Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Funkcionální vs. Objektové programování webových aplikací

Bc. Jan PLÍVA

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jan Plíva

Informatika

Název práce

Funkcionální vs objektové programování webových aplikací

Název anglicky

Functional vs object programming of web applications

Cíle práce

Diplomová práce je tematicky zaměřena na problematiku tvorby webových aplikací. Hlavním cílem práce je komparace funkcionálního a objektového přístupu programování webových aplikací na zvolených reálných příkladech s návazností na tvorbu komplexního řešení webové aplikace. Dílčím cílem práce je vypracování přehledu vývoje technologií a technik pro tvorbu webových aplikací.

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní práce spočívá v porovnání vybraných typů programování, porovnání jejich struktury a jejich běžné používaní. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry diplomové práce.

Doporučený rozsah práce

70 – 80 stran textu.

Klíčová slova

programování, oop, funkcionální programování, objektové programování, webová aplikace, informační systém

---

Doporučené zdroje informací

BROŽA, P. *Programování WWW stránek pro úplné začátečníky.* Praha: Computer Press, 2000. ISBN 80-7226-421-4.

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. PROVOZNĚ EKONOMICKÁ FAKULTA, – MERUNKA, V. – HAVLÍČEK, Z. *Objektově orientované paradigma ve tvorbě informačních systémů : disertační práce.* 1998.

DARIE, C. *AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně.* Brno: Zoner Press, 2006. ISBN 80-86815-47-1.

KRÁSENSKÝ, D. – LANE, D. – WILLIAMS, H E. *PHP a MySQL : vytváříme webové databázové aplikace : podrobný průvodce tvůrce WWW stránek.* Praha: Computer Press, 2002. ISBN 80-7226-760-4.

LUBBERS, P. – ALBERS, B. – SALIM, F. *HTML5 : programujeme moderní webové aplikace.* Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.

NOVÁK, T. *Turbo Pascal 5.0. 5. díl, Objektově orientované programování.* Praha: Technické a informační služby, 1990.

PÍCKA, M. – MERUNKA, V. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. KATEDRA INFORMAČNÍHO INŽENÝRSTVÍ, – PERGL, R. *Objektově orientovaná tvorba softwaru.* V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta ve vydavatelství Credit, 2004. ISBN 80-213-1159-2.

SCHUTTA, N T. – ASLESON, R. *Ajax : vytváříme vysoce interaktivní webové aplikace.* Brno: Computer Press, 2006. ISBN 80-251-1285-3.

Předběžný termín obhajoby

2017/18 LS – PEF

Vedoucí práce

Ing. Pavel Šimek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 30. 10. 2017

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2017

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 20. 11. 2017

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Funkcionální vs. Objektové programování webových aplikací" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce panu Ing. Pavlu Šimkovi, Ph.D., za jeho konstruktivní připomínky a inspiraci při konzultacích. Dále bych rád poděkoval Martinovi Hynkovi za jeho odborné rady k systému a připomínky k funkcionalitám a také panu Ing. Janu Černohorskému za jeho rady v oblasti informatiky k této problematice. V neposlední řadě patří poděkování i mé rodině za podporu při studiu.

Funkcionální vs. Objektové programování webových aplikací

Abstrakt

Diplomová práce se skládá ze dvou částí – teoretické a praktické. V teoretické práci vysvětluji, co jsou to webovém aplikace, zmínil jsem jejich historii a také to, jak se navrhují. Zaměřil jsem se i na technologie, ve kterých se dají webové aplikace vytvářet a programovat, či na jejich bezpečnost. Další kapitoly teoretické části práce se zaměřují na funkcionální a objektové programování. Vysvětluji zde, jak jednotlivá paradigmata fungují a jejich princip. Součástí této analýzy jsou také jejich výhody a nevýhody. Závěrem teoretické části je vyhodnocení jejich komparace.

Praktická část se skládá z ukázek kódu, které jsou napsány funkcionálně a naopak objektově. Dále jsem v jazyce PHP naprogramoval webovou aplikaci, kterou využívají bytová družstva. Vysvětluji zde, jak aplikace funguje, ukazuji části kódu, které vysvětluji. Závěrem pojednávám o technikách a technologiích, ve kterých se dají aplikace psát, na jejich vývoj, technologie a možnou budoucnost jejich vývoje.

Závěr diplomové práce je vyhodnocení všech výsledků, které jsem během své práce zjistil.

Klíčová slova: programování, OOP, funkcionální programování, objektové programování, webová aplikace, informační systém, objekt, třídy, Framework, databáze

Functional vs. Object programming of web applications

Abstract

The thesis is consisted from two parts – theoretical and practical. In the theoretical part I will explain what the web applications are and I also mentioned their history or how they are being proposed. I also focused on technologies which are, used to create and programming web applications. In the end I will look on their security. Other chapters of theoretical part are focused on functional and object programming – their history, principles, analysis advantages or disadvantages. The final part of this part is evaluation and comparison.

The second part – practical consist of a sample of functional and objective code. I also create web application in PHP for housing cooperatives. I explain how the application works is and I show part of code. At the end we look at the technologies which is used for web app development and their future. The conclusion of the thesis is the evaluation of all results I have found during my work.

Keywords: programming, OOP, functional programming, object programming, web application, information system, object, class, Framework, database

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafů

1 Úvod

V současné době vývoje internetu se celá problematika webových aplikací stává atraktivním a aktuálním tématem. Webové aplikace a jejich architektura se nyní nachází ve fázi tzv. WEB 3.0, který přináší interaktivní webové aplikace a mnoho dalšího. Vývojáři jsou nuceni neustále se zdokonalovat, protože nové technologie stále přicházejí a webové aplikace se každým rokem programují trochu jinak. Kvalitní programátoři jsou najímány firmami a dostávají to nejlepší ohodnocení. Je nutné mít všeobecný přehled v technologiích, ale nejlepším řešením je věnovat se pouze vybraným technologiím.

Ze světa pomalu odcházejí desktopové aplikace a jsou přesouvány na internet nahrazovány právě webovými aplikacemi. Tento fakt potvrzuje i velký počet aplikací, které již byly přesunuty na online řešení. Vývoj jde stále dopředu a již nyní jsme schopni programovat aplikace, které bychom si na přelomu tisíciletí nedokázali ani představit.

Většina webových aplikací, které se na Internetu nachází, jsou založeny na stejném principu – využívají internetový prohlížeč jako rozhraní pro práci v dané aplikaci. Nejčastěji se pro vývoj těchto aplikaci používá jazyk PHP či PERL. Oba tyto programovací jazyky jsou nezávislé na platformě a umožnují uživatelům kompatibilní přístup k technologiím typu XML a dokáží taktéž spolupracovat s databázovým systém MySQL. Největší rozdíly jsou pak v jejich syntaxi.

2 Cíl práce a metodika

2.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku tvorby webových aplikací. Hlavním cílem práce je komparace funkcionálního a objektového přístupu programování webových aplikací na zvolených reálných příkladech s návazností na tvorbu komplexního řešení webové aplikace. Dílčím cílem práce je vypracování přehledu vývoje technologií a technik pro tvorbu webových aplikací.

2.2 Metodika

Metodika řešené problematiky v diplomové práci týkající se funkcionálního a objektového programování, je založena na studiu a analýze odborných textů, článků a knih v oblasti informatiky. Teoretická část spočívá v analýze jednotlivých paradigmat programování a následné komparaci poznatků. V souvislosti s programováním webových aplikací dojde k vysvětlení obecných pojmů, jako jsou webová aplikace, technologie, apod. Dále bude následovat popis využitelných technologií pro vývoj webových aplikací v současnosti s přihlédnutím potenciálu technologie do budoucna. Své představení zde naleznou i techniky tvorby či doplňky webových aplikací, které webové stránky dělají modernější. Syntézou těchto informací dojde k posouzení výhod a nevýhod jednotlivých paradigmat a sepsání shrnutí k této problematice.

Vlastní práce pak spočívá v představení modelových situací, pro jednotlivá paradigmata, které budou porovnány na základě vícekriteriální analýzy variant pomocí metody váženého součtu. Další částí je tvorba komplexního řešení, kde bude využito znalostí programovacích paradigmat k vytvoření webové aplikace, která zde bude detailně představena a porovnána s jinými řešeními na trhu. Jednotlivé ukázkové bloky zdrojového kódu zde budou rozebrány a porovnány s jinými možnostmi vývoje.

Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry diplomové práce.

3 Teoretická východiska

3.1 Webové aplikace

Po obrovském růstu používání internetu, vytváření webových stránek a jejich následné používání se hledaly způsoby, kterými by se do stránek přidala nějaká funkčnost a dynamika. Projekty se začaly ukládat a dělit na verze, automatizovat a stále více se začaly využívat frameworky v podstatě na všechno, jelikož ohromně usnadňují další vývoj. Všechny tyto pokusy a zkoušení provozu nám postupem času ukázaly, že i webová stránka se na internetu může chovat jako desktopová aplikace. Příkladem jsou pak například Office 365 a jejich desktopový kolega MS Word. Těmto webovým stránkám se říká webové aplikace.

Orientovat se v současných trendech a stále rychlejším vývoji skriptovacích jazyků není jednoduché. V dnešní době se stále vyvíjí na bázi open-source, kde stovky vývojářů vytváří jeden projekt a společně pak pracují na současných náročných aplikacích. Zde se například hojně využije systém verzování projektu.

Webová aplikace je softwarová aplikace přístupná přes webový prohlížeč nebo program komunikující přes HTTP protokol. Skládá se tedy z tenkého klienta, který je reprezentovaný internetovým prohlížečem – dále z webového serveru a databázového serveru. (C., 2006)

3.1.1 Architektura webové aplikace

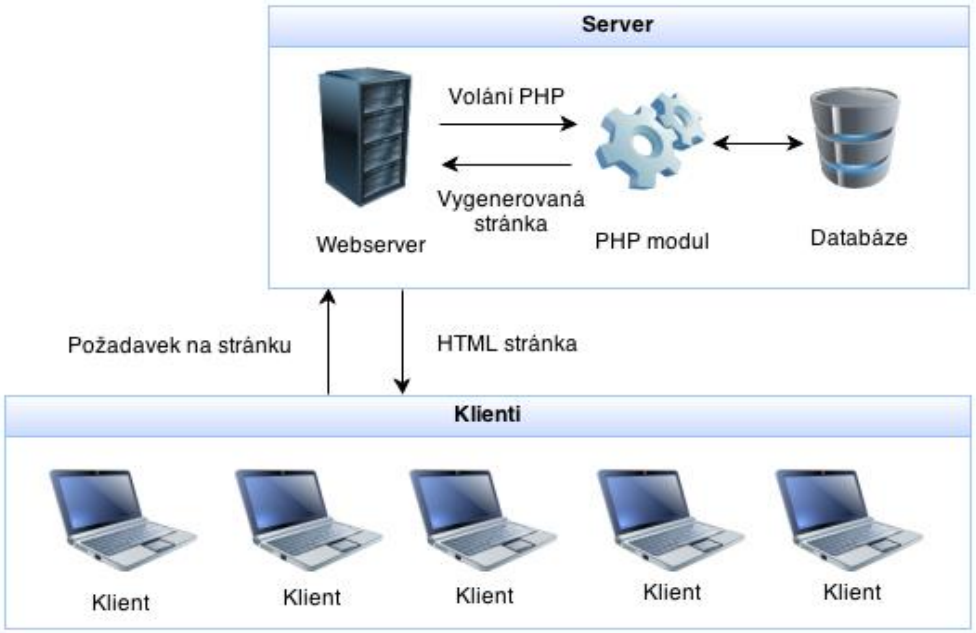
Pokud uživatel internetového prohlížeče sleduje webové stránky, dochází zde ke komunikaci mezi klientem a serverem. Při odeslání formuláře, kliknutí na odkaz nebo vyplnění odkazu do adresového řádku, dojde k zaslání požadavku na server, který internetovému prohlížeči vrátí zpět odpověď.

Webové aplikace nejčastěji fungují tak, že klient nebo uživatel se zeptá serveru na požadovaný dokument. Na straně serveru, kde běží **CGI**[1] skript, pak už tedy stránka neleží, ale je dynamicky vytvářena podle toho, co uživatel chce. Nejpoužívanějším CGI skriptovacím jazykem, ve které se webové aplikace programují je PHP. (Shklar & Rosen, 2009)

[1] CGI – program, který dokáže do stránky vygenerovat to, co uživatel požaduje

Dvouvrstvá architektura

Tento typ architektury informačních systému se nazývá „klient-server“. V tomto druhu aplikací klient zajišťuje uživatelské rozhraní a aplikační logiku a na straně serveru pak běží **relační databáze**[2]. Samotný scénář komunikace mezi klientem a serverem probíhá takto:



Obrázek 1 - Schéma komunikace - Klient - Server

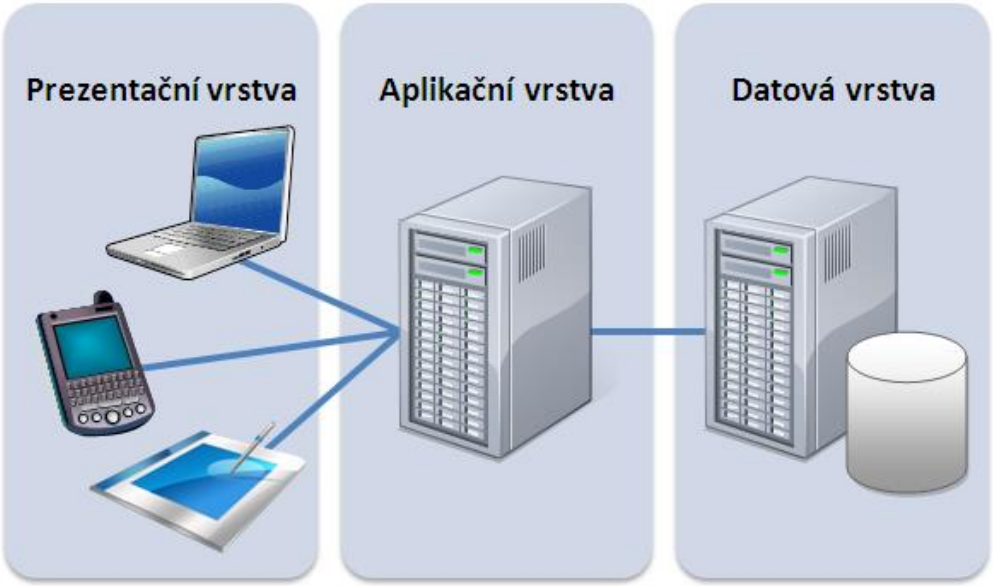
1. Uživatel zadá do prohlížeče URL adresu, kterou by chtěl navštívit a tím odešle požadavek na server.
2. Server zavolá modul PHP.
3. Pomocí PHP se zpracuje požadavek – Dojde k připojení databázi a následnému načtení dat, která klient chce. Na základě údajů, které server dostane z databáze, se vytvoří webová stránka.
4. Hotová stránka je odeslána na klientský PC, ten ji vidí jako klasickou statickou stránku, která byla dynamicky vytvořená podle jeho požadavku.

[2] Relační databáze – typ databáze, ve které jsou data uložena ve více tabulkách, které jsou vzájemně propojené

Třívrstvá architektura

Tato architektura označuje, jakým způsobem je rozdělena aplikace v rámci toho, co uživatel vidí a používá, a to co se odehrává na straně serveru. Tato architektura se snaží vyřešit nedostatky architektury dvouvrstvé a skládá se ze tří vrstev:

- **Prezentační vrstva** – viditelná část pro uživatele – zajišťuje vstup požadavků a prezentaci výsledků (webová aplikace, aplikace pro Windows, Android, iOS)
- **Aplikační vrstva** – jedná se o prostřední část modelu, komunikátor mezi prezentační vrstvou a datovou vrstvou. Vrstva zajišťuje výpočty a operace prováděné mezi vstupními a výstupními daty či požadavky. Aplikační vrstvě se někdy říká také Aplikační server.
- **Datová vrstva** – někdy označována jako databázová vrstva – nejnižší vrstva modelu, které pracuje s daty – systém řízení báze dat (SŘBD), zajišťuje výběr, agregaci, ukládání, integritu atd. dat



Obrázek 2 - Trojvrstvá architektura

Tento typ architektury využívá velké množství aplikací, které pracují s daty. Na tomto systému je postavena většina větších podnikových aplikací, některá portálová řešení či webové stránky. V současné době se stává tato architektura trendem pro výrobu robustnějších řešení, protože hlavní výhodou oproti dvouvrstvé architektuře je nezávislost

vrstev. Další výhodou je například pružnější rozdělení výkonu mezi zařízení uživatele a server – prezentační vrstva může běžet i na velmi levných zařízeních.

HTTP

HTTP je internetový komunikační protokol sloužící k přenosu dat. Využívá se k přístupu na WWW stránky a související data. Podobně jako několik dalších protokolů používá ke specifikaci umístění identifikátor URL[3]. Každá webová stránka je soubor. URL adresa nám umožňuje odkazovat se z jedné webové stránky na jinou, tím získáme umístění souboru souvisejících stránek. Idea odkazování ze stránky na jinou stránku a možnost okamžitého přístupu k odkazované stránce se nazývá hypertext.

Tento protokol byl navržen pro architekturu typu klient-server. O HTTP můžeme mluvit jako o řádkově orientovaném protokolu, který je jednoduchý na implementaci. Dále se využívá jako transportní protokol pro data ostatních služeb.

HTTP je bezstavový protokol pro prohlížení statických webových stránek a obsahuje dva typy zpráv. Jeden druh zpráv jsou zprávy, které nesou dotazy klienta na souboru na serveru, druhé pak ty, které nesou odpověď serveru na dotaz klienta.

Tento protokol má také své stavové kódy, které nám říkají, zda vše proběhlo tak jak má, nebo zda došlo k nějaké chybě. Můžeme se tedy setkat s:

1xx – informativní označení

2xx – úspěšné vyřízení požadavku

3xx – přesměrování

4xx – chyba klienta

5xx – chyba na straně serveru

HTTPS

O tomto protokolu hovoříme jako o nadstavbě protokolu http, který umožňuje výměnu hypertextových dokumentů ve formátu HTML. Nadstavba spočívá v možností zabezpečení spojení mezi webovým prohlížečem a webových serverem před

[3] **URL** – Uniform Resource Locator – řetězec, který slouží k přesnému popisu umístění dat na internetu. URL obsahuje adresu serveru, umístění souboru na serveru a protokol, kterým se přistupuje k souboru.

odposloucháváním, podvržením dat a umožňuje též ověřit identitu protistrany. HTTPS používá protokol HTTP, přičemž data, která přenášíme, jsou zašifrována pomocí TLS nebo SSL a používaným portem 443, na rozdíl od HTTP, který používá port 80.

I tento protokol využívá asymetrického šifrování, kde si obě strany před tím než zahájí komunikaci, vygenerují privátní a veřejný klíč. Pokud například uživatel zadá data do formuláře, aby se přihlásil do administrace webu, nakoupil v internetovém obchodě, nebo odebíral novinky, protokol HTTPS při samotném přenosu jeho údaje ochrání. Když se uživatelé přihlašují na webové stránky, automaticky očekávají, že jejich údaje budou zabezpečené. Data, která jsou odeslána prostřednictvím protokolu HTTPS jsou zabezpečena pomocí TSL nebo SSL, které nám poskytují tři hlavní vrstvy ochrany:

- **Šifrování** – Před odposlechem nám zašifruje přenášená data, to znamená, že při prohlížení webových stránek uživatelem, nemůže nikdo jeho konverzaci odposlouchávat, sledovat jeho aktivitu nebo ukrást jeho údaje.
- **Integrita dat** – Během přenosu nelze pozměnit ani poškodit přenášená dat, bez možnosti nezjištění této skutečnosti.
- **Ověření** – Potvrzuje, že uživatelé komunikují s požadovanými webovými stránkami. Poskytuje důvěru uživatelů.

Přenášená data jsou šifrována a při jejich přenosu nemůžou být poškozena či změněna bez detekce. Na závěr jsou pak údaje ověřeny, aby se prokázala, že uživatele komunikují s webovou stránkou.

Ve spojení s touto technologií však šifrované spojení nabízí výhodu i v hodnocení SEO. HTTPS nabízí zvýšení ranku u společnosti Google a celkově tak webová stránka působí bezpečněji. Nejnovější prohlášení společnosti Google říká, že v roce 2018 by tvůrci webových stránek a aplikací měli přejít na šifrované protokoly, jinak jejich stránky označí jako nedůvěryhodné a potencionálně nebezpečné.

Podle statistik až 43% uživatelů opustí webové stránky při zobrazení bezpečností výstražné zprávy. Do budoucna bude bezpečnost internetu jednou z hlavních věcí, na které

je potřeba se zaměřovat a je velmi pravděpodobné, že vývoj těchto technologií půjde rychlým tempem vpřed.

Výhody webových aplikací

- **Jednoduchá správa** – novou verzi aplikace jsme schopni nahrát i ve chvíli, kdy je používána ostatními uživateli
- **Vysoké zabezpečení** – pokud samotná webová aplikace neobsahuje chyby, je velmi složité aplikaci ukradnout nebo upravit, protože všechna data jsou uložena na serveru
- **Uživatelská základna** – většina populace, používající internet, jsou lidé, kteří jsou líní stahovat a instalovat, proto je pro ně pohlednějším řešením otevření odkazu a spuštění webové aplikace.
- **Vysoká kompatibilita** – ne každý má stejný počítač, operační systém či používá aplikaci v tabletu. Díky vývoji webových aplikací jsme schopni spouštět je na více zařízeních, kde vypadají úplně stejně, protože se spouští přes webový prohlížeč

Nevýhody webových aplikací

Webové aplikace mají i své nevýhody jako je například omezení rychlosti kvůli síťové komunikaci, náklady na pořízení serveru nebo nutnost připojení k internetu.

3.1.2 Návrh webové aplikace

Na počátku každého projektu na tvorbu webové aplikace je nutné převzít specifikaci od zadavatele. Zákazník, který si webovou aplikaci objednává, není odborník, a proto neví co by měla specifikace obsahovat – je tedy zapotřebí si ujasnit všechny tyto náležitosti na první schůzce.

Nejčastější chyby zákazníků je jejich zaměření na vzhled webové aplikace či objednaných stránek. Důležitými informacemi jsou fakta, která si zákazník musí uvědomit. Musí se zamyslet nad tím, proč aplikaci vytváří, co od ní očekává, dále pak jaké jsou jeho ambice, jaký je potenciál aplikace nebo v neposlední řadě komu aplikace bude sloužit.

Při koncepci nové aplikace je nejdůležitější uvědomit si její cílovou skupinu a nutnost porozumět budoucím uživatelům – vcítit se do jejich chování a potřeb. Neméně důležitá je také analýza trhu a podobných produktů na různých trzích – tím poznáme své konkurenty a jejich cílové skupiny. Vnímání přístupu jiných grafiků můžeme získat podvědomí o tom, co uživatelé chtějí a co naše cílová skupina může potřebovat, čímž se dostáváme k praktické ukázce toho, co uživatel může potřebovat. Je užitečné vypsat si nějaké informace o uživateli, pro kterého aplikaci vytváříme. Můžeme se zaměřit na jeho návyky, denní režim nebo indispozice k práci s PC a internetem. Během návrhu UX designu – user experience – je vhodné najít osoby z cílové skupiny tvořené webové aplikace a nechat je testovat vytvářené prostředí.

UX designér by měl mít při vytváření aplikace na paměti, že běžný uživatel PC nemá tak odborné znalosti o rozhraních a nedokáže tak jednoduše pochopit funkce, které se pro specialisty zdají zcela primitivní. Musí si uvědomit, že to není on, pro koho je aplikace navrhována. (Robbins, 2012)

Pokud se nám podaří identifikovat cílovou skupinu, pro kterou chceme vytvářet danou webovou aplikaci, měli bychom si také definovat, jaké úkoly bude aplikace provádět. Uživatelské rozhraní aplikace je potřeba navrhnout s ohledem na všechny tyto skutečnosti.

Ještě předtím, než začneme vybírat technologie, které budeme ve vývoji používat, se musíme zamyslet nad tím, kdo bude webovou aplikaci používat. Musíme si ujasnit, na jakých zařízeních bude aplikace používána nebo zda technologie splňuje funkcionalitu, kterou požadujeme. V současné době jsou nejpoužívanějšími technologiemi pro vytváření webových aplikací HTML ve verzi 5, které disponují už animacemi a dalšími efekty ve spojení s CSS3 a dále je to pak Javascript. Určitě nalezneme i další technologie, které by se pro vytváření tzv. Frontendu[4] daly využít. V předchozích letech se hojně využívala technologie Flash, která uspokojovala poptávku po animacích, ale ta velmi rychle upadá, a to od příchodu CSS3. S velkým rozmachem v používání mobilních zařízení a tabletu je důležité, aby webová aplikace byla přizpůsobivá a upravila se podle velikosti obrazovky zařízení.

Webová aplikace má ale dvě části. Kromě Frontendu vytváříme i Backend, který řeší všechny úkoly, úlohy a funkce na pozadí webu, které uživatel nevidí. Neví, jak probíhají

[4] **FrontEND** – část webové aplikace, kterou vidí uživatel, dokáže ji vnímat a pracovat s ní

a nemá k nim přístup. Backend se dá vyvíjet několika technologiemi a je důležité, abychom vybrali tu správnou, která nám práci usnadní. Usnadnit vývoj si můžeme například Frameworky, které mají předdefinované knihovny, které pak vývojář může využívat. Důraz musí být kladen také na řádné rozvržení aplikace ještě předtím, než začneme programovat, abychom se vyhnuli zpětnému opravování, přidělávaní nebo jiné úpravě již vytvořeného kódu.

Při návrhu webové aplikace se musí myslet i na její přístupnost, protože se jedná v podstatě o webovou stránku s dynamickým zpracováním obsahu a vzájemné interakce. Na přístupnost webové aplikace je zásadní dbát hlavně v případě, že tvoříme aplikace pro větší uživatelskou základnu a víme, že ji budou využívat různé skupiny lidí.

Přístupnost webové aplikace znamená, že ji budou bez problému využívat různé skupiny lidí. Zaměřuje se také na odstranění bariér pro uživatele s určitým zdravotním hendikepem. Špatně navržená webová aplikaci či její grafické rozhraní, které nerespektuje pravidla přístupnosti, může způsobit, že například lidé se zrakovým postižením, pomalejším připojením nebo senioři budou znevýhodněni a bude pro ně služba nedostupná. Základními důvody pro implementaci pravidel v návrhu GUI jsou:

- Rozdíl schopností uživatelů webové aplikace, jejich technické možnosti, rozumové vlastnosti, či hendikepy.
- Specifické potřeby uživatelů
- Nedostupnost stránek pro tyto osoby – snížení počtu zákazníků
- Dobře přístupný web posiluje dobré jméno značky
- Lepší optimalizace pro fulltextové vyhledávače

Existuje hned několik norem, které nám ukazují standardy a metodiky pro přístupností webových stránek a aplikací. Existují také pravidla pro **Blind friendly web**, které jak už název napovídá, pomáhají lidem se zrakovým postižením.

Před vypuštěním webové aplikace do světa pro zákazníky a budoucí klienty dochází ještě k testování aplikace. Jednou formou může být uživatelské testování. Rozdílem od heuristického testování aplikace, kdy je hodnocena a testována odborníkem, si aplikaci testují sami uživatelé, tedy konkrétně cílová skupina, pro kterou je aplikace

vytvářena. Jejich postřehy a podněty jsou zaznamenávány a důkladně monitorovány. Uživatelé mohou totiž odhalit skutečné problémy, které vývojář či odborník neodhalí, protože ho například nenapadne psát speciální znaky do položky číslo účtu.

Dalším druhem jak lze aplikaci ještě otestovat je A/B testování, které srovnává několik variant téhož objektu a jejich vliv na uživatele.

3.1.3 Technologie a techniky tvorby webových aplikací

Tato podkapitola je zaměřena na technologie a techniky vývoje webových aplikací v současnosti a do budoucna. V současné době existuje celá řada technologií a technik, které jsou pro vývoj webových aplikací nezbytné. Využíváme také technologie, které vývoj webových aplikací ulehčí nebo vytvoří větší komfort potencionálnímu zákazníkovi. Budoucnost vývojových jazyků a technologií závisí na potřebách zákazníků ale také na oblíbenosti mezi vývojáři.

3.1.3.1 Technologie

Vývoj webových aplikací přináší setkání se s mnohdy novými technologiemi, které jsou aktuální novinkou, nebo se jejich koncept vyvíjel již několik let. Technologie tvorby webových aplikací se velmi rychle mění, během jednoho roku se můžeme setkat s několika novinkami a naopak technologie používané už přestanou být atraktivní a trend se změní.

Webové aplikace musí být stále plné nových technologií, aby jejich kvalita byla pro zákazníky ta nejlepší a právě kvůli funkcím a efektům dochází ke zvyšování konverzí. V následujících pár odstavcích se podíváme, jaké technologie jsou nejvyužívanějšími při tvorbě webových stránek či aplikací, včetně zmínky a možném budoucím rozvoji této technologie. Existuje také řada systému, které se specializují na vytváření webových aplikací – jsou to tzv. editory. S využitím těchto programů dokážeme docílit lepších výsledků v počtu chyb v kódu aplikace, a to především díky jednoduchosti a větší přehlednosti kódu.

IDE vs. Editor

K vývoji takovýchto aplikací nám nestačí poznámkový blok, ale budeme potřebovat mnohem sofistikovanější nástroje. Jaký je tedy rozdíl mezi editorem a IDE, když v obojím softwaru máme zvýrazněný kód, nápovědu a přehledné rozhraní?

Na rozdíl od editoru IDE opravdu kódu rozumí a vytváří tak propojení mezi knihovnami, soubory či frameworky. Jednoduše dokáže porozumět tomu, co vývojář píše. Ve skutečnosti to znamená, že požadavek editoru na nápovědu v určitém místě, kde vývojář očekává proměnnou, dostane seznam funkcí, proměnných, anebo často se vyskytujících slov apod. Pokud budeme požadovat nápovědu na místě, kde očekáváme proměnnou, po **IDE**, dostaneme na výběr pouze proměnné, ke kterým si navíc můžeme zobrazit v rychlém náhledu také dokumentaci.

IDE obsahuje také spoustu nástrojů, jako například nástroje pro kompilaci, debugging[5], apod. Tyto nástroje se také do většiny chytrých editorů dají doplnit v podobě rozšíření, většinou vytvářené komunitou.

Internetový prohlížeč

Při vývoji jakékoli webové aplikace či webové stránky je nezbytné provést testování v různých druzích prohlížečů. Mezi ty nejznámější patří Google Chrome, Mozilla Firefox a Internet Explorer. Každý prohlížeč má jiné jádro, které se často liší od ostatních, a to jak v desktopové, tak v mobilní verzi. Tyto verze se mohou také lišit svými pravidly na zobrazování. Dalším rozdílem jsou základní styly, které se nejčastěji resetují pomocí speciálních stylů.

V dalších částech této kapitoly bude představeno několik skriptovacích a programovacích jazyků, technologie či techniky vývoje webových aplikací. Všechny tyto technologie mají společné vlastnosti, díky kterým došlo k jejich zařazení do této práce. Těmito vlastnostmi je jejich oblíbenost, či nenahraditelnost a všestranné použití.

Javascript

JavaScript je jednoduchý, multiplatformní, interpetovaný a objektově orientovaný programovací jazyk. Byl původně realizován jako součást webových prohlížečů tak, aby klientský skript mohl komunikovat s uživatelem, řídil prohlížeč, běžel asynchronně a mohl dynamicky měnit obsah už zobrazené stránky.

Syntakticky se jádro JavaScriptu podobá jazykům C, C++ a Java s programovacími konstrukcemi jako jsou IF, WHILE a && operátorem. Když se podíváme na syntaxi samotného jazyka, tak zde podobnost končí – JavaScript je totiž

[5] Debugging – nástroj, se kterým můžeme odhalit chyby

typovaný jazyk tzn. datový typ proměnné je vázán na hodnotu, nikoliv na proměnnou. Objekty v JavaScriptu jsou spíše jako asociativní pole Perlu než jako struktury C, C++ nebo Javy. Také jak již bylo uvedeno JavaScript je čistě interpretovaný jazyk na rozdíl od C nebo C, které jsou kompilovány a Javy, která je nejprve kompilována na bitový kód a poté interpretována.

JavaScript je ovšem využíván i v aplikacích mimo webové stránky – v PDF1 dokumentech, rozšířeních pro internetové prohlížeče nebo v aplikacích pro GNOME2 Shell. (Robbins, 2012)

Programování pomocí Javascriptu přineslo webovým aplikacím mnoho změn. Jednou z nich bylo představení nejnovější verze javascriptového frameworku Angular 4, který přináší této technologii jen výhody a zveličení již známých důvodu toho, proč je dobré Javascript využívat. Hlavními nosnými pilíři Javascriptu jsou jeho efektivita, bezpečnost a cena vývoje. Javascript umožňuje zrychlit již staré aplikace předěláním či upravením funkcí ze starých verzí. Nejvyhledávanější JS frameworky mají vysokou míru zabezpečení a jsou připravovány pro velké komunity s tisíce členy. Zajímavostí je, že většina JS frameworků jsou pod otevřenou licencí a zdarma. Existují také knihovny, manuály a příručky, které tento trend pomáhají rozšiřovat.

Budoucnost Javascriptu bude určitě vzkvétat obzvlášť ve spojení s využitím umělé inteligence. Je vysoce pravděpodobné, že vzniklé frameworky od technologických společností světa se budou již standardně držet na předních příčkách ve vývoji.

Angular JS je často představován jako MVW (Model-View-Whatever) framework, který skrývá mnoho benefitů a výhod pro začínající projekty po středně velké firmy. Mezi jeho přednosti patří rychlost vývoje, minimální čas nutný pro testování aplikace a dvoucestné zpracování dat, kde se změna na pozadí aplikace ihned zobrazí v uživatelském rozhraní. Jedná se nejpoužívanější JS framework pro jednoduché aplikace tzv. (Single-Page Apps) s početnou komunitou developerů ochotných poradit a dělit se o své zkušenosti.

Dalším javascriptovým frameworkem je React JS, představený společností Facebook. V největší míře je samozřejmě využíván jako hlavní vývojový nástroj pro aplikaci Facebook a Instagram od výše zmíněné společnosti. S MVC (Model-View Controller) může být jednoduše integrován s jakoukoli architekturou webové aplikace. Využíváním virtuálního DOM podporuje spolehlivé výsledky testování a zvyšuje své

možnosti na výkon aplikace. React není sice nejpoužívanějším, ale je nejrychleji rostoucím frameworkem co se týče počtu vývojářů. Jeho další předností je jednoduchá cesta k pochopení celého konceptu frameworku. Může být totiž směle jednoduchý k vytvoření komplexních, vysoce náročných a neuvěřitelných projektů.

Zatímco nejnovější z frameworků **Vue JS** se zatím netěší takové vývojářské základně jako Angular či React, tak jeho představení v roce 2016 vyvolalo rozruch. Tento framework přináší developerům webových aplikací to nejlepší z Emberu, Angularu a Reactu a spojuje všechny do jednoho vývojářského balíku. Tvůrci tím chtějí docílit větší oblíbenost, míru pochopení a rychlejší proces učení. Je správnou volbou pro rychlý vývoj v multiplatformním prostředí.

V neposlední řadě je určitě nutné zmínit poslední dva Javascript frameworky a to Ember JS a Meteor JS. Ember JS byl v roce 2015 jmenován jako nejlepší framework pro webové aplikace a Angular a React nechával lehce za svými zády. Nyní se také těší velké oblibě developerů a díky častým updatům a široké developerské základně se tento framework stále velmi hojně používá.

Meteor JS přichází s mnoha novými vlastnostmi pro vývoj backendu, pro renderování frontendu a správu databází či obchodní logistiku. Tato platforma umožňuje rychlý vývoj webových a mobilních aplikací typu end-to-end v čistém jazyku Javascript. Z hlediska výkonu jsou všechny změny ihned přenášeny z databáze do uživatelského rozhraní.

jQuery

Javascriptový framework, který nabízí jednoduché vyhledávání elementů v DOMu, jejich modifikaci a samotné vytváření. Jeho další hlavní možností je práce s událostmi, dále pak využívání velkého spektra funkcí pro práci s poli, či podpora Ajaxu a animací. Tento nejpoužívanější Javascript framework je vhodný pro realizaci menších a středních projektů. (Nixon, 2014)

	AngularJS	Angular 2	ReactJS	Vue.js	Ember.js	Meteor.js
Definition	MVW framework	MVC framework	JavaScript library	MVC framework	MVC framework	JavaScript app platform
1st Release	2009	2016	2013	2014	2011	2012
Homepage	angularjs.org	angular.io	reactjs.net	vuejs.org	emberjs.com	www.meteor.com
# Contributors on GitHub	1,562	392	912	62	636	328
GitHub Star Rating	54,402	19,832	57,878	39,933	17,420	36,496

Obrázek 3 - JS Frameworky do 1. 1. 2017[6]

Na obrázku výše můžeme vidět JS frameworky do začátku roku 2017. V roce 2017 jsme se mohli seznámit s již zmíněným Angularem 4, apod. Díky němu můžeme vidět, že správný výběr a nastavení jazyka Javascript není o funkcích, které může nabídnout. Kritériem je i jaká společnost ho vytvořila a jaké má cíle a nabídky na projekty. V budoucnu však Javascript bude dále nezbytnou součástí vývoje webových aplikací společně s jeho často využívanými frameworky. (Flanagan, 2011)

AJAX

Původem tato zkratka pochází z anglického Asynchronous Javascript and XML. Jedná se o moderní technologii v současné době často používanou při vytváření nejnovějších webových aplikací. Samotná technologie je součástí RIA – tedy nového směru vývoje webových aplikací, který vede k intuitivnějšímu ovládání, lepšímu uživatelskému komfortu a k lepší funkčnosti aplikace.

[6] Zdroj: https://cdn-images-1.medium.com/max/1600/1*ADuCd_GcORWlpCzTASmkxQ.png

Nejedná se ale o novinku, o novou technologii. AJAX je kombinací dvou již známých technologií – Javascriptu a XML. Nyní se podíváme, z čeho se AJAX detailněji skládá:

- Základní standardy prezentace pomocí XHTML a CSS
- Dynamické zobrazení a interakce pomocí DOM (Document Object Model)
- Výměna dat a manipulace s využitím XML a XSLT
- Asynchronní načítání dat pomocí technologií XMLHttpRequest a JavaScript

Java

Java je jedním z nejpopulárnějších programovacích platforem na světě. Její vývojářská základna má přes 9 milionů developerů. Hlavními důvody jejího úspěchu jsou jednoduchost a všestrannost – kód založený na jazyce Java může být spuštěn více platformami. Díky velkému počtu zařízeních a operačních systému, které jsou využívány pro spouštění aplikací má Java univerzální řešení a s novým vydáním tak přináší možnost pro další rozvoj vývoje.

V minulosti se jednalo o méně oblíbený jazyk, který by se v budoucnosti mohl těšit velké oblibě. S příchodem nové, již zmíněné, verze se Javě zvýšila rychlost a řádky kódu se viditelně zkrátily. Java dostala spoustu nových funkcí a její bohaté portfolio tak dělá z tohoto vývojářského jazyka atraktivní změnu ve vývoji.

Python

Python je objektově orientovaný dynamicky silně typový jazyk s velmi příznivou křivkou učení, dobrou přenositelností a přehlednou syntaxí. Python patří mezi jazyky, které se dají rychle naučit, ale dá se v něm snadno a efektivně pracovat. Jedná se o jazyk interpretovaný, zdrojový kód je tedy podobně jako u Javy překládán do mezikódu. Vývojář si zde může snadno a rychle vytvářet moduly v C/C++ - je tedy snadné optimalizovat některé kritické části aplikací. V současnosti se však Pythonu vrací jeho zašlá sláva. Mezi vývojáři se stává opět oblíbený a lze to pozorovat i v čím dál častějších zmínkách o tomto jazyce. Široké podpoře se těší především mezi vědci v oblasti uchovávání dat, ale také mezi lidmi z oblasti analýzy dat. Svou užitečnost ukazuje prostřednictvím softwarových inženýrů, vývojářů, apod. V žebříčcích hledanosti výukových programů

na Google se Python řadí na druhé místo, a to i díky tomu, že je to hlavní jazyk využívaný pro určování směru programování umělé inteligence.

PHP

Jedná se k skriptovací jazyk s širokým využitím pro vývoj webových aplikací ve spolupráci s HTML kódem. Skripty psané v PHP se spouštějí na straně serveru, což zamezí uživateli odhalení logické podstaty vykonávaných funkcí. Za největší výhodu je považováno rychlé a snadné pochopení logiky jazyka. Začátečníci nemají s pochopením či naučením se větší problémy. PHP taky umožňuje spoustě pokročilým programátorům možnosti, o kterých třeba ani oni nevěděli. Mezi hlavní funkce patří sběr dat z formulářů, generování dynamického obsahu stránky, nebo příjem a odesílání souborů Cookies. Hlavní výhodou PHP oproti PERLU je v jednoduchosti, protože díky jednoduchosti a rychlosti formulování a fungování všech funkcí je právě PHP předurčen ke skvělé spolupráci s MySQL.

V nejnovější verzi PHP 7 se tento skriptovací jazyk opět stává velmi oblíbeným mezi vývojáři. Mezi jeho hlavní výhody patří jeho dostupnost, počet frameworků a také možnost rychlého rozvoje a jeho všestrannost. Kód vytvořený v PHP umožní vývojářům multiplatformní přístup k aplikaci. Díky tomu můžeme o PHP říci, že je flexibilní a škálovatelný. Vesměs široká nabídka rozšíření a doplňků přidává na jeho ratingu. Kromě toho podporuje také systémy správy databází, taky i jejich open source řešení.

Práci s PHP ulehčují tzv. **Frameworky** – ulehčují nám a zefektivňují celý proces vývoje webových aplikací. Hlavními výhodami frameworků je rychlejší vývoj, dále poskytují dobře organizovaný a opětovně využitelný, udržovatelný zdrojový kód. Dokáží růst v průběhu času, protože webové aplikace běžící na frameworcích jsou škálovatelné. Dalším důležitým faktorem, proč využívat frameworky je, že se zbavíme starosti souvisejících s nízkým zabezpečením webu. Dodržují vzor MVC (Model View Controller), který zajišťuje separaci prezentace a logiky a v neposlední řadě prosazují moderní webové vývojové postupy, mezi něž patří nástroje objektově orientovaného programování.

1) Laravel

I přesto, že se jedná o poměrně nový PHP framework, který byl vydán v roce 2011, je podle nejnovějších průzkumů nejoblíbenějším frameworkem mezi vývojáři. Laravel má

obrovský ekosystém s platformou připravenou k okamžitému hostování a rozmisťování a jeho oficiální web nabízí mnoho návodů ve formě screencastů – Laracasty.

Laravel má mnoho schopností, které činí možným rychlý vývoj aplikací. Má také svůj odlehčený šablonový engine s názvem „Blade“, elegantní syntax usnadňující úlohy, které je třeba dělat často, jako jsou autentizace, relace neboli sessions, řazení do front či cáchování. Laravel také zahrnuje lokální vývojové prostředí zvané Homestead.

Zde přikládám oficiální text, kterým se Laravel prezentuje.

Laravel je webový aplikační framework s výstižnou a elegantní syntaxí. Věříme, že vývoj aplikací může být zábavná a tvůrčí činnost, která vývojáře opravdu baví. Laravel se snaží obtížnosti u vývoje ulehčit zjednodušením běžných funkčností většiny webových projektů, jako je např. autentizace, routování, práce se sessions nebo kešování. Cílem frameworku Laravel je, aby byl proces vývoje aplikací příjemný pro vývojáře, aniž by to ale negativně ovlivnilo jejich funkčnost. Šťastní vývojáři tvoří ten nejlepší kód. Pro tento účel jsme se snažili zkombinovat to nejlepší, co jsme se naučili u ostatních frameworků, a to bez ohledu na programovací jazyk (např. Ruby on Rails, ASP.NET MVC a Sinatra). Laravel je dostupný, ale výkonný framework, poskytuje nástroje pro velké a robustní aplikace. Vynikající IoC container, přehledný migrační systém, pevně integrovaná podpora pro jednotkové testování. To jsou nástroje pro aplikace, které vytváříme. (Laravel, 2017)

2) Symfony

Tento framework je určený pro vývoj a programování webových stránek a aplikací. Je napsán v jazyce PHP a jedná se o svobodný software, který se šíří pod licencí. První verzi tohoto frameworku jsme mohli spatřit v roce 2005, jehož autorem je Fabien Potencier z francouzské agentury Sensio, která nadále tento framework vyvíjí a dotuje.

Komponenty a funkce frameworku Symfony 2 používají mnohé robustní projekty. Nejznámějšími jsou Drupal, což je systém pro řízení obsahu stránek, dále pak například phpBB, které nám spouští webové stránky ve smyslu diskuzních fór. Mimo jiné se na tento framewok spoléhá i Laravel. Tento framework se také těší velké oblibě a má mnoho příznivců a rozsáhlou vývojářskou komunitu. (Symfony, 2017)

3) CakePHP

Framework, který je vhodný spíše pro menší aplikace. Je podstatně jednoduší než jiné frameworky jako jsou Zend či Symfony. I když patří mezi nejjednodušší frameworky, neznamená to, že je osekaný o komponenty a funkce. Je dobře vybaven. Dále je možné ho rozšiřovat o různé pluginy vytvořené komunitou, ale integrace komponent třetích stran je už obtížnější. Chybí mu systém na šablony. Je spíše vhodný pro začátečníky. Stejně jako Symfony je distribuován MIT licencí. (Software, 2017)

4) Zend

Zend je velmi robustní a stabilní PHP framework, který je obalený spoustou konfiguračních voleb, proto se obvykle nedoporučuje pro menší projekty – naopak je však vynikající pro ty složitější. Má velmi širokou základnu uživatelů, a proto je k němu mnoho návodů a různých rad, které se dají velmi rychle a snadno najít. Vytvořila jej společnost Zend Technologies, která se spolupodílela na tvorbě samotného jazyka PHP. Je založen na jednoduchosti objektově orientovaných osvědčených postupech. Celý framework je navržen tak, že vývojář si může vybrat jen jednotlivé části, které potřebuje, což zrychluje aplikaci, urychluje vývoj a snižuje náklady a čas nutný na údržbu aplikace. Využívá moderní technologie, jako jsou AJAX. Jeho zdrojový kód je licencovaný pod New BSD licencí. (Technologies, 2017)

5) Nette

Jedná se o framework, který umožňuje vytváření aplikací založených na architektuře MVP. Autorem tohoto frameworku je David Grundl, který framework vytvořil v jazyce PHP 5 s plným využitím objektů a otevřeným kódem. Vývojář, který v tomto frameworku vytváří aplikaci, se musí přizpůsobit určitým pravidlům, které mu mají usnadnit tvorbu aplikace. (Anon., 2017)

Podle světových statistik používá na straně serveru PHP více než 82% webových stránek a to nejen díky tomu, že se jedná o hlavní programovací jazyk WordPressu – systému správy obsahu. V budoucnosti se bude pomocí PHP a jeho frameworku vyvíjet většina webových aplikací a stránek. O jeho životnosti není radno pochybovat.

Perl

Perl je programovacím jazykem, který je primárně určen jako pomůcka pro správu systému. Je výborným nástrojem pro práci s textem. I když je Perl jazykem pro správu

systému, můžeme ho bez problémů použít i jindy, například pro prototypy složitějších programů a zajisté všude tam, kde byl až do teď používán skript psaný v Shellu.

Slovo „Perl“ je zkratkou z anglického *(Practical Extraction and Report Language)* a je svobodně šířen pod GNU licencí, proto jej lze volně používat na většině známých architektur a operačních systémů – patří sem například celé spektrum variant OS UNIX dále VMS a dokonce i DOS a jeho nástupci. V posledních letech získal Perl oblibu hlavně mimo jiné i v bioinformatice.

Tento programovací jazyk má určitou podobnost s přirozeným jazykem *(angličtinou)* a tím je poměrně snadno čtivý a čitelný, i pro člověka, který se programováním nezabývá. Samotná syntaxe není přesně definována jako u ostatních programovacích jazyků. Deklarace proměnných nemusí být přímá, dále v Perlu nemůžeme definovat vlastní datové typy a nenajdeme zde typovou kontrolu. Jazyk nedělá rozdíly mezi racionálními a celými čísly, bere všechny stejně – jako čísla racionální – tedy jako jeden datový typ.

Perl byl v minulosti připraven a napsán pro platformu UNIX, ale nyní se používá i na systémech jiných, jako jsou Windows, MS DOS, Novell. Tento programovací jazyk je vhodný pro práci s databázemi, protože jeho databázové rozhraní DBI podporuje práci s databázemi, jako jsou Oracle, Sybase, PostgreSQL, MySQL.

MySQL

V dnešní době webové aplikace nejčastěji používají MySQL databáze. Jedná se tedy o světově nejrozšířenější open source databázový systém, který je jednou z klíčových součástí LAMP (Linux, Apache, MySQL, PHP/PERL/PYTHON). LAMP se v současné době používá u velkého počtu webových aplikací, protože celé toto řešení je velice spolehlivé a postavení open source základu výrazně snižuje finanční náklady na vývoj i běh aplikace. Pro tento systém byla vyvinuta i sada grafických nástrojů, které jsou obdobou proprietárních databázových systémů. Tento druh databáze běží na širokém spektru platforem a také může běžet na jakémkoliv počítači. (Meloni, 2012)

HTML a CSS

Kaskádové styly jsou další stavební jednotkou webových stránek a aplikací. Pomáhají nám společně s HTML vytvářet vizuální podobu webových stránek – vzhled.

Na tento jazyk se můžeme dívat jako na soubor pravidel, které lze skrze své vlastnosti a hodnoty přidělit jednotlivým HTML elementům. Kaskádové styly disponují vlastnostmi pro formátování textu, kde nás zajímají fonty písma, velikost písma a jeho barva. Dále pak definice rozvržení, které nám pomáhají stránku intuitivně zorganizovat a umístit elementy tam, kam je potřeba. Další nepochybně důležitou vlastností je řízení tisku – je nutné vědět, co se má tisknout a co není potřeba. Za zmínku stojí taky možnost využití některých dynamických vlastností, které umožňují elementy zobrazovat a skrývat.

V dnešní domě jsou již hojně rozšířené preprocesory **SASS** a **LESS**. Tyto preprocesory jsou kompilované a umožňují vývojáři využívat proměnné, funkce a spoustu dalších prostředků, které usnadní práci a udělají kód přehlednější.

CSS3

Nejnovější verze kaskádových stylů obsahuje navíc oproti předchozím verzím pokročilé selektory – což nám dává možnost jednoduše identifikovat elementy dle různých kritérií a díky tomu učinit zdrojový kód mnohem přehlednější. Konkrétně se pak můžeme v tomto případě bavit například o identifikaci lichých a sudých řádků tabulky, všech vybraných zaškrtávacích políček nebo posledního odstavce ve skupině. CSS3 nám také nabízí vývoj bohatších a interaktivnějších webových aplikací. V neposlední řadě je důležité zmínit také možnost pracovat s vizuálními efekty. Napomáhají nám při přidávání stínů, přechodů mezi elementy, bez nutnosti spoléhat na obrázky na pozadí. Využít se dají i transformace, které zajišťují vytvoření kulatých rohů, zkosení či rotaci elementů.

Co se týče kompatibility – některá pravidla jsou speciální pro určité prohlížeče, ale v dnešní době jsou již prohlížeče natolik „sjednocené“ že nedochází až k tak markantním rozdílům v zobrazení stejného kaskádového stylu v několika prohlížečích. (Duckett, 2014)

Bootstrap

Bootstrap, dnes hojně využívaný framework, který se skládá v podstatě s HTML, CSS a Javascriptu. Obsahuje sadu nástrojů usnadňující práci při vývoji frontendu webové aplikace. Jeho vznik je spojen se sociální sítí Twitter. Nabízí mnoho komponent, které zjednodušují práci vývojářům – konkrétněji pak lehčí vytvoření responzivního webu – tedy zobrazení webové aplikace na různých druzích obrazovek a rozlišení. Můžeme tedy konstatovat, že v dnešní době je díky tomuto frameworku vývoj jednoduší a rychlejší.

Swift

Tento programovací jazyk neslouží primárně k vytváření webových aplikací, ale k potřebě zákazníků je velmi nutné webové aplikace upravovat do mobilních aplikací. Swift se v posledních letech rozšířil díky společnosti Apple a jeho budoucí vývoj předpovídá stálý růst. Většina mobilních vývojářů potvrzuje, že Swift je mnohem jednoduší než jeho předchůdce – C. Swift není programovací jazyk, který bude nahrazovat objektové C, ale je jeho lepší variantou. Vývojářům taky s příchodem Swift 2 byla připravena nová funkce pro práci s hostiteli. Swift rozhodně zůstane v programovacích trendech budoucnosti.

ASP. NET

Mezi vývojáři hodnocenými frameworky patří také velmi silný nástroj ASP.NET od firmy Microsoft používaný od roku 2002. Jedná se o nástroj vytvářející dynamické webové stránky, aplikace a portály. Tento programovací jazyk je založen na běžné jazykové proceduře CLR, která poskytuje příležitost psát kód a vytvářet aplikací pomocí libovolného jazyku podporující .NET. V současné době existuje velké množství individuálních vývojářů a společností, kteří se vývojem pomocí technologie ASP.NET zabývají a preferují tak tuto technologii pro vývoj webových aplikací. Tento framework je také pod otevřenou licencí a jeho podíl na trhu je téměř 16%. V budoucnosti se tato technologie bude také držet a je pravděpodobné že bude růst, protože je úzce spojena s Microsoftem a jeho platformami.

3.1.3.2 **Techniky**

Webové stránky tvoří také spoustu technik, které zajišťují dostatečný komfort pro zákazníka. Návštěvníci webových stránek si zvykly na standard a jsou nároční a proto je důležité pokračovat v nových trendech v oblasti webových aplikací, které budou zvyšovat úroveň webu, na místo toho, abychom se soustředili na technologie a trendy, které máme doposud.

V konkurenčním digitálním světe vedly podniky k vlastním ohromujícím výsledkům nové techniky a trendy. Podle průzkumu identifikovalo 77% digitálních agentur nedostatky na webu z hlediska UX – uvedli to jako svou největší slabinu. Stále více lidí nakupuje online a proto právě UX je hlavním pilířem vývoje webových aplikací. Každý rok se trh

online prodeje zvyšuje o 20% a očekává se, že v roce 2020 to bude 4 biliony dolarů oproti 2,3 bilionu v roce 2017.

Chatbots & UI (Umělá inteligence)

Chatbot je skvělá technologie v podnikání, která bez manažera komunikuje se zákazníkem. Podnik tak získává všechny potřebné informace a zákazník si myslí, že komunikuje se zákaznickou linkou. Tyto chatboty jsou inteligentnější a do budoucna zajistí kvalitu služeb, které poskytují uživatelům webových stránek.

Další výhodou je ulehčení práce vývojářům elektronického obchodu, kteří obchod vyvíjejí s online pomocí a dokáží se tak vžít do role zákazníka. Díky chatbotům je možné si objednat jídlo, lístky a další věci jediným kliknutím – do roku 2020 se předpokládá, že přes 85% interakcí ve webových aplikací se zákazníky budou dělat právě chatboti.

Další technologií, která se rozvine, bude spojení chatbotů s umělou inteligencí, kdy dojde k porozumění komunikace mezi chatbotem a zákazníkem – porozumí živému projevu a budou se stále učit.

Pohybové rozhraní

Pohybové rozhraní neboli „Motion UI“ může být přelomovým prvkem webových aplikací. Vzhledem k tomu, že většina uživatelů je unavená s velkých a četných animací na stránce, tato technologie stále dbá na jednoduchost. Vývojáři webových aplikací navrhují webové stránky s moderním designem, protože je zajímá maximální hodnocení – prostřednictvím pohybového rozhraní se tak šoupnou o příčku výše.

Sofistikované pohybové rozhraní bude brzy považováno za populární. Vývojářům se nově poskytne možnost dodání stylu a vytvořit tak webovou stránku unikátní oproti tisícům statických webů.

BlockChain

Jedná se o nejefektivnější způsob společného ukládání dat. Obrovské množství počítačů po celém světě uchovává mnoho informací, které nejsou uloženy na jednom místě. V tomto případě poskytuje BlockChain vysokou úroveň zabezpečení. Hlavní výhodou je, že mezi transakcemi nejsou žádní zprostředkovatelé. Každá transakce je ověřena a spoléhá na složitý algoritmus, a to i v případě, že již existuje velký počet počítačů.

O zavedení této technologie uvažuje mnoho mezinárodních bank, které chtějí zajistit maximální bezpečnost svých drahých dat.

Video na pozadí

Jedná se o jednu z nejzajímavějších platforem na získání maximální pozornosti návštěvníků webových aplikací. Tuto technologii hojně využívají soukromé firmy, kterým přehrávání videa na pozadí umožňuje lepší prezentaci svých produktů a služeb jednotlivým návštěvníkům. Jedná se tedy o jednoduchý a velmi účinný nástroj pro zvýšení konverze. Podle statistik ve skutečnosti video obsah zvyšuje tržby o 64 až 85% - lidé neradi čtou, ale raději se podívají na video. Předpokládá se, že v budoucnu do roku 2020 bude pro popis produktu využívat video obsah každý internetový obchod.

Rozšířená realita

Rozšířená realita patří mezi rychlý a interaktivní nástroj pro webové aplikace. V mobilních zařízeních pomáhá webovým aplikacím oslovovat cílové publikum. Jedním z příkladů je například společnost Snapchat, která díky rozšířené realitě umožňuje uživatelům vytvářet smajlíky z vhledu svého obličeje.

Webové stránky na jednu stránku

Jedná se o „nekonečné“ webové stránky, které svým uživatelům poskytují úplné informace o příslušných stránkách, aniž by se zákazník musel přesunout na stránku jinou. Z tohoto důvodu šetří čas navigační menu, ve kterém se uživatelé neztrácejí, další mnohdy nesrozumitelné masy textu a obrázků. Tyto typy stránek jsou nejčastěji používány pro prezentaci produktu, nikoliv však pro elektronický obchod. Pro firmy pak tyto stránky nabízejí několik výhod včetně snížení nákladů na jejich tvorbu či nárůst konverzí.

Push notifikace

Návštěvníci webu mají rádi interakci s prostředím což jim push notifikace přináší. V mobilních aplikacích či na webech jsou tyto notifikace velmi efektní a očekává se, že mnohem více firem zařadí tento druh interakce se zákazníkem do svých webových aplikací. Hlavním přínosem je, že lze jednoduše udržet angažovanost lidí a jejich spolupráci za využití malého úsilí.

3.1.4 Bezpečnost webových aplikací

Webové aplikace jsou pro zaměstnance či zákazníky velmi výhodné, protože mají nepřetržitý přístup ke svým datům a účtům. Můžeme také mluvit o situaci, když nějaký provozovatel intranetové sítě chce pustit tuto síť do internetu a umožnit tak svým klientům, aby měli větší komfort. Pokud se rozhodneme pro spojení aplikace s internetem, musíme dbát především na bezpečnost. Možností, jak lze webové aplikace ohrozit, existuje nespočet. Stejné riziko také padá na servery, které tyto aplikace provozují.

Hrozby pro webové aplikace

Téměř 80 % útoků je mířeno na aplikační vrstvu webové aplikace. Na tuto skutečnost by měli myslet především vývojáři a architekti aplikací a měli by nové webové aplikace navrhovat s ohledem na aktuální bezpečnostní hrozby. Pokud již ze začátku přistoupíme k přípravě, návrhu či vývoji aplikace zodpovědně, riziko napadení a prolomení se snižuje.

V současné době to ale takto bohužel nefunguje. Všichni provozovatelé webových aplikací, chtějí mít co největší nabídku funkcí, ženou se za co nejlepší dostupností a jednoduše se snaží vytvořit nejvíce komfortní a intuitivní web pro své zákazníky, pak jsou otázky bezpečnosti opomíjeny a zanedbávány.

Zabezpečení při návrhu a testování aplikace

Bezpečností webových aplikací se zabývá také organizace The Open Web Application Security Project[7], která již několik let sleduje bezpečností hrozby, kterými mohou být aplikace ohroženy. Díky monitoringu, který organizace zajišťuje je schopna vytvořit seznam nejčastějších chyb v zabezpečení webových aplikací.

Cross Site Request Forgery (CSRF)

Samotná podstata CSRF útoku spočívá v tom, že uživatele přimějeme navštívit stránku napadené aplikace, která provádí nějakou akci, aniž by o tom uživatel věděl. Díky tomu můžou být útoky vedené proti aplikacím, do kterých se potencionální útočník může sám přihlásit a tím zjistit jejich strukturu.

[7] **OWASP** – nevýdělečná organizace zabývající se především bezpečností softwaru zejména v oblasti webu

Obranou před útoky CSRF jsou ověření, zda uživatel provedl operaci dobrovolně. Toho můžeme dosáhnout vložením a následnou kontrolou autorizačního tokenu.

Cross Site Scripting (XSS)

U této chyby můžeme jednoznačně říci, že se jedná o nejčastější chybu, se kterou se můžeme setkat. XSS vzniká v okamžiku, kdy aplikace odesílá uživatelská data webovému prohlížeči, aniž by nejprve tento obsah ověřila nebo zašifrovala. Díky této situaci pak potenciální útočník může spustit škodlivé skripty v prohlížeči a číst uživatelskou relaci, změnit webové stránky nebo řídit phisingové a malwarové útoky. Nejčastěji se k těmto útokům využívá JavaScript, který umožňuje útočníkům manipulovat a měnit jakoukoliv vlastnost stránky. XSS se rozděluje dále na dva typy. Prvním z nich je **Persistent XSS**, které se nejvíce využívá i návštěvních knih, diskuzních fór, nebo jiných webových aplikacích, které ukládají data do databází. Oproti tomu existuje tzv. **Non-persistent XSS**, kde je problematika zaměřena na URL adresy, vyhledávací formuláře, zaškrtávací políčka – jednoduše řečeno na vše co zadaný řetězec neukládá, ale pouze ho zpracuje a pošle na výstup.

Bránit se před XSS není jednoduché. Každá webová aplikace má několik desítek, či stovek vstupů, které se musí ošetřit a hlídat. Celý proces ošetření můžeme například zjednodušit využitím FrameWorku, který těmto incidentům dokáže předcházet. Další důležitým faktorem je hlídání věcí, souborů, které ukládáme do databáze nebo jiných úložišť.

Nejznámější důsledek úspěšného XSS útoku je tzv. sessions hijacking. Jedná se o odcizení sessions_id, klíče, které představuje unikátní ID uživatele pro přihlašování ke vzdáleným serverům. K tomuto ID se lze dostat několika způsoby, a nejúčinnějším je právě XSS útok.

Na internetu existuje řada serverů, které tento typ útoků vysvětlují. Dokážou nám pomoci pochopit, jak útok funguje, ale také nás můžou navést na to, jak útok provést. Nicméně je důležité, že zde můžeme také nalézt informace o tom, jak se proti útokům bránit.

Directory Traversal

Webový server slouží hlavně k poskytování souborů, které mohou být statické nebo dynamické. Pokud vytvoříme požadavek na webový server, pak nám server vrátí

statický obsah, pokud se skládá ze statických souborů. V případě, že web je složen z dynamických – nejprve soubor zpracuje a pak ho interpretuje. Při útoku typu „directory traversal“ někdy je označován také jako „path traversal“ využívá útočník špatného či žádného omezení přístupu ke zdrojovým souborům pomocí dynamického obsahu webového serveru.

SQL Injection

SQL Injection podobně jako XSS využívá nechráněných vstupů, které se snaží napadnout databázové vrstvy namísto aplikační. Pomocí těchto vstupů jsme schopni upravovat SQL dotazy, vkládat do nich podmínky nebo využívat vnořené dotazy.

Existuje několik způsobů, jak SQL Injection předcházet. Prvním z nich je kontrolování příchozích dat na aplikační vrstvě. Vývojář jistě ví, co mu na jaký parametr přijímá – pokud daný parametr chce jako výchozí hodnotu číslo, pak mu nic jiného nedáme. Dále můžeme využít funkci, která nám přepíše speciální znaky na entity – (v php se dá využívat *mysqli_real_escape_string*). Tato funkce nahrazuje znaky, které můžou SQL dotaz poškodit nebo upravit na text. V neposlední řadě musíme mít správně nastavená přístupová práva – pro připojení webových aplikací nevyužívat administrátorský účet, ale je nutné si vytvořit účet s omezenými právy. (Pokud by například útočník objevil chybu, neudělá nám nějaké větší škody – smaže tabulku, či celou DB)

Narušená autentifikace a správa relace

Většina moderních webových aplikací potřebuje pro správnou funkčnost jednoznačně identifikovat konkrétního uživatele, který ke stránce přistupuje. Aplikace pak upravuje obsah poskytovaný návštěvníkovy. Uživatelé většinou mají v aplikacích své uživatelské účty, pomocí kterých se přihlašují a následně si dle vlastních preferencí nastavit prostředí nebo chování samotné aplikace. Každý uživatel má po přihlášení svoje **unikátní** session ID, které když se dostane do rukou útočníka, může dělat v dané webové aplikaci různé kroky jménem nic netušícího uživatele.

Zabezpečení řešíme, až když se problém objeví

Žádná webová aplikace na světě není bezchybná a k nutnosti opravovat bezpečností chyby může dojít i za chodu aplikace. V těchto případech může být řešením nasazení rychlých bezpečnostních záplat nebo využití externího systému, který nezasahuje

do aplikace jako takové. První řešení se může zdát relativně jednoduché, ale není tomu tak. V mnohých případech ho nelze aplikovat dostatečně rychle, navíc by nasazení znamenalo vypnutí systému na delší časový úsek a mohlo by být spojeno s dalšími nutnými úpravami – dalšími náklady.

Naopak druhé řešení, které využívá externí systémy, se může být v určitých případech jednoduší. Externími systémy se myslí možnosti ochrany prostřednictví webového firewallu. Firewall může být ve formě samostatného softwaru, který běží jako aplikace na straně serveru, nebo zařízení, které je umístěno mezi serverem a klientem. Tyto systémy dokáží na základě předem definovaných pravidel analyzovat veškerý provoz, který server přijímá přes HTTP a jiné dostupné protokoly. V praxi to pak znamená, že pokud útočník pošle webové aplikaci XSS útok – server jej pomocí pravidel dokáže zachytit a nepustit takový dotaz na server. Webové firewally mají vlastní skriptovací prostředí a jejich výhodou je prakticky nulová závislost na stávajících architekturách aplikací.

Hrozby pro servery

S jistotou můžeme konstatovat, že bezpečnost serveru je stejně důležitá jako bezpečnost aplikace, která na serveru běží. Pokud nemáme zabezpečenou aplikaci, otevíráme tak potenciálním útočníkům dveře k celému serveru. Nejznámějšími metodami pro zabezpečení serverů jsou firewally a IDP/IPS což jsou systémy detekce a prevence průniků.

Firewall

Firewall je software, který nám jednoduše řečeno propustí nebo nepropustí síťovou komunikaci na zadaném portu. Pokud na serveru ale provozujeme webovou aplikaci, je logické, že některé porty musí být otevřené.

IDS/IPD

Oba tyto systému, které mají na starost detekci a prevenci útoků sledují a v případě IPS i reagují na reálný síťový provoz – pokud tedy mluvíme o otevřených portech. Když IPS zachytí útok typu DoS nebo port scan, dokáže na základě sady definovaných pravidel situaci vyřešit a ihned informovat administrátory, ale také provést blokaci IP, přesměrování provozu apod.

Tento typ ochrany však selže, když potenciální útočník odešle škodlivá data přímo přes aplikaci – dojde k využití neošetřeného vstupu. Pak dojde k selhání firewallu a IPS – pomoci nám tedy může webový firewall (WAF – Web Application Firewall). Většina serverů pro webové aplikace běží na operačním systému Linux. Pokud se tak stane, máme několik možností co dělat. První z nich je **chroot** kritické aplikace, která nám aplikaci uzavřeno do jejího kořenového adresáře. Díky tomu pak nemá přístup do jiných adresářů – do souborů. V případě napadení útočník nezíská přístup k celému serveru. Další možností je nastavení **limitu maximálního počtu běžících procesů**, který zabrání vyvolání procesu, které by se větvil do té doby, než by na serveru vyvolal jeho pád. V neposlední řadě stačí **bezpečně nastavit PHP**.

3.1.5 Shrnutí

V současnosti se webové aplikace těší velké popularitě, ačkoli nás obklopují již řadu let. Trendem webových aplikací, vzhledem k výkonům počítačů, je hojné užití mezipaměti v kombinaci s odesíláním požadavku pomocí AJAXu (viz výše). Není sice možné si bez internetu objednat zboží či lístky do divadla, ale pokud by zboží bylo načtené v mezipaměti, můžete si ho prohlédnout a přečíst si bližší informace o produktu.

Dalším trendem je orientace na mobilní zařízení s ohledem na omezení datových limitů. Proto se hojně využívá funkce zvaná „lazyload“, neboli načtení obsahu, až když je potřeba. Tohoto principu se využívá u datově náročných obsahů, ať už v textové nebo mediální podobě.

Budoucnost vývoje webových aplikací

Pojmem WEB 3.0 je obecně označována další generace webových služeb. V současnosti se nacházíme ve fázi WEBU 3.0, který je také označován jako „sémantický web“. Dle předpovědí bychom se už v roce 2020 měli dostat do fáze WEBU 4.0, který říká, že v této době bude s webovou aplikací kooperovat umělá inteligence. V současné fázi vývoje se lidstvo zaměřuje na relativně malé aplikace, které mohou běžet na libovolném zařízení. Jejich hlavní přednosti jsou sdílení a personalizace služeb – dále pak cloud computing.

Nyní se nacházíme v době, kdy už se rozšiřuje hranice mezi lidmi, kteří webové aplikace dokáží vytvářet, technologie jsou modernější, složitější a dříve nadšený amatér nedokáže nyní vytvořit moderní webové stránky. Začali se také používat mikroformáty

pro sémantický web a na webových stránkách či aplikacích došlo k nárůstu videoobsahu či využívání 3D technologií.

3.2 Funkcionální programování

Funkcionální programování se zdá být skrze složitý a bezvýznamný druh programování. V dnešním světě se setkáváme především s imperativními jazyky, mohlo by se studium funkcionálních jazyků zdát zbytečné. Funkcionální jazyky totiž poskytují vývojářům mnohem lepší možnosti abstrakce dat, ale dokonce umí abstrahovat celé algoritmy neboli chování.

Referenční transparence má ještě jeden zajímavý výsledek. Díky tomu, že výsledky funkcí můžou záviset jen na hodnotách jejích argumentů, protože nemůžou mít vedlejší účinky, můžeme velmi jednoduše výpočty paralelizovat. Musíme si uvědomit, že toto nemůžeme provést v imperativních jazycích, kde například přirazení do globální proměnné v jedné funkci by mohlo ovlivnit výpočet druhé funkce, pokud by s touto globální proměnnou také pracovala. Spíše u funkcionálních počítačů narážíme na opačný problém, neboť díky těmto vlastnostem je potenciálně příliš mnoho výpočtů, které je možné paralelizovat a vývojáři musí hledat takové rozklady programu, při kterých režie paralelních výpočtů nespotřebuje nebo dokonce nepřesáhne to, co paralelním výpočtem získáme.

S programy, které jsou napsané funkcionálně, se mnohem lépe pracuje, pokud je chceme nějakým způsobem transformovat, optimalizovat nebo například dokázat jejich správnost. (Hansen & Rischel, 2013)

3.2.1 Historie

Vznik tohoto druhu programování nemůžeme doložit přesně na den, ale jeho prvopočátky se datují na 30 léta minulého století. V těchto letech profesor matematiky Alonzo Church vytvořil **lambda kalkul** (viz. dále) jako matematickou teorii funkcí.

Jeden z prvních jazyků, který v sobě zahrnoval funkcionální část, byl LISP vytvořený Johnem McCarthym pro IBM. LISP představil spoustu vlastností, které můžeme, najít v nynějších funkcionálních jazycích. Jednalo se tedy o první skutečný programovací jazyk s implementovaným překladačem. Měl jednoduchou syntaxi a nebyl typovaný. Později se z tohoto jazyka vyvinul jeho dialekt Scheme.

Koncem 70. let ve městě Edinburgh vznikl jazyk ML s velmi silnou typovou disciplínou. Na jeho principech bylo navrženo několik dalších jazyků, jako jsou Hope, Clean nebo Miranda. O dvacet let později vznikl jazyk Haskell, který se používá dodnes.

λ – kalkul

Lambda kalkul analyzuje, vyšetřuje nebo studuje definice funkcí jako metody výpočtu. Můžeme jej vnímat jako nejjednodušší programovací jazyk. Další vlastností tohoto jazyka je také jeho univerzálnost, neboť můžeme libovolně rekurzivně spočetnou funkci vyjádřit a vyčíslit. Lambda kalkul je tedy výpočetní silou přímo úměrné Turingovu stroji.

O Lambda kalkulu můžeme mluvit jako o teorii funkcí založených na velmi jednoduchém jazyce, kde základními prvky jsou tři jednoduchý lambda výrazy – proměnná, aplikace a abstrakce. **Proměnná** nám představuje více nespecifikovanou hodnotu, dá se označovat například písmeny x, y, z. Dalším lambda výrazem je **Aplikace**, která reprezentuje volání funkce s jedním argumentem. Posledním výrazem je **Abstrakce**. Funkce v lambda kalkulu jsou reprezentovány ve formě Abstrakce. Tvoří je proměnná označující parametr a tělo ve tvaru lambda výrazu.

Vyhodnocování těchto výrazů se provádí na základe pravidel označených jako konverze, které jsou založené na pojmu substituce. Substituce vyjadřuje náhradu všech volných výskytů proměnné. Nejjednodušším typem konverze je **alfa konverze**, představující přejmenování proměnných. (Michaelson, 2011)

3.2.2 Princip

Funkcionální programování je založeno na funkcionálním přístupu, vycházejícího z deklarativního programování, které patří mezi programovací paradigmata. Deklarativní programování se vyznačuje tak, že autor aplikace, říká, co se má udělat, nikoliv však jak se to má udělat *(čímž se vyznačuje imperativní přístup, kde je přesná posloupnost příkazů – Java, C).*

Jazyky, které patří do skupiny funkcionálního přístupu k programu, vnímají program jako matematický výraz. Provádění programu pak spočívá v deterministickém vyhodnocování výrazů. Program popisuje, co chceme vypočíst, ale nepodstatné už je to, jak se to provede. Pokud si vzpomeneme na jazyk SQL, můžeme zde vidět jisté

podobnosti u dotazů (SELECT * FROM tabulka) – chceme vybrat data z tabulky, ale to jak se vyberou je nám v tuto chvíli jedno. Shrneme-li funkcionální jazyky, pak už víme, že důležité je to co se má dělat, a nikoli, jakým způsobem se to provede.

Strukturu celkově funkcionálního programu můžeme popsat jako výraz zachycující požadovaný cíl. Každý **term**[8] výrazu je vyjádřením určité charakteristiky řečeného problému. Pokud vyhodnotíme každý z těchto termů, dostaneme se k řešení. Výpočtem je pak posloupnost vzájemně ekvivalentních postupně zjednodušujících funkcí. Výsledkem je výraz v normální formě, tedy takový, který se dále nedá zjednodušit. Celý program je chápán jako jedna funkce, obsahující vstupní parametry mající jediný výstup. Tuto funkci můžeme ještě dělit na podfunkce.

Pokud se na funkcionální programování podíváme z hlediska použití v praxi, pak vidíme, že existuje rozdíl mezi matematickou funkcí a představou funkce použité v imperativním programování. Funkce, které jsou napsány imperativně, mohou mít vedlejší účinky, kterou mohou měnit stav samotného programu. Z toho důvodu jim chybí **referenční transparentnost**. Můžeme říci, že funkce je referenčně transparentní, pokud její výsledek je daný pouze jejími argumenty. Tento stav má spoustu implikací: volání samotné funkce můžeme nahradit jejím tělem nebo dokonce rovnou výsledkem a na celkovém chování programu se nic nezmění. O programu pak můžeme uvažovat jako o soustavě rovnic, funkce jsou funkcemi a nemůžou mít žádné vedlejší účinky a všechny hodnoty jsou neměnné. I to je hlavní myšlenkou funkčního programování.

Díky všem těmto skutečnostem jsou funkcionální programy jednoduší na pochopení, jsou kratší a stačí zde sledovat vstupy a výstupy a nemusím si pamatovat, jaké vedlejší účinky má daná funkce a jak to celkově ovlivní chod celého programu. Zmizí nám tak spousta starostí.

Čistě funkcionální programovací jazyky se používají spíše v akademickém, než komerčním prostředí, ale i přesto velké spektrum firem využívá některé z těchto jazyků – R (statistický program), Mathematica, Haskell a dále pak doménově specifické programovací jazyky – XQuery/XSLT (XML). Další hlavní roli mají tyto jazyky v dalších odvětvích

[8] **Term** – jedná se o terminální podvýraz, který je v rámci zápisu dále nedělitelná součást – můžeme si ho představit jako volání požadované funkce.

informatiky, jako jsou umělé inteligence, formální specifikace, modelování nebo rychlé prototypování.

Kvůli pravidlům a dalším vlastnostem můžou být funkcionální jazyky „líné“, snadno napadnutelné, ale také užitečné pro **paralelní programování**. Pokud je hodnota neměnná, pak se nedá změnit, nemusím jí hlídat a čekat, až ji někdo někde změní – můžu si tuto hodnotu jednoduše sdílet mezi libovolným počtem vláken (což je dobré pro **CPU Cache**). Tohle je jeden z dalších důvodů, proč funkcionální programování získává v posledních letech opět na oblibě.

Funkcionálně se dá programovat v každém programovacím jazyku, ať už je to PHP, Java, C. V některých je to lehčí, ale můžeme tak programovat ve všech. Stačí psát kód, který se opírá o základní myšlenku referenční transparentnosti.

3.2.3 Výhody

Mnozí lidé považují funkcionální programování za složitost a tak to raději vzdají hned na počátku snažení. Najdou se mezi nimi ale tací, kteří mají rádi výzvy a baví je se učit novým věcem. Musí se naučit číst také odbornou literaturu, protože články na internetu už nejsou dostačující. Programovat funkcionálně můžou i naprostí laici, kteří mají alespoň trošku logické myšlení a pochopí, že programování je vlastně tak trochu matematika.

Pokud dojde k pochopení funkcionálního programování, zjistíme, že mnohé se dá naprogramovat jinak a většinou i jednodušeji. LISP a jeho dialekty Scheme nebo Clojure jsou plnohodnotné jazyky, kterými lze napsat téměř cokoliv.

Někteří vývojáři si chválí funkcionální jazyky z důvodu délky a přehlednosti kódu. Lze totiž psát krátký kód a díky tomu získává na přehlednosti.

Zde je několik výhod, které můžou potencionální vývojáře nadchnout:

- Funkcionální jazyky nám přináší větší míru abstrakce, na rozdíl od imperativních jazyků a snaží se o rozdělování do komponent. (Pokud opakujeme nějaký postup, lze ho často abstrahovat pomocí funkcí výších řádů). Rozdělováním do komponent vzniká elegantnější a srozumitelnější kód.

- Délka a srozumitelnost programů je na mnohem vyšší úrovni, než je to u imperativních jazyků.
- Silné typové systémy umožnují větší kontrolu kódu při překladu.
- Data, která se nemění, umožňují překladačům lepší optimalizaci výsledného kódu.
- Některé algoritmy mohou být značně urychleny pomocí tzv. líného vyhodnocování.

3.2.4 Nevýhody

- Počítače jsou navrženy a optimalizovány na imperativní programy.
- Některé imperativní algoritmy jsou moc složité na to, aby byly jednoduše zapsatelné funkcionálně.
- Neměnnost datových struktur, nebo spíše jejich častější konstrukce klade vyšší nároky a zátěž.
- Výsledky kód je často méně efektivní. Jedné se obvykle u jazyky, které používají tzv. líné vyhodnocování. S tím pak souvisí složitější překlad a nutnost specializovaných operací.

Shrnutí

Funkcionální programování je velmi důležité pro některé oblasti informatiky, jako je umělá inteligence, formální specifikace a modelování nebo rychle prototypování. Hodně myšlenek dnes tvoří teoretický základ našeho oboru, vzniklo jako důsledek experimentů s funkcemi a jejich vlastnostmi. Tento přístup světu ukázal metody, které popisují sémantiku programovacích jazyků, případně umožnily vznik základních pouček teorie složitosti algoritmů a vyčíslitelnosti.

Nesmíme také zapomínat na jednu vlastnost, která funkcionální jazyky přivedla až před studenty do lavic některých univerzit, jako první jazyk, se kterým se po dobu studia na školách studenti teoreticky seznámí. Z pohledu studenta informatiky je funkcionální programování zajímavá činnost, která pomáhá rozvíjet myšlení. U některých studentů se může stát, že změní jejich pohled na programování. Velké množství algoritmů má ve funkcionálních jazycích velmi elegantní vyjádření, které se neruší

mnoha implementačními detaily a přesněji vystihuje podstatu algoritmu. Díky tomu dokážeme hlouběji proniknout do podstaty některých algoritmů a dokážeme ocenit jejich eleganci a matematické čistoty.

Kdybychom se měli podívat na otázku funkcionální budoucnosti, tak nedokážeme jednoznačně odpovědět. Dlouhý rodokmen funkcionálního programování a atraktivita jeho hlubších principů naznačují, že jeho vyhynutí je velmi nepravděpodobné. I když budou zmíněné konkrétní implementace nahrazeny, zdá se, že roste pravděpodobnost toho, že budoucnost moderního vývoje softwaru bude funkcionální.

3.3 Objektové programování

Znalost objektově orientovaného programování (dále jen OOP) a um pochopit, jak vlastně funguje je důležité pro vývoj webových aplikací. OOP přináší jisté ulehčení pro programátory a vývojáře aplikací, protože dokáže tvořit architekturu aplikací, oddělovat výstupy HTML od logických částí aplikace, a zároveň používat komponenty a knihovny třetích stran. OOP přináší výhodu v efektivnosti práce. Programování aplikací bez objektů je pracnější, složitější a méně kvalitní.

3.3.1 Historie

Historie objektově orientovaného programování sahala až do šedesátých let 20. století, kdy prvním programovacím jazykem s třídami a objekty byla **Simula 67**, za jejímž vznikem stáli vědci z Norska. Tento jazyk byl původně určen k simulaci interakcí několika lodí.

Simula dále inspirovala několik vědců z firmy Xerox, kteří následně vytvořili jazyk vlastní – SmallTalk a začali jako první používat termín „OOP – objektově orientované programování“. Simula 67 také silně ovlivnila i návrháře jazyka C++.

3.3.2 Princip

Objekt

Objekty jsou všude tam, kam se podíváme, setkáváme se s nimi každý den. Pokud navrhujeme programový systém, pak se snažíme reálné objekty modelovat a reprezentovat pomocí objektů abstraktních. Tyto abstraktní objekty odrážejí část vlastností a chování reálných objektů. Při tomto procesu dochází ke zjednodušení – (*pokud budeme vytvářet*

informační systém na prodej zboží přes internet, budou nás u objednávky zajímat relevantní informace o zákazníkovi, jako jsou adresa, email, jméno – nikoliv však barva vlasů nebo váha).

Na objekt se můžeme dívat jako na diskrétní entitu s přesně daným rozhraním, které uchovává stav a chování. Stav objektu je určen hodnotami atributů v určitém okamžiku. To jak se bude objekt chovat je dáno operacemi, které s ním můžeme dělat. Mezi další společné vlastnosti objektů patří identita, která představuje jedinečnou existenci objektu v čase a prostoru – tím se odlišuje od jiných objektů. Na základě klíčových atributů definujících objekt, pak můžeme rozhodnout o jejich identitě.

Pojmem **zapouzdření** ukrýváme vnitřní strukturu objektu a implementaci jeho operací za veřejným rozhraním. Obvykle spočívá v práci s atribute, se kterými nemůžeme manipulovat přímo, ale jen díky operacím to lze, díky operacím, které jsou součástí veřejného rozhraní. Také nám zajištění zajišťuje plnou kontrolu nad objektem, uživatel objektu s ním může manipulovat, jen prostřednictvím veřejného rozhraní.

Třída

Každý objekt je instancí určité třídy, která definuje množinu vlastností – atributů nebo operací společných pro všechny své instance. Třída je tedy množina objektů s určitými vlastnostmi. Samotná třída přitom nedefinuje konkrétní objekty té třídy, jen udává, jaké vlastnosti bude mít každý objekt té třídy.

Instance

Jedná se o konkrétní objekt určité třídy. Instance pak atributu přiřadí konkrétní hodnotu. O instanci můžeme také říci, že se jedná o odkaz na třídu nebo o volání jiné třídy. Instance a objekty jsou svým způsobem synonyma.

Konstruktory

Konstruktory nazýváme speciální metody volané při vytváření nových instancí dané třídy. V konstruktoru se obvykle naplní proměnné objektu. Konstruktory nemají návratový typ, mohou mít parametry a také mohou volat konstruktor rodičovské třídy.

Vztahy mezi objekty a třídami

K vytvoření objektově orientované aplikace je nutné, aby spolu objekty komunikovali. Tato komunikace představuje volání metod mezi objekty, které jsou spolu svázány pomocí **spojení**. V programovacích jazycích jsou spojení nejčastěji implementována jako ukazatele nebo objektové reference. Mezi dvěma objekty můžeme vytvářet spojení jednostranná nebo oboustranná, kde oba objekty mají na sebe navzájem k dispozici odkaz. V jednostranném spojení má odkaz na druhý objekt pouze jeden z dvojice objektů – ten pak druhý objekt řídí. V OOP návrzích se jednosměrné spojení zobrazuje šipkou, která vede od objektu obsahujícího odkaz směrem k řízenému objektu. Spojení oboustranná se pak zobrazují spojovací čárou bez šipek. V každém spojení mohou objekty zastávat nějaké role. *(Ve spojení dvou osob může jedna role být zaměstnavatel, druhá zaměstnanec)*.

Spojení mezi objekty nelze vytvářet libovolně, musí být v souladu s definicí vztahů mezi třídami. Na úrovni tříd spojením odpovídají **asociace** – spojení jsou zde instancemi asociací. Každá z nich je pak popsána podrobnějšími vlastnostmi, jako jsou například název, role, násobnost a řiditelnost.

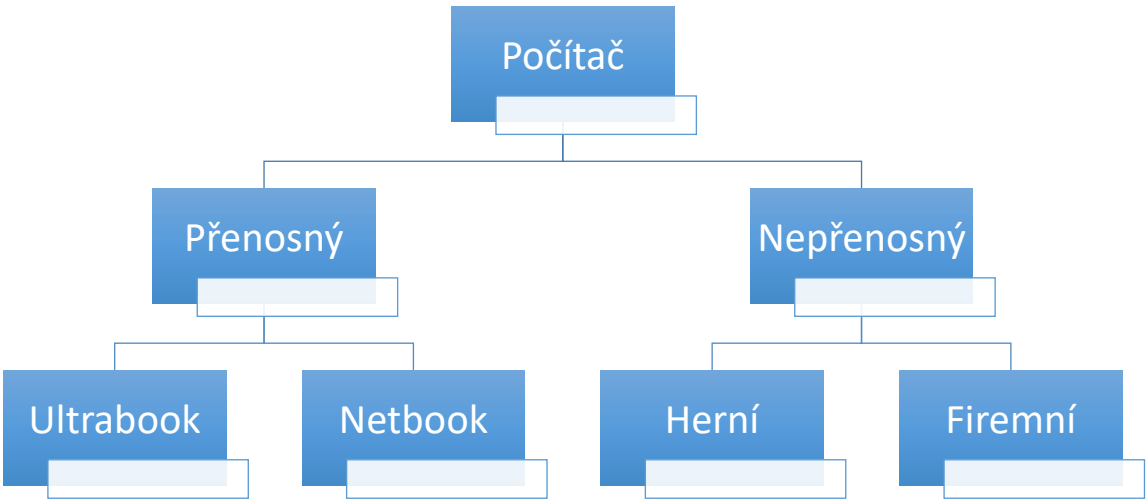
Agregace představuje volnou vazbu mezi součástí a celkem, kdy jeden objekt (celek) využívá služby ostatních objektů (součástí). *(Vztah mezi počítačem a tiskárnou je agregace)*. Jedná se o formu asociace a v grafických podobách návrhu se vyznačuje prázdným kosočtvercem na straně celku. Silnější formou agregace je **kompozice** – jedná se o stejný typ vztahu, ale na rozdíl od agregace, tento vztah je velmi těsný a neumožňuje samostatnou existenci součástí, aniž by byla připojena k nějakému celku. Dalším rozdílem je, že na rozdíl od agregace tato součást musí patřit jen jedinému celku a není možné ji sdílet více celky.

Dědičnost a polymorfismus

Mezi základní vlastnosti objektově orientovaného programování se řadí dědičnost. Ta slouží k vytvoření nových datových struktur na základě předešlých. Představuje možnost vytvoření hierarchie tříd. Tato vlastnost měla zabraňovat opakování stejného kódu – stejně tak jako zabraňovat opakování kódu, který je pro určité prvky společný.

Obvykle využijeme více tříd pro dostatečnou kvalifikaci objektů a běžnou praxí je vytvoření tzv. rodiče – od kterého jsou ostatní třídy odvozené tzv. potomci. Potomci pak tedy obsahují vlastnosti jak rodiče, tak nové, které už jsou specifické pro novou třídu – a tím se stává konkrétnější a specializovanější než její rodič.

Dědění nám tak umožňuje opakovaně využívat části kódu rodiče u určité množiny instancí se stejnými specifickými vlastnostmi. Díky vlastnosti dědění se nám program stává přehlednější, méně rozsáhlý a zároveň se vyhýbáme chybám, které bychom mohli vytvořit. V neposlední řadě má programátor o mnoho méně práce.



Graf 1 - Demonstrace dědičnosti

Z diagramu výše je patrná hierarchie dědění. Nejobecnější uzel diagramu je zároveň jeho kořenem, který nedědí žádné vlastnosti od jiné třídy. Čím dále je některý z uzlů od diagramu tím více vlastností zdědí od předchozích vlastností – od nadřazených prvků.

Polymorfizmus je v každodenním životě velice běžný a ne každý ho umí vnímat. Polymorfizmem se dá představit situace, kde určitý podmět vyvolává odlišné reakce. V objektovém programování to tedy znamená, že určité metody mají u různých objektů jiný význam. Jedná se o zápis více definicí jedné metody. Podstatou jsou metody se stejnou hlavičkou, které mají definované třídy potomků. Polymorfizmus se úzce vztahuje k dědičnosti a k implementování rozhraní. Najde také ale uplatnění při vytváření objektů – konstruktorů.

Zapouzdření

Mluvíme o dalším paradigmatu objektově orientovaného programování, které se využívá při skrývání implementací objektu před jinými objekty. Jeho důsledkem je jistý druh autorizovaného přístupu k datům. Tento přístup zajišťuje, aby kód fungoval se správnými daty a aby byly prováděny operace, které jsou k tomu určené. Zapouzdření také skrývá stav objektu a k jejich zpřístupnění dochází pomocí metod. Pokud máme robustnější a složitější kód je více pravděpodobné že nedopatřením budeme chtít ovlivňovat chod částí programu, tam kde to není vyžádáno. Data se budeme snažit zapouzdřit, nebo ochránit před nepovoleným přístupem deklarací proměnných, jejichž součástí jsou i přístupová práva k vyjádřená pomocí specifikátorů.

SPECIFIKÁTOR	V TÉŽE TŘÍDĚ	V JINÉ TŘÍDĚ	V PODTŘÍDĚ STEJNÉHO BALÍKU	V POTŘÍDĚ JINÉHO BALÍKU	V JINÉ TŘÍDĚ JINÉHO BALÍKU
PRIVATE	X				
NEUVEDENO	X	X	X		
PROTECTED	X	X	X	X	
PUBLIC	X	X	X	X	X

Tabulka 1 - Specifikátory přístupů

Nejvíce omezujícím specifikátorem je **private**. Když je metoda, nebo proměnná uvozena tímto specifikátorem – lze ji vidět pouze ve třídě, ve které je nadefinovaná (ve které vznikla). Pokud tedy chceme přistupovat z vnějšku, není to možné.

Mezi další způsob autorizace přístupu je **neuvedení** žádného – neuvést specifikátor při definování metody znamená, že metoda nebo proměnná jsou viditelné pro všechny třídy ve stejném balíku. Tyto metody nebo proměnné zároveň nejsou viditelné z odvozené třídy.

Dalším specifikátorem je **protected**, který zajišťuje, že metoda nebo proměnná jsou viditelné z balíku třídy a také z jejich potomků. Potomek třídy nemusí být ve stejném balíku, jako je rodičovská třída.

Poslední specifikátorem je **public** – tento specifikátor říká, že metody nebo proměnné jím uvozené jsou volně přístupné komukoliv. Běžně se toto přístupové právo používá pro metody.

3.3.3 Výhody

Mezi hlavní výhody objektového programování se řadí jednoduší a rychlejší vývoj aplikací. Při dokončení aplikace je menší chybovost na výslednou funkčnost a jejich provoz je snazší na údržbu.

Snadná údržba je jeden ze stěžejních nároků na vývoj aplikace. Dále se velký důraz klade na rozšiřitelnost aplikace na úkor výkonové optimalizace.

- Snadná lokalizace a oprava chyb
- Oddělení detailů od celku
- Rozšiřitelnost aplikace
- Sémantická bohatost
- Snazší definice celého systému

3.3.4 Nevýhody

Mezi nevýhody objektového programování se řadí správa objektů, která zabere určité systémové prostředky, takže i dokonale napsaný objektový program bude pomalejší, než dokonale napsaný procedurální program. Zastánci objektového programování ale argumentují, že praktický žádná aplikace nebude nikdy vytvořena dokonale, přičemž v OOP se díky vlastnostem, které nabízí, dostane dokonalosti lépe, čímž se tato výkonová nevýhoda eliminuje.

- Objektově orientovaný kód není efektivní
- Nová metodika si vyžádá zaškolení pracovníků

Shrnutí

Pramenů a zdrojů, ze kterých lze čerpat znalosti o objektovém programování je opravdu mnoho. Některé pojednávají o objektovém programování komplexně, jiné se

zaměřují přímo na určité části a dokonce některé jsou srozumitelné i pro začátečníky. Nutno také dodat, že každý vývojář a programátor má na objektové programování svůj názor – někomu vyhovuje více, někomu méně.

Naučit se programovat objektově znamená naučit se objektově myslet. To znamená, že programátor si nejprve musí svůj systém představit a „namalovat“ a pak až jej tvořit.

3.4 Shrnutí

V současné době a v období pár měsíců zpět, se stále více a více mluví o nástupu funkcionálního programovaní a jeho nadřazenosti nad objektovým. Tyto spekulace se začínají projevovat nejen u mladých a začínajících programátorů, ale také u seniorních inženýrů.

Funkcionální programování a objektové programování nejsou soupeřící koncepty programování, které mezi sebou soupeří, nýbrž dva projekty z větší části úplně mimoběžné. Objektově orientované programování je programovací paradigma, které je založené na konceptu „objektů“, které představují datové struktury obsahující data ve formě polí – často označované jako atributy. Další součástí je i kód ve formě procedur – známý jako metody. Naopak funkcionální programování je programovací paradigma, kde styl budování struktury a prvky počítačových programů řeší výpočet jako vyhodnocení matematických funkcí.

Obě programovací paradigmata mají stejný cíl, a ten je vytvářet srozumitelné a flexibilní programy bez chyb. Existují však dva různé způsoby, jak nejlépe tyto programy vytvořit. Ve všech programech existují dvě primární komponenty: data (věc, kterou program ví) a chování (věci, které program může s těmito daty dělat). OOP říká, že sdružování dat souvisejícího chování na jednom místě – objekt – usnadňuje pochopení toho, jak program funguje. Funkcionální paradigma říká, že údaje a chování jsou výrazně odlišné a měly by být pro jasnost odděleny.

Je samozřejmé, že vývojáři používají jazyk, který je činí nejproduktivnější. Všichni zkušení vývojáři mají také historii, mnoho zkušeností, které je činí efektivnějšími a proto programují v tom, co je potřeba a co je hlavně baví. Nejde jednoznačně říct, zda je lepší funkcionální přístup nebo objektový. Na místě je

mnoho faktorů, díky kterým si každý vybere tu správnou variantu, která se pro danou aplikaci hodí nejvíce.

Objektově orientované programování je jen tenká vrstva nad přístupem k různým typům databází, které podporují SQL. Hlavním přínosem relační databáze je možnost zpracovat budoucí požadavky. Když se na databázi podíváme komplexně, zjistíme, že se jedná o velkou datovou strukturu a aplikace jsou jen svazky operací, které na ni působí. Srdcem každé webové aplikace je velká funkční databáze, datová struktura s operacemi, které na ni působí. Dobře vytvořená OO architektura mění způsob, jakým se věci dají snadno dohromady. Snadno se dají měnit i vztahy mezi věcmi, tabulkami či objekty.

Objektově orientované jazyky jsou dobré, když existuje pevná sada operací na věcech a také na faktu, jak se náš kód vyvíjí – primárně přidáváme nové věci. Toho můžeme dosáhnout přidáním nových tříd, které implementují naše stávající metody a stávající třídy zůstávají osamocené. Funkční jazyky jsou také dobré, pokud máme pevnou sadu a přidáváme nové operace k již existujícím věcem. Funkční programy jsou vynikající při manipulaci se symbolickými daty ve stromové podobě. Oblíbeným příkladem jsou například kompilátory, kde se zdrojové a přechodové jazyky mění jen zřídka, ale tvůrci kompilátorů vždy přidávají nové překlady k vylepšení kódu nebo jeho optimalizaci. Kompilace a překlad jsou obecně zabijáky pro funkcionální jazyky. Výsledkem však může být věta, že „Dobrý software je napsán v obou programovacích stylech, protože dobrý software má více než jednu potřebu uspokojit“.

Jak to bude dál? Funkcionálně nebo programovat objektově? Pravděpodobně obojí. V následujících letech se masově rozšíří objektově funkcionální paradigma, které spojuje to nejlepší z obou světů. OOP pro organizaci a modularizace programů a FP kvůli referenční transparentnosti, neměnným typům a snazšímu paralelnímu programování.

4 Vlastní práce

4.1 Funkcionální vs. Objektové programování

První částí vlastní práce je vytvoření praktických příkladů funkcionálního a objektového programování webových aplikací. Objektově orientovaný design se celkově posunul tím správným směrem, ale vývojáři touží po dalších technologiích. Funkcionální programování se rozšiřuje na poli vývojářských jazyků, například Twitter je napsán ve Scala nebo WhatsApp pomocí Erlangu. Většina rozšířených programovacích jazyků buduje funkcionální přístup do svých funkcí, například Java, C#, Python, Ruby, Perl a další. Psaní funkcionálního kódu otevírá dveře k čistému a rozšiřitelnému řešení. Programovací jazyk, kterému se v této části budeme věnovat je **PHP**. Ve spojení s jeho nejznámějšími frameworky je světovou jedničkou při vývoji webových aplikací.

4.1.1 Funkcionální programování

Rozvoj PHP za poslední desetiletí byl velice rychlý. Mezi léty 2009 a 2014 došlo rozvoji PHP 5 a ještě předtím než bylo vypuštěno PHP 6, došla verze PHP 5.3 k mnohačetným změnám. Jednou z nich bylo připojení OOP + FP (funkcionální programování). Jednalo se o období, kdy byla do kódu přidána možnost využívání „closures“. V této kapitole se podíváme na základy PHP programování pomocí funkcionálního přístupu.

Před představením několika příkladů funkcionálního programování je nutné vědět, jak se dají hledat chyby v zdrojovém kódu. Nejčastější možností je zjištění hodnoty proměnné. Existuje několik možností, ale nejčastěji používanými jsou příkazy **print_r** a **var_dump**. Do těchto předdefinovaných funkcí se do závorek umístí řešená proměnná, ve které se chyba vyskytla. Pro názornou ukázku si vytvoříme příklad pole, kde jsou definované jeho hodnoty. Máme zde využito několik datových typů. Závěrem pak vložíme název pole jako proměnnou do výše zmíněných funkcí, a podíváme se na výsledek, který nám PHP vrátí.

```php
<?php

    define('MY_CONSTANT', 'apple');
    $my_function = function ($data) {
        return $data;
    };

    $my_array = [1, 2, 'cherry', MY_CONSTANT, $my_function];

    echo "print_r output :\n\n";
    print_r($my_array);

    echo "\n\n var_dump output :\n\n";
    var_dump($my_array);
```

Obrázek 4 - Využití příkazu print_r a var_dump

Výsledek tohoto zdrojového kódu bude mít u každého příkladu jiné výsledky. Ze začátku je zřejmé, že se jedná o podobný výsledek, ale není tomu tak. Výsledek **print_r** je lépe formátován, takže vývojář může výsledek snadněji vyčíst. Na rozdíl od příkazu **var_dump**, ale dostaneme mnohem méně informací.

```
print_r output :
    Array (
        [0] => 1
        [1] => 2
        [2] => cherry
        [3] => apple
        [4] => Closure Object
        (
            [parameter] => Array (
                [$data] => <required>
        )
    )
)
var_dump output :
    array(5) {
    [0]=>
    int(1)
    [1]=>
    int(2)
    [2]=>
    string(5) "cherry"
    [3]=>
    string(6) "apple"
    [4]=>
    object(Closure)#1 (1) {
        ["parameter"]=>
    array(1) {
        ["$data"]=>
    string(10) "<required>"
    }
} }
```

Obrázek 5 - Výsledek testování proměnných

V jazyce PHP je možné využívat funkci **„array_map**“, která využívá jako argument odkaz na funkci a také na předem vytvořené pole. Funkce následně pole projde a provede danou funkci pro každý prvek v tomto poli. Zde je jeden příklad toho jak vytvořit v PHP funkcionální řešení procházení pole pomocí této funkce .

```php
<?php
    function make_human_readable($item) {
        return preg_replace('/_/', ' ', $item);
    }
    $columns = ['card_type', 'full_name', 'transaction_date', 'transaction_amount', 'transaction_status'];

    print_r($columns);

    $human_names = array_map("make_human_readable", $columns);

    print_r($human_names);
?>
```

Obrázek 6 - Ukázka funkcionálního programování

Pokud je zdrojový kód napsán správně, ukazuje výsledná webová stránka tento výsledek:

```
Array (
    [0] => card_type
    [1] => full_name
    [2] => transaction_date
    [3] => transaction_amount
    [4] => transaction_status
)
Array (
    [0] => card type
    [1] => full name
    [2] => transaction date
    [3] => transaction amount
    [4] => transaction status
)
```

Obrázek 7 - Výsledek funkce "array_map"

Funkce „array_map()“ má i další výhody. Je neměnná, což znamená, že se nezmění obsah původního pole, které je předáno do jiné funkce. Neměnné proměnné přináší do programování i další výhody. Jednou z hlavních příčin chyb ve webových aplikacích nebo softwarech je chyba odkazu. Neměnné objekty jsou přenášeny, ale obsah zůstane stejný. Neměnné datové struktury jsou také důležité ve sdílení paměťových vícevláknových aplikacích. Bohužel na rozdíl například od skriptovacího jazyka Scala nebo F# podporuje PHP velmi nepatrnou podporu pro neměnné proměnné.

Dalším příkladem funkcionálního programování je využití již známé **rekurze**. Toto programovací paradigma nepodporuje žádné cykly typu **foreach**, **for**, **while** či **do while**. Všechny tyto cykly jsou nahrazeny konceptem rekurze.

Pokud v PHP chceme naprogramovat funkci, která najde sumu všech prvků v poli, můžeme využít rekurzi a zapomenout tak na již předdefinovanou funkci „**array_sum**“, která toto pole projde sama. Tento příklad bychom řešili způsobem s využitím rekurze.

```php
<?php
    function sum($array) {
        if (empty($array)) {
            return 0;
        } else {
            return $array[0] + sum(array_slice($array, 1));
        }
    }

    $total = sum([1, 2, 3]); // Výsledkem bude 6
?>
```

Obrázek 8 - Funkce na součet prvků v poli

Prázdné pole vrátí hodnotu 0, což je základní stav tohoto pole. S více hodnotami pole vrátí výsledky prvního prvku s rekurzivním součtem všech ostatních prvků.

Neméně důležitou je funkce tzv. vyššího řádu „**array_each**“. Tato funkce vytvoří abstraktní smyčku a umožní provádět akce, u kterých není požadována kolekce či jiná návratová hodnota. Od funkce „**array_map**“ se liší návratovou hodnotou, nevrací žádnou.

```php
<?php
    function array_each(array $items, $func){
        foreach ($items as $key => $item) {
            $func($item, $key);
        }
    }

    $users = [
        [
            "id" => "1",
            "username" => "John",
            "email" => "john@domain.tld"
        ], [
            "id" => "2",
            "username" => "Jane",
            "email" => "jane@domain.tld"
        ]
    ];

    array_each($users, function($user){
        emailTo($user['email'], 'Welcome to my blog');
    });
```

Obrázek 9 - Funkce "array_each"

U tohoto příkladu můžeme jasně vidět, že tato funkce neposkytuje žádnou návratovou hodnotu, protože se opakuje. Jejím účelem je sbírat data a dělat něco s každou položkou, kterou najde.

Tento příklad by se dal také řešit pomoc funkce „array_map“, kterou již známe. Ta by nám všem vrátila hodnotu.

Funkcionálně můžeme také psát funkce vyššího řádu, které se nejčastěji používají při implementaci strategií.

```php
<?php
    $input = [1, 2, 3, 4, 5, 6];

    $filter_even = function($item) {
        return ($item % 2) == 0;
    };

    $output = array_filter($input, $filter_even);

    print_r( implode(', ', $output) ); // 2, 4, 6
```

Obrázek 10 - Funkce implementace strategií

Nyní je na čase, abychom si vysvětlili, co jsou to tzv. „closures“ – ve volném překladu mluvíme o uzávěrech. Je důležité si uvědomit, jak moc jsou důležité a jakou roli hrají v implementaci funkcionálního programování.

Closure je anonymní funkce, která nám umožňuje přistupovat k proměnným importovaných z vnějšího rozsahu bez použití globálních proměnných. V podstatě se jedná o funkci s pevně daným argumentem, který je uzavřený v prostředí, kde byl nadefinován.

V následujícím příkladu se podíváme na closures, které jsou použité k definování funkce vyššího řádu.

```php
<?php
    $greater_than = function($min) {
        return function($item) use ($min) {
            return $item > $min;
        };
    };

    $input = [1, 2, 3, 4, 5, 6];
    $output = array_filter($input, $greater_than(3));

    print_r( implode(', ', $output) ); // 4, 5, 6
```

Obrázek 11 - Využití "closures" ve funkci pro filtrování

Do této funkce zapisujeme hodnotu minima, pro které chceme zjišťovat větší hodnoty. Představme si šablonu či vstupní ověřovací knihovnu, kde je definováno uzavření pro zachycení proměnných v rozsahu a přístup k nim je povolen později při vyhodnocování anonymních funkcí.

Dalším příkladem funkcionální programování je čtení ze souboru.

```php
<?php
    function readLines( $filename )
    {
        $f = fopen( $filename, "r" );

        return function() use ($f)
        {
            $l = fgets( $f );
            if ( $l == null )
            {
                fclose( $f );
            }
            return $l;
        };
    };

    $line = readLines( "robots.txt~" );
    while ( ( $l = $line() ) != null )
    {
        echo $l;
    };
```

Obrázek 12 - Funkce pro čtení ze souboru

V části objektového programování si pak ukážeme, jak by se tento příklad dal napsat v OOP, ale už teď prozradím, že tato funkce má mnohem více řádku v této variantě paradigmatu.

V této části kódu si všimněme, že jsme si vytvořili funkci, která nám vrací funkci. Tato funkce otevře soubor uložený na místě, které nám říká proměnná **$f**. Řádek číslo šest je pro náš důležitý. Zde můžeme vidět, vytvoření „closure“, o kterých jsme si již povídali.

Posledním příkladem je proces tzv. „curryfikace“ a částečně aplikované funkce, které jsou dalšími vlastnostmi funkcionálního paradigmatu. Pokud hovoříme o částečně aplikované funkci, myslíme tím funkci, která obsahuje menší počet zadaných argumentů, než požaduje. Výsledkem je funkce, která čeká, dokud na ní nebude aplikován zbytek požadovaných argumentů.

Naopak proces „curryfikace“ umožňuje funkci, aby měla schopnost stát se částečně aplikovanou.

```php
<?php
$volume = function ($length = 0, $width = 0, $height = 0) use (&$volume) {
    $args = func_get_args();
    $numArgs = func_num_args();
    if ($numArgs == 3) {
        return $length * $width * $height;
    }
    else if ($numArgs < 3) {
        return function() use(&$volume, $args) {
            $newArgs = array_merge($args, func_get_args());
            return call_user_func_array($volume, $newArgs);
        };
    }
    else {
        throw new BadFunctionCallException("Too many arguments");
    }
};


$standardVolume = $volume(10);

$vol = $standardVolume(5, 5);
```

Obrázek 13 - Částečně aplikovaná funkce

Funkcionálně se dá programovat v mnoha programovacích jazycích, každému vyhovuje více to, či ono.

4.1.2 Objektové programování

V teoretické části práce byl vysvětlen princip fungování objektově orientovaného programování, včetně jeho výhod a nevýhod. Součástí této kapitoly jsou praktické ukázky s vysvětlením. Podíváme se na konkrétní příklady využití objektově orientovaného programování a na jeho přednosti.

Objektově orientované programování je velmi populární a oblíbené také hlavně z důvodu přehlednosti kódu. Vývojáři jsou téměř nuceni psát strukturovaný kód a jsou nuceni rozdělovat některé části webových aplikací do více zdrojových souborů. Pro přehlednost rozdělení tříd je nejlepším řešení k hlavnímu souboru *index.php* vytvořit ještě soubor *class_lib.php*, který slouží k zdrojovým kódům všech tříd. Soubor je samozřejmě možné pojmenovat jinak, ale obvykle se využívá tento název. Jeho připojení do hlavního souboru bude pomocí PHP „includes“.

Třídy a objekty

Namísto používání několika funkcí, proměnných a kódu motajícího se kolem těchto dvou věcí, můžeme skripty navrhovat objektově a vytvářet si vlastní knihovny. V tomto příkladu si ukážeme, jak se v objektově orientovaném programování vytvářejí třídy obsahující vlastnosti. Vlastností se pak rozumí proměnná vytvořená uvnitř třídy.

```php
<?php

    class Car {
      public $comp;
      public $color = 'red';
      public $hasSunRoof = true;
    }
```

Obrázek 14 - Vytvoření třídy s vlastnostmi

Třídy jsou šablony PHP objektů a jsou důležitější více než kdy předtím. Jedním ze zásadních rozdílů mezi funkcí a třídou je jejich obsah. Třídy obsahují data (proměnné) a funkce a tvoří tak společně balíček nazvaný objekt.

```php
<?php

    $skoda = new Car ();
    $audi = new Car ();
```

Obrázek 15 - Vytvoření objektů

Pro práci s objekty můžeme v PHP využít volání vlastností pomocí příkazu **echo**. Na obrázku níže můžeme vidět zavolání objektu a vypsání vlastnosti třídy. Výsledkem bude pak barva auta značky Škoda a Audi.

```php
<?php

    echo $skoda -> color;
    echo $audi -> color;
```

Obrázek 16 - Použití vlastnosti třídy na objekt

Tímto způsobem lze vlastnosti třídy měnit a definovat tak novou hodnotu proměnné.

```php
<?php

    $skoda -> color = 'blue';
```

Obrázek 17 - Změna vlastnosti objektu

Metody

Z velké části se v objektově orientovaném programování zobrazují funkce uvnitř tříd. Takovým funkcím se říká metody. Na obrázku níže jsme vytvořili metodu „hello()“, která má prefix „public“ neboli veřejný.

```php
<?php

    class Car {

        public $comp;
        public $color = 'beige';
        public $hasSunRoof = true;

        public function hello()
        {
          return "beep";
        }
    }
```

Obrázek 18 - Metoda uvnitř funkce

Uvnitř objektu lze vytvořit také zajímavé funkce nebo metody, které jsou pojmenovány *set_* nebo *get_*. Za podtržítkem se doplní název proměnné ve třídě, která bude modifikována nebo volána. V programování se běžně využívají názvy proměnných v anglickém jazyce a z tohoto překladu lze poznat, že chceme zadat hodnotu nebo zavolat hodnotu. Například název může vypadat takto: *(set_name(), get_name())*. Jedná se o zápis, který je využíván ve více programovacích jazycích například dále ještě v Ruby nebo Javě. Tento druh zápisu je využíván celosvětově, a pokud dojde k vývoji aplikace novým programátorem, který uvidí tento zápis, pak mu v případě *(set_name)* bude jasné, že v této třídě existuje i proměnná „name“.

Využití $this

Tento operátor je nativně zabudován v každé vytvořené třídě a je s touto třídou také spojen. Jedná se o speciální druh proměnné, která odkazuje sama na sebe, neboli na třídu ve které je součástí. Zmínit bychom měli také operandy public, protected a private. Zatímco modifikátor přístupu **public** umožňuje pracovat s kódem metod zvenčí nebo uvnitř třídy, **private** modifikátor tomuto vstupu zabraňuje. Pokud deklarujeme proměnnou jakou **private**, pouze třída ve které je daná proměnná k ní může přistupovat. Modifikátor přístupu

protected umožňuje přístup k proměnným ve třídách, ve kterých se proměnná nachází nebo k těm třídám, ze kterých je daná třída odvozená.

```php
<?php

    class Car {

        // The properties
        public $comp;
        public $color = 'red';
        public $hasSunRoof = true;

        // The method that says hello
        public function hello() {
            return "Toto auto je <i>" . $this->comp .
                    "</i>, a jeho barva je <i>" . $this->color;
        }

    }
```

Obrázek 19 - Využití $this

Umělé metody a konstanty

Konstruktor nám v objektovém programování začíná dvěma podtržítky a zapisuje se jako **__construct()**. Tato metoda se používá v případě, že chceme něco udělat ihned po vytvoření objektu mimo třídu. Obvykle se používá, například pokud chceme přiřadit hodnotu vlastnosti.

Dalším prvkem usnadňující práci je konstanta **__CLASS__**, kterým získáme název třídy, ve které se metoda nachází. Dalšími pomocnými konstantami jsou například **__LINE__**, **__FILE__** nebo **__METHOD__**.

```php
<?php

    class Car {

        private $model;

        //__construct
        public function __construct($model) {
            $this->model = $model;
        }

        public function getCarModel() {
            return ' Model auta je: ' . $this->model;
        }

    }

    $car1 = new Car("Škoda");

    echo $car1->getCarModel();
```

Obrázek 20 - Využití konstruktoru

Každá třída pomocí příkazu **extends** může dědit vlastnosti z třídy jiné. Proto kdybychom v tomto případě vytvořili třídu sportovních aut a určili, že bude dědit všechny vlastnosti z třídy auto, pak zdědí všechny metody a vlastnosti, které nejsou privátní. A tyto vlastnosti a metody pak můžeme doplnit o specifické pro danou třídu.

```php
<?php
    class Car {

        protected $model;

        public function setModel($model) {
            $this->model = $model;
        }

    }

    class SportsCars extends Car {

        public function hello() {
            return "Tento model je: <i>" . $this->model . "</i><br />";
        }

    }

    $sportsCar1 = new SportsCars();
    $sportsCar1->setModel('Toytoa');
    echo $sportsCar1->hello();
```

Obrázek 21 - Spojování tříd

Čtení ze souboru

V části o funkcionálním programování jsme si ukazovali, jak se dá napsat čtení ze souboru. Pokud se podíváme na tu stejnou funkci, ovšem psanou pomocí OOP zjistíme, že funkce se zdá být jednoduší a kratší.

```php
<?php
    $f = fopen( "robots.txt~", "r" );
    while ( ( $l = fgets( $f ) ) != null )
    {
        echo $l;
    };
    fclose( $f );
```

Obrázek 22 - Funkce pro čtení ze souboru

4.1.3 Porovnání

Pro porovnání programovacích paradigmat byly využita mimo reálných příkladů také obecná kritéria za použití vícekriteriální analýzy variant – metody váženého součtu. Na základě analýzy internetu – firem, které vytvářejí webové stránky a aplikace, nebo hodnocení vývojářů byly vybrány následující kritéria pro výběr programovacího paradigmatu. Písmeno za kritériem slouží pro lepší přehlednost v kriteriální tabulce. Na základě bodovací metody byly rozděleny váhy jednotlivých kritérií, které představují jejich důležitost.

Bodové ohodnocení

Porovnávané vstupy jsou informace získané na základě analýzy vývojářských zkušeností, vývojářských prostředí a firem, které se specializují na vývoj webových aplikací. Každý porovnávaný parametr je ohodnocen váhou, která zdůrazňuje důležitost tohoto parametru. Váhy parametrů byly určeny po konzultacích s odborníky z praxe, kterými byly vybrány ty nejzásadnější parametry, podle kterých by se mělo vybírat. Váhy jsou v rozmezí 1 – 3, kde nejnižší váha má nejvyšší stupeň závažnosti.

Váha se stupněm 1 je označena jako nejvyšší a je přiřazena parametrům, které mají vliv na výslednou webovou aplikaci její rychlost, bezpečnost a kvalitu webové aplikace. Jedná se tedy především o systémové nároky, časovou náročnost, velikost a přehlednost zdrojového kódu nebo jeho opakovanou použitelnost. Prostřední hodnotu reprezentuje váha s hodnotou 2, která je nastavena u parametrů určujících podporu a dokumentaci. Poslední

hodnotou neboli nejnižší je váha se stupněm 3. Je nastavena u zbytku kritérií, které nemají přímý vliv na kvalitu webové aplikace či její bezpečnost.

V rámci výsledků hodnocení jednotlivých parametrů jsou udělovány body v rozpětí 0 – 3, které představují pořadí použitelnosti tohoto programovacího paradigmatu. Méně bodů znamená lepší výsledek v daném kritériu. Pokud u programovacích paradigmat nalezneme shodný parametr, ohodnotíme ho stejným počtem bodů. Následně body jednotlivých kritérií vynásobíme jejich váhou a body sečteme. Programovací paradigma, které získá menší počet bodů je označeno za lepší.

Porovnávané parametry

- Cenová náročnost na nasazení aplikace a vývojové prostředí

Programovat funkcionálně nebo objektově vyžaduje určitou znalost vývojového prostředí a práci s ním. Vývojová prostředí, která se používají, jsou buď volně dostupná, nebo jsou placená a obsahují velké množství funkcí, které vývojáři ulehčí práci. Toto kritérium zohledňuje i využité licence a využitelné nástroje pro vývoj. Dalším posuzovaným faktorem tohoto kritéria je náročnost na nasazení. Webové aplikace programované funkcionálně nebo objektově nemají žádné specifické požadavky pro nasazení těchto systémů. Toto kritérium má váhu **3**, která představuje nejmenší váhu v bodovém hodnocení.

- Cenová náročnost vývoje

Celková cena na tým vývojářů se bude odvíjet od jejich pracovního nasazení, tedy časové náročnosti a rozsahu zadaného projektu. Dalším faktorem může být počet řádků kódu či znovupoužití funkcí nebo objektů. Toto kritérium má váhu **3**, která představuje nejmenší váhu v bodovém hodnocení, protože úzce nesouvisí s bezpečností či kvalitou vytvářené webové aplikace.

- Dostupnost odborných literatur

Toto kritérium posuzuje dostupnost literatury pro naučení daného programovacího paradigmatu. Také je zde zahrnut výskyt možných tutoriálů na internetu včetně instruktážních videí a knihoven, které se dají při učení využít. Toto kritérium má váhu **2**, která představuje prostřední váhu v bodovém hodnocení, protože není přímo vázaná na výsledek webové aplikace.

- Systémové nároky

Systémové nároky programovacích jazyků v podobě funkcionálního nebo objektového paradigmatu mají také roli v tomto hodnocení. Jedná se především o hardwarové nároky na provoz výsledné webové aplikace. Kritérium zabývající se systémovými nároky dostalo váhu **1**, protože přímo ovlivňuje chod výsledné webové aplikace.

- Časová náročnost

Časová náročnost vývoje webových aplikací závisí na velikosti a rozsáhlosti samotného projektu a tudíž nemá přímý vliv na její bezpečnost či kvalitu a proto je ohodnocena váhou **3**.

- Velikost zdrojového kódu

Velikostí zdrojového kódu se nerozumí počet znaků, ale celkově počet řádků s přihlédnutím na formátování a přehlednost kódu. Zdrojový kód by měl být také řádně okomentován. Tento parametr hraje jednu z hlavní rolí při výsledném hodnocení kvality webové aplikace, její rychlosti a náročnosti na paměť, proto je ohodnocen váhou **1**, tedy nejvíce důležitou.

- Přehlednost kódu

Z hlediska vývoje webové aplikace do budoucnosti je nutné, aby webové aplikace měli přehledný kód pro intuitivní příchod nových vývojářů, kteří se bez větších problému zorientují a budou moci aplikaci vytvářet dále. Tento parametr tak hraje důležitou roli, ne však takovou abychom mu dali lepší váhu než **2**, protože bezprostředně nezasahuje do kvality či bezpečnosti aplikace.

- Opakovaná použitelnost kódu

Vytváření webových stránek či aplikaci přináší také spoustu situací, kde se vyskytne nutnost využít stejný kód použitý jinde v aplikaci. Může se jednat o znovupoužití funkcí, metod či objektů, které byly předem vytvořeny. Toto kritérium má váhu **3**, která představuje nejmenší váhu v bodovém hodnocení.

- Využití operační paměti

Každá aplikace díky svému zdrojovému kódu zabere určitou část operační paměti a výkonu pro své výpočty. Při posuzování kvality webové aplikace toto kritérium hraje důležitou roli, a proto je ohodnocen váhou **1**.

- Využitelnost programovacího paradigmatu

Některé programovací paradigmata se využívají spíše pro menší aplikace, některé pro větší webové aplikace. Není to však dogma a proto tento parametr je ohodnocen nejmenší váhou **3**.

Parametry	Funkcionální programování	Objektové programování
Cenová náročnost na nasazení aplikace a vývojové prostředí	2	2
Dostupnost odborných literatur	3	2
Systémové nároky	2	2
Časová náročnost	2	3
Velikost zdrojového kódu	2	3
Přehlednost kódu	3	1
Opakovaná použitelnost kódu	2	1
Využití operační paměti	3	1
Využitelnost programovacího paradigmatu	2	2
Výsledek	**43**	**39**
Pořadí	**2.**	**1.**

Tabulka 2 - Tabulka kritérií

4.2 Webová aplikace

Další kapitolou této diplomové práce je plynulá návaznost na řešené příklady funkcionálního a objektového programování. V této části vytvoříme komplexní internetové řešení webové aplikace. Na základě poznatků z teoretické části práce ohledně vývoje

webových aplikací je zřejmé, jaké body je nutné řešit přednostně, ještě před tím, než webovou aplikaci začne vývojář vytvářet a také by mělo dojít k vytvoření představy o následném postupu vývoje. Pro začátek je důležité, aby si vývojář vytvořil představu o tom, jakou aplikaci chce vytvářet a hlavně, komu bude aplikace sloužit.

Webová aplikace, kterou nyní budeme vytvářet, by měla sloužit co největšímu počtu **bytových družstev v ČR**. Každé bytové družstvo má ze zákona povinnost mít své vlastní webové stránky, na kterých je nutné svolávat členské schůze družstevníků, prezentovat zápisy z těchto členských schůzí, vystavovat stanovy a mít k dispozici prostor, kde budou umísťovány soubory a další informace o dění v těchto družstvech.

Cílovou skupinou této webové aplikace jsou tedy bytová družstva a jejich družstevníci, kteří chtějí získávat informace o dění v bytovém družstvu a chtějí dostávat informace přicházející od představenstva. Systém nemusí fungovat jako informační kanál či fórum, ale jednou z jeho částí může být společná komunikace mezi družstevníky. Vývojář musí počítat s faktem, že systém budou chtít také využívat lidé ze starších generací, kteří nemají takový vztah a zkušenosti s výpočetní technikou, nebo lidé, kteří jsou nějakým způsobem hendikepovaní. I proto je nutné vytvářet webovou aplikaci, která bude jednoduchá, intuitivní a bude splňovat základní pravidla přístupnosti webových stránek.

Specifikace vývojových nástrojů a služeb

Webová aplikace bude vytvářena ve vývojářském prostředí NetBeans IDE verze 8.2. Jedná se o plně funkční vývojové prostředí, které nabízí mnoho vychytávek, připojení k FTP, formátování zdrojového kódu a možnost nápovědy. Pro tento projekt byla vybrána nejvýhodnější technologie databázového prostředí – MySQL. Tato databáze disponuje přístupem pomocí **phpmyadmin**, který pomáhá k rychlé editaci vytvořených tabulek a sloupců.

Doména a hosting pro představení a uložení zdrojových kódů pro webovou aplikaci budou zaregistrovány u společnosti Wedos a.s, která se řadí mezi největší české registrátory a patří mezi jedničky na českém trhu.

Systém bude programován a vytvářen v programovacím jazyce PHP, který se po vyhodnocení stal nejlepším možným řešením a to i z hlediska mých osobních zkušeností a sympatií k tomuto jazyku. V programování celé webové aplikace je

využito funkcionálního i objektového paradigmatu programování webových aplikací. V částech kódu je programováno i procedurálně.

4.2.1 Rozdělení webové aplikace

Bytové družstvo je typem družstva, který úzce souvisí s bydlením. Obecně je definováno jako právnická osoba. Každé bytové družstvo má své představenstvo, které se stará o administrativu, informace a rozhoduje na základě hlasů členů družstva. Členové družstva jsou součástí této skupiny lidí a mají právo rozhodovat a proto i oni budou mít přístup do tohoto systému, který bude pro všechny ostatní nepřístupný.

Některá bytová družstva jsou tvořena spojením několika vchodů, domů či dokonce bloků a proto je nutné všechny od sebe správně rozlišit.

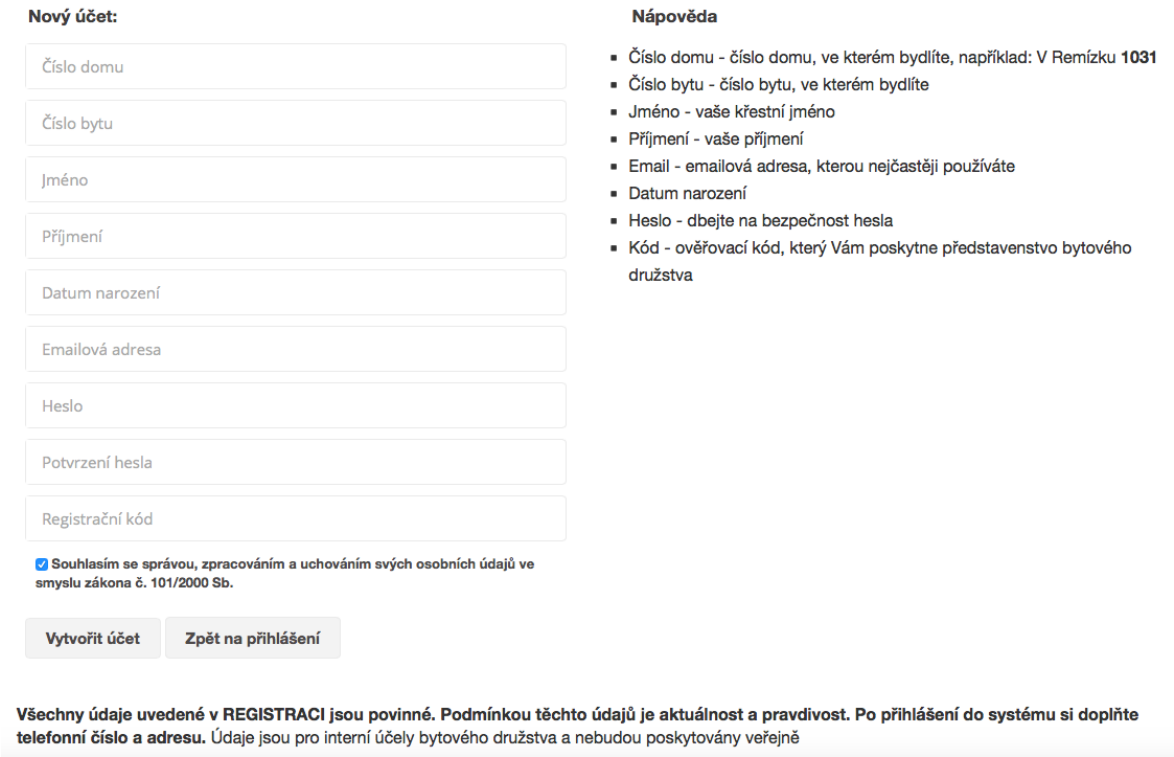
Registrace a přihlášení

Každý člen bytového družstva má právo se do systému registrovat. Tento proces lze v tomto informačním systému dokončit pouze s platným registračním kódem, který je generován unikátní pro každou bytovou jednotku. Bez tohoto klíče není možné se do systému zaregistrovat. Registrační kódy pak obdrží představenstvo družstva od administrátorů systému po dokončení nastavení a spuštění webové aplikace. Představenstvo si také může generovat registrační kódy samostatně. V tomto ohledu komunikace s administrátorem slouží pouze pro registraci představenstva.

Všichni ostatní, kteří nemají právo se do systému registrovat a nahlížet tak do nepřístupné části webu mohou sledovat veřejnou část, která může obsahovat informace jim určené.

Systém je tedy plně uzavřený pro potřeby družstva s výjimkou databáze třetí strany, která umožňuje jednoduší správu administrátorům a zároveň nabízí členům družstev využít funkce, které sdružují komunitu, ovšem bez narušení jakéhokoli soukromí bytového družstva.

Při celém procesu registrace se předpokládá, že uživatel zadává pravdivé údaje, které jsou na závěr v systému kontrolovány představenstvem.



Obrázek 23 - Registrační formulář do IS pro bytová družstva

4.2.2 Návrh webové aplikace

Návrhu webové aplikace jsme se věnovali v teoretické části. Nyní se ale zaměříme na systém pro bytová družstva. Tato webová aplikace byla vytvořena co nejvíce intuitivní, aby se jednoduše ovládala a spravovala. Důležité bylo, aby na sebe prvky navazovaly a byly dobře dohledatelné.

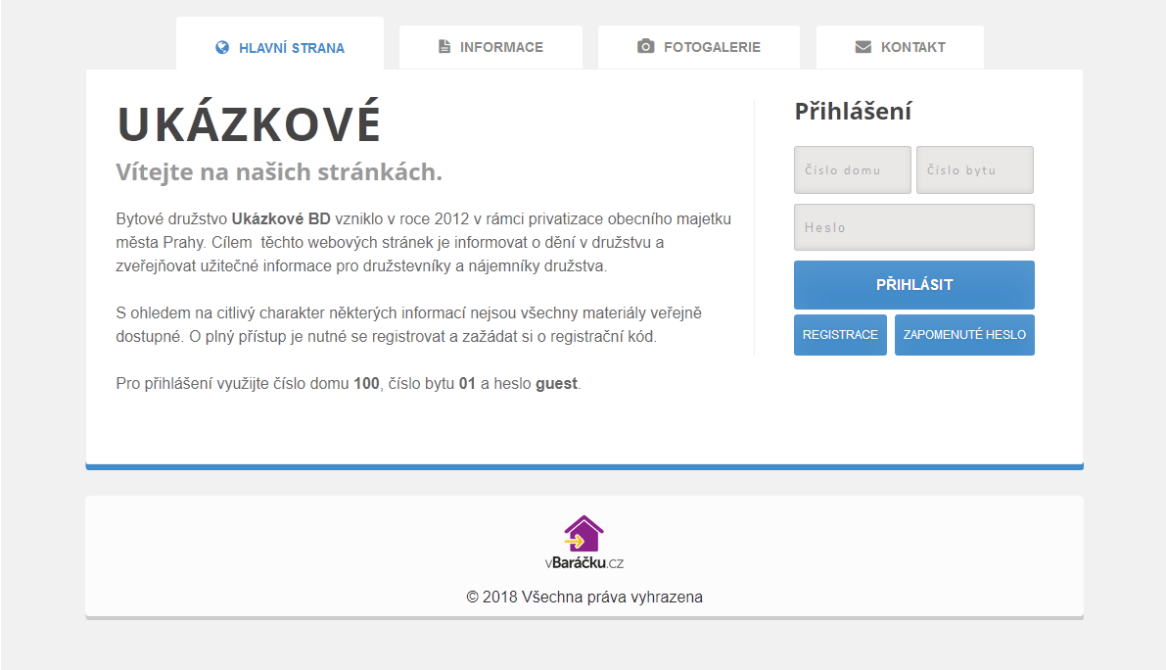
Další podmínkou návrhu byl jednoduchý a decentní design, který nebude křiklavý a uživatel se zde bude cítit pohodlně. Všechny tyto věci dělají přehledný systém, který se samozřejmě i nadále dá vylepšovat.

V neposlední řadě ve světě moderních technologií, bylo nutné, aby byla aplikace responzivní, což se nám z větší části podařilo, ale vše záleží na administrátorovi systému, jak bude systém spravovat a co do něj bude vkládat.

Grafické zpracování

Internet je prostředí, kde lze naleznout spousty informací k vytváření webových aplikací. Lze také zde nalézt velké množství grafických šablon. Pro tento systém jsem použil šablonu, která je volně a bezplatně ke stažení a doopravil jsem si ji k obrazu svému. S tímto úkolem mi pomohl kamarád, který se živý jako profesionální grafik.

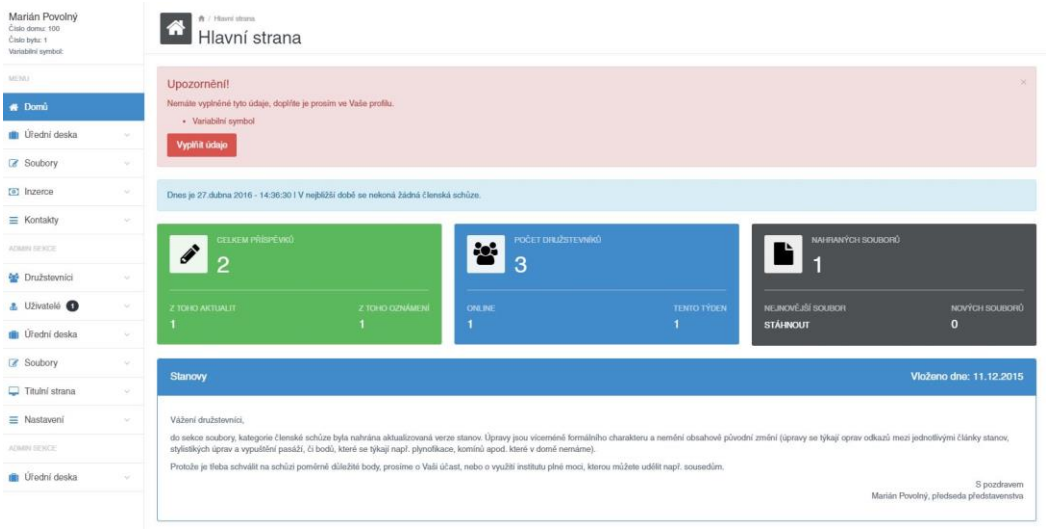
Úvodní strana



Obrázek 24 - Vyobrazení titulní strany Webové aplikace

Úvodní strana se skládá ze čtyř hlavní kategorií a boxu pro přihlášení, registraci či změnu hesla při jeho zapomenutí. Stránka využívá technologie JavaScriptu na hezké a jednoduché efekty a je barevně laděna stejně tak jako je systém po přihlášení. Všechny součásti této stránky si edituje a spravuje samo představenstvo / administrátor webu přímo z aplikace.

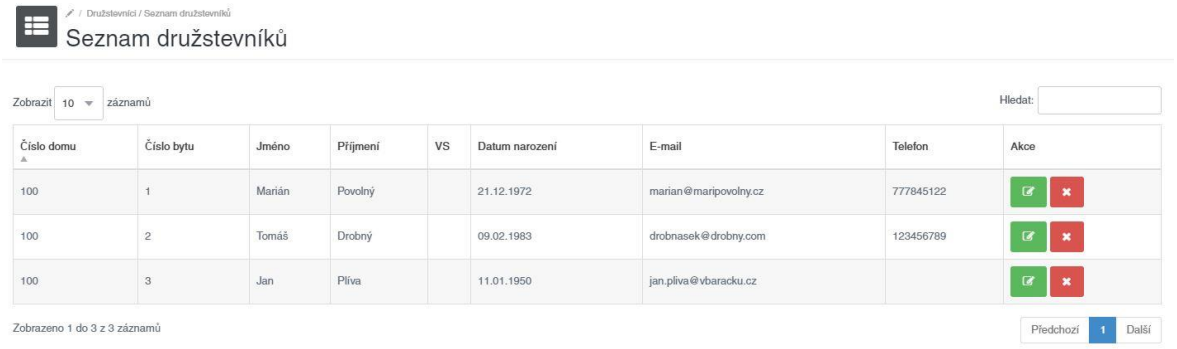
Hlavní strana webové aplikace – Dashboard



Obrázek 25 - Hlavní strana webové aplikace – Dashboard

Stránka webové aplikace po přihlášení. Systém již sám rozezná, pod jakou úrovní oprávnění byl uživatel přihlášen a podle toho se zobrazí to co je pro tato oprávnění povolené. Na hlavní straně se zobrazují jen nejnovější aktualita, oznámení a dále statistiky uživatelů, příspěvků a souborů. Také je na hlavní straně notifikační lišta a v případě problému i lišta upozornění.

Tabulky a seznamy

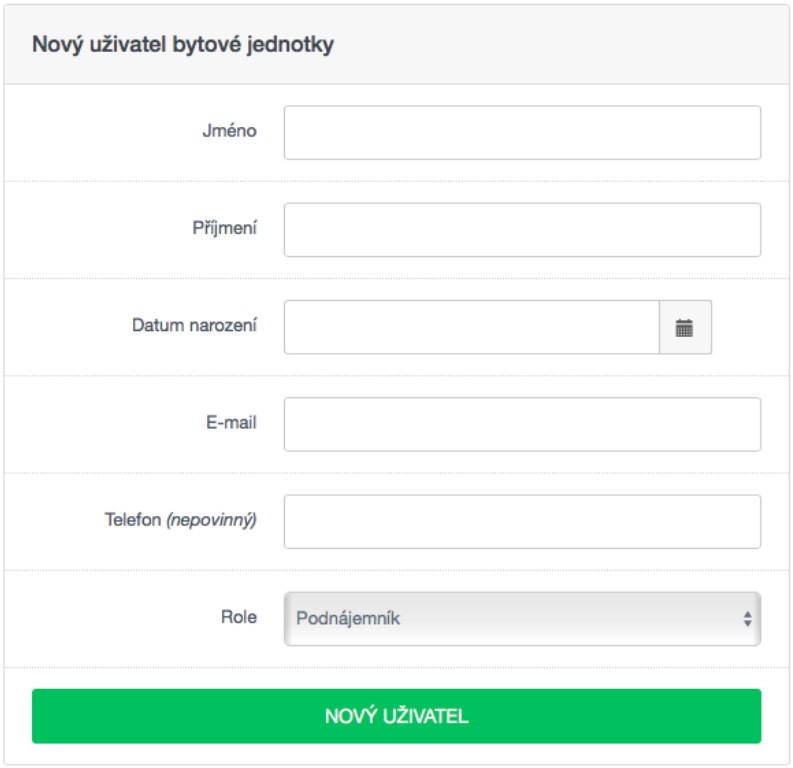


Obrázek 26 - Tabulky a seznamy

V systému je nespočet tabulek a seznamů, které zobrazují různá data a informace. Tabulky jsou spojeny s technologií tzv. Datatables, která umožňuje vyhledávání a řazení dat podle potřeby.

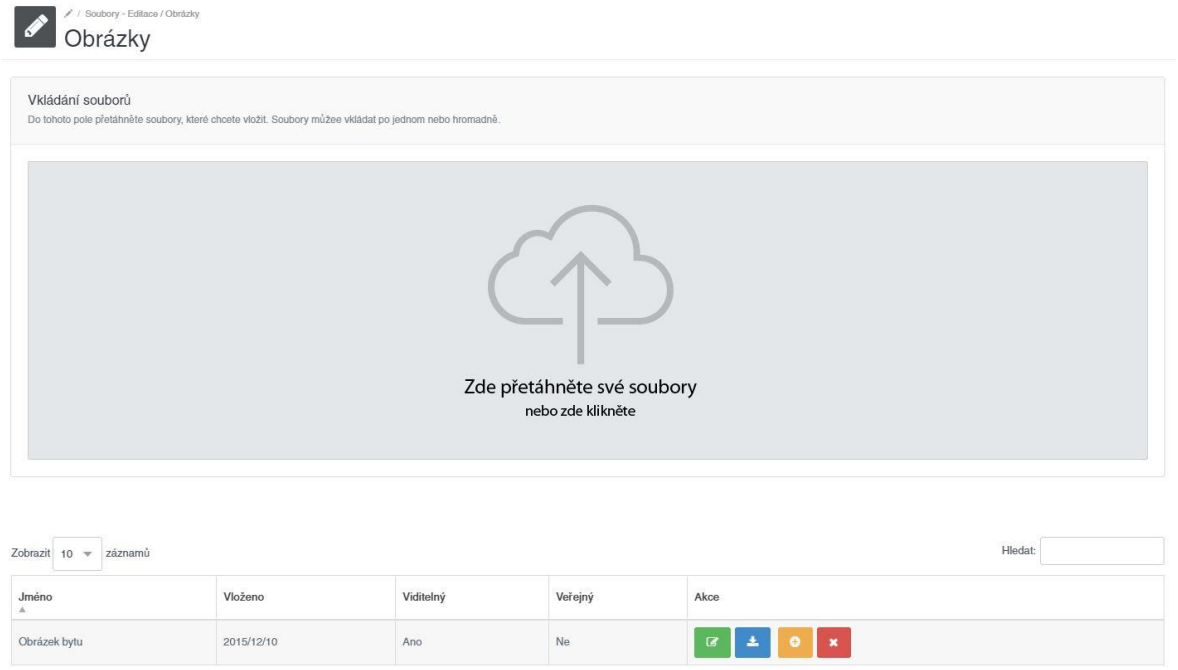
Formuláře

Formuláře jsou nezbytnou součástí každé webové aplikace. Pomocí formulářů uživatelé systému ukládají nová data do databáze nebo editují data stávající. Při využití formulářů je důležité nezapomenout na ověřování vstupních dat, protože uživatelé se mohou nechtěně či záměrně splést a snažit se pak do databáze vložit data, která jsou nevalidní nebo data, která by dokonce aplikaci mohli ohrozit z hlediska její bezpečnosti – SQL Injection, apod. Jednotlivé inputy ve formuláři se dají taky rozdělit do jednotlivých typů jako jsou text, password či email.



Obrázek 27 – Formuláře

Souborový systém



Obrázek 28 - Souborový systém

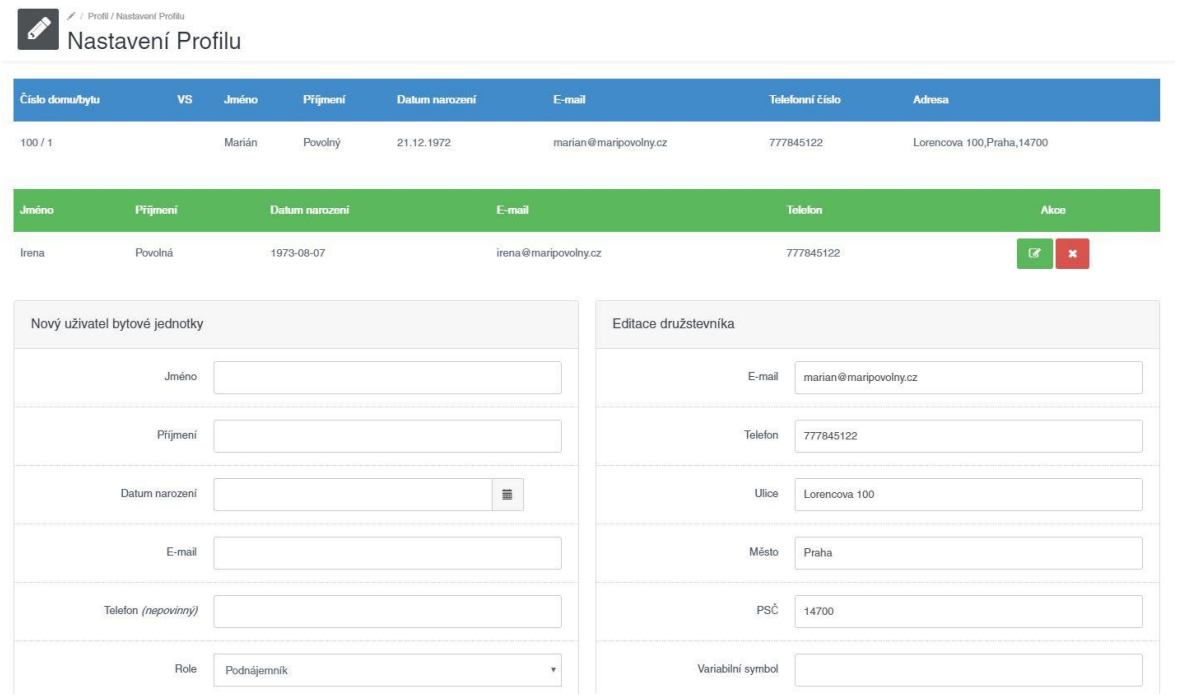
Tento obrázek nám ukazuje, jak se do systému vkládají soubory a jak se dají administrovat. Nyní se nacházíme ve složce „Obrázky“ kde je vložen soubor –

obrázek „Obrázek bytu“. Každý soubor má také své vlastnosti jako viditelnost a veřejnost, kde ho buď zobrazí, nebo skryje v určitých částech systému.

Některé dokumenty je nutné vystavovat i veřejnosti, například stanovy družstva či smlouvu o nájmu. Toto nám zpracovává pole „Veřejné“, jakmile je soubor veřejný může být viděn na úvodní straně webové aplikace v sekci informace.

Systém zpracování souborů umožňuje také archivovat soubory. To jsou takové, které se logicky přesunou do archivu a nejsou zobrazené v souborech.

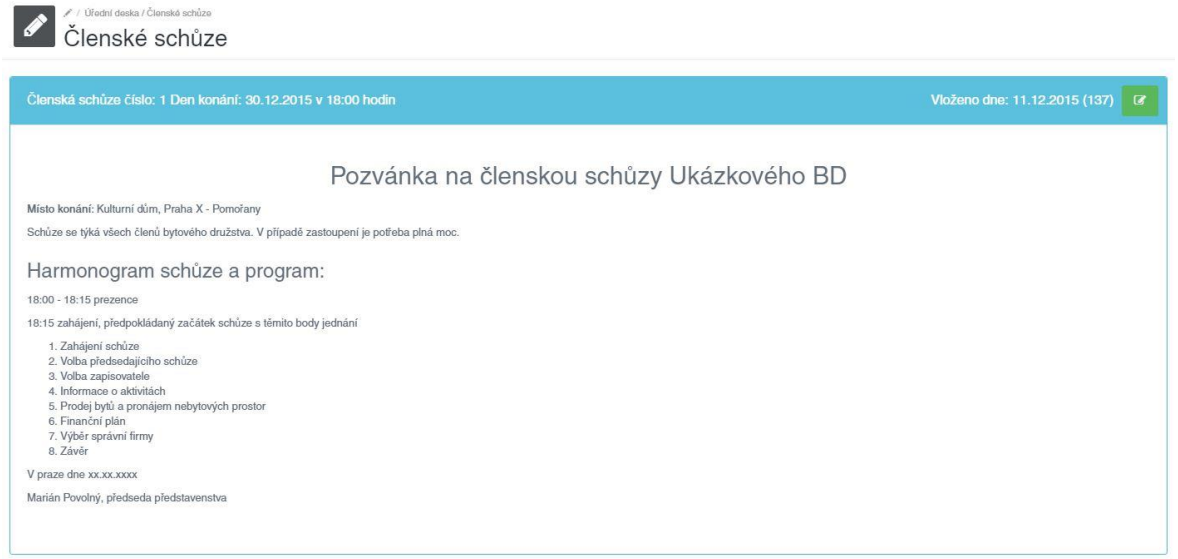
Nastavení profilu uživatele



Obrázek 29 - Nastavení uživatelského profilu

Každý uživatel má svůj profil, kde nastavuje své kontaktní informace, přidává uživatele nebo podnájemníky bytu či si může změnit jméno.

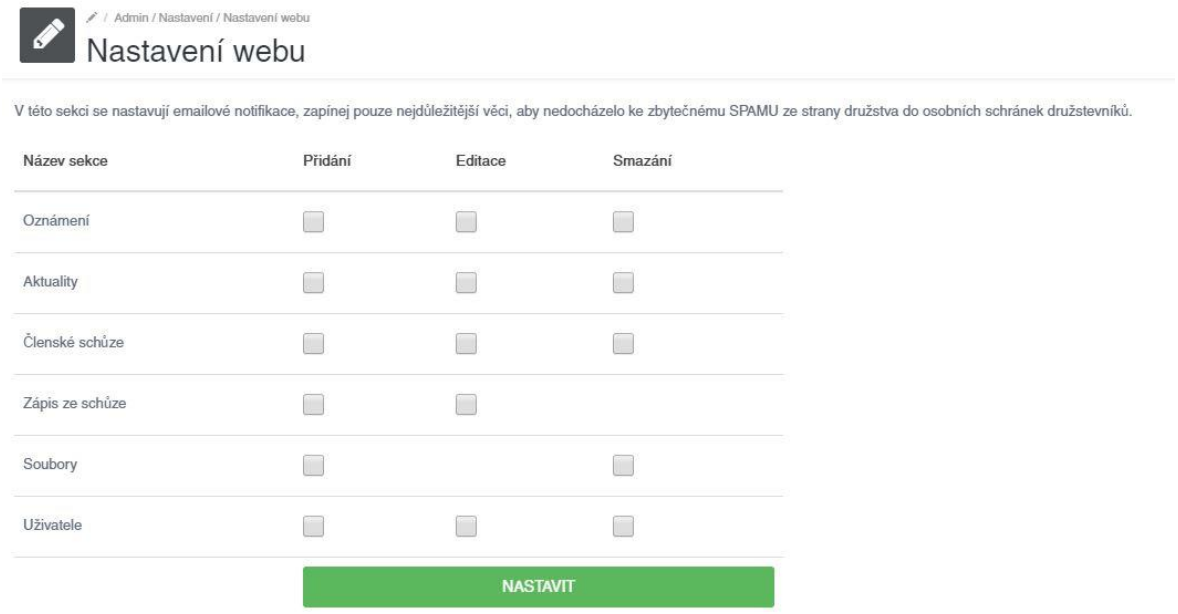
Členská schůze



Obrázek 30 - Návrh členské schůze

Členská schůze je jednou z hlavních součástí celého systému. Můžeme říci, že se jedná o nejpoužívanější funkcionalitu. Takto se v systému zobrazují pozvánky na členskou schůzi.

Notifikace

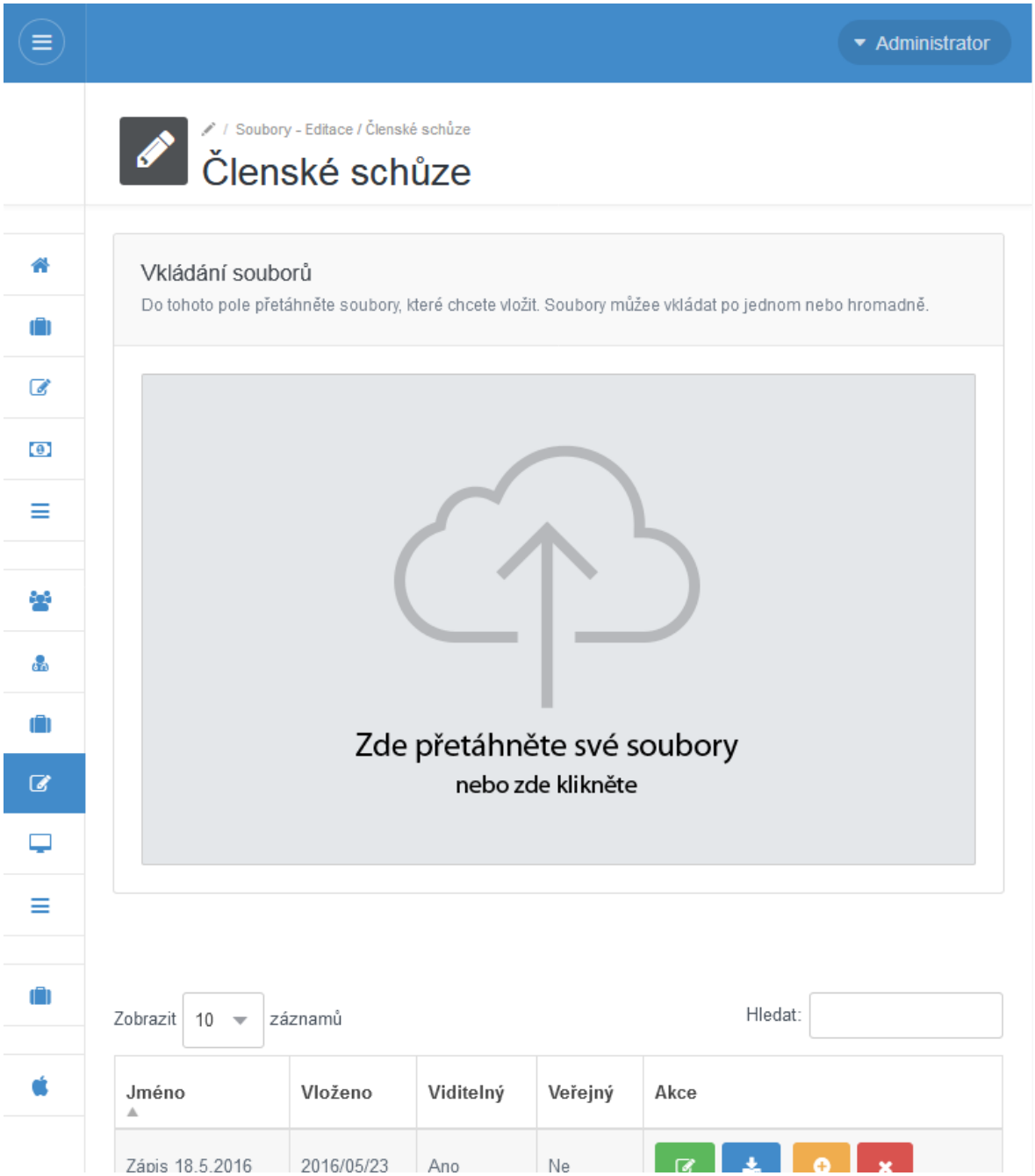


Obrázek 31 - Nastavení notifikací

V systému se tímto způsobem dají nastavit notifikace. Po zaškrtnutí se začnou odesílat na emailové adresy družstevníku informace o přidání, editaci či smazání některé z vybraných věcí.

Responzivní design

V současné době je nesmírně důležité, aby každá webová aplikace disponovala moderním designem, který bude současně optimalizovaný pro různé typy zařízeních včetně těch mobilních. Responzivní design se deklaruje v zdrojovém kódu kaskádového stylu pomocí tzv. **media_queries**. Definovat velikost můžeme v dynamických velikostech a pro určitou velikost obrazovky. Aplikace se nám díky dynamické šířce přizpůsobí sama danému zařízení. Některé elementy se však musí deklarovat v šířce pevné, aby byl zachován jejich původní design či účel.



Obrázek 32 - Ukázka responzivního designu aplikace

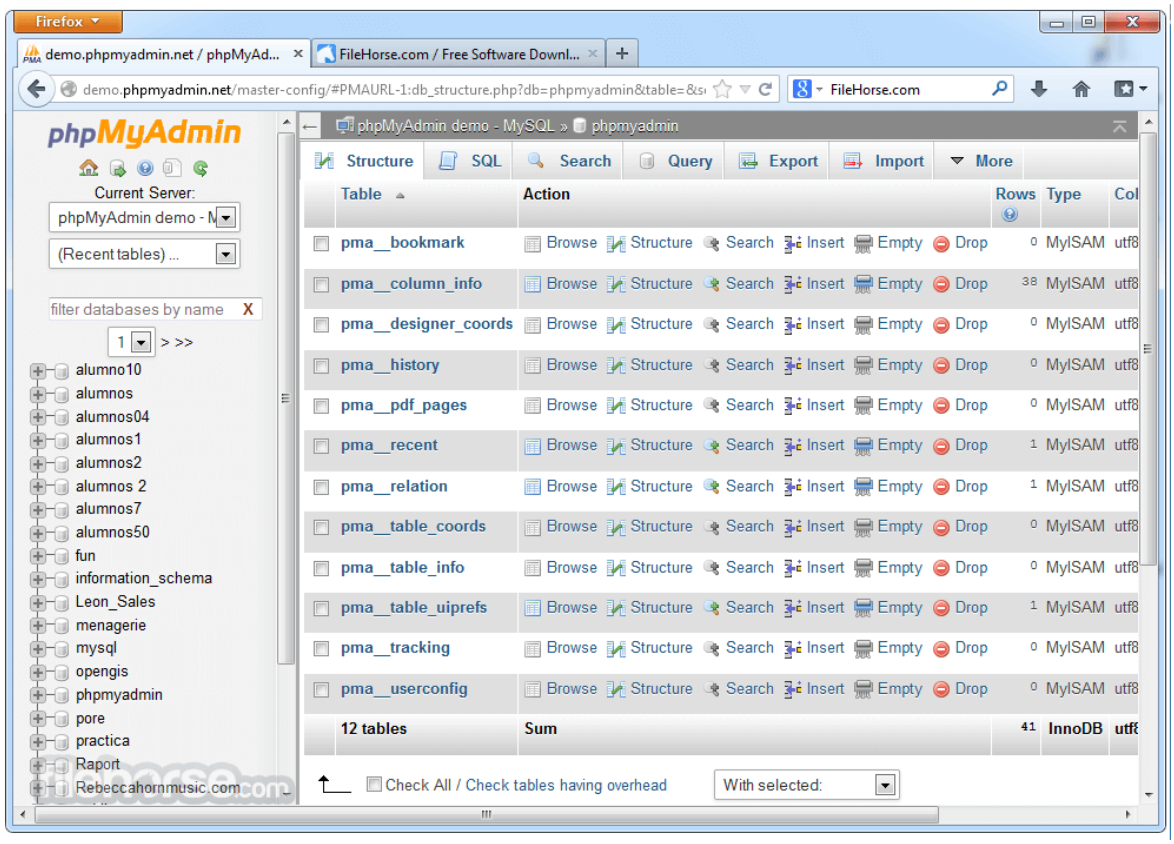
Databázová struktura

Použitá databáze je **MySQL**, která nabízí spoustu výhod a je často používaná pro projekty webových aplikací. Systém pro bytová družstva obsahuje celkem tři přístupy do databází, které slouží k různým funkcím.

Hlavní databází tohoto systému je soubor tabulek, do kterého se ukládají všechny informace o členských schůzí, oznámeních, aktualitách nebo družstevnících a jejich oprávnění.

Vedle této hlavní databáze jsou další dvě, které propojují systém s ostatními bytovými družstvy ať už ve formě nastavení a správy pro administrátory nebo propojují systém inzerce.

Tabulky v těchto databázích obsahují několik desítek sloupců, které uchovávají textové hodnoty, data vložení příspěvků, číselné hodnoty nebo hodnoty logické. Kódování pro textové typy jako jsou **VARCHAR** či **TEXT** je použito **utf8_general_ci**. Stejné kódování pak používá samotná webová aplikace.



Obrázek 33 - Ilustrační obrázek - PHP MyAdmin[9]

Na ilustračním přiloženém obrázku je možné vidět jednoduchost administrace databáze. Elegantně tak můžeme zpravovat celou databázi, vytvářet tabulky, vkládat dotazy či mazat data. Při chodu webové aplikace se však nevyplatí měnit uložená data v databázi

[9] Zdroj: https://www.filehorse.com/download-phpmyadmin/16806/

přímo, ale lepší je použít webovou aplikaci z důvodů kódování a ukládání dat do MySQL. Může zde dojít k chybám.

4.2.3 Funkce

Každá webová aplikace disponuje hned několika funkcemi, které ji dělají více atraktivní pro okolí a čím více funkcí aplikace má, tím větší množství potenciálních zákazníků přiláká.

Tato aplikace obsahuje nejdůležitější funkce, které každé bytové družstvo nezbytně potřebuje s vizí vytvoření dalších funkcionalit, které nejsou pro chod družstva nezbytně nutné. Některé z těchto funkcí můžou být i dobrovolné, protože je zde možnost je zapnout či vypnout, a některé z funkcionalit jsou vytvořeny jako moduly přímo pro určité bytové družstvo a dají se zapínat či vypínat.

Aktuální seznam funkcí

Členské schůze

Jedna z povinností představenstva je informovat členy družstva o plánovaných schůzkách. Získávat informace o konání členské schůze nám pomůže tato funkce, díky které může pověřená osoba rozesílat pozvánky, vkládat zápis či úplně zrušit členskou schůzi. Uživatelé jsou na nově konající se schůzi informování pomocí emailové adresy, kterou zadávali do systému a mají v systému i notifikační bar, kde vidí, za jak dlouho se nejbližší členská schůze koná. Jedná se tak o kompletní správu členských schůzí a družstvo tak splní svou povinnost a informuje družstevníky o zasedáních bytového družstva.

Oznámení

Informace je důležité poskytovat i jiným způsobem než slovy. Touto funkcí představenstvo plní svůj závazek informování členů družstva o blízkých akcích, skutečnostech či ostatních věcech týkajících se věci bytového družstva. Oznámení musí být dle zákona vyvěšena několik týdnů předem, a proto jsou zde ohlídány termíny a nevratná historie, která hlídá, zobrazuje příspěvky. Tato funkcionalita je jednou z majoritních funkcí systému, a proto je zobrazeno nejnovější oznámení na hlavní straně systému.

Ankety

Anketní systém je vytvořen tak jako ho všichni znají. Pomáhá představenstvu rozhodovat o věcech, ke kterým je potřeba hlasovat, ale není zapotřebí svolávat hned členskou schůzi. Každý družstevník má jeden hlas. Představenstvo tak má kompletní přehled o hlasování.

Soubory

Každé bytové družstvo potřebuje vydávat nové dokumenty, které obsahují informace důležité pro chod. V systému je možnost vytvoření a třídění souborů do složek. Dále je možné soubory zpět stahovat, přejmenovat či pouze zjistit kdo je nahrál a kdy. V souborovém systému se dá vyhledávat, i když nativně jsou soubory řazeny podle doby vložení.

Inzerce

Tato funkce využívá propojení všech bytových družstev. Ke spojení dojde mezi všemi, kdo mají tuto funkci aktivovanou ve svém profilu. Od té doby pak můžou nabízet ostatním bytovým družstvům své služby a dávat jim nabídky na určitý sortiment zboží.

Notifikace

Bytové družstvo musí informovat každého družstevníka, a proto jsou zde notifikace, které odešlou informaci na emailovou adresu o skutečnosti přidání nové důležité zprávy. Družstevník se pak tedy nemůže vymlouvat, že o dané informaci nevěděl. O čem všem budou chodit na emailovou adresu informace, si může určit samo představenstvo.

Všechny tyto funkce jsou volně dostupné pro každého uživatele systému. Poslední možností je ještě **Kontaktní formulář**, kde může napsat zprávu představenstvu, či nastavení jeho profilu a úprava hesla či kontaktních údajů.

Další funkce jsou již vázané na zapnuté doplňky systému, které musí hlavní administrátor stránek povolit.

Schůze představenstva

Jak jsem již napsal, každé bytové družstvo má své představenstvo, které se také musí scházet. Z toho důvodu, zde byl vytvořen doplněk pro schůzi představenstva, který funguje

stejně jako členské schůze s rozdílem, že je dostupný pouze představenstvu. Ostatní uživatelé pouze vidí zápis z této schůze.

Kontrolní komise

Bytové družstvo musí mít také kontrolní komisi, která musí dávat také zprávy o šetření na webové stránky. Uživatelé si pak v této kategorii mohou přečíst zprávy od kontrolní komise.

Variabilní symbol

V důsledku toho, že každý družstevní je veden zvlášť tak pro rozeznání plateb je dobré mít variabilní symbol. Proto doplněk umožní zapsat a zobrazovat variabilní symboly ve výsledku hledání.

Další **funkcionality** jsou dostupné z určitého oprávnění v systému. Všechny výše zmíněné jsou funkce a doplňky, které vidí každý uživatel. **Představenstvo** a **členové kontrolní komise** mají svou administrační sekci, kde nové záznamy mohou vkládat. Představenstvo zde dále vidí seznam všech družstevníků a jejich kontaktní údaje. Dále je zde vypsán i seznam uživatelů bytu, protože družstevník může svůj byt pronajímat a proto není uživatelem bytu. Představenstvo může také nastavovat informace na titulní straně webové aplikace, která je dostupná všem bez toho aniž by se museli přihlásit.

Na titulní stranu se mohou psát aktuality z dění v bytovém družstvu. Informace o pronájmech a prodejích podílů v bytovém družstvu, či jsou zde zobrazeny fotografie domu nebo kontaktní údaje na představenstvo.

Návrhy dalších funkcí

V této kapitole dojde k zamyšlení nad dalšími funkcionalitami, které pro webovou aplikaci můžeme naprogramovat a udělat ji tak ještě lepší pro správce a představenstvo bytových družstev stejně tako jako pro samé uživatele.

Účetnictví

Každé bytové družstvo si dělá vlastní správu příjmů a výdajů za dané období. Většinou tyto družstva využívají MS Office, které bychom touto funkcionalitou mohli nahradit. Jedná se o jakýsi systém zápisu příjmů a výdajů a počítaní potřebných výsledků.

S touto funkcí by se do systému dal naprogramovat doplněk této funkce, který by komunikoval s bankou a podle variabilního symbolu by ověřoval zaplacení nájmu, splátky či jiných příspěvků bytovému družstvu. Představenstvo družstva by tak mohlo efektivně sledovat pohyby na účtech a aktuální zůstatek.

Všechny tyto čísla by se automaticky vykreslovala v grafech ve webové aplikaci.

Revizní kalendář

Každý družstevník chce mít přehled o akcích a termínech, které nesmí propásnout. Není lepší způsob než tyto termíny vypisovat do speciálního kalendáře, který se bude dát exportovat do Apple kalendáře či Outlooku.

Chatbox

V nejnovějších webových aplikacích se začíná vyvíjet tzv. Chatbox, který v reálném čase komunikuje přímo s klientem nebo zákazníkem a je formou umělé inteligence. Zákazník si myslí, že hovoří s administrátorem, ale odpovědi jsou generovány počítačem. Tento systém komunikace by se dal využít například v komunikaci se správci, kde by družstevníci mohli být interaktivně informovaní o událostech, které se blíží, apod.

4.2.4 Programování

Webová aplikace byla naprogramována v jazyce **PHP** a při jejím vývoji byly použity cykly, funkce, proměnné a další objekty, které tento jazyk nabízí. Z velké části tvoří kód webové aplikace **SQL** dotazy.

Z hlediska statické části stránky, která je vytvořena za použití **HTML** a **kaskádových stylů**. Vše co se týče designu a grafiky je také spojeno s technologií zvaná **Bootstrap**, která nám zajišťuje responzivitu pro jiná elektronická zařízení.

Při vytváření informačního systému bylo také použito jazyka Javascript, který nám celý web doplňuje o dynamické prvky a efekty. Jeho technologie a hlavně jeho frameworků je využito například při nahrávání souborů, dále například v přechodech mezi stránkami na úvodní straně. Javascriptové knihovny byly staženy z internetu pod open source licencí.

Ukázky kódu

```php
$timeout = $timeout * 60; // minuty -> sekundy
if (isset($_SESSION['start_time'])) {
    $elapsed_time = time() - $_SESSION['start_time'];
    if ($elapsed_time >= $timeout) {
        session_destroy();
        header("Location: $logout_redirect_url");
        $datumLogOut = date("Y-m-d H:i:s");
        if($_SESSION['[redacted]'] == 1){
            $logON = mysqli_query($link_bd, "UPDATE [redacted] SET online = 0 WHERE ID = '$_SESSION[id]'");
            $logout = mysqli_query($link_bd, "UPDATE [redacted] SET date_logout = '$datumLogOut' WHERE ID ='$_SESSION[id]'");
        } else {
            $logON = mysqli_query($link, "UPDATE [redacted] SET online = 0 WHERE ID = '$_SESSION[id]'");
            $logout = mysqli_query($link, "UPDATE [redacted] SET date_logout = '$datumLogOut' WHERE ID ='$_SESSION[id]'");
        }
    }
}
$_SESSION['start_time'] = time();
```

Obrázek 34 - Ukázka kódu - Čas přihlášení a odhlášení

Výše uvedený zdrojový kód odhlašuje uživatele portálu po 15 minutách neaktivity. Odhlásit uživatele můžeme pouze tedy tak, že stopujeme jeho poslední krok na stránce, nebo spíše čas jeho poslední aktivity a ten porovnáváme s časem o 15 minut větším. Pokud zde bude platit podmínka, že čas odhlášení je větší než čas poslední aktivity + 15 minut, pak to uživatel odhlásí.

```php
<?php
    $dny["Mon"] = "Pondělí";
    $dny["Tue"] = "Úterý";
    $dny["Wed"] = "Středa";
    $dny["Thu"] = "Čtvrtek";
    $dny["Fri"] = "Pátek";
    $dny["Sat"] = "Sobota";
    $dny["Sun"] = "Neděle";

    $mesice = array(1=>"ledna", "února", "března",
    "dubna", "května", "června",
    "července", "srpna", "září",
    "října", "listopadu", "prosince");

    $dnes = Date(" j.") .
    $mesice[(int)Date("m")] . Date(" Y - H:i:s");

    echo 'Dnes je '.$dnes;

    $dnesek = date('Y-m-d');

    $vypisSchuzeH = mysqli_query($link, "SELECT * FROM [redacted] WHERE denkonani > now()order by denkonani asc limit 1");
    $resultSchuzeH = mysqli_fetch_assoc($vypisSchuzeH);
    $datumKon = date("d.m.Y", strtotime($resultSchuzeH['denkonani']));
    $casS = date("H:i", strtotime($resultSchuzeH['denkonani']));
    if($resultSchuzeH >= 1)
    {
    echo " |"." Nejbližší schůze, na kterou jste zváni se koná: <b> ".$datumKon."</b> v <b>".$casS."</b>";
    }
    else{
        echo " | V nejbližší době se nekoná žádná členská schůze.";
    }


?>
```

Obrázek 35 - Ukázka kódu - Infopanel - Termín členské schůze

V dalším příkladu kódu můžeme vidět výpis nejbližší členské schůze v infopanelu na hlavní stránce. V SQL dotazu můžeme vidět, že se porovnává sloupec „denkonani“ s aktuálním časem databáze. Následně se pak vypíše pouze jedna schůze, která je nejbližší – datum je formátováno v českém jazyce.

```php
function random_password($len) {
    $pool = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890+-*/';
    $stringPass = '';
    for ($i=0; $i < $len; $i++) {
        $stringPass .= substr($pool, mt_rand(0, strlen($pool) -1), 1);
    }
    return $stringPass;
}
$genPass = random_password(7);
```

Obrázek 36 - Ukázka kódu - Generování nového hesla

Funkce pro generování nového hesla je ve webové aplikaci použita hned dvakrát. Poprvé se s touto funkcí můžeme setkat na úvodní straně webové aplikace, pokud si nepamatujeme heslo pro přihlášení – stejně tak může administrátor vygenerovat nové heslo družstevníkovi přes systém. Druhým výskytem je vytváření družstevníka přes systém. Není nutné vyplňovat zde heslo, které bude uživatel používat, ale heslo se vygeneruje samo a odešle se na vyplněný email spolu s dalšími údaji.

4.2.5 Příležitosti ke zlepšení

Po dokončení webové aplikace se nabízejí otázky týkající se vylepšení či příležitostí. Nejedná se o odborný rozbor, a proto není zapotřebí dělat SWOT analýzu, ale jedná se pouhý tok myšlenek.

První věcí, která by se v případě rozšíření systému musela udělat je jeho certifikace a zacílení na větší zabezpečení webové aplikace proti potenciálně škodlivým útokům. Další překážkou je nová směrnice Evropské unie GDPR, která udává nová pravidla pro systémy jako je tento ve správě osobních, citlivých údajů. V případě budoucího rozvoje a využívání systému by se webová aplikace musela právně ošetřit z této stránky věci. Myšlenkou distribuce této webové aplikace je momentálně sdílení zdrojových kódů, což v případě rozmachu aplikace není šťastné řešení a proto by se aplikace musela upravit tak, aby její data ležela na jednom serveru (vlastním či pronajmutím) a na vlastní doménu by se přesměrovávalo přes registrátora. V případě velkého rozmachu aplikace se dají všechny základní funkce předělat i do webových aplikace a udělat tak větší komfort všem družstvům, které systém používají. V neposlední řadě se dá rozšířit i funkce notifikací formou SMS upozornění. Družstevníci by si pak mohli vybrat, jakým způsobem chtějí být upozorňováni.

Budoucí vývoj

Budoucí vývoj této webové aplikace nám nabídne nové možnosti a technologie. Rozšíření funkcí a nápady do budoucna máme napsány výše, ale nové technologie nás zajímají nyní. Určitě využijeme technologie **AJAX**, která nám umožní synchronizaci dat z formulářů bez znovunačtení stránky. Dalšími technologiemi, které se dají využít, jsou preprocesory kaskádových stylů **SASS** a **LESS**, které přidávají do možnost si definovat proměnné a další skvělé vychytávky pro vývoj grafické části webové aplikace.

Nejsou jen technologie, které ženou vývoj webových aplikací směrem vpřed, ale této aplikaci se bude hodit mnoho vylepšení, které by mohlo zvýšit komfort a následně počet klientů. Komunikace pomocí Chatboxů, líbivější grafické zpracování titulní strany a další vychytávky, které by webovou aplikaci udělali líbivější.

Všechny zdrojové kódy budou zjednodušeny a je zde možné, že dojde k vymyšlení lepších a efektivnějších řešení pro určité funkcionality. Tento druh webových aplikací lidé uvítají a proto je v zájmu samotných vývojářů tuto aplikaci tvořit i do budoucna.

4.2.6 Analýza jiných řešení

Na internetu, nebo minimálně na českém internetu, existuje několik webových portálů, či aplikací, které se zabývají také systémy pro bytová družstva. Z hlediska vývoje těchto aplikací tuzemské firmy nejčastěji využívají technologie využité i pro vývoj této aplikace jako jsou **PHP**, **CSS**, **HTML** ve spojení s frameworky. Důležitou součástí všech těchto aplikací je také **Javascript**. Některé firmy dělají webové aplikace pomocí technologie od firmy Microsoft – ASP. NET, jiné zase vytvářejí webové aplikace ve starších jazycích a to například v Ruby, Pythonu, Perlu, apod. Všechny tyto aplikace ale mají jednu věc společnou a to je, že jsou vyvíjena za využitím principů funkcionálního, objektového a ještě procedurálního programování.

V konkrétním případě webové aplikace pro bytová družstva se nabízí na českém internetu hned několik řešení. Stručná analýza funkcí, které přináší, nám přiblíží, jak vlastně fungují a jaká je firemní politika vývojářů těchto aplikací.

- **Sousedé.cz**

Webový portál, který je určen pro správu vlastníků jednotek či bytových družstev nese rysy sociální sítě. Jednotlivý družstevníci jsou zde v roli sousedů, kteří mají

přístup do svého bytového družstva. Díky rozsáhlé uživatelské základně může aplikace vydělávat na reklamě a využívat funkcionality ke komunikace v čele s veřejným fórem, kde se dá otevřít diskuze na jakékoli téma. Webový portál zároveň nabízí své služby v širším rozsahu v podobě magazínu či obchodu s nábytkem – ovšem vše je stále cíleno na bytová družstva. Portál disponuje stejnými funkcemi jako vytvořená aplikace s výjimkou kalendáře akcí. Webová aplikace je responzivní a optimalizovaná pro prohlížeče a k vývoji byly s největší pravděpodobností využity technologie CSS a HTML. (Sousedé.cz, 2018)

- **WebDomu.cz**

Webová aplikace, která umožňuje vytvořit zdá se samostatnou webovou stránku, která se na první pohled tváří jako jedinečná stránka bytového družstva. Na rozdíl od předchozí služby je zpoplatněná a můžeme zde spatřit i reklamní bannery. WebDomu nabízí také několik dalších služeb navíc jako službu právníka, apod. A to vše opět díky celkem velké uživatelské základně. Webová stránka je responzivní a také nabízí šifrovanou komunikaci pomocí protokolu https. Vývoj této aplikace probíhal za použití technologií HTML, PHP, CSS a Javascript. (WebDomu.cz, 2018)

Toto jsou dvě nejrozšířenější webové aplikace, které jsou určené primárně pro bytová družstva. Existují však firmy, které nabízí své služby vytváření webových stránek pro BD, což je individuální přístup a individuální web. V této oblasti je rozhodně ale lepší variantou hromadný přístup a díky tomu spojovat lidi z okolí, ovšem musí se dbát na bezpečnost celé aplikace. Další aplikace, se kterými jsem se v tomto ohledu setkal, byly vytvořeny pomocí technologie ASP. NET, či C#. V dnešní době, kdy se vše přesouvá na web a vše bude tzv. online, je ovšem celkem nešťastné řešení tuto aplikaci řešit lokálně.

5 Zhodnocení výsledků a doporučení

5.1 Porovnání programovacích paradigmat

V návaznosti na reálně řešené případy a na komparaci funkcionálního a objektového programování pomocí metody vícekriteriální analýzy variant došlo k vyhodnocení těchto paradigmat.

- Funkcionální programování není těžké na pochopení a naučení či na samotné programování. Odborník nebo laik, který se rozhodné vytvářet webovou aplikaci pomocí funkcionálního programování se nemusí učit nové programovací jazyky, stačí začít používat principy tohoto paradigmatu v běžné práci na běžných věcech a malými krůčky tak dojde k pochopení principu.

- Existují případy, kdy je lepší a jednoduší využít principy funkcionálního programování, který přináší své výhody do menších projektů či funkcí. **Zdrojové kódy bývají často delší a robustnější a jsou méně přehledné**. Programovat funkcionálně se dá téměř v jakémkoli programovacím jazyce a někteří vývojáři dokonce prosazují principy funkcionálního programování i v jiných jazycích než jen v PHP. Často se také využívá v Javascriptu, nebo také v čistě funkcionálních programovacích jazycích jako je Haskel či Ocaml. Díky rozmanitosti těchto programovacích paradigmat se vývojář může naučit další programovací jazyk a zdokonalit se tak ve svém oboru.

- Funkcionální programování není bezchybný a bezmyšlenkový styl programování. V tomto směru je opravdu nutné dodržovat jiný přístup ke kódu a to z funkcionálního hlediska, které se zdá být těžší než objektové vnímání, konkrétně například v objektově orientovaných termínech a pojmech, protože je zde podobnost s modely z reálného světa. Přemýšlení nad situacemi z reálného světa je ve funkcionálním paradigmatu mnohem složitější. Vzhledem k těmto obtížím **existuje méně lidí**, kteří funkcionální programování **využívají**, podporují tento styl, nebo jsou schopni ho naučit či vysvětlit dalším lidem. Z tohoto důvodu existuje také **méně odborné literatury** a možných tutoriálů. Funkcionální programování je vhodné zvolit

spíše na menších projektech, ovšem i obrovské webové aplikace (Facebook) jsou psané celé funkcionálně.

- U objektového programování došlo k seznámení se se základy s následným rozborem praktických příkladů. Objektově orientované programování je pro vývojáře, kteří byly zvyklý na procedurální programování či funkcionální programování, novou výzvou. K vývoji webových aplikací se dnes využívá převážně programovací jazyk PHP, který se, i za pomoci frameworků, kterých je nespočet, přibližuje spíše do objektového stylu programování. Příkladem mohou být důležité PHP rozšíření, které jsou založeny na objektově orientovaném programování.

- Pro menší projekty se tento druh programování příliš nedoporučuje, není totiž nutné vytvářet třídy, instance a vše důkladně rozdělovat tak, jako je to u komplexních řešení, na kterých většinou pracuje i více vývojářů a **přehlednost kódu** je tedy výrazně důležitější. Z tohoto hlediska je objektové paradigma lepší – přehlednost kódu je jedna z hlavních vlastností objektového programování, která napomáhá lepšímu rozvoji webových aplikací do budoucna.

Signálem pro využití objektově orientovaného programování při vývoji webové aplikace může být existence 10 – 20 funkcí. Pokud programátor zjistí, že některé z nich dělají přibližně to samé, pak je to impuls, který by měl vést k používání objektů a samotného OOP.

- Objektově orientované programování je moderní cesta k vývoji kvalitního softwaru a webových stránek či aplikací. Všechny majoritní a nejpoužívanější programovací jazyky jako jsou Ruby, Java, PHP, PERL, C#, tento způsob programování využívají.

- S využitím OOP se vývojář webových aplikací také stává důležitým členem pracovního týmu, protože díky OOP má základní znalosti tohoto paradigmatu a nebude tak mít problém programovat v jiném jazyce.

- Programování pomocí objektového programování usnadňuje vývojářům práci a šetří jim další minuty času, jenž mohou využít jinde. Tento čas

zároveň šetří i využíváním svých předpracovaných objektů v jiných projektech.

- S vývojem objektově orientovaných aplikací je důležité si vzít tužku a papír a objekty si nakreslit – shrnout diagramy a představit si aplikaci. Jednoduše pak vývojář nezapomene na detaily, které by mu mohli chybět, protože vidí svou aplikaci na papíře. Naopak u funkcionálního programování toto nejde tak jednoduše, protože se většina projektu píše pomocí funkcí, které si jen stěží nakreslíme na papír.

- I objektově orientované programování není bez chyby a nad vývojem se musí náležitě přemýšlet. Objektově orientované programování má výhodu ve své znovu použitelnosti – není nutné opět vytvářet objekty, které již byly vytvořeny. Může ale nastat situace, kde je nutné využít funkce, které sice už vytvořené jsou, ale bohužel je využít nelze z důvodu jejich přiřazení jiné třídě.

- Pomocí obou těchto programovacích paradigmat je možné tvořit bezpečné a kvalitní webové stránky či aplikace nebo srozumitelné programy bez vnitřních chyb a s následným možným rozvojem do budoucna.

- Oba koncepty mají různé metody ukládání dat a manipulaci s daty. V objektově orientovaném programování se data ukládají do atributů objektů a existují funkce, které pro daný objekt pracují a manipulují s ním. Ve funkcionálním programování vše vidíme jako transformaci dat. Data nejsou uložena v objektech, jsou transformována vytvořením nových verzí těchto dat a jejich manipulaci provádíme pomocí jedné z mnoha funkcí.

- Co se týče hodnoceného kritéria vývojového prostředí, je zřejmé, že obě paradigmata jsou na tom stejně a je jen volbou vývojáře rozhodnuto o vývojovém prostředí. O objektově orientované programování se však opírá několik desítek frameworků, které jsou náročnější na instalaci a přípravu před startem samotného vývoje. Některé z nich mají širší možnosti rozvoje a dokumentaci, větší počet tutoriálů či knihoven použitelných v aplikacích.

- I za předpokladu, že funkcionální programování obsahuje méně řádků kódu či je časově méně náročné je zde i faktor využití operační paměti. Funkcionální programování v celkové míře této paměti spotřebuje více, protože data nejsou ukládána v objektech.

Rys / Vlastnost	Funkcionální	Objektové
Styl programování	Deklarativní	Imperativní
Data a chování	Volně spojený s čistými, samostatnými funkcemi	Těsně spojený ve třídách s metodami
Správa stavů	Zachovává objekty jako neměnné hodnoty (minimalizuje stavové změny)	Podporuje mutaci objektů (stav) pomocí metod
Řídící tok	Funkce vyššího řádu a rekurze	Smyčky a podmíněné podmínky
Bezpečnostní rizika	Souběžné programování – větší šance chyb	Těžké dosáhnout

Tabulka 3 - Funkcionální vs. Objektové programovací paradigma

FP využívá neměnnosti dat společně s využitím čistých funkcí a referenční transparentnosti, která nám například přináší výhodu v testování aplikací v podobě souběžně běžícího testovaného kódu (můžeme testovat více částí zdrojového kódu najednou). Čisté funkce obohacují funkcionální programování menší mírou postranních efektů a zvyšují modularitu kódu – lehčí kód na testování a znovupoužití – generalizaci. Neměnnost dat nám přináší vysokou bezpečnost webové aplikace, ale míra zabezpečení OOP je na stejné úrovni.

Objektové programovací paradigma se opírá o vlastnosti, jako jsou dědičnost, polymorfismus, abstrakce či zapouzdření. Dědičnost je zde využívána namísto rekurze. Využívá třídy a metody pro vytváření objektů, které mohou být znovupoužitelné a učí tak programátora přemýšlet objektově. OOP přináší zlepšení v produktivitě vývoje

webových aplikací s využitím lepších jednoduše implementovatelných komponent. Objektově programované aplikace mají více řádků kódu, jsou časově náročnější na zpracování, ale jedná se o mnohonásobně přehlednější kód, který půjde lépe upravovat novými vývojáři. OOP také využívá pro chod aplikace méně paměti.

5.2 Webová aplikace

O webové aplikaci nikdy nemůžeme říci, že je naprogramovaná v nejnovějších technologiích za použití nejmodernějších prostředků. Webové technologie se vyvíjejí a jejich vývoj neustále postupuje vpřed.

Vytvořená webová aplikace obsahuje základní a nezbytné funkcionality, které bytové družstvo potřebuje pro svůj chod, a které ze zákona musí mít. Důležité je, že vytvořená aplikace podporuje registraci a zajišťuje soukromí družstevníků bytového družstva. V systému tak budou komunikovat pouze lidé, kteří spolu bydlí v jednom bytovém družstvu a budou zde mít svá data v soukromí a relativně dobře zabezpečena. V této webové aplikaci můžou členové představenstva vytvářet a kompletně spravovat členské schůze. Pod pojmem kompletní správa si představíme vkládání, editaci, mazání, ale také vkládání zápisů z těchto schůzi. Výhodou tohoto systému je notifikace nových oznámení, a všech novinek v systému v závislosti na nastavení celého webu. Webová aplikace dále umožňuje představenstvu vkládat soubory, vytvářet ankety, mít přehled o družstevnících a uživatelích bytových jednotek. Výhodou této aplikace je i komunikace s družstevníky pomocí oznámení a aktualit – včetně soukromých zpráv na emailové adresy. V budoucnosti by se do této aplikace v rámci jejího rozvoje daly naprogramovat další moduly, které by umožnili komunikaci s družstevníky pomocí soukromých zpráv, správu účetnictví či pokladny. Nejlepší nápady na nové moduly mají však samy bytové družstva, které systém využívají a vědí, co by byly schopni využívat.

Tento systém má však také několik potenciálně možných částí vývoje do budoucna. Zabezpečení systému je na vysoké úrovni, ale při využívání aplikace mezi bytovými družstvy by bylo nutné zabezpečit webové stránky pomocí SSL certifikace, což začne podporovat i firma Google, která označí nezabezpečené webové stránky jako potencionální hrozbu. V tomto případě si pak většina uživatelů internetů myslí, že jejich webové stránky byly napadeny, pokud se jím zobrazí hláška o nebezpečnosti či chybějícím certifikátu. V neposlední řadě by bylo nejlepší řešení systém provozovat na vlastním hostingu a pouze

přesměrovávat DNS záznamy na domény bytových družstev, z důvodu nepřístupnosti ke zdrojovému kódu.

Při programování webové aplikace bylo využito technologií funkcionálního a objektového programování v PHP společně se technologiemi na zobrazení frontednu jako jsou HTML, CSS, Javascript a databázi MySQL. Možností zrychlení aplikace a zpřehlednění zdrojového kódu by mohlo být využití nějakého ze zmíněných PHP frameworků, které ve spojení s frameworky LESS či SASS z kaskádových stylů mohou vytvořit lepší zdrojový kód a dokonce webové stránky tak zrychlit. Javascriptové knihovny, které se v této aplikaci využívají, jsou staženy z internetu jako open source, tudíž je také nutné dbát na jejich aktualizaci, aby nedošlo k nefunkčnosti webové aplikace díky zastaralému souboru. Nejlepší řešení tohoto problému je odkazování na online zdroje a vyloučení metody o fyzickém vlastnictví souboru na webhostingu.

V České republice existuje několik firem, které tento druh webové aplikace, tedy pro bytová družstva vytvářejí a provozují. V předchozí části práce (viz. Analýza jiných řešení) je zahrnuta rešerše nejpoužívanějších nalezených konkurenčních portálů. Ostatní portály nejsou zaindexovány nebo se specializují na individuální vývoj těchto stránek pro bytová družstva. Porovnáním těchto stránek došlo ke zjištění, že všechny z níže uvedených mají své silné stránky, avšak mají i rezervy, které by se dali vylepšovat. Rozdíly jsou i v technologiích, které jsou při vývoji použity – některé jsou vytvořené pomocí PHP, jiné pomocí ASP a dokonce existuje software pro správu bytových družstev naprogramovaný v .NET a funguje jako aplikace pro Windows. V rámci ochrany webových aplikací bohužel tyto služby neposkytují zdrojové kódy – nelze tedy porovnat zdrojové kódy, ale pouze funkce těchto dvou systémů.

Závěr

Vývoj webových aplikací byl, je a bude nezbytnou součástí vývoje samotného internetu. Webové stránky či aplikace jsou v dnešní době atraktivním a hlavním tahounem marketingu a dokonce nezbytným prvkem každé firemní nebo produktové identity. Všechno to, co na stránce vidí zákazník, by mělo být intuitivní, jednoduché, sladěné do firemních barev. Tyto všechny vlastnosti spolu s dalšími prvky tvoří kvalitní frontend webových stránek a aplikací, který se postará o první dojem a majiteli těchto stránek může zvýšit konverze a potenciální zákazník na stránkách zůstane mnohem déle – neodejde. Backend aplikace a webových stránek se dá vytvářet pomocí několika technologií a programovacích paradigmat.

O funkcionálním a objektovém programovacím paradigmatu lze říci, že jsou stejně kvalitní a obě se využívají pro jiný typ funkcionalit. V hodnocených parametrech je jasně vidět, že celkově lepší bude využití objektového programování, nicméně rozdíl mezi porovnávanými paradigmaty je minimální. Trendem se však podle prostudovaných webových aplikací stává ve větší míře objektové programování s využitím prvků funkcionálního. Nelze však popřít existenci aplikací naprogramovaných čistě funkcionálním stylem. Zkušení vývojáři programují webové aplikace v objektově orientovaném programování za použití funkcionální programování.

Při vývoji webových aplikací je také nutné sledovat inovace v technologiích a technikách vývoje, které se mohou ze dne na den rychle změnit. Pokud provozovatelé webových stránek a aplikací chtějí mít vysokou míru konverzí a klientely je nutné mít webové stránky vytvořeny moderními technologiemi s využitím nejnovějších trendů, které se neustále mění. Autor zde zmínil nejpoužívanější a nejvíce pravděpodobné trendy, techniky a technologie v současnosti a do budoucna, které web dělají webem.

Web se pomalu ale jistě dostává do fáze tzv. WEB 4.0, který by měl přinést a integrovat do webových stránek a aplikací umělou inteligenci. Díky tomu mohou firmy růst a připravit zákazníkům lepším komfort a pohodlnost nakupování a práci s aplikacemi. Je nutné se ale zamyslet nad tím, zda právě větší pohodlnost nebude spíše na škodu, nebylo by dobré, kdyby ve světě zanikla možnost komunikace s prodejcem a místo toho komunikovat s umělou inteligencí. Všechny technologie, techniky a postupný vývoj přináší své výhody, ale také mnoho nevýhod, proto programujme tak, abychom byly

spokojení a dělali webové stránky bezpečné, protože kybernetické útoky jsou stále nejrozšířenější hrozbou budoucnosti, která může postihnout jakýkoliv sektor internetu od bankovnictví po webové aplikace pro uživatele. Společnost Google na základě bezpečnostních hrozeb vydala prohlášení o používání protokolu HTTPS. Internet musí být zabezpečený a webové aplikace musí být programovány tak, aby sdílení dat neohrozilo samotné uživatele.

6 Seznam použitých zdrojů

Anon., 2017. Nette Foundation. [Online] Available at: http://nette.org/

C., D., 2006. AKAX a PHP: tvoříme interaktivní webové aplikace profesionálně. Brno: Zoner Press.

Duckett, J., 2014. Web Design with HTML, CSS, JavaScript and jQuery Set. místo neznámé:Wiley.

Flanagan, D., 2011. JavaScript: The Definitive Guide: Activate Your Web Pages. místo neznámé:O'Reilly Media.

Friedl, J. E. F., 2006. Mastering Regular Expressions. místo neznámé:O'Reilly Media.

Hansen, M. R. & Rischel, H., 2013. Functional Programming Using F#. místo neznámé:Cambridge University Press.

Hugh E. Williams, D. L., 2002. PHP a MySQL. Praha: Computer Press.

Laravel, 2017. Laravel Introduction.. [Online] Available at: http://laravel.com/docs/4.2/introduction

Meloni, J. C., 2012. Sams Teach Yourself PHP, MySQL and Apache All in One. místo neznámé:Sams Publishing.

Michaelson, G., 2011. An Introduction to Functional Programming Through Lambda Calculus. místo neznámé:autor neznámý

Nixon, R., 2014. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. místo neznámé:O'Reilly Media.

Ondřej, Č., 2009. Objektové programování. místo neznámé:Grada.

P., B., 2000. Programování WWW stránek pro úplné začátečníky. Praha: Computer Press.

Purewal, S., 2014. Learning Web App Development: Build Quickly with Proven JavaScript Techniques. místo neznámé:O'Reilly Media.

Robbins, J. N., 2012. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics. místo neznámé:O'Reilly Media.

Shen, H., 2010. Functional Architecture: An Approach for Effective Execution of Functional Programming Languages. místo neznámé:VDM Verlag Dr. Müller.

Shklar, L. & Rosen, R., 2009. Web Application Architecture: Principles, Protocols and Practices. místo neznámé:Wiley.

Snoyman, M., 2012. Developing Web Applications with Haskell and Yesod. místo neznámé:O'Reilly Media.

Software, C., 2017. Cake Software Foundation Inc.. [Online] Available at: https://cakephp.org/

Sousedé.cz, 2018. Sousedé.cz. [Online] Available at: https://www.sousede.cz/ [Přístup získán 28 1 2018].

Symfony, W. s. -., 2017. Symfony. [Online] Available at: https://symfony.com/

Technologies, Z., 2017. Zend Technologies. [Online] Available at: http://framework.zend.com/

Ullman, L., 2012. PHP Advanced and Object-Oriented Programming. místo neznámé:Peachpit Press.

WebDomu.cz, 2018. WebDomu.cz. [Online] Available at: https://webdomu.cz/ [Přístup získán 28 1 2018].

Welling, L. & Thomson, L., 2016. PHP and MySQL Web Development. místo neznámé:Addison-Wesley Professional.

Willard, W., 2013. HTML: A Beginner's Guide, Fifth Edition. 5 editor místo neznámé:McGraw-Hill Education.