VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

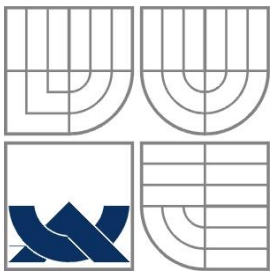
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ SLUŽBA PRO EFEKTIVNÍ KOMUNIKACI SKUPINY UŽIVATELŮ

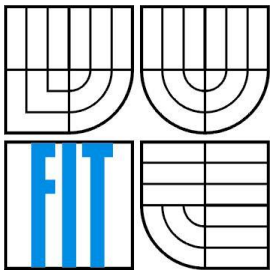
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Petr Koláček
AUTHOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ SLUŽBA PRO EFEKTIVNÍ KOMUNIKACI SKUPINY UŽIVATELŮ

WEB SERVICE FOR EFFECTIVE COMMUNICATION OF GROUP OF USERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Petr Koláček
AUTHOR

VEDOUCÍ PRÁCE — Ing. Beran Vítězslav, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato bakalářská práce je zaměřena na návrh a realizaci knihovny „Chat of future“. Výsledkem práce je služba, která umožňuje uživateli na svých internetových stránkách vytvořit jednoduchý a efektivní nástroj pro efektivní komunikaci skupiny uživatelů.

Abstract

This bachelor’s thesis is focused on design and implementation of web library „Chat of future“. The result of this work is service, which allows user to create an easy and effective communication tool on his web pages, used for effective communication of group of users.

Klíčová slova

Síť, internet, nástroj, služba, knihovna, komunikace, efektivní komunikace, chat, uživatel, skupina, skupina uživatelů.

Keywords

Web, internet, tool, service, library, communication, effective communication, chat, user, group, group of users.

Citace

Koláček Petr: Webová služba pro efektivní komunikaci skupiny uživatelů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Webová služba pro efektivní komunikaci skupiny uživatelů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Koláček
14. května 2014

Poděkování

Děkuji vedoucímu práce za výborný přístup k plánování práce a odbornou pomoc při řešení problémů a za přínosné konzultace. Rovněž děkuji svým rodičům, přátelům a přítelkyni za podporu.

Obsah

1 Úvod

Většina odvětví informačních technologií je v dnešní době rychle vyvíjena. Současné technologie jsou zlepšovány a zefektivňovány a stále více a rychleji se objevují technologie nové. Kladou čím dál větší důraz na uživatelskou přívětivost, intuitivnost a jednoduchost. Tyto vlastnosti jsou pro uživatele moderních technologií velice důležité. Pokud jsou, nejlépe všechny současně, při vývoji informačních technologií dodržovány, výsledné produkty se poté vyznačují vysokou kvalitou zpracování a zpravidla mívají kladné uživatelské ohlasy.

Rozvoj informačních technologií má také obrovský dopad na technologie pro tvorbu webových stránek a aplikací. Vývojáři těchto technologií se snaží oprostit uživatele od používání prostředků pro tvorbu webových stránek starým způsobem, snaží se jim zjednodušovat a urychlovat práci. Za tímto účelem vznikly různé modely, návrhové vzory, frameworky, knihovny, šablony a další nástroje.

Dokument se zabývá vytvořením jednoduchého komunikačního nástroje, služby, sdružující v sobě prvky různých on-line komunikačních forem. Nástroj je nazván *Chat of future*. Tento nástroj je realizován jako knihovna, kterou si může uživatel integrovat do své webové stránky a tím jednoduše vytvořit komunikační kanál například pro zaměstnance své firmy.

Projekt je zaměřen na komunikaci mezi uživateli či vývojáři, tedy na tvorbu komunikačního nástroje. Aplikace je rozdělena na dva celky. První z nich je „frontend“, popředí, což je uživatelské rozhraní pro komunikaci. S tímto rozhraním operují uživatelé, kteří knihovnu využívají. „Backend“, pozadí aplikace, je server, na kterém běží aplikační rozhraní (API) a který se stará a poskytuje data, se kterými knihovna pracuje.

Obsah technické zprávy je rozdělen do částí, ve kterých jsou popsány teoretické postupy a techniky při návrhu internetových komunikačních nástrojů, dále konkrétní návrh komunikačního nástroje s následným popisem způsobu jeho realizace. Po přečtení této dokumentace by měl čtenář chápat, co výsledná aplikace dělá, jak funguje, k čemu je vhodná a jaké jsou její výhody a nevýhody. Dále by měl být seznámen s použitými technologiemi a postupy, které byly při vývoji použity či nepoužity, a z jakých důvodů. Také mu bude předložen průběh experimentů a testování aplikace, a jeho výsledky.

2 Teorie

Tato kapitola se zabývá studiem podkladů a technologií pro realizaci aplikace. Jelikož se aplikace zaměřuje na tvorbu uživatelského rozhraní pomocí moderních webových technologií, teoretické studium se rozdělilo na dvě části. Dále jsou pak v kapitole popsány různé druhy forem on-line komunikace, jejich způsoby řešení a výběr těch prvků, které je vhodné využít pro realizaci mého komunikačního nástroje.

První částí je samotné pochopení funkcionality klasických nástrojů pro tvorbu webových aplikací jako jsou HTML, CSS a dalších nástrojů pro tvoření vysoce interaktivních a dynamických portálů jako jsou skriptovací jazyky PHP a JavaScript, MySQL databáze, JavaScriptové technologie jQuery framework a asynchronní komunikace pomocí Ajaxu, a další knihovny. Dále pak existují různé návrhové vzory (obvykle bývají základem webových frameworků), které udávají již dopředu vymyšlené postupy a způsoby tvorby aplikací. Do této části spadá i studium doporučené odborné literatury.

Druhá část pokrývá studium samotných uživatelských rozhraní. Jde o problematiku, která se zabývá otázkami, jak správně tvořit uživatelská rozhraní. Jak je navrhovat tak, aby byla jednoduchá, intuitivní a uživatelský přívětivá. Existují různé odborné knihy, které o tvorbě uživatelských rozhraní pojednávají a vysvětlují, jakým způsobem prvky (grafického) uživatelského rozhraní realizovat a jakým nikoliv.

2.1 Technologie pro tvorbu webových aplikací

Architektura klient-server

Na principu této architektury v dnešní době funguje většina webových služeb, které nějakým způsobem potřebují uchovávat data. Webová knihova *Chat of future* rovněž bude uchovávat data, bude tudíž potřebovat zprostředkovatele, který s nimi bude operovat, tedy server. Architektura klient-server se zabývá oddělením prezentace dat od jejich zpracování. Je to síťová architektura oddělující klienta (často s GUI) a server, kdy komunikace mezi nimi probíhá přes počítačovou síť. Klient-server popisuje vztah mezi dvěma počítačovými programy, v nichž první z nich, klient, žádá o služby druhý, server. Příkladem může být **webový prohlížeč**, který je klientskou aplikací, přistupující k datům (webovým stránkám) na nějakém serveru. Při užívání této architektury je kladen důraz na snahu o

Obrázek 2.1 - Architektura klient-server [1]

minimalizaci objemu přenášených dat, rozdělení zátěže, snadnější údržbu a centralizaci správy aplikací.

Většina operací s daty probíhá na straně serveru, který velmi často bývá připojen přímo na databázi. Na obrázku 2.1 je znázorněna v dnešní době je nejvíce využívaná tzv. **třívrstvá architektura** [2] vycházející z obecně **vícevrstvé architektury**, na jejímž principu je provozováno mnoho webových aplikací. **Prezentační vrstva** zobrazuje informace pro uživatele, převážně formou GUI. Představuje tedy klientskou část aplikace, protože neobsahuje zpracování dat. **Aplikační vrstva** obsahuje jádro aplikace, její logiku, funkce a zpracování dat. Třetí **datová vrstva** je nejčastěji tvořena **databází,** která uchovává a zpřístupňuje data, zároveň udržuje jejich konzistenci pomocí **integritních omezení.** Datovou vrstvu nemusí nutně tvořit databáze, může zde být také **(síťový) souborový systém, webová služba a jiné.** Datová vrstva bývá obvykle přímo spojená s vrstvou aplikační, dohromady tvoří serverovou část aplikace. Aplikační logika a data mohou být umístěna na jednom fyzickém zdroji (serveru), ale i odděleně. Existuje zde jisté podobenství s návrhovým vzorem MVC.

Moderní webové technologie

Technologie pro tvorbu moderních interaktivních aplikací se již v dnešní době označují pojmem **Web 2.0**. Tento termín charakteristicky vyjadřuje skutečnost, že je stránka vytvořena množstvím moderních webových technologií. Uživatel je vtažen do tvorby obsahu, ve kterém jsou umístěny grafické prvky, obrázky, videa, zásuvné moduly, komponenty a další. Obsah stránek bývá velmi často aktualizován asynchronně, což vysoce zvyšuje interaktivitu a pohodlnost užívání.

Ještě před uvažováním o návrhu sktruktury knihovny spolu s jejím uživatelským rozhraním jsem přemýšlel, jaké moderní webové technologie by mohly být pro realizaci využity. Většina webových služeb a knihoven je implementována pomocí skriptovacích jazyků, které umí pracovat s technikami pro tvorbu internetových stránek. Internetové prezentace se v dnešní době v drtivé většině tvoří pomocí značkovacího jazyka HTML, grafickou podobu stránkám přidávají kaskádové styly. Jistou dynamiku a funkčnost přidává skriptovací jazyk PHP, pro dynamickou práci s elementy jazyka HTML máme k dispozici knihovnu jQuery s jejími nadstavbami. V následujících odstavcích jsou tyto technologie pro tvorbu moderních webových aplikací zmíněny s podrobnějším popisem.

Značkovací jazyky a kaskádové styly

Značkovací jazyky [4] a kaskádové styly spolu tvoří dvojici, která realizuje základní strukturu značek a vlastností, které jsou základem klasické www stránky sdělující uživateli určité informace. HTML (*HyperText Markup Language*) je derivátem metajazyka SGML *(Structured Generic Markup Language*) a jeho značky (tagy) mohou nabývat různých vlastností a ty se nejčastěji upravují CSS styly. Všechny značky tvoří **DOM strukturu** (Document Object Model), což je objektově orientovaná reprezentace, umožňující přístup a modifikaci jednotlivých elementů.

Kaskádové styly CSS jsou atributy, které se nastavují značkám. Představují vizuální podobu webové stránky. CSS atributy se skládají z dvojic *„jméno-hodnota“* a pomocí tříd nebo identifikátorů se přiřazují značkám jazyka HTML. HTML a CSS jsou standardy, spadající pod skupinu W3C [3].

PHP, MySQL

Tato dvojice přinesla první dynamické prvky k tvorbě www stránek. MySQL [5] je nástroj pro tvorbu relačních databází, v jejichž tabulkách jsou uložena data a ve spojení se skriptovacím jazykem PHP

(*Hypertext Preprocessor*) je možné tato data vybírat, editovat, mazat, filtrovat a poté je ve formě webové stránky či XML dat poslat klientovi (webovému prohlížeči).

PHP [6] je skriptovací jazyk, dodávající webovým stránkám jistou dynamiku. Pomocí programových konstrukcí může parametricky ovlivňovat zdrojové texty stránek. Jeho syntaxe vychází z jazyka C, nechybí zde větvení, cykly, práce s různými datovými typy a široká škála vestavěných funkcí.

JavaScript

Technologie, která umožňuje práci se samotnými HTML prvky, uloženými v DOM struktuře. Přinesla první možnosti tvořit interaktivní webové aplikace. JS skripty jsou vykonávány na straně uživatele, není tedy potřeba mít aplikační server. Pomocí JS můžeme téměř libovolně měnit elementy (prvky) webové stránky bez nutnosti stránku znova načítat. [7]

Ajax

Vychází z JavaScriptu, někdy také nazýván asynchronní JavaScript. Umožňuje asynchronně komunikovat se serverem, pomocí JS manipulovat se získanými daty a interaktivně upravovat obsah webové stránky bez nutnosti opětovného načtení. S příchodem této technologie se interaktivita stránek vysoce zvýšila.

jQuery

jQuery je JavaScriptová knihovna, která klade důraz na interakci mezi JS a HTML. Odděluje „chování“ od struktury HTML. Přináší širokou škálu operací s DOM elementy včetně podpory XPath. K elementům lze přistupovat podle jejich ID, tříd, jmen a dalších vlastností díky selektorům, které tvoří základ knihovny. Mezi nejvíce užívané operace a funkce knihovny jQuery patří efekty (skrývání a zobrazování elementů, animace, řetězení), práce s asynchronními HTML požadavky, přidávání atributů k elementům a podobně. Tato knihovna také výrazně zvyšuje interaktivitu webových stránek. [8]

REST

Výše je zmíněna architektura klient-server a její princip komunikace. Na straně serveru je velmi často implementováno aplikační programové rozhraní, ve zkratce API. Toto rozhraní koresponduje s aplikační vrstvou třívrstvé architektury klient-server. REST (anglicky *Representational State Transfer*) je architektura rozhraní, navržená pro distribuované prostředí. Je použitelné pro jednotný a snadný přístup ke zdrojům (*Resources*). Zdroji mohou být data, stejně jako stavy aplikace, pokud je lze popsat konkrétními daty. Všechny zdroje mají vlastní identifikátor URI. REST implementuje čtyři základní metody, které jsou známé pod označením CRUD (*Create*, *Retrieve*, *Update* a *Delete*). Jde vytváření, získávání, změnu a mazání objektů, tedy dat. Tyto čtyři metody jsou implementovány pomocí odpovídajících metod HTTP protokolu. Pomocí REST architektury se velmi často implementují API – aplikační rozhraní na straně serverů. [9]

Návrhové vzory

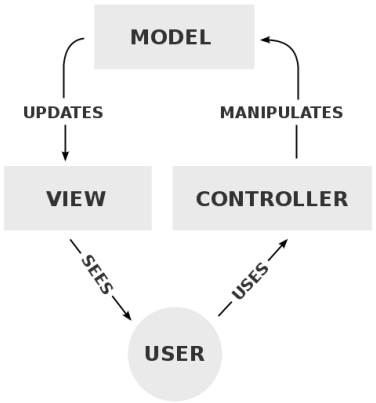
Návrhové vzory (anglicky *design patterns*) představují jisté obecné řešení problému, které se využívá při návrhu počítačových programů, v našem případě při návrhu webových aplikací. Primárně se jedná o jistý popis řešení, šablonu. Jsou zde zmíněny, protože v dnešní době se při realizaci webových knihoven vychází právě z předem definovaných postupů a způsobů řešení. Vždy lze začít „od píky“,

ale využít technik návrhového vzoru výrazným způsobem zrychluje proces tvorby jisté webové aplikace. Ve většině případů jsou návrhové vzory objektově orientované, ukazující vztahy a interakce mezi třídami a objekty. Jelikož jde o *návrhové* vzory, nepoukazují na samotnou implementaci, ale na návrh realizace určitého problému. Každý návrhový vzor má své pojmenování, v následujících podkapitolách jsou uvedeny ty nejčastěji užívané při návrhu a realizaci webových portálů či aplikací.

Model – View – Controller

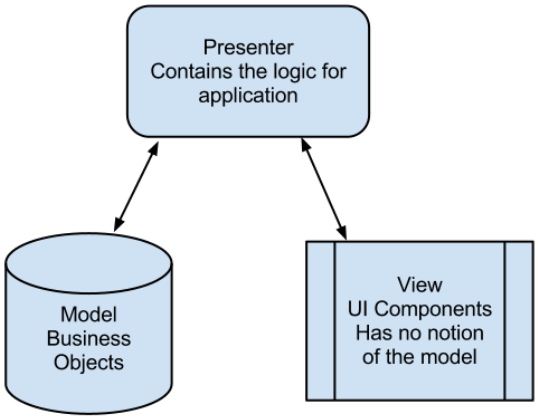
Ve zkratce MVC [10]. Tento návrhový vzor odděluje datový model, uživatelské rozhraní a řídící logiku aplikace do tří nezávislých komponent. Modifikace jedné z nich má pouze minimální dopad na ostatní, tudíž výraznou výhodou tohoto vzoru je možnost oddělené práce realizátorů, kdy se každá skupina věnuje jedné z oblastí. První z komponent je **model**, což je doménově specifická reprezentace informací, s nimiž aplikace pracuje. Ve smyslu webových aplikací tato komponenta zahrnuje práci s databází – výběr a editace dat a podobně. Druhou komponentou, označovanou jako **view**, je pohled, který převádí data získaná z modelu do podobné pro prezentaci. **Controller** je komponenta, která reaguje na typicky uživatelské události a provádí změny v **modelu** nebo v **pohledu (view).**

Model – View – Presenter



Obrázek 2.2 - Typická spolupráce mezi komponenty MVC [15]

MVP [11] je návrhový vzor, který vychází z definice MVC. Nejčastěji je využíván při tvorbě uživatelských rozhraní, velmi často i tedy při tvorbě webových aplikací, které velmi často s tvorbou uživatelských rozhraní úzce souvisí. Logika aplikace je opět rozdělena do tří nezávislých modulů. **Model** je rozhraní, které definuje operaci s daty. **View (pohled)** je pasivní rozhraní, které převádí získaná data z **modelu** do formy, která se prezentuje uživateli. **Presenter** hraje spojovací roli mezi **modelem** a **pohledem**. Získává data z **modelu** (někdy také *repositáře*), a převádí je do formátu, který je použit v modulu **view**. Tento návrhový vzor je základem PHP frameworku Nette, kde **view** představuje šablonovací systém *latte*.



Obrázek 2.3 - Spolupráce prvků MVP [16]

Model – View – ViewModel

Další z návrhových vzorů, MVVM [12], který je odvozen z MVC. Je využíván v softwarovém inženýrství, jako první s ním přišla firma Microsoft. Zaměřuje se na vývoj uživatelských rozhraní, podporuje **událostmi řízené programování** [13] v technologiích WPF a jim podobných, jejichž součástí jsou jazyky XAML a .NET. Po vzoru MVC je struktura tohoto návrhového vzoru opět rozdělena do tří nezávislých komponent. **Model** reprezentuje vrstvu, která přistupuje k datům. **View (pohled)** představuje prezentaci prvků v grafickém uživatelském rozhraní (GUI). **ViewModel** je abstrakce pohledu, která opět podobným způsobem tvoří spojovací mezivrstvu mezi **view** a **modelem**. Spojení je uskutečněno pomocí tzv. **data bindings**, což jsou aspekty, které definují, jak změny v **pohledu** souvisí s operaci s daty v **modelu.**

2.2 Principy uživatelských rozhraní

Webové knihovny a služby mají uživatelská rozhraní. Uživatelé s nimi nějakým předem definovaným způsobem komunikují a operují, uživatelské rozhraní (anglicky *User Interface*) je kolekce způsobů a nástrojů, kterými lidé ovlivňují chování různých strojů či systémů. V oblasti informačních technologií jde zpravidla o počítačové programy a aplikace, webové nástroje, vestavěné systémy a podobně. V kontextu s informatikou se uživatelské rozhraní dělí na dva typy – textové a grafické. Rozhraní pro tyto nástroje a systémy jsou typicky souhrny různých ovládacích prvků – u programů a aplikací to jsou textové vstupy a výstupy, ovládací tlačítka, textová pole a mnohem více, vestavěné systémy mají zpravidla své rozhraní, navržené přesně pro typ daného systému.

Rozhraní příkazového řádku

CLI (anglicky *Command Line Interface*) [14] je základní uživatelské rozhraní, kdy uživatel s programem či aplikací komunikuje prostřednictvím terminálu. Pomocí tohoto typu rozhraní se programy ovládají pomocí předem definovaných příkazů, které mohou mít různé parametry a argumenty. Na základě nich program poskytuje výstup, který může být přesměrován na standardní výstup, který zobrazí terminál, nebo do souboru. Rozhraními příkazového řádku jsou například *Příkazový řádek* v systému Windows a *Terminál* v Linuxových distribucích.

Textové uživatelské rozhraní

TUI (anglicky *Text User Interface*) [15] představuje rozhraní, které je jakýmsi bodem mezi rozhraním příkazové řádky (CLI) a grafickým uživatelským rozhraním (GUI). Typicky pracuje v textovém režimu, kdy je zobrazovací zařízení, obvykle obrazovka, rozděleno na řádky a sloupce, kdy jedna buňka adresovaná číslem řádku a sloupce obsahuje nejvýše jeden znak. Znaky patří do předem daných rodin, známe například ASCII. Pomocí znaků mohou být sestaveny textové objekty, které představují ovládací prvky aplikací a systémů, jde tedy o jistý druh grafiky ve formě textu. Programy s textovým uživatelským rozhraním byly běžné v systému MS-DOS – souborový manažer *Norton Commander*, textový editor *T602* a vývojové prostředí *Turbo Pascal*.

Grafické uživatelské rozhraní

V dnešní době nejvíce rozšířeným uživatelským rozhraním je GUI (anglicky *Graphic User Interface*) [16]. GUI je kolekce široké škály různých grafických prvků, které realizují četné způsoby ovládání programů a systémů. Na monitoru počítače jsou rozmístěna okna, ve kterých programy zobrazují svůj výstup. Pro ovládání programů uživatel využívá klávesnici, myš a grafické vstupní prvky jako jsou tlačítka, textová pole, táhla, zaškrtávací pole, výběry z možností, seznamy, formuláře a podobně. GUI je nejrozšířenějším typem uživatelského rozhraní z důvodů, že jej najdeme jak v počítačích, tak i v přenosných zařízeních, přenosných přehrávačích médií, herních zařízeních, kuchyňských spotřebičích a kancelářských zařízeních. Představuje informace a akce, které jsou uživateli zobrazovány pomocí ikon a vizuálních indikátorů. Akce jsou typicky prováděny na základě přímé interakce s grafickými ovládacími prvky. Různá GUI mívají často úplně odlišnou vizuální podobu, protože pomocí současných moderních technologií pro jejich tvorbu mohou tvůrci navrhovat rozmanité podoby GUI. Grafika, rozložení a částečně i funkčnost se často odvíjí od specifikace zákazníka.

Návrh uživatelských rozhraní

Kvalitní uživatelské rozhraní vždy vychází z kvalitního a dobře specifikovaného návrhu. V době návrhu je možno provádět jakékoliv úpravy, doplňování a aktualizace dle požadavků investora (zákazníka). Během samotné realizace navrhovaného systému již nelze provádět zásadní změny, jinak by bylo nutné začít znova, opět návrhem, což je časově velice náročný proces.

V této podkapitole uvedeme osm základních principů a postupů při návrhu uživatelských rozhraní, nazvaných **Osm zlatých pravidel pro tvorbu uživatelských rozhraní,** která vychází z odborné literatury [17]:

1. Usilujte o konzistenci. Používejte konzistentní technologii, stanovte a dodržujte pravidla pro tvorbu rozhraní daného systému či prostředí.
2. Respektujte širokou skupinu uživatelů. Ujasněte si, jaké cílové skupiny budou výsledný produkt užívat (děti, dospělí, zdravotně postižení, …). Tvořte řešení s ohledem na tyto skupiny.
3. Poskytujte zpětnou vazbu. Uživatel by měl být přiměřeně informován o výsledcích práce s aplikací, zda proběhla korektně či nikoliv, a proč.
4. Navigujte uživatele. Řada uživatelských úloh se skládá z posloupností více akcí. Rozdělte úlohy na jednoduché logické kroky, které respektují pracovní postup.

5. Předcházejte chybám. Uživatelské rozhraní by mělo minimalizovat možné chyby ze strany uživatele. V případě, že dojde k neočekávané chybě, je třeba o jejím vzniku informovat.
6. Umožněte uživateli kdykoliv se vrátit a buďte tolerantní k jeho chybám. Je důležité, aby se uživatel mohl všude tam, kde je to možné, vracet zpět a akci opětovně provést.
7. Vytvářejte předvídatelné uživatelské rozhraní. Uživatel musí být iniciátorem, tedy tím, kdo řídí aplikaci. Aplikace by neměla řídit uživatele.
8. Nepřetěžujte krátkodobou paměť uživatele, nabízejte přehlednost. Uživatel nesmí být nucen si rozhraní pamatovat. Mělo by být navrženo tak, aby vždy logicky usoudil, jak danou akci provést, jako kdyby ji dělal poprvé.

Podobně jako pravidla pro tvorbu, existují také charakteristiky kvalitních a úspěšných uživatelských rozhraní, viz [18]. Je spousta informací o různých konstrukčních technikách, které lze využít při tvorbě UI, webových stránek a řešení společných problémů. Charakteristik kvalitních a efektivních UI je rovněž 8:

1. Čisté.
   Čistota je nejdůležitější prvek uživatelského designu. Vskutku, hlavním účelem uživatelského rozhraní je umožnit lidem jednoduše komunikovat s naším systémem. Pokud uživatel nemůže přijít na to, jak aplikace funguje, bude zmatený a frustrovaný.
2. Stručné.
   Cílem je dodržovat čistotu, ale zároveň také stručnost, což se nemusí vždy dařit. Pokud můžete vysvětlit pojem jednou větou místo vět tří, udělejte to. Ušetřete drahocenný čas uživatelů dodržování stručnosti. Poté je odměna nevyhnutelná.
3. Povědomé.
   Mnoho designerů usiluje o intuitivní uživatelské rozhraní. Co ale intuitivnost ve skutečnosti znamená? Je to něco, co může být přirozeně a instinktivně pochopeno. Jak ale udělat něco intuitivní? Udělat to povědomým. Povědomé je něco, co jste už viděli dříve. Když je vám něco povědomé, víte, jak se to chová a víte co od toho čekat. Zjistěte, které subjekty jsou vašim cílovým uživatelům povědomé, a integrujte je to vašeho návrhu UI.
4. Reaktivní.
   Důležité je také navrhnout a vytvořit rozhraní, které nějakým způsobem reaguje na vstupy od uživatele. Uživatel by měl jednoduše poznat a vědět, že rozhraní právě něco provádí, že nějak interaguje.
5. Konzistentní.
   Konzistentní uživatelská rozhraní „učí“ uživatele jistým vzorům. Naučí se, jak jednotlivé tlačítka, tabulky či ikony vypadají a rozpoznají jejich (podobný) vzhled v odlišných kontextech. Také se naučí, jak jisté věci pracují, a to jim přinese rychlejší sžití a práci s něčím novým, co už ale viděli dříve.
6. Atraktivní.
   Toto je nejdiskutabilnější charakteristika každého uživatelského rozhraní. V kontextu s UI atraktivnost značí zábavu při užívání. Cílem je udělat rozhraní „dobré“ nebo „pěkné“ – vhodně zvolené barvy, rozložení prvků, grafické doplňky a jiné. Diskutabilní je tato

charakteristika z toho důvodu, že to, co se líbí určité skupině uživatelů, se může ostatním příčit. V poslední době se velmi osvědčuje jednoduchost ve formě minimalismu.

7. Efektivní.
   Vše, co skutečně potřebujete pro vytvoření dostatečně efektivního uživatelského rozhraní je uvědomit si, co přesně uživatel potřebuje, čeho přesně se pokouší dosáhnout, a nechat jej to provést bez „povyku a rozruchu“. Musíte si stanovit cíle, jak má vaše aplikace fungovat – které funkce má obsahovat (a které nikoliv). Realizujte rozhraní, které umožňuje lidem jednoduše udělat to, co chtějí místo vyhledávání funkcí například v seznamu.

8. Odpouštějící.
   Nikdo není dokonalý a lidé přirozeně dělají chyby při užívání uživatelských rozhraní. Proto by vždy měla být možnost se vrátit či jiným způsobem chybu odstranit.

2.3 Druhy on-line komunikace

On-line komunikace je takový způsob komunikace, kde na sebe lidé mluví nepřímo. To však neznamená, že spolu nemohou pomocí hlasového ústrojí mluvit vůbec, dorozumívání však neprobíhá formou přímého setkání dvou lidí, kteří jsou vzájemně na doslech v reálném prostoru. Lze to zobecnit tím způsobem, že k předání jisté informace je využito nějakého jiného média, než jsou pouze zvukové vlny vycházející z hlasového ústrojí řečníka a přijímané sluchovým ústrojím posluchače.

On-line komunikaci lze rozdělit do dvou druhů, jsou jimi **přímá** a **nepřímá** [19] komunikace. Do kategorie přímé komunikace patří **chat**, **telefon** a **sms**. Přímá komunikace získala svůj název z toho důvodu, že zde řečník obvykle obdrží odpověď ihned nebo s krátkým časovým zpožděním. K nepřímé se řadí **elektronická pošta (e-mail)**, **sociální sítě**, **diskuze** a **blog**. Zde obvykle není očekávána brzká odpověď, ale mohou vznikat sporné případy, například u elektronické pošty.

Chat

Chat [20] je krátká komunikace či rozhovor dvou nebo více lidí, uskutečněná prostřednictvím komunikační (obvykle internetové) sítě. Je vždy provedena v reálném čase. Klasicky se komunikuje formou psaného textu, moderní technologie však možnosti chatu značně rozšířily – můžeme spatřit doplňky ve formě emotikonů, souborových a hlasových příloh.

Telefon

I telefonní hovory se řadí do on-line komunikace, protože pro realizaci hovorů je zde využita elektricky napájená telefonní síť. Obvykle jsou spojeni pouze dva účastníci, kteří pro svůj hovor obdrží rezervovanou část přenosového pásma, existují však také způsoby telefonních hovorů pro více lidí formou přepojování na telefonních ústřednách.

SMS

SMS (anglicky *Short Message Service*) je služba dostupná na většině digitálních mobilních telefonů. Jsou to krátké textové zprávy, které lze posílat mezi mobilními telefony, jinými zařízeními, na pevné linky nebo prostřednictvím internetové sítě. Délka zprávy je v závislosti na použité znakové sadě omezena, typicky na 160 znaků.

Elektronická pošta (e-mail)

E-mail je elektronická forma klasické pošty, způsob odesílání, doručování a přijímání zpráv přes internetové komunikační systémy. Každý uživatel je identifikován jednoznačnou e-mailovou adresou, která se používá k adresaci. Pro úspěšné užívání elektronické pošty je potřeba mít zařízení s připojením do internetové sítě a e-mailový klient, což je aplikace, která je uživatelským rozhraním, spravujícím uživatelovu e-mailovou schránku. Aplikace pro správu e-mailové schránky může být realizována formou programu, který lze nainstalovat, nebo webovým portálem. Nespornými výhodami oproti klasické poště je okamžité, nebo téměř okamžité doručení zprávy odesílateli, možnost jednoduché archivace zpráv a většinou také adresář kontaktů. Nevýhodou je nemožnost běhu e-mailového serveru v případě výpadku dodávky elektrické energie a také potenciální náchylnost na útoky hackerů.

Sociální sítě

Sociální nebo také společenská síť [21] je internetová služba, která registrovanému uživateli umožňuje vytvoření osobní (či firemní) veřejný profil na internetu. Pomocí něj může komunikovat s ostatními uživateli formou zpráv, sdílení informací, fotografií, videí, souborů a více. Pojmenování vychází ze sociologického pojmu – sociální síť je skupina lidí, která spolu udržuje komunikaci různými prostředky. Z počátku vznikly sociální sítě za účelem výměny informací na různých vzdělávacích a vědeckých institutech, později byly rozšířeny mezi širokou veřejnost, svůj internetový profil tedy může mít opravdu každý. Za nejrozšířenější sociální sítě lze v dnešní době považovat Facebook, Twitter, LinkedIn a GooglePlus. [22]

Diskuze

Diskuze je způsob on-line komunikace, kdy se uživatelé formou komentářů vyjadřují k různým elektronicky publikovaným periodikům, jako jsou články, příspěvky a recenze. Na komentáře je možno reagovat a každý z nich je možno citovat. Obvykle bývají k dispozici také prostředky pro zhodnocení kvality komentářů.

Blog

Blog je osobní či firemní profil realizovaný samostatnou webovou stránkou. Tuto stránku obvykle spravuje jeden uživatel. Na bloku uživatel publikuje informace o své osobě (firmě), články, příspěvky, obrázky; tedy téměř cokoliv, co chce sdělit čtenářům, a co je v souladu s pravidly pro publikování informací v internetové síti.

2.4 Realizace druhů on-line komunikace

V této podkapitole jsou uvedena řešení forem on-line komunikace, jsou popsány jejich výhody a nevýhody.

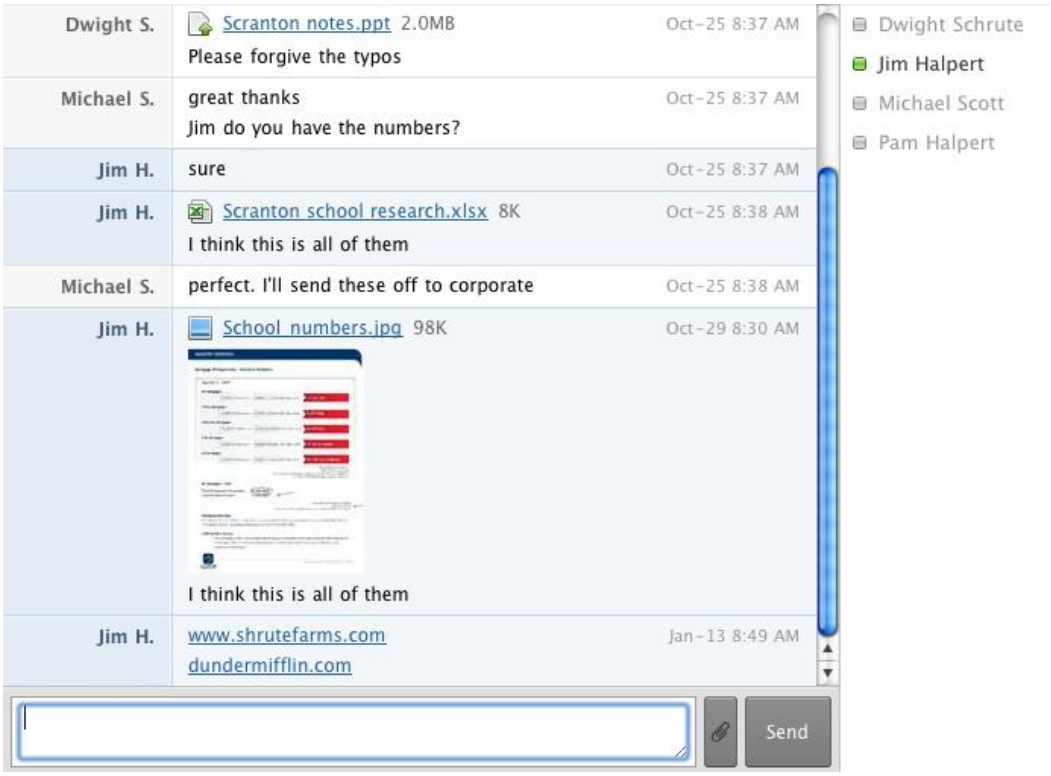
Chat

Jak je uvedeno v kapitole 2.3, chat je krátká forma on-line komunikace, rozhovor dvou či více lidí a obvykle bývá uskutečněn v reálném čase. Pokud chce uživatel odesílat krátké zprávy ostatním, expresivně označeno „chatovat“, musí mít na serveru, který chatovací službu poskytuje, vytvořen

účet. K autorizaci uživatele se obvykle využívá kombinace přihlašovacího jména (někdy e-mailové adresy) a hesla.

Nyní již přejdeme ke skutečně existujícím způsobům realizace chatu. První verze tohoto způsobu on-line komunikace byly vizuálně znázorňovány pomocí vhodně umístěné tabulky, ve které byly zobrazovány zprávy, seřazené podle času. Nejnovější ze zpráv zaujímala pozici buď nahoře, nebo dole v tabulce. Každá z nových zpráv začínala novým řádkem a byla označena časovým razítkem. Takových tabulek mohlo na jednom serveru obecně existovat obrovské množství, tudíž se každé z tabulek přiřadil název a byla označena jako místnost. Do nich se vstupovalo pomocí hypertextových odkazů. Každá místnost obsahovala seznam uživatelů, kteří aktuálně v dané místnosti chatovali. Ten byl fyzicky umístěn na okraji layoutu místnosti, mohl být ale také skryt pod nějakým ovládacím prvkem, například rozbalovacím seznamem. Servery obsahující tyto místnosti se označovaly jako chatovací fóra, jejich primárním cílem bylo umožnit přihlášeným uživatelům komunikovat mezi sebou právě prostřednictvím chatu. Začaly se rojit internetové seznamky.

Později začaly používat chat i webové portály, které nebyly primárně zaměřeny pouze na on-line komunikaci. Jsou to různé firemní stránky a sociální sítě. Chat je zde modulem zakomponovaným do webové aplikace, funkce chatu je oddělená od funkce stránky. Seznam přihlášených uživatelů se obvykle fyzicky vyskytuje na okrajích layoutu stránky, každá komunikace mezi uživateli je speciálně umístěna, obvykle v nezávislém okně (bloku). Moderní chatovací okna podporují rychlé odesílání souborů či obrázků, toto bývá využito i v chatovacích modulech sociálních sítí.



Obrázek 2.4 - Ukázka aplikace realizující chat s moderními prvky [23]

Princip chatu nevyužívají pouze webové aplikace, existují také programy, které využívají připojení k internetové síti. Jsou jimi různé IM (*Instant Messengers*). Mezi nejznámější patří *ICQ* a *QIP*.

Výhodami on-line chatu jsou možnost rychlého odeslání zpráv, často vizuální zvýraznění aktuálně přítomných uživatelů, jednoduchost, svižnost, potenciální jistota, že si přítomný uživatel doručenou zprávu přečte brzy.

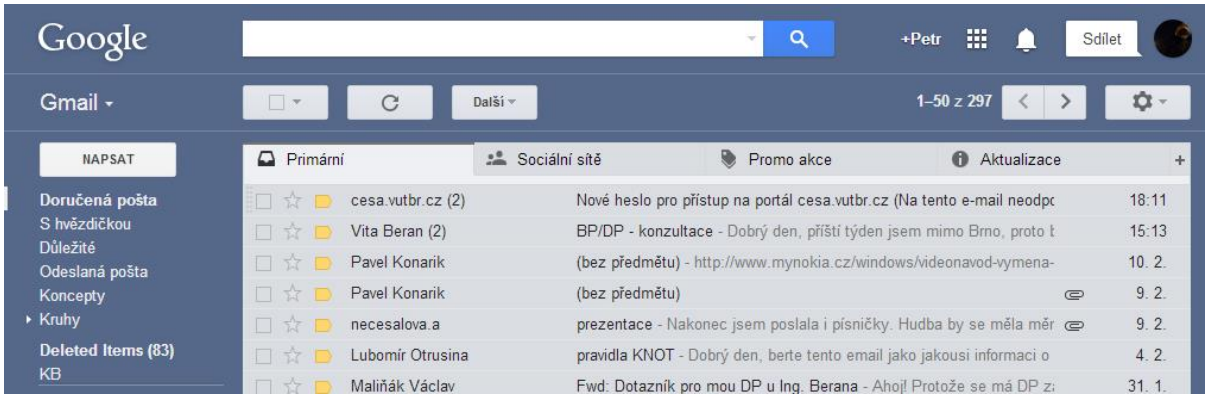
Nevýhodou je nepřehlednost příspěvků, která se ale týká spíše chatu jakožto internetové seznamky. Pokud je v místnosti přítomno velké množství lidí, nové zprávy mohou rychle přibývat a uživatelé je nemusí stihnout číst. Tato nevýhoda byla později částečně eliminována soukromými chatovacími okny v rámci místností.

E-mail

Dnes lze elektronickou poštu odesílat několika způsoby. Jeden z nich jsou programy (aplikace) přímo zaměřené na práci s e-mailovou schránkou, nazvané e-mail klienti (Thunderbird, MS Outlook, Postbox). Ty fungují tím způsobem, že se v nastavení vybere e-mailový server a schránka, se kterou chce uživatel pracovat, proběhne autorizace pomocí e-mailové adresy a hesla, a poté se začne program využívat (nejen) pro odesílání a příjem zpráv.

Dle mého názoru lepším ze způsobů správy e-mailové schránky jsou webové aplikace, vytvořené jako on-line formy e-mail klientů. Tyto webové aplikace jsou velmi často poskytovány portály, na jejichž serverových stanicích běží e-mailová služba. Velmi často se také formuláře, které slouží pro přístup do schránek, nacházejí na internetových stránkách v rámci daných portálů (email.cz, gmail.com).

On-line e-mail klienti jsou rozděleni na bloky. V layoutu klienta obvykle nalezneme složky pro filtrování zpráv, samotné zprávy, zobrazené dle vybrané složky, ovládací prvky pro nastavení schránky, odesílání pošty a další operace se zprávami. Nově příchozí a nepřečtené zprávy jsou vizuálně odlišeny, často tučným písmem. V okně se zprávami jsou zobrazeny ty zprávy, které spadají do složky, kterou máme aktuálně označenou, nejčastěji a automaticky označenou složkou po přihlášení do schránky bývá složka *Doručená pošta*. Zde jsou klasicky časově řazeny všechny doručené zprávy kromě těch, které si nepřejeme dostávat a k jejich automatickému odstraňování či jiným operacím využíváme *filtry*. Ukládá se také odeslaná pošta, která obvykle má svou složku.



Obrázek 2.5 - Ukázka rozložení prvků e-mailového klienta [24]

Většina on-line e-mailových klientů obsahuje adresář, kde si uživatel může spravovat svůj seznam kontaktů. Další funkce je odesílání obrazových či souborových příloh. Velikost příloh bývá omezena, stejně jako je omezena velikost schránky, při jejím zaplnění je žádoucí promazat nedůležité zprávy.

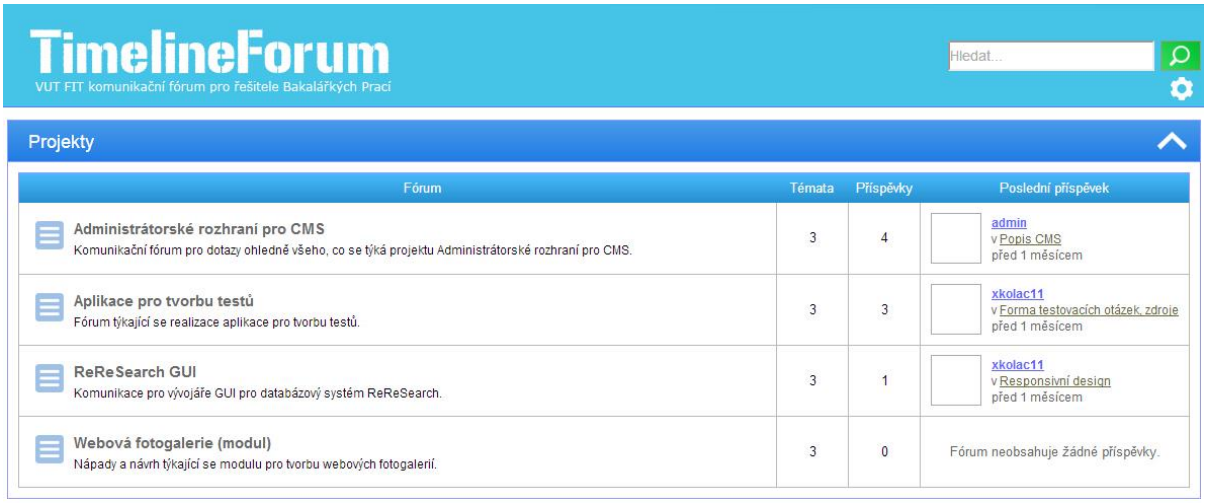
Použití webových e-mailových klientů má řadu výhod. Největším pozitivem je fakt, že není třeba nic instalovat ani konfigurovat, důležité je pouze si pamatovat přihlašovací údaje. Data jsou uložená na serveru a nezabírají místo na disku v počítači. Nevýhodou je potenciální nebezpeční úniku informací, data jsou ve větším bezpečí na lokálním pevném disku.

Fórum

Internetová komunikační fóra vznikla jako celky, které v sobě sdružují různé diskuze, příspěvky a komentáře. Jedno fórum je obvykle zaměřeno na jedno konkrétní (rozsáhlé) téma, které dává podnět k rozsáhlé diskuzi. V dnešní době fóra poskytují různým firmám veřejnou formu zpětné vazby od zákazníků.

Registrovaní uživatelé na fóru pro svou autorizaci využívají přihlašovací jméno (někdy e-mail adresu) a heslo. Každý uživatel má svůj profil, který může libovolně upravovat. Často na něm bývají znázorněny statistiky aktivity uživatele – počet příspěvků, nejvíce komentované téma a podobně. Bývají využívány také různé hodnosti, jejichž názvy se odvíjí od tématu fóra. Každý uživatel má aktuálně jednu z hodností, která odráží jeho aktivitu na fóru. Po dosažení určitého počtu příspěvků nebo doby aktivity se hodnost zvýší. Vyšší hodnosti mohou mít vyšší práva a více možností interakce s fórem a příspěvky. Existují také speciální hodnosti jako *administrátor*, který je hlavním správce fóra v rámci něj může „úplně vše“. *Moderátoři* nemají taková práva jako *administrátor*, mohou ovšem také omezovat práva nižších hodností, jejich primárním cílem je spravovat jim přidělené diskuze a hlídat, zda uživatelé dodržují pravidla. *Globální moderátor* je člen nadřazený *moderátorům* a přiřazuje jim různé diskuze pro správu. Uživatelé mohou být z různých (ale předem známých) důvodů omezování správci, každému z omezení se říká *ban*. Fóra mají stanovená svá vnitřní pravidla, se kterými uživateli při registraci souhlasí. Pravidla musí být veřejně publikovaná.

Každé fórum je obvykle hierarchicky rozděleno na sub-fóra, sub-fóra se dělí na témata a v každém z témat jsou již samotné příspěvky. Pro přehlednost se mohou sub-fóra řadit do kategorií. Nejčastěji



Obrázek 2.6 - Klasické rozložení prvků komunikačního fóra (můj grafický návrh)

je využit tabulkový layout, kde každý řádek tabulky obsahuje sub-fórum, téma nebo příspěvek - záleží na tom, v jaké hloubce hierarchie fóra jsme zanořeni. V jednotlivých buňkách řádků jsou umístěny názvy sub-fór/témat/příspěvků a statistiky podřazených elementů (počet témat/příspěvků, poslední příspěvek). Sub-fórum nebo téma je odemčené nebo uzamčené. Do uzamčeného tématu již

nelze přidávat příspěvky. Nové příspěvky v rámci témat a sub-fór jsou pro přihlášeného uživatele vizuálně označeny. Po přečtení se příspěvek již nepovažuje za nový.

Při přidávání nového příspěvku využívají moderní fóra různé doplňky. Základem je odlehčený webový textový editor, příspěvky tedy mohou mít rozmanitou vizuální podobu. Příspěvky obsahují nejen text, ale i obrazové a souborové přílohy. Fóra zaměřená na témata z oblasti IT technologií mívají v příspěvcích podporu speciálního vizuálního zobrazení zdrojových kódů.

Výhodami diskuzních fór jsou:

- Jednoduché vytváření témat a příspěvků.
- Přehlednost, jednoduchá navigace.
- Funkční a příspěvkově bohatá fóra jsou vynikajícím nástrojem pro SEO. Čím více tematických příspěvků fórum obsahuje, tím kvalitnější pro vyhledávače bude internetová stránka obsahující tuto formu online komunikace.
- Uživatel fóra, které je zaměřeno na různé produkty, má tendenci „falešně“ se cítit jako anonymní přispěvatel, je pro něj snazší vytvořit příspěvek než zavolat či napsat e-mail. Tímto faktorem se také zvyšuje uživatelnost diskuzních fór.

Nevýhody diskuzních fór:

- Čím rozsáhlejší fórum, tím větší náročnost na správu.
- Větší nároky na diskový prostor.

3 Návrh řešení

V této kapitole je rozebrán cíl práce, motivace, specifikace a návrh řešení. Jsou zde popsány případy užití aplikace a prezentovány návrhy uživatelského rozhraní aplikace. Návrh nastiňuje, jak bude aplikace vypadat a jak se s ní bude pracovat.

3.1 Motivace a rozbor cíle

Hlavním cílem práce je vytvořit komunikační nástroj, pomocí kterého budou moci lidé sdílet své názory, nápady, myšlenky či cokoliv jiného. Sdílení je hlavním směrem aplikace. Uživatelé tohoto komunikačního nástroje budou vytvářet veřejné konverzace. V daných konverzacích budou sdílet své nápady s ostatními uživateli, které si zvolí. Do konverzací však mohou nahlížet i jiní uživatelé. Každý z řešitelů může obecně řešit jiný problém, pracovat na něčem jiném, cíl jeho práce může být odlišný. Komunikační nástroj se nebude zaměřovat pouze na jedno předem stanovené téma, bude souborem různých veřejných konverzací. Cílem a výsledkem bude produkt, nazvaný *Chat of future*, který bude široce využitelný mezi řešiteli projektů či obecně různých problematik.

Konverzace budou pro všechny přihlášené uživatele veřejné. Je důležité zpracovat problematiky, které se týkají vytváření konverzací, přidávání zpráv, přihlašování uživatelů, vybírání adresátů zpráv. Proto potřeba vytvořit vhodný a přesný návrh datového modelu. Pro jednoduchost, efektivnost a snadnou znovu-použitelnost bude aplikace již od jádra vytvářena jako knihovna. Tato knihovna se poté přesně definovaným postupem integruje do internetové stránky. Důležitou částí návrhu je také přesné definování akcí, které bude uživatel provádět. Je potřeba uvážit, která data bude uživatel zadávat a co bude jakým způsobem vizualizováno.

Jak je výše uvedeno, výsledkem práce bude moderní nástroj, který umožní uživatelům jednoduše a rychle vytvářet komunikace na libovolné téma. Při zpracovávání vize a cíle jsem narazil na množství problematik, které se dají rozdělit do dvou větších celků:

1. Otázky, týkající se funkcí a vlastností uživatelského rozhraní.
    - Jakým způsobem využít hlavní přednosti a vlastnosti forem on-line komunikací, uvedených v kapitole 2.4?
    - Jakým způsobem označovat adresáty zpráv?
    - Jak odeslat zprávu jednomu / více lidem naráz?
    - Jak vytvářet konverzace?
    - Jakým způsobem označovat konverzace?
    - Uzavírat staré konverzace?
    - Jak upozorňovat na nové zprávy v konverzacích,?
    - Jak přidávat nové zprávy do již existujících konverzací?
    - Která data bude potřeba uchovávat?

- Jakým způsobem zadávat potřebná data?

2. Otázky, týkající se nástrojů pro realizaci a výsledku.
    - Jak utvářet aplikaci, aby cílový program měl vlastnosti spíše knihovny, než klasické aplikace?
    - Je důležitější vytvořit čisté a jednoduché uživatelské rozhraní či rozhraní s komplexním ovládáním?
    - V jaké formě a kde budou uložena data potřebná pro běh komunikačního nástroje?
    - Jakým způsobem se bude s daty operovat, jak budou získávána?
    - Jaké technologie a postupy bude vhodné využít při realizaci nástroje?
    - V jakém prostředí bude nástroj využíván?
    - Jak náročný (výkonově, paměťově, síťově) bude nástroj?

3.2 Případy užití komunikačního nástroje

V této podkapitole jsou popsány již konkretizované případy užití rozhraní komunikačního nástroje. Jsou zde uvedeny konkrétní akce, které může uživatel provádět a konkrétní způsoby vytváření a prohlížení konverzací.

První případ užití komunikačního nástroje je jednoduché odeslání (sdílení) zprávy jednomu konkrétnímu adresátovi. V tomto případě uživatel vybere daného adresáta ze seznamu uživatelů a budou mu zobrazeny konverzace, které s ním souvisí. Uživatel může buďto vytvořit novou konverzace a do ní umístit novou zprávu, nebo napsat novou zprávu do již existující konverzace. Druhou možnost může zvolit pouze v případě, že se daná konverzace uživatele týká. Všechny konverzace jsou veřejné, ale zprávy do nich mohou přidávat pouze ti uživatelé, se kterými souvisí.

Druhým případem užití komunikačního nástroje je možnost vytvořit hromadnou konverzaci, tedy sdílet zprávu s více lidmi najednou. Taková konverzace se vytvoří tím způsobem, že uživatel zobrazí formulář pro vytvoření nové konverzace. Poté začne vybírat adresáty ze seznamu uživatelů, přičemž změny při vybírání se budou okamžitě reflektovat v jednom z testových polí formuláře. Po výběru adresátů zadá uživatel obsah zprávy do dalšího textového pole a zprávu odešle.

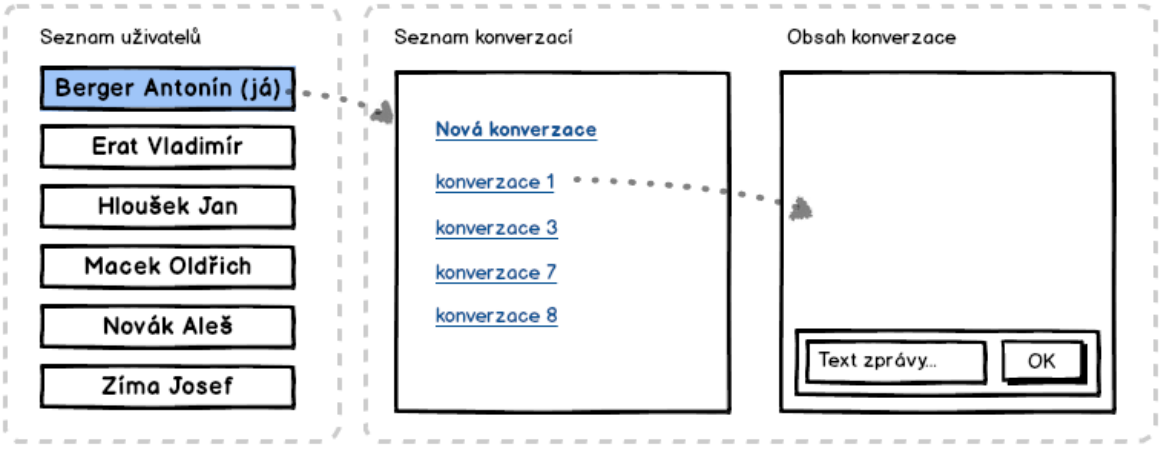
Další z případů užití nástroje nastane, když si uživatel bude chtít prohlédnout, případně přidat zprávu do již existující konverzace. Ve většině případů si uživatel chce prohlédnout konverzaci, které souvisí právě s ním samotným. Proto vybere ze seznamu uživatelů sám sebe a budou mu zobrazeny konverzace, které se ho týkají. Poté vybere jednu z nich a získá zprávy obsažené v dané konverzaci. V tomto okamžiku může do dané konverzace přidávat nové zprávy pomocí jednoduchého formuláře, obsahujícího pouze textové pole s tlačítkem.

3.3 Návrh rozhraní

Prvním odrazovým můstkem návrhu aplikace je návrh komunikačního rozhraní. Zde je potřeba si ujasnit, z jakých funkčních částí (bloků) se bude uživatelské rozhraní skládat.

Primárně budeme mít k dispozici seznam uživatelů. Ten musí být nějakým způsobem zadán. Předpokládá se, že každý z uživatelů by měl mít možnost zahájit komunikaci pouze v případě, že je přihlášen do systému. Pro uživatelskou komunikaci bude k dispozici navenek jednoduchý, uvnitř komplexní nástroj, který kombinuje přednosti a důležité faktory forem online komunikací – chatu, elektronické pošty a fóra. Z chatu bude využita forma rychlé komunikace, možnost psaní rychlých a krátkých zpráv označených časovým razítkem. Bude možno vytvářet hromadné konverzace, což je technika vytažená z emailové komunikace. To, že konverzace budou ukládány, označovány, a budou veřejně přístupné všem přihlášeným uživatelům, koresponduje s vlastnostmi on-line komunikačního fóra.

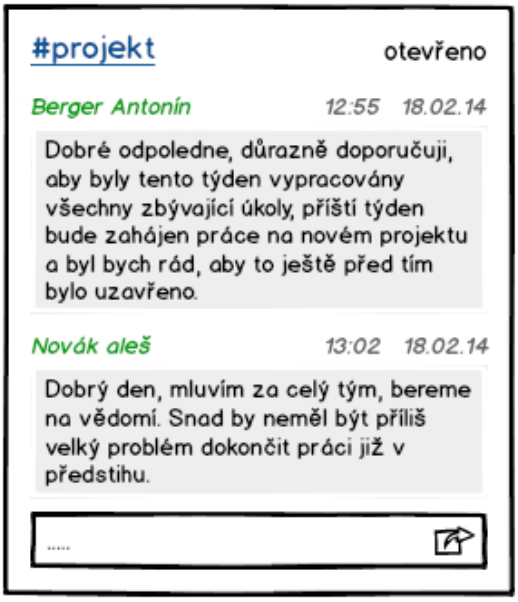
Seznam uživatelů je prvním ze dvou hlavních celků, bloků, ze kterých se bude uživatelské rozhraní skládat. Druhý oddělený celek budou tvořit jednotlivé konverzace a zprávy. Tyto dva bloky budou vizuálně odděleny. Ve druhém bloku budou zobrazovány konverzace, které se týkají uživatele, jenž byl označen (vybrán) ze seznamu uživatelů. Po zvolení jedné z konverzací budou zobrazeny zprávy, které pod ni spadají. Zprávy budou seřazeny podle času, od nejstarší po nejnovější, přičemž je vhodné zobrazit uživateli zprávy nové.



Obrázek 3.1 – Rozdělení částí uživatelského rozhraní

Rozhraní bude podporovat dva způsoby vytváření konverzací. Ty budou rozlišeny podle toho, jestli chceme sdílet své myšlenky s jedním nebo s více lidmi. Pokud chceme napsat zprávu jednomu člověku, zpravidla ji chceme napsat rychle a jednoduchým způsobem. Pro tento způsob bude k dispozici jednoduchý formulář (input-box a tlačítko), do kterého se zadá text zprávy. Druhý způsob sdílení zpráv uživatel využije v případě, že sdílí myšlenky s více lidmi, jinak řečeno vytváří hromadnou konverzaci. Pro tento způsob vytvoření konverzace bude k dispozici druhý formulář, který bude obsahovat textové pole, kde budou vypsáni označení uživatelé (adresáti), a textové pole pro obsah zprávy.

Kromě vytváření jednotlivých konverzací spadá do funkcionality komunikačního rozhraní také prohlížení již existujících konverzací, ve kterých jsou sdíleny informace, a přidávání informací nových. Pro správnou funkčnost téhle funkcionality je zapotřebí jednotlivé konverzace nějakým způsobem označovat. Každá z konverzací bude mít svůj jednoznačný, v rámci aplikace jedinečný identifikátor. Po výběru konverzace budou zobrazeny její detaily. Detaily zahrnují účastníky konverzace, a jednotlivé zprávy konverzace, seřazené dle času. Každá ze zpráv proto musí být označena časovým razítkem. Staré konverzace lze označovat jako uzavřené, přičemž zprávy lze přidávat pouze do otevřených konverzací. Přidávání zpráv bude realizováno pomocí formuláře umístěného v okně konverzace.



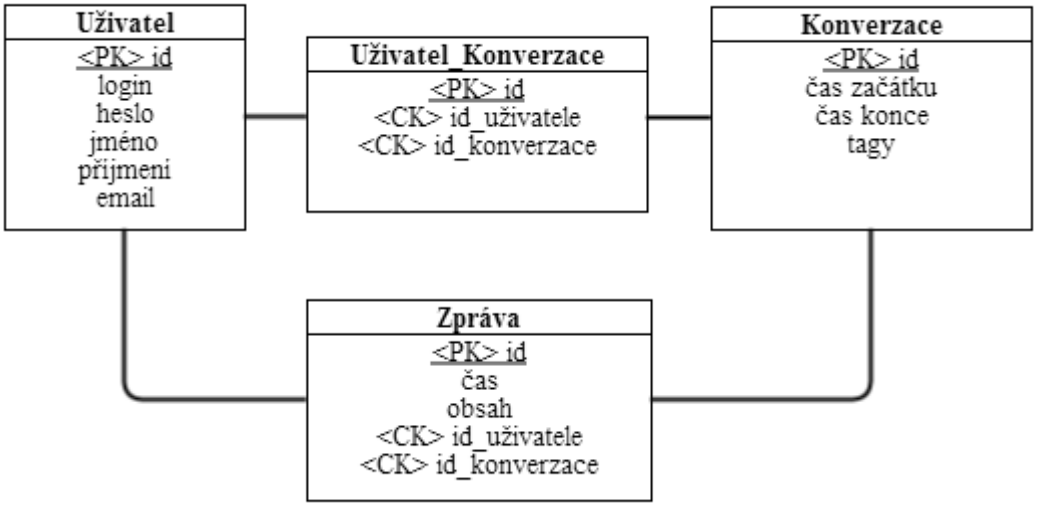
Obrázek 3.2 - Okno označené konverzace

Na návrh funkcionality navazuje grafický návrh komunikačního rozhraní. Jak má nástroj fungovat je jedna věc, jak bude vizuálně reprezentován uživateli, je věc druhá. Důraz je kladen na jednoduchost, intuitivnost a uživatelskou přívětivost. Obecným předpokladem je vytvořit vysoce interaktivní GUI. To znamená vytvořit jej tak, aby nebylo nutno znovu načítat stránku při použití libovolného ovládacího prvku formuláře, při zobrazování a skrývání částí (bloků) komunikačního rozhraní, při přidávání či odebírání adresátů konverzací a podobně. Všechny prvky grafického uživatelského rozhraní si nesmí navzájem překážet, musí být sladěny a vykresleny v pořadí, v jakém je má uživatel využít. Z GUI by měl uživatel jednoduchým způsobem vydedukovat, jak se jednotlivé ovládací prvky používají, k čemu slouží a jakých nežádoucích operací se vyvarovat.

3.4 Návrh datového modelu

V dnešní době téměř každá webová aplikace funguje na základě určitého přesně specifikovaného datového modelu. A nejinak je tomu i v případě tohoto komunikačního nástroje. Aplikace bude použitelná na různých platformách a prohlížečích, obecně na počítačích, které od sebe mohou být fyzicky libovolně vzdáleny. Z toho důvodu potřebujeme databázi, do které se budou ukládat data o jednotlivých uživatelích, konverzacích a zprávách.

Výše je uvedeno, že primárně bude mít přihlášený uživatel k dispozici seznam uživatelů. Z toho vyplývá, že je potřeba ukládat informace (data) o uživatelích. Vzniká první entitní množina, nazvaná *Uživatel*. Aby každý uživatel při přihlášení do systému mohl ověřit svou totožnost, potřebuje znát své přihlašovací jméno (login) a heslo. Další údaje, jako jsou jméno, příjmení a e-mailová adresa mohou sloužit pro zpřehlednění komunikačního nástroje a snadnou implementaci rozšiřujících funkcí.



Obrázek 3.3 – Diagram vztahů a entit

Dále je potřeba uchovávat data o jednotlivých zprávách. Každá ze zpráv je jedinečná, je vytvořena (odeslána) v určitý čas a má určitý obsah. Žádná ze zpráv se nevytvoří automaticky sama, vždy ji vytváří nějaký uživatel, vzniká nám tedy první vztahová množina mezi entitními množinami *Zpráva* a *Uživatel*. Jeden uživatel může za čas svého působení v komunikačním nástroji vytvořit žádnou až libovolný počet zpráv.

Každá ze zpráv patří pod jeden určitý pojmenovaný celek, pod konverzaci. Z této informace vyplývá, že potřebujeme vytvořit další entitní množinu, nazvanou *Konverzace*, která představuje část modelu, jenž definuje, jak se budou uchovávat data o konverzacích. Každá z konverzací je jedinečná, „zná“ čas svého začátku a může „znát“ čas svého konce. Případ, kdy není znám čas konce konverzace, považuje se konverzace za otevřenou a mohou do ní být přidávány zprávy, jinak se považuje za uzavřenou. Každá z konverzací může být označena libovolným počtem *tagů*, tudíž potřebujeme další atribut pro jejich ukládání. Vztahová množina mezi entitními množinami *Zpráva* a *Konverzace* reprezentuje, že každá ze zpráv spadá pod určitou konverzaci. Předpokládá se, že po vytvoření konverzace uživatel odešle alespoň jednu zprávu, jinak by vytvoření konverzace bylo nesmyslné.

Dále jsem vyvodil důsledek, že každý z uživatelů může vytvořit libovolný počet konverzací, a zároveň v každé z konverzací může být libovolný počet (ale alespoň jeden) účastníků. Vzniká vztah typu *M:N* mezi entitními množinami *Uživatel* a *Konverzace*. Jelikož se vtah tohoto typu modeluje přidáním vazební entitní množiny, je vytvořena množina *Uživatel_Konverzace*, jejímiž atributy jsou identifikátor *id* a dále cizí klíče, získané z primárních klíčů tímto vztahem spojených entitních množin, *id uživatele* a *id konverzace*.

Tento návrh datového modelu je základem pro správnou funkčnost komunikačního nástroje. Při samotné realizaci velmi často vyvstávají určité problémy a jejich řešení občas vyžadují zásah do návrhu datového modelu ve formě úprav či přidávání nových atributů. Více v kapitole 4.

4 Realizace

Tato kapitola se zabývá realizací aplikace. Je zde uvedeno, do jakých částí se komunikačních nástroj při realizaci rozdělil, jakým způsobem byly jednotlivé čísti realizovány, jak mezi sebou interagují a které technologie byly při realizaci použity.

4.1 Struktura aplikace

V dnešní době je struktura většiny webových aplikací a nástrojů rozdělena na dvě hlavní části. Princip fungování těchto dvou částí se velmi blíží principu klient-server modelu. Jedna část, klient, se nachází na počítači uživatele, který aplikaci či nástroj využívá. Druhá část je server, který poskytuje data či různé služby. Server obvykle své služby nenabízí aktivně sám, běží v pasivním režimu a čeká na požadavky od klientů. K realizaci struktury komunikačního nástroje jsem přistoupil způsobem, který odráží princip klient-server modelu.

Nejprve jsem začal s implementací databáze dle datového modelu, navrženého v kapitole 3.4. Za účelem pozdějšího zveřejnění aplikace pro testování jsem databázi vytvořil na serveru www.endora.cz, který mi poskytl testovací adresu www.chatoffuture.tode.cz. Z řetězce adresy je patrné, že je databáze umístěna na doméně třetího řádu, která je neplacená. Pro ostré nasazení aplikace bude v budoucnu vhodné umístit databázi na registrovanou doménu druhého řádu. Pro vytvoření databáze jsem využil jazyk MySQL. Název databáze je „chatoffuture“, databáze obsahuje 4 tabulky, jejichž atributy vycházejí z návrhu datového modelu.

Dále bylo potřeba vytvořit nástroj, pomocí kterého se dá jednoduchým způsobem přistupovat k datům v databázi. Rozhodl jsem se využít techniku nazvanou API (*Application Programming Interface)*, což je zkratka pro *Aplikační programové rozhraní*. Toto rozhraní je stejně jako databáze umístěno na serveru. Existuje spousta způsobů realizace API, zde bylo vhodné a jednoduché využít REST API. REST je architektura, která zpracovává požadavky protokolu HTTP směřované od klientské části aplikace a na základě nich operuje s daty v databázi – poskytuje je, vytváří, maže nebo upravuje. Protože server poskytl podporu skriptovacího jazyka PHP, implementoval jsem REST API pomocí něj.

Při realizaci klientské části aplikace jsem vycházel z předpokladu, že uživatel komunikačního nástroje (knihovny) *Chat of future* bude chtít nástroj do svých webových stránek nasadit rychle a jednoduchým způsobem. Jako jednoduchý způsob nasazení knihovny jsem zvolil zadání uživatelů, tedy účastníků konverzací, pomoci jazyka HTML. Tohle je prvním z pouhých dvou úkonů, který bude muset uživatel udělat pro úspěšné nasazení a rozběhnutí komunikačního nástroje. Druhým úkonem je vložení JavaScriptového souboru *chatoffuture.js*, který je samotnou knihovnou. Dle struktury zadaného HTML kódu vytvoří komunikační prostředí a poskytuje funkce, které komunikují se službami na straně serveru, tedy s API. Tento jediný soubor se stará téměř o vše. Slovo „téměř“ je uvedeno z toho důvodu, že soubor ve svých funkcích využívá ještě funkce JavaScriptové knihovny jQuery, je tedy potřeba do hlavičky stránky, do které má být komunikační nástroj integrován, vložit ještě kód, který připojí knihovnu jQuery. Je sice možné připojit jeden JavaScriptový soubor do jiného,

nicméně jsem zjistil, že tento úkon by negativně ovlivnil správnou funkčnost knihovny z důvodu asynchronní interpretace JavaScriptového kódu.

Během implementace klientské části aplikace jsem zjistil, že by bylo možné aplikaci nerozdělovat do dvou částí, klientské a serverové. Vše by mohlo běžet pouze na jednom počítači, na kterém je umístěna webová stránka, do které má být knihovna integrována. V tomto případě by si ale musel uživatel vždy vytvořit svou databázi, a proto by také musel znát přesnou strukturu databáze. Musel by k ní poskytnout přihlašovací údaje všem ostatním uživatelům knihovny. Dále by musel každý z uživatelů, byť jen nepatrně, zasáhnout do JavaScriptového kódu knihovny. Těmito úkony by se jednoduchost nasazení komunikačního nástroje odstranila. Rozhodl jsem se, že všechna nasazení knihovny budou využívat jediné aplikační programové rozhraní (API) a jedinou databázi. Bylo tedy potřeba provést první rozšíření struktury databáze, více v následující podkapitole 4.2.

4.2 Detaily implementace

V této kapitole budou popsáni bližší detaily implementace jak serverové, tak klientské části komunikačního nástroje. Také budou vysvětlena implementační úskalí, na které jsem při realizaci narazil a jejich způsob řešení.

Server - REST API

Princip funkčnosti tohoto typu aplikačního programového rozhraní spočívá v tom, že přijímá požadavky protokolu HTTP, ty zpracuje a na základě nich odešle odpověď ve formě dat. Klient vytvoří požadavek, a odešle jej na adresu serveru, na kterém je API umístěno. Kód API je rozdělen do dvou souborů. Soubor *Rest.php* obsahuje třídu *REST*, která je základní třídou aplikačního programového rozhraní, obsahuje metody pro nastavení hlaviček požadavku, zpracování požadavku – rozpoznání typu požadavku (GET, POST, OPTION, PUT, DELETE), získání řetězce odpovědi požadavku a získání URL adresy požadavku. Soubor *api.php* obsahuje třídu *API*, která rozšiřuje třídu REST o další metody. Konstruktor této třídy vytvoří instanci rodičovské třídy *REST*, pomocí metody *dbConnect()* je provedeno připojení k databázi dle zadaných parametrů v třídních proměnných. Dále třída *API* obsahuje metodu *processApi()*, která na základě URL adresy požadavku „volá“ jednotlivé metody, které již pracují s databází formou SQL dotazů a vracejí data pomocí metody *response()*.

Způsob realizace aplikačního programového rozhraní byl vysvětlen, jak ale dojde ke spuštění či „zavolání“ PHP skriptu s API, když vytvoříme požadavek a odešleme jej? Uveďme si příklad URL adresy HTTP požadavku typu GET:

```
GET chatoffuture.tode.cz/API/login?user=username&password=pwd
```

Tento požadavek potřebujeme přesměrovat tak, aby byl vstupním parametrem třídy *API*. K tomu výborně posloužil soubor *.htaccess*. Tento soubor umožňuje upravit chování adresáře na serveru, v němž je umístěn. Kód obsažený v souboru *.htaccess* při každém přístupu do adresáře */API* automaticky extrahuje část řetězce požadavku za *„chatoffuture.tode.cz/API/“* a spustí skript *api.php*, s parametrem, ve kterém je uložena právě ta extrahovaná část řetězce požadavku. Po spuštění skriptu *api.php* je vytvořena instance třídy *API*, je provedeno připojení k databázi a je invokována metoda *processApi()*, která spustí příslušnou metodu pro operaci s daty v databázi. Uvedený řetězec

požadavku znázorňuje, že bude invokována metoda *login()*. V asociativním poli *_request* boudou uloženy hodnoty *„username“* a *„pwd“* na indexech *„user“* a *„password“*. V metodě *login()* je kód, který pomocí SQL dotazu SELECT získá potřebná data z databáze a pomocí metody *response()* odešle odpověď klientovi, ze kterého bylo API voláno. Podobným způsobem jsou implementovány všechny ostatní metody pro práci s databází. Název metody v řetězci požadavku vždy odpovídá názvu metody v třídě *API*. Požadavek typu GET se používá pro získání dat, požadavek typu POST pro vytváření, mazání či změnu dat. Pomocí požadavku typu POST jsou data odeslána v těle požadavku, u požadavku typu GET je to v URL adrese požadavku. Jelikož jsou data u všech typů požadavků HTTP protokolu odesílána v otevřené podobě, je nutno zvlášť u odesílání citlivých dat, jako je například heslo, data šifrovat. K zašifrování hesla jsem použil šifrovací algoritmus *md5*, jež jazyk PHP nabízí.

Klient – JavaScript + jQuery

Klientská část aplikace, samotná knihovna, je implementována v JavaScriptovém souboru *chatoffuture.js*. Při počátku její realizace jsem vycházel z předpokladu, že je potřeba zpracovat data, zadaná uživatelem ve formě HTML kódu, tedy seznam uživatelů. Každý JavaScriptový kód je spuštěn vždy při každém načtení stránky, do kterém je vložen. Bylo tedy potřeba zařídit, aby zpracování seznamu uživatelů a inicializace chatu proběhla pouze jednou, při prvním načtení stránky. Ke kontrole, zda již počáteční inicializace proběhla, jsem využil cookie soubor. Každý cookie soubor má své jméno a svou hodnotu a je uložen v paměti internetového prohlížeče. Implementoval jsem funkci, která vytvoří pojmenovaný cookie soubor, dále funkci, která přečte hodnotu (řetězec) v něm uloženou a funkci, která cookie soubor smaže. Při načtení stránky, do které je knihovna integrována, se zjistí existence cookie souboru s názvem *READY*. Pokud existuje, inicializace již proběhla. Pokud neexistuje, musí se provést následující. I v případě, že cookie soubor *READY* neexistuje, již mohla inicializace knihovny proběhnout na jiném počítači nebo prohlížeči. Inicializace knihovny je proces, který zahrnuje zpracování uživatelem zadaného HTML kódu a vytvoření uživatelů, účastníků komunikačního nástroje – naplnění databáze příslušnými daty, které je realizováno pomocí asynchronního požadavku typu POST. Zde je patrné využití asynchronního JavaScriptu, Ajaxu. Pokud tedy cookie soubor *READY* neexistuje, zjistí se, jestli data již v databázi existují, je zkontrolována existence e-mailové adresy účastníka. Pro jednoduchost řešení jsem se rozhodl, že každá z e-mailových adres může být použita pouze v jedné aplikaci knihovny.

Po inicializaci knihovny je zobrazen přihlašovací formulář. Po jeho vyplnění a odeslání je zkontrolována korektnost zadaných dat a v případě úspěšného přihlášení je vytvořeny cookie soubory s názvem *LOGGED, UID a USER*. Pokud tyto soubory existují, znamená, to že je uživatel přihlášen. V souboru *LOGGED* je uloženo jméno a příjmení přihlášeného uživatele, v *UID* je jeho jednoznačný identifikátor, který odpovídá identifikátoru získaného z databáze a v souboru *USER* je uložen řetězec e-mailové adresy uživatele. Hodnoty uložené v těchto souborech jsou nadále využívány pro různé kontroly a vykreslování.

Layout inicializované knihovny je rozdělen na dva bloky. První z bloků je seznam uživatelů, druhý je okno konverzací a zpráv. Toto okno konverzací a zpráv je implementováno jako dialogové okno, je možné jej libovolně přemisťovat po stránce. Pro vytvoření této funkcionality jsem využil nástroj *draggable* z knihovny *jQuery UI*. Chtěl jsem také využít možnost vertikální úpravy velikosti dialogového okna s využitím nástroje *resizable*, rovněž z knihovny *jQuery UI*. Rozhodl jsem se tuto

možnost nevyužít, protože pro správnou funkčnost by bylo potřeba vložit ještě jeden soubor s kaskádovými styly, což by byla další práce navíc pro uživatele, a tím pádem snížení jednoduchosti integrace knihovny do internetové stránky.

V seznamu uživatelů má každá položka ovládací prvek, pomocí něhož jsou zobrazeny konverzace daného uživatele. Tyto konverzace jsou zobrazeny ve druhém bloku rovněž jako seznam, kde každá položka (konverzace) má své ID, uživatele, kterých se konverzace týká a ovládací prvek, pomocí kterého jsou zobrazovány zprávy spadající pod danou konverzaci. CSS vlastnosti všech elementů, které knihovna vykresluje, jsou nastavovány pomocí JavaScriptového kódu s využitím jQuery selektorů. Pomocí těchto selektorů, jsme schopni operovat s jakýmkoliv elementem a uložit si referenci na něj do proměnné. Kód je rozdělen do funkcí, kde se různé funkce starají o vykreslování různých částí knihovny, jsou tři hlavní vykreslovací funkce. Funkce pro vykreslení seznamu uživatelů, seznamu konverzací a posloupnosti zpráv. O grafický vzhled knihovny se starají kaskádové styly CSS, které nastavují elementům jisté vlastnosti. Je použita i grafika ve formě obrázků, obrázky jsou rovněž uloženy na serveru s API, ve složce *SRC*.

Řešení problémových částí realizace

Během rozšiřování funkcionality prototypu knihovny jsem narazil na první vážnější problém. Ten spočíval v tom, že pokud byl chat na jedné webové stránce integrován a inicializován, tak při vytvoření nového chatu na jiné stránce, s jinými uživateli, se kromě nově zadaných uživatelů do seznamu vykreslili i uživatelé z prvního použití knihovny. Bylo potřeba rozlišit, kteří uživatelé patří pod kterou realizaci chatu. Tento problém jsem vyřešil přidáním atributu *session_id* do tabulky *uzivatel*. Při vytvoření nového chatu a přidání nových uživatelů databáze se nově přidaným uživatelům přiřadí *session_id* o jedničku větší než *session_id* u dříve vložených uživatelů. Při přihlášení uživatele do komunikace se zjistí, pod jaké *session_id* uživatel patří a do bloku se seznamem uživatelů se vykreslí pouze příslušné kontakty se stejným *session_id*.

Dále bylo potřeba vyřešit automatické přidávání nových zpráv do otevřených konverzací. K realizaci této funkcionality jsem použil JavaScriptovou funkci *setInterval()*, která umožňuje opakované volání zvolené funkce v zadaném časovém intervalu. Při zobrazení zpráv určité konverzace jsem použil příkaz, který nastaví interval pro volání funkce *downloadMessages()*, která automaticky přidává nové zprávy. Zprávy jsou získávány opět pomocí Ajax požadavku. Bylo nutné použít globální proměnnou, ve které vždy byl uložen identifikátor posledně přidané zprávy, aby byly skutečně přidávány pouze zprávy nové. Při zavření okna s konverzacemi či skrytí okna se zprávami konverzace se interval pomocí funkce *clearInterval()* odstraní.

Nejsložitějším implementačním úskalím bylo vymyšlení způsobu, jak upozorňovat na nově obdržené zprávy v konverzacích přihlášeného uživatele. Vyšel jsem ze základní informace, která říká, že každá nově přidaná zpráva má svoje časové razítko, tedy datum a čas. Informaci o nové nepřečtené zprávě jsem se rozhodl vizualizovat v seznamu konverzací. Každý z uživatelů se může účastnit více konverzací a každá z konverzací může mít více účastníků. Podle této logiky jsem vytvořil část datového modelu. Informace o tom, který uživatel se účastní které konverzace, jsou uloženy v tabulce *uzivatel_konverzace*. Tato tabulka původně obsahovala atributy *id*, *id_uzivatele* a *id_konverzace*. Atribut *id* je jednoznačným identifikátorem n-tice relace (tabulky), je tedy identifikátorem jednoho členství v konverzaci. Atributy *id_uzivatele* a *id_konverzace* určují, který uživatel je členem které

konverzace. Do tabulky jsem přidal atribut *datum_precteni*, ve kterém bude uloženo časové razítko určující datum a čas přečtení dané konverzace daným uživatelem. Toto časové razítko bude porovnáváno s razítkem poslední zprávy dané konverzace. Pokud bude razítko poslední zprávy větší, značí to, že konverzace obsahuje nepřečtenou zprávu či více zpráv.

4.3 Integrace knihovny do internetové stránky

Cílem bylo vymyslet co nejjednodušší způsob použití knihovny, čili její integrace do internetové stránky. Vyšel jsem z předpokladu, že zdrojem internetové stránky je jeden soubor. Uživatel, který chce knihovnu použít, musí tedy tento uživatelem vybraný soubor upravit, a to předem definovaným způsobem, kdy bude mít k dispozici návod.

První částí integrace je vložení souboru (skriptu), ve kterém je naimplementována samotná knihovna spolu se soubory obsahující implementaci JavaScriptové knihovny jQuery, kterou knihovna *Chat of future* využívá. Skripty se vloží do hlavičky HTML stránky v následujícím pořadí:

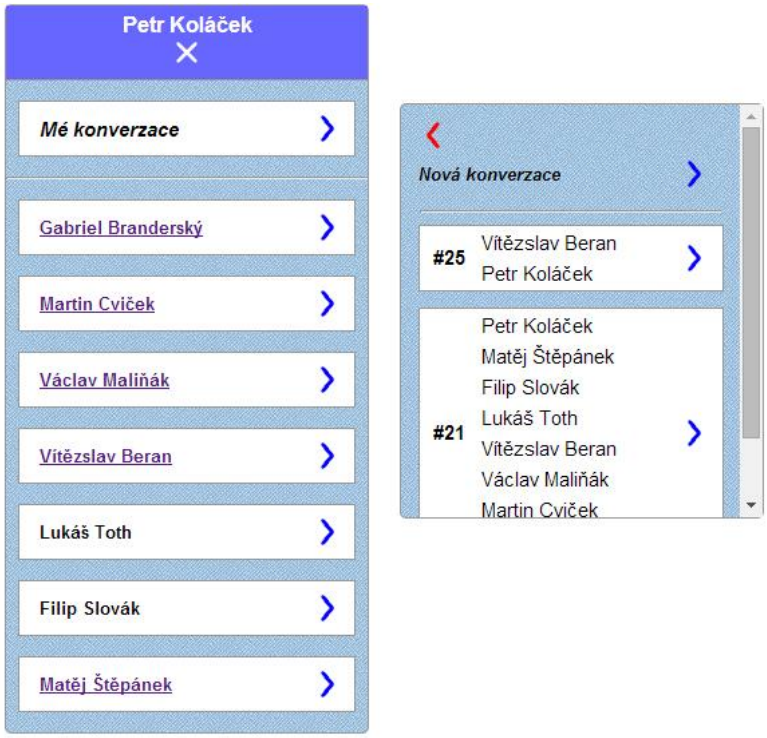
```
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
<script type="text/JavaScript" src="chatoffuture.js"></script>
```

Knihovna *jQuery* musí být vložena jako první, protože funkce v knihovně *Chat of future* využívají její funkce. *jQuery-UI* je rozšířením knihovny *jQuery*.

Druhým krokem po vložení skriptů je definice seznamu uživatelů. Ta se provede pomocí HTML kódu, který se umístí do těla stránky. Zdrojový kód seznamu uživatelů má následující tvar:

```
<div id="chat">
<ul id="chatList">
<li data-email="EMAIL" data-name="JMENO" data-surname="PRIJMENI"
data-target="CIL"></li>
</ul>
</div>
```

Tato struktura seznamu uživatelů je závazná, je důležité dodržet správné názvy tříd u elementů *div* a *ul*, které představují obecný blok a nečíslovaný seznam. Z tohoto důvodu je potřeba zkontrolovat, zda ostatní elementy ve zdrojovém kódu dané HTML stránky nemají stejnou hodnotu u atributu *id*, tedy stejný název třídy. Seznam uživatelů se skládá z položek, kdy jedna položka představuje jednoho uživatele. Každá z položek má následující atributy. Atribut *data-email* je povinný, jeho hodnotou je e-mailová adresa uživatele. Atributy *data-name* a *data-surname* jsou rovněž povinné, obsahují jméno a příjmení uživatele, které se po inicializaci knihovny zobrazí v seznamu uživatelů. Atribut *data-target* je nepovinný, ukládá se do něj odkaz na libovolnou webovou stránku, na kterou se přihlášený uživatel může dostat po kliknutí na jméno daného uživatele v seznamu.



Obrázek 4.1 - Ukázka inicializované knihovny po přihlášení uživatele a zobrazení konverzací

Po dokončení definice seznamu uživatelů a uložení zdrojového souboru stránky stačí pouze stránku navštívit, či obnovit, po jejím opětovném načtení proběhne automatická inicializace knihovny a nástroj pro komunikaci uživatelů bude úspěšně vytvořen. V případě že se inicializace nezdaří, bude uživatel informován informačním výpisem v konzoli prohlížeče.

4.4 Vytváření testů

Při vytváření podkladů pro testování komunikačního nástroje jsem postupoval následovně. Jako první bylo potřeba vytvořit testovací protokol. Obsah testovacího protokolu musí být sepsán v bodech a velmi srozumitelně pro uživatele. Testovací protokol je rozdělen na dvě části. V první části je sepsáno a vysvětleno prvotní nastavení, které uživatelé provedou v první den použití knihovny, tedy její integrace do internetové stránky. Uživatelům je zde v přesně daných krocích vysvětleno, jak integraci provést, a jakým způsobem začít knihovnu využívat. Jsou zde také navedeni na testovací dotazník pro každodenní vyplnění. Druhá část obsahuje dotazník, který se týká testovacího protokolu. Dotazník se skládá z otázek, které směřují na jasnost a srozumitelnost testovacího protokolu, dále pak na složitost a průběh integrace knihovny. Odpovědi na každou z otázek jsou rozděleny dle Likertovy škály, která je jednou z nejpoužívanějších a nejspolehlivějších technik měření postojů v dotaznících. Škála má rozmezí pěti odpovědí od „Rozhodně ano“ k „Rozhodně ne“, přičemž středová hodnota je odpověď „Nevím“. Otázky obsažené v dotazníku testovacího protokolu jsou následující:

1. Je obsah testovacího protokolu napsán srozumitelně? - povinná otázka
2. Je potřeba obsah testovacího protokolu upřesnit? – povinná otázka

3. Shledáváte problematickým vkládání dalších skriptů kromě zdrojového skriptu knihovny? – povinná otázka
4. Je integrace knihovny do Vašich stránek problematická? – nepovinná otázka
5. Ovlivnila integrace knihovny vzhled Vaší stránky? – nepovinná otázka

Poslední dvě otázky jsou nepovinné z důvodu, že ne všichni uživatele po přečtení obsahu testovacího protokolu provádějí integraci knihovny. Tuto činnost obvykle provede ten uživatel, který komunikační službu vytváří, ostatní uživatele se již pouze přihlašují.

Testovací dotazník pro každodenní testování se skládá z otázek, jejichž odpovědi rovněž odpovídají Likertově škále. Otázky v tomto dotazníku směřují na pocity uživatele po denním užívání knihovny. Zjišťují, zda se uživateli s nástrojem pracovalo dobře, zda práce v daný den přinesla nové poznatky, zda jej povzbudila, či zda by nástroj doporučil ostatním uživatelům. Tyto otázky jsou každý den stejné. Výsledky získané z každodenního testování jsou zpracovány v kapitole 4.5.

Finální etapa testování spočívá ve vytvoření závěrečného dotazníku, který bude uživatelem vyplněn pouze jednou. Odpovědi na otázky jsou opět rozděleny dle Likertovy škály. Otázky míří na různé části implementace knihovny, jejich účelem je zjistit, zda se během testování a odstraňování chyb zlepšily či zhoršily operace s problematickými či jinak důležitými částmi. Všechny dotazníky byly vytvořeny pomocí služby *Google Forms*, jejíž výhodou je, že automaticky zaznamenává odpovědi na otázky do tabulky.

Po vyhotovení testovacího protokolu a dotazníků započalo samotné testování. Vytvořil jsem testovací prostředí formou registrace domény *futurechat.tode.cz*. Na tuto doménu jsem umístil webovou stránku, do které jsem integroval knihovnu. Poté jsem definoval seznam uživatelů, jako účastníky komunikace jsem zvolil studenty, kolegy, kteří rovněž řeší problematiky z oblasti webových služeb. Do komunikace jsem rovněž zapojil vedoucího své bakalářské práce a další dva nezávislé účastníky, s nimiž se osobně znám. Představil jsem jim testovací protokol, poskytl adresu testovací domény a přihlašovací údaje do integrované knihovny. Většina z uživatelů denně zjišťovala, jaký pro ně má knihovna přínos a na základě nově nabytých pocitů téměř každý den vyplňovala testovací dotazník. Po 14 dnech testování jsem účastníkům komunikace zveřejnil závěrečný testovací dotazník, který svými otázkami míří na různé části systému. Na základě odpovědí na všechny testovací dotazníky jsem vyvodil výsledky, uvedené v kapitole 4.5.

4.5 Výsledky testování

První částí výsledků testování knihovny je zpracování dotazníku, obsaženém ve druhé části testovacího protokolu. Tento dotazník zjišťuje jasnost a srozumitelnost testovacího protokolu a integrace knihovny. Odpovědi na otázky dotazníku jsem získal od tří uživatelů. Dalšími dvěma položkami dotazníku je označení uživatele (id, jméno, přezdívka) a prostor pro jeho poznámky. Ten zůstal u každé z odpovědí nevyužit.

Cílem mnou vytvořeného dotazníku v druhé části testovacího protokolu bylo zjistit, zda je obsah testovacího protokolu srozumitelný a dostatečný k tomu, aby uživatel či tester věděl, co a jakým způsobem bude testováno. Dále jsem se zaměřil na zjištění výše problémovosti při integraci knihovny

do internetových stránek. Protože lze míru vlivu na vzhled stránek zjistit pouze při integraci knihovny do stránky, připadla testovací otázka na toto téma rovněž do dotazníku k testovacímu protokolu. Dle odpovědí shrnutých v tabulce 4.1 jsem vyvodil závěry, že obsah testovacího protokolu je dostatečný a je napsán srozumitelně. Dalšího cíle, jakožto jednoduchosti integrace knihovny, bylo dle uživatelské zpětné vazby také dosaženo, uživatelé se nesetkávali se žádnými většími problémy při integraci, a pokud vznikly nějaké menší, nebylo na ně poukázáno. Původní očekávání, že vkládání skriptů do hlavičky kódu stránky nebude přítěží, bylo potvrzeno. Použití knihovny částečně ovlivnilo vzhled stránky. Tato skutečnost se týká převážně seznamu uživatelů a je vhodné ji do budoucna eliminovat.

	Uživatel č. 1	*Uživatel č. 2*	*Uživatel č. 3*
Datum a čas	23.4.2014 0:37	28.4.2014 9:12	7.5.2014 9:18
Je obsah testovacího protokolu napsán srozumitelně?	Spíše ano (2)	Spíše ano (2)	Rozhodně ano (1)
Je potřeba obsah testovacího protokolu upřesnit?	Spíše ne (4)	Spíše ne (4)	Spíše ne (4)
Je integrace knihovny do Vašich stránek problematická?	Spíše ne (4)	Nevím (3)	Spíše ne (4)
Shledáváte problematickým vkládání dalších skriptů kromě zdrojového skriptu knihovny?	Spíše ne (4)	Spíše ne (4)	Spíše ne (4)
Ovlivnilo vložení zdrojového kódu knihovny vzhled vaší stránky?	Nevím (3)	Spíše ano (2)	Spíše ano (2)

Tabulka 4.1 – Zpracované odpovědi na otázky z dotazníku k testovacímu protokolu

Do každodenního testování se opět zapojilo pouze několik z uživatelů, počet odpovědí na otázky tedy nedosáhl očekávaného počtu, přesto jich však je dostatek na to, aby se jejich zpracování a vyhodnocení mohlo považovat za důvěryhodné. Počet odpovědí u tohoto dotazníku je mnohem více než u dotazníku k testovacímu protokolu, proto tabulka 4.2 znázorňuje číselný průměr ze všech odpovědí k daným otázkám. K bodům Likertovy stupnice jsou přiřazeny tyto číselné hodnoty. Rozhodně ano = 1, spíše ano = 2, nevím = 3, spíše ne = 4, rozhodně ne = 5.

Pomocí každodenního testovacího dotazníku jsem chtěl zjistit, jak se průběžně mění postoj uživatelů ke knihovně, jejich pocity při práci s ní. V průběhu testování jsem zdokonaloval funkcionalitu knihovny a odstraňoval existující chyby, proto jsem chtěl od uživatelů získat zpětnou vazbu, zda lepší funkcionalitu zaregistrovali. Rovněž mě zajímalo, zda je práce s knihovnou jednoduchá a svižná, dále pak jestli by se uživatelé se svými zkušenostmi s prací s knihovnou chtěli podělit s ostatními potenciálními uživateli. Tento jejich názor je pro mě důležitý, poněvadž aplikace, kterou jeden uživatel nerad doporučí druhému, nepovažuji za dostatečně kvalitně zpracovanou.

	Odpověď
Přinesla Vám dnešní práce s knihovnou nové poznatky?	3
Zlepšila či zjednodušila se Vaše práce s knihovnou?	2,75
Povzbudila Vás dnešní práce s knihovnou?	2,625
Těšíte se na další práci s knihovnou, doporučil(a) byste ji ostatním uživatelům?	2,625
Pracovalo se Vám dnes s knihovnou dobře?	2,125

Tabulka 4.2 - Zpracované odpovědi na otázky z denního testovacího dotazníku

Z odpovědí je patrné, že uživatelé si všimli některých z oprav či zlepšení funkcionality knihovny. Číselný výsledek 3 byl očekávaný, protože byl větší počet oprav, které se vizuálně neprojevily. Opravy a zlepšení se pozitivně projevily na jednoduchosti práce s knihovnou. Více než polovina uživatelů by knihovnu doporučila ostatním potenciálním uživatelům, což není špatný výsledek, nicméně je vhodné na tomto faktoru zapracovat.

	Odpověď
Bavilo vás s knihovnou pracovat, zkoušet její možnosti?	2,33
Jsou v knihovně její důležité prvky viditelné?	1,66
Pracovalo se Vám s knihovnou svižně a jednoduše?	2,66
Doporučil(a) byste knihovnu ostatním uživatelům?	2,66
Měla by podle Vás být knihovna robustnější, měla by obsahovat více funkcí?	3,33
Je pro Vás způsob vytváření konverzací jednoduchý a vhodný?	2
Je pro Vás způsob odesílání zpráv jednoduchý a vhodný?	1,66
Poznal(a) jste dle chování knihovny, že máte v některé z konverzací novou zprávu?	1,66

Tabulka 4.3 - Zpracované odpovědi na otázky z finálního testovacího dotazníku

Závěrečný testovací dotazník se skládá z více odpovědí než dotazník pro každodenní testování, jeho otázky míří na různé problematické či jinak důležité části knihovny. Představuje souhrn a závěr testování, z odpovědí se vyvozují konečné pocity uživatelů a jejich názory na kvalitu vyhotovení knihovny. Uživatelé na tento dotazník odpověděli pouze jednou. Tabulka 4.3 opět znázorňuje zprůměrované odpovědi na otázky finálního dotazníku. Při tvoření otázek závěrečného dotazníku jsem se zaměřil na dvě hypotézy. Prvním cílem bylo otestovat přívětivost uživatelského rozhraní knihovny. Na tu se zaměřují první čtyři otázky dotazníku. Druhým cílem bylo otestovat efektivnost

práce s knihovnou, její robustnost a dostatečnou funkcionalitu. Na tyto faktory je zaměřena druhá čtveřice otázek.

Hypotéza jednoduchosti implementované knihovny se potvrdila, i když jisté rezervy jsou podle číselných výsledků první čtveřice otázek patrné. Na svižnosti a jednoduchosti realizace knihovny je možné nadále v budoucnu zapracovat, převážně zjednodušit asynchronní požadavky směřující na aplikační programové rozhraní a snížit jejich počet. Velmi dobrých výsledků dosáhla efektivnost a robustnost knihovny. Uživatelé nutně nepotřebují větší funkcionalitu, dosavadní dle jejich názorů postačuje. Větší funkcionalita by se mohla také negativně odrazit v jednoduchosti řešení. V efektivnosti se pozitivní odráží způsob vytváření konverzací a odesílání zpráv spolu s vizuálním upozorňováním na nově obdržené zprávy v konverzacích.

5 Závěr

Hlavním cílem této bakalářské práce bylo vytvoření internetové služby pro efektivní komunikaci skupiny uživatelů. Dalším z cílů bylo realizovat službu jako webovou knihovnu, kterou si uživatel jednoduchým způsobem integruje, tedy vloží do své internetové stránky.

Při návrhu služby bylo předpokladem, že uživatel musí nějakým způsobem pouze definovat seznam uživatelů, kteří tuto službu budou využívat. Bylo tedy potřeba vytvořit program, který tento seznam zpracuje a vytvoří komunikační nástroj, který budou seznamem definovaní uživatele využívat. Cílem byla jednoduchost použití knihovny, její kód je tedy obsažen v jediném JavaScriptovém souboru. Za tímto účelem byla struktura aplikace rozdělena do dvou hlavních celků, klientské a serverové části. Klientskou část představuje samotná knihovna realizovaná ve skriptu, serverovou část tvoří aplikační programové rozhraní, které pracuje s databází. Uživatelé knihovny budou tvořit konverzace a zprávy, a ty je potřeba na nějaké místo ukládat, proto je využití databáze více než vhodné.

Testování knihovny odhalilo její klady i nedostatky. Klady aplikace spočívají v jednoduchosti použití, tedy integrace, v pohodlnosti využívání, vytváření zpráv a konverzací a v přehlednosti jejich vizuálních prvků. Nedostatky knihovny na sebe poukázaly ve svižnosti uživatelského rozhraní, do budoucna je vhodné zjednodušit funkce, které pracují s aplikačním programovým rozhraním a snížit jejich počet. Pro ještě větší zvýšení efektivity práce s knihovnou lze do budoucna uvažovat o spojení nynější struktury aplikace, která je nyní rozdělena na dva celky, do celku jednoho, ve kterém budou všechny funkce pohromadě. Komunikační knihovna by byla distribuovaná. V tomto případě by byla cenou vyšší efektivity větší náročnost na diskový prostor počítače a síťový přenos dat.

Přínos mé práce na knihovně shledávám v nově nabytých zkušenostech v oblastí tvorby webových knihoven. Dále pak v získání přehledu o výhodách, nevýhodách a důležitých prvcích různých forem on-line komunikace. Na konec řadím získání nových a cenných programátorských zkušeností s technologiemi pro tvorbu moderních webových nástrojů.

Literatura a zdroje

[1] Architektura klient-server. *Wikipedie: Otevřená Encyklopedie* [online]. 2014 [cit. 2014-01-21]. Dostupné z: http://cs.wikipedia.org/wiki/Klient-server

[2] Three-Tier Architecture. *Techopedia* [online]. 2008 [cit. 2014-01-21]. Dostupné z: http://www.techopedia.com/definition/24649/three-tier-architecture

[3] W3C. *W3C Consortium* [online]. 2012 [cit. 2014-01-20]. Dostupné z: http://www.w3.org/Consortium/

[4] HTML Tutorial. *W3C Consortium* [online]. 2012 [cit. 2014-01-20]. Dostupné z: http://www.w3schools.com/html/

[5] MySQL. In: *Oracle Datasheet* [online]. 2013 [cit. 2014-01-19]. Dostupné z: http://www.mysql.com/products/enterprise/mysql-datasheet.en.pdf

[6] PHP Manual. *The PHP Group* [online]. 2013 [cit. 2014-01-19]. Dostupné z: http://www.php.net/manual/en/

[7] JavaScript. *Wikipedie: Otevřená Encyklopedie* [online]. 2013 [cit. 2014-01-18]. Dostupné z: http://cs.wikipedia.org/wiki/JavaScript

[8] How jQuery Works. *JQuery: Write less, do more* [online]. 2014 [cit. 2014-01-17]. Dostupné z: http://learn.jquery.com/about-jquery/how-jquery-works/

[9] REST: Architektura pro webové API. *Zdroják.cz* [online]. 2009 [cit. 2014-02-26]. Dostupné z: http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[10] Model - View - Controller. *Wikipedia: The Free Encyclopedia* [online]. 2014 [cit. 2014-01-13]. Dostupné z: http://en.wikipedia.org/wiki/Model-view-controller

[11] Model - View - Presenter. *Wikipedia: The Free Encyclopedia* [online]. 2014 [cit. 2014-01-13]. Dostupné z: http://en.wikipedia.org/wiki/Model-view-presenter

[12] Model - View - ViewModel. *Wikipedia: The Free Encyclopedia* [online]. 2014 [cit. 2014-01-13]. Dostupné z: http://en.wikipedia.org/wiki/Model_View_ViewModel

[13] HALLER, Phillip a Martin ODERSKY. Event-Based programming Without Inversion of Control. [online]. 2006. Dostupné z: http://lampwww.epfl.ch/~odersky/papers/jmlc06.pdf

[14] Command Line Interface. *Wikipedia: The Free Encyclopedia* [online]. 2014 [cit. 2014-01-19]. Dostupné z: http://en.wikipedia.org/wiki/Command-line_interface

[15] Text-based User Interface. *Wikipedia: The Free Encyclopedia* [online]. 2013 [cit. 2014-01-19]. Dostupné z: http://en.wikipedia.org/wiki/Text-based_user_interface

[16] Graphical User Interface. *Wikipedia: The Free Encyclopedia* [online]. 2014 [cit. 2014-01-19]. Dostupné z: http://en.wikipedia.org/wiki/Graphical_user_interface

[17] SHNEIDERMAN, Ben a Catherine PLAISANT. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Boston: Pearson Addison Wesley, 2004. 4. ISBN 0321197860.

[18] STONE, D., C. JARETT, M. WOODROFFE a S. MINOCHA. *User Interface Design And Evaluation*. Burlington, Massachusetts: Morgan Kaufmann, 2005. ISBN 0120884364.

[19] Druhy komunikace. *Metodický portál: inspirace a zkušenosti učitelů* [online]. 2011 [cit. 2014-02-05].
Dostupné z:
http://wiki.rvp.cz/Knihovna/1.Pedagogický_lexikon/K/Komunikace/Druhy_komunikace

[20] Chat. *Wikipedie: Otevřená encyklopedie* [online]. 2014 [cit. 2014-02-05]. Dostupné z: http://cs.wikipedia.org/wiki/Chat

[21] Social network. *Wikipedia: The Free Encyclopedia* [online]. 2014 [cit. 2014-02-05]. Dostupné z: http://en.wikipedia.org/wiki/Social_network

[22] Top 15 Most Popular Social Networking Sites. In: *EBiz MBA: The eBusiness Knowledgebase* [online]. 2014 [cit. 2014-02-05]. Dostupné z: http://www.ebizmba.com/articles/social-networking-websites

[23] HipChat. In: *HipChat.com* [online]. 2010 [cit. 2014-02-11]. Dostupné z: http://blog-content.hipchat.com/wp-content/uploads/2010/03/web_version2.png

[24] Doručená pošta - Gmail. *Google Mail* [online]. 2014 [cit. 2014-05-08]. Dostupné z: www.gmail.com

Příloha A

Obsah CD

- /src/ - zdrojový soubor knihovny, databázový skript
- /src/API/ - zdrojové soubory API
- /src/SRC/ - grafické soubory pro uživatelské rozhraní
- /doc/ - text technické zprávy a návod k použití knihovny
- /pre/ - plakát a demonstrační video