Masarykova univerzita

Fakulta informatiky

Responzivní uživatelské rozhraní v kontextu moderního webového designu

Diplomová práce

Martin Peška

Brno, 2016

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: RNDr. Barbora Kozlíková, Ph.D.

Poděkování

Tímto bych rád poděkoval RNDr. Barboře Kozlíkové, Ph.D., za vedení a možnost realizace této práce. Dále bych chtěl poděkovat Ing. Michalu Cerhákovi a Ing. Liboru Leichmannovi za pomoc při tvorbě zadání této práce a za velmi přínosné konzultace.

Shrnutí

Teoretická část této diplomové práce stručně shrnuje moderní designové trendy pro tvorbu webových stránek a poté se zaměřuje na jeden z klíčových designových trendů a to responzivní design. Tato práce obsahuje poměrně obsáhlou analýzu principů responzivního designu, přehled současných metod a nástrojů pro tvorbu responzivního designu. Dále obsahuje hlubší analýzu existujících nástrojů – převážně aplikačních rámců, a jejich srovnání. Na základě této analýzy tato práce popisuje základní metodiku práce s tímto nástrojem. Zároveň zvolený nástroj je použit pro implementaci praktické části práce. Práce obsahuje také popis obecných metodik pro návrh a tvorbu responzivní webové stránky. Praktická část práce se týká návrhu vlastní *front-end* aplikace pro internetové bankovnictví. Pro implementaci byly zvoleny technologie HTML 5, CSS 3, JavaScript a aplikační rámec AngularJS. Požadavky na funkcionalitu výsledné aplikace byly konzultovány s vývojáři společnosti *Unicorn Systems* a.s. a shrnuty v textové části týkající se návrhu webové aplikace. Textová část tedy obsahuje ještě stručný popis návrhu webové aplikace, její architekturu a popis implementačně zajímavých částí webové aplikace. Hlavním cílem práce je poskytnout společnosti *Unicorn Systems* a.s. metodiky pro tvorbu responzivních uživatelských rozhraní, které by v budoucnu mohly být využity na jejich projektech.

Klíčová slova

Responzivní design, trendy moderního webového designu, fluidní mřížka, dotazy na médium, adaptivní obrázky, responzivní rozložení, front-end, internetové bankovnictví

Obsah

1 Úvod ... 1

2 Moderní web design ... 2

2.1 Vývoj webu ... 2

2.2 Designové trendy ... 4

2.2.1 *Flat* design ... 4
2.2.2 *Material* design ... 7
2.2.3 Minimalismus ... 8
2.2.4 Animace ... 9
2.2.5 Mikrointerakce ... 10
2.2.6 Responzivní design ... 10
2.2.7 *Mobile first* ... 11
2.2.8 *Modular* design ... 12
2.2.9 *Single page* webdesign ... 13
2.2.10 Large image background ... 13
2.2.11 Ostatní designové trendy ... 14

3 Responzivní design ... 17

3.1 Zařízení pro prohlížení webových stránek ... 17

3.1.1 Stolní počítače ... 17
3.1.2 Notebooky, netbooky ... 18
3.1.3 Počítačové tablety ... 18
3.1.4 Mobilní telefony ... 19
3.1.5 Ostatní zařízení ... 20

3.2 Rozložení webových stránek ... 20

3.2.1 Statické rozložení ... 20
3.2.2 Fluidní rozložení ... 22
3.2.3 Adaptivní rozložení ... 23
3.2.4 Responzivní rozložení ... 24

3.3 Principy responzivního designu ... 24

3.3.1 Fluidní mřížky ... 25
3.3.2 Fluidní typografie ... 26
3.3.3 Adaptivní obrázky ... 27
3.3.4 Dotazy na média ... 27

3.4 Návrhové vzory pro responzivní design ... 29

3.4.1 Částečně fluidní rozložení ... 29
3.4.2 Zahazování sloupců v rozložení ... 29
3.4.3 Přeuspořádání rozložení ... 30
3.4.4 Drobná vylepšení v rozložení ... 31
3.4.5 Skrývání sloupců rozložení ... 31

3.5 Návrhové vzory pro navigační nabídku ... 32

3.5.1 Navigace pro webové stránky ... 32
3.5.2 Navigace pro mobilní zařízení ... 35

3.6 Přístupy k tvorbě responzivního designu ... 37

3.6.1 Vlastní responzivní řešení ... 37
3.6.2 RESS ... 39
3.6.3 Současné nástroje pro responzivní design ... 40
3.7 Srovnání nástrojů pro responzivní design ... 42
3.7.1 Kritéria pro porovnání nástrojů ... 43
3.7.2 Metodika porovnání ... 44
3.7.3 Výsledky analýzy a experimentů ... 44
3.7.4 Závěr porovnání ... 51
4 Návrh responzivní webové stránky ... 52
4.1 Návrh responzivního uživatelského rozhraní ... 52
4.2 Tvorba responzivní webové aplikace ... 54
4.3 Základy práce s Twitter Bootstrap ... 55
4.3.1 Tvorba responzivní mřížky ... 55
4.3.2 Fluidní typografie ... 58
4.3.3 Práce s obrázky ... 58
4.3.4 Responzivní komponenty ... 59
5 Analýza webové aplikace ... 62
5.1 Analýza problému ... 62
5.2 Analýza požadavků ... 62
5.3 Analýza případů použití ... 63
6 Implementace webové aplikace ... 67
6.1 Použité nástroje ... 67
6.1.1 HTML, JavaScript a JSON ... 67
6.1.2 AngularJS ... 67
6.1.3 Twitter Bootstrap ... 68
6.1.4 NPM ... 68
6.1.5 LESS ... 68
6.1.6 Browserify ... 68
6.1.7 Grunt ... 69
6.2 Architektura informačních systémů ... 69
6.3 Architektura navržené webové aplikace ... 70
6.4 Implementace ... 72
6.4.1 Instalace a integrace Twitter Bootstrap ... 72
6.4.2 Konfigurace Grunt ... 73
6.4.3 Routing a navigace ... 74
6.4.4 Komunikace klient – server ... 75
6.4.5 Grafy a responzivní komponenty ... 76
6.4.6 Responzivita aplikace ... 79
6.4.7 Lokalizace ... 81
6.5 Testování ... 82
7 Diskuze ... 83

8 Závěr 84
9 Literatura 85

1 Úvod

Od svého počátku se webové prostředí velmi dynamicky mění a s ním i užívané metodiky, technologie a v neposlední řadě i webový design. Webový design podléhá zejména trendům, které se velmi často mění a mají zásadní vliv na design webových aplikací. V posledních letech se objevilo několik trendů, jejichž změny mají pro webdesign a design uživatelských rozhraní až téměř revoluční dopad. Největší roli v těchto změnách hrají společ-nosti Google a Apple, které již delší dobu udávají směr pro vývoj webdesignu. Jedním z nejzásadnějších milníků byl příchod responzivního designu, který umožňuje zobrazení webových stránek na různých typech zařízení. Od jeho představení v roce 2009 [18] se vyvinula spousta technik, metodik a nástrojů pro tvorbu responzivních webových stránek.

Cílem této práce je představit principy a metodiky responzivního designu a metodiky návrhu a tvorby responzivního uživatelského rozhraní tak, aby na základě provedené analýzy mohla z výsledků této práce čerpat společnost *Unicorn Systems* při vývoji velkých podnikových systémů s responzivní *front-end* částí systému. Dalším požadavkem je uvedení současných technik a nástrojů pro vývoj responzivních a jejich srovnání. Na základě tohoto srovnání bude zvolen vhodný nástroj a s jeho pomocí implementována *front-end* aplikace internetového bankovnictví. Zadavatel zvolil pro implementaci technologie HTML 5, CSS3, JavaScript a aplikační rámec AngularJS. Pro tvorbu je užití prvních tří technologií dnes již standardem. AngularJS byl zvolen, protože společnost *Unicorn Systems* zejména v Brně preferuje použití tohoto aplikačního rámce při vývoji svých projektů.

Úvodní kapitola práce chronologicky popisuje vývoj webových stránek a dále obsahuje přehled významných designových trendů posledních let. Další kapitola představuje zařízení, na kterých je v současnosti možné zobrazit webové stránky a podrobně popisuje principy responzivního designu a responzivních webových stránek. Dále obsahuje analýzu současných nástrojů a technik pro tvorbu responzivního designu a jejich srovnání. Čtvrtá kapitola je věnována detailnějšímu popisu metod pro návrh a tvorbu responzivních uživatelských rozhraní a následně stručně představuje práci s vybraným nástrojem, který bude použit pro implementaci praktické části této práce. Pátá kapitola stručně shrnuje požadavky ze strany společnosti *Unicorn Systems* na praktickou část práce a shrnuje také stručnou analýzu internetového bankovnictví a případů užití. Šestá kapitola popisuje implementaci. Popisuje užité nástroje a architekturu aplikace a poté představuje některé zajímavé části implementace. Poslední část této kapitoly se věnuje testování aplikace. Poslední kapitola se věnuje diskuzi užitých technologií a případných vylepšení.

2 Moderní web design

Tato kapitola obsahuje chronologický přehled vývoje webových stránek za posledních přibližně třicet let a poté budou popsány konkrétní významné designové trendy posledních let.

2.1 Vývoj webu

Jak již bylo výše zmíněno, webový design nyní prochází velice bouřlivým obdobím díky technologickému pokroku. V roce 1991 byla vytvořena první webová stránka, kterou publikoval Tim Berners-Lee. Stránka obsahovala pouze statický textový obsah, který byl zarovnán v jednom sloupci. Jednotlivé obsahy stránek byly propojeny pomocí odkazů [14]. Pro popis struktury stránky vznikl nový značkovací jazyk HTML. HTML (*HyperText Markup Language*) je značkovací programovací jazyk, který slouží pro tvorbu statického obsahu a struktury webových stránek [1].

V druhé polovině devadesátých let se začala struktura webu měnit, přešlo se k používání tabulek, jejichž pomocí bylo možné webovou stránku rozdělit na velké množství sloupců. Tímto způsobem bylo možné vizuálně oddělovat statický obsah stránky. Dále se v tomto období objevují první obrázky na pozadí, animované texty, tehdy velice oblíbené rámce a také byly velice oblíbená počítadla návštěvnosti webu.

Koncem devadesátých let vznikly tzv. kaskádové styly, které revolučně změnily vzhled webových stránek postupně až do dnešní podoby. Kaskádové styly (CSS – *Cascade Style Sheet*) jsou stylovacím jazykem popisujícím vzhled webové stránky napsaném v jazycích HTML, XHTML nebo XML [4]. Pomocí jednoduchých selektorů na prvky webové stránky (např. elementy, atributy, třídy, identifikátory) nastavují jejich grafické vlastnosti.

Současně s kaskádovými styly vznikl jako jejich alternativa Macromedia Flash. Macromedia Flash (dnes Adobe Flash), zkráceně Flash, je grafický vektorový program vydaný v roce 1996, který zpočátku poskytoval animační nástroje a byl volitelným plug-inem webových prohlížečů. Od té doby získal spoustu nových rysů a stal se standardním plug-inem moderních webových prohlížečů [3]. Na přelomu tisíciletí dosahoval Flash svého vrcholu a vývojáři byl používán ve většině tehdejších webových stránek a to v kombinaci s tabulkovou strukturou. Mezi hlavní rysy webových stránek tohoto období patří vytvořené animace, hry, ale také animovaná navigační menu naprogramovaná převážně pomocí Macromedia Flash.

Na konci devadesátých let se poprvé objevuje jazyk PHP, který získal velkou popularitu mezi vývojáři. PHP (*Hypertext preprocessor*, původně *Personal Home Page*) je populární skriptovací programovací jazyk, který je určen k vývoji webových aplikací. Je rychlý flexibilní,

pragmatický a s jeho pomocí se vytváří vše od blogu po nejpopulárnější webové stránky [5].

Kolem roku 2000 započal vzestup kaskádových stylů. Velký podíl na tomto vzestupu měla společnost Microsoft, která ve svém prohlížeči Internet Explorer nabízela devadesáti devíti procentní podporou kaskádových stylů. Bylo tedy možné oddělit statickou strukturu a obsah webu od webového designu, taktéž použití kaskádových stylů výrazně zjednodušilo tvorbu vzhledu webové stránky.

V první polovině první dekády 21. století byl na vzestupu jazyk JavaScript, což je vysokoúrovňový programovací jazyk užívaný pro tvorbu dynamického obsahu webové stránky a stal se jednou ze základních technologií pro vývoj současného webu [2]. Pomocí jazyka JavaScript bylo možné nahradit Macromedia Flash, což mělo pozitivní dopad na rychlost webových stránek a také měl jazyk JavaScript širší využití pro tvorbu dynamických webových stránek. Mezi hlavní rysy webových stránek v tomto období patří komponenta tzv. *dropdown* menu (v jazyce HTML element <select></select>), dále se navigace přemístila do horní části stránky a začaly využívat webové formuláře, které jsou považovány za jeden z nejdůležitějších rysů jazyka HTML.

Přibližně v polovině první dekády 21. století došlo k výraznějšímu posunu webového designu, který cílil na strojové porozumění obsahu webové stránky tak, aby stroj byl schopen stránce stejně porozumět jako člověk, čehož se vývojáři snažili využít pro vyhledávání webových stránek pomocí webových vyhledávačů, jako např. vyhledávače Google. V tomto období se začalo více hovořit o UI designu webových stránek a to z toho důvodu, že webové stránky navštěvovalo čím dál vyšší množství uživatelů a bylo třeba zajistit, aby stránky byly dostatečně srozumitelné a intuitivní pro co nejširší skupinu uživatelů. UI design je design se zaměřením na potřeby uživatele a ujištěním se, že elementy rozhraní jsou dostupné, srozumitelné a použitelné pro daný účel [7]. Z pohledu využívaných webových technologií se nadále pokračovalo v již využívaných technologiích a vzrůstaly tendence ke zjednodušování vývoje stránek.

Ve druhé polovině první dekády však již došlo k dramatickým změnám ve webovém designu. Hlavní důraz byl kladen na interakci s uživatelem. Za tímto účelem vznikl AJAX (*Asynchronous* JavaScript *And* XML), což je knihovna sloužící k asynchronní komunikaci mezi klientem a serverem [6]. Hlavním rysem tohoto období byla interakce a hlavně interaktivní změna dat bez nutnosti opětovného načtení webové stránky. Webové stránky tak již neměli pouze informační funkci, ale navíc sloužily k prezentaci, obchodu, komunikaci pomocí chatu či sociálních sítí.

Od první poloviny druhé dekády 21. století lze webový design charakterizovat snahou zjednodušit vzhled webové stránky, hlavně kvůli její přehlednosti a přizpůsobení se mobilním zařízením různých velikostí (mobilní telefony, tablety). Webové stránky jsou převážně úzkého profilu, ale svým obsahem se nebojí narůstat do výšky. Bylo provedeno několik studií na téma týkající se scrollování obsahu či případného přepnutí do jiné sekce s dalším

obsahem. Výsledkem bylo, že drtivá většina uživatelů upřednostňovala scrollování, protože je pro ně pohodlnější a o to víc pohodlnější na mobilních zařízeních ovládaných intuitivními gesty pro vertikální posun. Dále je zde snaha minimalizovat počet ovládacích prvků na stránce, stejně tak i minimalizovat zobrazený obsah. S rostoucím počtem uživatelů mobilních zařízení lze tedy očekávat, že potřeba přizpůsobit webovou stránku pro mobilní zařízení bude narůstat a proto bude růst důležitost responzivního webového designu. Koncem roku 2016 došlo k mimořádné události, která potvrzuje důležitost responzivního webu (webu přizpůsobujícímu se různým zařízením). Počet mobilních zařízení (mobilní telefony a tablety) připojeným k Internetu již převýšil počet klasických stolních počítačů a notebooků.

2.2 Designové trendy

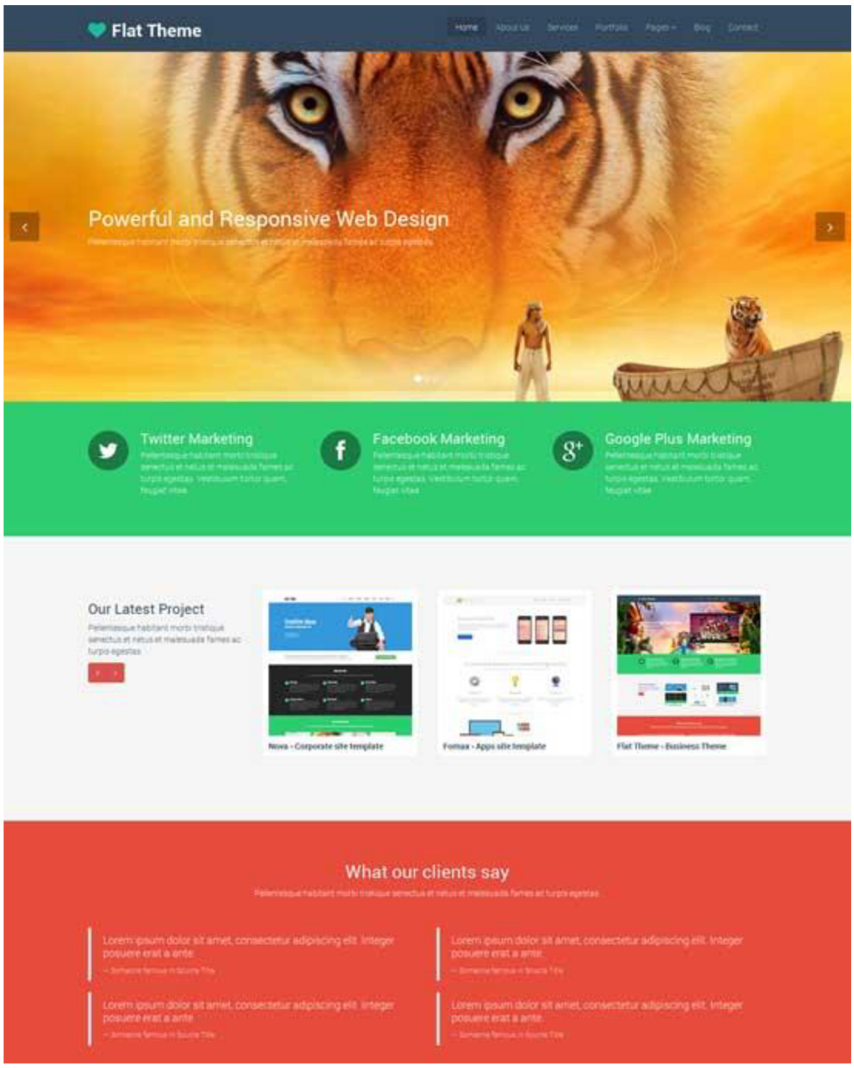
V přecházející kapitole bylo zmíněno, že v posledních letech webový design prochází velkými změnami, převážně skrze snahu o přizpůsobení webu pro mobilní zařízení. Současný web design tedy obecně cílí na zjednodušení struktury webu, minimalizaci obsahu, snadno dostupné a smysluplné ovládací prvky a také v neposlední řadě na uživatelský prožitek (angl. *user experience*) a použitelnost. Při tvorbě takového webu můžeme narazit na velké množství nejrůznějších konceptů webového designu mající společné charakteristické rysy, které utvářejí designový styl. Pokud vezmeme v úvahu časové období, kdy je designový styl populární, lze ho označit jako designový trend v uvažovaném období. Tato kapitola stručně popisuje základní designové trendy, které jsou charakteristické pro období přibližně od roku 2010 až do současnosti. Níže se nachází stručný souhrn designových trendů s jejich charakteristickými rysy a demonstrativními ukázkami.

2.2.1 *Flat* design

Flat design je jedním z minimalistických designů uživatelských rozhraní, běžně využívaných ve webových nebo mobilních aplikacích. *Flat* design se vyznačuje minimálním užitím stylistických elementů, které dávají pocit trojrozměrného zobrazení, snaží se zobrazit minimální množství informací a používat minimalistické jednoduché ovládací komponenty, minimalistickou typografii a tzv. *flat colors*, což je označení pro barvy, které nepoužívají žádné stínování, zvýrazňování ani jiné modifikace – jde pouze o základní barvy [14].

Z pohledu implementace webové stránky ve *flat* designu je velice snadné nadefinovat základní webové komponenty, protože jsou vizuálně velice jednoduché a minimalističnost designu napomáhá snadné aplikaci moderních aplikačních rámců pro responzivní uživatelská rozhraní.

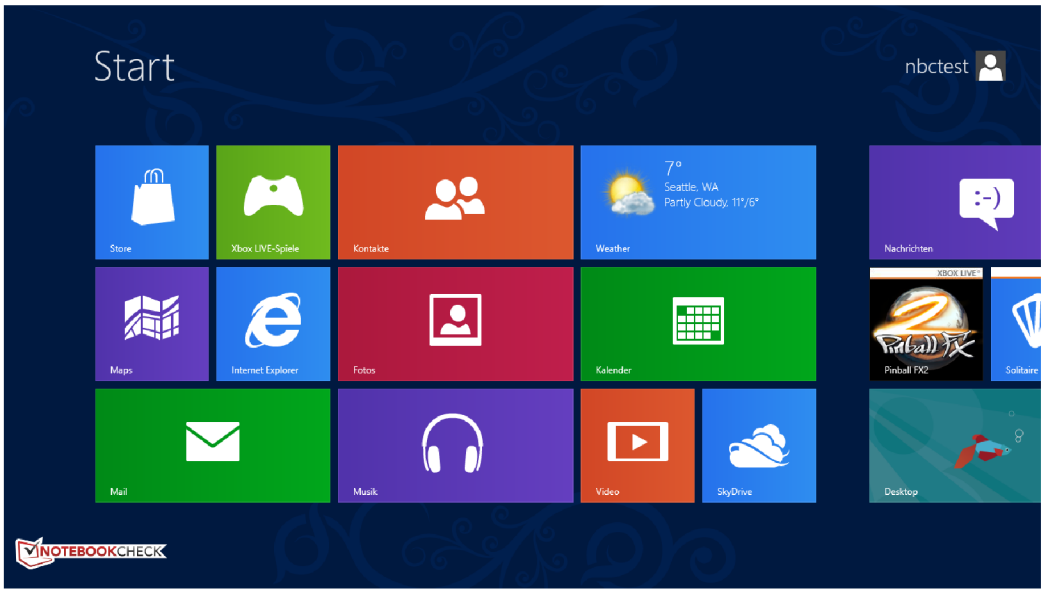
Na obrázku 1 je zobrazena šablona, která je designovaná pomocí *flat* designu. Je zde vidět použití základních barev, minimální použití stylistických elementů jako stínování, gradientů, či textur, horní menu je velmi jednoduché a obsahuje pouze nezbytné odkazy. *Slider* komponenta s obrázkem s tygřím motivem má za úkol upoutat uživatelovu pozornost. Jednotlivé sekce jsou vertikálně uspořádané pod sebou a jsou barevně odlišené. Celkově šablona působí minimalistickým dojmem a právě díky vertikálnímu uspořádání bude uživateli zobrazeno jen nezbytné minimum informací.



Obrázek 1: *Flat* design – šablona pro tvorbu webové aplikace. Převzato z [50].

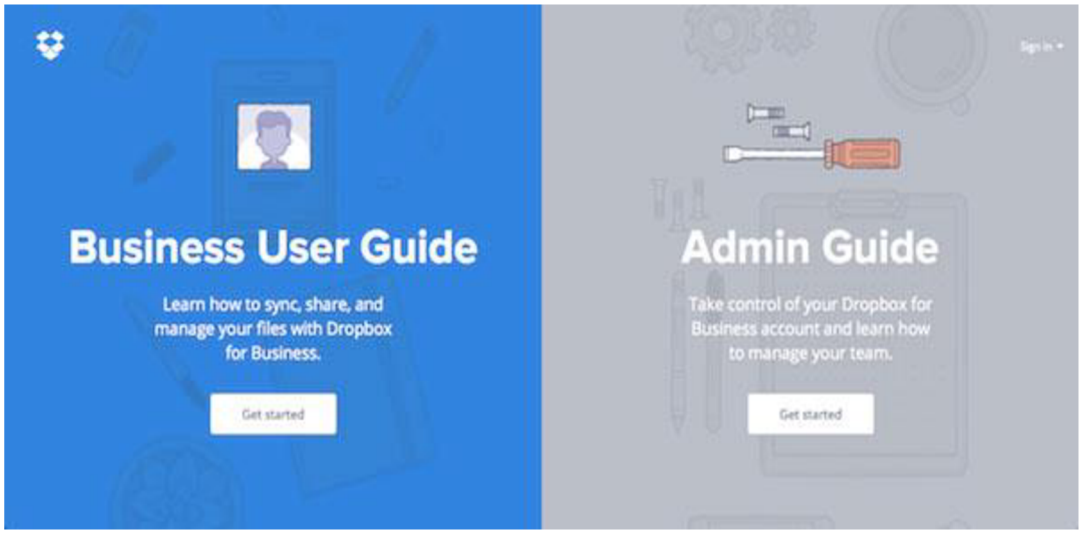
Flat design byl vytvořen již kolem roku 2002, kdy jeho prvky využívala společnost Microsoft ve svých aplikacích. V roce 2010 Microsoft vydal Windows Phone 7, což byl revoluční operační systém pro mobilní telefony, jehož uživatelské rozhraní bylo navrženo právě ve *flat* designu a zaznamenalo velký úspěch. Microsoft pomocí *flat* designu definoval svůj nový designový jazyk zvaný Metro (viz obrázek 2). Úspěch tohoto designového stylu nabádal k jeho využití pro webové aplikace, protože v tomto období se tvůrci webových aplikací snažili přizpůsobit své webové aplikace pro mobilní zařízení a *flat* design, který již byl apli-

kován pro mobilní platformy, byl skvělou volbou.



Obrázek 2: Designový jazyk Metro užívající *Flat* design.

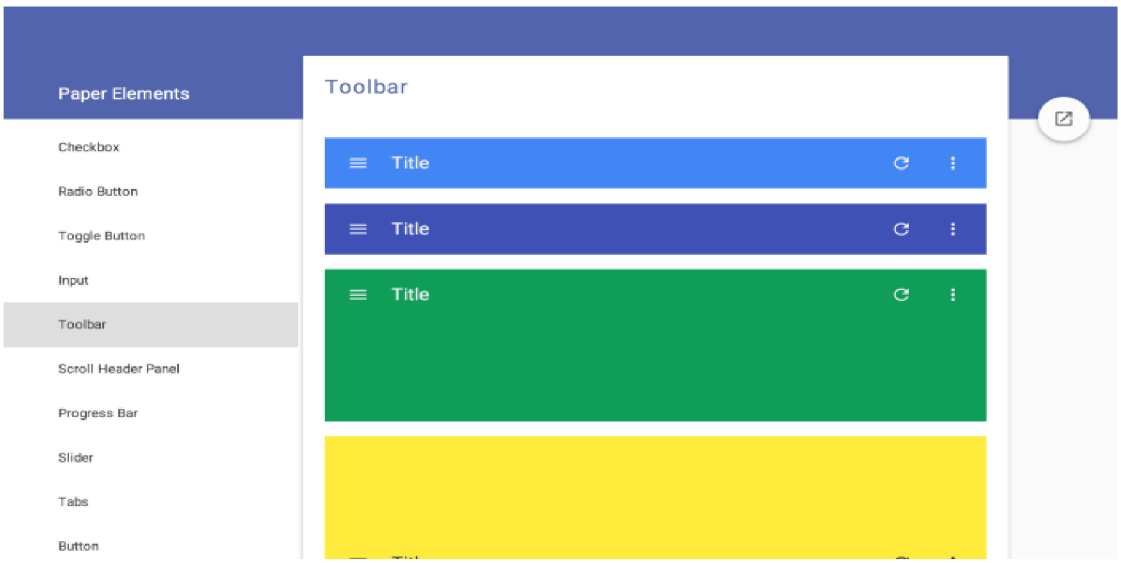
Flat design se od doby, kdy byl představen, neustále vyvíjí a časem se z něj vyvinuly i jiné designové trendy, např. „*Material* design“. Ale také *flat* design samotný má svoje vlastní vývojové proudy. Příkladem může být tzv. „*Flat* design 2.0“ [51], který tak byl pojmenován významným americkým designérem Ryanem Allenem. *Flat* design 2.0 se vyznačuje podobnými rysy, které byly výše popsány v popisu rysů *flat* designu. Avšak od klasického *flat* designu se liší především v tom, že není tak striktní ve využívání stylistických elementů jako stínování komponent, texturování nebo užití gradientů. Příkladem *flat* designu 2.0 může být webový průvodce aplikací Dropbox, viz obrázek níže.



Obrázek 3: *Flat* design 2.0 – webová stránka pro uživatelskou příručku aplikace Dropbox. Převzato z [51].

2.2.2 *Material* design

Material design je dalším zástupcem z celé řady minimalistických designových stylů, který byl inspirován *flat* designem. Od *flat* designu se liší v mnoha aspektech. *Material* design společně s *flat* designem používá mřížkový layout, avšak není tak striktní s užíváním animací, posunů, odsazování či hloubkových efektů. *Material* design si klade za cíl přivodit dojem práce obecně s materiály, snaží se lehce přiblížit k realistickému zobrazení avšak stále je dosti abstraktní, pohyb a animace na stránce musejí mít nějaký význam a nutí designéra dodržovat některé základní principy, které jsou dány vlastnostmi materiálů. Základem je tedy materiál a materiálem je míněna každá viditelná komponenta na obrazovce, která má následující vlastnosti: je pevná; zabírá unikátní body v prostoru; neproniknutelná; má proměnlivý tvar; může měnit velikost, ale pouze podél své roviny; neohebná; může se spojit s více materiály, může být vytvořena či odstraněna; může být rozdělena, oddělena nebo spojena, může se pohybovat podél kterékoliv osy. Z této obecné definice vlastností materiálů lze vytvářet velmi zajímavé efekty: např. překrývání komponent, interakce s celou komponentou s využitím animace (např. stlačení dlaždice), kolize komponent atd. [15].



Obrázek 4: Ukázka webové stránky s prvky material designu. Převzato z [15].

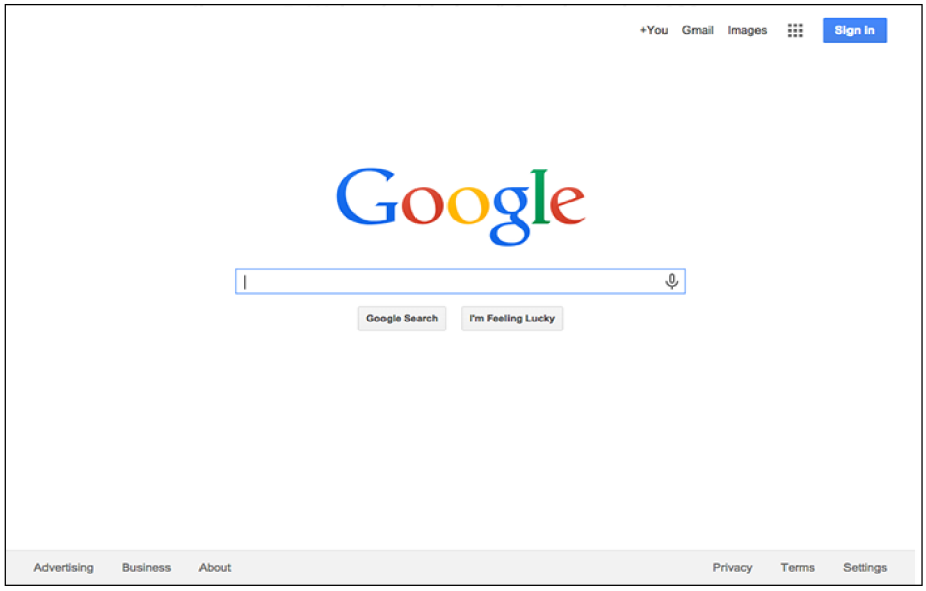
Na vzniku *material* designu se podílela hlavně společnost Google, která v roce 2014 představila *material* design jako svůj designový jazyk. Z pohledu implementace je materiál design opět velice snadný pro implementaci webové stránky – opět se zde nacházejí jednoduché webové komponenty, a taktéž implementace responzivního uživatelského rozhraní je poměrně snadná díky mřížkové struktuře stránek. Nevýhodou však mohou být četnější animace a překrývání jednotlivých komponent, jež mohou mít vliv na výkon a responzivitu

webové stránky. Pokud webový vývojář bude s animacemi zacházet opatrně a překrýváním neporuší responzivní mřížku, pak tyto nevýhody eliminuje. Výše se nachází na obrázku 4 ukázka webové stránky designované pomocí *material* designu.

2.2.3 Minimalismus

Minimalismus jako takový je možné také považovat za designový trend, který má obrovský vliv na ostatní trendy moderního webového designu. Jeho obecným cílem je zjednodušit webovou stránku na nejjednodušší elementy, tím pádem na stránce zůstanou nejpodstatnější informace a nejdůležitější ovládací komponenty. Tento trend se drží pravidla: „Méně je více“. To znamená, že mimo minimalizaci elementů na obrazovce se snaží i odstranit vše, co by mohlo uživatele rozptylovat od obsahu. Minimalismus taktéž využívá jednoduché stylování webové stránky, kde většinou bývá využito bílé pozadí, jemné linky pro oddělení sekcí a typografie, která zaujme uživatele [13]. Velmi často také bývá kombinován s jinými trendy, jako např. *Flat* designem. Mezi další rysy patří jednoduchá navigace, obvykle v horní části obrazovky, symetrie, kontrast. Při tvorbě minimalistické webové stránky je dobré udržovat obsah stránky konzistentní a vyvážený, vytvářet vizuálně vztahy mezi komponentami, efektivně zobrazovat obsah, využít mřížkový layout, občas porušit zavedená pravidla (např. převrácená část textu) nebo výjimečně použít jednoduché textury. Mezi časté chyby při tvorbě minimalistické webové stránky patří: příliš mnoho zobrazovaného obsahu, zobrazování reklam na stránce, stránky cílené pro děti a mladé lidi by se měli minimalismu vyhýbat, protože mladým lidem tento trend připadá příliš nudný a jednotvárný [14].

Minimalismus jako designový styl se objevil již v 60. letech 20. století, kdy získal na velké popularitě. Je taktéž spojován se světoznámou uměleckou školou *Bauhaus*, která působila v Německu v předválečném období. V mnoha dílech této školy, zejména geometrických abstrakcích, se objevují prvky minimalismu. V 80. letech až do roku 2000 se začal projevovat i v oblasti informačních technologií, kde jeho prvky byly aplikovány pro interakci člověka s počítačem. Kolem roku 2000 pronikl minimalismus i do oblasti web designu. Jedna z prvních minimalistických webových stránek byla stránka webového vyhledavače Google. Od roku 2000 se minimalismus stával čím dál více oblíbeným designovým trendem a dnes se těší obrovské popularitě [16]. Na obrázku 5 je ukázka minimalistického designu webového prohlížeče Google.



Obrázek 5: Webový vyhledavač Google z roku 2015, minimalismus. Převzato z [14].

2.2.4 Animace

Jedním z neustále se vracejících trendů ve webovém designu je využívání animací. Jejich použití na stránce udělá stránku vizuálně interaktivnější a pro uživatele zajímavější. Pomocí animací lze uživatelům více přiblížit jejich obsah a lze tak ovlivňovat uživatelovu pozornost a do jisté míry ji i řídit. Animace nelze vkládat libovolně po celé webové stránce. Při jejich velkém počtu mají velice rušivý efekt nebo mohou být špatně interpretovány uživatelem. Webový designér tak musí pečlivě zvážit jejich použití na webové stránce. Animace ve webovém designu se zpravidla dělí na dva druhy podle rozsahu své působnosti na tzv. „*Large scale*“ animace a „*small scale*“ animace. *Large scale* animace mají výrazný dopad na interakci s uživatelem, do této kategorie patří například otevírání modálních oken, pop-up okna, *slideshow*, navigace a menu. Do *small scale* animací se řadí drobnější animace jako zobrazování pomocných zpráv, animace načítání [14]. Na obrázku 6 se nachází ukázka animované načítací obrazovky.

Obrázek 6: Ukázka animace načítací obrazovky.

2.2.5 Mikrointerakce

Mikrointerakce jsou dalším z velice populárních trendů užívaných ve webovém designu. Mikrointerakce nás na webu neustále provázejí, např. při vypínání budíku v mobilním telefonu nebo při ohodnocení příspěvku na sociální síti. Jsou to drobnosti, které každý uživatel běžně dělá a nevyžadují až tak velkou pozornost. Pokud ale designér navrhne vyšší množství mikrointerakcí, docílí tak vyšší interakce s uživatelem a uživatel tak nevědomě může používat aplikaci více, než by původně kdy chtěl. Mikrointerakce obecně pomáhají uživateli manipulovat s aplikací, zobrazují výsledky akcí uživatele nebo komunikují s uživatelem a vyžadují od něj nějakou odezvu. Při tvorbě webové stránky užívajícím interakce je dobré zajistit, aby mikrointerakce zůstaly skryté a také, aby jich najednou nebylo mnoho, aby příliš nezatěžovaly uživatele. Správně navržené mikrointerakce vedou pak k tzv. *Human-centred* designu, který si klade za cíl aplikaci maximálně přizpůsobit potřebám člověka a to jak s ohledem na funkcionalitu, tak i z ohledu jednoduchosti uživatelského rozhraní a jeho použitelnosti.

2.2.6 Responzivní design

V posledních pěti letech se stal responzivní design nesmírně populární hlavně díky tomu, že se uživatelé internetu připojují k webu z mobilních zařízení jako např. chytré telefony, tablety aj. Kvůli stále se zvyšujícímu počtu uživatelů internetu na mobilních zařízeních bylo třeba zavést techniky pro přizpůsobení webu pro dané zařízení. Jednou z těchto technik je právě responzivní design. Responzivní design se snaží rozdělit velikost zařízení podle jejich velikostí do určitých skupin a pokud dané mobilní zařízení spadá do oné velikostní skupiny, tak se mu webová stránka přizpůsobí na předem definovanou velikost pro danou skupinu.

Průkopníkem v responzivním webovém designu je společnost Audi, která měla již v roce 2001 responzivní webové stránky, které se přizpůsobovaly velikosti obrazovky uživatele. Mnohem později v roce 2009 Ethan Marcotte poprvé použil termín responzivní webový design a definoval jej jako trojici parametrů responzivního webu a to fluidní mřížky, adaptabilních obrázků a dotazů na médium. Od roku 2009 je responzivní design na vzestupu a dnes se stává již zavedeným standardem při vývoji webu [8]. Podrobně se těmto termínům a responzivnímu designu bude věnovat třetí kapitola. Na obrázku 7 je zobrazeno schematické rozložení webové stránky na různých zařízeních.

Obrázek 7: Ilustrace funkce responzivního webu pro různá zařízení.

2.2.7 *Mobile first*

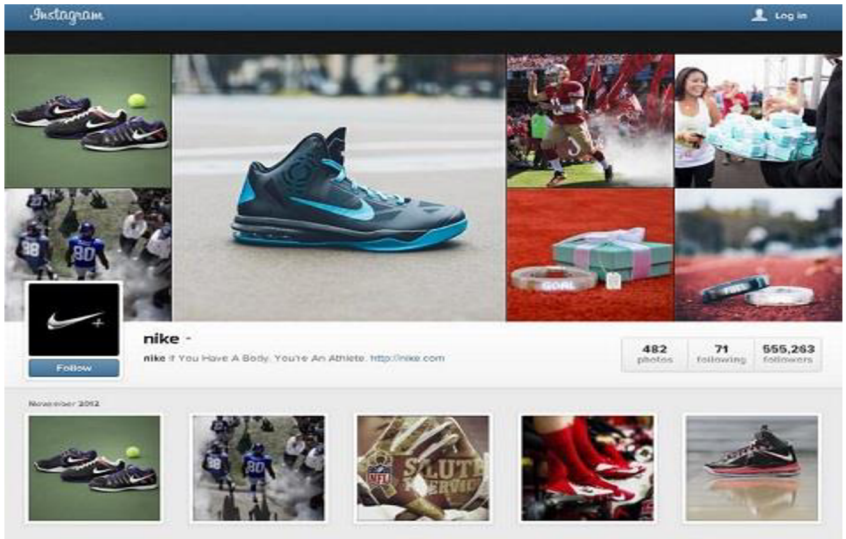
Mobile first je jedním z přístupů vývoje responzivního uživatelského rozhraní. A lze jej také považovat za moderní trend ve webovém designu. V současnosti je velice populární a doporučovaný velkým množstvím odborníků na responzivní uživatelská rozhraní. Je to velice efektivní technika vývoje. Spočívá v tom, že prvotní návrh webové aplikace je navržen pro nejmenší mobilní zařízení, kde webový designér musí zvolit nejnutnější informace a komponenty, které se uživateli zobrazí. To vede webové designéry často k využití minimalismu a z něj odvozených designových stylů. Jakmile je prvotní návrh pro mobilní zařízení hotov, designér se věnuje větším zobrazení aplikace na větších zařízeních. Pro větší zařízení může přidat nové webové komponenty, případně může zobrazit více obsahu pro větší zařízení. Tento přístup má obrovskou výhodu v tom, že vývojář nemusí stránku – resp. její strukturu neustále upravovat pro různá zařízení – v prvotním návrhu vytvoří minimální funkcionalitu, obsah a sestaví fluidní mřížku pro mobilní zařízení, totéž potom opakuje pro větší zařízení a tím pokryje větší množinu zobrazitelných komponent a informací a zároveň nedochází k neustálým úpravám struktury webu. Problémem však může být přesně opačný přístup, kdy se vývojář snaží o úpravu velké stránky pro malé zařízení. V tomto

případě je přizpůsobení webu velice časově náročné a z pohledu řízení projektů velmi drahé. V rámci přizpůsobování webu je třeba rozhodovat, které webové komponenty se ve fluidní mřížce přemístí, které budou skryty a vizualizovány a tak je struktura webu neustále upravována.

2.2.8 *Modular* design

Modulární design je designový trend, který je definován pomocí mřížkového layoutu. Rozděluje systém na řadu samostatných celků – modulů, které jsou zcela nezávislé na systému a mohou být znovupoužity v jiných systémech. Tyto moduly jsou pak zasazeny do mřížkového layoutu. Samostatné moduly musejí poskytovat rozhraní pro komunikaci se systémem, což je dnes průmyslovým standardem [9]. Modulární design se dnes využívá často ve spojení s responzivním designem. V tomto případě je obvykle využita fluidní základní mřížka a jednotlivé moduly jsou vkládány do této mřížky. Mřížka samotná poté zajistí, aby byla taková modulární webová stránka responzivní. Pro modulární design se taktéž používá jako tzv. *card layout*, což je mřížka obsahující karty s obsahem. Opět lze pro *card layout* využít fluidní mřížku zajišťující responzivitu webové stránky. Prakticky se modulární design a *card layout* využívá hlavně při tvorbě internetových obchodů, kde na jednotlivých kartách bývá obvykle uvedeno zboží a jeho detailní informace.

Mřížka a modulární design se poprvé objevily v roce 2007 a dodnes se těší velké popularitě. Tento přístup využívají i některé sociální sítě, jako např. Instagram (viz obrázek 8). Ale také tento přístup se používá v architektuře frontendové části velkých podnikových aplikací, kdy se aplikace snaží rozdělit na logické celky – moduly a menší komponenty a z nich se sestaví konečná podoba webové stránky.



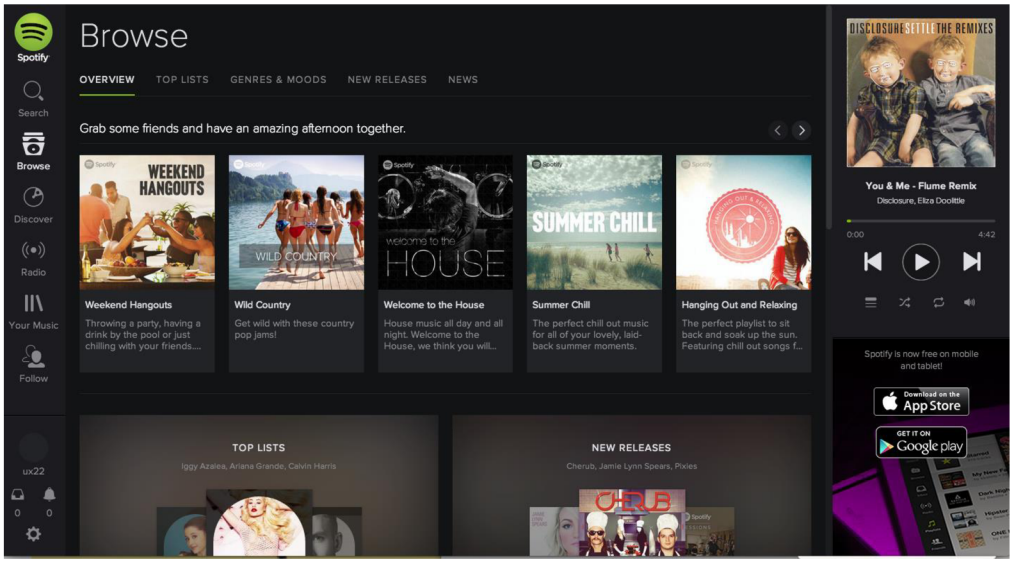
Obrázek 8: Modulární design sociální sítě Instagram.

2.2.9 *Single page* webdesign

V současnosti zažívá webový vývoj bouřlivé období, hlavně díky rychlému vývoji front-end technologií, zejména v posledních několika letech jsou velice oblíbené aplikační rámce vytvořené v jazyce JavaScript (AngularJS, ReactJS). Tyto aplikační rámce mají společnou vlastnost a to, že se pomocí nich snadno vytváří tzv. *single page* aplikace. *Single page* aplikace je webová aplikace nebo webová stránka zobrazující pouze jednu webovou stránku s cílem poskytnout podobný uživatelský prožitek jako desktopová aplikace [11].

Single page design je designový styl úzce spjatý s vývojem *single page* aplikací, jehož cílem je navrhnout konzistentní uživatelské rozhraní pro dynamicky se měnící obsah webové stránky. Dynamicky se měnícím obsahem je zde míněna především různorodost dat pro vizualizaci během interakce s uživatelem. V praxi se tento designový styl často kombinuje s ostatními, v současnosti především s materiálním a responzivním designem. Na obrázku 9 je uvedena ukázka *single page* aplikace

Z pohledu implementace tohoto designu je v počátečních fázích projektu třeba navrhnout konzistentní základ uživatelského rozhraní, který se vyznačuje definovanou typografií, barvami a poté se přejde k tvorbě konzistentních komponent, s jejichž pomocí se sestavuje výsledná webová stránka.



Obrázek 9: Ukázka *single page* designu webového přehravače hudební služby Spotify.

2.2.10 Large image background

Jedním z velice pozoruhodných designových trendů je trend užívající velké obrázky ve vysokém rozlišení na pozadí webové stránky. Přesněji se jedná o fotografie, které vhodně

doplňují obsahovou část webové stránky. Fotografie na pozadí je obvykle překryta bílou průhlednou vrstvou, aby nebyla příliš rušivá a aby vizuálně nezanikal textový obsah webové stránky. Hlavním posláním tohoto stylu je pokus o ilustraci zprávy, kterou chceme uživateli sdělit. Člověk vnímá nejvíce informací vizuálně a tento styl se snaží maximalizovat uživatelský prožitek během návštěvy takto navrhnuté webové stránky [12]. Spolu s tímto stylem se využívá minimalismus a to zejména z toho důvodu, že je třeba minimalizovat obsah pouze na nejnutnější sdělení a nejnutnější ovládací prvky tak, aby uživatel zároveň vnímal fotografii a obsah.

Tento designový styl je poměrně snadný pro implementaci, avšak je třeba mít na paměti technologická omezení prohlížečů. Není vhodné užívat příliš velké obrázky, protože stránka může načítat velice dlouhou dobu. Webovou stránku obsahující velký obrázek na pozadí není složité udělat responzivní. Jen je třeba předpřipravit různé velikosti obrázku na pozadí pro různá zařízení či rozlišení a poté pomocí tzv. dotazů na média měnit obrázky v závislosti na zařízení nebo pomocí elementu <picture> definovat obrázek pro různá zařízení.

Obrázek 10: Ukázka webové stránky užívající velký obrázek na pozadí. Převzato z [10].

2.2.11 Ostatní designové trendy

Za zmínku ještě stojí několik designových trendů, které jsou velice blízké k výše uvedeným trendům nebo jsou z nich přímo odvozeny. Prvním takovým zástupcem je design využívající video namísto obrázku na pozadí. Pro tento design platí velmi podobná pravidla jako pro obrázky. Video by mělo být součástí sdělení uživateli a mělo by vhodně doplňovat

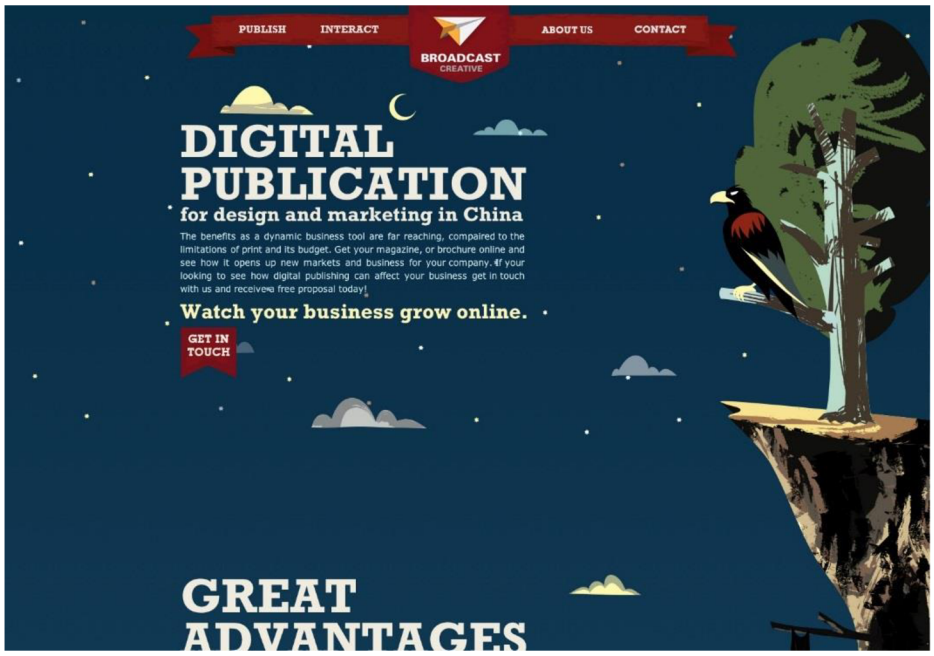
textovou část, která by zde měla být minimální. Cílem tohoto stylu je co nejvíce zaujmout uživatele a efektivně mu sdělit danou informaci. Na obrázku 11 je ukázka tohoto designu.

Obrázek 11: Ukázka webové stránky užívající video na pozadí. Převzato z [10].

Dalším velice zajímavým designovým trendem je design webové stránky, který připomíná design mobilní aplikace. Tento design stránek je vhodný pro mobilní zařízení (mobilní chytré telefony, tablety) a to z důvodu své jednoduchosti a velikosti ovládacích elementů, které je možné ovládat pomocí dotykového rozhraní. Velice úzce souvisí s responzivním uživatelským rozhraním a i ostatními výše popsanými designovými trendy, pomocí nichž může být vytvořen. Na obrázku 12 je zobrazena ukázka tohoto designu.

Obrázek 12: Ukázka webové stránky v designu připomínajícím mobilní aplikaci. Převzato z [10].

Posledním uvedeným designovým trendem nebo předněji efektem, který se nyní velmi často užívá ve webovém uživatelském rozhraní, je tzv. „p*arallax scrolling*“. *Parallax scrolling* je designový trend webových stránek, kde se obsah webové stránky na pozadí (např. obrázek) posouvá jinou rychlostí při vertikálním posunu stránky než obsah stránky v popředí. [17] Na obrázku 13 je zobrazena ukázka webové stránky užívající tento efekt. U tohoto stylu je třeba mít na vědomí, že nefunguje správně na mobilních zařízeních a je třeba tento efekt pro tato zařízení vypnout.



Obrázek 13: Ukázka webové stránky využívající efekt *parallax scrolling*.

3 Responzivní design

Tato kapitola se bude hlouběji zabývat responzivním designem, budou zde představeny hlavní pilíře, na kterých responzivní design staví a popsány principy, pomocí kterých lze vytvářet responzivní webové stránky. Než budou níže vysvětleny principy samotného responzivního web designu, nejprve je nutné si definovat zařízení, pomocí kterých lze dnes prohlížet webové stránky. Dále je třeba definovat responzivní design v kontextu rozložení webové stránky, aby byly patrné rozdíly, které responzivní uživatelské rozhraní přináší pro uživatele webových stránek.

3.1 Zařízení pro prohlížení webových stránek

Pro pohlížení webových stránek na Internetu stačí v současnosti mít zařízení, které podporuje webové prohlížeče a umožnuje samotné připojení k této síti. Těchto zařízení existuje celá řada a každé z nich má svá specifika, co se týče zobrazení webové stránky. Tato podkapitola shrne základní typy zařízení, pomocí nichž lze prohlížet webové stránky a stručně charakterizuje jejich vlastnosti.

3.1.1 Stolní počítače

Stolní počítač je počítač, který je navržen tak, aby zůstal v jedné lokaci. Jedná se buď o systémovou jednotku nebo *all-in-one* počítače jakým je kupříkladu iMac. Na rozdíl od přenosných zařízení nemůžeme stolní počítač napájet z interní baterie, a proto musí být připojen neustále do elektrické sítě [24]. Systémovou jednotkou je v této definici zamýšlen klasický dnešní počítač obsahující skříň s většinou hardware zařízení a zároveň periferní vstupně výstupní zařízení. Zobrazovacím zařízením pro stolní počítače jsou monitory. V současné době přes 96% uživatelů stolních počítačů používá webové prohlížeče ve vyšším rozlišení než 1024x768 pixelů [25]. Charakteristické jsou tedy pro monitory velké rozměry zobrazované plochy, které umožňují zobrazit velké množství informací. Stolní počítače jsou jedním z nejužívanějších zařízení pro prohlížení webových stránek, protože stolní počítače vlastní velké množství domácností a toto zařízení se již používá desítky let. První webové stránky byly vytvářeny právě pro stolní počítače a pro jejich rozlišení, proto se považuje dnes za standard, aby bylo možné pomocí webového prohlížeče na stolním počítači zobrazit danou webovou stránku.

3.1.2 Notebooky, netbooky

Notebook je malý přenosný osobní počítač napájený z interní baterie nebo z elektrické sítě. Tato zařízení mají integrovanou zobrazovací plochu, která v současnosti nabízí rozlišení 1366 x 768 a vyšší [26]. Notebooky jsou schopné zastávat stejné funkce jako stolní počítače. Některé notebooky v současné době nabízejí i dotykovou zobrazovací plochu s poměrně vysokým rozlišením. Z toho důvodu webové stránky, které podporují tato zařízení, by měla mít ovládací prvky na stránce (tlačítka, posuvníky, vstupní pole apod.) dostatečně velké, aby je bylo možné snadno ovládat pomocí dotykového rozhraní. Stejně jako u stolních počítačů i u notebooků platí to, že jejich pořád ještě dostatečně velká zobrazovací plocha umožňuje zobrazování velkého množství informací. Notebooky jsou tak velmi efektivní zařízení, nicméně jsou ve srovnání se stolními počítači o stejném výkonu mnohem dražší.

Netbook označuje počítač menší než notebook (tzv. subnotebook), který se zaměřuje na mobilitu, upřednostňuje nízkou spotřebu, cenu i váhu, a orientuje se především na poskytnutí přístupu k Internetu (WWW, E-mail) a jednodušší kancelářské práce. Zobrazovací plocha a rozlišení obrazovky je mnohem menší než u notebooků, nejčastěji 800 × 480 až 1 280 × 768 pixelů. Statické webové stránky zobrazované na tomto zařízení lze při vyšším rozlišení prohlížet bez jakýchkoliv potíží, avšak při nižších rozlišeních je třeba počítat s horizontálním a vertikálním posunem stránek pro správné zobrazení celého obsahu, což může být pro uživatele rušivé. Pokud je cílem vývojáře vývoj webu podporující toto zařízení je třeba vzít v úvahu využití adaptivního nebo responzivního designu.

3.1.3 Počítačové tablety

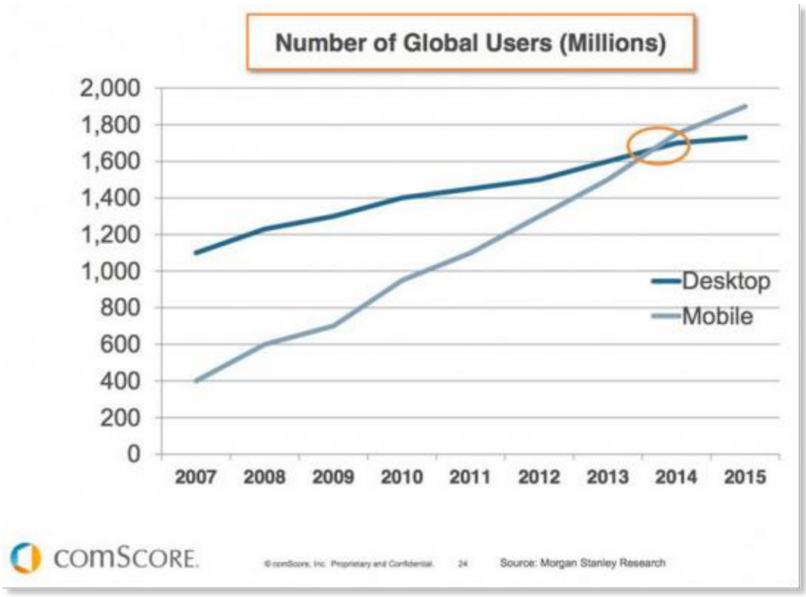
Počítačový tablet, zkráceně tablet, je tenký, plochý mobilní počítač s dotykovou obrazovkou, která je nejčastěji barevná, s procesorem a dobíjecí baterií v jednom zařízení. Tablety často obsahují také senzory, digitální fotoaparát, mikrofon a akcelerometr, takže obrázky na obrazovce jsou vždy zobrazeny zpříma [28]. Dotykové obrazovky mají typicky rozlišení obrazovky od 800 x 480 pixelů až po současných obvyklých 1900 x 1200 pixelů. Nižší rozlišení mívají typicky starší zařízení. Statické webové stránky zobrazované na tabletech lze na novějších zařízeních prohlížet bez jakéhokoliv omezení. Pro podporu starších zařízení a nižších rozlišení je třeba zvážit užití adaptivního nebo responzivního designu, protože zobrazení statické webové stránky na přístroji s nižším rozlišením nutí uživatele horizontálně a vertikálně posouvat obsah.

3.1.4 Mobilní telefony

Prohlížení webových stránek umožňují takzvané chytré telefony. Chytrý telefon je mobilní telefon, který obsahuje pokročilou funkcionalitu kromě pouhých telefonních hovorů a zasílání textových zpráv. Většina chytrých telefonů je schopná zobrazovat fotografie, přehrávat video nahrávky, prohlížet a zasílat e-mailovou poštu a surfovat na webu. Moderní chytré telefony, jakým je iPhone nebo telefony užívající operační systém Android, mohou spouštět aplikace třetích stran, což poskytuje neomezenou funkcionalitu [27].

Zobrazovací plocha chytrých telefonů je malá a velice různorodá. Standardně se v současnosti počítá s nejmenší šířkou a výškou 320 pixelů. U nejvýkonnějších chytrých telefonů je zobrazovací plocha široká až 1920 pixelů a to hlavně díky technologiím zvyšující hustotu pixelů v obrazu. Například pro telefony společnosti Apple jsou typické Retina obrazovky, které poskytují tento efekt.

V roce 2014 došlo k situaci, kdy počet uživatelů chytrých telefonů s připojením na Internet překročil počet uživatelů stolních počítačů [23] (Obrázek 14). A tento vzestup užívání chytrých telefonů pokračuje až do současnosti, proto se responzivní design webových stránek stal velmi významným designovým trendem, aby bylo možné co nejkomfortněji zpřístupnit uživatelům chytrých telefonů obsah webové stránky.



Obrázek 14: Graf znázorňuje počet uživatelů chytrých telefonů a stolních počítačů s připojením k Internetu. Převzato z [23].

3.1.5 Ostatní zařízení

Mezi další zařízení, pomocí kterých lze prohlížet webové stránky, patří takzvané chytré televize. Chytré televize obsahují integrovaný webový prohlížeč a ovládat je lze pomocí běžného ovládání nebo klávesnice. Zobrazovací plocha je obvykle mnohem větší než u stolních počítačů a rozlišení je udáváno obvykle od 1920 x 1200 pixelů výše. Poskytuje dostatek místa pro zobrazení velkého množství informací.

V současnosti jsou velice oblíbeným zařízením herní konzole, které díky technologickému pokroku mají možnost se připojit k internetu a prohlížet webové stránky. Stejně jako chytré televize obsahují integrovaný webový prohlížeč. Ovládání herních konzol je primárně pomocí ovladačů, ale v současnosti jsou podporovány i klávesnice. Herní konzole využívají k zobrazení informace obvykle externí zařízení, nejčastěji televize či monitory, které nabízejí velkou zobrazovací plochu s rozlišením od 1920 x 1200 pixelů. Opět je tak poskytována dostatečná velikost pro zobrazení velkého množství informací.

Existuje ještě celá řada dotykových zařízení poskytujících prohlížení webových stránek jako například chytré hodinky, čtečky elektronických knih nebo počítače uvnitř automobilů. Tato zařízení obsahují obvykle dotykovou obrazovku, která nemá příliš velké rozměry, nejčastěji do 7 palcové uhlopříčky. Rozlišení obrazovky je stejné jako u počítačových tabletů uvedených v sekci 3.1.3.

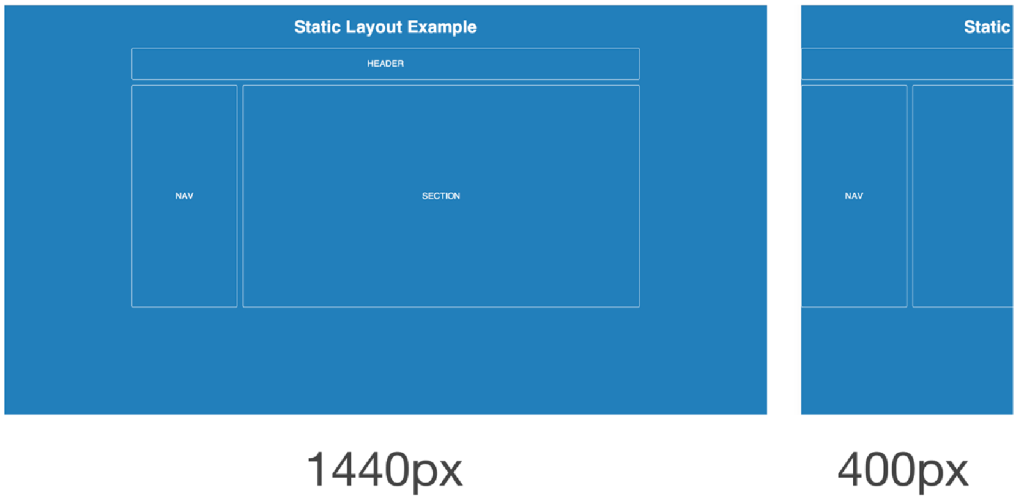
3.2 Rozložení webových stránek

Při návrhu webových stránek web designéři používají termíny jako statické, fluidní, adaptivní či responzivní rozvržení webové stránky. Obecně se jedná o popis toho, jak se webová stránka chová v různých webových prohlížečích s různou šířkou zobrazovací plochy. Následující podkapitola obsahuje popis těchto termínů a popis principů zobrazování webové stránky v daném rozložení.

3.2.1 Statické rozložení

Statické rozložení webové stránky (někdy nazývané jako fixní rozložení nebo rozložení s fixní šířkou) užívá přednastavenou velikost webové stránky a nemění ji při změně šířky webového prohlížeče. Jinými slovy, webová stránka má za všech okolností pevně danou šířku. Je to způsob, jakým byly webové stránky tradičně budovány po mnoho let až do období kolem roku 2010, kdy byly představeny moderní přístupy, jakými byly dotazy na média či responzivní design [20]. Na obrázku 15 je schematicky znázorněno statické rozložení

webové stránky její chování pro úzké obrazovky.



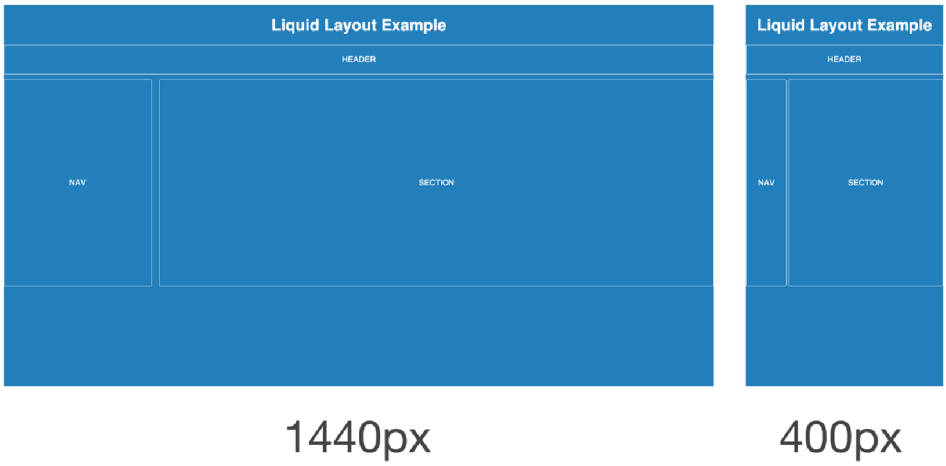
Obrázek 15: Schéma statického rozložení webové stránky pro různé obrazovky. Převzato z [20].

Různá zařízení zacházejí se statickým rozložením mnoha různými způsoby, proto se zobrazená webová stránka může chovat nepředvídatelně. Například, v prohlížeči na stolním počítači, pokud je okno prohlížeče horizontálně příliš malé, menší než samotná webová stránka, pak je webová stránka oříznuta a uživateli se zobrazí horizontální posuvník. Na mobilních zařízeních, jakým je například iPhone od společnosti Apple, je díky dostatečně velkému rozlišení obrazovky moderních chytrých telefonů webová stránka zmenšena automaticky (zobrazí se celá webová stránka) a umožňuje uživateli, aby si přiblížil obsah, který ho zajímá. Je třeba mít na vědomí, že toto chování na mobilních zařízeních je standardní a není možné toto chování pro statické rozložení nijak ovlivnit.

Při vývoji nových webových stránek se v současnosti statické rozložení příliš nepoužívá, protože pro správně zobrazovanou webovou stránku pro mobilní zařízení je třeba vytvořit oddělenou webovou stránku. Kvůli velké rozmanitosti obrazovek pro mobilní zařízení tento přístup není příliš vhodný. Je-li webová stránka vytvořena pro nejmenší zařízení o fixní šířce například 320 pixelů, pak se na moderním mobilním telefonu s šířkou 1024 pixelů zobrazí úzké a spatně čitelné. A také naopak, pokud webová stránka je vytvořena pro moderní mobilní telefon, pak na starších zařízeních s malou šířkou je třeba obsah webové stránky horizontálně posouvat. Existují však i výjimky, jakou je například webová stránka služby Apple store. Služba nabízí podporu pro mobilní telefony iPhone, které jsou schopny zobrazit statické rozložení stránky.

3.2.2 Fluidní rozložení

Likvidní rozložení (někdy nazývané jako fluidní rozložení nebo rozložení s fluidní šířkou) používá relativní jednotky namísto absolutních jednotek. Typicky likvidní rozložení užívá jednotky v procentech namísto počtu pixelů, ale I jiné relativní jednotky budou fungovat, například jednotka *em* [20]. Na obrázku 16 je schematicky znázorněno likvidní (fluidní) rozložení stránky a jeho chování pro menší obrazovky.



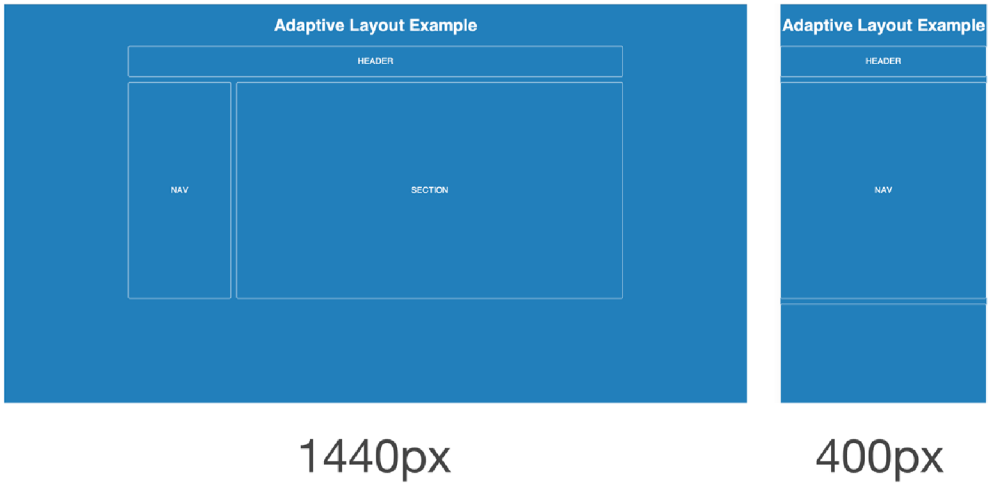
Obrázek 16: Schéma fluidního rozložení webové stránky pro různé obrazovky. Převzato z [20].

Fluidní rozložení často vyplňuje celou šířku webové stránky, nehledě na to, jaká je aktuální šířka webového prohlížeče. Fluidní rozložení není tak složité jako rozložení webové stránky užívající responzivní design, ale obsahuje některé velké nedostatky pro velké a velmi malé šířky prohlížeče. Pokud je okno prohlížeče příliš široké, pak může obsah stránky být příliš roztažen do šířky. Například na velkých obrazovkách se může odstavec textu zobrazit na jeden velmi dlouhý řádek. Naopak pro malé obrazovky může být obsah webové stránky horizontálně smrštěn a vertikálně roztažen. Příkladem opět může být odstavec, který po smrštění obsahuje na každém řádku jen několik slov, celý odstavec tak může být roztažen na desítky řádků, což velice snižuje čitelnost textu.

V praxi se fluidní rozložení stále používá, ale při vytváření webové stránky s tímto rozložením je třeba mít na paměti nevýhody plynoucí z použití na velkých a malých obrazovkách. Dobrou praktikou je udržovat obsah webové stránky co nejjednodušší. V testovací fázi je třeba ověřit zobrazení webové stránky ve všech podporovaných rozlišeních.

3.2.3 Adaptivní rozložení

Adaptivní rozložení webové stránky používá dotazy na média ze specifikace CSS3 pro detekci šířky okna webového prohlížeče a poté podle detekované šířky okna prohlížeče přizpůsobí rozložení stránky. Adaptivní rozložení užívá absolutní jednotky, jakými jsou například pixely, podobně jako statické rozložení, ale rozdíl je především v tom, že užívá několik pevných hodnot šířky definovaných danými dotazy na média [20]. Níže (obrázek 17) je znázorněno schematicky adaptivní rozložení webové stránky a jeho chování pro menší obrazovky.



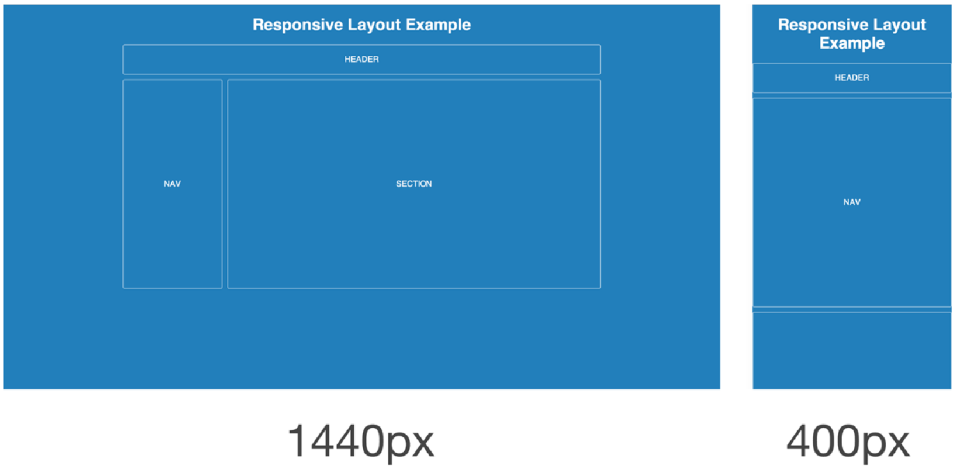
Obrázek 17: Schéma adaptivního rozložení pro různé velikosti obrazovky. Převzato z [20].

Na dotaz na médium lze pohlížet jako na logický výraz. Pokud se použije kombinace více dotazů na média, pak vytváří jednoduchý algoritmus. Příkladem může být webová stránka s adaptivním rozložením, která se bude chovat díky dotazům na média následujícím způsobem: „Pokud bude šířka okna webového prohlížeče 500 pixelů, pak nastav kontejner s obsahem stránky na šířku 400 pixelů. Pokud bude šířka okna prohlížeče 1000 pixelů, pak nastav šířku kontejneru na 960 pixelů.“ Mimo kontejner mohou být ostatní části jeho obsahu upravovány. Například změnou šířky komponent, změnou rozmístění obsahu nebo jejich odstraněním.

Adaptivní rozložení je dobré východisko ze situace, kdy je třeba přepracovat statické rozložení webové stránky na rozložení podporující mobilní zařízení. Typicky vyžadují méně času pro vývoj než responzivní rozložení. Hlavním nedostatkem je případ, kdy se šířka okna webového prohlížeče pohybuje mezi dvěma body definovanými pomocí dotazů na média. V tomto případě zobrazená webová stránka nemá ideální rozložení pro danou šířku, protože obsahuje buď velké množství volného prostoru, nebo naopak málo volného prostoru.

3.2.4 Responzivní rozložení

Responzivní rozložení webové stránky užívá relativní jednotky a také dotazy na média. Zdánlivě tak kombinuje přístupy pro fluidní a adaptivní rozložení. Se změnou šířky okna webového prohlížeče se responzivní rozložení chová stejně jako fluidní. Avšak při určitých šířkách definovaných pomocí dotazů na média se dramaticky změní rozložení webové stránky, tak aby se rozložení více přizpůsobilo šířce okna prohlížeče [20]. Obrázek 18 obsahuje schéma responzivního rozložení pro různé velikosti obrazovky.



Obrázek 18: Schéma responzivního rozložení pro různé velikosti obrazovky. Převzato z [20].

Responzivní design typicky využívá přístup „mobile first“. To znamená, že je prvně definováno rozložení webové stránky pro mobilní (chytré) telefony a poté se rozložení rozšiřuje pro počítačové tablety až po monitory stolních počítačů. Designér webových stránek tak musí nalézt vhodný způsob, jak rozšířit rozložení pro mobilní telefony tak, aby bylo uživateli majícímu větší zobrazovací zařízení zobrazeno dostatečné množství informací a na jeho obrazovce nebylo příliš prázdných míst. Tento trend má celkově velice dobré výsledky, protože je snažší rozšiřovat jednoduchý design než zjednodušovat složitý design s velkým rozložením pro obrazovky mobilních telefonů.

3.3 Principy responzivního designu

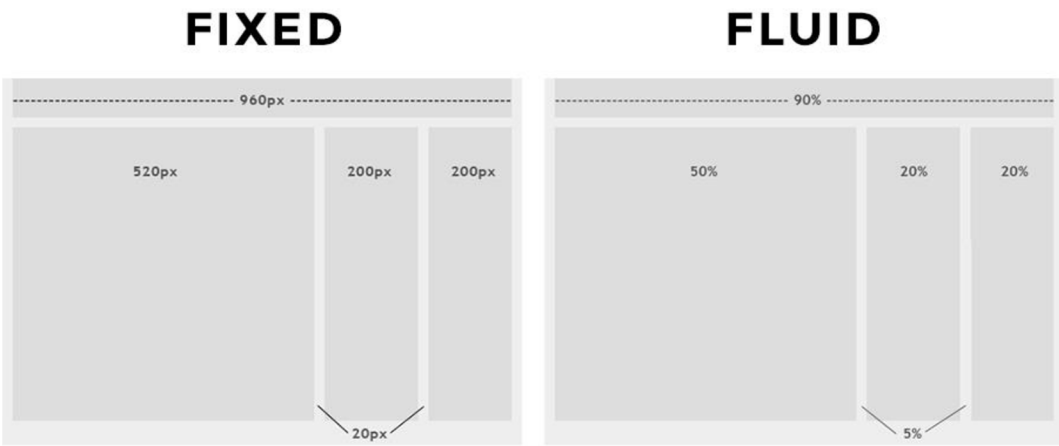
Jak bylo shrnuto v sekci 2.1.6, hlavním cílem responzivního designu je dosažení toho, že webová stránka bude naprogramována pouze jednou a bude ji možné spustit z jakéhokoliv

zařízení. Jinými slovy, se uživatel díky responzivnímu designu může spolehnout na to, že se aplikace, kterou si uživatel spustil, přizpůsobí jeho zařízení.

Tvůrcem responzivního designu je zkušený webový vývojář a designér Ethan Marcotte, který v roce 2009 vydal článek pro magazín *A list part* článek, který nazval „*Responsive Web Design*“. V něm se zaměřil na principy, které jsou pro responzivní design stěžejní a to fluidní mřížky, flexibilní obrázky a dotazy na médium [18]. Marcotte strávil řadu let výzkumem a obhajování technik popsaných ve svém článku. Tyto techniky se zakládají na praxí osvědčených postupech vývoje, které je snadné si osvojit.

3.3.1 Fluidní mřížky

Fluidní mřížky, první pilíř responzivního webového designu, jsou nástupci fluidních rozložení, která vznikala kolem roku 2006 z důvodů přechodu z 800 pixelové šířky na šířku 1024 pixelů [8]. Mřížka představuje způsob, jak zacházet s volným místem v rozložení webové stránky. Základem mřížky jsou řádky a sloupce. V komplexnějších mřížkách fungují sloupce jako registrační body pro rozvržení obsahu řádku, které je možné plnit jednotlivě nebo je slučovat do větších celků. Pro příklad uvažme dvanácti sloupcovou mřížku, její řádek může zahrnovat všech 12 sloupců tvořících jeden celek nebo vytvářet spojité oblasti sloučením několika sloupců do jednoho celku. Ve dvanácti sloupcové mřížce tak lze použít libovolné kombinace, které jsou však limitovány právě dvanácti sloupci.



Obrázek 19: Schematické porovnání pevné a fluidní mřížky. Převzato z [29].

Fluidní mřížky rozšiřují mřížkovou strukturu o proměnlivou šířku sloupců. Obsahují tedy sloupce a mezi nimi oddělovací mezery, kde jejich šířky jsou relativní k šířce obalujícího elementu. Obalující elementy mohou mít pevnou nebo proměnlivou šířku, ale sloupce musí být definovány relativně k těmto elementům, aby je bylo možné označit jako fluidní. Na obrázku 19 je zobrazeno schématické porovnání pevné a fluidní mřížky.

Výhoda používání fluidních mřížek je v tom, že se obsah webové stránky nerozvrhuje do pevných sloupců, ale fluidní mřížky samy určí finální rozměr, aby se obsah dobře zobrazil na zařízení uživatele [29]. Z toho důvodu se designér nebo vývojář může více věnovat obsahu a uživatelskému prožitku než struktuře aplikace.

Fluidní mřížky nejsou součástí jazyka HTML ani verze HTML 5. Pro fluidní mřížky se v praxi používají obvykle nástroje, které jsou již k dispozici, nejčastěji se jedná o aplikační rámec, který umožní rychlé zavedení mřížkové struktury.

3.3.2 Fluidní typografie

Fluidní typografie není jeden z klíčových pilířů responzivního designu, ale má pro něj poměrně velký význam. Fluidní typografie je obvykle součástí řešení fluidních mřížek v responzivních aplikačních rámcích. Znaky jsou nejmenší jednotky zobrazené na webové stránce. Cílem fluidní typografie je tedy správně zobrazený text, resp. jeho velikost, pro dané zařízení. Toho se dosáhne užitím relativních jednotek pro všechny aspekty typografie, díky tomu budou typografické prvky snadno reagovat na změnu zařízení. V praxi se používají dva přístupy pro určení hodnoty, ze které relativní hodnoty vycházejí. Výchozí hodnota vlastnosti *font-size* je buď nastavena pevně vývojářem, nebo je získána od uživatele, přičemž je preferována druhá možnost, kde se vychází z výchozí velikosti písma prohlížeče, která činí 16 pixelů pro všechny webové prohlížeče. Zajímavostí je to, že tato hodnota není nikde specifikovaná ani standardizovaná a není žádný důvod k tomu, aby všechny moderní prohlížeče používaly stejnou výchozí hodnotu.

Relativní jednotky používané pro fluidní typografii jsou obvykle *em* nebo *rem* z jazyka CSS. Rozdíl mezi nimi je v tom, že jednotka *em* je relativní vzhledem k rodičovskému elementu a jednotka *rem* je relativní přímo ke kořenovému elementu webové stránky, který je nejčastěji definován podle výchozí velikosti písma ve webovém prohlížeči – 16 pixelů.

Výpočet relativní hodnoty se provádí obvykle podle vzorce:

$$relativní\ hodnota = \frac{cílová\ hodnota\ (v\ pixelech)}{kontextová\ hodnota\ (v\ pixelech)}$$

Cílová hodnota je hodnota uvedená v pixelech, které se snažíme docílit. Kontextová hodnota je výchozí hodnota, která se určí z kontextu, například velikosti pro rodičovský či koře-

nový element. Relativní hodnota je tedy poměrná hodnota, která se udává ve výše zmíněných jednotkách *em* nebo *rem*.

3.3.3 Adaptivní obrázky

Adaptivní obrázky, druhý pilíř responzivního designu, umožňuje poskytovat obrázky bez ohledu na omezení rozměrů obrazovky. Adaptivní obrázky reagují na různé rozměry a rozlišení obrazovek. Usilují o překonání dvou problémů grafických prvků. Prvním problémem je, že pokud má stránka fluidní strukturu, její obsah musí být také fluidní. V opačném případě by vznikalo prázdné místo vedle fixního obrázku [8]. Docílit toho, aby byl obrázek fluidní, není tak snadné jako v případě textu. Druhým problémem je to, že je nutné reagovat na kvalitu obrazovek na moderních zařízeních. Jsou to právě techniky zhušťování pixelů (např. retina obrazovky od společnosti Apple nebo obrazovky AMOLED na zařízeních se systémem Android). Je tedy třeba vyřešit problémy s velikostí obrázku a rozlišením zařízení. Existuje několik řešení této problematiky a to pomocí jazyka CSS nebo jazyka HTML.

Pomocí kaskádových stylů lze řešit problém s fluidním obsahem pomocí vlastnosti šířky, která se nastaví na hodnotu 100% a výška se nastaví na hodnotu *auto.* To znamená, že se výška dopočítá na příslušnou hodnotu. Při zobrazení malého obrázku na velkých obrazovkách však dochází k degradaci obrazu. To je možné částečně eliminovat dvojím způsobem, buď pomocí dotazů na média nebo pomocí jazyka HTML 5. Při použití dotazů na média užíváme několik různých velikostí obrázku, každý obrázek pro určitou šířku obrazovky. Tento princip pokrývá i problém se zvýšenou hustotou pixelů obrazovky, podrobnější informace o tomto přístupu viz sekce 3.3.4. Jazyk HTML 5 přináší nový element <picture></picture> [52], který pomocí atributů *srcset* a *media* nastaví pro určitou šířku okna prohlížeče daný obrázek a navíc si dokáže pomocí modifikátorů pro atribut *srcset* poradit s vyšší hustotou pixelů v obrazovkách a správně tak reagovat na taková zařízení. Vhodnou kombinací tohoto přístupu a užití kaskádových stylů pro fluidní obrázky jsme schopni získat velice dobré výsledky v moderních prohlížečích. Pro starší prohlížeče je nutné využít tzv. „polyfilly“. „Polyfill“ je kód, jenž překlenuje mezi specifikací a implementací v prohlížeči – díky tomu můžeme používat nové technologie v jinak nekompatibilních prohlížečích.

3.3.4 Dotazy na média

Posledním velkým pilířem podle Marcotta [18] jsou dotazy na médium. Jedná se o nejsilnější z pilířů, který je velmi dobře podporován mezi moderními webovými prohlížeči. Dotazy na médium jsou součástí specifikace jazyka CSS 3 a také specifikace jazyka HTML 5, jež umožňuje, aby schopnosti obrazovky určovaly, jaké kaskádové styly aplikuje prohlížeč.

Prohlížeč díky nim rozpozná mobilní zařízení od monitoru stolního počítače a může tak aplikovat jiné kaskádové styly. Standardně vývojáři přizpůsobují rozložení, pozadí a obrázky, ale je možné použít i zcela novou sadu stylů.

Existují tři způsoby, jak aplikovat kaskádové styly v závislosti na dotazech na médium. Prvním způsobem je použití atributu *media* v elementu <link> jazyka HTML nebo XHTML:

```
<link rel="stylesheet" type="text/css" media="all and (color)"
href="/styles.css">
```

Druhým způsobem je použití atributu *media* v jazyce XML:

```
<?xml-stylesheet media="all and (color)" rel="stylesheet"
href="/styles.css">
```

Třetím způsobem je použití kaskádových stylů s prostřednictvím pravidla *@import* nebo *@media*:

```
@import url("/styles") all and (color);

@media all and (color){/* deklarace vlastností */}
```

Specifikace CSS3 [53] počítá s 13 vlastnostmi media – *width, height, device-width, device-height, orientations, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan* a *grid*. Kromě zmíněných dvou posledních vlastností mohou mít vlastnosti předponu *min-* nebo *max-*. Tyto vlastnosti přinášejí vekou míru kontroly. Výrobci prohlížečů nabízejí některé další vlastnosti a k nim přidávají příslušné předpony. Jednotlivé vlastnosti média lze skládat pomocí logických operátorů AND, OR a NOT.

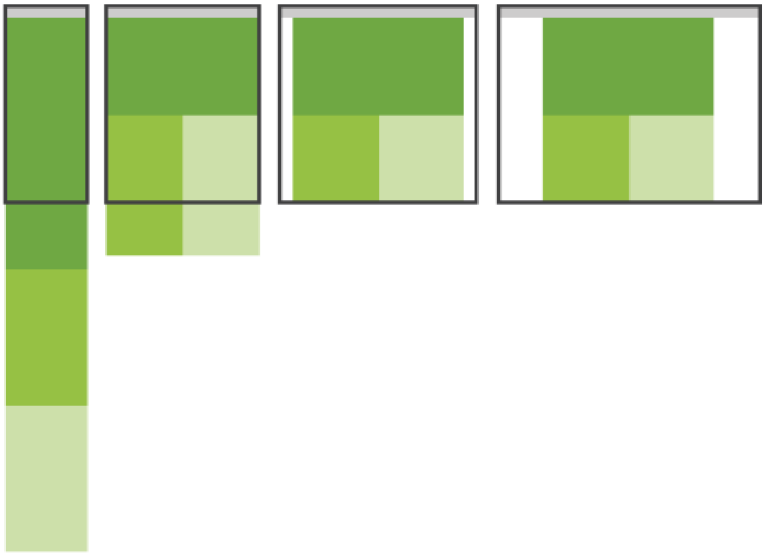
Pro aplikaci dotazů na média na webovou stránku nejsou žádná striktní pravidla, která by definovala, jak je přesně použít pro responzivní design. Nejdůležitější z pohledu responzivního designu je určení takzvaných bodů zlomu. Bod zlomu reprezentuje pravidlo (nebo skupinu pravidel) určující bod, v němž prohlížeč aplikuje obsah daného dotazu na médium na dané rozložení webové stránky [8]. Pro správnou volbu bodů zlomu je třeba zvážit, jaká zařízení budou podporována. Je také třeba stanovit zisk na základě toho, jak uživatelé přistupují k dané aplikaci. Jinými slovy, zda se ještě vyplatí investovat čas do vývoje pro dané zařízení. Dobrou praktikou je nalezení rovnováhy mezi body. Toho lze docílit tak, že počet bodů bude malý (ideálně 4-5 bodů pro velmi malá, malá, střední, velká a velmi velká zařízení). Rovnováha je ještě důležitější, pokud cílíme na nějaké konkrétní zařízení. Dále je dobrou praktikou nezkracovat zbytečně kód pro dotazy na média, to se v praxi stává velice často a vede to ke ztrátě udržovatelnosti kódu.

3.4 Návrhové vzory pro responzivní design

Na základě výše uvedených principů responzivního designu vznikly určité typy návrhových vzorů, které popisují chování responzivního rozložení webové stránky pro různá zařízení. Tato sekce je věnována stručnému představení hlavních návrhových vzorů, které se v praxi nejčastěji vyskytují.

3.4.1 Částečně fluidní rozložení

Tento návrhový vzor je nejoblíbenějším rozložením pro responzivní design a je překvapivě velice jednoduché. Užívá vícesloupcové fluidní rozložení pro všechny kromě nejmenších obrazovek mobilních (resp. chytrých) telefonů. Pro malé obrazovky se využije dotazů na média, pomocí nichž se přeskládá rozložení stránky tak, že sestaví sloupce s obsahem pod sebe. Obrázek 20 schematicky zachycuje tento návrhový vzor.

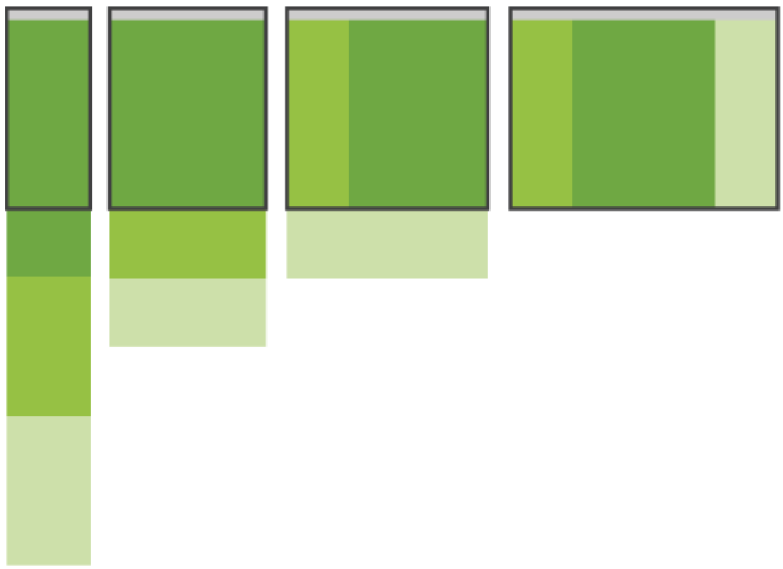


Obrázek 20: Schéma částečného fluidního rozložení na různě velkých obrazovkách. Převzato z [30].

3.4.2 Zahazování sloupců v rozložení

Dalším návrhovým vzorem je vzor užívající více sloupcové rozložení webové stránky a to tak, že na pravém okraji stránky je jeden sloupec. Při zobrazení na menších zařízeních než na obrazovkách monitorů se poslední sloupec vždy zahodí a jeho obsah se přesune pod ostatní

sloupce. Jinými slovy se rozložení webové stránky chová fluidně až po dosažení bodu zlomu, ve kterém se poslední sloupec zahodí, vytvoří se nový sloupec na novém řádku, který obsahuje veškeré elementy zahozeného sloupce. Velikost elementů zůstává konzistentní při změně velikosti okna webového prohlížeče. Tento mechanismus je schematicky zobrazen na obrázku 21.

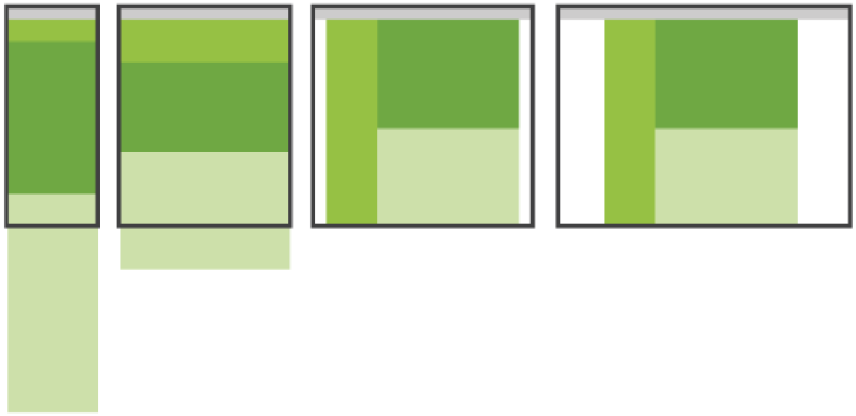


Obrázek 21: Schéma vzoru zahazujícího poslední sloupec na různě velkých obrazovkách. Převzato z [30].

Moment, kdy a jak se každý sloupec zahodí a přesune pod ostatní sloupce pro různě velké obrazovky, závisí na rozložení bodů zlomu a také je to závislé na použitém designu, který je tímto způsobem realizován. Obecně platí, že navigace a důležitý obsah se obvykle umisťuje do horní části při zobrazení na menších obrazovkách [30].

3.4.3 Přeuspořádání rozložení

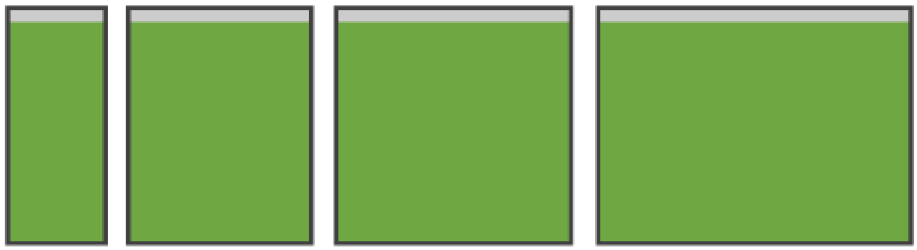
Tento návrhový vzor se snaží nejvíce přizpůsobit pro různé velikosti obrazovek. Pro velké, střední a malé obrazovky je definováno jiné rozložení, které přesně pasuje na příslušné zařízení. Tento vzor je pracnější než ostatní vzory a tedy i méně oblíbený mezi vývojáři [30]. Nicméně poskytuje mnohem lepší výsledky. Opět se tedy jedná o vzor, který se chová fluidně a v bodech zlomu dochází k přeuspořádání rozložení webové stránky. Obrázek 22 schematicky zachycuje tento vzor a jeho chování pro různě velké obrazovky. Tento vzor používá velká spousta moderních řešení pro responzivní design.



Obrázek 22: Schéma vzoru přeuspořádavajícího rozložení webové stránky pro různá zařízení. Převzato z [30].

3.4.4 Drobná vylepšení v rozložení

Tento vzor je vhodný pro velmi jednoduché webové stránky s minimálním obsahem – obsahem majícím velmi málo elementů uvnitř jedno sloupcového rozložení. Tento vzor je velmi málo používaný, protože existuje malé množství webových stránek s velice jednoduchým obsahem. Podporu responzivity lze získat pomocí několika málo úprav, nejčastěji zavedením fluidní typografie a adaptivních obrázků. Obrázek 23 zobrazuje schéma tohoto návrhového vzoru pro různě velké obrazovky.

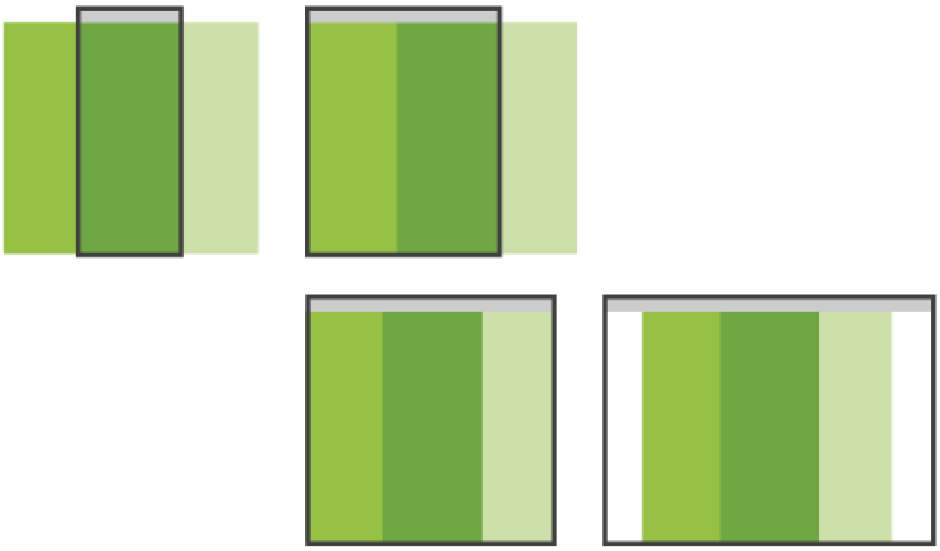


Obrázek 23: Schéma vzoru provádějícího drobná vylepšení v rozložení webové stránky pro různá zařízení. Převzato z [30].

3.4.5 Skrývání sloupců rozložení

Posledním z uváděných návrhových vzorů je vzor skrývající sloupce v rozložení webové stránky při zobrazování na určitém zařízení. Základem je více sloupcové rozložení stránky, které zobrazuje pro velké obrazovky celý obsah. Pokud danou stránku prohlížíme zařízením s užší obrazovkou a některý ze sloupců by byl jen zčásti viditelný nebo vůbec by nebyl vidět, pak takový sloupec pro dané zařízení skryjeme a zobrazíme jen ten nejnutnější obsah.

Na obrázku 24 je k dispozici schéma tohoto vzoru. Jak je vidět, na nejmenších zařízeních se zobrazí pouze jeden ze sloupců a ostatní jsou skryty a pro dané zařízení nejsou dostupné. Pro větší zařízení je již k dispozici více sloupců a tím tedy i zobrazeno více informací. Tento vzor vyžaduje důkladnou analýzu informací, které chceme zobrazit pro jaké zařízení, tak, aby bylo uživatelské rozhraní konzistentní.



Obrázek 24: Schéma vzoru skrývajícího pro menší zařízení sloupce v rozložení webové stránky. Převzato z [30].

3.5 Návrhové vzory pro navigační nabídku

Navigační nabídka je jednou z klíčových komponent každé webové stránky. Dobře navržená navigační nabídka může zvýšit použitelnost a uživatelský prožitek webové stránky. Během posledních let vývoje responzivního designu se ustálily některé techniky tvorby navigačních nabídek. Tato sekce shrnuje základní návrhové vzory pro navigační nabídku webových stránek. Dále se věnuje návrhovým vzorům nabídek pro mobilní zařízení, které jsou velmi důležité, protože při zobrazení webové stránky na mobilním zařízení zde dochází k transformaci původní navigace webové stránky na navigaci pro mobilní zařízení.

3.5.1 Navigace pro webové stránky

Navigace pro webové stránky bývá zpravidla umístěna ve vrchní části obrazovky v závislosti na designu, jaký je použit. Níže jsou představeny základní typy navigací, které se v součas-

nosti nejčastěji používají.

Statická a fixní navigační lišta

Statická navigační lišta je prvním a dnes nejběžněji užívaným vzorem pro navigační nabídku. Je umístěna pevně v horní části obrazovky a obsahuje samotnou navigaci, velmi často však i komponentu pro vyhledávání obsahu a někdy obsahuje i odkaz pro uživatelské nastavení. Fixní navigační lišta rozšiřuje statickou o to, že lišta je při vertikálním posunu po stránce neustále připevněná v horní části obrazovky. Pro ilustraci viz obrázek 25.

Obrázek 25: Ukázka webové stránky s fixní navigační lištou. Převzato z [33].

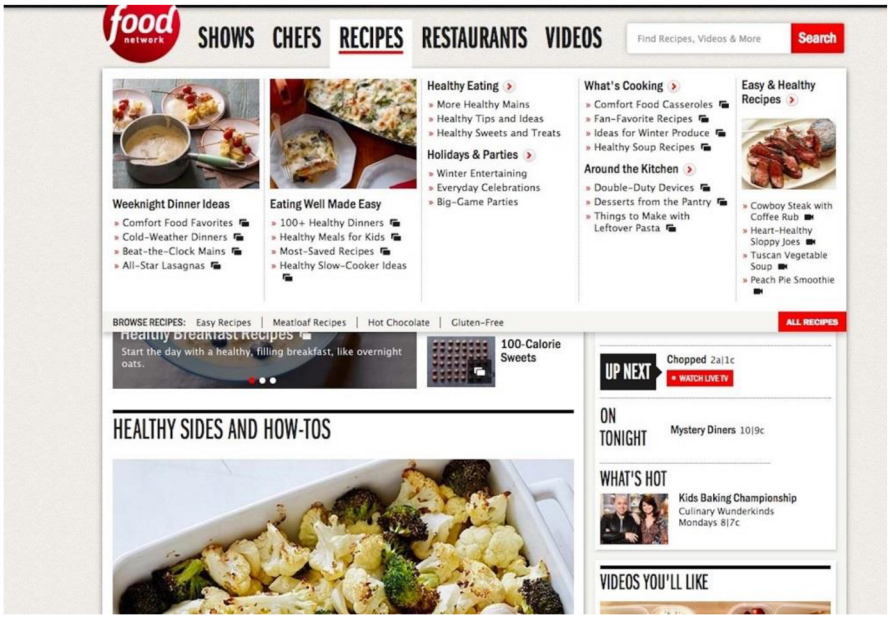
Hypertextové odkazy

Navigační nabídka pomocí hypertextových odkazů je velmi oblíbený přístup tvorby navigační nabídky, hlavně z toho důvodu, že je plně přizpůsobitelný. Úprava hypertextového odkazu lze snadno provést pomocí kaskádových stylů. Tímto přístupem lze pokrýt velké množství typů navigace. Na obrázku 26 se nachází příklad webové stránky užívající hypertextové odkazy pro navigaci.

Obrázek 26: Ukázka webové stránky s navigací tvořenou hypertextovými odkazy. Převzato z [33].

Rozsáhlá navigační nabídka

V současných webových stránkách je významným trendem implementovat velké navigační nabídky a to tak, že uživatel má k dispozici základní jednoduchou nabídku, při výběru prvku se zobrazí sekundární nabídka a teprve při jejím výběru se mění obsah. Tento přístup přináší několik výhod posun obsahu a pomáhá lépe strukturovat rozsáhlý obsah webové stránky [33]. Na obrázku 27 se nachází ukázka webové stránky užívající rozsáhlou navigační nabídku.



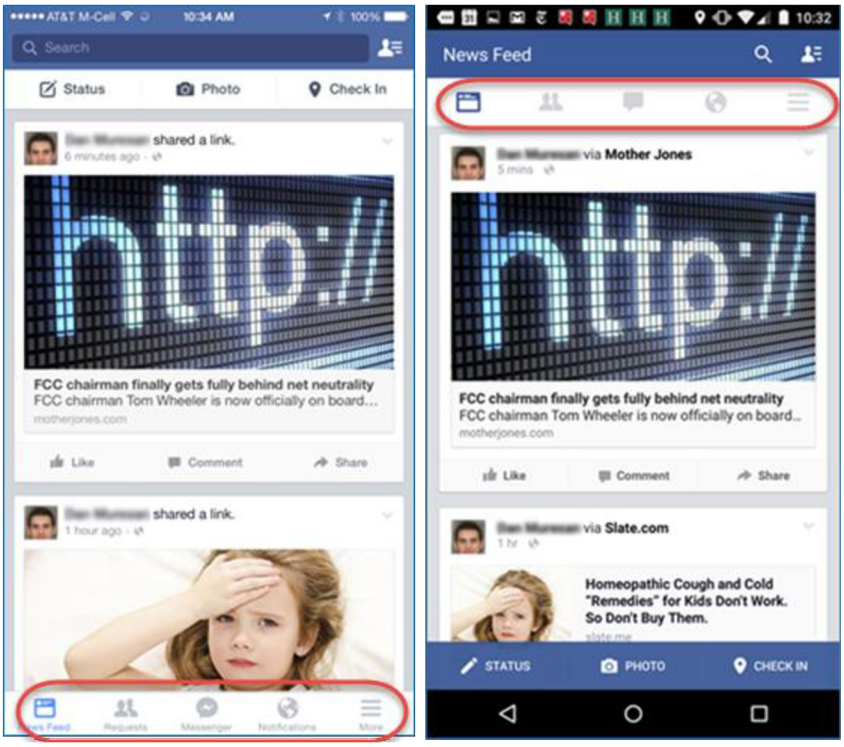
Obrázek 27: Ukázka webové stránky s rozsáhlou navigací. Převzato z [33].

3.5.2 Navigace pro mobilní zařízení

Problematiku pro mobilní zařízení shrnula ve svém článku Raluca Budiu [31], která se zabývá komerčním výzkumem v oblastech uživatelských rozhraní a uživatelského prožitku, a popsala tak několik základních návrhových vzorů pro navigační nabídky. Tato podsekce stručně představí ty nejvýznamnější návrhové vzory.

Navigační lišta

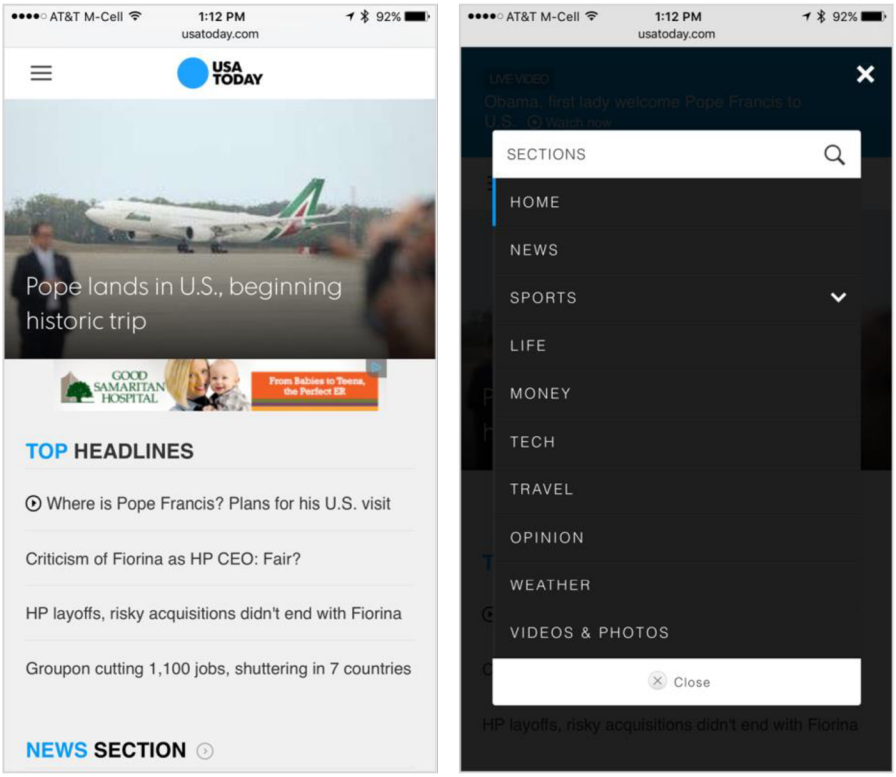
Navigační lišta zobrazující se v mobilních zařízeních je dvojího druhu. První typ lišty je odvozený od statické či fixní navigační lišty. Zobrazuje se v horní části obrazovky a často zobrazuje pouze základní nabídku, zpravidla zjednodušenou použitím ikon pro prvky nabídky. Lišta bývá obvykle fixní, to znamená, že je viditelná stále a to i během vertikálního posunu obsahu. Druhým typem je fixní navigační lišta umístěná ve spodní části obrazovky. Hlavním důvodem přemístění této lišty dolů bylo zjednodušení interakce s uživatelem, protože současné chytré telefony mají velké obrazovky a navigační lišta v horní části obrazovky je obtížněji dostupná při běžném uchopení telefonu. Některé mobilní aplikace nabízejí obě varianty navigační lišty pro různá zařízení (obrázek 28). Současným trendem je preference druhého typu, tedy navigační lišty ve spodní části obrazovky.



Obrázek 28: Mobilní aplikace Facebook s různou navigační lištou pro různá zařízení, vlevo zobrazeno na telefonu iPhone a vpravo na zařízení s operačním systémem Android. Převzato z [31].

Hamburger menu

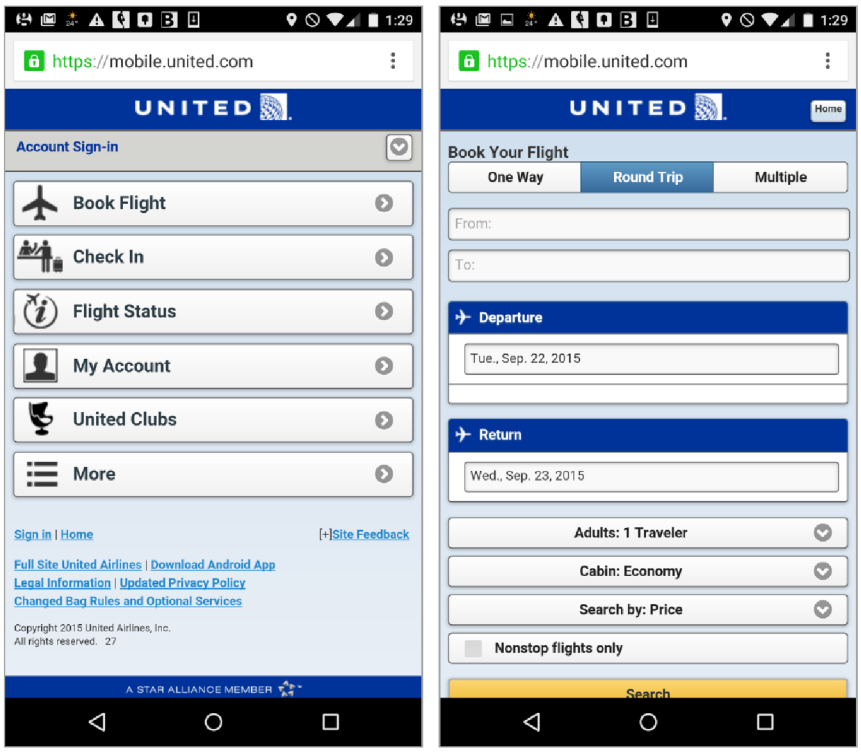
Dalším typem nabídky pro mobilní zařízení je tzv. „hamburger menu“. Tato nabídka se vyznačuje umístěním jednoho tlačítka obsahujícím tři horizontální pruhy pod sebou do horní části obrazovky. Při interakci s tímto tlačítkem se uživateli zobrazí hlavní nabídka vertikálně seřazena pod sebou a při opětovné interakci nabídka se skryje nebo ji je možné skrýt pomocí tlačítka pro uzavření. Na obrázku 29 se nachází ukázka webové aplikace s tímto typem nabídky. V současnosti je trend takový, že se pomalu od tohoto přístupu upouští, protože tlačítko v horní části je obtížněji dosažitelné.



Obrázek 29: Obrázek vlevo zobrazuje v horní části obrazovky tzv. „hamburger menu“, vpravo je ukázka zobrazené hlavní nabídky po interakci s uživatelem. Převzato z [31].

Navigační centrum

Tímto pojmem se označuje navigační nabídka, která je zobrazena na celé domovské stránce a zobrazuje seznam všech možností výběru najednou. Pokud tedy zvolíme některou sekci a uživatel se chce přepnout jinam, je třeba jeden krok navíc – vrátit se na domovskou stránku a odtud zvolit cílovou sekci. Příklad tohoto typu navigační nabídky je na obrázku 30. Tento přístup je vhodný pro webové stránky a mobilní aplikace založené na úlohách. Zejména pokud je cílem omezit uživatele pouze na jednu větev navigační stromové struktury v průběhu jedné relace [31]. V současnosti tento přístup není až tak častý a designéři dávají přednost předchozím typům navigačních nabídek.



Obrázek 30: Ukázka webové aplikace pro mobilní zařízení s navigačním centrem. Vlevo se nachází nabídka nejvyšší úrovně. Vpravo je ukázka aplikace v zanořené stromové struktuře navigace. Převzato z [31].

3.6 Přístupy k tvorbě responzivního designu

Pro vývoj responzivního webového designu existuje celá řada nejrůznějších nástrojů. V praxi se používají převážně již hotová řešení, jako různé knihovny či aplikační rámce, jejichž součástí je podpůrná funkcionalita pro tvorbu responzivních webových stránek. Responzivních webových stránek lze docílit i pomocí vlastní implementace fluidní mřížky a vlastní implementace responzivního rozložení užitím adaptivních obrázků a dotazů na média. Tato podkapitola obsahuje přehled základních nástrojů a přístupů pro tvorbu responzivních webových stránek.

3.6.1 Vlastní responzivní řešení

Prvním způsobem tvorby responzivního uživatelského rozhraní je vytvořením vlastního řešení pro responzivní webový design. Základem tohoto přístupu je tvorba responzivní fluidní mřížky, pomocí které se definuje základní rozložení webové stránky. Jelikož je v sou-

časnosti nejpoužívanější přístup k vývoji *mobile-first,* je základem vytvořit rozložení nejprve pro mobilní zařízení a nadefinovat jeho chování podle některého z návrhových vzorů uvedených v podkapitole 3.4, tzn. definovat rozložení pro větší zařízení.

Vlastní responzivní mřížka

Základem každého responzivního rozložení je fluidní mřížka, viz sekce 3.3.1. Před samotným vytvářením mřížky je třeba si zvolit některý z návrhových vzorů, rozmyslet si body zlomu pro responzivní rozložení a určit chování mřížky na velkých obrazovkách.

Jedním z prvních kroků při tvorbě je definovat si hlavní element, který obsahuje veškerý obsah webové stránky. Obvykle se označuje jako „*container*" a ten je definován dvojím způsobem podle typu užitého responzivního rozložení. První typ má relativně danou šířku nastavenou na 100% nebo na 97,999% (s odsazením zleva a zprava o 0,999%). Díky tomu je mřížka plně fluidní. Druhý typ má několik pevně daných hodnot, které odpovídají bodům zlomu responzivního rozložení a definují tak maximální šířku obsahu pro daná zařízení.

Dalším krokem je definice fluidní typografie popsané v sekci 3.3.2. Dobrou praktikou je vycházet z výchozí velikosti písma prohlížeče. Konkrétní nastavení však silně závisí na použitém designu webové stránky. V praxi se často typografie nastavuje užitím dotazů na média.

Posledním krokem je nastavení sloupců a řádků mřížky. To se nastavuje relativně ke svému rodičovskému elementu. V moderních aplikačních rámcích se pro řádky definuje pouze negativní odsazení ve velikosti odsazení sloupců. Počet definovaných sloupců může být libovolný, nejčastěji se používá 12 sloupců, protože číslo 12 je dělitelné 1,2,3,4,6,12 a nabízí tak spoustu možností rozložení obsahu. Definovat je nutné všechny možné spojité bloky, které mohou vzniknout sloučením několika sloupců. Relativní velikost se určí poměrnou částí z celku a je nutné vzít v úvahu i odsazení sloupců mezi sebou. Hodnota se vypočítá pomocí vzorce uvedeného v sekci 3.3.2. Pro relativní hodnoty je důležité nepoužívat celá čísla, ale použít čísla desetinná. Často se používají hodnoty o tisícinu nebo setinu menší, protože pak nedochází k nežádoucím chybám v zaokrouhlování čísel. Je to výhodné řešení, protože tak malou odchylku člověk pouhým okem nerozpozná.

Užití dotazů na média a adaptivních obrázků

Pomocí vlastní responzivní mřížky jsme schopni definovat základní hrubé responzivní rozložení. Pomocí dotazů na média jsme schopni toto rozložení vylepšit tak, že je možné na dané zařízení reagovat změnou samotného rozložení nebo obsahu webové stránky. Způsob, jakým se bude rozložení měnit, závisí na užitém návrhovém vzoru. Změna obsahu je závislá na použitém designu webových stránek. Pro responzivitu obrázků se použijí techniky pro adaptivní obrázky uvedené v sekci 3.3.3.

Užitím výše popsaných principů a metod lze snadno vytvořit vlastní responzivní řešení. Důvody, proč se tento přístup příliš nepoužívá, jsou v tom, že vytvoření vlastního řešení je časově náročné, stojí spoustu finančních prostředků a v současnosti je zbytečné, protože současná řešení poskytují dostatečnou a osvědčenou podporu pro tvorbu responzivních webových stránek.

3.6.2 RESS

RESS (Responsive web design + server side components) je koncept užívající responzivní design a doplňuje jej o detekci vlastností stránky tak, aby poskytl upravený obsah kdykoliv je třeba [34]. Jinými slovy, tento přístup detekuje zařízení a jeho vlastnosti, resp. vlastnosti prohlížeče, kterým je webová stránka prohlížena. Poté front-endová část aplikace odešle dotaz na server. Server vrací responzivní komponenty, tedy responzivní rozložení webové stránky, responzivní obsah včetně příslušně velkých obrázků pro dané zařízení. Všechny responzivní komponenty jsou maximálně přizpůsobené pro dané zařízení a prohlížeč.

Tento přístup představil Lucas Wreblowski v roce 2011 [36] a má poměrně dobré výsledky z pohledu responzivity webové stránky. V témže roce převzala tento přístup i společnost Twitter, která vlastní jeden z nejpoužívanějších responzivních aplikačních rámců – Bootstrap [38]. Twitter však nepřišel s žádným nástrojem užívajícím tento přístup. Hlavním problémem tohoto přístupu je detekce vlastností prohlížeče a zařízení uživatele. Vývojářská komunita pro řešení problému s detekcí doporučuje použít knihovnu Modernizr [54], která slouží právě k účelu detekce vlastností prohlížeče a zařízení uživatele na straně klienta. I přes toto doporučení není RESS příliš rozšířen skrze časovou a finanční náročnost implementace.

Výhody RESS: [35]

- **Snadná navigace**: Navigace může být přizpůsobena uživateli, jeho zařízení nebo jeho roli a příslušným oprávněním v aplikaci.
- **Menší velikost komponent**, které jsou přizpůsobeny pro dané zařízení a webový prohlížeč.
- **Rychlé načítání**: Načítán je jen malý obsah bez nepotřebných kaskádových stylů či nepotřebných skriptů v jazyce JavaScript.

Nevýhody RESS: [35]

- **Velké množství prostředků**: Pro každý typ zařízení či prohlížeče.
- **Dynamické webové stránky:** Zvyšují zátěž serverové části aplikace.
- ***Cache* – vyrovnávací paměť:** Pro zvýšení výkonu aplikace je nutná implementace vyrovnávací paměti pro responzivní komponenty.
- **Detekce zařízení**: Současná řešení pro detekci zařízení jsou velmi nespolehlivá.

3.6.3 Současné nástroje pro responzivní design

Současné nástroje pro vývoj responzivního designu jsou velmi často komplexní aplikační rámce, jejichž součástí je implementace responzivního rozložení – resp. responzivní mřížky. Navíc obvykle nabízejí i sadu responzivních komponent, pomocí kterých lze vývoj webových stránek značně urychlit. Některé z těchto aplikačních rámců nabízejí přímou podporu pro aplikační rámce pro tvorbu dynamických webových stránek (AngularJS, ReactJS aj.). To má taktéž pozitivní vliv na rychlost vývoje webových stránek a front-endové části webových aplikací. Tato sekce obsahuje stručný přehled významných aplikačních rámců pro vývoj responzivních webových stránek. Detailnější informace o uvedených nástrojích viz podkapitola 3.7.

3.6.3.1 Twitter Bootstrap

Twitter Bootstrap je nejpopulárnější HTML, CSS a JavaScriptový aplikační rámec určený pro vývoj responzivních *mobile-first* webových stránek [38]. Bootstrap obsahuje sadu přizpůsobitelných šablon pro úpravu typografie, formulářů, tlačítek, navigace a dalších komponent. Bootstrap vyvinul Mark Otto a Jacob Thornton ze společnosti Twitter jako aplikační rámec podporující konzistenci mezi interními nástroji. Před Bootstrapem byla pro vývoj rozhraní využívána řada knihoven, což vedlo k nekonzistenci zdrojových kódů a velké náročnosti na údržbu. Mark Otto podle Twitteru prohlásil: „*Velmi malá skupina vývojářů společně se mnou měla za úkol navrhnout a vybudovat nový interní nástroj a viděla příležitost udělat něco více. Skrze tento proces jsme chtěli vytvořit něco o moc důležitějšího než další nástroj pro vnitřní využití. O měsíc později jsme skončili s první verzí Bootstrapu jako metodou k dokumentaci a sdílení běžných designových vzorů a přínosů ve společnosti* [37].“ V srpnu 2011 byl vydán Twitter Bootstrap jako open source nástroj a v únoru 2012 se stal nejoblíbenějším vyvíjeným projektem na serveru GitHub. V současnosti se používá Bootstrap verze 3 vydaný roku 2013, avšak ve vývoji je již čtvrtá verze, která má být vydána v průběhu roku 2017.

3.6.3.2 Zurb Foundation

Foundation [39] je responzivní front-endový aplikační rámec. Foundation poskytuje responzivní mřížku, komponenty pro uživatelská rozhraní, šablony, fragmenty kódu včetně typografie, formulářů, tlačítek, navigace a ostatních komponent a také obsahuje JavaScriptová rozšíření. Foundation je spravován společností Zurb. Koncem roku 2011 Zurb vydal Foundation 2.0 jako open-source projekt pod MIT licencí. Na podzim roku 2016 byla vydána nová současná verze Foundation 6.

3.6.3.3 Skeleton

Skeleton není přímo aplikačním rámcem, ale pouze malá kolekce CSS souborů, která umožňuje snadnou a rychlou tvorbu responzivních webových stránek. Poskytuje nástroje pro tvorbu responzivního rozložení a několik stylů pro komponenty uživatelského rozhraní jako například tlačítka, formuláře, záložky aj. [40].

3.6.3.4 Semantic UI

Semantic je front-endový aplikační rámec navržený pro tvorbu šablon. Zahrnuje více než padesát responzivních komponent včetně responzivní mřížky, stovky proměnných pro přizpůsobení komponent pro výsledný design. Semantic umožňuje vývojářům rychlý vývoj webových stránek se stručným HTML, intuitivním JavaScript kódem a zjednodušeným laděním pomáhajícím udělat z vývoje front-endu příjemný zážitek. Semantic je navržen responzivně a umožňuje zobrazit webovou stránku na libovolném zařízení [41].

3.6.3.5 Materialize

Materialize [42] je frontendový aplikační rámec, který byl navržen pro tvorbu responzivních webových stránek s využitím *material* designu (viz sekce 2.2.2). Obsahuje responzivní komponenty, kaskádové styly pro stylování vlastních elementů v *material* designu a několik fragmentů kódu pro řešení běžných efektů a problémů.

3.6.3.6 Material UI

Material UI [55] je množina responzivních komponent v *material* designu implementovaných pomocí aplikačního rámce React. Neobsahuje však responzivní mřížku ani rozložení, pouze responzivní komponenty. Material UI lze integrovat s jinými aplikačními rámci (Ember, Backbone), přičemž preferovaný je React.

3.6.3.7 Pure

Pure [56] je odlehčený modulární aplikační rámec pro tvorbu responzivních webových stránek. Obsahuje některá jednoduchá rozložení, šablony, velké množství jednoduchých responzivních komponent. Tento nástroj vznikl v roce 2014 jako alternativa k složitým a příliš velkým aplikačním rámcům, jakým je například Twitter Bootstrap. Za vývojem tohoto nástroje stojí společnost Yahoo.

3.6.3.8 UI kit

UI kit [57] je dalším představitelem jednodušších front-endových aplikačních rámců. Je určen primárně pro tvorbu responzivních webových stránek. K tomuto účelu poskytuje responzivní rozložení, responzivní komponenty a nástroj pro snadné přizpůsobení komponent pro daný design.

3.7 Srovnání nástrojů pro responzivní design

Podkapitola 3.7 je věnována porovnání existujících nástrojů pro tvorbu responzivního designu uvedených v sekci 3.6.3. Srovnání obsahuje detailnější informace o daných nástrojích a také může obsahovat subjektivní kritéria, podle kterých byly nástroje porovnány, např. hodnocení časové náročnosti pro osvojení si daného nástroje či jiná hodnocení vyplývající ze zkušenosti autora této práce.

3.7.1 Kritéria pro porovnání nástrojů

Kromě podrobnějších informací o daných nástrojích, porovnání obsahuje následující kritéria:

- **Hodnocení vývojářskou komunitou** – Většina současných nástrojů jsou *open-source* projekty mající své repozitáře se zdrojovými kódy nejčastěji na serverech GitHub. Na GitHubu je pro vývojářskou komunitu možnost hodnotit dané nástroje. Hodnocení je převzato z tohoto serveru.
- **Princip, koncept** – Princip, na kterém je daný nástroj navržen.
- **Velikost nástroje** – Velikost paměti, kterou nástroj zabírá ve webovém prohlížeči.
- **Responzivní rozložení a komponenty** – Kritérium, zda nástroj podporuje responzivní rozložení, zavedení responzivní mřížky a responzivní komponenty.
- **Modularita –** Kritérium, které určuje, zda lze s nástrojem pracovat jako s moduly, se kterými lze pracovat pomocí knihoven CommonJS nebo RequireJS, které slouží ke snadnému importování modulů do webové aplikace.
- **Užití CSS preprocesorů** – Výčet podporovaných CSS preprocesorů.
- **Sada vlastních ikon** – Kritérium určuje, zda nástroj obsahuje vlastní sadu ikon.
- **Dostupnost základních šablon** – Kritérium, které určuje, zda má nástroj k dispozici nějakou sadu předdefinovaných šablon různých typů webových stránek.
- **Dokumentace** – Toto kritérium stručně zhodnocuje stav dokumentace daného nástroje.
- **Instalace** – Popis možností instalace daného nástroje. Toto kritérium bylo požadavkem ze strany *Unicorn Systems*.
- **Integrace s aplikačními rámci** – Kritérium určující, zda je možné daný nástroj integrovat do jiných front-endových aplikačních rámců a uvedení příslušných příkladů. Toto kritérium bylo požadavkem ze strany *Unicorn Systems*.
- **Přizpůsobitelnost komponent** – Kritérium určuje, zda nástroj má nějakou podporu pro přizpůsobení responzivních komponent pro výsledný design webové aplikace.
- **Dostupnost tutoriálů** – Toto kritérium určuje, zda jsou dostupné tutoriály pro daný nástroj.
- **Časová náročnost** – Kritérium určující odhad časové náročnosti na osvojení si daného nástroje. Toto kritérium bylo požadavkem ze strany *Unicorn Systems*.
- **Práce s nástrojem** – Kritérium, které cílí na ověření, jak je jednoduché a intuitivní pracovat s daným nástrojem. Toto kritérium bylo požadavkem ze strany *Unicorn Systems*.

- **Velikost komunity** – Počet přispěvatelů, uživatelů sledujících vývoj nástroje.
- **Podpora prohlížečů** – Výčet podporovaných webových prohlížečů.
- **Licence** – Druh licence.

3.7.2 Metodika porovnání

Pro porovnání byly zvoleny nejčastěji užívané nástroje z nástrojů uvedených v sekci 3.6.3 a to Twitter Bootstrap, Zurb Foundation, Semantic UI, Materialize a Pure CSS jako zástupce menších nástrojů. Každý z porovnávaných nástrojů byl podroben důkladné analýze, pro některá kritéria byl proveden experiment, jehož výsledkem bylo vytvoření jednoduché webové stránky s responzivní mřížkou a s užitím několika responzivních komponent daného nástroje. Mohla tak být porovnána syntaxe, instalace nástroje, práce s daným nástrojem a odhadnuta časová náročnost. Na základě analýzy bylo možné provést hodnocení dokumentace, integrace s jinými aplikačními rámci užívanými pro vývoj webových stránek. Výsledky byly zpracovány v následující sekci.

3.7.3 Výsledky analýzy a experimentů

Následující podsekce shrnují výsledky analýzy a experimentů pro zvolené nástroje popsané v sekci 3.7.2. Výsledky jsou zapsány strukturovaně podle kritérií uvedených v sekci 3.7.1.

3.7.3.1 Twitter Bootstrap

- **Hodnocení vývojářskou komunitou** – 105 tisíc hvězd na serveru GitHub, je nejlépe hodnoceným nástrojem tohoto serveru
- **Velikost komunity** – Více než 800 přispěvatelů, přes sedm tisíc sledujících vývojářů a desiti-tisíce vývojářů užívajících tento nástroj. Jedná se o největší komunitu.
- **Princip, koncept** – *Mobile-first*, responzivní webový design (viz podkapitola 2.2).
- **Velikost nástroje** – 145 kB.
- **Responzivní rozložení a komponenty** – Nástroj nabízí podporu pro tvorbu responzivní fluidní mřížky a sadu responzivních komponent – *dropdown* nabídka, tlačítka, vstupní pole, navigace, navigační lišta, drobečková navigace, stránkování popisky, odznaky, notifikační zprávy, nadpisy, *progress bar*, panely, seznamy, tabulky, formuláře, obrázky a nástroje pro typografii.

- **Modularita** – Nástroj je modulární.
- **Užití CSS preprocesorů** – Nástroj užívá CSS preprocesory LESS nebo SASS.
- **Sada vlastních ikon** – Nástroj nabízí sadu vlastních ikon – Glyphicons.
- **Dostupnost základních šablon** - Nástroj nabízí sadu předdefinovaných šablon a témat.
- **Dokumentace** – Nástroj má přehlednou dokumentaci k jednotlivým responzivním komponentám, responzivní mřížce, užívaným kaskádovým stylům a skriptům v jazyce JavaScript. Výtku by si zasloužila navigace jednotlivých responzivních komponent, její užívání není příliš intuitivní.
- **Instalace** – Nástroj je možné přímo stáhnout a naimportovat do projektu nebo je možné využít CDN (*Content Delivery Network*) nebo je možné využít nástroje pro instalaci balíčků (Node Package Manager, Bower), jejichž pomocí lze nástroj snadno automaticky nainstalovat a snadno importovat do projektu.
- **Integrace s aplikačními rámci** – Nástroj je možné integrovat s běžně užívanými aplikačními rámci, jakým je například Angular, React, Ember aj.
- **Přizpůsobitelnost komponent** – Nástroj umožňuje přizpůsobení stylu a typografie responzivních komponent pro výsledný design.
- **Dostupnost tutoriálů** – Na oficiálních webových stránkách [38] nástroje je pouze jednoduchý tutoriál, avšak součástí dokumentace jsou příklady použití responzivní mřížky a responzivních komponent. Podrobnější tutoriál je k dispozici na webu W3Schools [58].
- **Časová náročnost** – Bootstrap je ve své podstatě sada kaskádových stylů a skriptů v jazyce JavaScript. Má-li uživatel tohoto nástroje základní znalosti v HTML 5, CSS a JavaScriptu, tak odhaduji náročnost na osvojení si nástroje na tři až pět dní. Pokud začne knihovnu užívat někdo bez těchto znalostí, pak časová náročnost bude mnohem vyšší.
- **Práce s nástrojem** – Pro podporu responzivní mřížky obsahuje Bootstrap podporu pro čtyři typy zařízení podle šířky obrazovky. Bootstrap podporuje dvanácti sloupcové rozložení webové stránky, kde nastavení rozložení pro dané zařízení je syntakticky rozlišeno a sloupce je možné do sebe zanořovat. Pro responzivní komponenty je třeba užít určité elementy a k nim přidat příslušné CSS třídy. Práce s Bootstrapem je, při osvojení si znalosti tohoto nástroje, velice snadná a intuitivní. Problémem může být přizpůsobování responzivních komponent přímo v aplikaci pomocí CSS nebo preprocesoru, kde je nutné překrýt existující styly komponent. Trvá pouze delší dobu si osvojit koncept správné tvorby responzivní mřížky.
- **Podpora prohlížečů** – Firefox, Chrome, Safari, Internet Explorer 8+, Edge, Opera
- **Licence** – MIT

3.7.3.2 Zurb Foundation

- **Hodnocení vývojářskou komunitou** – Necelých 26 tisíc hvězd na serveru GitHub
- **Velikost komunity** – Více než 900 přispěvatelů, přes patnáct set sledujících vývojářů a tisíce vývojářů užívajících tento nástroj.
- **Princip, koncept** – *Mobile-first*, responzivní webový design (viz podkapitola 2.2) se zaměřením na sémantiku.
- **Velikost nástroje** – 326 kB.
- **Responzivní rozložení a komponenty** – Foundation nabízí podporu pro tvorbu responzivní fluidní mřížky a velkou sadu responzivních komponent a fluidní typografii. Obsahuje následující komponenty – mřížku, flexibilní mřížku, formuláře, tabulky, různé typy tlačítek, posuvníky, přepínače, různé typy nabídek, responzivní navigaci, stránkování, drobečková navigace, záložky, panely, karty, odznaky, popisky, nápovědy, *progress bar,* podpora animací
- **Modularita** – Nástroj je modulární.
- **Užití CSS preprocesorů** – Nástroj užívá CSS preprocesor SASS.
- **Sada vlastních ikon** – Nástroj nabízí sadu vlastních ikon *Foundation Icon Fonts*.
- **Dostupnost základních šablon** - Nástroj nabízí sadu předdefinovaných šablon a témat pro různé typy webů.
- **Dokumentace** – Nástroj má velmi přehlednou dokumentaci k jednotlivým responzivním komponentám, responzivní mřížce, užívaným kaskádovým stylům a skriptům v jazyce JavaScript s velmi dobrou strukturou obsahu.
- **Instalace** – Nástroj je možné přímo stáhnout a naimportovat do projektu nebo je možné využít CDN (*Content Delivery Network*) nebo je možné využít nástroje pro instalaci balíčků (Node Package Manager, Bower), jejichž pomocí lze nástroj snadno automaticky nainstalovat a snadno importovat do projektu.
- **Integrace s aplikačními rámci** – Nástroj je možné integrovat s některými aplikačními rámci – Angular, React.
- **Přizpůsobitelnost komponent** – Nástroj umožňuje přizpůsobení stylu a typografie responzivních komponent pro výsledný design.
- **Dostupnost tutoriálů** – Na oficiálních webových stránkách nástroje [39] je k dispozici série tutoriálů a video tutoriálů.
- **Časová náročnost** – Foundation je stejně jako Bootstrap sada kaskádových stylů a skriptů v jazyce JavaScript, která má velice dobře sémanticky pojmenované třídy. Na druhou stranu obsahuje spoustu komponent a nastavitelného obsahu. Osvojení

tohoto nástroje při znalostech HTML, CSS a JavaScript odhaduji na sedm dní, ale pro začátečníka je snažší porozumět struktuře stránky díky důrazu na sémantiku.

- **Práce s nástrojem** – Pro podporu responzivní mřížky obsahuje Foundation podporu pro tři typy velikosti obrazovek, mezi těmito body zlomu se mřížka chová jako fluidní. Foundation podporuje dvanácti sloupcové rozložení webové stránky, kde nastavení rozložení pro dané zařízení je syntakticky odlišené a sémanticky dobře popsané. Foundation podporuje zanořování sloupců. Pro responzivní komponenty je třeba užít určité elementy a k nim přidat příslušné CSS třídy. Práce s Foundation je, při osvojení si znalosti tohoto nástroje, velice snadná a intuitivní. Mnohem více intuitivní než je práce s Bootstrap. Přizpůsobování komponent lze snadno provádět pomocí změn příslušných proměnných preprocesoru SASS. Nevýhodou je množství obsahu, které je třeba si osvojit pro využití plného potenciálu tohoto aplikačního rámce a také menší komunita vývojářů.
- **Podpora prohlížečů** – Firefox, Chrome, Safari, Internet Explorer 9+, Edge, Opera, Android Browser 2.3+
- **Licence** – MIT

3.7.3.3 Semantic UI

- **Hodnocení vývojářskou komunitou** – Necelých 31 tisíc hvězd na serveru GitHub.
- **Velikost komunity** – Přibližně 140 přispěvatelů, necelých třináct set sledujících vývojářů a tisíce vývojářů užívajících tento nástroj.
- **Princip, koncept** – *Mobile-first*, responzivní webový design (viz podkapitola 2.2) se zaměřením na sémantiku.
- **Velikost nástroje** – 532 kB.
- **Responzivní rozložení a komponenty** – Semantic opět nabízí podporu pro tvorbu responzivní mřížky a velkou sadu responzivních komponent a fluidní typografii**.** Responzivní komponenty vytváří ze základních elementů jazyka HTML pomocí CSS stylů. Dále obsahuje komponenty jako navigaci, několik typů nabídek, různé typy tlačítek, formuláře, tabulky, popisky, karty, zprávy, komentáře, reklamy, statistiky a podporuje velké množství efektů. Výběrem komponent je z porovnávaných nástrojů ten nejpestřejší.
- **Modularita** – Nástroj je modulární.
- **Užití CSS preprocesorů** – Nástroj užívá CSS preprocesor LESS.
- **Sada vlastních ikon** – Nástroj nemá vlastní sadu, používá interně knihovnu *Font Awesome*, která poskytuje v současnosti nejširší výběr ikon.

- **Dostupnost základních šablon** – Nástroj nabízí sadu předdefinovaných šablon pro zjednodušení tvorby webových stránek a responzivní mřížky.
- **Dokumentace** – Nástroj má velmi přehlednou dokumentaci k jednotlivým responzivním komponentám, responzivní mřížce, užívaným kaskádovým stylům s velmi dobrou strukturou obsahu.
- **Instalace** – Nástroj je možné přímo stáhnout a naimportovat do projektu. K instalaci je možné využít Node Package Manager (NPM) pro importování a snadnou instalaci do aplikace.
- **Integrace s aplikačními rámci** – Nástroj je možné integrovat pouze s některými aplikačními rámci – React, Ember, Meteor. Podpora pro Angular je uváděna ve vývoji, ale vývoj se zastavil na jaře roku 2016, očekává se až podpora Angularu 2, který má být vydán v roce 2017. V současnosti podpora Angularu chybí.
- **Přizpůsobitelnost komponent** – Nástroj umožňuje přizpůsobení stylu a typografie responzivních komponent pro výsledný design.
- **Dostupnost tutoriálů** – Na oficiálních webových stránkách nástroje [41] je k dispozici pouze jednoduchý tutoriál pro zprovoznění aplikačního rámce ve webové aplikaci. Ukázky použití responzivní mřížky a komponent jsou přítomny v dokumentaci.
- **Časová náročnost** – Semantic UI je zřejmě nejvíce komplexní aplikační rámec pro návrh responzivních webových stránek. Osvojení tohoto nástroje při znalostech HTML, CSS a JavaScript odhaduji stejně jako Foundation na sedm dní.
- **Práce s nástrojem** – Pro podporu responzivní mřížky poskytuje Semantic UI mřížku, která může mít libovolný počet sloupců a může být definována pro čtyři typy obrazovek různých velikostí. Mezi těmito body zlomu se mřížka chová jako fluidní. Semantic také podporuje různé strategie chování responzivního rozložení a to *stackable*, která sloupce řadí pod sebe, nebo *doubling*, která počet sloupců pro větší zařízení zdvojuje. Je také možné vytvořit vlastní chování responzivního rozložení. Responzivní komponenty se definují pomocí CSS tříd. CSS třídy jsou sémanticky nazvány tak, aby přímo určovaly svůj význam (např. mřížka je definována třídou *ui grid*). Po osvojení nástroje je práce velmi rychlá a snadná a je třeba ocenit velké možnosti přizpůsobitelnosti tohoto nástroje.
- **Podpora prohlížečů** – Firefox, Chrome, Safari, Internet Explorer 10+, Opera, Android Browser 4+
- **Licence** – MIT

3.7.3.4 Pure

- **Hodnocení vývojářskou komunitou** – Necelých 15 tisíc hvězd na serveru GitHub.

- **Velikost komunity** – Přibližně 50 přispěvatelů, necelých sedm set sledujících vývojářů a tisíce vývojářů užívajících tento nástroj.
- **Princip, koncept** – Minimalistický nástroj, architektura SMACSS [43].
- **Velikost nástroje** – 18 kB.
- **Responzivní rozložení a komponenty** – Pure nabízí minimalistické responzivní rozložení a minimalisticky stylované responzivní komponenty. Pure nabízí omezený počet komponent – mřížky, formuláře, tabulky, tlačítka a nabídky.
- **Modularita** – Nástroj je modulární.
- **Užití CSS preprocesorů** – Nástroj nepoužívá preprocesor.
- **Sada vlastních ikon** – Nástroj nemá žádnou sadu ikon. Sadu ikon je třeba řešit importováním knihovny obsahující sadu ikon.
- **Dostupnost základních šablon** – Nástroj nabízí základní sadu šablon pro tvorbu webových stránek.
- **Dokumentace** – Nástroj má přehlednou dokumentaci k jednotlivým responzivním komponentám, responzivní mřížce, užívaným kaskádovým stylům s velmi dobrou strukturou obsahu. Avšak není příliš obsáhlá.
- **Instalace** – Nástroj je možné importovat pomocí služby CDN (Content Delivery Network) nebo pomocí nástrojů pro instalaci balíčků (NPM, Bower).
- **Integrace s aplikačními rámci** – Nástroj je možné integrovat pouze s některými aplikačními rámci – React, Ember. Podpora pro Angular není dostupná. Je možné kombinovat i s Twitter Bootstrap.
- **Přizpůsobitelnost komponent** – Nástroj umožňuje v omezené míře přizpůsobení stylu a typografie responzivních komponent pro výsledný design, ale je třeba vzít v úvahu, že základní design komponent je minimalistický.
- **Dostupnost tutoriálů** – Na oficiálních webových stránkách nástroje [56] je k dispozici pouze jednoduchý tutoriál pro zprovoznění aplikačního rámce a užití responzivní mřížky ve webové aplikaci.
- **Časová náročnost** – Pure je velmi jednoduchý nástroj. Menší problémy může způsobovat jiný styl zápisu velikosti sloupců – není příliš intuitivní. Osvojení si tohoto nástroje vyžaduje pouze jeden až dva dny.
- **Práce s nástrojem** – Pro podporu responzivní mřížky nástroj poskytuje mřížku, která není definována přímo pomocí sloupců, ale pomocí poměru šířky a výšky rodičovského elementu. Poměr šířky je definovat pomocí dvou čísel, první číslo odpovídá čitateli zlomku a druhé jmenovateli. Takto definované spojité oblasti mohou být dále specifikovány pro čtyři různě velké velikosti obrazovek – od mobilních telefonů po monitory. Práce s tímto nástrojem je snadná, ale lze s touto knihovnou pokrýt

pouze tvorbu jednoduchých webových stránek, v ideálním případě s minimalistickým designem.

- **Podpora prohlížečů** – Firefox, Chrome, Safari, Internet Explorer 7+, Opera, Android Browser 4+
- **Licence** – Yahoo! Inc. BSD

3.7.3.5 Materialize

- **Hodnocení vývojářskou komunitou** – Necelých 24 tisíc hvězd na serveru GitHub.
- **Velikost komunity** – Přibližně 150 přispěvatelů, tisíc sledujících vývojářů a tisíce vývojářů užívajících tento nástroj.
- **Princip, koncept** – Responzivní webový design, *material* design (viz podkapitola 2.2).
- **Velikost nástroje** – 1100 kB.
- **Responzivní rozložení a komponenty** – Materialize nabízí 12 sloupcové responzivní rozložení a fluidní typografii. K dispozici jsou také responzivní komponenty ve stylu *material* designu a to: navigace, různé typy nabídek, tabulky, formuláře, popisky, tlačítka, karty, stránkování, kolekce, zápatí, *preloader*, *chips*. Dále nabízí řadu efektů, které jsou součástí komponent a mají svůj specifický význam v rámci *material* designu.
- **Modularita** – Nástroj je modulární.
- **Užití CSS preprocesorů** – Nástroj podporuje preprocesor SASS.
- **Sada vlastních ikon** – *Material Design Icons* obsahující 740 ikon.
- **Dostupnost základních šablon** – Nástroj nabízí základní sadu šablon pro tvorbu webových stránek.
- **Dokumentace** – Nástroj má přehlednou dokumentaci k jednotlivým responzivním komponentám, responzivní mřížce, užívaným kaskádovým stylům s velmi dobrou strukturou obsahu.
- **Instalace** – Nástroj je možné stáhnout přímo z oficiální webové stránky [materialize] nebo je možné importovat pomocí služby CDN (Content Delivery Network) nebo pomocí nástrojů pro instalaci balíčků (NPM, Bower).
- **Integrace s aplikačními rámci** – Nástroj je možné integrovat pouze s některými aplikačními rámci – Ember a Meteor. Podpora pro Angular a React není oficiální, ale vývojářská komunita již tuto podporu rozvíjí a již nyní je k dispozici neoficiální podpora.

- **Přizpůsobitelnost komponent** – Nástroj umožňuje v omezené míře přizpůsobení stylu a typografie responzivních komponent pro výsledný design.
- **Dostupnost tutoriálů** – Na oficiálních webových stránkách nástroje [Materialize] je k dispozici pouze jednoduchý tutoriál pro zprovoznění aplikačního rámce. Kvalitní tutoriály pro tvorbu webových stránek je možné nalézt na serveru tutorialspoint.com [44].
- **Časová náročnost** – Materialize je obsahově i časově náročný podobně jako Twitter Bootstrap, čili tři až pět dnů zabere osvojení si tohoto nástroje.
- **Práce s nástrojem** – Pro podporu responzivní mřížky nástroj poskytuje mřížku, která je definována tradičně pomocí řádků a sloupců. Jak bylo zmíněno výše, užívá se dvanácti sloupcové responzivní rozložení. Materialize používá 3 body zlomu pro malé, střední a velké obrazovky. Práce s tímto nástrojem je intuitivní, ale vyžaduje širší znalosti příslušných CSS tříd pro dané komponenty a efekty. Po osvojení je práce s tímto aplikačním nástrojem velice rychlá a je možné snadno získat dobře vypadající moderní webové stránky**.**
- **Podpora prohlížečů** – Firefox, Chrome, Safari, Internet Explorer 10+, Opera
- **Licence** – MIT

3.7.4 Závěr porovnání

Společnost Unicorn Systems se zabývá převážně tvorbou velkých webových aplikací a portálů v nejrůznějších odvětvích průmyslu a obchodu. Velké webové aplikace obvykle obsahují netriviální komponenty, při jejichž implementaci se občas dojde až na hranice technologických možností užívaných nástrojů a webových prohlížečů, proto je důležité volit komplexnější nástroje s podporou široké vývojářské komunity, jež se mohla se vzniklými problémy během vývoje setkat a vyřešit je. Ze srovnání je patrné, že většina aplikačních rámců má společné rysy a nabízí takřka stejné komponenty pro vývoj webu. Společnost *Unicorn Systems* používá pro vývoj i jiné front-endové aplikační rámce, se kterými je třeba nástroje pro responzivní webové stránky integrovat. Integrace je velice důležitá, protože vyžaduje instalaci integračního modulu do aplikace a tento modul musí být stále aktuální a nejlépe s oficiální podporou. Na základě těchto faktů a výsledků analýzy obecně doporučuji pro vývoj responzivních webových stránek Twitter Bootstrap. Pro vývoj webových stránek s aplikačním rámcem React bych zvolil spíše Semantic UI, který poskytuje širší nabídku komponent a přizpůsobitelné responzivní rozložení webové stránky. Pro vývoj webových stránek v *material* designu bych použil nástroj Materialize, protože jeho použití ušetří designerům a vývojářům velké množství práce s designováním komponent a efektů.

4 Návrh responzivní webové stránky

Jedním z hlavních požadavků ze strany společnosti *Unicorn Systems* na tuto práci bylo vytvoření metodiky pro tvorbu responzivního uživatelského rozhraní informačních systémů. Kapitola 4 tedy shrnuje základy metodiku při návrhu a tvorbě responzivního uživatelského rozhraní. Na základě srovnání v podkapitole 3.7 bude pro vývoj použit nástroj Twitter Bootstrap a proto v této kapitole jsou shrnuty základní metody práce s Bootstrap: tvorba responzivní mřížky, bližší seznámení se s nejčastěji užívanými komponentami, práce s obrázky, typografie.

4.1 Návrh responzivního uživatelského rozhraní

Obecný návrh grafického uživatelského rozhraní je jedna z nejdůležitějších úloh během vývoje informačních systémů. Informační systémy se vyvíjejí převážně pro běžné uživatele a právě uživatelské rozhraní je to, s čím uživatel přichází do styku. Uživatelské rozhraní je tedy klíčový prvek informačního systému, který může rozhodovat o tom, zda si uživatel daný systém oblíbí. Proto je snaha o co nejpřívětivější uživatelské rozhraní. K tomuto cíli slouží analýzy uživatelské rozhraní, analýzy požadavků a uživatelské testování.

Hlavní rysy uživatelského rozhraní informačních systémů:

- Kvalita systému
- Porozumění uživateli
- Efektivní a rychlá tvorba
- Prostor pro snadnou aplikaci změn
- Účelnost a adekvátnost
- Upotřebitelnost
- Použitelnost
- Snadná ovladatelnost
- Uživatelský zážitek

V současnosti se nejčastěji užívá při vývoji grafického uživatelského rozhraní *human-centered* design. Je to styl návrhu rozhraní, do kterého je zahrnut i uživatel a to aktivní či pasivní spoluprácí na vývoji. Snahou je co nejlépe navrhnout rozhraní pro uživatele, případně co nejlépe přizpůsobit systém potřebám uživatelů.

Dobře navržené uživatelské rozhraní by mělo plnit následující vlastnosti:

- Konzistence
- Zpětná vazba při interakci
- Navigace uživatele v dialozích
- Tolerance k chybám uživatele
- Předcházení chybám
- Předvídatelnost
- Respekt k různým typům uživatelů

Při návrhu uživatelského rozhraní se nejčastěji používá obecně metodika GUIDE popsaná Davidem Redmond–Pylem a Alanem Moorem [47]. Tato metodika dělí návrh uživatelského rozhraní do několika fází:

- Stanovení požadavků a požadavky na použitelnost
- Vytvořit modely úloh a scénáře úloh prováděných uživateli
- Vytvořit modely objektů používaných uživateli
- Návrh grafického uživatelského rozhraní – nejčastěji pomocí *wireframe*
- Vytvoření funkčního prototypu
- Hodnocení použitelnosti

Po zapracování změn na základě hodnocení použitelnosti je výsledkem návrhu funkční prototyp rozhraní. Je třeba vzít na vědomí, že se jedná pouze o prototyp. V praxi se obvykle na základě prototypu znovu implementuje uživatelské rozhraní v implementační fázi projektu a prototyp slouží pouze jako předloha.

Při návrhu responzivního uživatelského rozhraní je třeba v prvních fázích návrhu specifikovat minimální množinu úloh a případů použití pro mobilní zařízení. Tuto množinu úloh je možné dále rozšiřovat pro větší zařízení, avšak mělo by platit, že většina případů užití aplikace na různých zařízeních by měla být stejná.

Dalším krokem je specifikace zařízení, která bude webová aplikace podporovat, resp. pro která zařízení se bude přizpůsobovat. Obvykle jsou pokryta veškerá zařízení. Na základě této informace může být navrženo uživatelské rozhraní. Uživatelské rozhraní je třeba navrhnout zvlášť pro každé zařízení a je vhodné, aby při návrhu byl přítomen člověk s technologickým *know-how*, aby mohl specifikovat, jak se chová responzivní rozložení nástroje, jenž bude při vývoji použit. Výstupem této fáze je *wireframe* prototyp, což je náčrt uživatelského rozhraní, který umožňuje základní interakci. Na tomto prototypu se odladí velké množství chyb v interakci a rozložení prvků na obrazovce. Pro tvorbu *wireframe* prototypů existuje celá řada nástrojů, společnost *Unicorn Systems* užívá *Balsamiq Mockup*.

Dále následuje vytvoření funkčního prototypu. Funkční prototyp vypadá takřka jako výsledné uživatelské rozhraní, jsou definovány barvy, konkrétní vzhled komponent a je zde zakomponován i design webové stránky. Opět prototyp je nutné vytvářet pro každé zařízení zvlášť, současné nástroje pro prototypování nemají příliš dobrou podporu pro tvorbu responzivních funkčních prototypů. Funkční prototyp umožňuje interakci s uživatelem a užívá obvykle nějakou testovací sadu dat. Pro prototypování existuje spousta nástrojů, designéři ve společnosti *Unicorn Systems* obvykle používají nástroj *Axure*.

Poslední fáze vyhodnotí navržené rozhraní z hlediska použitelnosti. Užitím formálních ověřovacích metod a pomocí uživatelského testování. Na základě získané zpětné vazby z hodnocení použitelnosti a uživatelského testování se velmi často ještě funkční prototyp upravuje tak, aby eliminoval vzniklé chyby v uživatelském rozhraní. Tento krok obvykle mívá více iterací, než se dojde k dobrému výsledku. Výstupem této fáze je tedy hodnocení a upravený funkční prototyp. Poté již následuje fáze implementační.

4.2 Tvorba responzivní webové aplikace

Na základě analýzy a návrhu responzivní aplikace vzniká funkční prototyp aplikace. Jak bylo zmíněno v podkapitole 4.1, pro každé podporované zařízení vzniká samostatný prototyp. Pro tvorbu responzivní webové stránky je vhodné použít přístup *mobile-first*, popsaný v sekci 2.2.7. Aplikace tedy bude vyvíjena prvně pro nejmenší zařízení a poté je upravováno responzivní rozložení webové stránky podle velikosti obrazovky. V prvním kroku je třeba zvolit vhodný nástroj a rozmyslet si a poté definovat responzivní rozložení a responzivní mřížku, která určí základní strukturu aplikace. V tomto kroku je vhodné zároveň nastavit typografii, kterou bude webová stránka používat. Je třeba vzít v úvahu podporovaná zařízení a podporované webové prohlížeče hlavně proto, aby byla stránka realizovatelná.

Dalším krokem je vytvoření či úprava responzivních komponent tak, aby vizuálně byly totožné s komponentami užitými ve funkčním prototypu. Toho je obvykle možné docílit přizpůsobením kaskádových stylů pomocí nástrojů pro přizpůsobení, které jsou obvykle k dispozici na oficiálních stránkách nástrojů. Druhým způsobem, jak upravit komponenty, je překrytím kaskádových stylů Bootstrapu. To ale vyžaduje hlubší znalost kaskádových stylů kvůli prioritám selektorů. Pokud je daná komponenta designovaná vícenásobně, pak je nutné také dbát na správnou prioritu a sekvenci selektorů, protože pak může dojít k nechtěnému přestylování všech komponent. V tomto kroku tedy dochází k tvorbě sady komponent, jejichž pomocí je aplikace pak velice rychle sestavena.

Třetím krokem je samotná aplikace komponent do sloupců responzivního rozložení a řešení adaptivních obrázků. Jinými slovy, v tomto kroku je vyplněn obsah webové stránky. V této fázi dochází i k úpravám responzivního rozložení a je možné v této fázi rozložení upravit specificky pro dané zařízení pomocí dotazů na média.

Dosud je připravena pouze statická část webové stránky, v dalším kroku následuje přidání JavaScriptu, který se postará o to, aby byla výsledná webová stránka dynamická a měla nějakou funkcionalitu.

Posledním krokem je fáze ověření funkcionality proti zadání a funkčnímu prototypu. Toho se většinou docílí testováním webové aplikace. Toto testování zpravidla probíhá formou testování podle testovacích scénářů, které by měly pokrýt veškerou požadovanou funkčnost a vizuálně by mělo být ověřeno, zda je aplikace správně implementovaná proti funkčnímu prototypu. Pokud je webová aplikace příliš komplexní, je vhodné užít jednotkové a integrační testy v průběhu vývoje, které zefektivní a zkvalitní proces vývoje tak, aby poslední fáze neobsahovala zbytečné chyby v implementaci a tak se mohly řešit převážně jen úpravy uživatelského rozhraní.

4.3 Základy práce s Twitter Bootstrap

Tato podkapitola stručně shrnuje práci s nástrojem Twitter Bootstrap, jenž byl použit pro implementaci praktické části práce na základě analýzy v podkapitole 3.7. V následujícím textu je kladen důraz na správnou tvorbu responzivní mřížky, která je nejzásadnější komponentou při tvorbě responzivní webové stránky.

4.3.1 Tvorba responzivní mřížky

Twitter Bootstrap nabízí responzivní fluidní *mobile-first* dvanácti sloupcový mřížkový systém pro tvorbu responzivního rozložení webové stránky. Obsahuje předdefinované CSS třídy, které se obvykle navazují na kontejnerové elementy jazyka HTML, případně HTML 5 (Např. , <main>, <section> atd.). Pro základní definici mřížky se užívají třídy: *.container, .container-fluid, .row, .col-.*

Pro tvorbu responzivní mřížky platí několik zásad a dobrých praktik:

- Důležité je dodržování dvanácti-sloupcového rozložení v každém řádku.
- Řádky mřížky musejí být umístěny do kontejneru s třídou *.container* mající fixní šířku mřížky podle šířky obrazovky, nebo do kontejneru s třídou *.container-fluid*, který má maximální šířku, obvykle celou šířku obrazovky.
- Sloupce je nutné organizovat do řádků, každý řádek by tedy měl obsahovat dvanáct sloupců.

- Obsah webové stránky by měl být umístěn do sloupců.
- Pomocí třídy *.row* je definován řádek responzivní mřížky.
- Pomocí třídy *.col* je obecně definován sloupec. Plný zápis však vyžaduje ještě dodatečné modifikátory pro typ zařízení a počet sloupců tvořící spojitou část. Kupříkladu *col-md-4* je obecně definována spojitá část o čtyřech sloupcích pro menší monitory stolních počítačů.
- Mezi jednotlivými spojitými částmi v jednom řádku vznikají mezery, resp. jde o odsazení mezi dvěma spojitými částmi. Toto odsazení je pro první a poslední spojitou část v řádku negováno pomocí třídy *.row*, která obsahuje negativní hodnotu okraje. Proto je nutné, aby byly sloupce uvnitř řádků, protože by mohlo jinak dojít k neočekávanému chování responzivní mřížky.
- Pokud řádek obsahuje více než dvanáct sloupců, pak dojde k přesunu spojité části nebo i více částí, které tvoří větší než dvanácti sloupcové rozložení, na další řádek tak, aby nebylo porušeno toto rozložení. Je dobrou praktikou toto rozložení striktně dodržovat, aby nedocházelo k chybám.
- Modifikátory pro zařízení, které se přidávají do definice sloupců, se aplikují na obrazovky, které odpovídají velikosti stejné nebo rovné velikosti bodu zlomu pro dané zařízení. Proto je výhodné použít při tvorbě responzivní mřížky přístup *mobile-first*.
- Řádky a sloupce je možné zanořovat do sebe.
- Pro jeden element je možné definovat více tříd sloupců, ale pro různá zařízení.
- Sloupce je možné posouvat pomocí modifikátoru třídy sloupce *offset.* Například pro mobilní zařízení může vypadat posun o tři sloupce následovně *col-xs-offset-3.*
- Pro úpravu viditelnosti a skrývání prvků pro dané zařízení slouží třídy *.visible-* a *.hidden-*, ke kterým se přidává modifikátor označující zařízení. Na obrázku 31 se nachází přehled nastavení viditelnosti pro různá zařízení.

Classes	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
.visible-xs-*	Visible	Hidden	Hidden	Hidden
.visible-sm-*	Hidden	Visible	Hidden	Hidden
.visible-md-*	Hidden	Hidden	Visible	Hidden
.visible-lg-*	Hidden	Hidden	Hidden	Visible
.hidden-xs	Hidden	Visible	Visible	Visible
.hidden-sm	Visible	Hidden	Visible	Visible
.hidden-md	Visible	Visible	Hidden	Visible
.hidden-lg	Visible	Visible	Visible	Hidden

Obrázek 31: Přehled viditelnosti pro různá zařízení. Převzato z [58].

Následující obrázek 33 přehledně znázorňuje základní vlastnosti a nastavení responzivní mřížky v Bootstrap pro různé typy zařízení.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
# of columns	12			
Column width	Auto	~62px	~81px	~97px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

Obrázek 32: Základní informace o nastavení responzivní mřížky v nástroji Twitter Bootstrap. Převzato z [58].

Příklad responzivní mřížky:

```
<div class="container">
 <div class="row">
       <div class=" col-xs-12 col-sm-6 col-md-4"></div>
       <div class=" col-xs-12 col-sm-6 col-md-4"></div>
       <div class="hidden-xs hidden-sm col-md-4"></div>
 </div>
 <div class="row">
       <div class=" col-xs-12 col-sm-6 col-md-4"></div>
       <div class=" col-xs-12 col-sm-6 col-md-4"></div>
       <div class="hidden-xs hidden-sm col-md-4"></div>
 </div>
</div>
```

Integrovaný Bootstrap do Angularu používá responzivní mřížku shodně.

4.3.2 Fluidní typografie

Bootstrap má předdefinovanou fluidní typografii se základní velikostí fontu 14 pixelů. Zároveň jsou předdefinovány velikostí nadpisů, které jsou dostupné pomocí tříd *.h1* až *.h6*, případně relevantních elementů jazyků HTML a HTML 5. Bootstrap podporuje typografii upravenou pomocí inline HTML elementů. Dále poskytuje speciální třídu *.lead* pro úpravu textu odstavce a to tak, že text v odstavci zvětší a zvýrazní. Úpravu zarovnání textu Bootstrap řeší pomocí tříd *.text-left*, *.text-center, .text-right, .text-justify, .text-nowrap*. Další změny písma je možné provést standardně pomocí úprav kaskádových stylů. Integrovaný Bootstrap do Angularu používá fluidní typografii stejným způsobem.

4.3.3 Práce s obrázky

Práce s obrázky v Bootstrapu je velice snadná stačí k obrázku v HTML dokumentu přidat třídu *.img-responsive.* Ta aplikuje interně sadu kaskádových stylů, které odpovídají první technice pro adaptivní obrázky popsané v sekci 3.3.3. Pokud je nutné centrovat obrázek v rámci nějakého sloupce mřížky, pak k řešení tohoto Bootstrap poskytuje třídu *.center-block*. Pokud dochází u obrázků k degradaci kvality, pak je nutné užít element <picture> a jeho atribut *media* určený pro dotaz na medium. Níže je uveden příklad tohoto řešení pro dvě různá zařízení a obrazovky s různě velkou hustotou pixelů:

```
<picture>
 <source
   media="(min-width: 992px)"
   srcset="images/image-medium.png,
           images/image-medium@1.5x.png 1.5x,
           images/image-medium@2x.png 2x">
 <source
   media="(min-width: 768px)"
   srcset="images/image-small.png,
           images/image-small@1.5x.png 1.5x
           images/image-small@2x.png 2x">
 <img
   src="images/image.png"
   srcset="images/image@1.5x.png 1.5x,
           images/image@2x.png 2x"
   alt="example of responsive image">
</picture>
```

4.3.4 Responzivní komponenty

Bootstrap nabízí velké množství responzivních komponent. Komponenty se aplikují do sloupců responzivního rozložení. V této sekci budou přestaveny vybrané nejčastěji užívané responzivní komponenty a jejich protějšky v modulu UI Bootstrap. Ve velkých informačních systémech, jejichž front-endová část je budována jako *single-page* aplikace, jsou nejvýznamnějšími komponentami panely, formuláře a tabulky. Tyto vybrané komponenty budou představeny, ostatní komponenty lze dohledat v dokumentaci Bootstrapu [38]a UI Bootstrapu [46].

4.3.4.1 Formuláře

Bootstrap pracuje se všemi elementy formulářů z HTML 5, pro jednotnost stylu používaných elementů aplikuje na tyto elementy kaskádové styly pomocí třídy *.form-control*, která se aplikuje na elementy <input>,<textarea> a <select> a nastavuje těmto elementům 100% šířky. Jednotlivé prvky formuláře lze od sebe vertikálně oddělit užitím třídy *.form-group*, která se aplikuje na element obalující prvky formuláře.

Pro samotný formulář Bootstrap nabízí dva předdefinované typy formulářů. První typ je jednořádkový formulář, který je získán aplikací třídy *.form-inline* na element <form>. Tento typ formuláře zarovnává všechny své prvky do jednoho řádku. Druhým typem je horizontální formulář, který je získán aplikací třídy *.form-horizontal* na element <form>. Horizontální formulář upravuje chování formuláře tak, že všechny prvky uvnitř elementu s třídou *.form-group* jsou zarovnány do řádku, resp. se poté třída *.form-group* chová podobně jako třída *.row*. Bootstrap podporuje také různé velikosti prvků formuláře, nastavení velikosti prvků lze docílit pomocí tříd: *.input-xs, input-sm, .input-md, .input-lg*, které se aplikuje na elementy s třídou *.form-control*.

Dále jsou podporovány některé stavy prvků formuláře a to: *read-only, disabled* a *focus*. Stav *read-only* lze dosáhnout užitím bezparametrického atributu *read-only* k danému prvku. Pro stav *disabled* je nutné nastavit bezparametrický atribut *disabled*. Pro *focus* je nutné upravit kaskádový styl pro třídu *.form-control* a jeho vlastnosti *:focus*. Bootstrap umožnuje prvkům formuláře nastavit i validační stavy a to pomocí tříd : *.has-success, .has-error, .has-warning*, které se aplikují na element obsahující třídu *.form-group.*

Integrovaný Bootstrap nemá oproti běžnému aplikačnímu rámci žádné odlišnosti.

Příklad formuláře:

```
<form class="form-horizontal">
  <div class="form-group">
    <label for="inputEmail3"
       class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
       <input type="email" class="form-control"
       id="inputEmail3" placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword3"
       class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
       <input type="password" class="form-control"
       id="inputPassword3" placeholder="Password">
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Remember me
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">
       Sign in</button>
    </div>
  </div>
</form>
```

4.3.4.2 Tabulky

Pro tabulky definici tabulek Bootstrap využívá běžnou strukturu jako v jazyce HTML užitím elementu <table>. Bootstrap nabízí pro tabulky jednotné stylování pomocí třídy *.table*, která se zapisuje k výše uvedenému elementu. To ale nestačí k tomu, aby byla tabulka responzivní. K tomu je třeba přidat k elementu obalujícím tabulku třídu *.table-responsive*. S touto třídou se pak při zobrazení na menších zařízeních zobrazí pouze viditelná část tabulky a horizontální posuvník. Pro prohlížení skryté části stačí jen použít u dané tabulky posuvník. Tabulky nabízejí i některé úpravy stylů tabulek. Pro zvýraznění každého druhého řádku se užívá třída *.table-striped,* pro zvýraznění řádku, kterém se nachází kurzor myši, se užívá třída *.table-hover*, pro přidání ohraničení tabulky, resp. její mřížky, slouží třída

.table-bordered. Všechny tyto třídy se aplikují na element <table>. Níže se nachází jednoduchý příklad responzivní tabulky:

```
<div class="table-responsive">
  <table class="table table-hover table-bordered table-striped">
       <thead>
             <tr>
             <th>header</th>
             <th>header</th>
             </tr>
       </thead>
       <tbody>
             <tr>
                   <td>content</td>
                   <td>content</td>
             </tr>
       </tbody>
  </table>
</div>
```

4.3.4.3 Panely

Panely jsou vizuálně rozlišené fluidní kontejnery, do kterých je možné vkládat libovolný obsah. Je možné nastavit jejich záhlaví a zápatí. Obsah uvnitř panelu lze skrýt pomocí JavaScriptu. Panely jsou realizovány třídou *.panel* a třídou určující typ panelu, nejčastěji se užívá třída *.panel-default.* Obě třídy se aplikují na element . Pro nastavení záhlaví, těla a zápatí se užívají třídy *.panel-header, .panel-body a panel-footer*. Níže se nachází příklad definice jednoduchého panelu:

```
<div class="panel panel-default">
  <div class="panel-header">Panel header</div>
  <div class="panel-body">Panel content </div>
  <div class="panel-footer">Panel footer</div>
</div>
```

V UI Bootstrap je panel definován jiným způsobem. UI Bootstrap poskytuje angularovou direktivu <uib-accordion>. Tato direktiva obsahuje vnořenou direktivu <uib-accordion-group>, u které je nutné nastavit pouze třídu určující typ panelu. Do této direktivy se vkládá obsah, resp. tělo daného panelu. Pro definici záhlaví UI Bootstrap nabízí direktivu <uib-accordion-heading>. Níže je uveden jednoduchý příklad:

```
<uib-accordion>
 <div uib-accordion-group class="panel-default">
      <uib-accordion-heading>Hello </uib-accordion-heading>
      World
 </div>
</uib-accordion>
```

5 Analýza webové aplikace

Tato kapitola je věnována analýze praktické části této práce – analýze *front-end* aplikace pro internetové bankovnictví. Je zde stručně uvedena analýza problému z technologického pohledu, dále zde byly sepsány požadavky na aplikaci kladené vývojáři společnosti *Unicorn Systems.* Následně byly stručně analyzovány případy užití a na jejich základě vytvořena základní navigace a návrh webové aplikace.

5.1 Analýza problému

Jedním z cílů práce je navrhnout a implementovat front-end aplikaci internetového bankovnictví tak, aby výsledná webová aplikace byla responzivní a přizpůsobovala se všem typů zařízení. Pro implementaci byly zvoleny technologie HTML5, JavaScript a Angular. Výsledkem analýzy z podkapitoly 3.7 je doporučení pro užití aplikačního rámce Twitter Bootstrap (viz podsekce 3.7.3.1), který má pro Angular velmi dobrou podporu a velice širokou vývojářskou komunitu. Bootstrap však bude třeba integrovat do Angularu pomocí dodatečného modulu UI Bootstrap. Jelikož se práce zabývá užitím responzivního designu v prostředí moderního designu, zvolil jsem a navrhnul jsem vzhled výsledné aplikace s užitím *flat* designu (viz sekce 2.2.1).

První fází analýzy byla analýza technologie a metodiky, která je popsána v kapitolách 3 a 4. Další fází bylo analyzovat požadavky na aplikaci a konzultovat funkcionalitu aplikace s vývojáři společnosti *Unicorn Systems.* Na základě této analýzy byla provedena analýza případů použití aplikace pro internetové bankovnictví.

5.2 Analýza požadavků

Na základě konzultací s vývojáři *Unicorn Systems* byly stanoveny následující požadavky na vytvoření responzivní aplikace:

- Webová aplikace bude vyvíjena na technologiích HTML 5, CSS, AngularJS.
- Na základě srovnání nástrojů pro tvorbu responzivních webových stránek zvolit nástroj, který má širokou komunitu vývojářů, pestrou nabídku komponent a je snadno integrovatelný do angularových aplikací.
- Webová aplikace bude responzivní.

- Webová aplikace bude vyvíjena jako *single-page* aplikace.
- Použití nástroj NPM pro instalaci nástrojů a pomocí knihovny RequireJS bude vyřešen import knihoven.
- Pro vývoj bude nakonfigurován a použit *task manager.*
- Aplikace bude lokalizovaná do dvou jazyků.
- Jelikož je front-end aplikace pro elektronické bankovnictví velice rozsáhlý, tak bude vybráno 5 případů užití a ty budou plně implementovány od *view* až po komunikaci se serverem. Komunikace se serverem bude řešena pomocí angularové služby $httpBackend, která bude na požadavky na server reagovat a bude vracet testovací data. Ostatní případy užití budou implementovány pouze staticky.
- Budou vytvořena testovací data pro ověření funkcionality užívaných komponent.
- Aplikace bude obsahovat přihlašovací formulář a jeho zjednodušenou funkci opět řešenou pomocí služby $httpBackend.
- Aplikace bude užívat preprocesor LESS.
- Aplikace musí obsahovat implementaci komponent pro výběr, výběr datumu, přehledové grafy, tabulky, formuláře i s validací.
- Aplikace bude užívat nástroj pro sloučení JavaScriptových skriptů a zkompilovaných CSS souborů do jednoho skriptu a CSS souboru, tzv. *bundler.*

5.3 Analýza případů použití

Tato podkapitola je věnována analýze internetového bankovnictví a případů užití. Internetové bankovnictví je nástroj, který zprostředkovává uživateli Internetu vlastnícímu účet v bance bankovní služby. Díky tomu, nemusí klienti bank neustále navštěvovat pobočku a velké množství operací jsou schopni vyřídit z jakéhokoliv počítače připojeného k Internetu. Většina bank poskytuje pro mobilní zařízení oddělené aplikace a přístup z jiných zařízení řeší pouze fluidním zmenšováním nebo zvětšováním až do určitých mezí. Stránky tedy nejsou příliš responzivní. Cílem této práce je naimplementovat responzivní front-end pro internetové bankovnictví, jinými slovy klientskou část webové aplikace pro internetové bankovnictví s využitím některého moderního designu.

Základní případy užití v aplikaci lze členit podle frekvence jejich užívání. Jinými slovy, nejrychleji dostupné případy užití by měly být ty, které jsou nejčastěji používané. Dále by dělení mělo být podle povahy daných služeb, např. veškeré případy užití týkající se správy

účtů by měly patřit do této sekce. Tento přístup prokázala analýza několika existujících bankovních řešení.

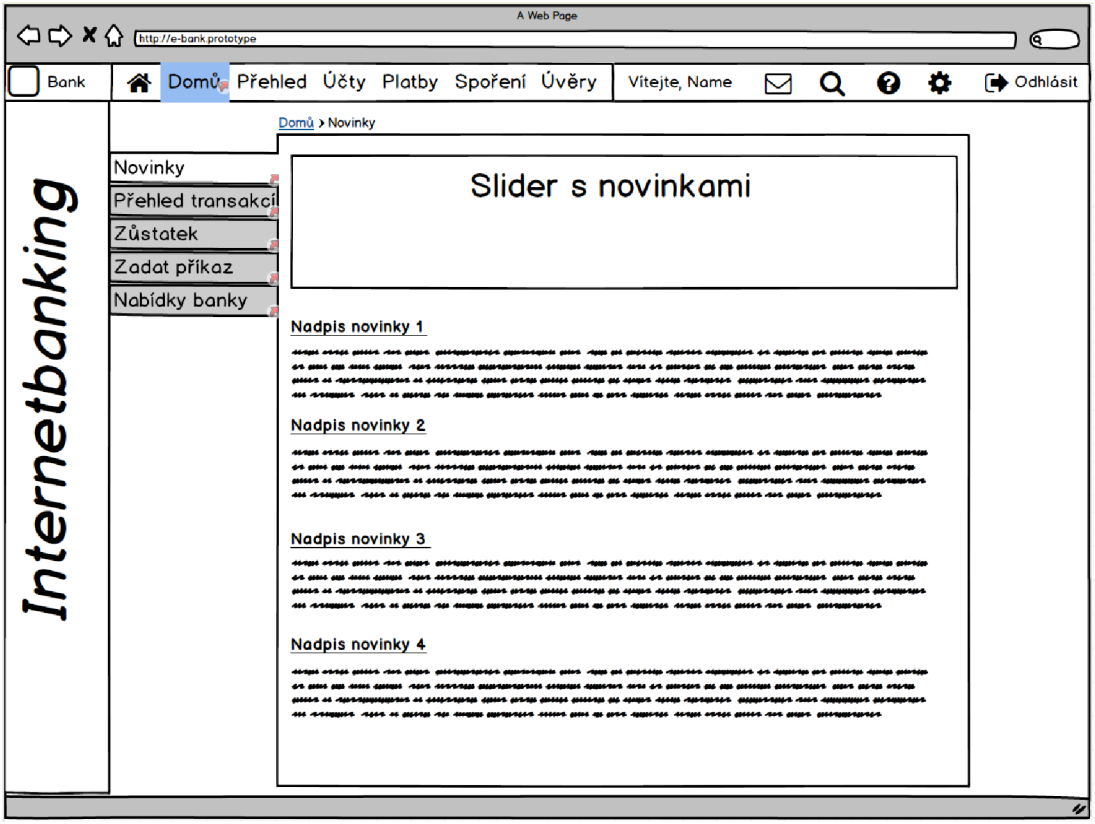
Důležitým krokem je tedy správně navrhnout navigaci aplikace internetového bankovnictví. Domovská stránka by tedy měla obsahovat často užívané případy užití. Mezi ně patří přehled účtů, přehled transakcí, přehled zůstatků a platební příkaz. Často se však banka snaží uživateli poskytnout přehled novinek a různé nabídky bankovních produktů. Dalším krokem je definovat hlavní sekce aplikace podle služeb. Ty jsem definoval podle typu služeb a to jako: účty, platby, spoření a úvěry. Každá taková sekce obsahuje ještě sekundární nabídku, která odpovídá případům užití, které přísluší dané sekci.

Navrhovaná navigace:

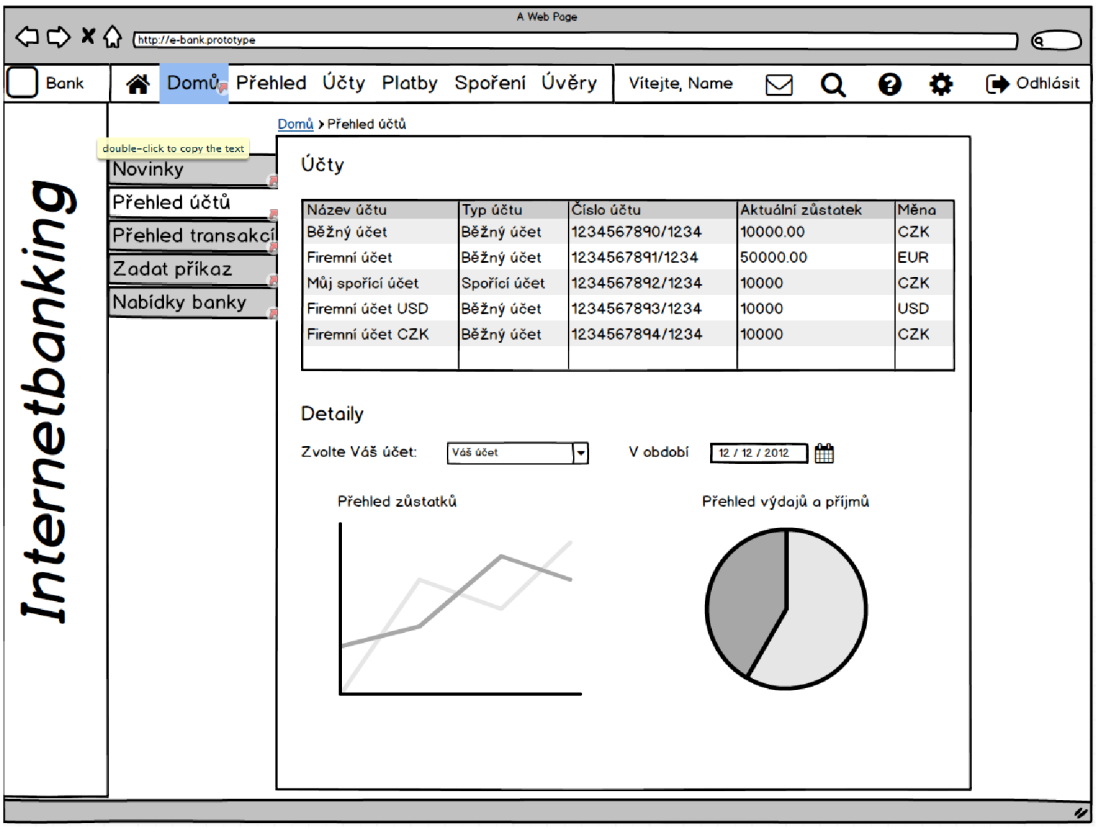
```
Home:
   • News
   • Account overview
   • Transaction overview
   • Balance
   • Payment order
   • Bank offers
Accounts:
   • General overview
   • Account overview
   • Transaction overview
   • Balance
   • Card overview
   • Account settings
   • Card settings
Payments:
   • Payment order
   • Standing order
   • Mobile payments
   • Order templates
Savings:
   • Saving overview
   • Saving transactions
   • Withdrawal
   • Saving offers
Loans:
   • Loan overivew
   • Loan transactions
   • Contracts and documents
   • Simulations
   • Loan offers
```

Jednotlivým podsekcím navigace odpovídají případy užití v aplikaci. Plně implementované budou případy užití na domovské stránce a ze sekce účtů je implementován obecný přehled a přehled karet. Zbytek případů užití je pouze statický, ale stále responzivní.

Po vytvoření navigace jsem přešel k návrhům jednotlivých případů užití, výstupem byla série *wireframe* obrazovek s touto navigací (viz Obrázky 33 a 34) a základním rozložením prvků. Avšak při řešení práce se implementace od těchto návrhu mírně liší, protože šlo o prvotní návrhy.



Obrázek 33: Jedna z počátečních *wireframe* obrazovek s navrhovanou aplikací – sekce novinek



Obrázek 34: Jedna z počátečních *wireframe* obrazovek s navrhovanou aplikací – sekce přehledu účtů

Za zmínku ještě stojí návrh testovacích dat, která počítají ve většině případů se stránkováním dat v tabulkách čí přehledech. Některé sekce aplikace mohou obsahovat velká množství dat, například přehled transakcí. To by znamenalo příjem velkého množství dat, což může mít vliv na výkon aplikace. Řešení v praxi vypadá tak, že se stránkování řeší na straně serveru. A dotazy na server vypadají tak, že se odešle požadavek o data s informací, jakou stránku a kolik položek je třeba navrátit. Aby bylo možné připravit stránkování, je nutné znát počet položek na jedné stránce a celkový počet položek. Z toho důvodu nemohou být odeslána data pouze jako pole daných položek. Proto bylo třeba vytvořit univerzální a vhodnou reprezentaci v JSON formátu.

Ukázka JSON pro data v aplikaci:

```
{
 data:[… array of items …],
 page: 1,
 itemsPerPage:5,
 totalItems:15
}
```

6 Implementace webové aplikace

Praktická část práce je věnovaná implementaci webové aplikace internetového bankovnictví. Tato kapitola stručně shrnuje použité nástroje, architektury, poté uvádí některé implementačně zajímavé ukázky z výsledné webové aplikace, jako například přesměrování mezi stránkami, grafy nebo ukázky responzivity aplikace.

6.1 Použité nástroje

Pro implementaci této webové aplikace byla použita celá řada nástrojů a technologií, tato podkapitola obsahuje jejich stručný popis.

6.1.1 HTML, JavaScript a JSON

Oba jazyky byly představeny již v kapitole 2. Při vývoji aplikace bylo použito HTML 5 pro tvorbu statické části webové stránky. JavaScript byl použit pro dynamickou část webu společně s aplikačním rámcem AngularJS. JSON (JavaScript Object Notation) je odlehčený formát pro výměnu dat. Je jednoduše čitelný i zapisovatelný člověkem a snadno analyzovatelný i generovatelný strojově. Je založen na podmnožině programovacího jazyka JavaScript [48].

6.1.2 AngularJS

AngularJS [45] je strukturální aplikační rámec určený pro vývoj dynamických webových stránek. Umožňuje užití HTML pro tvorbu šablon a rozšíření HTML skrze jeho syntaxi tak, aby stručně a jasně vyjádřil definované komponenty užité ve webové aplikaci. Datové vazby (*data binding*) a *dependency injection* eliminují velké množství kódu, které by bylo nutné napsat. To vše se děje uvnitř webového prohlížeče a dělá tak z Angularu dobrého společníka kterékoliv serverové technologii [45].

6.1.3 Twitter Bootstrap

Vytvořit responzivní webovou stránku nebo aplikaci lze snadno s použitím aplikačního rámce. Pro tento účel na základě porovnání v kapitole 3 byl zvolen Twitter Bootstrap [38]. Detailnější informace o tomto aplikačním rámci viz kapitola 3.

6.1.4 NPM

NPM (Node Package Manager) [59] je nástroj pro správu balíčků umožňující snadnou instalaci různých front-endových knihoven a aplikačních rámců. Pro jeho použití je třeba mít na svém zařízení nainstalovanou aplikaci Node.js, která tento nástroj zprostředkovává. Používané knihovny se ukládají do souboru package.json, kde je možné nastavit dané aplikaci dodatečné informace a spravovat verze instalovaných knihoven. Díky NPM, tak není třeba při stažení zdrojových kódů webové aplikace mít u webové aplikace nakopírované všechny knihovny. O dodatečné stažení správných knihoven se postará NPM jedním příkazem *npm install*.

6.1.5 LESS

LESS [49] je CSS preprocesor, který rozšiřuje jazyk CSS o nové rysy jakými jsou např. proměnné, mixiny (speciální konstrukce pro snadnou kombinaci kaskádových stylů), funkce a spoustu dalších technik, které umožňují, aby CSS byly snáze spravovatelné, aby bylo možné snáze vytvářet grafická témata, a umožňují tak posouvat hranice CSS [49].

6.1.6 Browserify

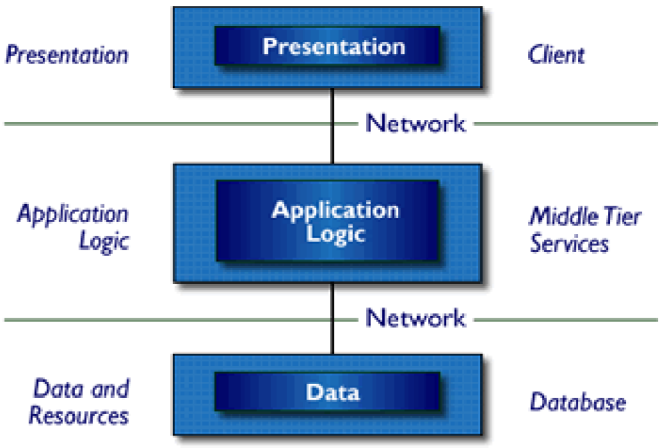
Browserify [60] je nástroj, který rekurzivně analyzuje všechna volání funkce *require()* ve zdrojovém kódu webové aplikace. Pro její užívání je třeba mít nainstalovaný Node.js. Pomocí této funkce se vytváří moduly a modulární aplikace. Z těchto modulů se poté sestaví webová aplikace a výstupem je tak pouze jediný soubor obsahující veškerý kód jazyka JavaScript užitý v dané aplikaci. Pro importování kódu tedy stačí pouze jediný element <script> namísto vícenásobného použití.

6.1.7 Grunt

Grunt [61] je nástroj pro automatické spouštění a správu neustále se opakujících úkolů v průběhu vývoje webové aplikace. Práce s Gruntem vypadá tak, že se do aplikace musí nainstalovat příslušné úkoly, které jsou k dispozici formou modulů. Tyto moduly je poté třeba nakonfigurovat v souboru Gruntfile.js. Příklad takového úkolu může být kompilace LESS souborů nebo spuštění browserify při každé změně zdrojového kódu. Samotné spuštění závisí na konfiguraci, ale obvykle je Grunt spouštěn jediným příkazem z příkazové řádky.

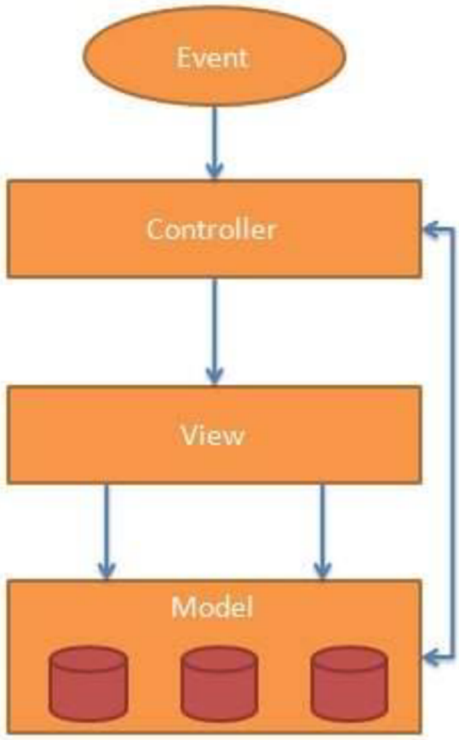
6.2 Architektura informačních systémů

Informační systém je soubor lidí, technických prostředků a metod (programů), zabezpečujících sběr, přenos, zpracování, uchování dat, za účelem prezentace informací pro potřeby uživatelů činných v systémech řízení. Společnost *Unicorn Systems* vytváří pro své zákazníky velké podnikové aplikace, které využívají třívrstvou architekturu. Třívrstvá architektura (obrázek 35) dělí aplikace na tři vrstvy: prezentační vrstva, vrstva aplikační logiky a databázová vrstva. Prezentační vrstva je část aplikace, kterou uživatel vidí a zajišťuje příjem požadavků a prezentaci výsledků. Podoba této vrstvy je závislá na platformě. Může jí být například webová aplikace, mobilní aplikace nebo desktopová aplikace. Vrstva aplikační logiky zajišťuje operace mezi vstupně-výstupními požadavky a daty. Databázová vrstva zajišťuje práci s daty, tedy základní řízení databáze a základní datově-funkční operace zajišťující ukládání, výběr, agregaci, předzpracování, integritu a audit dat.



Obrázek 35: Třívrstvá architektura informačních systémů. Převzato z [62].

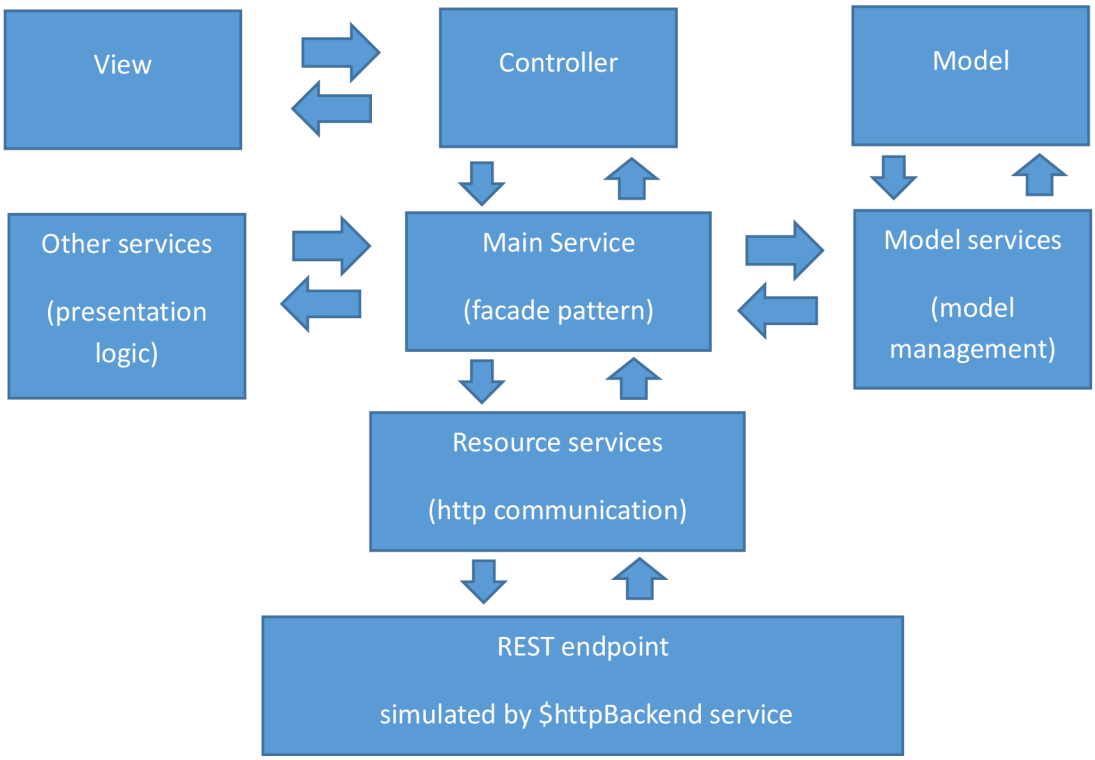
Front-end aplikace odpovídá prezentační vrstvě informačního systému. V případě této práce je implementována webová aplikace za pomocí aplikačního rámce Angular. Angular užívá pro implementaci návrhový vzor MVC (*Model View Controller*). Schéma MVC architektury viz Obrázek 36. Skládá se tedy ze tří částí: *model,view* a *controller. Model* je nejnižší vrstvou tohoto návrhového vzoru a je zodpovědný za správu dat. *View* se stará o zobrazení a prezentaci daných informací uživateli a *controller* je kód, který zajišťuje interakci mezi *view* a *model*.



Obrázek 36: MVC architektura aplikačního rámce Angular. Převzato z [63].

6.3 Architektura navržené webové aplikace

Aplikace je ve svém základu navržena podle MVC architektury angularových aplikací, je třeba si uvědomit, že se tato architektura aplikuje pro každý případ užití v aplikaci. Tento návrhový vzor je příliš obecný a díky tomu existuje v Angularu spousta možností, jak vyvíjet takovou aplikaci. Většinou se to týká práce s modelem a kontrolerem, které je možné rozvíjet za pomocí takzvaných služeb, které se užívají pro vývoj prezentační logiky a komunikaci se serverem skrze http protokol a REST webové služby. Následující schéma zobrazuje architekturu navržené webové aplikace.



Obrázek 37: Architektura jednotlivých případů užití implementované webové aplikace.

Implementace takové architektury je výhodná díky snadné udržovatelnosti kódu. Ta bývá problematická zejména u aplikaci mající komplexnější prezentační logiku. Tato architektura rozvíjí komunikaci, kterou zprostředkovává kontroler. V praxi to to vypadá tak, že kontroler komunikuje pouze s jednou službou, která zaštiťuje veškeré služby, které jsou dostupné pro daný případ užití. Tato hlavní služba spolupracuje interně se službami pro prezentační logiku, pro úpravu modelu a ukládání dat a se službami zprostředkující komunikaci se serverem skrze http protokol. REST koncový bod je přístupovým bodem serverové části aplikace. Ten je v aplikaci simulován pomocí služby $httpBackend, která poslouchá na dané URL adrese a pokud je daná URL adresa zavolána pomocí http protokolu, pak podle kontextu vrátí odpověď, obvykle s daty nebo chybovým hlášením. Odpověď s daty je zpracována pomocí modelové služby, model je uložen a odeslán kontroleru, který data zpropaguje do *view*. Modelová služba je výhodná, protože obsahuje konstrukční funkce objektů, které daný případ užití používá. Tím pádem je pod kontrolou práce s objekty jazyka JavaScript. Pokud poté dojde ke změně nějakého objektu na straně serveru, pak stačí upravit konstrukční funkci a obvykle její zobrazení ve *view*.

6.4 Implementace

Podkapitola se věnuje tématům týkajících se implementace výsledné aplikace. Je zde shrnuta instalace nástroje Twitter Bootstrap a poté jsou předvedeny ukázky zdrojových kódů implementačně zajímavých části výsledné webové aplikace.

6.4.1 Instalace a integrace Twitter Bootstrap

Tato sekce popisuje dostupné možnosti instalace nástroje Twitter Bootstrap a popisuje způsob, jakým tento aplikační rámec lze integrovat do aplikačního rámce AngularJS.

6.4.1.1 Instalace Twitter Bootstrap

První možností instalace Twitter Bootstrap do webové aplikace je stažením z oficiální webové stránky [38]. Zde jsou dvě možnosti instalace. První možností je stáhnout kompilované a minifikované kaskádové styly, skripty v jazyce JavaScript a fonty. Druhou možností je stažení zdrojových kódu tohoto aplikačního rámce včetně příslušných kaskádových stylů a fontů. Instalace se provede tak, že je stažený obsah nakopírován do adresářové struktury aplikace a poté je třeba importovat příslušné skripty jazyka JavaScript a kaskádové styly do HTML dokumentu. Součástí obou instalačních balíčků jsou i soubory CSS preprocesoru LESS. Pokud vývojář preferuje použití preprocesoru SASS, pak musí příslušné soubory dodatečně stáhnout z oficiální webové stránky.

Druhou možností je použití CDN (*Content Delivery Network*) a to tak, že do HTML dokumentu webové stránky jsou importovány skripty a kaskádové styly, které jsou uloženy na serverech CDN.

Třetí možností je instalace pomocí nástrojů pro správu balíčků. V současnosti je nejužívanějším nástrojem NPM (Node Package Manager). Instalace Bootstrapu pomocí tohoto nástroje je velice snadná, je třeba se přesunout do kořenového adresáře webové aplikace a spustit následující příkaz:

```
npm install bootstrap@3
```

Nástroj NPM poté automaticky vytvoří adresář node_modules, do kterého automaticky nainstaluje kompilované a minifikované kaskádové styly, JavaScriptové skripty a fonty. Poté stačí už jen importovat dané soubory do HTML dokumentu s relativními cestami k daným

souborům.

Ve všech třech případech se předpokládá, že uživatel bude mít naimportovanou knihovnu jQuery, kterou Bootstrap interně využívá ve svých skriptech.

6.4.1.2 Integrace do aplikačního rámce Angular

Integraci do aplikačního rámce Angular lze provést pomocí instalace modulu UI Bootstrap. Instalovat tento modul lze stažením z oficiální webové stránky [46] nebo pomocí nástroje NPM. V prvním případě je třeba nakopírovat modul do adresářové struktury aplikace a naimportovat soubory s kaskádovými styly, JavaScriptovými skripty a fonty do webové aplikace. V případě nástroje NPM je třeba se v příkazové řádce operačního systému přesunout do kořenového adresáře webové aplikace a spustit následující příkaz:

```
npm install angular-ui-bootstrap
```

Tento příkaz vytvoří nový adresář v adresáři node_modules a stáhne daný modul do webové aplikace, poté je třeba kaskádové styly, skripty a fonty moduly naimportovat do HTML dokumentu.

Posledním krokem integrace je přidání závislosti do angularové aplikace, to se provádí obecně následujícím způsobem:

```
angular.module('myModule', ['ui.bootstrap']);
```

6.4.2 Konfigurace Grunt

První velkou výzvou implementace byla správná konfigurace vývojového prostředí pomocí nástroje Grunt. V aplikaci tedy bylo třeba vytvořit soubor Gruntfile.js, do kterého přišla veškerá konfigurace. Pokud uživatel užívá nástroj NPM pro instalaci knihoven a modulů, pak je integrace úkolů, které Grunt spouští, velice snadná a provádí se pomocí kódu níže:

```
grunt.loadNpmTasks('grunt-angular-templates');
```

Uvedený kód načte pomocí služby NPM úlohu pro tvorbu šablon v jazyce JavaScript ze šablon v jazyce HTML. Toho se užívá právě v *single page* aplikacích. Šablony v jazyce JavaScript jsou součástí sloučeného JavaScriptového kódu, který vytváří nástroj Browserify.

Další úlohou, která byla implementována, je již zmíněný nástroj Browserify, který projde veškeré skripty v jazyce JavaScript a sloučí je do jednoho velkého souboru. Zároveň do tohoto souboru importuje i zdrojové kódy použitých knihoven a aplikačních rámců. Pro práci

s LESS soubory bylo třeba použít LESS kompilátor, který veškeré použité LESS soubory převede do jednoho souboru s kaskádovými styly. Grunt nabízí pro toto řešení úlohu, která se postará o kompilaci souborů a uložení obsahu do nového CSS souboru. Do cílového adresáře výsledné aplikace díky úkolům Browserify a Less dostaneme skripty a kaskádové styly. To však nestačí a je třeba ještě zkopírovat základní soubor index.html a fonty užité v dané aplikaci. To se provádí pomocí úkolu *copy*, který jednoduše překopíruje zdroje z jednoho místa na druhé. Posledními používanými úkoly jsou *connect* a *watch*, které jsou spolu spojeny. *Connect* vytváří webový server na portu 8000 a umožňuje v reálném čase opětovně načíst aktuální obsah zdrojových kódů. K tomu ale potřebuje zjistit, kdy došlo ke změnám ve zdrojových kódech. K tomuto účelu slouží úloha *watch*, která hlídá veškeré změny ve zdrojových kódech. Při jejich změně může spouštět jiné úlohy, například pro kompilaci. Pomocí této kombinace a zřetězení správných úloh je možné vytvořit vývojové prostředí, které automaticky zareaguje na změnu zdrojové kódu a automaticky znovu načte v prohlížeči aktualizovaný obsah. To je velice výhodné pro velké *front-end* aplikace, protože to urychluje vývoj.

6.4.3 Routing a navigace

Při tvorbě *single page* aplikací je poměrně problematickou částí navigace a správné přesměrování obsahu, protože *single page* aplikace se zobrazují pouze do jednoho HTML dokumentu a obsah se dynamicky mění. Angular řeší přesměrování pomocí modulu *angular-route*, avšak tento modul je velice jednoduchý a jeho funkčnost nedostačuje pro přesměrování ve větších aplikacích. Proto vývojářská komunita Angularu vytvořila modul *ui-route*, který je rozšířením knihovny *angular-route*. Základem navigace je direktiva *ui-view*, která se váže na kontejnerový element a umožňuje do svého obsahu naimportovat dynamický obsah, který je v určitém stavu. Pro každou šablonu, která se zobrazí v kontejneru s *ui-view,* je nutné definovat její stav. Ten se definuje pomocí služby $stateProvider a funkce *state().* Níže je uvedená ukázka nastavení stavu pro sekci se správou účtů:

```
$stateProvider.state('accounts',{
     parent:'main',
     url:'/accounts',
     templateUrl:'src/app/accounts/accounts.html',
     controller: 'AccountCtrl',
     params:{
         parentHome:false
     }
 })
```

Parametry funkce *state()* jsou název stavu a konfigurační objekt, který obsahuje informaci o rodičovských stavech, URL adrese, cestě k dané šabloně, název příslušného kontroleru a poté je možné přidat i dodatečné parametry, které se předávají při přechodu mezi stavy.

Přechod mezi stavy se provádí pomocí direktivy *ui*-sref, které se předá parametr název stavu. Tato direktiva se zpravidla aplikuje na element hypertextového odkazu. Tímto způsobem je možné definovat hierarchickou strukturu stavů a jejich přepínání, která tak definuje základní navigaci aplikace.

6.4.4 Komunikace klient – server

Dalším zajímavým problémem řešeným v aplikaci je komunikace klienta se serverem. Ze strany klienta se zprostředkovává pomocí angularových služeb $http a $resource. Zprostředkovávají tak komunikaci přes http protokol s REST rozhraním serverové části aplikace. Identifikace REST koncového bodu je definována pomocí užité metody – GET, POST, PUT nebo DELETE a příslušné URL adresy. Definice na straně klienta může vypadat následujícím způsobem:

```
$resource('iBanking/resource/rest/accounts/accountOverview', {},
{
        getAccounts:{
             method:'GET',
             url:'iBanking/resource/rest/accounts/
             accountOverview/:pageNum/:itemsPerPage',
             isArray:false
        }
    });
```

V uvedeném kódu je definována funkce *getAccounts()* užívající metodu GET, která má definovanou URL adresu s dvěma parametry *pageNum* a *itemsPerPage* a očekává jako odpověď JSON objekt.

Pro potřeby aplikace bylo nutné nějakým způsobem simulovat komunikaci se serverem. Pro tyto účely nabízí Angular službu $httpBackend, která se užívá pro integrační a jednotkové testování a pro tzv. *backend-less* vývoj, což je vývoj front-end aplikace bez přítomnosti serverové části aplikace. V praxi se tento způsob velmi často používá. Vývojové týmy se musí předem dohodnout na podobě REST rozhraní, které musejí mít oba týmy stejné a poté mohou vyvíjet nezávisle na sobě. Pomocí služby $httpBackend se zaregistrují metody s příslušnou URL adresou a na nich služby poslouchá, zda nedochází ke komunikaci. Pokud dojde ke komunikaci, spustí se kód, který je definován při registraci. Obvykle se vrátí stav odpovědi na požadavek a případně příslušná data. Níže je ukázka služby $httpBackend, která zachycuje požadavek na získání účtů:

```
$httpBackend.whenGET('iBanking/resource/rest/accounts/
accountOverview/1/5').respond(
       function(method, url){
            return[200,
            AccountBackendModel.getAccountOverviewPage1()];
```

```
            });


    $httpBackend.whenGET('iBanking/resource/rest/accounts/
    accountOverview/2/5').respond(
           function(method, url){
                return[200,
                AccountBackendModel.getAccountOverviewPage2()];
            });
```

Uvedený kód tedy poslouchá na daných URL a očekává požadavek metodou GET. Pokud jej obdrží, pak vrátí stav 200, což znamená úspěšný požadavek, a vrátí příslušná testovací data.

6.4.5 Grafy a responzivní komponenty

Grafy jsou v implementaci řešené pomocí knihovny chart.js, která podporuje velké množství nejrůznějších typů grafů, například sloupcové, čárové, kruhové atd. Pro integraci do angularové aplikace je nutné nainstalovat příslušný integrační modul *angular-chart.js*. Řešení skrze Angular umožňuje grafu snadno nastavit vstupní parametry a to: popisky osy x, popisky osy y, barevné schéma, typ grafu, a samotná data. Samotný graf se přidává pomocí direktivy na element <canvas>, který je součástí specifikace HTML 5. Tato knihovna pracuje s číselnými hodnotami, které je nutné ukládat do dvourozměrného pole. V aplikaci jsou grafy využity zejména pro přehled základních informací o vývoji v daném časovém období. Níže je uveden příklad sloupcového grafu v responzivní mřížce a s lokalizovaným popiskem:

```
<div class="col-md-6">
     <h4><span translate="accounts.incomeStructure"></span></h4>
     <canvas id="bar2" class="chart chart-bar"
      chart-data="generalOverviewCtrl.incomeStructureChart.data"
      chart-labels="generalOverviewCtrl.incomeStructureChart.labels"
      chart-legend="true"
      chart-series="generalOverviewCtrl.incomeStructureChart.series"
      chart-colors="generalOverviewCtrl.incomeStructureChart.colors"
     >
     </canvas>
</div>
```

Implementace také obsahuje velké množství responzivních komponent, které jsou umístěny do responzivní mřížky. Komponenty jsou používány podle dokumentace Bootstrapu či UI Bootstrapu. Avšak díky Angularu je možné si tyto komponenty funkčně přizpůsobit. Pro ukázku byly vybrány implementačně zajímavé komponenty a to panely, pole pro výběr prvku, pole pro výběr data.

Panely jsou definovány pomocí direktivy <accordion> a jsou blíže popsány v sekci 4.4.4. Níže je uvedená ukázka panelů, u kterých je možné skrývat jeho obsah, a jehož obsah podporuje formátování textu pomocí HTML tagů. Direktiva *ng-repeat* provede násobné vytvoření elementu na stránce podle počtu prvku v poli *newsCtrl.news.data*.

```
<accordion close-others="true"
 ng-repeat="new in newsCtrl.news.data">
        <accordion-group class="panel-default"
             is-open="new.isOpen">
            <accordion-heading>
                <h4>
                    <span>{{new.header}}</span>
                </h4>
            </accordion-heading>
            <div ng-bind-html="new.body"></div>
        </accordion-group>

    </accordion>
```

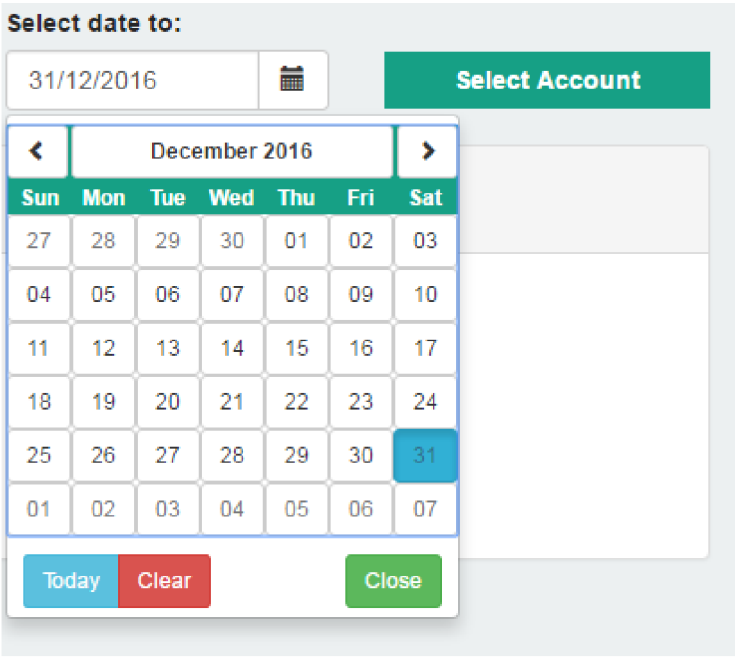
Pole pro výběr prvků v angularové aplikaci není obvykle řešeno použitím pouhého elementu <select>. Ve větších aplikacích je třeba při výběru prvku specifikovat zobrazenou hodnotu z objektu a na jejím základě při výběru nastavit jinou hodnotu nebo je třeba zvolit JavaScriptový objekt. Typicky při práci s číselníky, kde jsou pro každou hodnotu k dispozici dvě hodnoty – identifikátor a popisek. Práce s elementem <select> neumožňuje takovou možnost přizpůsobení komponenty a proto vývojářská komunita doporučuje užít komponentu ui-select. Tu lze do aplikace importovat jako angularový modul a její instalace je snadná – přes nástroj NPM. Níže je uveden příklad implementovaného výběru:

```
<ui-select
 id="generalOverviewAccountSelection"
 name="selectedAccount"
 ng-model="generalOverviewCtrl.selectedAccount"
 theme="bootstrap"
 validation="required">
 <ui-select-match>{{$select.selected.name}}</ui-select-match>
 <ui-select-choices
       repeat="account in generalOverviewCtrl.accounts.data">
  <div
  ng-bind-html="account.name | highlight:$select.search"></div>
  </ui-select-choices>
 </ui-select>
```

Uvedený kód nabízí výčet všech účtů v poli *generalOverviewCtrl.accounts.data* identifikovaných svým jménem. Při výběru je do modelu *generalOverviewCtrl.selectedAccount* vybrán celý JavaScriptový objekt a ne pouze textová hodnota. Atribut *theme* v příkladu zajiš-

ťuje to, aby byla komponenta stylována ve stylu Bootstrapu a direktiva *validation* zajišťuje validaci dané komponenty ve formuláři.

Poslední zde uvedenou zajímavou komponentou z pohledu implementace je pole pro výběr data. Tuto komponentu poskytuje UI Bootstrap, ale součástí klasického Twitter Bootstrap není. Komponenta se skládá z pole a přidruženého tlačítka s ikonou kalendáře, po kliknutí na ikonu se otevře modální okno s výběrem data. Na obrázku níže je ukázka této komponenty v aplikaci v momentě po kliknutí na ikonu kalendáře.



Obrázek 38: Ukázka komponenty pro výběr data.

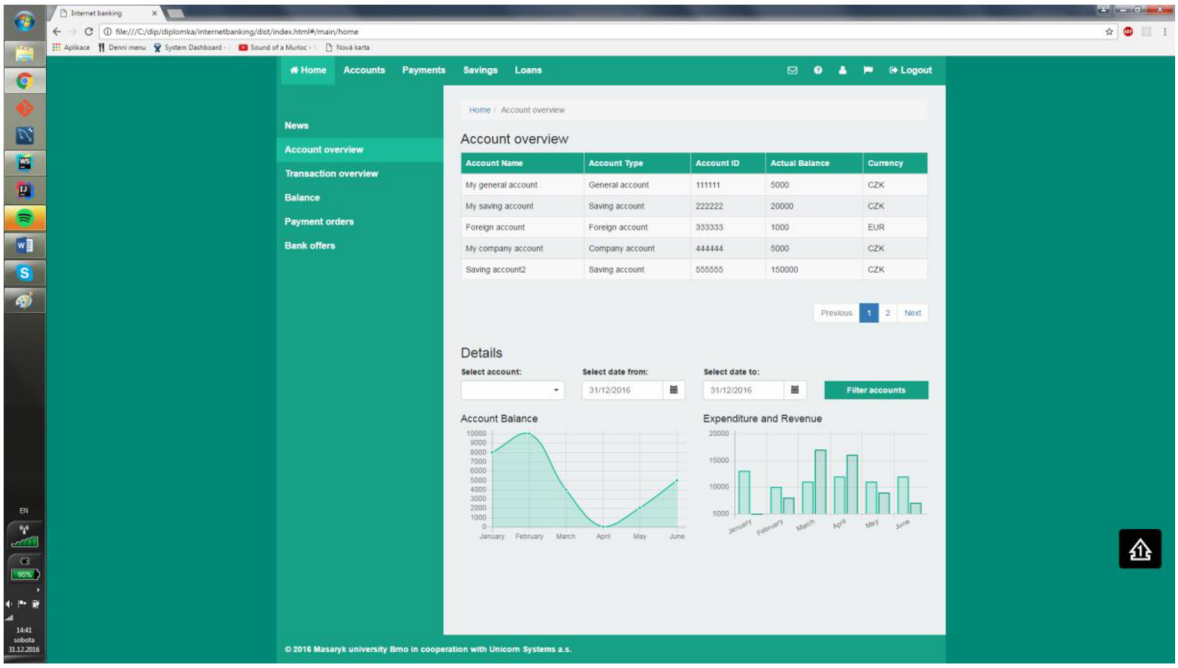
Níže je uvedena ukázka zdrojového kódu této komponenty:

```
<p class="input-group">
 <input type="text" name="dateFrom"
 class="form-control"
 datepicker-popup="{{transactionHistoryCtrl.format}}"
 ng-model="transactionHistoryCtrl.accountFilter.dateFrom"
 is-open="transactionHistoryCtrl.dateFromDatepickerOpened"
 datepicker-options="transactionHistoryCtrl.dateOptions"
 ng-required="true"
 close-text="Close"
 validation="required">
 <span class="input-group-btn">
       <button type="button" class="btn btn-default"
       ng-click="transactionHistoryCtrl.openDatePicker($event,
 'dateFrom')">
       <i class="glyphicon glyphicon-calendar"></i>
       </button>
  </span>
</p>
```

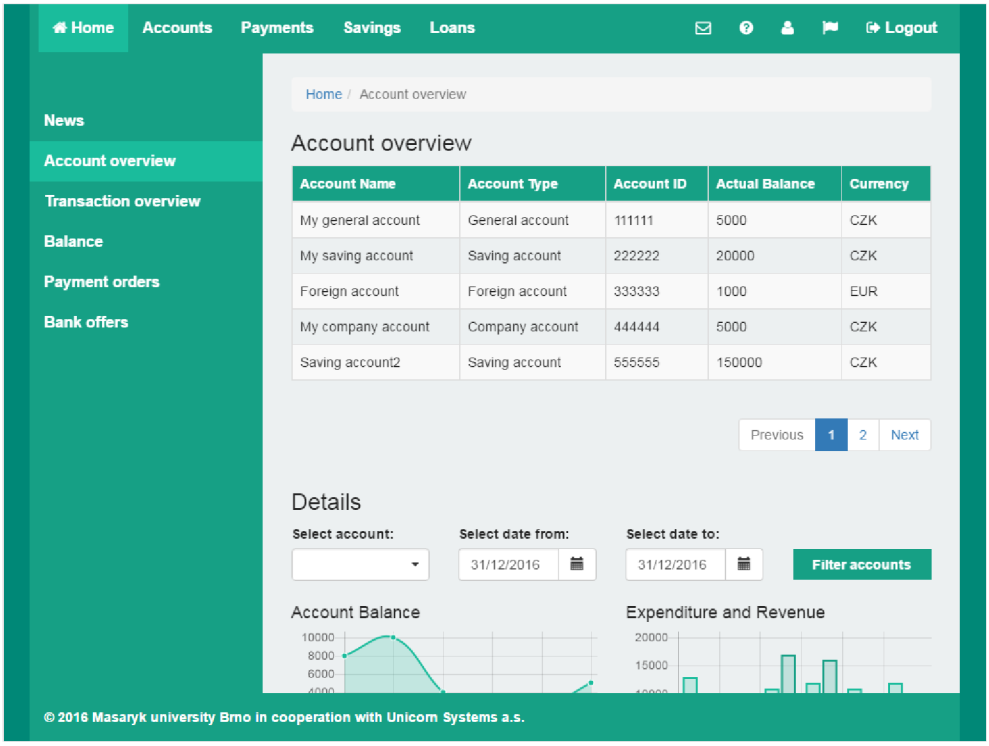
Pro tuto komponentu je nutné implementovat funkcionalitu pro otevírání modálního okna, dále je třeba nastavit formát data a nastavení komponenty.

6.4.6 Responzivita aplikace

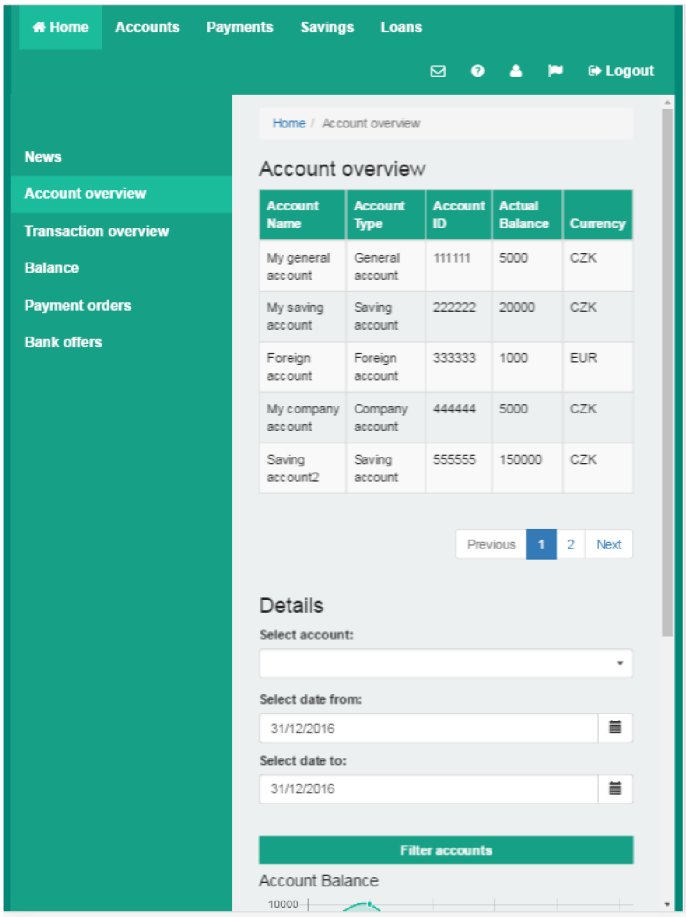
Implementace responzivní mřížky a komponent proběhla stejným způsobem, jako je popsáno v kapitole 4. Implementace probíhala přístupem *mobile-first*, kdy byla prvně definována responzivní mřížka pro malá zařízení a poté bylo upravováno rozložení mřížky pro zařízení větší. Následující obrázky ukazují zobrazení aplikace na různých obrazovkách.



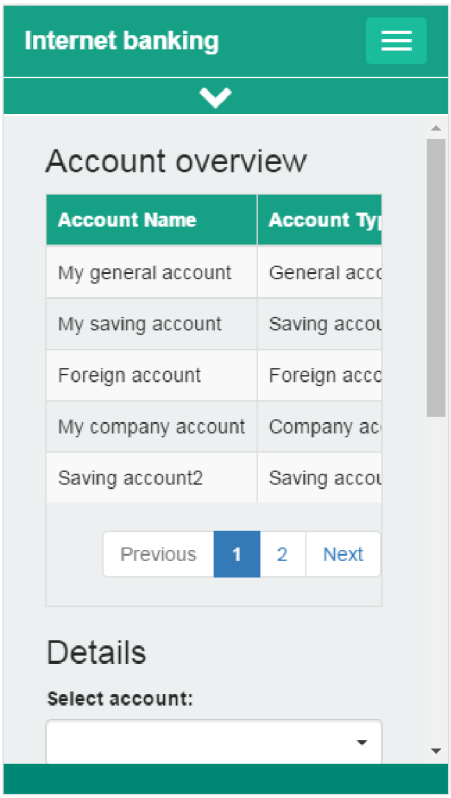
Obrázek 39: Implementovaná aplikace zobrazená na obrazovce s rozlišením 1920*1200.



Obrázek 40: Implementovaná aplikace se simulovaným zobrazením na tabletu s velikostí obrazovky 1024*768.



Obrázek 41: Implementovaná aplikace se simulovaným zobrazením na tabletu s velikostí obrazovky 768*1024.



Obrázek 42: Implementovaná aplikace se simulovaným zobrazením na mobilním telefonu s velikostí obrazovky 320*568.

Z obrázků jsou patrné změny responzivního rozložení pro různá zařízení a pro různě velké obrazovky. Konkrétně při zobrazování pro tablet při přetočení tabletu a tedy změně šířky obrazovky dojde k přeskládání prvků formuláře. Při zobrazení na mobilním zařízení dojde ke změně navigace.

6.4.7 Lokalizace

Poslední poměrně zajímavou částí implementace je lokalizace webové aplikace. Lokalizace se implementuje do angularové aplikace pomocí modulu *angular-translate*. Pro správnou lokalizaci je třeba veškeré popisky zapsat do JSON souboru, typicky s názvem en.json, kde je udržována anglická lokalizace. Pro překlad do jiného jazyka je třeba vytvořit další soubor, který bude obsahovat stejné klíče objektů jako původní soubor, ale již bude obsahovat lokalizované popisky a bude zaregistrován pomocí služby $translate, kterou nabízí *angular-translate.* Pro přepnutí lokalizace služba $translate nabízí funkci *use()*, kde se jako parametr předá zaregistrovaný název souboru. V HTML dokumentu se pro překlad používá direktiva nebo filtr *translate*, které se jako parametr předá příslušný klíč k danému popisku.

Příklad popisku v HTML dokumentu:

```
<h3><span translate="accounts.generalOverview"></span></h3>
```

Příklad záznamu v souboru en.json:

```
{"accounts.generalOverview": "General Overview"}
```

6.5 Testování

Po implementaci bylo třeba aplikaci důkladně otestovat a případné chyby opravit. Bylo nutné ověřit funkcionalitu implementovaných případů užití, dále provolání simulovaného REST rozhraní, ověřit responzivitu uživatelského rozhraní na různých zařízeních a v různých webových prohlížečích.

Testování aplikace bylo provedeno na stroji Lenovo Thinkpad T540P (s procesorem Intel Core i7-4710MQ, operační pamětí 16 GB DDR3, dedikovanou grafickou kartou NVidia GeForce GT 730M a SSD diskem 256 GB), operačním systému Windows 7 s nainstalovanými webovými prohlížeči Internet Explorer 11, Google Chrome 55, Mozilla Firefox 49, Opera 42 a nástroji Google Devtools.

V prvním kroku bylo třeba vizuálně ověřit zobrazení aplikace na prohlížečích, které jsou uvedené výše, aby bylo možné zajistit integritu responzivní mřížky a konzistenci uživatelského rozhraní. Poté byla otestována responzivita pomocí nástroje Google Devtools, což je nástroj pro webové vývojáře integrovaný do prohlížeče Google Chrome podporující vývoj webový aplikací, který mimo jiné umožňuje simulovat různá zařízení pro ověření responzivity webové aplikace. Opět bylo vizuálně ověřeno, zda je aplikace při zobrazení na menších obrazovkách konzistentní a zda se responzivní rozložení stránky chová tak, jak je očekáváno.

Pro správné zobrazení plně implementovaných responzivních komponent bylo nutné ověřit provolání funkcionality na simulované REST rozhraní. To bylo provedeno pomocí Google Devtools, který pro tyto účely nabízí funkcionalitu pro ladění aplikace. Při testování se do aplikace nastaví body, ve kterých se aplikace zastaví a jsou k dispozici dostupné proměnné, podobně jako při ladění programů ve vývojových prostředích (InteliJIdea, NetBeans, Visual Studio). Tímto způsobem bylo odladěno správné provolání příslušných funkcí, samotná funkcionalita responzivních komponent byla ověřena vizuálně podle toho, zda se zobrazovala správná data a komponenta byla správně renderovaná v prohlížeči.

7 Diskuze

Nevýhodou Angularové *single-page* aplikace je jeho špatný výkon pro rozsáhle webové aplikace, jakmile dojde k nedodržení některých z konvencí, které tomuto problému předcházejí. Např. ruční používání funkce $watch na modelu $scope, nadužívání angularových událostí nebo načítání velkého množství dat do prohlížeče během interakce. Proto by bylo velice zajímavé užití jiného aplikačního rámce jakým je Ember nebo React ve spojení s některým z aplikačních rámců pro responzivní uživatelské rozhraní. Například React s Bootstrapem má velice zajimavý zápis zdrojového kódu, kde užívá komponenty a HTML přímo v JavaScriptových souborech a to díky jazyku JSX [64], který je rozšířením jazyka JavaScript. Nicméně i přes nevýhody Angularu, je Angular současným nejkomplexnějším řešením, které pokrývá velké množství problémů, které se řeší během vývoje webové aplikace. Zajímavým řešením bude Angular 2, který má být v roce 2017 oficiálně vydán. Nyní je k dispozici beta verze, která má již nyní skvělé výsledky výkonu v porovnání s první verzí a stále si udržuje svoji komplexnost.

Výběr užívaného responzivního aplikačního rámce pro tuto práci nebyl také příliš jednoduchý, protože současné nástroje si jsou v mnohém podobné. Z mého pohledu a pohledu vývojářů v *Unicorn Systems* má obrovskou přidanou hodnotu široká vývojářská komunita, protože dojde-li k problémům během vývoje s komponentami nebo zobrazením v prohlížečích, pak díky široké je snažší nalézt správné řešení. I přes to si myslím, že by bylo velice zajímavé vyzkoušet na takové aplikaci například Semantic UI, který by mohl být díky své komplexnosti přínosem. Problémem je však špatná integrace do angularové aplikace. Proto byl zvolen spíše Bootstrap jako nástroj pro implementaci responzivního webu.

Flat design je podle mého názoru stále ještě novým designovým trendem, který se tak často ve velkých podnikových aplikacích nepoužívá. V současnosti je nejpoužívanější trend minimalistického designu, zejména pro svoji jednoduchost a přehlednost zobrazovaných informací. Výběr *flat* designu byl tak trochu experimentem, který by bylo možné ještě dále rozvíjet a upravovat. Responzivní design je dostatečně obecným přístupem a z popisu metodik je patrné, že jej lze aplikovat za pomocí popisovaných technik, uvedených v kapitolách 3 a 4, na jakýkoliv designový trend. Chce to pouze dostatečnou kreativitu na straně webdesignéra a webového vývojáře.

8 Závěr

Tato diplomová práce se zabývala tvorbou responzivního uživatelského rozhraní v moderním webovém prostředí. Písemná část nejprve shrnula současné designové trendy. Poté byly detailněji popsány principy responzivního designu, dále byly popsány přístupy k tvorbě responzivních webových stránek, jejichž součástí bylo představení současných nástrojů pro tvorbu responzivního webu a jejich srovnání. Součástí práce byl také popis metodik pro návrh a tvorbu responzivního uživatelského rozhraní. Textová část práce dále obsahovala analýzu požadavků na webovou aplikaci, jež byla praktickou částí této práce, a jednoduchou analýzu případů užití webové aplikace pro internetové bankovnictví. V následující kapitole byla popsána samotná implementace webové aplikace, kde byly uvedeny použité nástroje a architektury a byly uvedeny implementačně zajímavé části výsledné aplikace. Závěrem textová část práce diskutuje výběr užitých technologií a možná budoucí vylepšení webové aplikace.

Výsledkem praktické části této práce byla responzivní webová *single page* aplikace, která splňovala veškeré požadavky na ni kladené, jež byly popsány v podkapitole 5.2. Vybrané případy užití byly schopné komunikace přes http protokol. Na dané požadavky na server odpovídala část angularové aplikace, která byla zodpovědná za simulaci REST koncových bodů, které se obvykle nacházejí na straně serveru. Na základě testovacích dat byla aplikace schopná zobrazit data pomocí responzivních komponent, ze kterých byla aplikace sestavena. Vybrané případy užití specifikované v podkapitole 5.3 byly plně inter-aktivní, ostatní byly statické.

V budoucnu by měl vytvořený nástroj fungovat jako předloha pro metodiku návrhu responzivního uživatelského rozhraní, z toho důvodu by bylo vhodné vylepšení aplikace o další ukázky responzivních komponent, různých přístupů k implementaci, např. techniky pro adaptivní obrázky. Dále by v budoucnu měly být principy, techniky a metodiky popsané v kapitolách 3 a 4 podrobněji specifikovány, přeloženy do anglického jazyka a měly by být k dispozici formou dokumentu analytikům a vývojářům ze společnosti *Unicorn Systems* pro tvorbu responzivních uživatelských rozhraní komerčních projektů.

9 Literatura

1. LUBBERS Peter, ALBERS Brian a Frank SALIM. HTML 5: Programujeme moderní webové aplikace. Vyd. 1. Brno: Computer Press. c2011. 304s. ISBN 978-80-251-3539.
2. FREEMAN Adam. Pro AngularJS. Vyd. 1. New York: Springer. c2014. 659 s. ISBN 978-1-4302-6449.
3. Techterms: Flash [online]. [cit. 2017-01-03]. Dostupné na World Wide Web: <http://techterms.com/definition/flash>
4. GASSTON Peter. CSS 3: Průvodce vývojáře moderního webdesignu. Vyd.1. Brno: Computer Press. c2016. 296s. ISBN-978-80-251-4641-5.
5. PHP: Official websites [online]. [cit. 2016-12-03]. Dostupné na World Wide Web: <https://secure.php.net/>
6. Experti komunity jQuery. jQuery: Kuchařka programátora. 1. Vyd. Brno: Computer Press. c2010. 436s. ISBN 978-80-251-3152-7.
7. Usability.gov: User Interface Design Basics [online]. [cit. 2017-01-03]. Dostupné na World Wide Web: < https://www.usability.gov/what-and-why/user-interface-design.html>
8. SHARKIE Craig a Andrew FISHER. Responzivní webdesign. Vyd. 1. Brno: Computer Press. c2015. 141 s. ISBN 978-80-251-4384-1.
9. The evolution of webdesign [online]. [cit. 2016-12-03]. Dostupné na World Wide Web: <https://blog.kissmetrics.com/evolution-of-web-design/?wide=1>
10. LAURIVICINIUS Tomas. The top graphic and web design trends of 2016. [online]. [cit. 2016-12-03]. Dostupné na World Wide Web: <http://1stwebdesigner.com/graphic-web-design-trends-2016/
11. FLAVIAN Carlos, GURREA Raquel a Carlos ORÚS. Web design: a key factor for the website success [online]. [cit. 2016-12-05]. Dostupné na World Wide Web: <http://www.emeraldinsight.com/doi/abs/10.1108/13287260910955129>
12. SUBIĆ Nataša, KRUNIĆ Tanja a Biljana GEMOVIĆ. Responsive web design – Are we ready for the new age? [online].[cit. 2016-12-05]. Dostupné na World Wide Web: <http://www.iiakm.org/ojakm/articles/2014/volume2_1/OJAKM_Volume2_1pp93-103.pdf>
13. CHAPMAN Cameron. Principles Of Minimalist Web Design, With Examples [online]. [cit. 2016-12-06]. Dostupné na World Wide Web: <https://www.smashingmagazine.com/2010/05/principles-of-minimalist-web-design-with-examples/>
14. UxPin: Web design: Book of trends 2015 [online]. [cit. 2016-12-06]. Dostupné na World Wide Web: <https://www.uxpin.com/studio/ebooks/>

15. Material design [online]. [cit. 2016-12-07]. Dostupné na World Wide Web: <https://material.google.com/>

16. MEYER Kate. The Roots of Minimalism in Web Design [online]. [cit. 2016-12-08]. Dostupné na World Wide Web: <https://www.nngroup.com/articles/roots-minimalism-web-design/>

17. W3Schools: Parallax scrolling [online]. [cit. 2016-12-09]. Dostupné na World Wide Web: <http://www.w3schools.com/howto/howto_css_parallax.asp>

18. MARCOTTE Ethan. Responsive design. Vyd. 1. New York: A book apart. c2011. 186s. ISBN 978-0-9844425-7-7.

19. NIELSEN Jacob. Scrolling and Attention [online]. [cit. 2016-12-10]. Dostupné na World Wide Web: < https://www.nngroup.com/articles/scrolling-and-attention/>

20. PETTIT Nickolas. Which Layout? Static, Liquid, Adaptive, or Responsive [online]. [cit. 2016-12-11]. Dostupné na World Wide Web: <http://blog.teamtreehouse.com/which-page-layout>

21. CAO Jerry. Responsive vs. Adaptive Design [online]. [cit. 2016-12-11]. Dostupné na World Wide Web: <https://www.uxpin.com/studio/blog/responsive-vs-adaptive-design-whats-best-choice-designers/>

22. PIEJKO PAWEL. Which types of devices have browsers? [online]. [cit. 2016-12-11]. Dostupné na World Wide Web: <https://deviceatlas.com/blog/which-devices-have-browsers>

23. CHAFFEY David. Mobile Marketing Statistics compilation [online]. [cit. 2016-12-12]. Dostupné na World Wide Web: < http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/>

24. Techterms : Desktop computer [online]. [cit. 2016-12-12]. Dostupné na World Wide Web: <http://techterms.com/definition/desktop_computer>

25. W3Schools: Browser displays [online]. [cit. 2016-12-12]. Dostupné na World Wide Web: <http://www.w3schools.com/browsers/browsers_display.asp>

26. PILTCH Avram. Laptop screen resolutions [online]. [cit. 2016-12-13]. Dostupné na World Wide Web: <http://www.laptopmag.com/articles/laptop-screen-resolution-ripoff>

27. Techterms: Smart phone [online]. [cit. 2016-12-13]. Dostupné na World Wide Web: <http://techterms.com/definition/smartphone>

28. Intel: What is tablet PC? [online]. [cit. 2017-01-03]. Dostupné na World Wide Web: < http://www.intel.com/content/www/us/en/tech-tips-and-tricks/a-guide-to-tablet-pcs.html>

29. Smashing magazine: Fixed vs. Fluid vs. Elastic Layout: What's The Right One For You? [online]. [cit. 2016-12-15]. Dostupné na World Wide Web:

<https://www.smashingmagazine.com/2009/06/fixed-vs-fluid-vs-elastic-layout-whats-the-right-one-for-you/>

30. WROBLEWSKI Lukas. Multi-Device Layout Patterns [online]. [cit. 2016-12-15]. Dostupné na World Wide Web: <http://www.lukew.com/ff/entry.asp?1514>
31. BUDIU Raluca. Basic Patterns for Mobile Navigation: A Primer [online]. [cit. 2016-12-15]. Dostupné na World Wide Web: <https://www.nngroup.com/articles/mobile-navigation-patterns/>
32. CAO Jerry. 4 Creative Navigation UI Patterns for Great UX [online]. [cit. 2016-12-15]. Dostupné na World Wide Web: <http://www.awwwards.com/4-creative-navigation-ui-patterns-for-great-ux.html>
33. SCHENCKER Marc. Essential navigation patterns in 2016 [online]. [cit. 2016-12-16]. Dostupné na World Wide Web: <http://www.webdesignerdepot.com/2016/01/essential-navigation-patterns-in-2016/>
34. BUCKLER Craig. Improving Responsive Web Design With RESS [online]. [cit. 2016-12-18]. Dostupné na World Wide Web: <https://www.sitepoint.com/improving-responsive-web-design-ress/>
35. WOLFERMANN Sven. RESS: RWD and Server Side Components [online]. [cit. 2016-12-18]. Dostupné na World Wide Web: <http://maddesigns.de/RESS/#50>
36. WROBLEWSKI Lukas. RESS: Responsive Design + Server Side Components [online]. [cit. 2016-12-18]. Dostupné na World Wide Web: <http://www.lukew.com/ff/entry.asp?1392>
37. OTTO Mark. Bootstrap in A List Apart No. 342 [online]. [cit. 2016-12-18]. Dostupné na World Wide Web: <http://markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342/>
38. Twitter Bootstrap : Documentation [online]. [cit. 2016-12-20]. Dostupné na World Wide Web: <http://getbootstrap.com/>
39. Zurb Foundation: responsive Framework [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < http://foundation.zurb.com/>
40. GAMACHE David. Skeleton [online]. [cit. 2016-12-21]. Dostupné na World Wide Web: <https://responsivedesign.is/resources/frameworks/skeleton>
41. Semantic UI: responsive Framework. [online]. [cit. 2016-12-23]. Dostupné na World Wide Web: <http://semantic-ui.com/>
42. Materialize: responsive front-end framework [online]. [cit. 2016-12-25]. Dostupné na World Wide Web: <http://materializecss.com/>
43. SMACSS: Scalable and Modular Architecture for CSS [online]. [cit. 2016-12-25]. Dostupné na World Wide Web: <https://smacss.com/>
44. Tutorials Point: Learn Materialize [online]. [cit. 2016-12-25]. Dostupné na World Wide Web: <http://www.tutorialspoint.com/materialize/>

45. AngularJS: Documentation [online]. [cit. 2016-12-28]. Dostupné na World Wide Web: <https://docs.angularjs.org/guide/introduction>

46. UI Bootstrap: Bootstrap components for AngularJS [online]. [cit. 2016-12-28]. Dostupné na World Wide Web: <https://angular-ui.github.io/bootstrap/>

47. GRIFFITH, Richard. Notes on GUIDE [online]. Brighton (GB) : University of Brighton. 1999. [cit. 2016-12-29]. Dostupné na World Wide Web: <http://www.it.bton.ac.uk/staff/rng/teaching/notes/GUIDE/>

48. JSON: Úvod do JSON [online]. [cit. 2016-12-30].]. Dostupné na World Wide Web: <http://www.json.org/json-cz.html>

49. LESS: CSS preprocessor [online]. [cit. 2016-12-30]. Dostupné na World Wide Web: <http://lesscss.org/>

50. Shape Bootstrap: Flat theme – free responsive multipurpose site template [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <https://shapebootstrap.net/item/1524965-flat-theme-free-responsive-multipurpose-site-template>

51. SCHENKER Marc. Design trends: Flat design 2.0 [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <http://www.webdesignerdepot.com/2016/02/design-trends-flat-design-2-0/>

52. W3Schools: Responsive web design- images [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < http://www.w3schools.com/css/css_rwd_images.asp>

53. WWWConsorcium: CSS specifications [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < https://www.w3.org/Style/CSS/specs>

54. Modernizr: Detector of browser features [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < https://modernizr.com/>

55. Material UI: React components for materiál design [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < http://www.material-ui.com/#/>

56. Pure.css: Responsive CSS modules [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < http://purecss.io/>

57. UI kit: Lightweight modular front-end Framework [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < https://getuikit.com/>

58. W3Schools: Twitter Bootstrap tutorial [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: < http://www.w3schools.com/bootstrap/>

59. NPM: Package manager [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <https://www.npmjs.com/>

60. Browserify: Build tool [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <http://browserify.org/>

61. Grunt: Task manager [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <http://gruntjs.com/>

62. Visco software: Three layer architecture [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <http://viscosoftware.com/product-info/landed-cost-software-for-importersvisco-12/>

63. Tutorials Point: AngularJS tutorial [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <https://www.tutorialspoint.com/angularjs/angularjs_quick_guide.htm>

64. React: Documentation [online]. [cit. 2017-01-02]. Dostupné na World Wide Web: <https://facebook.github.io/react/docs/hello-world.html>