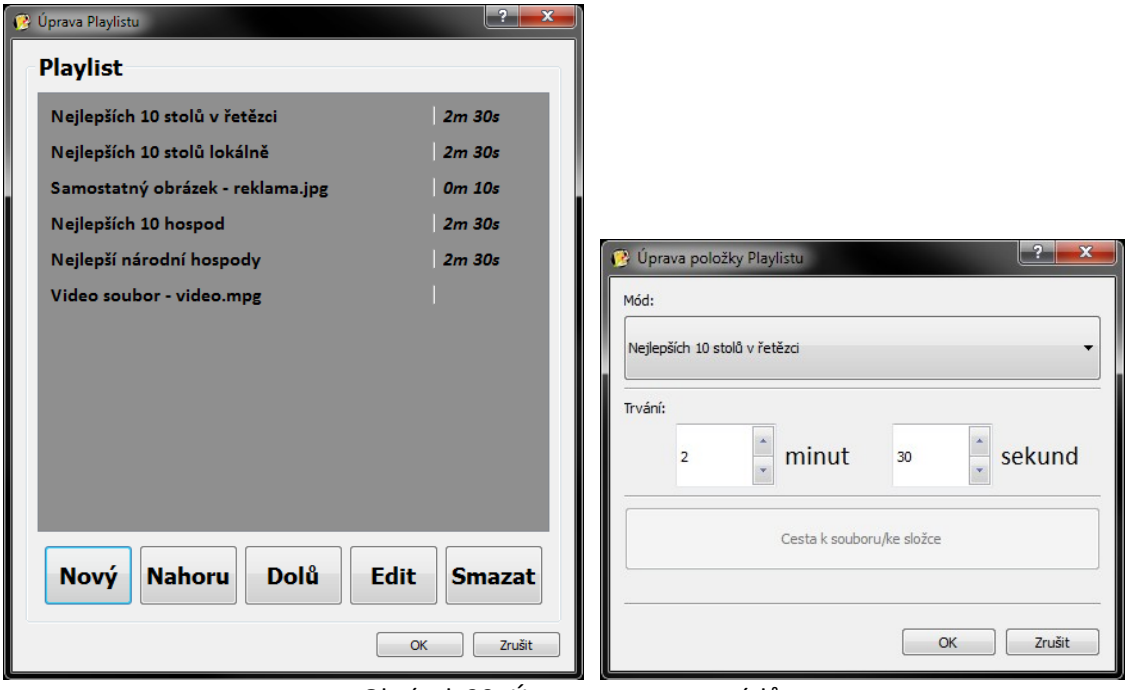
Menu slouží k nastavení programu, vizualizace, jejího vzhledu, apod. Struktura menu je znázorněna v tabulce 5.

Struktura menu		
Program	**Projekce**	**Nastavení**
O programu Ukončit program	Úprava playlistu Loga Projekční obrázky Pozadí Vzhled projekce Názvy stolů	Jazyk Přihlašovací údaje Administrátorské

Tabulka 5: Struktura menu

4.4.3 Úprava playlistu

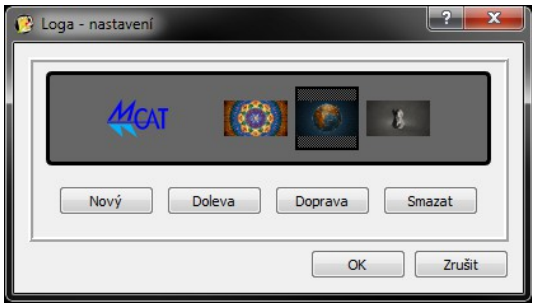
Vybráním položky „Úprava playlistu“ v menu je možné změnit seznam módů k přehrání (viz obrázek 29). Tlačítkem „Nový“ a „Edit“ je vyvoláno okno pro úpravu stávajícího/přidání nového módu. U módů vyžadujících cestu k souboru je automaticky povoleno tlačítko pro vyhledání systémové cesty.



Obrázek 29: Úprava seznamu módů

4.4.4 Loga

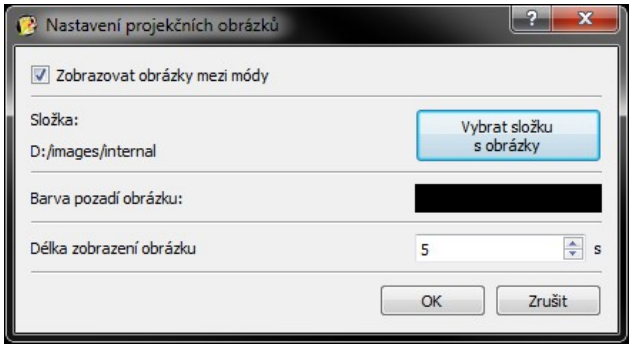
Položka menu „Loga“ slouží k vybrání obrázků sloužících jako reklamní loga na spodní panel spuštěné vizualizace. Pozice načtených log může být změněna pomocí tlačítek „Doleva“ a „Doprava“ (viz obrázek 30).



Obrázek 30: Okno výběru log pro vizualizaci

4.4.5 Projekční obrázky

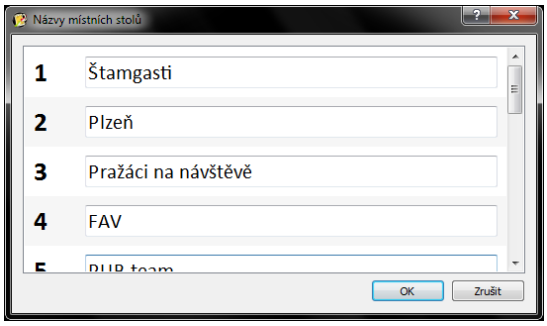
Položka menu „Projekční obrázky“ (viz obrázek 31) zprostředkovává výběr složky, ve které se nacházejí obrázky, které by měly být automaticky zobrazovány mezi vizualizačními módy. Zároveň je zde možné tyto obrázky vypnout, nastavit časový interval a barvu pozadí při jejich zobrazení.



Obrázek 31: Nastavení projekčních obrázků

4.4.6 Názvy stolů

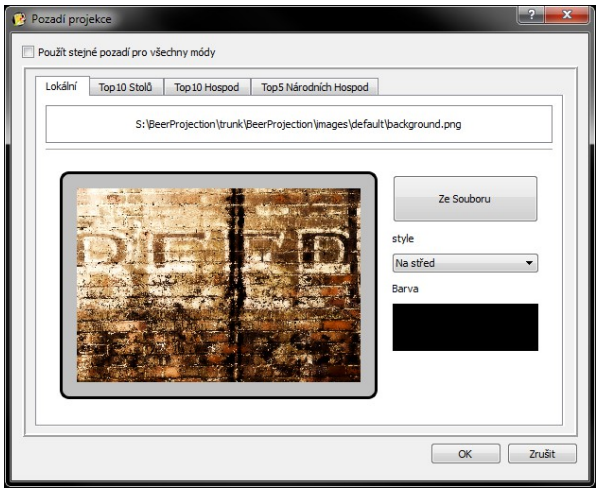
V tomto dialogovém okně (viz obrázek 32) je možné každému stolu v restauraci přiřadit individuální jméno, které se bude následně zobrazovat na vizualizacích všech restaurací ve skupině, které mají povolenou funkci individuálních názvů.



Obrázek 32: Názvy místních stolů

4.4.7 Pozadí projekce

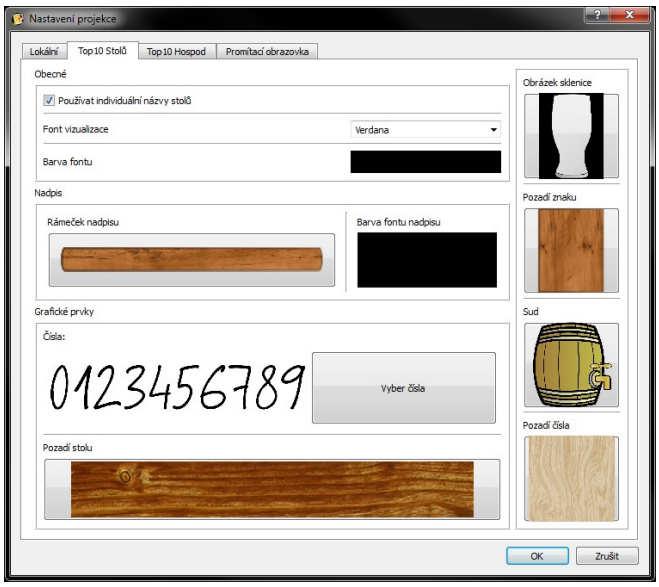
Toto okno (viz obrázek 33) slouží k nastavení pozadí pro vizualizační módy. Vybráním zaškrtávacího pole „Použít stejné pozadí pro všechny módy“ bude použito pouze jediné unifikované pozadí, při deaktivaci tohoto pole má každý dostupný mód samostatné pozadí. Nastavení pozadí sestává z výběru obrázku, barvy pozadí a volby, zda–li má být obrázek roztažen nebo zarovnán na střed.



Obrázek 33: Výběr pozadí projekce

4.4.8 Vzhled projekce

Toto okno (viz obrázek 34) slouží k nastavení všech grafických prvků použitých ve vizualizaci, aby si každý řetězec nebo majitel restaurace mohl nastavit vzhled projekce podle vlastního záměru. Všechny obrázky použité ve vizualizaci je možné měnit pomocí tohoto dialogového okna. Kromě standardních obrázkových formátů jsou podporovány i vektorového obrázky ve formátu SVG (scalable vector graphic), což umožňuje velmi kvalitní vizuální zpracování. V tomto okně se zároveň vybírá cílová obrazovka pro vizualizaci (z aktuálně připojených obrazovek).



Obrázek 34: Vzhled projekce

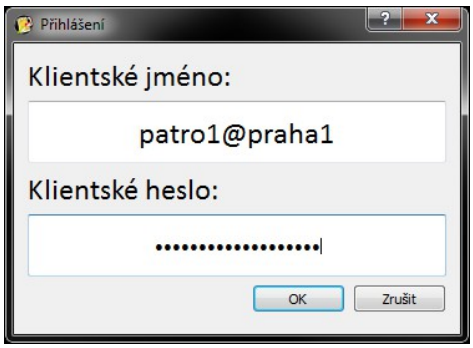
4.4.9 Jazyk

Tlustý klient využívá knihovnu GNU Gettext, je tedy jednoduše lokalizovatelný do jakéhokoliv jazyka. Z dostupných lokalizačních souborů je pak možné zvolit žádaný jazyk vybráním položky „Nastavení/Jazyk“ v menu (viz obrázek 35).

Obrázek 35: Výběr jazyka

4.4.10 Přihlašovací údaje

Okno na obrázku 36 slouží k zadání přihlašovacích údajů. Přihlašovací jméno a heslo se musí shodovat se záznamy serveru.



Obrázek 36: Přihlašovací údaje

4.4.11 Administrátorské nastavení

Okno administrátorského nastavení slouží k uložení adresy a portu serveru, typu místní komunikace, systémové cestě k přehrávači videí, k nastavení údajů místní restaurace pro případ výpadku internetového připojení a také módů, které může konfigurovaná restaurace využívat. Toto nastavení je chráněno heslem a nepřístupné běžnému uživateli.

4.5 Vizualizace dat

Vizualizace dat je nejdůležitějším prvkem tlustého klienta. Jedná se o jeden z prostředků zaměřených na zatraktivnění návštěvy samovýčepní restaurace a podpoře soutěživosti mezi jednotlivými restauracemi. Vizualizační část softwaru je jednoduše rozšiřitelná, což umožňuje přidávání dalších módů vyžádaných majiteli řetězců a restaurací. Data uchovávaná na serveru jsou natolik podrobná, že je možné získávat a zobrazovat výtoče zákazníků, stolů, pater v restauracích, celých restaurací, měst i zemí.

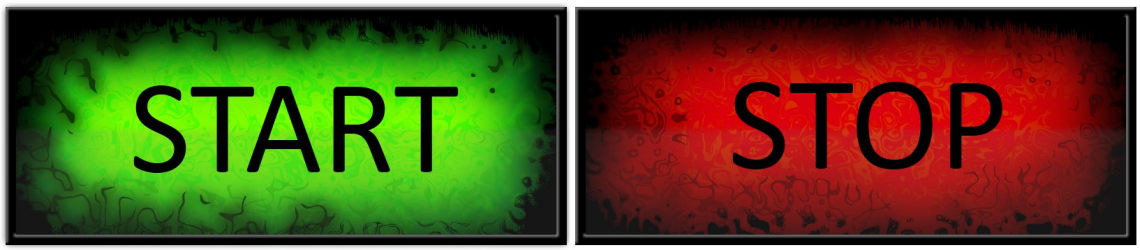
V současné době jsou používány čtyři módy používající informace o výtočích a dva pro reklamní použití:

- 10 Nejlepších stolů v restauraci
- 10 Nejlepších stolů v řetězci
- 10 Nejlepších restaurací
- 5 Nejlepších národních restaurací
- Obrázek – pro automatické přehrání reklamního obrázku
- Video soubor – pro automatické přehrání reklamního video souboru

Okno vizualizace se dělí na tři hlavní části:

- Záhlaví – určeno pro nadpis s názvem daného módu
- Hlavní plochu – určeno pro vizualizaci dat
- Zápatí – určeno pro reklamní loga

Spuštění a zastavení vizualizace je prováděno stiskem zeleného a červeného tlačítka Start a Stop (viz obrázek 37). Po spuštění vizualizace dochází k vytvoření vizualizačního plátna, na které jsou následně nanášeny všechny grafické prvky vázané na konkrétní mód projekce.



Obrázek 37: Tlačítka pro spuštění a zastavení vizualizace

Pořadí módů je dáno seznamem módů k přehrání (playlistem). Aktivní mód je vždy zvýrazněn. Vizualizační módy se přepínají automaticky, případně dvojím poklepáním na jiný mód nebo šipkami (viz obrázek 38) spouštějícími následující nebo předchozí mód projekce.

Obrázek 38: Tlačítka pro přepínání módů

4.5.1 Nejlepších 10 stolů

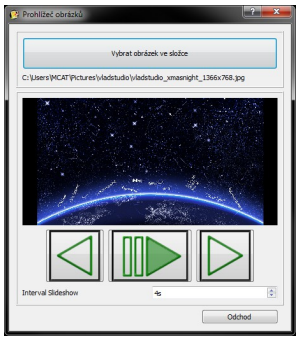
Zobrazovací mód je stejný pro lokální variantu (v restauraci) i pro globální (v řetězci). Sestává z desíti prvků Sprite reprezentujících stoly zobrazujících název restaurace nebo stolu, čísla stolu, celočíselné výtoče znázorněné v půllitrech a desetinné výtoče znázorněné množstvím nápoje ve sklenici (viz obrázek 39). Za každých překročených padesát půllitrů se tato výtoč reprezentuje blikajícím sudem. V pravé straně obrazovky se pak zobrazují loga pěti restaurací na nejvyšších příčkách.

Obrázek 39: Nejlepších 10 stolů

4.5.2 Promítání obrázku

Na vizualizační obrazovku je promítán samostatný obrázek (v rámci módu nebo reklamních projekčních obrázků – viz kapitola 4.4.5) nebo sekvence obrázků mimo módy projekce, například soukromé promítání fotografií, součást propagačních akcí, apod.

V takovém případě není součástí vizualizačního módu záhlaví ani zápatí, ale pouze samotný obrázek. Promítání sekvencí obrázků je prováděno nástrojem „Prohlížeč obrázků“ (viz obrázek 40) z levého nástrojového panelu hlavní obrazovky.



Obrázek 40: Prohlížeč obrázků

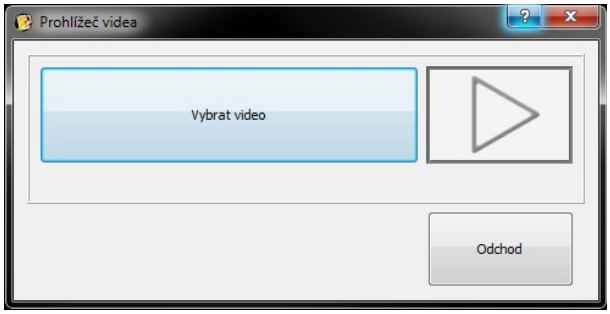
4.5.3 Nejlepších 10 restaurací

V tomto vizualizačním módu jsou zobrazeny restaurace s největší výtočí. Řazeny jsou zleva doprava a výška sloupce na kterém jsou zobrazeny informace je proporcionální podle výtoče vzhledem k výtoči restaurace na prvním místě (viz obrázek 41).

Obrázek 41: Nejlepších 10 restaurací

4.5.4 Promítání videa

Videa mohou být součástí seznamu módů k přehrání, ale stejně tak mohou být na vizualizační obrazovku přehrávána zvlášť, bez nutnosti přidávání do fronty. K tomu slouží nástroj „Prohlížeč videa“ (viz obrázek 42) dostupný v levém nástrojovém panelu hlavní obrazovky. Video je přehráno prostřednictvím externího přenositelného přehrávače, ke kterému jsou nastaveny cesty v administrátorském nastavení.



Obrázek 42: Prohlížeč videí

4.5.5 Nejlepších 5 národních restaurací

Tento vizualizační mód zobrazuje nejlepších pět restaurací v různých zemích. Položky Sprite národní restaurace sestávají z městského loga, názvu restaurace, výtoče restaurace a národní vlajky země, ve které je daná restaurace zaregistrována (viz obrázek 43).

Obrázek 43: Nejlepší národní restaurace

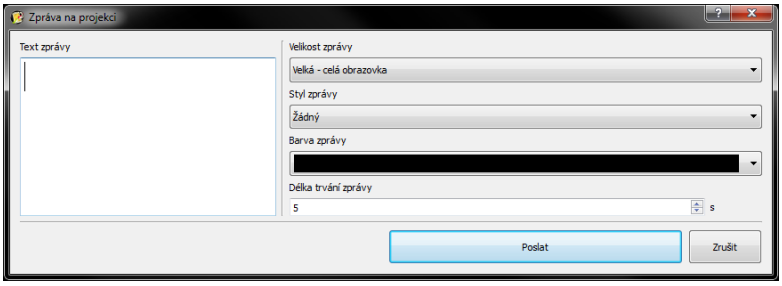
4.5.6 Projekční zprávy

Pomocí modrého tlačítka s motivem zprávy na hlavní obrazovce je možné odesílat vzkazy na vizualizační obrazovku. Tyto vzkazy mohou mít tvar prostého textu, ale současně také různorodou grafickou úpravu a je možné je podle přání majitele restaurace přidávat. Tyto zprávy mohou sloužit jak k pobavení zákazníků tak k propagačním účelům.

Před odesláním zprávy je nutné vyplnit vzkaz, délku zobrazení zprávy, její velikost (¼ obrazovky, ½ obrazovky a celá obrazovka) a barvu (deset předdefinovaných barev pro rychlé zvolení) pro adekvátní viditelnost na současném projekčním módu a také její styl, který určuje grafickou úpravu zprávy (viz obrázek 44)

Styly grafické úpravy využívají kombinaci textu, jeho úprav (rotace, efekty, fonty)a vektorových obrázků, které jsou i při roztažení na celou obrazovku velmi kvalitní.

Po odeslání se zpráva zobrazí na právě zobrazeném vizualizačním módu podle vyplněných nastavení (viz obrázek 45).

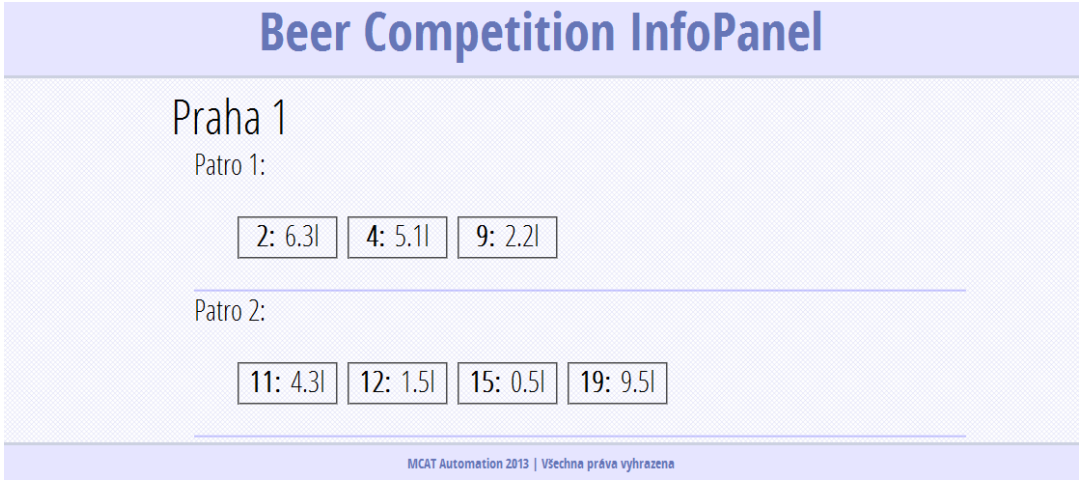


Obrázek 44: Odeslání projekční zprávy

Obrázek 45: Zobrazení projekční zprávy

4.6 Tenký klient

Tenkým klientem je pro tuto aplikaci libovolný webový prohlížeč. Po připojení na webovou adresu serveru (případně specifickou URL dané restaurace), zobrazí se uživateli aktuální výtoče (viz obrázek 46). Tato funkce není dostupná pro zákazníky, ale pouze pro majitele restaurací nebo řetězců, které tak může odkudkoliv informovat o aktuální obsazenosti a spotřebě.



Obrázek 46: Informační přehled tenkého klienta

Závěr

Cílem této práce bylo seznámit se s principy komunikačních aplikací typu klient—server, prostudovat moderní techniky internetového přenosu dat, jejich zabezpečení a možnosti prezentace. Největší část této práce se věnuje návrhu vhodné architektury typu klient—server a implementace této architektury v konkrétní aplikaci pro internetovou komunikaci samovýčepních restaurací se splněním hlavních a vedlejších požadavků.

Úvodní kapitola se zabývá moderními aplikacemi typu klient—server, vysvětluje základní pojmy, které jsou v této práci dále používány, popisuje funkci aplikací tohoto typu včetně možných alternativ a jiných architektur a zároveň prezentuje moderní přístupy k této problematice.

Další kapitola analyzuje stanovené požadavky na jednotlivé části aplikace, porovnává možné klientské varianty i databázové systémy, které se dají k vytvoření žádané architektury použít.

V následující kapitole se přistupuje k samotnému návrhu architektury a dílčích komponent aplikace. Jsou vybrány programovací prostředky, knihovny použité k vytvoření serveru, databáze i klientské části. Je zde nastíněna struktura přenášených dat i principy přenosu zpráv. Dále se tato kapitola zabývá knihovnami určenými k vizualizaci dat a popisuje princip zobrazení použitím vybraných prostředků.

Poslední kapitola obsahuje informace o konkrétní aplikaci vytvořené podle navržené architektury a její využití na problém sdílení a přenosu dat mezi klientskými počítači samovýčepních restaurací. Popisuje serverovou část aplikace, strukturu použité databáze i princip autentizace klienta. Dále se zabývá popisem tlustého klienta, jeho komunikace se serverem i datových struktur, které používá k následné vizualizaci dat. Je zde představeno uživatelské rozhraní a jeho nejdůležitější části, které slouží k obsluze klientské části programu a dalšímu nastavení. Podstatnou částí je také prezentace vizualizace dat, která se dělí podle jednotlivých předprogramovaných módů, promítání obrázků či videí nebo zobrazení zpráv na vybrané projekční obrazovce.

Hlavní i vedlejší cíle práce byly splněny – vytvořená aplikace je multiplatformní i snadno rozšiřitelná a moduly serveru i vizualizační módy jsou konfigurovatelné. Do budoucna je plánován další rozvoj v podobě nových multimediálních módů a hlubší využití webového serveru rozšířeného o zobrazení používající HTML5 canvas a javascript, které by mohlo být využito nejen majiteli restaurací, ale i samotnými zákazníky.

Seznam použité literatury

[1] **Server**[online], [27.4.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Server_%28computing%29*>.

[2] **YADAV, S. Ch., SINGH S. K.**. *An Introduction to Client/Server Computing.*1. vyd. Varanasi: *New Age International Publishers, 2009*

[3] **Client–server model**[online], [8.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Client%E2%80%93server_model*>.

[4] **Client**[online], [8.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Client_%28computing%29*>.

[5] **Vícevrstvá architektura: tenký, tlustý a chytrý klient**[online], [24.5.2010]. Dostupné z: <*http://www.cleverandsmart.cz/vicevrstva-architektura-vyhody-a-nevyhody/*>.

[6] **Thin client**[online], [8.4.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Thin_client*>.

[7] **Webové stránky, jejich historie a technologie**[online], [10.3.2013]. Dostupné z: <*http://silverlight.cs.vsb.cz/01-technologie-a-ria.aspx*>.

[8] **Dr. Ibrahim Khalil**. *Client Server Architecture.*1. vyd. Melbourne: *RMIT University, 2006*

[9] **Fat client**[online], [17.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Fat_client*>.

[10] **Diskless node**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Diskless_node*>.

[11] **Computer network**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Computer_network*>.

[12] **Network Socket**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Network_socket*>.

[13] **KAUR, Jasleen**. *Client/Server Computing & Socket Programming.*1. vyd. Rollins Hall: *Old Dominion University, 2009*

[14] **HTTPS**[online], [13.3.2013]. Dostupné z: <*http://cs.wikipedia.org/wiki/HTTPS*>.

[15] **Database Fundamentals**[online], [1995]. Dostupné z: <>.

[16] **Popular Database Management Systems**[online], [13.3.2013]. Dostupné z: <*http://dbconvert.com/overview.php*>.

[17] **MySQL**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/MySQL*>.

[18] **SQL**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/SQL*>.

[19] **Klient-server**[online], [13.3.2013]. Dostupné z: <*http://cs.wikipedia.org/wiki/Klient-server*>.

[20] **Peer-to-peer**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Peer-to-peer*>.

[21] **Client-queue-client**[online], [13.3.2013]. Dostupné z: <*http://en.wikipedia.org/wiki/Client%E2%80%93queue%E2%80%93client*>.

[22] **BroserStatistics**[online], [30.4.2013]. Dostupné z: <*http://www.w3schools.com/browsers/browsers_stats.asp*>.

[23] **GOCEK, P., HARTMANN, M., SCHLEUSENER, H.**. *Modern Technologies in Client-Server Architecture for Geo-based Interactive Web Portals.*1. vyd. Berlin: *University of Applied Sciences, 2006*

[24] **Adobe confirms Flash player is dead for mobile devices**[online], [2011]. Dostupné z: <*http://www.engadget.com/2011/11/09/adobe-confirms-flash-player-is-dead-for-mobile-devices/*>.

[25] **TIOBE Programming Community Index**[online], [22.3.2013]. Dostupné z: <*http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html*>.
[26] **Programming Language Popularity**[online], [30.4.2013]. Dostupné z: <*http://www.langpop.com/*>.
[27] **Python**[online], [22.3.2013]. Dostupné z: <*http://cs.wikipedia.org/wiki/Python*>.
[28] **PRECHELT, L.**. *An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl.*1. vyd. Karlsruhe: *Universitat Karlsruhe, 2000*
[29] **SuccesStories**[online], [22.3.2013]. Dostupné z: <*http://twistedmatrix.com/trac/wiki/SuccessStories*>.
[30] **Event-Driven Programming with Twisted and Python**[online], [22.3.2013]. Dostupné z: <*http://www.linuxjournal.com/article/7871*>.
[31] **CSS**[online], [22.3.2013]. Dostupné z: <*http://www.w3.org/Style/CSS/*>.

Seznam ilustrací

Seznam tabulek