Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Univerzitní informační panel

Bc. Radek Marčan

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Radek Marčan

Informatika

Název práce

Univerzitní informační panel

Název anglicky

University information dashboard

Cíle práce

Prostudujte možnosti využití existujících rozhraní „Moodle“, „is.czu.cz“, ale i rozhraní Gmail (kalendář, TODO). Na základě zjištěných skutečností navrhněte vhodný informační panel určený pro dotyková zařízení – tablety. Tento informační panel bude editovatelný z webového rozhraní přímo lektorem. Zařízení implementujte a nainstalujte na katedře Informačního Inženýrství.

Metodika

Metodika diplomové práce je založena na podrobné analýze existujících elektronických informačních panelů v současnosti používaných pro informování studentů o konzultačních hodinách, konzultačních rezervací lektora, informací o výsledcích zkoušek a jiné. Prostudujte možnosti využití existujících rozhraní „Moodle“, „is.czu.cz“, ale i rozhraní Gmail (kalendář, TODO). Na základě zjištěných skutečností navrhněte vhodný informační panel určený pro dotyková zařízení – tablety. Tento informační panel bude editovatelný z webového rozhraní přímo lektorem. Zařízení implementujte a nainstalujte na katedře Informačního Inženýrství. Implementaci kriticky zhodnoťte a navrhněte případná vylepšení komunikačních rozhraní studovaných systémů.

Doporučený rozsah práce

60

Klíčová slova

iOS, Hybridní aplikace, Informační panel, Moodle

Doporučené zdroje informací

Fairchild, Charlie. Mobile App Development: 5 Words Security Dangers
iOS Human Interface Guidelines
Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. Salesforce Developers.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 22. 2. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 22. 2. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 22. 03. 2016

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Univerzitní informační panel" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2016 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Josefu Pavlíčkovi, Ph.D. za připomínky a odborné vedení.

Univerzitní informační panel

University information dashboard

Souhrn

Tato práce je zaměřena na problematiku vývoje mobilních aplikací. Nejdříve provádí porovnání podílu jednotlivých mobilních platforem. Dále jsou prezentovány jednotlivé technologie spolu s možnými vývojovými prostředími. Pak jsou zde uvedeny trendy v mobilních aplikacích a jejich vývoji a možnost získání dat pro aplikaci. Praktická část se zaměřuje na volbu vývoje mobilní aplikace. Dále na návrh, implementaci a tvorbu aplikace včetně sestavení a import na zařízení.

Summary

This thesis is focused on issues with development of mobile applications. At first it shows comparison of market shares by mobile platforms. Next there are presented individually technologies together with their possible development environment. Then there are trends in mobile applications with their progressions and possible ways to get data for application. Practical part is focused on choice of way of development of mobile Application. After there is draft, implementation and creating application including build and import to device.

Klíčový slova: Mobilní aplikace, iOS, Android, Windows Phone, Intel XDK, hybridní aplikace, nativní aplikace, trendy v mobilních aplikacích, způsoby vývoje.

Keywords: Mobile Application, iOS, Android, Windows Phone, Intel XDK, hybrid application, native application, trends in mobile applications, ways of developmen

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

V 21. století dochází k čím dál většímu rozvoji mobilních zařízení a s nimi souvisejícími technologiemi. S jejich rozvojem dochází k výraznému poklesu ceny těchto zařízení. Mobilní zařízení již nejsou pouze notebooky, či mobilní telefony. V dnešní době je k dispozici více, jak 5 druhů mobilních zařízení a stávají se tak nepostradatelnou součástí každodenního života.

Potenciál těchto zařízení není ani zdaleka vyčerpán a v důsledku toho dochází k čím dál většímu rozšiřování těchto zařízení. Každé nové vydané zařízení přináší nové technologie, které se vejdou do stále menšího zařízení. Dochází také ke zkvalitňování displejů, výkonu zařízení a optimalizace operačních systémů.

V důsledku výše uvedených faktorů dochází k využívání mobilních zařízení i pro informativní účely. Lze vidět v obchodních center velké dotykové informační panely, ale lze se i setkat s menšími informačními panely na zdech. Tyto menší informační panely jsou vytváření především z tabletů. Informační panel (tablet) má veliký potenciál, jednoduchou údržbu a hlavně velmi snadnou aktualizaci.

Tato práce si klade za cíl zjistit dostupné mobilní platformy a jejich rozšířenost. Dále si klade za cíl zjistit způsoby vývoje mobilních aplikací a s nimi souvisejících technologií. Následuje zjištění trendů v mobilních aplikacích a jejich vývoji. Poté možnosti využití stávajících systémů na univerzitě v rámci informačního panelu (tabletu).

2. Cíl práce a metodika

2.2. Cíl

Prostudujte možnosti využití existujících rozhraní „Moodle“, „is.czu.cz“, ale i rozhraní Gmail (kalendář, TODO). Na základě zjištěných skutečností navrhněte vhodný informační panel určený pro dotyková zařízení – tablety. Tento informační panel bude editovatelný z webového rozhraní přímo lektorem. Zařízení implementujte a nainstalujte na katedře Informačního Inženýrství.

2.3. Metodika

Metodika diplomové práce je založena na podrobné analýze existujících elektronických informačních panelů v současnosti používaných pro informování studentů o konzultačních hodinách, konzultačních rezervací lektora, informací o výsledcích zkoušek a jiné. Prostudujte možnosti využití existujících rozhraní „Moodle“, „is.czu.cz“, ale i rozhraní Gmail (kalendář, TODO). Na základě zjištěných skutečností navrhněte vhodný informační panel určený pro dotyková zařízení – tablety. Tento informační panel bude editovatelný z webového rozhraní přímo lektorem. Zařízení implementujte a nainstalujte na katedře Informačního Inženýrství. Implementaci kriticky zhodnoťte a navrhněte případná vylepšení komunikačních rozhraní studovaných systémů.

3. Teoretické východisko

Tato kapitola se zaměřuje na problematiku aplikací pro mobilní zařízení a na dostupné platformy. Ukazuje druhy mobilních aplikací, trendy v mobilních aplikacích a jejich vývoji. Dále ukazuje možnosti využití stávajících systému s využitím v mobilních aplikaci.

3.2. Mobilní zařízení

Definice pojmu mobilní zařízení se do jisté míry liší podle výkladu autora. Podle Ing. Pavlíčka Ph.D. lze popsat mobilní zařízení následovně: *„přenosná zařízení nebo malý přenosný elektronický bezdrátový přístroj s vlastním napájením s různými aplikacemi. Často je vybaven dotykovým displejem a/nebo miniaturní klávesnici.„*. V dnešní době patří mezi nejvíce rozšířené mobilní zařízení chytrý telefon (smartphone) a tablet. Novinou mezi mobilními zařízeními jsou takzvané chytré hodinky, které z pohledu vývoje aplikací obsahují pouze pár modelů, pro které lze tvořit. V neposlední řadě jsou zde Google Glass, které přinášejí virtuální realitu do reálného světa. Přinášejí tedy, možnost pořídit fotografii toho, co v daný okamžik člověk vidí, navigaci implementovanou do zorného pole, zobrazení informací o památce při pohledu na ní a další aplikace známé z mobilních zařízení. V této práci se budu zabývat především tablety a mobilními telefony.

3.3. Mobilní aplikace

Mobilní aplikace má více definic lišící se podle výkladu. Vycházím z následující definice: „A mobile app is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers.“(56) Volně přeložena definice do češtiny zní následovně: „Mobilní aplikace je softwarová aplikace vytvořená speciálně pro malé, bezdrátové výpočetní zařízení, jako chytrý telefon a tablet spíše než desktop, nebo notebook.“. Mobilní aplikace rozšiřují základní vlastnosti zařízení a využívají specifické vlastnosti mobilních zařízení, jako například dotykové ovládání, otáčení obrazovky, ale i technologie jako GPS či fotoaparát.

Každá mobilní aplikace se snaží rozšířit možnosti daného zařízení s využitím nabízejících technologií daného zařízení. K největšímu „boomu“ došlo v rámci mobilních aplikacích s příchodem telefonu iPhone v roce 2007, kdy mobilní telefon iPhone přinesl na tehdejší dobu převratné technologie širší společnosti. Po příchodu iPhonu na trh začal konkurenční boj v rámci mobilních zařízení, mobilních aplikací a všichni se předháněli,

kdo bude mít co nejlepší obchod na dané platformě s co největší počet aplikací a co nejvýkonnější zařízení.

V dnešní době jsou nejvíce rozšířené tři platformy Android, iOS a Windows Phone. Každá platforma má svůj vlastní obchod, přes který se dají získávat aplikace.

Platform	Oficiální obchod	URL
Android	Google Play	https://play.google.com
iOS	App store	https://itunes.apple.com
Windows Phone	Windows Phone store	http://www.windowsphone.com/store

Tabulka 1 Obchody podle platforem (zdroj: vlastní)

3.4. Přehled trhu

V dnešní době je na českém trhu přibližně 41 výrobců mobilních zařízení (především tabletů)[1] [2]. V této kapitole se podívám na trh z pohledu tržního podílu platforem (operačních systémů) a výrobců zařízení.

První podnětem k rozvoji trhu s mobilními zařízeními a aplikacemi byl rok 2007, kdy byl představen první Apple iPhone. V té době to byla do jisté míry revoluce v mobilních zařízeních, jelikož využíval kapacitní dotykový display (konkurence využívala rezistivní). Ovšem největším „tahákem“ tohoto zařízení byl právě nový operační systém iOS, který předčil veškeré dostupné systémy v dané době.

V dnešní době se mezi platformy s největším tržním podílem řadí Android, iOS a Windows Phone. Samozřejmě existují i alternativní operační systémy, které mají nižší hardwarové nároky, jako například Firefox OS. Mobilní zařízení s alternativním operačním systémem jsou mířeny především na rozvojové trhy, jako trh v Indii a podobně.

Dnes je trh s mobilními zařízeními, především tablety a smartphony obrovský, jelikož spoustě lidí naprosto vyhovuje jedno ze zmiňovaných zařízení a nahrazují tak počítače v domácnosti (16). Dále jsou zde zařízení, které získávají na popularitě a to jsou tzv. chytré hodinky, které umožňují spárování s telefony a různé notifikace. Chytré hodinky by se daly rozdělit na dva druhy a to takové, které umožňují ovládat aplikace na hodinkách a ty, které mají pouze navíc oproti klasickým hodinkám notifikace.

[1] https://www.alza.cz/tablety/18852388.htm
[2] http://www.czc.cz/tablety/produkty

V současnosti má největší podíl na trhu operační systém Android následovaný iOS a na třetím místě je platforma Windows Phone. Z pohledu čísel byl poměr tržních podílů za první a druhý kvartál roku 2015 následující:

Kvartál	iOS	Android	Windows Phone
2Q	13.9%	82.80%	2.60%
1Q	18.30%	78.00%	2.70%

Tabulka 2 Podíl na trhu 2015 (zdroj: 52)

Jak jsem již zmiňoval tak operační systém Android je na prvním místě s 82.8% podílem na trhu, následovaný s velkým odstupem iOS s 13.9% a na konec Windows Phone s nikterak velikým tržním podílem 2.6%. Důvodů ovlivňujících podíly na trhu je spousta. Pozastavím se u každého operačního systému a pokusím se je shrnout.

Android

Jelikož se jedná o open source software tak jej využívá nespočet výrobců. Výrobci se odlišují především odladěností systému, který ovlivňuje například výdrž baterie, plynulost systému a podobně. Jako příklad lze uvést společnost Xiaomi, která staví na svém systému a nízkých cenách zařízení. Tento systém je vhodný především pro lidi, kteří nechtějí, nebo nemají finanční prostředky na zařízení s iOS a jsou rádi, když si mohou systém upravovat podle chuti.

Nejdůležitějšími faktory ovlivňující prodejnost zařízení se systémem Android je především fakt, že se jedná o nejrozšířenější alternativu k operačnímu systému iOS a na tuto platformu je obrovské množství aplikací.

Mezi největší prodejce patří společnosti Samsung, Huawei, Lenovo (zahrnuje také společnost Motorola) a Xiaomi. Vzhledem k cenové politice společností dosahují velikých prodejů což přispívá k velkému tržnímu podílu operačního systému Android.

Nevýhodou ovšem je, že je zde velká různorodost hardwarového vybavení telefonu a proto systém běží na každém modelu zařízení odlišně a v případě špatné optimalizace vytváří negativní dojem.

iOS

Zařízení obsahující operační systém iOS jsou pouze od společnosti Apple a jedná se o prémiové zařízení. Velký vliv na prodeje má fakt, že se jedná o módní zboží a lidé jsou ochotni kvůli takovému zařízení prodávat orgány.

Hlavní výhodou je, že je systém optimalizovaný na dané zařízení a nedochází k zasekávání a podobně. Ovšem to má za následek, že je systém uzavřený a uživateli je dovolenou pouze několik málo možností co s daným zařízením provádět.

Od smrti Steva Jobse dochází k odchýlení se od vize společnosti Apple. Jako příklad lze uvést, že při představení Apple iPhone bylo zmiňováno, že stylus je naprosto zbytečná a nepotřebná věc, kterou společnost Apple nikdy nebude využívat a v roce 2015 byl představen stylus, jako převratná novinka. Dále dochází ke snižování kvality operačního systému, který se krátce po uvedení na trh potýkal s problémy.

Je zde spousta faktorů, které působí neblaze na společnosti Apple a díky tomu dochází ke snižování tržního podílu. Pokud tedy společnost nepřijde s něčím převratným, tak bude nejspíše klesající trend pokračovat.

Windows Phone

Operační systém od společnosti Microsoft staví především na tom, že je zde synchronizace s desktopovými aplikacemi (office apod), ale především míří na podnikovou sféru. Výhodou oproti konkurenci je, že není potřeba výkonného hardwaru pro dosažení plynulosti systému a díky tomu dosahují nižších cen zařízení.

Velkou nevýhodou je nedostatek aplikací, které jsou pro většinové zákazníky klíčové, jako například Snapchat a podobně. Bohužel je tento systém především podnikovou sféru a proto tomu odpovídá podílu na trhu.

V třetím kvartálu roku 2015 dosahovaly podíly na trhu následujících velikostí (16) (počet jednotek uveden v tisících kusů a podíl na trhu v %):

Platforma	Počet jednotek (3Q2015)	Podíl na trhu (3Q2015)	Počet jednotek (3Q2014)	Podíl na trhu (3Q2014)
Android	298 797	84.7	254 354	83.3
iOS	46 062	13.1	38 187	12.5
Windows Phone	5 874	1.7	9 033	3.0

Tabulka 3 Podíly na trhu (zdroj: 16)

Jak jsem již zmiňoval dříve, tak Android dosahuje největšího podílu na trhu a to s velkým náskokem před konkurencí. V případě třetího kvartálu se jedná o porovnání daných kvartálů z daného roku a nikoliv za celkový prodej za všechny tři kvartály.

V případě platformy Android (16) došlo k nárůstu podílu na trhu o 1.3% oproti předchozímu roku a v počtech kusů je to nárůst o 44 443 000 kusů. V případě iOS (16)je

nárůst pouze o 0.6% s tím, že oproti předchozímu roku se prodalo o 7 875 000 kusů více. Windows Phone (16) si pohoršil o 1.3% což je převedeno na kusy 3 159 000 kusů méně než v předchozím roce.

Z pohledu jednotlivých trhů (52) je ovšem poměr závislý na ekonomice dané země a především životní úrovně obyvatelstva. Například ve Španělsku je vidět, jak nedávné ekonomické trable přispěly k prodejnosti levnějších mobilních zařízení pracujících na operačním systému Android. Oproti tomu v Japonsku, jako v jediné zemi je podíl zařízení s operačním systémem Android menší než podíl zařízení s iOS. Důležité je také brát v potaz velikost populace dané země, protože například Čínský trh ovlivňuje celkové prodeje mnohem více než například Italský trh (52). V případě Ruska jsou data pouze do října 2015 (včetně).

Země /Platforma	Austrálie	Čína	Francie	Itálie	Japonsko	Německo	Rusko	Španělsko	USA	Velká Británie
Android	53.5 %	71.4%	69.9%	76.9%	44.4%	72.6%	74.7%	86.3%	51.9%	51.9%
iOS	39.6 %	27.1%	20.5%	14.5%	54.1%	20.2%	12.6%	12.2%	39.1%	38.6%
Window Phone	6.3 %	1.2%	8.7%	8.1%	0.0%	6.4%	10.8%	1.5%	1.6%	9.2%

Tabulka 4 Podíly na trzích (zdroj: 52)

V případě podílu na trhu jednotlivých firem a jejich prodejů je to následovně:

Společnost	Počet jednotek (3Q2015)	Podíl na trhu (3Q2015)	Počet jednotek (3Q2014)	Podíl na trhu (3Q2014)
Samsung	83 586.7	23.7	72 929.4	23.9
Apple	46 062	13.1	38 186.6	12.5
Huawei	27 262.3	7.7	15 935	5.2
Lenovo*	17 439.2	4.9	21 314.1	7
Xiaomi	17 197.2	4.9	15 772.5	5.2
Ostatní	161 296.6	45.7	141 246.5	46.3

Tabulka 5 Prodeje podle značek (zdroj: 16)

*Zahrnuje prodeje mobilních zařízení od značek Lenovo a mobilní divizi Motorola

Již nějaký čas kraluje prodejnosti společnost Samsung (16), které sice pokles podíl na trhu o 0.2% oproti předchozímu roku, ale v počtu zařízení se jim povedlo zvýšit prodej o 10 657 300 kusů. Hned na druhém místě je společnost Apple (16), kde se jim povedlo zvýšit podíl na trhu o 0.6% a prodeje o 7 875 400 kusů. Další tři místa jsou obsazeny společnostmi, které se zaměřují na customizaci systému Androidu a nízkou cenu. V případě společnosti Huawei (16) došlo k růstu podílu o 2.5% a v počtu jednotek o

11 327 300 kusů. V případě společností Lenovo a Xiaomi (16) došlo k poklesu podílu na trhu, ale v případě Xiaomi došlo k růstu počtu jednotek o 1 424 700 kusů.

3.5. Platformy

Mobilní platforma je pojem, který označuje hardware a software řešení s prezentováním výstupů pochopitelných pro uživatele. V případě definic je to takto: *„The hardware/software environment for laptops, tablets, smartphones and other portable devices.* „(15) v případě české definice následovně: *„Mobilní platforma označuje kombinaci hardwaru (chytrého mobilního telefonu) a mobilního operačního systému.“* (47). V obou případech se jedná o shodné definice a to tedy, že mobilní platforma je kombinace hardware a software zařízení poskytující funkční celek.

3.5.1. iOS

iOS je operační systém od společnosti Apple vytvořený pouze pro zařízení od dané společnosti. Název iOS se používá od čtvrté verze tohoto systému. Předtím byl nazýván iPhone OS a byl pouze pro telefony Apple iPhone. Ihned po zveřejnění přejmenování systému bylo na společnost Apple podána žaloba společností CISCO, jelikož společnost CISCO využívá označení IOS pro software na svých routerech a přiměla tak společnost Apple si nechat patentovat název pro svá zařízení. Jedná se o druhý nejrozšířenější mobilní operační systém hned po systému Android.

Tento systém byl představen v roce 2007 společně s první verzí telefonu Apple iPhone. iOS je založen na odlehčené verzi operačního systému Mac OS X používaného na počítačích Apple. Jde tedy o UNIXový systém, který ovšem neobsahuje plnou funkcionalitu systému Mac OS X, ale rozšiřuje jej o dotykové ovládání. Společnost Apple poskytuje novou verzi přibližně jednou za rok s tím, že v průběhu roku vycházejí drobné vylepšení či případně opravy chyb systému. V dnešní době lze nalézt iOS na zařízeních iPhone, iPad, iPod touch, AppleTV.

Pro iOS se vyvíjí v programovacím jazyku Objectiv-C, který vychází z jazyku C a dalo by se říci, že se jedná o určitou nadstavbu. K vývoji se využívá vývojového programu Xcode. V dnešní době se využívá jazyku Swift 2. Jedná se o moderní programovací jazyk zaměřený především na bezpečnost, rychlost a expresivnost.

Výhody

- Jednoduché ovládání pro uživatele

- Plynulost systému
- Široká škála aplikací
- Bezpečnost

Nevýhody

- Uzavřenost systému
- Delší schvalovací proces App storu

Lze tedy říci, že zařízení s iOS je vhodné především pro lidi, kteří hledají jednoduchý systém, který hezky vypadá, funguje bez sekání a nepředstavuje velké bezpečnostní riziko.

Co se týče grafické stránky systému tak je značně odlišná od konkurence. Společnost Apple vydává příručku zvanou iOS Human Interface Guidelines, která je zdarma dostupná na iTunes. V této příručce jsou vysvětleny principy grafiky pro iOS a také rady, které urychlují schvalovací proces aplikace.

Aby člověk dostal svoji aplikaci do App storu tak musí splňovat kritéria stanovená společností Apple. Díky přísným kritériím předchází společnost duplikaci aplikací, výskytu zbytečný či nedoladěných aplikací a také zamezuje přístupu aplikace k tomu, co nepotřebuje pro svůj provoz. Například aplikace pro sdílení fotografií rozhodně nepotřebuje přístup k sms a podobným informacím. V obchodu App Store je přibližně 2 169 038 aplikací (13).

Některé z důvodů zamítnutí aplikace jsou následující (55)

- Padá při provozu
- Vykazuje chyby
- Není v souladu s tím, co vývojář uvedl
- Zahrnuje nedokumentované nebo skryté funkce v rozporu s popisem aplikace
- Používá neveřejné API
- Čte nebo zapisuje data mimo svůj určený kontejner
- Stahuje kód jakýmkoli způsobem nebo formou
- Instaluje nebo spouští jiný spustitelný kód
- Jde o beta, demo nebo zkušební verzi
- iPhone aplikace musí běžet na iPadu bez úprav
- je kopií aplikací v App Store již umístěných, zejména pokud takových aplikací existuje mnoho

- není velmi užitečná, například jde o prosté webové stránky přibalené v rámci aplikace
- je primárně zaměřená na marketing nebo reklamu
- jde o trik nebo obsahuje falešné funkce, které nejsou jasně označeny
- podporuje nadměrnou konzumaci alkoholu, nelegálních látek, nebo nabádá děti nebo mladistvé konzumovat alkohol či kouřit cigarety
- poskytuje nesprávné diagnostické nebo nepřesné údaje
- vývojáři „spameři“ s mnoha verzemi podobných aplikací budou odstraněni z programu Developer iOS
- obsahuje například jen jednu skladbu nebo film – takové by mělo být nahráno na iTunes Store
- obsahuje knihu – takové by mělo být nahráno do iBookstore
- svévolně omezuje uživatele, kteří mohou aplikaci používat například podle místa
- neřídí se pokyny pro ukládání dat pro iOS

Jak je vidět z některých požadavků výše, tak Apple dbá na přidanou hodnotu a kvalitu aplikací, které budou k dispozici což přispívá k bezpečnosti a uchování plynulosti systému.

3.5.2. Android

Operační systém Android je vyvíjen společností Google. Společnost Android Inc vznikal v roce 2003 a posléze byla v roce 2005 odkoupena společností Google Inc a udělal z ní svoji dceřinou společnost. 5. listopadu 2007 bylo vytvořeno uskupení Open Handset Alliance, které zahrnuje společnosti zabývající se výrobou mobilních telefonů, čipů nebo mobilních aplikací. Cílem tohoto uskupení bylo vyvinout otevřený standard pro mobilní zařízení. První telefon s operačním systémem android se objevil v roce 2008 ve Spojených státech amerických (v říjnu) a v České republice v lednu roku 2009.

Systém je založen na Linuxovém jádru a jedná se tedy o open-source systém. Díky tomu je tak moc rozšířený a dává možnost výrobcům jej upravovat podle chuti. V dnešní době nalezneme systém Android na spoustě mobilních zařízení od mobilních telefonů přes chytré hodinky až po automobily.

Pro Android se využívá programovacího jazyku JAVA s tím, že zde existuje velké množství vývojových prostředí a není potřeba vlastnit počítač s konkrétním operačním

systémem. Od verze 5.0 Lollipop začal systém Android využívat tzv material designu (26) s tím, že jsou zde oficiální stránky od společnosti Google, kde se dají najít rady, tipy triky a podklady či případně tutoriály (26), což by se dalo označit za obdobu guidelines od společnosti Apple.

Výhody

- vysoká upravitelnost systému
- obrovské množství aplikací
- velká škála zařízení
- možnost dostat do zařízení aplikace mimo Google play

Nevýhody

- minimální kontrola aplikací
- zranitelnost systému
- v případě špatné nadstavby výrobce zařízení dochází k zasekávání zařízení

V případě vývoje mobilních aplikací je lze distribuovat mimo Google play a vyhnout se tak kontrole ze strany společnosti Google, ale má to své rizika. Jako příklad lze uvést aplikaci, která sloužila k prohlížení pornografického materiálu, ale v případě, kdy uživatel spustil aplikace tak jej vyfotila, fotku nahrála na stránku s diváky pornografického materiálu a následně uzamkla telefon s tím, že jej odemkne po zaplacení určité částky (přibližně 500€)(5). Samozřejmě to neznamená, že veškeré aplikace mimo Google Play jsou škodlivé, ale vždy záleží na uživateli. Tento způsob se využívá především k získávání placených aplikací bez nutnosti platit.

Google Play z prvu neměl kontrolu aplikací a docházelo tak k výskytu malwarů a podobných škodlivých aplikací a proto zavedl ruční kontrolu aplikací. Dále je zde hodnocení aplikací podle standardů, které jsou založeny na regionálních normách, tedy například v Evropě se jedná o PEGI. Země, které nemají jednotný systém hodnocení se používá věková stupnice. V praxi to znamená, že autor aplikace musí vyplnit dotazník týkající se jeho aplikace a posléze aplikace dostane příslušné hodnocení. V případě nevyplnění dotazníku budou aplikace označené, jako nehodnocené a může dojít k blokování v některých zemích.

V dnešní době je tedy kontrola aplikací žádající o umístění na Google Play následující. Aplikace je nejdříve ohodnocena dotazníkem od autora, která jí přiřadí určité hodnocení, posléze je aplikace do jisté míry přezkoumány ručně (tedy lidmi) a posléze jsou

na ní uplatněny algoritmy a různé automatizované postupy pro odhalení malware, nebo obsahu chráněný autorskými zákony. V Google Play je přibližně 1 967 222 aplikací (49).

Podle mého názoru je to nepříliš vhodná kombinace s tím, že samotný operační systém Android obsahuje bezpečnostní rizika a neúplná kontrola aplikací (přístupy apod) tyto rizika zvyšují. Lze tedy říci, že operační systém Android je vhodný především pro lidi, kteří si rádi uzpůsobují svůj mobilní telefon či tablet a mají rádi otevřený systém.

3.5.3. Windows Phone

Windows Phone je operační systém pro mobilní zařízení od společnosti Microsoft. Označení Windows Phone se používá od verze 7, ale více se rozšířilo ve verzi 8 a vyšší. První vydání bylo v roce 2010 s tím, že Windows Phone 7 a 8 obsahují různé platformy, které nejsou vzájemně kompatibilní. S příchodem verze 8 přišlo nové uživatelské rozhraní metro, jako je tomu u Windows 8 a 8.1 na počítačích.

Windows Phone 8 a novější již stojí na jádru Windows NT, které se využívá i u počítačové verze systému Windows a přináší tak propojení trhu mezi mobilními zařízeními a počítačovou verzí. Je tedy možné vytvořit aplikaci, která bude bez změn fungovat jak na mobilním zařízení, tak i na počítači.

Verze 10 již není označována jako Windows Phone, ale pouze jen Windows, jelikož se jedná o univerzální operační systém, který je zatím v testovací verzi na mobilních zařízeních a je tedy dostupný pouze přes Windows Insider.

Aplikace lze vytvářet v jazycích C#, Visual Basic.NET (.NET), C++ a HTML5/JavaScript. K vývoji je dostupné Visual Studio zdarma (Express verze), která obsahuje vše potřebné včetně emulátoru mobilního zařízení a není tedy nutné disponovat fyzickým kusem. Na rozdíl od vývoje pro Android je tedy nezbytné vlastnit počítač s operačním systémem Windows, což na druhou stranu není v dnešní době problém, jelikož se jedná o nejpoužívanější operační systém na počítačích. V dnešní době společnost Microsoft pracuje na způsobu, jak portovat aplikace z jiných operačních systémů na Windows Phone, aby nebylo tolik pracné pro vývojáře vytvářet aplikace na Windows Phone. V dnešní době se jedná především o portování ze systému iOS, jelikož to není tak náročné časově a technicky, jako v případě Androidu.

Výhody

- stabilní systém
- plynulý systém s kancelářskými balíky od Microsoft zdarma

- kvalitní navigace HERE+
- malé hardwarové nároky

Nevýhody

- Málo aplikací
- Malý podíl na trhu, tedy není moc velký zájem o tvorbu aplikací

V případě publikování aplikací je to podobné, jako v případě App store u iOS. Každá nahraná aplikace prochází procesem testování, který kontroluje například rychlost spuštění, obsah aplikace a podobně. V případě, že aplikace obsahuje nějaké VIP věci, nebo například věci, které se kupují za peníze tak je nezbytné poskytnout tyto věci testerům, aby bylo možné kompletně otestovat funkcionalitu aplikace.

Výhodou při vytváření aplikací pro Windows Phone je jednoduchý systém verzování aplikací ve Visual Studiu a následně v obchodu. Dále je zde uvedený hodnocení podle věkové přístupnosti, jelikož Windows Phone obsahují rodičovský zámek, který je automaticky aktivován (v případě obchodu) pokud je uživateli, který si vytvoří účet méně, jak 18 let.

Důvody k zamítnutí schválení aplikace jsou podobné, jako v případě App storu tedy například (60)

- Aplikace obsahuje nebo zobrazuje sexuální či pornografický obsah
- Obsahuje funkcionality, které nabádají či napomáhají k ilegálním aktivitám v reálném světe
- V případě potřeby vytvoření účtu je nezbytné poskytnout testovací účet

Windows Phone je bohužel umírající platforma, jelikož její prodeje jsou oproti konkurenci mizivé. Na druhou stranu je tato platforma určena především pro podnikovou sféru a také pro lidi, kteří mají rádi vše propojené bez zbytečných konverzí formátu a podobně. Velkou výhodou je, že na levném zařízení s poměrně nevýkonným hardware (oproti konkurenci) běží systém plynule a veškeré aplikace též.

3.6. Způsoby vývoje a prostředí

V dnešní době existuje několik druhů tvorby aplikací. Složitost tvorby je dána volbou způsoby tvorby a na dovednostech jedince. Například pro Windows Phone existuj Microsoft Project Sienna, kde je možné vytvořit aplikaci bez nutnosti programovat.

Před samotnou tvorbou aplikace je důležité se rozhodnout, na jakou platformu chceme cílit a jestli je cílem vytvořit univerzální aplikaci, kterou lze využívat na všech platformách. Toto rozhodnutí ovlivní náklady na danou aplikaci, její kvalitu, ale i celý proces vývoje.

Možností jsou následující:

3.6.1. Nativní mobilní aplikace

Nativní aplikace jsou prvotním způsobem tvorby aplikací pro mobilní zařízení. Aplikace je vytvořena pro jednu platformu s tím, že je tvořena v konkrétním programovacím jazyce a splňuje veškeré požadavky na uživatelské rozhraní a guidelines dané platformy. Jediným společným prvkem u všech platforem je ovládání. Hlavní výhodou nativní mobilní aplikace je rychlost, vysoký výkon a spolehlivost. Nevýhodou ovšem je nákladný vývoj na danou platformu a je nezbytné aplikaci vyvíjet od začátku včetně grafického rozhraní.

Nevýhodou vývoje nativní mobilní aplikace je fakt, že vývoj je pouze na jednu platformu a je značně finančně náročný oproti jiným řešením.

V dnešní době je možné vytvářet aplikace mimo nativní jazyk, jelikož existují vývojové programy, které posléze překompilují vytvořenou aplikace (např. v C#) do nativního jazyka. Nativní aplikace jsou tvořeny pro operační systém Android v jazyce Java, pro iOS v jazyce Swift a Objective-C a pro Windows Phone v jazyce C++.

3.6.2. Hybridní mobilní aplikace

Lze říci, že je to alternativní způsob tvorby aplikace pro lidi, kteří ovládají webové technologie (HTML, CSS, JavaScript) a chtějí vytvářet aplikace na mobilní zařízení. Aplikace jsou instalovány a spouštěny, jako nativní s tím rozdílem, že kód aplikace je psaný v jiném jazyce a aplikace jsou sestaveny do balíčků pro dané platformy. Výhodou je, že uživatelské prostředí je tvořenou pouze jednou a to pro všechny platformy stejně.

V roce 2011 řekl David Fetterman (Mobile Engeneering Manager ve společnosti Facebook) následující: *„All of our developers are good at HTML. Only a few of them are*

really good at Objective-C and Android. We are able to make our web developers the same as our clientside developers in some respects.“ (18).

Narážel tím na problém, se kterým se potýkaly společnosti, které chtěli vyvíjet mobilní aplikace, ale vývojáři se schopnostmi pro tvorbu nativních aplikací jsou mnohem dražší než kvalitní vývojáři se znalostí mobilních technologií.

Možnou alternativou pro vývoj mobilních aplikacích by tedy mohla být cesta webových technologií a uzavřít je do obálky, která umožňuje spouštět HTML a JavaScript v nativním prostředí na různých platformách. Nativních vlastností je tedy dosaženo zasazením webové aplikace od kontejneru. Tímto způsobem lze tedy získat aplikaci, u které je aplikační logika napsána pouze jednou s možností využití na více platformách. Standardní webová aplikace neumožňuje přístup ke specifickým technologiím, jako například gyroskop, geolokace a podobně, ale její zasazení do nativního kontejneru tento přístup umožňuje.

Hlavní výhodou hybridní aplikace je to, že není nezbytně nutné umět jazyky pro tvorbu nativních aplikací, jako například Javu, C#, Objective-C a podobně. I přes to lze vyvinout aplikace, kterou bude možné provozovat na všech platformách. Při vývoji hybridní aplikace se setkáme s několika technologiemi, které zde rozepíši.

HTML

Jedná se o značkovací jazyk, který je používán pro tvorbu webových stránek (prezentací). Jednotlivé stránky jsou propojené pomocí hypertextových odkazů a slouží k publikaci dokumentů na Internetu. Jazyk vychází z dříve vyvinutého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj jazyka HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. První verze jazyka byla v roce 1991 a poslední verze vyšla v roce 2014 (HTML5).

Koncepcí jazyka jsou značky, které mají vlastnosti definované pro danou verzi jazyka. Mezi značky se uzavírají části textu a tím se určuje význam obsaženého textu. Například u vytvoření odkazu na jiný HTML soubor je text uzavírán do značek <a> s atributem href odkazující na daný soubor (relativní, nebo úplná cesta). HTML soubor má

základní strukturu skládající se z hlavičky, těla a patičky, jako v případě dokumentů obecně. K upravení grafické stránky se využívá jazyka CSS.

CSS

CSS je jazyk, který slouží k popsání způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XTHML a XML. První verze jazyka CSS se objevila v roce 1996 a byl vytvořen, jako reakce na vyvíjející se jazyk HTML. Snahou bylo oddělit informace o obsahu od pokynů k formátování. V dnešní době je verze CSS3, která rozšiřuje jazyk o mnoho prvků, jako například animace.

CSS lze aplikovat přímo v dokumentu HTML, kde se píše mezi značky <style> </style>, nebo do externího souboru s příponou css. V případě externího souboru je nezbytné jej připojit pomocí kódu link s atributy rel (druh souboru), href (cesta k souboru). V případě, že se jedná o základní stavební jednotku jazyka HTML není nutné dávat před název formátu (body) značku, v případě třídy (class) je to . (tečka) a v případě jediného identifikátoru se před název dává # (hashtag). Třídu využiji v případě, že mám více elementů, kterým chce nastavit stejné vlastnosti formátování a identifikátor pouze pro jeden element jazyka HTML (například pro div, či obrázek). V případě, že chci rozšířit funkcionalitu webové prezentace mohu využít další technologií, jako například JavaScript.

JavaScript

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk. Vznikl v roce 1995 a jeho autorem je Brendan Eich. Syntaxe jazyku JavaScript patří do rodiny C/C++/Java a má tedy podobnou syntaxi, jako jazyk Java. Slovo Java v názvu je pouze z marketingových důvodu a původně se jazyk jmenoval ECMAScript.

Javascript slouží k přiřazení rozšířené funkcionality webové prezentace. Samozřejmě jsou zde další jazyky, které také rozšiřují funkcionalitu, jako například PHP, ale Javascript na rozdíl od PHP je na klientské části, tedy o výpočetní výkon se stará klient (uživatel) a nedochází k zahlcení serveru. Kód lze psát přímo v daném HTML dokument mezi značky <script></script>, kdy bez specifikování jazyka se bere, že je nastaven Javascript. Lze jej psát do externího souboru s příponou jss a posléze jej napojit do HTML dokumentu přes element <script> s přidáním atributu src (cesta k danému souboru). Je nutné si dát pozor na velká a malá písmena, jelikož Javascript je Case Sensitive. Samozřejmostí je fakt, že Javascript obsahuje spousty rozšiřujících knihoven, jako například jQuery, kterou využiji dále v této práci.

jQuery

Jedná se o Javascriptovou knihovnu, které má širokou podporu prohlížečů a klade důraz na interakci mezi Javascriptem a HTML. Tato knihovna byla vydána Johnem Resignem v roce 2006. Jedná se o svobodný a otevřený software pod licencí MIT. Filozofie byla podobná, jako v případě CSS a HTML a to tedy, že je cílem oddělit chování od struktury HTML. Ve výsledku to pro koncového uživatele (programátora) znamená to, že je ušetřen při psaní kódu. Cílem jQuery je rozšířit funkcionalitu Javascriptu a zároveň zjednodušit psaní kódu zjednodušením syntaxe. Knihovnu jQuery jsem si vybral právě proto, že zde využiji API na počasí, které odesílá data v souboru JSON.

JSON

Jedná se o objektově orientovaný způsob zápisu dat, který je nezávislý na platformě. JSON (JavaScript Object Notation) je určený pro přenos dat organizovaných v poli, nebo v objektech.

API

Jedná se o zkratku Application Programming Interface a označuje rozhraní pro programování aplikací. Jde o sbírku procedur, tříd a protokolů nějaké knihovny, které může programátor využívat. Vlastně API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu. V případě počasí jde o tzv. Web API což je webová služba, které je definována přes http a požaduje zprávy spolu s definicí struktury odpovědi obvykle v XML či JSON formátu. Web API je prakticky synonymem pro webovou službu, ale nedávný trend (tzv. web 2.0) se vzdaluje od Simple Object Access Protocol, tedy služby založené na více přímých REST stylů komunikace. Web API umožňují kombinaci různých služeb do nových aplikací známých, jako mashups.

Například u počasí je to tak, že na základě získaných geografických pozic (zeměpisná délka, šířka) odešlu dotaz příslušnému API spolu s API klíčem, který slouží ke kontrole přístupu k danému API. API přijme dotaz, provede získání dat a vytvoření příslušného souboru, který je posléze zaslán, jako odpověď.

3.6.3. Webové mobilní aplikace

Webové mobilní aplikace jsou v samé podstatě webové stránky zobrazované pomocí prohlížeče v mobilním zařízení. Taková to aplikace je závislá na připojení k internetu a nelze jej nainstalovat do zařízení. S uživatelským rozhraním je to stejné, jako u hybridní mobilní aplikace a to tedy, že je tvořeno pouze jednou pro všechny platformy.

Výhodou je, že pokaždé když uživatel přistupuje k aplikace tak pracuje s nejnovější verzí. V případě ostatních způsobů tvorby je nezbytné vytvořit novou verzi aplikace a nahrát jí na obchod a vyčkat, než si ji uživatel stáhne. Na druhou stranu velkou nevýhodou je, že webové mobilní aplikace nemohou přistupovat k většině hardwarového vybavení daného zařízení. Například nelze využít interního úložiště, číst kontakty, nebo plně využívat akcelerometru.

3.6.4. Vývojové prostředí

Pro vývoj nativní aplikace slouží Android Stuido (Android), Xcode pro iOS a Visual Studio pro Windows Phone. Dále tu existují multiplatformní programy, jako například Xamarin a Intel XDK.

Android Studio[3]

Jedná se o oficiální vývojové prostředí od společnosti Google. Jelikož se jedná o studio pro platformu Android tak programování probíhá v jazyce Java. Studio je plně zdarma a je k dostání na všechny platformy, tedy Windows, Linux a Mac OS. Jak již bylo zmíněno tak Android studio je od společnosti Google a funguje na IntelliJ IDEA CE (komerční vývojové prostředí pro programování v jazycích Java, Groovy a další). Oficiálně bylo Android studio představeno na konferenci Google I/O 16. května 2013.

Mezi velké výhody Android Studia patří to, že při instalaci je nainstalováno vše potřebné (včetně emulátoru) a není nutné doinstalovávat další věci a také to, že při návrhu UI je možné si nechat zobrazit aplikaci na více různě velikých obrazovkách zařízení, kde se okamžitě pomítají provedené změny.

Výhody

- rychlost
- menší nároky na výkon PC, především paměť
- rychlá a jednoduchá instalace

Nevýhody

- menší počet pluginů
- nemožnost vytvářet jiné projekty než pro Android

[3] http://developer.android.com/tools/studio/index.html

Xcode[4]

Xcode je oficiální vývojové prostředí pro iOS a Mac OS X. Toto vývojové prostředí je zdarma ke stažení z App Store, ale nezbytností je vlastnit Mac OS X. Bohužel zde neexistuje možnost vývoje nativní aplikace pro iOS bez toho, aby člověk vlastnil počítač od společnosti Apple.

Vývoj pobíhá v jazyce Objective-C. Jedná se o upravený jazyk C, který je objektově orientován. Každý objekt je zde reprezentován třídou a objekty spolu komunikují pomocí zpráv. Zprávy představují volání metod na ostatních objektech. Relativně novým přínosem prostředí Xcode je jazyk Swift, který má zpříjemnit vývoj aplikací a zrychlit jejich běh.

Výhody

- rychlost
- široká podpora od společnosti Apple

Nevýhody

- pouze na zařízeních s Mac OS X
- nemožnost vytvářet jiné projekty než pro iOS

Visual Studio[5]

Visual Studio je vývojové prostředí od společnosti Microsoft. První verze vznikla v roce 1997. Na rozdíl od výše uvedených vývojových prostředí lze ve Visual Studiu vytvářet více druhů aplikací od konzolových přes webové (ASP.NET apod.) až po mobilní aplikace. V případě mobilních aplikací je zde ovšem stejné omezení, jako u ostatní a to tedy takové, že bez Mac OS X nelze vytvářet aplikace pro iOS.

Visual Studio je především pro vývoj Windows Phone aplikací a je k dostání ve verzi Express, která je zdarma, ale také v profesionálních verzích, které poskytují více možností. V případě Windows Phone lze aplikace vytvářet v jazycích C#, C++ a případně i hybridní aplikace v HTML/CSS/JavaScript.

Výhody

- verze zdarma
- možnost výběru mezi více jazyky

Nevýhody

[4] https://developer.apple.com/xcode/
[5] https://www.visualstudio.com/

- vývoj pouze pro Android a Windows Phone (na počítači s operačním systémem Windows)

Xamarin[6]

Studio Xamarin slouží k vývoji aplikací pro mobilní zařízení s operačním systémem Android, iOS a Windows Phone. Rozdílem oproti Xcode či Android Studiu je to, že ve studiu Xamarin se programuje v jazyce C#. Rozdílu je dosaženo tím, že je aplikace kompilována do jazyku dané platformy.

Jsou zde dvě možnosti a to buď nainstalovat Xamarin knihovny do Visual Studia a posléze v něm vytvářet aplikace, nebo si rovnou stáhnout Xamarin Studio a vše vytvářet v něm. Xamarin lze získat po určitou dobu zdarma v rámci zkušební verze a posléze je nutné jej zakoupit.

Výhoda

- možnost vývoje aplikací v jednotném prostředí
- znalost C# (není potřeba pro každou platformu umět daný jazyk)

Nevýhody

- pouze placená verze
- pro iOS je potřeba Mac OS X

Alternativní vývojové prostředí

Pro vývoj hybridních aplikací existuje nespočet vývojových studií. Jako příklad lze uvést Intel XDK (28). Tyto studia slouží k vývoji aplikací za pomocí jazyku HTML, CSS a JavaScriptu. Výhodou je možnost vytvoření jedno aplikace a posléze ji kompilovat (rozdělit do balíčků) pro jednotlivé platformy. Aplikace se tváří, jako nativní, ale ve skutečnosti nativní nejsou. Další výhodou je jednotné uživatelské rozhraní aplikace. Většina studií je dostupná na platformy Windows, Linux a Mac OS X. Kompilace je zajišťována prostřednictvím frameworku Cordova. Apache Cordova je open source framework (dříve známý, jako PhoneGap) od společnosti Adobe, který slouží ke cross-platform vývoji aplikací za pomocí HTML, CSS a JavaScriptu.

Výhody

- dostupnost zdarma
- možnost vývoje na všechny mobilní platformy nehledě na operační systém počítače, na kterém je instalováno

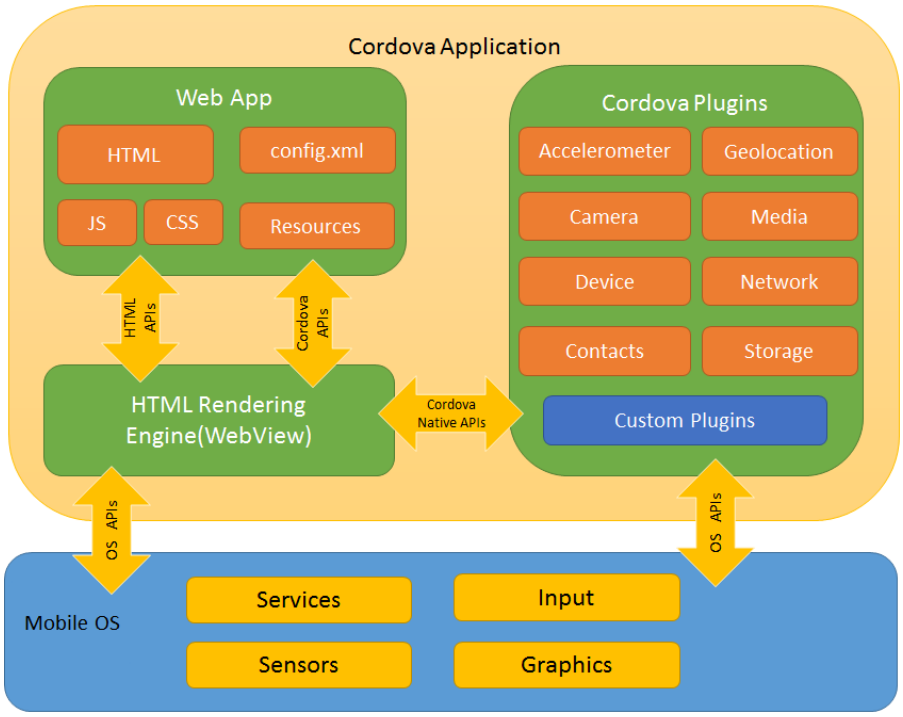
[6] https://xamarin.com/

Nevýhody

- pouze hybridní aplikace

3.6.5. Cordova

Jak již bylo zmíněno výše tak Cordova je Framework od společnosti Adobe a jedná se o open-source Framework pro vývoj mobilních aplikací. Umožňuje využívat webových technologií, jako HTML, CSS, JavaScript pro vývoj multiplatformních aplikací, které se vyhýbají vývoji v nativním jazyku aplikace. Aplikace je vždy zabalena do balíčku pro danou platformu, která umožňuje přístup k zařízení a jeho senzorům, datům, stavu internetového připojení a podobně.



Obrázek 1: Schéma přístupu k prostředkům Cordova (zdroj: 63)

Cordova využívá možností nových mobilních operačních systému, které obsahují možnosti přistupovat k prostředkům zařízení přes HTML a JavaScript.

Cordova je zdarma a vždy bude pod licencí Apache License Version 2.0.[7] Možnosti vývoje jsou dvě a to buď „Cross-platform (CLI) workflow“, nebo „Platform-centered workflow“. CLI je způsob vývoje pro všechny možné a dostupné platformy na trhu s minimem zásahů do úprav aplikace, jako takové. Ve výsledky se o vše stará vývojové prostředí. V případě Platform-centered workflow se jedná o vývoj jen na jednu platformu

[7] http://www.apache.org/licenses/LICENSE-2.0

s tím, že se zde mixuje nativní kód dané platformy a webové technologie. Jako příklad vývojového prostředí s Cordovou lze uvést Visual Studio od společnosti Microsoft[8].

3.7. Trendy v mobilních aplikacích a jejich tvorbě

S příchodem prvních smartphonů zažil trh s mobilními aplikacemi obrovský boom a během krátké doby se z neznámého stal velký byznys. Prvopočátky trhu s mobilními aplikacemi lze datovat k uvedení mobilního telefonu Apple iPhone, tedy do roku 2007. S postupem času se objevila spousta druhů aplikací, ale jen některé se ujali a dostalo se jim dostatečné popularity. Jako v každém odvětví zde byly různé slepé uličky, které neměli popularitu u zákazníků, jelikož bylo špatné načasování, nebo jen špatný koncept.

V dnešní době patří trh s mobilními aplikacemi mezi velmi ziskové a to především díky pár nenápadným trikům, jak na aplikaci zdarma vydělat miliony. V případě trendů je nezbytné rozdělit aplikace do kategorií, ale některé trendy jsou stejné napříč všem kategoriím. Rozhodl jsem se trendy rozdělit na dvě sekce a to trendy v mobilních aplikacích a trendy ve vývoji mobilních aplikací.

3.7.1. Trendy v mobilních aplikací

Co se týče trendů celkově tak většina článků v některých věcech shodu a v některých rozchází. Je to především způsobeno tím, že každý vidí trendy z jiného pohledu, respektive mají jiné názory na to, co je důležité a žádané trhem. Rozhodl jsem se tedy do této práce zapracovat také vlastní názory na danou problematiku. Velkou míru na vývoji trendů mají především módní značky, jako například společnost Apple v rámci designu, různé nové sociální aplikace, které expandují na všechny možné trhy a také nové technologie.

Z pohledu uživatele je to tedy zaměření na poskytování nových technologií (rozpoznání uživatele a podobně), zaměření se na cloud (zprávy, kontakty a podobně jsou uloženy na cloudovém serveru). Lze to tedy shrnout, že v případě uživatelů je kladen důraz na jednoduchost, intuitivnost aplikace a především mít vše po ruce bez zbytečného hledání navíc.

Například když si budu chtít objednat jídlo přes mobilní telefon tak mě nebudou zajímat restaurace, které jsou daleko od mé pozice, a tedy nebudou rozvážet. Ale na

[8] https://cordova.apache.org/

základě GPS polohy aplikace vybere pouze podniky, které vyhovují podmínkám a informace z poslední objednávky si uloží na server k danému uživateli a při příštím nákupu navrhne na základě posledního nákupu.

3.8. Trendy ve vývoji mobilních aplikací

V případě vývoje mobilních aplikací je zde orientace na výše uvedené, ale z technologického hlediska je to odlišné. Velký důraz je kladen především na ziskovost dané aplikace a k tomu jsou využívané především tyto modely: (19, 7)

- Platba při stažení
- Freemium
- Reklama
- Platby v aplikaci
- Předplatné

Dále mimo ziskovost je také velký důraz kladen na bezpečnost dané aplikace, jelikož bylo dost skandálů, kdy kvůli špatnému zabezpečení či chybám v aplikaci došlo k úniku citlivých dat uživatelů, což vytváří velmi špatný obraz firmy. Mezi největší úniky dat lze zařadit iCloud od společnosti Apple a SnapChat od Snapchat Inc.

Mezi další velké trendy v poslední době patří především multiplatformní vývoj z několika hledisek. Je to díky levnějšímu vývoji daných aplikací a s tím související větší rozšíření aplikace. Velkou výhodou je fakt, že aplikace má jednotné grafické rozhraní a s minimální úpravou aplikaci ji lze provozovat i jako webovou aplikaci. Tedy vytvořit aplikace na které lze přistupovat ze všech zařízení a mít data uložena centrálně.

S dostupností dat centrálně souvisí další trend a to je cloud řešení aplikace. V dnešní době, kdy je internet dostupný téměř všude včetně metra (v Pražském metru pouze na vybraných stanicích, ale například ve Vídni všude v metru) je žádané, aby lidé měli přístup ke svým datům vždy a všude. Díky tomu dochází k rozvoji cloudově založených služeb, kde některé mají pouze úložiště dat a následnou práci s nimi (posílání, úpravy a podobně), ale jsou i aplikace, kde je výpočetní výkon přenášen na cloudový server a mobilní zařízení pouze přijímá výstupy.

Relativně novým aspektem je tzv. wearable technology, neboli se jedná například o chytré hodinky. S příchodem této technologie došlo ke vzniku nových aplikací a podle růstu tohoto trhu se časem ukáže, jestli bude tento trh výhodný, jako ostatní.

Jelikož je dnes přístup k internetu hodně přístupný tak s tím vzniká i ochota uživatelů platit on-line. S růstem on-line plateb v e-shopech dochází k růstu aplikací, přes které lze platit a nejedná se o obchody dané platformy. Jako příklad lze uvést aplikace od společnosti PayPal a jejich využití na mobilní platformě. Vznikají s tím samozřejmě obrovské nároky na bezpečnosti, ale také možnost zakomponovat platbu přes tyto aplikace do jiných aplikací a usnadnit tak život uživatelům.

V neposlední řadě dochází k rozvoji tzv. Internetu věcí (Internet of Things). Internet věcí je propojení chytrých spotřebičů a jejich následné ovládání přes chytré zařízení. Jedná se tedy například o propojení různých senzorů, klimatizací, spotřebičů, automobilů a dalších prvků chytrého obydlí s tím, že jej lze ovládat a spravovat vzdáleně. V případě, že v létě pojedu z práce a budu mít doma vysokou teplotu mohu vzdáleně zapnout klimatizaci a v momentě kdy dorazím domu bude již obydlí vychlazeno. Rozvoj této technologie je pozvolný vzhledem k tvůrcům chytrých spotřebičů, ale jelikož je tento trh na vzestupu tak lze říci, že dříve nebo později se to v tvorbě mobilních aplikací projeví.

3.8.1. Bezpečnost

Zabezpečení mobilních aplikací je obrovský problém zejména v dnešní době, kdy jsou podvodníci čím dál tím více rafinovaní. Uživatelé často instalují aplikace do mobilních zařízení bez přečtení přístupů v rámci zařízení, nebo vůbec neřeší bezpečnost, pokud jim aplikace přijde vhodná, nebo potřebná.

Aplikace může bezpečnost uživatele ohrozit v několika případech. Jak již bylo zmíněno tak jedním z případů je aplikace, která může přistupovat v zařízení k věcem, které nepotřebuje ke své funkci. Například může aplikace přistupovat ke zprávám a kontaktům, které jsou v telefonu uloženy a posléze je odesílat tvůrci aplikace, který s nimi může nakládat podle libosti.

Větším rizikem je fakt, že může v případě instalace nebezpečné aplikace dojít k monitorování zařízení. Tedy uživatel dává možnost tvůrci aplikace sledovat vše, co na telefonu dělá, jaké stránky navštěvuje, jaké aplikace využívá, ale i polohu kde se nachází. V tomto případě je riziko zneužití obrovské, jelikož může mít přístup také k mikrofonu, fotoaparátu a sledovat takto veškeré dění daného uživatele.

Nejhorší možnou kombinací ovšem je, pokud uživatel stáhne aplikaci, která monitoruje celé zařízení a využívá v mobilním zařízení internetové bankovnictví či aplikaci pro přístup k bankovnímu účtu. V takovém to případě uživatel tvůrci aplikace bez

vědomí sdělil přístupové údaje včetně potvrzující SMS. Je to proto, jelikož aplikace v telefonu zachytí SMS zprávu a informace odešle tvůrci, který má možnost převod peněž potvrdit.

Z výše uvedených důvodů vznikla kontrola aplikací při nahrávání do obchodu na každé platformě. V případě platformy Android je uživatel o povolení přístupu ke zdrojům požádán před instalací aplikace. U platformy iOS jsou oprávnění vyžadována ve chvíli, kdy se k nim aplikace poprvé pokouší přistoupit a tímto připadá odpovědnost na uživatele. V případě Windows Phone je to schvalováno při nahrávání na obchod. I přes snahu společností omezit tyto hrozby dochází k tomu, že uživatelé nečtou, co povolují a bez přemýšlení vše potvrdí a posléze na to mohou doplatit (jedná se především o platformu Android, jelikož samotná platforma obsahuje spousty bezpečnostních chyb).

3.8.2. Platba při stažení

Jedná se o základní model, kdy vývojář neřeší získání peněž z aplikace. Vývojář aplikace vydělává každou staženou aplikací a více nemusí řešit. Důležité je, aby profil aplikace předal zákazníkovi informace a zaujal jej na tolik, aby se rozhodl ke koupi aplikace bez možnosti ji vyzkoušet zadarmo. Do jisté míry toto ovlivňují komentáře uživatelů, kteří si již aplikaci pořídili.

Zavést platbu při stažení je vhodné u aplikace, která nabízí více než konkurence, nebo něco, co nelze jinde sehnat. V dnešní době je konkurence obrovská a je proto velmi složité přijít s něčím, co neexistuje, nebo obohatit o něco, co by předčilo konkurenci. Lze využívat různého druhu propagace, jako v případě výrobků (za předpokladu, že se jedná o aplikace od velké společnosti).V případě provizí si každý obchod určuje jinou část z prodejů, ovšem nepřesahuje to 30%.

3.8.3. Freemium

Jedná se o model, kdy je aplikace nabízena ke stažení zdarma, ale část funkcionality je zpoplatněna. Uživatel tedy není omezen v přístupu k základní funkci aplikace, ale v případě, že uživatel projeví zájem o nějaké rozšířené funkce tak je zpoplatněn. V případě her se jedná o to, že například pár úrovní či epizod je zdarma, ale v případě odemčení další části hry je nutné zaplatit.

Tento model je zaplacen na tom, že se snaží nalákat uživatele a posléze z něj vytáhnout peníze. Většinou se jedná o takové aplikace a hry, které během chvilky zaujmou na tolik, že je uživatel zmaten a zakoupí jej.

Pozitivem tohoto modelu je fakt, že uživatel nejdříve vyzkouší aplikace a nedochází k nákupu něčeho, co nezaujme, nebo nesplňuje požadavky. Tento model lze zavést u téměř jakékoliv aplikace.

3.8.4. Reklama

Reklamu v aplikaci lze využít téměř v jakémkoliv modelu. Je nezbytné si dát pozor na to, aby v případě zakoupení aplikace (premium účet a podobně) došlo k vypnutí reklam. Reklamy se využívají především u aplikací, které jsou zdarma ke stažení s cílem vybudovat širokou uživatelskou základnu. V praxi to vypadá tak, že v momentě, kdy uživatel zapne a začne využívat aplikace se ukazují reklamní bannery s odkazem na nějaký produkt či službu. Vývojář získává provizi za každý provedený pro klik přes banner a v případě chytrého umístění reklamy může docházet k pro kliknutí velmi často. Díky tomu to způsobu vyplácení odměn vývojáři dochází k častému zobrazování bannerů náhle a ve chvíli, kdy se očekává interakce uživatele tak, aby uměle navýšili počet pro kliků.

Každá platforma má jiné pravidla pro zobrazování a umisťování reklam. Většinou je zde kontrolována a určována velikost reklam a možné umístění, jelikož se obchody jednotlivých platforem snaží zabránit vyskakování a jinému druhu překážení reklam, které následně způsobují buď nepoužitelnost aplikace, nebo pohoršení u uživatelů.

Výhodou je pokud vývojář využívá rozšířeného poskytovatele reklamy a je možné reklamy personalizovat a tím získat více pro kliknutí. Ovšem nevýhodou je, že v případě nevhodně umístěné reklamy či velkého množství reklamy dojde k znechucení uživatele a rozhodne se danou aplikaci smazat.

3.8.5. Platby v aplikace

Platby v aplikaci využívají především hry, kde se využívá virtuální měny, nebo jiného zboží, které může uživateli v aplikaci přinášet časovou výhodu, nebo jej zvýhodňovat. Jedná se tedy o případy, kdy je aplikace ke stažení zdarma včetně veškeré funkcionality a jejím cílem je přimět a zaujmout uživatele na tolik, aby byl ochotný utratit peníze za virtuální měnu a jiná různá vylepšení.

Ve zkratce se nejčastěji jedná o aplikaci, kdy vytvoření objektu a podobné záležitosti zaberou určité množství času a lze je urychlit zaplacením nějaké měny v dané aplikaci. Měnu lze získávat v malém množství přímo v aplikaci, ale samozřejmě je tu možnost si ji zakoupit a vše tak urychlit. Tento způsob je především účinný u dětí, které bez rozmyslu zaplatí nemalé částky, aby mohli hrát.

Výhodou je možnost nabízet několik různých možností nákupu. Tímto lze zaujmout uživatele, kteří chtějí utratit a zároveň přidávat do aplikace mnoho vylepšení a vydělávat na nich.

3.8.6. Předplatné

Tento model se nezaměřuje primárně na zpoplatnění funkcionality dané aplikace, ale na její obsah, nebo obsah obecně. Aplikace je pro uživatele volně ke stažení s určitým množstvím obsahu, které je často na omezené období a po uplynutí tohoto období je vyzván k registraci a zaplacení. Jako příklad lze uvést Spotify, které je zdarma, ale v případě, že chce člověk naplno využívat potenciálu této aplikace tak je nezbytné si zaplatit premium účet (za každý měsíc).

Předplatné je vhodné pro aplikace, u kterých je obsah často aktualizován, nebo se mění a umožňuje jej zpoplatnit. Většinou se lze setkat s tím, že je část obsahu dostupná pro všechny uživatele zdarma, ale prémiový, nebo nejnovější obsah je již zpoplatněn. Velkou výhodou je fakt, že v případě zaplacení předplatného je zde velká šance na získání dlouhodobého zákazníka a celkově je zde měsíční pravidelný příjem. Nevýhodou ovšem je fakt, že tento model nelze využít ve všech typech aplikací.

V dnešní době se dbá hlavně na jednoduchý a rychlý (časově nenáročný) přístup ke správným datům ve správný čas a na základě toho si lze účtovat přiměřenou odměnu. V dnešní době je především zavedena kombinace několika druhů zpoplatnění aplikací naráz, aby došlo k maximalizaci zisku.

3.9. Volba vývoje

Rozhodnout, jaký způsob vývoje mobilní aplikace zvolit leží na bedrech jedné, nebo více osob podle druhu živnosti. Pokud se jedná o jediného vývojáře pracujícího samostatně tak je volba pouze na něm. Jedno je ovšem jisté a to je fakt, že by do tohoto rozhodnutí měli zasahovat budoucí vývojáři aplikace a správci stávajícího systému pokud existuje. V případě střední firmy či korporátní sféry jsou doporučení pro co se rozhodnout, ale finální rozhodnutí musí učinit osoba, která má k tomu potřebnou pravomoc. Často může dojít k tomu, že tuto pravomoc má osoba, které nemá o této problematice žádné informace a v lepším případě může jít alespoň o běžného uživatele mobilních aplikací.

Nejprve je nutné rozhodnout, jestli aplikace bude v prohlížeči (tedy webová), nebo jestli ji bude potřeba nainstalovat. Dále je nezbytné se rozhodnout, jestli chceme vyvíjet pouze na jednu platformu, nebo na více (multiplatformní). Nejdůležitější otázkou je: **Bude**

aplikace vyžadovat vysoký výkon? V případě, že tedy není nutnost vysokého výkonu se lze rozhodnout na základě dalších faktorů a to především náklady na vývoj a podobné, které se mohou pro každou společnost či vývojáře lišit.

3.10. Moodle[9]

V případě informačního panelu lze propojit serverovou část aplikace se stávajícími systémy běžící na České zemědělské univerzitě v Praze. Jedním ze systému je volně šiřitelný systém Moodle. Moodle je vzdělávací systém pro tvorbu kurzů a výukových systémů. Systém Moodle slouží především k vedení kurzů a sdílení studijních materiálů. Také lze provádět testy včetně výsledků, vytvoření statistik, sledování docházky, nebo například termín odevzdání (včas / se zpožděním). Výhodou systému je dostupnost pluginů a možná budoucí rozšíření, která zvyšují použitelnost systému Moodle. Systém je upravitelný, jak vzhledově tak i v rámci uspořádání prvků na nástěnce a podobně. V rámci informačního panelu nelze využít systému Moodle, respektive v systému Moodle nejsou dostupná potřebné informace.

3.11. IS.CZU.CZ[10]

Mezi další systémy, které běží na České zemědělské univerzitě v Praze je informační systém UIS. UIS (Univerzitní Informační Systém) je komplexní řešení pro správu a řízení univerzity. Tento systém využívá například i Vysoká Škola Ekonomická v Praze. V systému je správa studentů, vyučujících, prací (bakalářské, diplomové a podobně), rozvrhů včetně zápisů do předmětu a tak. Výhodou je, že jsou zde údaje o vyučujících a studentech a lze tento systém využít pro serverovou část této aplikace. Z UIS lze vzít následné informace o vyučujícím:

- Jméno, příjmení a příslušné tituly
- Konzultační hodiny
- Fotografii
- Kontaktní údaje včetně názvu místnost

Jediné co nelze získat z UIS je stav vyučujícího (přítomen / nepřítomen / zaneprázdněn). Tuto část je nezbytné implementovat v rámci webového rozhraní serverové části aplikace.

[9] https://moodle.org/
[10] http://www.uis-info.com/cs/index

3.12. Google[11]

Společnost Google poskytuje služby emailem počínaje přes cloudové úložiště po kancelářské balíky. Samozřejmě společnost Google poskytuje i jiné služby (například Google Analytics), ale ty jsou vzhledem k informačnímu panelu nepotřebné. Ke službám od společnosti Google se využívá Google API, které je volně dostupné. V rámci informačního panelu by bylo možné využít pouze Google Calendar a Picasa Web Album. Google Calendar by se dalo využít pro získávání konzultačních hodin (den a čas). Výhodou je dostupnost na zařízeních (desktop, mobilní zařízení) a vzájemná synchronizace. Picasa Web Album lze využít pro doplňkové služby místo počasí, kde by mohly být zobrazovány fotky univerzity, univerzitních akcí a podobně. Výhodou využití Picasa Web Album je synchronizace a jednoduchost nahrání fotek z PC.

Vzhledem k existenci a využívání UIS na univerzitě, je zbytečné využívat Google API. Při využití Google API by bylo nezbytné získávat data z UIS i Google API což by mohlo způsobit problémy v případě výpadku jedné ze služeb. Jelikož data o konzultačních hodinách a vyučujících se nemění často tak není nutné využívat Google API a bohatě stačí propojení s UIS.

3.13. Shrnutí

V teoretické části jsem popsal nezbytné základy pro úvod do problematiky vývoj informačního panelu a celkově mobilních aplikací. Z teoretické části vychází najevo, že v této práci využiju pouze UIS pro serverovou část. V praktické části se rozhodnu pro vhodný způsob tvorby aplikace, propojení se serverovou částí aplikace a následnou tvorbu.

[11] http://wiki.gug.cz/pro-vyvojare-1/google-api

4. Praktická část

V praktické části se zaměřím na tvorbu aplikace v rámci dané problematiky. Začnu definováním požadavků a následnému rozhodování pro volbu vhodného způsobu vývoje aplikace, které byly zmíněny v teoretické části práce. Po výběru způsobu vývoje aplikace je nezbytné zvolit vývojové prostředí. Práce se zabývá vývojem od návrhu aplikace až po sestavení a testování na zařízení.

4.2. Zadání

Ze zadání jsem vyvodil problematiku, kterou pokrývá moje aplikace. Problematikou je vyhledání konzultačních hodin vyučujícího bez nutnosti navštívit internetové stránky univerzity. Další důležitou částí problematiky, kterou jsem řešil na základě zadání je, zjištění přítomnosti vyučujícího v místnosti (kabinetě), kontaktní údaje a fotografie. Potřebné data získá aplikace ze serveru, kde půjde upravovat informace o stavu, vyučujícím a podobně. Student očekává, že po vyhledání dané místnosti nalezne před místností název místnosti, vyučujícího (případně více vyučujících), stav (přítomen/nepřítomen), kontaktní údaje a fotografii.

4.3. Funkcionalita

Z výše uvedeného tedy vyplývá, že myšlenkou této aplikace je prezentovat aktuální data u každé příslušené místnosti s lehkou nápravou případných chyb. Jak již bylo zmíněno tak aplikace bude získávat data ze serveru, ke kterému budou mít kantoři přístup z počítačů, kde mohou například měnit stav přítomen/nepřítomen, konzultační hodiny, dny a další informace. Cílenou funkcionalitou je to, že člověk přijde k místnosti (kabinetu), podívá se na tablet, kde budou zobrazeni všichni lidé působící v dané místnosti. Vedle jména bude indikátor stavu (přítomen/nepřítomen) a po rozkliknuti dojde k zobrazení dodatečných informací.

V případě výskytu volného místa v aplikace je možné jej zaplnit například počasím, které každý ocení, nebo čímkoliv jiným.

4.4. Volba vývoje a prostředí

Volba způsobu vývoje není lehká, jelikož je zde spousta ovlivňujících faktorů. Volbu způsobu vývoje jsem zkoumal především z pohledu finanční náročnosti, jelikož je podle mého názoru zbytečné vydávat více peněž, než je doopravdy nezbytné jen kvůli tomu, aby zařízení vypadalo dobře. Hlavními faktory, které ovlivnily mé rozhodování jsou:

- Cena zařízení
- Nezávislost na platformě

Cena zařízení je myšlena tak, že vzhledem k typu aplikace, kde není kladen důraz na výkon zařízení není nutné, aby aplikace běžela na drahém a výkonném modelu. Ve výsledku to znamená úsporu finančních prostředků (nákladů) na pořízení tabletů, kde rozdíl může být v řádě tisícikorun za kus.

Nezávislost na platformě navazuje na cenu zařízení tak, že v případě postupného dokupování tabletů může docházet k migraci na jinou platformu a proto je vhodné, aby nebylo nutné aplikace upravovat, nebo dokonce předělávat, ale jen ji nahrát na nové zařízení.

Z výše uvedených faktorů je tedy jasné, že nejvhodnější volbou je hybridní aplikace. Hybridní aplikaci jsem zvolil, protože její vývoj je relativně snadný, je multiplatformní a je zde široká škála technologií, které lze využít. Při volbě vývojového prostředí jsem hledal takové, které je volně dostupné a obsahuje vše nezbytné. Vhodné je, aby bylo rozumné velikosti, přehledná a mělo podporu nejnovějších technologií a stále bylo vylepšované. Předem bych rád upozornil, že jsem zvolil vývojové prostředí na základě mých preferencí a je zde pravděpodobnost, že nemusí vyhovovat každému. Na základě rešeršní části jsem se rozhodl pro hybridní aplikaci a vývojové prostředí Intel XDK. Hybridní aplikaci jsem zvolil především z ekonomických důvody, kdy v dnešní době je možné sehnat tablet již za 1290kč[12].

4.5. Analýza zadání

Ze zadání jsem získal potřebné informace. Ovšem je nezbytné vyfiltrovat ty nejpodstatnější a případně je dále rozšířit o doplňkové vlastnosti aplikace. Základem tedy je informační panel v tomto případě tablet. Jedná se o tablet (mobilní zařízení) umístěné před místností, ke které bude zobrazovat příslušné informace. Na tabletu poběží aplikace, která bude prezentovat data získaná ze serveru a tím se dostáváme k zásadnímu bodu zadání a tedy získávání dat ze serveru. Tablet bude tedy zobrazovat název místnosti, vyučující (jméno, příjmení, titul(y), konzultační hodiny, kontakt) dále přítomnost či nepřítomnost vyučujícího (stav přítomnosti či nepřítomnosti lze měnit přes serverovou část

[12] https://www.alza.cz/ltlm-d7-premium-d2220095.htm?catid=18852388

aplikace, kde je webové rozhraní pro správu). Stav je možné rozšířit o položku zaneprázdněn. Lze tedy shrnout požadavky na aplikaci následovně:

- Název místnosti
- Jméno a příjmení vyučujících včetně titulu či titulů
- Konzultační hodiny (dny včetně časového rozpětí)
- Kontaktní informace (telefon, email)
- Stav přítomen / nepřítomen

Po stanovení základních údajů, které bude aplikace prezentovat přichází otázka se získáváním těchto dat. Původně bylo v plánu získávat data ze serveru, který měl být vytvořen kolegou v rámci jiné diplomové práce. Data by měla být posílána přes soubor JSON, který vysvětlím dále. Bohužel v momentě, kdy píši tuto diplomovou práci není k dispozici serverová část aplikace a v době dokončování aplikace a diplomové práce nebude k dispozici. Proto jsem se rozhodl serverovou část nahradit lokálním JSON souborem, takže zde nebude vliv na funkcionalitu aplikace a posléze stačí pouze pozměnit cestu k JSON souboru.

Jelikož se jedná o aplikaci zobrazující informace o konkrétní místnosti tak je zde kladen důraz na dynamičnost (při předpokladu, že bude více tabletů u více místností) a to především s tím, že se konzultační hodiny mohou lišit v každém semestru, ale hlavně stav přítomnost, nepřítomnost či zaneprázdněnost vyučujícího se liší v rámci vteřin. Tedy zde dochází k dělení na část relativně statickou (dny a hodiny konzultačních hodin, případně další titul a podobně) u které dochází k aktualizaci jednou za čas a na část dynamickou (stav přítomen a nepřítomen). Vyvstávají tedy dva základní problémy, kdy jedním je **získávání dat** a druhým **dynamičnost**. V případě dynamičnosti je nezbytné brát na zřetel fakt, že je zde možnost, že aplikace poběží na více tabletech zároveň a je proto nezbytné tomu přizpůsobit kód aplikace, aby zde nebyly na pevno nastavené hodnoty, nebo hodnoty proměnných. Závěrem kapitoly tedy shrnu nejdůležitější body této kapitoly do následujícího:

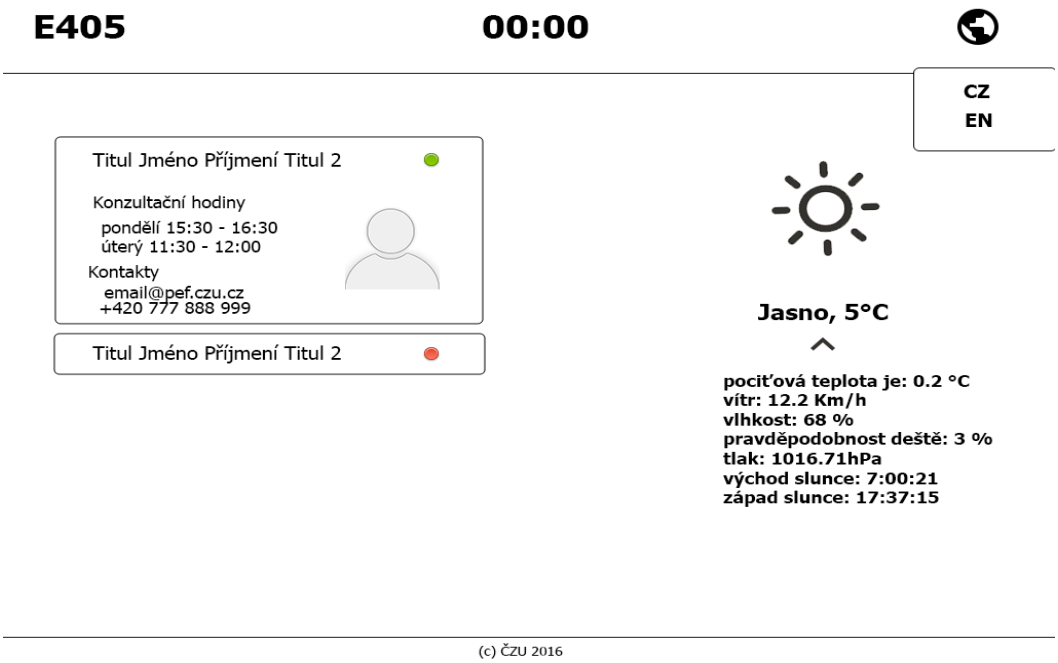
- Získání a zobrazení příslušného názvu místnost, jako například E405 v rámci České zemědělské univerzity v Praze.
- Získání a zobrazení příslušných vyučujících osob v dané místnosti.
- Získání doplňujících informací o vyučujících, jako jsou konzultační hodiny a kontaktní informace například telefon či email.

- Získání a udržení aktuálního stavu příslušného vyučujícího (přítomen, nepřítomen).
- Získání a udržení aktuálnosti veškerých výše uvedených dat.

4.6. Rozložení prvků

Po sepsání základních požadavků na aplikaci přichází další část a to logické rozložení prvků v aplikaci. První co je na mysli v případě tabletu je, jestli je lepší jej mít na šířku, nebo na výšku. Vzhledem k faktu, že uživatelé nejvíce pozornosti a času věnují levé horní části obrazovky (je jedno, jestli jde o mobilní aplikace, webovou prezentaci, nebo například Microsoft Word) tak je vhodné vše důležité v mísit do této oblasti od nejvíce důležitého po nejméně důležité. S přihlédnutím k tomuto faktu jsem se rozhodl pro výběr orientace na šířku, jelikož vznikne dodatečný prostor, který lze využít.

U volného prostoru, který vznikl orientací na šířku jej lze využít pro spoustu věcí, jako například novinky z univerzity, nadcházející akce, nebo například počasí. Samozřejmě vše má své pro a proti a nelze se zavděčit každému. Přemýšlel jsem nad tím, co bych jako student rád viděl za doplňující informaci, která by mne ulehčila život. Rozhodl jsem se tedy pro zobrazení aktuálního počasí s tím, že zde budou i dodatečné informace, jako například rychlost větru, vlhkost a podobně. Vytvořil jsem tedy následující rozložení prvků.



Obrázek 2: Rozložení prvků (zdroj: vlastní)

Z obrázku je patrné, jak jsem se rozhodl rozložit jednotlivé části aplikace. Počínal jsem tak, že jsem řadit nejpodstatnější informace z levé strany do pravé. V levé horní části je nejpodstatnější (stěžejní) části aplikace a dále jsou umístěny doplňkové, jako například volba jazyka. Tedy nejprve by si uživatel měl všimnout názvu místnosti a posléze seznamu vyučujících. V případě vyučujících dojde k zobrazení jejich jména, příjmení, titulů a stavu a po rozkliknuti dojde k zobrazení dodatečných informací. V právě části se nachází počasí, jak již bylo zmiňováno s tím, že pokud dojde ke kliknutí na šipku tak se zobrazí dodatečné informace.

Při tvorbě grafické (vizuální) stránky aplikace jsem se snažil využít barev fakulty, univerzity a případně související přírody. Vytvořil jsem více barevných schémat (viz přílohy 1, 2) s tím, že v testovací verzi aplikace lze mezi nimi volně přepínat.

Výsledek by měl být například takovýto.

Obrázek 3 příklad barevného schématu (zdroj: vlastní)

4.7. Vytvoření projektu

Před tvorbou grafické stránky samotné aplikace (stylizace přes CSS) jsem založil nový projekt ve vývojovém prostředí Intel XDK (28). Z nabízených možností jsem zvolil Blank Application (prázdná aplikace), která neobsahuje předdefinované prvky (pouze

napojení knihoven pro tvorbu hybridní aplikace). Po vytvoření projektu jsem nastavil orientaci na šířku a mód fullscreen (viz příloha 3). Fullscreen mód znamená, že dojde ke skrytí status baru (ukazatel baterie, čas, signál a podobně), ale i softwarových tlačítek. Jedinou nevýhodou fullscreen modu je, že nelze aplikaci ukončit, pokud zařízení nemá hardwarová tlačítka a je nutné zařízení restartovat. V tomto případě je to výhoda, jelikož studenti nebudou moci tablet využívat k jiným účelům. Jelikož jsem si vybral, že zde bude počasí tak jsem připojil plugin Geolocation (viz příloha příloha 3), který umožňuje přístup k geolokačnímu hardwaru zařízení. Potom jsem nastavil Launch icons (viz příloha 3). Využil jsem k tomu obrázku s logem univerzity (viz příloha 4) s tím, že jsem jej upravil na čtyři různé velikosti. Konkrétně se jedná o velikosti 36x36, 48x48, 72x72 a 96x96 pixelů. Dále jsem nastavil Splash Screen což je načítací obrazovka (obrázek při načítání aplikace). Nastavil jsem zde logo univerzity (viz příloha 4https://student.czu.cz/index.php?id_menu=374&id_submenu=407&id_con_kind=5&id_con=3103). Zde jsem musel opět upravit velikost na čtyři varianty a to na 320x426, 320x470, 480x640 a 720x960 pixelů. V tento moment jsem nastavil vše podstatné pro aplikaci a postoupil dále k tvorbě vizuální stránky.

4.8. Tvorba grafiky

Při převádění grafického schématu na kód CSS jsem rozdělil aplikaci do příslušných sekcí (divů). Aplikaci jsem rozdělil na tři základní sekce (elementy HTML 5) a to na *header* (hlavička), *main* (hlavní část dokumentu) a *footer* (patička). V rámci hlavičky jsou v této sekci tři pod sekce a to název místnosti, čas a jazyky. V rámci hlavní sekce, tedy elementu *main* jsem jej rozdělil na dvě části a to na vyučující (na levé straně) a počasí (na pravé straně). U elementu *footer* (patička) jsem nevytvářel žádný div, jelikož zde budou pouze informace o copyrightu.

Než jsem se pustil do samotné tvorby tak jsem si vytvořil čtyři css soubory. Jedná se o soubory header, main, počasí a vyučující (názvy souborů jsou bez diakritiky). Rozdělil jsem je kvůli přehlednosti a snadné orientaci s tím, že v každém souboru se řeší formátování příslušné části aplikace. Pro nastavení barev a jednotlivých doplňků jsem vždy vytvořil nový soubor css, kde dochází k nastavení příslušných barev a podobně. Jedná se tedy o osm souborů nastavující různé grafické prvky. Důležité je zmínit, že jsem se snažil veškeré velikosti nastavit v relativních jednotkách, tedy v procentech. Relativní jednotky jsou zvolené, protože dochází k přizpůsobování grafiky k rozlišení, které se liší u přístrojů.

Začnu od shora a to hlavičkou. V sekci header jsem vytvořil čtyři divy. První div zobrazuje název místnosti s id *mistnost*. Jako další je div čas, který má id *cas* a div jazyk s id *jazyk*. Poslední divem je div, který je v základu neviditelný a má id *jazyk2*.

V následující sekci main jsou dva divy. První div je vyučující s id *vyucujici*, ve kterém dochází ke generování divů (podle počtu vyučujících v místnosti), tedy jsou zde minimálně dva. Druhý div je počasí s id *pocasi*, ve kterém jsou další divy a s konkrétními id *pocobr*, *poctext* a *pocasivice*. Rozdělení na tři divy je úmyslně pro lepší práci s jednotlivými daty. Poslední sekcí je footer, tedy patička ve které nejsou žádné divy.

Nejdříve se zaměřím na css soubor *header.css*, který je o nastavení formátu pro hlavičku aplikace. Vezmu to postupně od prvního divu v levém horním rohu a budu postupovat podle sekcí (header, main, footer) od shora dolů. Prvním divem je div *mistnost* (podle id). Zde jsem nastavil pouze float (obtékání) na left, jelikož chci aby název místnosti byl vždy v levém horním rohu. Dále jsem nastavil šířku na 10% z celkové šířky aplikace (zde je použití relativních jednotek a tudíž je to nezávislé na konkrétním rozlišení). Ke konci jsem nastavil zarovnání textu na střed divu (aby nebyl text nalepený přímo na kraj aplikace) a zvolil velikost 5vw. Vw je velikost, která se vypočítává ze šířky zobrazované oblasti (v tomto případě aplikace). Jednotky vw jsou uváděné v procentech, tedy 5vw je 5% zobrazované oblasti.

Dalším divem je *cas*. V tomto divu je zobrazován aktuální čas. Nastavil jsem šířku na 80% se zarovnáním textu na střed (text-align: center), obtékáním z leva (float:left) a velikostí textu 5vw. Následuje div *jazyk*, který slouží k zobrazení schovaného divu *jazyk2*. V tomto divu je zobrazen pouze obrázek. Nastavil jsem proto obtékání zprava, aby byla zeměkoule vždy v na pravé straně. Dále jsem nastavil margin (zarovnání) z leva a zprava na auto čímž došlo k vycentrování obrázku v rámci divu. Šířka je nastavena pouze na 4.5% což je dostačující vzhledem k velikost zeměkoule. V rámci sekce *header* je zde poslední div a to *jazyk2*. Tento div je neviditelný nastavením atributu display na hodnotu none. Šířka je nastavena na 10% a velikost textu na 2vw. Dále je zde odsazení od vrchu o 10%, jelikož velikost hlavičky je 10%. Text je zarovnán na střed a levý dolní roh je zaoblen o 5 pixelů. Mezera mezi řádky je zde zvýšena o 50% (line-height), tedy na 150%. Důležitým atributem je zde pozice, která je nastavena na fixed. To znamená, že ať je uživatel scrollovaný níže v aplikace stále má na vrchu zobrazovaného obsahu k dispozici daný div (nezůstává na úplném vrchu aplikace, ale pohybuje se společně s uživatelem). Jelikož zde

jsou tlačítka pro změnu grafického rozhraní tak jsem nastavil pro tlačítka v tomto divu (*jazyk2*) nulové ohraničení, průhledné pozadí (bez pozadí) a velikost textu 2vw. Tímto jsem nastavil formát celé hlavičce a mohu se přesunout na hlavní sekci (main).

V css souboru *main* jsou pouze základní nastavení hlavičky, těla (*main*) a patičky. Hlavičce je nastavena šířka 100%, aby nebyly po stranách bílé okraje. Výška je nastavena na 10%, jak již bylo zmíněno výše a z-index na hodnotu 1. Z-index slouží k řazení divů v rámci z osy. Spolu s z-indexem jsem nastavil pozici na fixed, top 0, left 0. Toto vzájemné nastavení způsobilo to, že v případě kdy uživatel scrolluje níže a přejíždí přes ostatní divy je vždy viditelná hlavička. Bez z-indexu by byly divy zobrazeny přes hlavičku. Dále je zde nastaveno odsazení sekce *main* shora o 10% a automatická výška (kvůli generovaným divům). V poslední řade je zde nastavení patičky. Zde jsem nastavil pozici na fixed (vždy bude na spodu obrazovky), bottom 0 a left 0. To má za následek, že zde nejsou okraje a odsazení. Velikost textu jsem nastavil na 1.5 vw a vycentroval text. Dalším css souborem je *vyucujici*, ve kterém je nastavení generovaných divů a divu *vyucujici*.

Tyto dva divy na rozdíl od ostatních mají nastavené třídy, jelikož se vyskytují více jak jednou. Také proto, že chci nastavit stejné vlastnosti všem těmto divům nehledě na to, jestli se jedná o první, nebo i-tý div. V prvním divu je vložený další div s třídou *stavy*, ve kterém dochází k zobrazování stavu (zobrazení vhodného obrázku). Div *vyucujici* ve kterém jsou generovány divy má nastavený obtékání vlevo a šířku 60%. Prvnímu divu (třída jedna) jsem nastavil zarovnání z leva a zprava na auto (vycentrování), velikost písma 2vw, z-index na 0 (kvůli hlavičce), mezeru mezi řádky na 250% a šířku 77%. V případě výšky nefungovalo nastavení v relativních jednotkách (nejspíše protože nebylo jasné z čeho vypočítávat velikost) a proto jsem nastavil fixní výšku 50 pixelů.

Třídě *dva* jsem nastavil neviditelnost (display:none), aby se zobrazil pouze v případě kliknutí na div s třídou *jedna*. Nastavení je víceméně stejné, jako u třídy jedna s rozdílem ve velikosti písma (zde 1.5vw), výškou 150 pixelů a s výškou řádků 150%. Dále jsem nastavil pro obrázky v divech s třídou *dva* obtékání zprava, které způsobí, že obrázek bude vždy na pravé straně. Posléze jsem nastavil výšku na 70% a šířku na 20%. Poslední třídou, kterou jsem v rámci generovaných divů nastavoval je třída *stavy*. V této třídě jsem nastavil pouze obtékání zprava, aby byl stav vždy na pravé straně, relativní pozici, top 22% a right 7%. Relativní pozice znamená to, že se počítá pozice v rámci daného divu a nikoliv v rámci celé aplikace.

Posledním souborem css bez nastavování barev je *pocasi.css*. Do tohoto souboru jsem vložil veškeré nastavení pravé části hlavního divu (*main*). Je zde nastavení pro divy *pocasi*, *pocasiobrazek*, *poctext*, *dolu*, *nahoru* a *pocasivice*. Vezmu to postupně a začnu divem s id *pocasi*. Jedná se o div, ve kterém jsou ostatní divy a nastavil jsem zde šířku na 40%, fixní pozici (position fixed), obtékání z leva a odsazení shora o 12%. Obtékání zleva je zde proto, aby nebyl tento div pod divem s vyučujícími, ale napravo vedle něj. Co se týče fixní pozice tak to je proto, aby při scrollování bylo vždy vidět počasí. Další v pořadí je *pocasiobrazek* v tomto případě se nejedná o nastavení divu, ale elementu canvas (HTML 5 element (20)). Nastavil jsem zde výšku a šířku na 70%, odsazení z leva a zprava na auto (vycentrování) a odsazení uvnitř divu (padding) na 5% shora a zdola. Odsazení uvnitř divu je z čistě estetického hlediska.

Následovalo nastavení divu *poctext*, který zobrazuje základní údaje o počasí, tedy shrnutí (slunečno a podobně) a teplotu. Zde jsem nastavil velikost textu na 3vw, zarovnání textu na střed a šířku 100%. Dále jsem nastavil obrázkům nahoru a dolu relativní pozici a zarovnání na střed. Schválně zde pracuji s id pro jednotlivé obrázky a nevyužití třídy, kvůli práci s těmito elementy. Jako poslední jsem nastavil div s id *pocasivice*. V tomto divu jsou zobrazovány dodatečné informace o počasí. Je tedy v základu neviditelný (display:none), dále jsem nastavil vycentrování pomoci odsazení (margin), velikost textu 2vw, šířku 75%, zarovnání textu na střed a odsazení od spodu o 10% kvůli *footeru*.

V tomto okamžiku mám nastavené formátování elementů a dále se pustím na nastavení grafiky. Mám zde osm css souborů, ale s tím, že u většiny je nepatrný rozdíl. Rozdělil jsem css soubory do dvou kategorií a to **simple** a **modern**. Každá kategorie obsahuje čtyři css soubory a začnu kategorií **simple**. V kategorii simple jsou čtyři css soubory a to:

- cervenasimple
- hnedasimple
- modrasimple
- zelenasimple

Začnu souborem *hnedasimple*, jelikož je nastaven, jako výchozí css soubor. Nejdříve jsem nastavil barvu hlavičce (#34322e) a barvu písma (#f3f4e5). Vycházel jsem z kombinace barev hnědá a světla žlutá. Dále jsem nastavil font písma na Tahomu. Barva písma v hlavičce je stejná, jako barva pozadí v elementu body (*main* div) a v celém tomto

schématu jsem využil pouze dvě barvy (výše uvedené). V rámci hlavičky jsem nastavil divu *jazyk2* hnědé pozadí (barva stejná, jako v hlavičce) a barvu písma stejnou, jako v hlavičce. V rámci tlačítek (button) v divu *jazyk2* jsem nastavil pouze barvu písma na #f3f4e5. U patičky jsem postupoval stejně, jako v případě hlavičky a nastavil barvu pozadí hnědou (#34322e) a barvu textu světle žlutou (#f3f4e5). Textu o počasí (*poctext, pocasivice*) jsem nastavil hnědou barvu písma (#34322e). Nejvíce formátování proběhlo u tříd *jedna, dva*. Ve třídě jsem nastavil hnědou barvu pozadí, světle žlutý text a zaoblení ohraničení (celého divu) o 5 pixelů. U třídy dva jsem nastavil hnědou barvu pozadí, světle žlutý text a spodní levý a pravý okraj zaoblený o 5 pixelů. Toto zaoblení se mění s rozklinkáváním divů s třídou *jedna*.

Dalším css souborem je soubor *cervenasimple.css*. Jak již z názvu vyplývá tak jsem využil barev Provozně ekonomické fakulty. Postupoval jsem stejně, jako v případě hnědé varianty s mírnými rozdíly. Nejdříve jsem nastavil pozadí hlavičce na červenou barvu (#b62a33) a barvu textu na bílou (dostupná přes označení white). Elementu body jsem nastavil bílé pozadí a font Tahoma. V případě divu *jazyk2* jsem nastavil červené pozadí a bílý text s tím, že pro tlačítka v divu *jazyk2* jsem nastavil také bílou barvu textu. V případě patičky jsem nastavil červené pozadí a bílý text (shodné s hlavičkou). Divům *poctext* a *pocasivice* jsem nastavil pouze černou barvu textu (dostupná přes označení black). Divům s třídou *jedna* (divy obsahující jméno, příjmení atd.) jsem nastavil bílé pozadí, černou barvu textu. Opět jsem nastavil zaoblení ohraničení o 5 pixelů, ale k tomu také ohraničení o tloušťce jednoho pixelu v barvě hlavičky (#b62a33). Jako ohraničení jsem zvolil nepřerušovanou čáru (solid). Ke konci jsem nastavil spodní zaoblení levého a pravého rohu o 5 pixelů.

Následujícím souborem na seznamu je soubor *modrasimple.css*. V tomto souboru jsem použil barvy bílá, černá a modrá. Hlavičce jsem nastavil modrou barvu pozadí (#6093ac) a bílou barvu textu. Dále jsem nastavil bílé pozadí elementu body a font Tahoma. Divu *jazyk2* jsem nastavil modré pozadí a bílou barvu textu s tím, že všem elementům button v tomto divu jsem nastavil barvu textu na bílou. V případě patičky jsem postupoval stejně, jako u hlavičky. Tedy nastavil jsem modrou barvu pozadí a bílou barvu textu. Divům *poctext* a *pocasivice* jsem nastavil barvu textu na černou. V případě divů s třídou jedna jsem nastavil bílé pozadí, černou barvu písma. Opět jsem nastavil zaoblení o 5 pixelů a ohraničení o tloušťce jednoho pixelu. Ohraničení je v modré barvě a

nepřerušovanou čarou (solid). Na závěr jsem třídě dva nastavil bílé pozadí, černou barvu textu a zaoblení spodních rohů (levý a pravý) o 5 pixelů. Dále jsem nastavil ohraničení pro dolní, levou a pravou stranu divu na jeden pixel nepřerušované čáry v barvě modré.

Poslední souborem ze seznamu je *zelenasimple.css*. Zde jsem nastavil hlavičce zelenou barvu pozadí (#006940) a bílou barvu písma. Tělu jsem nastavil bílé pozadí a černou barvu textu. Divu *jazyk2* jsem nastavil zelenou barvu pozadí s bílou barvou textu. Všem tlačítkům v rámci divu *jazyk2* jsem nastavil bílou barvu pozadí. U patičky jsem nastavil zelenou barvu pozadí (stejnou, jako v případě hlavičky) a bílou barvu textu. Divům *poctext* a *pocasivice* jsem nastavil černou barvu textu. Divům s třidou jedna jsem nastavil bílou barvu pozadí, černou barvu textu a zaoblení o 5 pixelů. Také jsem nastavil ohraničení o velikost jednoho pixelu v zelené barvě s nepřerušovanou čárou. U třídy dva jsem nastavil bílé pozadí, černou barvu textu a zaoblení spodní pravé a levé části divu o 5 pixelů. Posléze jsem nastavil ohraničení pro spodní, levou a pravou část divu v zelené barvě o velikosti jednoho pixelu nepřerušovanou čárou.

V tomto momentě mám již vytvořenou jednu kategorii grafických schémat a pokračoval jsem na druhou nazvanou **modern.** V této kategorii jsou opět čtyři css soubory a to:

- cervenamodern
- modramodern
- oranzovamodern
- zelenamodern

Jelikož jako výchozí schéma jsem nastavil soubor „hneda.css“ tak to zde vezmu podle abecedního pořadí. Začnu souborem „cervenamodern.css“.

Začal jsem nastavením pozadí. Jako pozadí jsem místo barvy nastavil obrázek red.png. Dále jsem nastavil opakování pouze po ose y. V praxi to znamená, že v případě že dojde k „překročení“ velikosti obrázku se vykreslí další pod ním. Na konec jsem nastavil font na Tahoma. Hlavičce jsem nastavil barvu #2a2323. Jedná se o tmavě hnědou barvu, se kterou jsem pracoval dále. V posledním kroku jsem nastavil bílou barvu textu v rámci hlavičky. Divům *poctext* a *pocasivice* jsem nastavil bílou barvu textu. Divu *jazyk2* jsem nastavil hnědou barvu pozadí (jako v případě hlavičky), bílou barvu textu a zaoblení levého spodního rohu o 5 pixelů. Také jsem nastavil bílou barvu textu pro všechny elementy button v rámci divu *jazyk2*. U patičky jsem nastavil hnědou barvu pozadí a bílou

barvu textu. V případě divů s třídou *jedna* jsem nastavil černou barvu pozadí, bílou barvu textu a zrušil jsem zaoblení divu (border-radius: 0px). V poslední kroku jsem nastavil průhlednost (opacity) na 0.85 (85%). U třídy dva jsem nastavil bílé pozadí, černou barvu textu a zrušil zaoblení divu. Posléze jsem nastavil průhled opět na 85%.

Další css soubory, tedy *modramodern.css, oranzovamodern.css* a *zelenamodern.css* jsem změnil pouze pozadí, jinak zůstal kód css stejný. V případě souboru *modramodern.css* jsem nastavil, jako obrázek pozadí blue.png. V souboru *oranzovamodern.css* jsem nastavil obrázek pozadí orange.jpg. U posledního souboru *zelenamodern.css* jsem nastavil pozadí green.jpg. Formát obrázku se liší, jelikož jsou z různých zdrojů a nejedná se o jeden stejný obrázek v různých barvách.

V tento moment jsem vytvořil kompletně veškeré schémata a před tvorbou samotné aplikace jsem si vytvořil JSON soubor.

4.9. Tvorba JSON souboru

JSON soubor jsem využil, jelikož nebyla k dispozici serverová část aplikace. Jak jsem již zmiňoval výše tak v době tvorby této aplikace jsem neměl k dispozici serverou část aplikace. Ovšem ničemu to nevadí, jelikož jsem si vytvořil lokální JSON soubor. Po vytvoření serverové části aplikace stačí pozměnit cestu k JSON souboru a vše bude plně dynamické. Při tvorbě JSON souboru jsem si nejdříve sepsal, co vše má zde být. Vytvořil jsem si následující seznam:

- název místnosti
- počet vyučujících v dané místnosti
- jméno
- příjmení
- tituly před jménem
- tituly za jménem
- den konzultačních hodin (2x, případně podle potřeby vícekrát)
- čas konzultačních hodin (podle dnů)
- telefon
- email
- fotografii
- stav

Po sepsání seznamu požadavků jsem si vytvořil strukturu souboru. Využil jsem vícerozměrného pole a vytvořil následující strukturu. Nejdříve jsem vytvořil asociativní pole učebna, ve kterém se vyhledává přes znaky (název konkrétní učebny). Každá konkrétní učebna je pole. V poli konkrétní učebny jsem dal na první pozici v poli (index 0) prvek *velikost*. Tento prvek uchovává počet vyučujících v dané místnosti. Jako další prvek jsem vytvořil *příjmení* na druhé pozici v poli (index 1). Následoval prvek *jméno* na třetí pozici v poli (index 2). Jako další prvek jsem vytvořil prvek *titul*, který uchovává tituly (nebo více titulů) před jménem a je na čtvrté pozici v poli (index 3). Dále jsem vytvořil prvek *titul2*, který uchovává titul za jménem (Ph.D. a podobně, jeden nebo více titulů) a je na páté pozici v poli (index 4). Následoval prvek *den*, který uchovává hodnotu 0-7 (8 možných hodnot), které reprezentují den v týdnu. Tento prvek je na šesté pozici v poli a má index 5. Dále jsem vytvořil prvek *den2*, který opět reprezentuje den v týdnu (8 možných hodnot). Tento prvek je na sedmé pozici v poli a má index 6. To proč jsem vytvořil 8 prvků, když je 7 dní v týdnu vysvětlím později. Dále jsem vytvořil prvek *konzultacky1*, který uchovává časové rozmezí konzultačních hodin (den uchovaný v prvku den). Tento prvek je na osmé pozici v poli a má index 7. Jelikož jsou zde dva dny konzultačních hodin tak jsem vytvořil prvek *konzultacky2*, který opět uchovává časové rozmezí konzultačních hodin (den uchovaný v prvku den2). Prvek je na deváté pozici v poli a má index 8. Zbývajícími prvky jsou *telefon*, *email*, *fotografie* a *stav*. Prvek *telefon* jsem vytvořil hned po konzultačních hodinách 2 a nachází se na desáté pozici v poli (index 9). Následovně jsem vytvořil *email* na jedenácté pozici v poli (index 10). Předposlední prvek jsem vytvořil *foto*, který uchovává název fotografie. Prvek foto se nachází na dvanácté pozici v poli s indexem 11. Jako poslední jsem vytvořil prvek *stav*, který uchovává jednu z možných hodnot (true = přítomen, false = nepřítomen, busy = zaneprázdněn). Tento prvek se nachází na třináctém pozici v poli s indexem 12.

Po vytvoření struktury JSON souboru jsem jej naplnil daty (pro místnost E405). Soubor vypadá následovně.


```
{
   "ucebna": {
      "E405": [
         {"velikost":"5"},
         {"prijmeni":"Gojda", "jmeno":"Ondřej", "titul" : "Ing.", "titul2": "
","den":"1","den2":"3", "konzultacky1" : "12:30 - 13:30","konzultacky2" : "14:00 - 14:30",
"telefon" : "+420 22438 2423", "email":"gojda@pef.czu.cz","foto":"gojda",
"stav":"false"},
```

{"prijmeni":"Hanzlík", "jmeno":"Petr", "titul" : "Ing.", "titul2": "","den":"3","den2":"4", "konzultacky1" : "11:00 - 12:00","konzultacky2" : "09:30 - 11:00", "telefon" : "+420 733 510 143", "email":"hanzlikp@pef.czu.cz","foto":"hanzlik", "stav":"true"},
{"prijmeni":"Kedaj", "jmeno":"Petr", "titul" : "Ing.", "titul2": "","den":"3","den2":"4", "konzultacky1" : "14:00 - 15:00","konzultacky2" : "10:00 - 11:00", "telefon" : " ", "email":"kedaj@pef.czu.cz","foto":"kedaj","stav":"busy"},
{"prijmeni":"Pavlíček", "jmeno":"Josef", "titul" : "Ing.", "titul2": "Ph.D.","den":"2","den2":"3", "konzultacky1" : "10:00 - 12:00","konzultacky2" : "10:00 - 12:00", "telefon" : "+420 22438 3244", "email":"pavlicek@pef.czu.cz","foto":"pavlicek", "stav":"true"},
{"prijmeni":"Konopásek", "jmeno":"Jakub", "titul" : "Ing.", "titul2": "","den":"2","den2":"7", "konzultacky1" : "15:30 - 19:00","konzultacky2" : "", "telefon" : "", "email":"jakub@konopasek.net","foto":"konopasek", "stav":"busy"}
]
}
}

Veškeré informace jsem získal z portálu www.czu.cz a ze štítku před místností. Po vytvoření podkladů (JSON souboru) jsem postoupil k programování samotné funkcionality.

4.10. Změna loga zeměkoule

Tento soubor jsem vytvořil, jelikož obrázek zeměkoule je rozdílný u schémat. Rozdíl je takový, že v případě hnědého schéma je potřeba ve světle žluté barvě. V případě ostatních schémat je potřeba zeměkoule v bílé barvě. V tomto souboru jsem vytvořil funkci barva, která má jeden parametr barv. Tento parametr uchovává hodnotu zvoleného css souboru s tím, že pokud je zvolený css soubor hnedasimple.css tak je zobrazena zeměkoule světle žluté barvy. Dál jsem zde vytvořil proměnnou divbarva, která má výchozí hodnotu hnědá (bez diakritiky). Tato proměnná slouží k nastavení formátování o kterém budu psát dále. Pokud je soubor jiný, je nastavena bílá zeměkoule. Nastavení například u červeného simple schématu je else if(barv==“css/schema/cervenasimple.css“){divbarva = „cervena“; document.getElementById(„zeme“}.src = „obrazky/earthwhite.png“.

Funkci jsem vytvořil tak, že nejdříve se nastaví parametr barva (css soubor). Nejdříve dojde k porovnání, jestli se jedná o soubor *hnedasimple.css*. Pokud ano tak dojde k nastavení světle žluté zeměkoule. Pokud ne dojde k nastavení bílé zeměkoule. Po porovnání dojde k nastavení css souboru pro aplikaci. Této funkci předchází nastavení u elementu pro připojení externího css souboru id *pagestyle* a výchozího css souboru (v tomto případě *hnedasimple.css*). Posléze jsem aplikoval tuto funkci na element button.

Aplikoval jsem si tak, že jsem nastavil akci onclick (na kliknutí) barva s parametrem css souboru. Tedy například barva („css/schema/cervenasimple.*css*“). Takto jsem vytvořil elementy button pro všechny barevné schémata, který jsem vytvořil dříve. Na závěr jsem připojil JavaScriptový souboru přes příkaz <script> do aplikace. Pokud to shrnu tak tato funkce funguje tak, že po kliknutí na příslušný tlačítko se zavolá funkce barva s parametrem příslušného schématu (souboru css), porovná se jaká barva zeměkoule se má nastavit a posléze dojde k nastavení příslušného css souboru pro aplikaci.

Po nastavení barevných schémat a zeměkoule jsem se přesunul na funkci čas.

4.11. Získání a nastavení času

Jelikož jsem již měl téměř kompletní hlavičku tak jsem po zvolení schématu vytvořil funkci získávající čas. Vytvořil jsem funkci cas (bez parametru), která je v JavaScriptovém souboru *cas.js*. V této funkci jsem vytvořil pět proměnných. Proměnné jsou *today, h, m , s*. Jako první jsem vytvořil proměnnou *today*. V této proměnné je objekt date (30). Tento objekt umožňuje pracovat se dny, měsíci, roky, ale i s hodinami, minutami a vteřinami. Jsou zde dvě možnosti a to buď pracovat s řetězcem (string), nebo s číslem (number). V případě čísel je vracena hodnota v milisekundách od 1.1.1970 00:00. Rozhodl jsem se pro práci s řetězcem a následnou aplikací metod díky kterým jsem získal hodiny, minuty a vteřiny. Rozhodl jsem se pro řetězec, jelikož je jednodušší z tohoto formátu získat aktuální čas. Řetězec vypadá následovně: Wed Mar 02 2016 16:20:10 GMT+0100 (Střední Evropa (běžný čas)). V proměnné *h* jsem využil metody getHours()(31), která získává z proměnné *today* hodiny. Dále jsem do proměnné *m* nastavil minuty přes metodu getMinutes()(32). Pro proměnnou *s* jsem nastavil metody getSeconds()(33). V tento moment jsem tedy měl hodiny, minuty a vteřiny s tím, že se čas získal pouze v jeden okamžik a neaktualizoval se. Proto jsem vytvořil pomocnou funkci checkTime, která kontroluje jestli je číslo menší jak 10. V případě, že je číslo menší jak 10 přidá před číslo 0. Toto opatření jsem provedl proto, aby byl čas zobrazen stejně, jako v případě operačního systému, nebo například digitálních hodinek. Bez této funkce by čísla menší jak 10 byly zobrazovány například takto: 7:7:7 (7 hodin, 7 minut, 7 vteřin). Kód této pomocné funkce je velmi jednoduchý a vypadá následovně. Function checkTime (i) { if(i<10){i=“0“+i}; return i;}.

Tato pomocná funkce je uvnitř funkce *cas* hned pod setTimeout(39) příkazem. Po nastavení proměnných jsem nastavil dodatečnou funkci checkTime pro danou proměnnou.

Vypadá to tedy následovně: m = checkTime(m). Tímto jsem ošetřil proměnné *h*, *m* a *s*. Po ošetření příslušných proměnných jsem nastavil vložení do divu s id *cas* ve formátu h : m. Rozhodl jsem se neuvádět vteřiny, ale v případě potřeby stačí přidat proměnnou *s*. Na závěr jsem nastavil, aby byla funkce cas volána každou půl vteřinu, tedy 500milisekund. Využil jsem k tomu příkaz setTimout s parametry *cas* a 500. Ve výsledku tedy vypadal příkaz následovně: setTimout(cas,500).

V tento okamžik mám téměř kompletní hlavičku aplikace a dále jsem vytvořil soubor *vyucujici.js*.

4.12. Získání a generování vyučujících

Po nastavení hlavičky (krom názvu místnost) jsem se pustil do získání vyučujících. Data lze získat buď přes jQuery, nebo pomocí AJAXu. Rozhodl jsem se pro jQuery, jelikož mi díky tomu odpadly problémy s nastavováním parametrů u AJAX příkazu. Jelikož jQuery je rozšiřující knihovna pro JavaScript tak jsem nejdřív musel napojit jQuery knihovnu. Udělal jsem tak přes internetové rozhraní[13]. Po napojení jsem vytvořil jsem funkci lide() (bez parametru). V této funkci jsem vytvořil proměnnou *mistnost*, ve které jsem nastavil příslušnou místnost (v tomto případe E405). Hned po vytvoření proměnné jsem hodnotu této proměnné nastavil, jako obsah divu s id *mistnost*. K přidělení hodnoty jsem využil jQuery příkazu $(„mistnost“).html(mistnost). Ekvivalentem k této funkci je document.getElementById(„mistnost“).innerHTML = mistnost. Je jedno co z toho zde využiju a daný moment jsem využil jQuery příkazu. V tomto okamžiku jsem již měl kompletní hlavičku včetně názvu místnosti.

V rámci funkce lide() jsem využil příkazu $.getJSON. Tento příkaz jsem využil k získání (načtení) dat z JSON souboru. Konkrétně jsem nastavil parametry *lide.json* (JSON soubor co jsem vytvořil výše) a function(data). Příkaz tedy vypadá následovně: $.getJSON(„lide.json“, function(data)). Function(data) jsem nastavil proto, aby se nahrál obsah JSON souboru do pole (vícerozměrného v tomto případě). Po sepsání příkazu jsem přidal alert(data.ucebna[mistnost][1].jmeno) abych otestoval, jestli se povedlo načtení. Nejdříve jsem si vyhledal v JSON souboru, co by mi měl alert vypsat a posléze spustil funkci. Podle souboru JSON by mi měl alert vypsat „Ondřej“ a to se po spuštění aplikace stalo.

[13] http://code.jquery.com/jquery-2.1.0.min.js

Jelikož mi získání dat fungovalo bez problému tak jsem následně řešil dny. Jelikož v JSON souboru jsou pouze čísla dne o který se jedná tak jsem vytvořil pole. Vytvořil jsem jednorozměrné pole obsahující dny od pondělí do neděle a přidal jednu prázdnou položku (uvozovky bez mezery). Poslední položka je z důvodu, když není vyplněn den, nebo má vyučující pouze v jeden den konzultační hodiny. V případě, že bych nezadal položku navíc a v JSON souboru by byla prázdná hodnota vypsalo by se undefined. S touto „prázdnou“ položkou dojde k zobrazení volného místa a nebude problém s tím, že by došlo k vypsání něčeho nežádaného. Toto pole jsem vytvořil mimo funkci lidé, aby nedocházelo k jeho opětovnému vytváření při každém spuštění funkce, ale pouze při spuštění aplikace.

V tomto momentu jsem měl vytvořené pole se *dny* a funkci na získávání dat z JSON souboru. Sepsal jsem si, co vše potřebuji získat z JSON souboru. Vznikl mi následující seznam údajů, které potřebuji získat.

- jméno
- příjmení
- titul
- titul2
- konzultace
- konzultace2
- email
- telefon
- fotografie
- stav
- den
- den2

Následně jsem vytvořil proměnné *jmeno*, *prijmeni*, *titul*, *titul2*, *konzultace*, *konzultace2*, *email*, *telefon*, *fotografie*, *stav*, *den* a *den2*. Jelikož jsem chtěl mít jistotu, že vše funguje jak má a získávám správná data tak jsem nejdříve nastavil hodnoty fixně pro prvního vyučujícího. Nastavil jsem tedy následující hodnoty proměnným. Proměnné *jmeno* jsem nastavil hodnotu data.ucebna[mistnost][1].jmeno. U proměnné *prijmeni* jsem nastavil hodnotu data.ucebna[mistnost][1].prijmeni. Následující proměnné *titul* jsem nastavil hodnotu data.ucebna[mistnost][1].titul. U proměnné *titul2* jsem nastavil hodnotu podobně a

to data.ucebna[mistnost][1].titul2. Pro proměnnou *konzultace* jsem nastavil hodnotu data.ucebna[mistnost][1].konzultacky1. Pro proměnnou *konzultace2* jsem nastavil hodnotu data.ucebna[mistnost][1].konzultacky2. Následuje proměnná *email,* kde jsem nastavil hodnotu data.ucebna[mistnost][1].email. Poté jsem nastavil hodnotu proměnné *telefon* přes data.ucebna[mistnost][1].telefon. Následovala proměnná *fotografie,* které jsem nastavil hodnotu přes data.ucebna[mistnost][1].foto. Dále jsem nastavil proměnnou *stav* přes data.ucebna[mistnost][1].stav. Jako předposlední jsem nastavil proměnnou *den* přes data.ucebna[mistnost][1].den. Poslední proměnnou, kterou jsem nastavil je proměnná *den2* a to přes data.ucebna[mistnost][1].den2. U všech proměnných mám teď nastavené získávání z JSON souboru, respektive z pole ve kterém je obsah JSON souboru nahraný.

Proměnná místnost je stejná proměnná přes kterou nastavuji název místnosti a je deklarována uvnitř funkce. Proměnné mají tedy hodnotu data(označení pole).ucebna[„E405“](výběr konkrétní položky v poli)[1](pozice vyučujícího).jmeno a podobně (konkrétní položka příslušící k vyučujícímu). Abych otestoval, že jsem získal správné položky tak jsem vložil alert. Alert obsahoval měl následující obsah. Alert(*jmeno* +

„,“+*prijmeni*+„,“+*titul*+„,“+*titul2*+„,“+*konzultace*+„,“+*konzultace2*+„,“+*email*+„,“+*telefon* +„,“+*fotografie*+„,“+*stav*+„,“+*den*+„,“+*den2*). Očekávaný výstup byl: Ondřej, Gojda, Ing.,(mezera),12:30 - 13:30,14:00 – 14:30,gojda@pef.czu.cz,+420 22438 2423,gojda,false,1,3. Po spuštění funkce a provedení alertu byl výstup podle očekávání a tak jsem se přesunul dále.

V tomto momentu jsem řešil problém, jak vypsat veškerý počet vyučujících. Využil jsem proto jednoduchého for cyklu. Nejdříve jsem si vytvořil cyklus s pevně daným počtem vyučujících. Ručně jsem nastavil v tomto cyklu maximální hodnotu na 5. Nastavení cyklu bylo tedy následující: i=1, i<=5,i++. Cyklus tedy začínal s hodnotou jednou a měl 5 opakování. Abych měl jistotu, že dojde k vypsání hodnot od 1 do 5 tak jsem nastavil alert(i). V tomto případě mi byly postupně vypsány hodnoty od 1 do 5 a mohl jsem se posunout dále. Schválně jsem nenastavoval hodnotu i na 0, jelikož by v případě dat z JSON souboru proběhlo 6 opakování.

Nyní mi schází pouze získat hodnotu o počtu vyučujících v místnosti. K tomu jsem vytvořil proměnnou *pocetvyucujicich*. Tato proměnná získává hodnotu přes data.ucebna[mistnost][0].velikost. Nula je zde schválně, jelikož velikost je vždy na prvním

místě v poli. Tato proměnná je v rámci funkce lide(). V tento moment má proměnná *pocetvyucujicich* hodnotu 5.

Po nastavení proměnné vyučující jsem nahradil v for cyklu hodnotu 5 proměnnou *pocetvyucujicich*. Nastavení cyklu tedy bylo následující: i=1,i<=pocetvyucujicich, i++. Dále jsem nahradil u proměnných druhou dimenzi pole proměnnou *i*. To mi zajistilo, že při každém opakování cyklu byla dosazena hodnota (podle čísla cyklu). Při vložení alertu a vypsání hodnot proměnných (viz výše) jsem dostal 5 rozdílných vyučujících. Vypsané údaje se shodují s údaji v JSON souboru. V tomto kroku jsem měl nastavené hodnoty proměnných s získáním všech vyučujících k příslušené místnosti. Dále jsem se rozhodl pro vytvoření zobrazení údajů v rozumné formě.

Jako požadovaný cíl jsem si v tento moment stanovil vytvoření dvou divů. První div bude zobrazovat titul (před jménem), jméno, příjmení, titul (za jménem), stav. Druhý div bude zobrazovat konzultační hodiny (popisek), první konzultační hodiny (den, čas), druhé konzultační hodiny (den, čas), fotografii. Dále popisky kontaktních údajů (telefon, email) a konkrétní údaje.

Vytvořil jsem proměnnou *barva*, která je bez výchozí hodnoty. Tato proměnná slouží k přiřazení správného obrázku podle stavu získaného z JSON souboru. Tedy v případě false je její hodnota *red*, v případě true je její hodnota *green*. Pokud stav nenabývá hodnoty *true* nebo *false* tak je nastavena proměnné *barva* hodnota *orange*. Když jsem měl zajištěné nastavení příslušného obrázku podle stavu tak jsem se vrhnul na tvorbu samotných divů. Vytvořil jsem si dvě proměnné *div1* a *div2*. Hodnota proměnné *div1* je document.createElement(„div“). Hodnota proměnné *div2* je také document.createElement(„div“). Tato hodna umožňuje jednodušší následnou práci s divy, které se teprve vytvoří. Již dříve jsem nastavoval formátování třidě *jedna* a třídě *dva*. Zde tuto třídu využiju a nastavím *divu1* classname. Provedu to příkazem div1.className = „jedna“. Obdobně to nastavím u *divu2*, přes příkaz div2.className = „dva“. Tyto třidy jsem nastavil proto, aby veškeré vygenerované divy měly stejné formátování a grafiku. V poslední řade jsem nastavil obsah (proměnné) prvnímu divu. Využil jsem příkazu div1.innerHTML. Příkaz vypadá následovně: div1.innerHTML = *titul* + " " + *jmeno* + " " + *prijmeni* + " " + *titul2* + " <img id='stav"+i+"' src='obrazky/"+*barva*+".png'> ". U zobrazení stavu jsem využil divu kvůli tomu, aby s obrázkem stavu byla jednoduší manipulace. U obrázku (img element) jsem nastavil id

stav + *i* (hodnota 1-5) proto, aby v případě změny jednoho ze stavů vyučujících došlo k změně stavu (příslušného obrázku) pouze u daného vyučujícího a nikoliv u všech. Více o tomto se zmíním později u kontroly stavu.

V této fázi nastavím pouze div2.innerHTML (nebudu se teď zaobírat viditelností). Zde jsem nastavoval více proměnných, jelikož je zde více informací. Příkaz vypadá následovně. div2.innerHTML = "<img width='50px' height='50px' src='obrazky/fotky/"+ *fotografie* +".jpg'> Konzultační hodiny"+"
"+ *dny*[*den*] + " " + *konzultace* + "
"+ *dny*[*den2*] +" "+*konzultace2* + "
 Kontakty
" + *email* + "
 " + *telefon*>. Jak jsem již zmiňoval dříve tak využívám pole *dny* k získávání názvu dne (pondělí a podobně). Na rozdíl od *divu1* jsem zde nenastavoval třídu obrázku, jelikož zde nedochází ke změnám fotek vyučujících (proč tomu tak je zmíním později). Po innerHTML příkazu jsem vložil příkaz appendChild. Tento příkaz slouží k vytvoření a vložení výsledného divu (včetně obsahu) tam kam si zvolím.

Jelikož mám již připraven div s id *vyucujici* tak vypadá příkaz následovně. document.getElementById("vyucujici").appendChild(div2). Zde se liší pouze div a to u prvního divu je *div1* u druhého jsem dal *div2*. Jelikož by to nevypadalo esteticky (dvojce divů nalepených na sebe) tak jsem vložil pod druhý div br element. Tento element jsem využil k tomu, aby bylo odřádkování po vložení druhého divu.

Postupoval jsem stejně, jako v případě divů a tedy vytvořil proměnnou br. Této proměnné jsem přiřadil hodnotu document.createElement(„br“). Následně jsem nastavil třídu *oddelovac*. Tato třída nemá žádné formátování ani grafické nastavení. Je zde pouze pro případné odstraňování (viz dále). Na závěr jsem dal příkaz appendChild do divu *vyucujici*.

Teď mám již nastavené vše. Po spuštění funkce lide() dojde k vytvoření divů pro všech pět vyučujících s příslušnými údaji a fotografiemi. Jelikož chci, aby se údaje o vyučujícím zobrazovali až po kliknutí na první div tak jsem se vrátil zpět do kódu. Konkrétně do funkce *lide* do části, kde dochází k tvorbě divů. Nastavil jsem u *divu2* neviditelnost. Tedy příkaz div2.style.display = „none“. Tím jsem vyřešil problém se zobrazováním druhého divu. V tomto okamžiku nastal problém v tom, že u každého vyučujícího se mi vytvořily divy pětkrát. Tedy, že jsem měl 5x prvního vyučujícího (stejné údaje) a takhle to bylo u všech pěti vyučujících.

Tento problém jsem vyřešil tak, že jsem vložil další cyklus před tvorbu divů. Tedy po deklarování proměnných s tím, že proměnná *barva* se vytváří až v tomto novém divu. Chtěl jsem, aby tvorba divů proběhla pouze jednou za jeden cyklus prvního cyklu. Tedy, aby pro prvního vyučujícího byl vytvořen pouze jedna dvojce divů. Proto jsem nastavil druhý cyklus následovně: f=1, f<=1, f++. Tímto mám zaručeno, že cyklus proběhne pouze jednou a dojde tedy k jednomu vytvoření divů na jeden cyklus prvního cyklu.

Teď mám již vše nastaveno. Následoval problém s tím, že při kliknutí na první div se nic nestalo. Řešil jsem tedy problém s tím, jak udělat, aby při kliku na konkrétní div došlo k zobrazení pouze jednoho divu a ne všech.

Vytvořil jsem si tedy funkci change s parametry *d1, d2* (*div1,div2*). Tato funkce mi slouží k zjištění, jestli je *div2* viditelný, nebo neviditelný a následné akci. Nejdříve zjišťuji, jestli je *div2* viditelný. V případě, že je viditelný dojde k zaoblení dolních rohů (levý a pravý) o 5 pixelů. Po nastavení zaoblení rohů nastává problém, jelikož v případě kliknutí na první div a zviditelnění *divu2* (zobrazení) dojde k zrušení ohraničení na spodní straně *divu1* (border-bottom). Kvůli tomuto problému jsem vytvořil ověření o jaké barevné schéma se jedná. V případě schémat simple dochází k nastavení příslušné barvy (viz dále). V případě schémat modern nedochází k nastavení ohraničení. Porovnání jsem provedl přes větvení if. Porovnávám proměnnou *divbarva* z funkce barva(*barv*). Ve funkci *barva* dochází k nastavení této proměnné (pouze u simple schémat) podle příslušné barvy (například u zvolení souboru *cervenasimple.css* je nastavena hodnota proměnné *divbarva* na „cervena“). Pokud se hodnota proměnné *divbarva* rovná hodnotě „cervena“ je nastaveno spodní ohraničení o šířce 1 pixelu rovné čáry (solid) v příslušné barvě. Takto jsem nastavil porovnávání pro všechny dostupné stavy proměnné divbarva. V případě, že je zde jiná hodnota (hodnota nic) tak dojde k nastavení nulové šířky spodního ohraničení. Je to proto, že schémata modern nepoužívají ohraničení. Po nastavení příslušného ohraničení dojde k zneviditelnění *divu2*. V případě, že není zobrazen *div2* (style.display = „none“) tak dojde ke změně hodnoty none na block. Dále dojde k zrušení zaoblení ohraničení *divu1* (spodní levý a pravý roh) a zrušení spodního ohraničení *divu1*.

V tomto momentě mám již funkci vytvořenou jen ji musím aplikovat. Vložil jsem tedy po vytvoření druhého divu s třídou *dva* zavolání funkce. Zavolal jsem funkci change s parametry *d1* a *d2*. Tuto funkci jsem volal bez jména (function (*d1,d2*)) jelikož ji potřebuji zavolat pouze jednou. Uvnitř této funkce jsem nastavil, že v případě kliknutí na

div1 dojde k zavolání funkce change(*d1*,*d2*). Pro spuštění funkce jsem za deklaraci funkce (za kód celé anonymní funkce) vložil závorku s parametry *div1*, *div2*. Díky těmto závorkám dojde k zavolání anonymní funkce s parametry *div1* a *div2*. Takto jsem zajistil to, že při kliknutí na *div1* dojde k zobrazení *divu2* a při kliknutí na *div1*, nebo *div2* dojde k schování *divu2*.

Na závěr jsem nastavil funkci lide spouštění jednou za 24 hodin. K tomu jsem využil metodu setInterval(39). U tohoto příkazu jsem nastavil funkci (*lide*) a interval v milisekundách (86400000). Teď již mám funkční získávání a generování lidí a potřebuji vyřešit aktualizaci stavu vyučujících.

4.13. Kontrola a aktualizace stavu

Jelikož ke změně přítomnosti, nepřítomnosti či zaneprázdněnosti dochází v rámci vteřin tak jsem vytvořil funkci na kontrolu aktuálního statusu. Abych se dostal k aktuálnímu stavu tak jsem musel opět načíst data z JSON souboru. Využil jsem k tomu opět jQuery příkaz $.getJSON. Prvním parametrem je cesta k souboru, v tomto případě pouze název souboru, jelikož mám JSON soubor umístěn ve stejném adresáři, jako soubory aplikace. Druhý parametr nahrává data z JSON souboru do pole s názvem *data*. Jedná se o vícerozměrné pole. Nejdříve jsem si vytáhl konkrétní stav vyučujícího a otestovat, jestli funguje správně. Nejdříve jsem vytvořil lokální proměnnou *status*. Této proměnné jsem nastavil získávání hodnoty z pole. Jelikož testuji nejdříve prvního vyučujícího tak jsem nastavil získání dat z pole následovně: data.ucebna[mistnost][1].stav. V tomto momentě by měla mít proměnná status hodnotu *false*. Nastavil jsem alert(*status*), abych ověřil nastavenou hodnotu. V této funkci využívám proměnných *mistnost* a *pocetvyucujicih*. Tato funkce na kontrolu stavu se nachází v jiném JavaScriptovém souboru a proto jsem nastavil proměnné *mistnost* a *pocetvyucujicih* na globální proměnné. Jelikož mi hodnota proměnné *status* seděla se stavem v JSON souboru tak jsem pokročil dále. Vložil jsem for cyklus do funkce na kontrolu stavu. Podmínka cyklu je stejná, jako v případě generování vyučujících (divů), tedy i=1, *i*<=*pocetvyucujicich*,*i*++. Upravil jsem proměnnou *status* z data.ucebna[mistnost][1].stav na data.ucebna[mistnost][*i*].stav.

Tímto jsem zajistil, že dojde k načtení všech stavů vyučujících příslušné místnosti. Před samotným cyklem jsem vytvořil proměnnou *dostupnost*. Tato proměnná je bez výchozí hodnoty. Vytvořil jsem ji proto, abych mohl na základě získaného stavu nastavit správný název obrázku a posléze jej změnit. Uvnitř for cyklu jsem vytvořil rozhodovací

konstrukci. Tato konstrukce porovnává hodnotu proměnné *status*. Jsou zde tři rozhodovací větve a to protože proměnná *status* nabývá tří hodnot. Hodnoty jsou *true, false, busy*. V případě, že hodnota proměnné *status* je *true* dojde k nastavení hodnot proměnné dostupnost na *green*. Po nastavení hodnot proměnné *dostupnost* dojde ke změně obrázku na green.png což je zelený indikátor (přítomen). V případě, že proměnná *status* bude mít hodnotu *false* dojde k nastavení proměnné *dostupnost* na *red*. Posléze dojde k nastavení obrázku red.png tedy k červenému indikátoru (nepřítomen). Pokud hodnota proměnné *status* nenabývá hodnoty *true*, nebo *false* dojde k nastavení proměnné *dostupnost* na hodnotu *orange*. Posléze dojde k nastavení obrázku orange.png neboli oranžový indikátor (zaneprázdněn).

V tomto kroku mám nastavenou kontrolu a změnu obrázku ovšem pro všechny indikátory naráz. Upravil jsem tedy změnu indikátoru následovně: document.getElementById(„stav“+i).src = „obrazky/“+*dostupnost*+“.png“. Id stav+i zaručuje, že je vždy kontrolován příslušný prvek (příslušný vyučující). Jedná se tedy o nastavování obrázku vždy s tím rozdílem, že pokud je stav stejný nedochází k vykreslování nového obrázku a tudíž je zátěž na hardware minimální. Na závěr jsem nastavil metodu setInterval na 30 000 milisekund. Dojde ke spuštění příslušné funkce (kontrola stavu) každých 30 vteřin. 30 vteřin jsem zvolil, jelikož než dojde k vypnutí počítače a podobně tak lze zastihnout vyučujícího a kontrolovat stav každých například 5 vteřin je zbytečná zátěž pro hardware. V tomto kroku mám již vyplněnou hlavičku, nastavené vyučující včetně kontroly stavů. Další na seznamu je funkce na vytvoření počasí.

4.14. Získání, prezentace a aktualizace počasí

Údaje o aktuálním počasí lze získat ze spousty zdrojů. Když jsem přemýšlel, jaký zdroj využít tak jsem se především díval na internetu po kompletní sadě údajů. Kompletní sadou mám na mysli aktuální stav (základní údaje), rozšířené údaje (vlhkost a podobně) a příslušný obrázek. Jediným kritériem pro mne bylo, aby zde byla verze zdarma. Verze, které jsou zdarma se především liší v počtu zavolání (zavolání API a následné získání dat) zdarma. Hledal jsem API pro získávání počasí se zasíláním dat v JSON souboru. Vybral jsem „The Dark Sky Forecast“[14].Toto API má zdarma 1 000 zavolání denně, ale především

[14] https://developer.forecast.io/

pracuje s tzv. Skycons. Skycons jsou animované obrázky aktuálního počasí a jsou zdarma dostupné.

Po vybrání příslušného API jsem se pustil do práce. Nejdříve jsem vytvořil soubor *pocasi.js*. V tomto souboru jsem vytvořil veškeré funkci související s počasím. Nejdříve jsem vytvořil funkci *ziskanipocasi*. Prvně jsem použil jQuery příkaz $.getJSON. Prvním parametrem je odkaz na soubor. Odkaz se skládá z části odkazující na příslušnou adresu daného API[15]. Další částí je, co chci od daného API (v tomto případě forecast). Předposlední částí je API key. Tento klíč slouží k identifikace uživatele (kvůli platbám především) a lze jej získat po vytvoření účtu. V mém případě je API klíč následující *6b6aee281a2645c5214dbffa4ab120df*. Poslední částí odkazu (požadavku) je poloha. Polohu jsem získával dvěma způsoby nejdříve přes vyhledání polohy (zeměpisné šířky a délky) podle map a posléze podle polohy zařízení. Pracoval jsem s polohou univerzity a zeměpisnou šířku a délku získal z Google maps. Celý jQuery příkaz vypadá následovně: $.getJSON('https://api.forecast.io/forecast/6b6aee281a2645c5214dbffa4ab120df/50.1297, 14.373“,function(data)). Tímto jsem získal JSON soubor pro danou polohu.

Problém nastává s hierarchii daného JSON souboru. Hierarchie je následující s tím, že popíši každý prvek:

- latitude – zeměpisná šířka, zobrazuje zadanou hodnotu
- longitude – zeměpisná délka, zobrazuje zadanou hodnotu
- timezone – ukazuje lokaci získanou na základě zeměpisné šířky a délky (Europe/Prague)
- currently – Obsahuje aktuální data. Obsahuje 15 údajů s tím, že jsem vybral pouze takové, které běžným uživatelům něco říkají. Vybral jsem tedy *summary*, *icon*, *precipProbability*, *temperature*, *apparentTemperature*, *humidity*, *windSpeed* a *pressure*. K tomu co který údaj znamená se dostanu později.
- hourly – Obsahuje hodinové předpovědi. Tyto data jsem nevyužil.
- daily – Obsahuje předpověď na 7 dní. Odtud jsem získal *sunriseTime* a *sunsetTime*.
- Flags – Obsahuje dodatečná nastavení, jako například výběr meteorologické stranice a podobně.

[15] https://api.forecast.io

Předtím než vysvětlím konkrétní data získaná z JSON souboru (API) tak jsem otestoval příkaz (API url) v prohlížeči, abych věděl, že jsem vše vyplnil správně. Vytvořil jsem si 10 proměnných, které jsou následující:

- *vitr* – Obsahuje rychlost větru v míli za hodinu.
- *vlhkost* – vlhkost se zde udává v procentech, tedy v hodnotě menší jak 1
- *ikona* – Obsahuje řetězec (string), který má název ikony Skycons. Bohužel není text stejně formátovaný, jako v případě Skycons.
- *vychod* – Obsahuje čas východu slunce. Čas je ve formátu unix.
- *zapad* – Obsahuje čas západu slunce. Čase je ve formátu unix.
- *pocitovateplota* – Obsahuje pocitovou teplotu v jednotkách Fahrenheit.
- *teplota* – Obsahuje aktuální teplotu v jednotkách Fahrenheit.
- *pocasi* – Obsahuje slovní shrnutí aktuálního počasí.
- *sance* – Obsahuje pravděpodobnost deště.
- *tlak* – Obsahuje aktuální tlak.

Příkaz $.getJSON mi nahrál obsah JSON souboru do pole *data*. Nejdříve nastavím hodnoty proměnným a posléze se budu zabývat jejich převedením na jednotky používané v České republice. Začal jsem proměnnou *vitr*. Data jsem získal následovně: data.currently.windSpeed. Následovala proměnná *vlhkost*. Této proměnné jsem přiřadil data přes data.currently.humidity. Další proměnnou je *ikona*. Této proměnné jsem přiřadil hodnotu přes data.currently.icon. Následuje proměnná *vychod*. Této proměnné jsem přiřadil hodnotu přes data.daily.data[0].sunriseTime. Data[0] je zde proto, aby se jednalo o konkrétní den kdy je zaslaná žádost. Dále je zde proměnná *zapad*. Této proměnné jsem přiřadil hodnotu přes data.daily.data[0].sunsetTime. Proměnnou *pocitovateplota* jsem nastavil data.currently.apparentTemperature. Následovala proměnná *teplota* u které jsem nastavil získávání dat přes data.currently.temperature. Další je proměnná *pocasi* u které jsem nastavil data.currently.summary. Předposlední proměnnou je proměnná *sance*, které jsem nastavil data.currently.precipProbability. Poslední proměnnou je proměnná *tlak*. Zde jsem nastavil data.currently.pressure.

Po nastavení proměnných jsem si vytvořil prvek button. Tomuto prvku jsem přiřadil akci onlick se zavoláním funkce *ziskanipocasi*(). Abych otestoval, že jsem získal data správně tak jsem vytvořil alert. Syntaxe alertu byla následující: alert(*vitr* + "," +

vlhkost + "," + *ikona* + "," + *vychod* + "," + *zapad* + "," + *pocitovateplota* + "," + *teplota* + "," + *pocasi* + "," + *sance* + "," + *tlak*). Jediný problém, který zde nastal jsou jednotky. Jelikož jsou data v imperiálních jednotkách je nutné je převést do metrických jednotek.

Při převodu jsem bral proměnné prostupně. Začal jsem proměnnou *vitr*. Ta je uváděna v milí za hodinu a potřeboval jsem ji převést na kilometry za hodinu. Využil jsem funkce Math.round(34), která zaokrouhluje čísla na celá čísla. Zaokrouhlil jsem výsledek na jedno desetinné číslo, aby to vypadalo lépe. Není už takový rozdíl mezi 2.888888km/h a 2.9km/h. K tomu, abych převedl míle za hodinu na kilometry za hodinu jsem vynásobil získanou hodnotu 1.609. Poté jsem ji vynásobil 10 a následně vydělil 10. Při vynásobení 10 získám hodnotu v rámci celých čísel například 28.5. Poté dojde k zaokrouhlení na 29 a následně vydělení 10 na 2.9km/h. Převod vypadá následovně: Math.round((data.currently.windSpeed * 1.609)*10)/10.

Následuje proměnná *vlhkost*. Tato proměnná je v základu číslo menší jak 1 a větší, nebo rovnou 0. Abych z toho čísla dostal procenta tak jsem hodnotu proměnné *vlhkost* vynásobil 100. Tímto jsem dostal číslo v procentech, ale s neomezeným počtem míst za čárkou. Proto jsem přidal metodu toFixed(2)(38). Číslo v závorce určuje, kolik bude zobrazeno míst za čárkou. Poslední místo je zaokrouhleno. Úprava na procenta tedy vypadá následovně: (data.currently.humidity*100).toFixed(2).

Další proměnnou, kterou jsem upravil je *vychod*. Hodnota této proměnné je v unixovém čase. To je čas v milisekundách od 1.1.1970 00:00. Vytvořil jsem pět proměnných. Tyto proměnné jsem vytvořil proto, abych mohl se získaným časem pracovat a převést jej na čas, kterému člověk rozumí. Vytvořil jsem proměnné: *date*, *hours*, *minutes*, *seconds* a *vychodslunce*. Proměnné *date* jsem přiřadil hodnotu new Date (*vychod**1000). Tedy vytvořil jsem nový objekt datum s tím, že jsem vložil aktuální datum z unixového času. Vynásobil jsem jej 1 000 abych dostal čas v sekundách. Proměnné *hours* jsem přiřadil hodnotu získanou přes metodu getHours(). Tím jsem získal hodiny (pro využití metod jsem převedl čas z milisekund na sekundy). Následuje proměnná *minutes*. Zde jsem využil metody getMinutes(). Předtím, než jsem využil metody getMinutes tak jsem zde připsal „0“ + date.getMinutes(). Je to proto, aby v momentě kdy je číslo menší, jak 10 tak se přidá nula (např. 07 místo 7). Ovšem zde nastal problém s tím, že nula se přidá i v případě čísla většího, jak 10. U proměnné *seconds* jsem využil metodu getSeconds(). Opět jsem ze připsal „0“. Je to pro případ menšího čísla, jak 10. Udělal jsem to, protože

lépe vypadá čas 00:00:07 než 00:00:7. Po naplnění všech proměnných časem jsem nastavil poslední proměnnou *vychodslunce*. Do této proměnné jsem dal řetězec z proměnných. Podoba je následující: *hours* + „:“ + *minutes*.substr(-2) + „:“+ *seconds*.substr(-2). Metodu substr(36), aby došlo k zobrazení dvou znaků od konce. Tedy v případě, že je číslo menší, jak 10 zobrazí se před číslem 0. V případě, že je číslo větší, jak 10 dojde k zobrazení konkrétního čísla.

Postoupil jsem dále k proměnné *zapad*. Tato proměnná má opět hodnotu v unixovém čase. Proto jsem opět vytvořil pět proměnných pro převedení do času lidem srozumitelnému. Vytvořil jsem proměnné *date2*, *hours2*, *minutes2*, *seconds2* a *zapadslunce*. Využil jsem opět stejných metod, jako v případě východu slunce. Tedy proměnné *date2* jsem nastavil new Date (*zapad**1000). Tím jsem získal čas v sekundách. Následovala proměnná *hours2* u které jsem využil metodu getHours(). Dále jsem nastavil proměnnou *minutes2*. Zde jsem využil metody getMinutes() s tím, že jsem před tuto metodu připojil „0“(hodnotu nula). Předposlední proměnnou je proměnná *seconds2*. Zde jsem využil metody getSeconds(). Opět jsem dal před metodu „0“. Poslední proměnnou je proměnná *zapadslunce*. Nastavil jsem řetězec z proměnných, který vypadá následovně: *hours2* + „:“ + *minutes2*.substr(-2) + „:“ + *seconds2*.substr(-2). Využil jsem metody substr() k zobrazení dvou znaků od konce. V případě hodnoty menší, jak 10 dojde k zobrazení 0 + proměnná. V případě hodnoty větší, jak 10 dojde k zobrazení pouze proměnné.

V tento moment mám získaný západ i východ slunce a tak jsem to otestoval přes alert(*vychodslunce* + „„ + *zapadslunce*), jestli mám dobře nastavené metody. Vše fungovalo bez problému a tak jsem pokračoval dále. Teď zbývá vložit získané údaje do divů a posléze získat a vytvořit obrázek o počasí. Pro text zde mám dva divy s id *poctext* a *pocasivice*. První div (*poctext*) bude obsahovat pouze základní údaje o počasí. Tedy slovní vyjádření (jasno a podobně), aktuální teplotu a obrázek pro rozkliknuti více informací o počasí. Druhý div (*pocasivice*) bude obsahovat dodatečné informace o aktuálním počasí, jako je rychlost větru, pravděpodobnost deště a podobně.

Pro vložení dat do prvního divu jsem využil jQuery příkazu $(„poctext“).html(). Potřebné data jsou uloženy v proměnných *pocasi* a *teplota*. Dále potřebuji vykreslit obrázek šipky dolu a přiřadit onlick funkci. Na kliknutí potřebuji nastavit takovou akci, která z viditelní druhý div a zároveň schová obrázek šipky dolu. Vytvořil jsem řetězec

složený z proměnných a prvku img (obrázek). Řetězec vypadá následovně: *pocasi* + ", " + *teplota* + " °C. Po tomto řetězci jsem připojit odřádkování (br element). Za br element jsem vložil element img. Jelikož jsem chtěl šipku nahoru tak jsem nastavil id *nahoru*. Dále jsem přiřadil zdroj (cestu k obrázku) obrazky/up.png(45). Dále jsem nastavil akci onlick funkci vice() o které se zmíním dále. Obrázku jsem nastavil id pro další práci s ním. Kompletní příkaz se vším všudy tedy vypadá následovně: $("#poctext").html(*pocasi* +", " + *teplota* + " °C
 <img id='dolu' src='obrazky/down.png' onclick='vice()'>"). V tomto momentě mám hotový div se základním přehledem aktuálního počasí.

U druhého divu (*pocasivice*) jsem opět využil jQuery příkazu $(„pocasivice“).html(). Mohl jsem využít i JavaScriptového příkazu, ale takto jsem ušetřil práci. Zde využiji zbylé proměnné tedy *pocitovateplota*, *vitr*, *vlhkost*, *sance*, *tlak*, *vychodslunce* a *zapadslunce*. Než začnu vkládat data z proměnných tak je potřeba vykreslit obrázek pro zviditelnění více informací o počasí. Vložil jsem tedy nejdříve element img. Přiřadil jsem id *dolu* pro další práci s tímto elementem. Posléze jsem nastavil zdroj obrázku na obrazky/down.png(45). Jako poslední jsem nastavil akci onlick funkci vice(). Po obrázku jsem vložil element pro odřádkování (br), aby se nepletl text s obrázky. V posledním kroku jsem vložil proměnné s příslušnými jednotkami (označením). Kompletní příkaz vypadá následovně: $("#pocasivice").html("<img id='dolu' src='obrazky/up.png' onclick='vice()'>
" + "pociťová teplota je: " + *pocitovateplota* + " °C
 vítr: " + *vitr* + " Km/h" + "
 vlhkost:" + *vlhkost* + " % " + "
 pravděpodobnost deště: "+ *sance* + " %"+"
 tlak: " + *tlak* + "hPa" + "
 východ slunce: "+ *vychodslunce* + "
 západ slunce: " + *zapadslunce*);

Teď již jen zbývá vytvořit funkci vice() a mohu se posunout k obrázku počasí. Funkci vice() jsem vytvořil za účelem zobrazení/skrytí dodatečných informací o aktuálním počasí a s tím spojených šipek (obrázků). Vytvořil jsem funkci vice() s dvěma proměnnými. Proměnné jsou *nahoru*, *dolu*. Proměnná *nahoru* má přiřazený obrázek jako hodnotu. Konkrétně obrázek *dolu* (down.png). Proměnná *dolu* má jako hodnotu přiřazen div s id *pocasivice*. Obě tyto proměnné jsem vytvořil pro větší přehlednost a jednoduchost. V této funkci dojde nejdříve ke kontrole proměnné *dolu*. Kontroluje se, jestli je style nastaven na „block“ (viditelný). Pokud ano dojde k nastavení style.display = „none“ a u proměnné *nahoru* dojde k nastavení style.display = „block“. Pokud proměnná *dolu* má jakoukoliv jinou hodnotu u atributu style.display než „block“ dojde k nastavení atributu

style.display = „block“ u proměnné *dolu*. U proměnné *nahoru* dojde k nastavení atributu style.display = „none“. Teď mám již nastavené zobrazování a mohu se posunout k obrázku aktuálního počasí.

K zobrazení aktuálního počasí využiji animovaných obrázků Skycons. Tyto animované obrázky jsou volně dostupné(53). Nejdříve je potřeba připojit potřebnou knihovnu přes příkaz <script>. Knihovna je dostupná na následující adrese: https://rawgithub.com/darkskyapp/skycons/master/skycons.js. Skycons jsou vytvářeny pomocí HTML 5 elementu canvas. Nejdříve tedy vytvořím element canvas s id *pocasiobrazek*. Nic více zde nenastavuji a vracím se do souboru *pocasi.js*. Nejdříve nastavím výchozí obrázek na jasno. Jako první provedu definici Skycons přes příkaz var skycons = Skycons({"color":"#34322e"}). Nastavil jsem tmavě hnědou barvu (ze schématu *hnedasimple*). Tímto mám vytvořený objekt Skycons v tmavě hnědé barvě. Dále jsem přidal objekt Skycons přes příkaz skyoncs.add(„pocasiobrazek“, Skycons.CLEAR_DAY). Přiřadil jsem canvas elementu s id *pocasiobrazek* Skycons element a nastavil obrázek pro jasno. Výhodou zde je, že dochází k vykreslování do canvasu (plátna) a není tedy nutné nahrávat obrázek a tím dochází k úspoře hardwaru. Jako poslední příkaz použiju skycons.play() čímž dochází ke spuštění animace.

Problém nastává u získané ikony. Jelikož v proměnné *ikona* je řetězec ve tvaru clear-day tak je nutné tento řetězec upravit do formátu CLEAR_DAY. K převedení jsem využil metody toupperCase()(37). Tato metoda převede všechny znaky dané proměnné, nebo řetězce na velké znaky. Tím jsem dostal CLEAR-DAY. Teď mi zbývá nahradit pomlčku podtržítkem. K tomu jsem využil metodu replace()(35). Pokud zadám pouze replace(“-“,“_“) dojde k nahrazení první pomlčky. Jelikož potřebuji nahradit více, jak jednu (pokud je jich více) tak poupravím nastavení. Ve výsledku použiji metodu replace následovně: replace(/-/g,“_“). Díky tomu dojde k nahrazení všech pomlček v řetězci za podtržítka. Vytvořil jsem si proměnnou *ikonapocasi*. Této proměnné jsem přiřadil následující hodnotu: ikona.replace(/-/g,“_“).toupperCase(). Výsledkem toho je, že proměnná *ikonapocasi* má hodnotu například RAIN, CLEAR_DAY a podobně. Bohužel jsem zjistil, že Intel XDK potažmo Cordova v některých případech fungují odlišně od moderních webových prohlížečů. Znamená to, že nelze nahradit část určující obrázek počasí proměnnou. Také není možné měnit barvu obrázku, jelikož když je jednou vykreslen tak ji nelze změnit na mobilním zařízení.

Vytvořil jsem rozhodovací konstrukci na jejímž základě dochází k nastavení příslušeného obrázku. Tato rozhodovací konstrukce (if) porovnává hodnotu proměnné *ikonapocasi*. Ikona může mít 10 různých stavů. K nastavení příslušného obrázku dochází pomocí příkazu skycons.set. Tento příkaz má dva parametry. Prvním parametrem je id (nebo třída) canvas (v tomto případě *pocasiobrazek*) a druhým parametrem je příslušné počasí (například Skycons.RAIN). K dispozici je 10 možných stavů, které jsou následující: *partly cloudy day* (částečně zataženo přes den), *partly cloudy night* (částečně zataženo přes noc), *clear day* (jasný den, jasno), *clear night* (jasná noc, jasno), *rain* (déšť), *sleet* (déšť se sněhem), *snow* (sněžení), *wind* (větrno) a *fog* (mlha). Výhodou vybraného API je především to, že je vždy řečeno, jaká má být použita animace (ikona) počasí. V tomto momentě mám již hotové celé počasí.

Chtěl jsem udělat dynamické počasí. Proto je potřeba vytvořit a nastavit proměnné, které budou získávat polohu na základě hardwaru. K získání pozice využiju HTML5 Geolocation API. Přistoupím k tomuto API přes příkaz navigator.geolocation.getCurrentPosition(21). Zde jsou tři parametry *success, error, opt*. První dva parametry jsou činnosti, co mají nastat v případě úspěšného získání pozice, nebo chyby. Poslední parametr je nastavení API. Vytvořil jsem tři proměnné. Proměnné jsou *longi, lattig* a *opt*. Proměnné *opt* jsem nastavil tři možnosti: enableHighAccuracy, timeout, maximumAge. HighAccuracy jsem nastavil na true (povolení možnosti vysoké přesnosti lokace), dále jsem nastavil timeout na 5000 (5 vteřin) a maximumAge na 0. MaximumAge jsem nastavil na hodnotu 0, jelikož pak dochází okamžitě k pokusu získat lokaci.

V případě úspěchu jsem vytvořil funkci *success* s parametrem *pos*. V této funkci jsem vytvořil proměnnou *cord*. Tato proměnná slouží k přístupu k souřadnicím. Proměnným *longi* a *lattig* nastavuji hodnoty. Konkrétně přes crd.latitude (pro lattig) a crd.longitude (pro longi). V případě chyby jsem vytvořil funkci *error* s parametrem *err*. V této funkci dochází pouze k zobrazení chybové hlášky přes příkaz console.warn. Chybová hláška vypíše kód chyby a error message. Na závěr jsem nastavil spuštění funkce pro získání počasí na 300 000 milisekund což je 5 minut.

Teď už jen zbývá vyřešit změnu jazyka (čeština, angličtina) a aplikace bude kompletní.

4.15. Volba jazyka

Pro zajištění jazyka jsem vytvořil JavaScriptový soubor jazyky.js. Jako první jsem se vrhnul na vyřešení zobrazení/schování nabídky s jazyky. Vytvořil jsem funkci změna(). Tato funkce je bez parametru a obsahuje pouze jedinou proměnnou jazyky. V proměnné „jazyky“ uchovávám div (jazyk2). Tento div má v základu atribut style.display nastaven na „block“. Vytvořil jsem rozhodovací konstrukci if. Pokud je div s id jazyk2 viditelný tedy má atribut style.display = „block“ tak dojde k nastavení atributu na „none“. Pokud je div s id jazyk2 neviditelný (atribut „none“), nebo jakýkoliv jiný atribut (in-line a podobně) dojde k nastavení atributu style.display = „block“.

V rámci divu jazyk2 jsem vytvořil dvě tlačítka (button element). Přiřadil jsem těmto tlačítkům hodnoty CZ a EN. Příslušnému tlačítku jsem přiřadil událost onlick podle hodnoty. Pro tlačítko s hodnotou CZ jsem nastavil funkci cz(). Pro tlačítko s hodnotou EN jsem nastavil funkci en().

Ze všeho nejdříve jsem vytvořil globální proměnnou jazyk. Této proměnné jsem přiřadil výchozí hodnotu cs. Jako první jsem vytvořil funkci cz(). V této funkci nejdříve nastavuji proměnné jazyk hodnotu cs. Dále jsem odstranil veškerý obsah vyučujících včetně odřádkování mezi jednotlivými divy (dvojicemi divů). Jelikož v rámci těchto divů a br elementu mám nastavené třídy tak je odstranění jednoduché. Využil jsem k tomu jednoduchý jQuery příkaz $(„třída“).remove(). Vytvořil jsem tedy tři jQuery příkazy. Příkazy jsou následující: $('.dva').remove(); $('.jedna').remove(); $('.oddelovac').remove(). V tomto momentě došlo k odstranění divů v divu vyucujici. Jelikož chci, aby nedocházelo k bugování obrázku u počasí tak jsem nastavil u divu s id pocasivice atribut style.display na hodnotu „none“. Po schování počasí jsem znovu zavolal funkci pro získání počasí a vyučujících. Na závěr jsem v této funkci nastavit divu jazyk2 atribut style.display na hodnotu „none“.

Dále jsem vytvořil funkci en(). Nejdříve jsem nastavil hodnotu proměnné jazyk na en. Dále jsem odebral veškerý obsah, který by mohl obsahovat české znaky, nebo informace v češtině. Využil jsem opět jQuery příkazu remove. Příkazy jsou stejné, jako v případě funkce cz. Po odstranění divů a br elementu jsem opět nastavil schování počasí (nastavení atributu style.display na „none“ pro div pocasivice). Zavolal jsem znovu funkce pro získání počasí a vyučujících. Na závěr jsem nastavil neviditelnost pro div jazyk2 (style.display = „none“).

Jako poslední funkci jsem vytvořil funkci na kontrolu připojení k internetu nazvanou net(). Zde jsem si vytvořil proměnnou net, které jsem přiřadil hodnotu získanou přes navigator.connection.type. Navigator.connection.type vrací hodnotu připojení například WIFI, 3G a podobně. Vytvořil jsem rozhodovací konstrukci if, kde dochází k porovnání hodnoty proměnné net. Nastavil jsem, že pokud je hodnota proměnné „none“ dojde k vypsání chybové hlášky „Není k dispozici připojení k internetu“. V rámci testování jsem si nastavil, že pokud je hodnota proměnné net jiná dojde k jejímu vypsání. V emulátoru jsem tuto funkci testoval na všech dostupných připojeních a vždy došlo ke správnému zobrazení. Jelikož potřebuji v ostré verzi tuto funkci spouštět častěji kvůli aktualizaci stavu u vyučujícího tak jsem vypustil zobrazení druhu připojení pokud je jiný než „none“. Na závěr jsem nastavil interval spouštění funkce na kontrolu připojení k internu na 25000 milisekund což je 25 vteřin. Je to proto, že kontrola aktuálního stavu vyučujícího je nastavena na každých 30 vteřin.

Jelikož nemám nastavenou změnu jazyka, jak v případě počasí tak vyučujících tak v případě kliknutí na tlačítko EN a CZ dojde pouze k znovu načtení dat v češtině. Jako první jsem vytvořil anglickou verzi pole na dny. Nazval jsem toho nové pole days a vložil hodnoty příslušných dnů v angličtině od pondělí do neděle a jeden prázdný řetězec (prázdné uvozovky) na závěr (na poslední místo pole). Po vytvoření pole jsem vložil rozhodovací konstrukci hned pod deklaraci proměnných jmeno, prijmeni a podobně. V této rozhodovací konstrukci porovnávám hodnotu proměnné jazyk. Pokud má hodnotu cs dojde k vytvoření divů vyučujících s využitím pole dny. Nastavil jsem také české popisky proměnných. Pokud má proměnná jazyk hodnotu en dojde k využití pole days a využití anglických popisků. Po nastavení vyučujících jsem otestoval, jestli vše funguje jak má. Jelikož zde je změna pouze v dvou nadpisech (konzultační hodiny, kontaktní údaje) a ve dnech konzultačních hodin tak zde nebyl žádný problém. Jako úplně poslední v celé aplikaci zbývá nastavit počasí.

U počasí musím nastavit nejdříve proměnné pro pozici (longitude, lattitude). Vyřešil jsem to tak, že do jQuery příkazy ve kterém dochází k přístupu JSON souboru a následnému načtení jsem vložil proměnné longi a lattig. Dále jsem potřeboval změnit jazyk slovního vyjádření počasí například z jasno na clear day. Připojil jsem tedy do url API část ?lang= kde dochází k nastavení jazyka API. Za tuto poslední část url jsem přidal

proměnnou jazyk. Tímto mám nastavené téměř vše a zbývá nastavit ještě popisky u více informací počasí.

V rámci počasí jsem po deklarování a nastavování proměnných vytvořil rozhodovací konstrukci if. Zde kontroluji hodnotu proměnné jazyk. Pokud je hodnota rovna en tak dojde k vložení proměnných do divu včetně anglických vysvětlivek (například temperature místo teplota a podobně). Jelikož proměnná jazyk může nabývat pouze dvou hodnot a to en a cs tak není potřeba ošetřovat případ jiné hodnoty. Proto tato rozhodovací větev kontroluje pouze jestli je hodnota rovna en. V případě, že není dochází k nastavení českých popisků.

Na závěr aplikace jsem nastavil na spuštění aplikace funkce pro získání vyučujících, získání počasí, čas, kontrolu stavu vyučujících a kontrolu internetového připojení.

4.16. Sestavení aplikace a nahrání na zařízení

Před samotným sestavením aplikace jsem otestovat v emulátoru, jestli fungují všechny funkce správně a jestli aplikace funguje tak jak má. Po otestování v emulátoru jsem provedl testování na zařízení. Aplikaci jsem testoval na dvou zařízeních. Testovacími zařízeními jsou tablet Nvidia Shield 8“ a mobilní telefon Xiaomi Redmi 2. Jelikož obě zařízení pracují na platformě Android tak je nutné nastavit přístupová práva. Přístupová práva k geolokačnímu hardwaru jsem nastavil v nastavení projektu přes příkaz *LOCATION_HARDWARE*(44).

Dále jsem nastavil fullscreen mode s orientací na šířku (viz příloha 3). Po veškerém nastavování jsem se přesunul k sestavení aplikace. Jelikož jsem potřeboval Android verzi tak jsem aplikaci sestavil pouze pro platformu Android. Vzhledem k charakteru aplikace jsem ji neumisťoval na Google Play. Aplikaci jsem do zařízení nahrál přes PC. Při instalaci aplikace do zařízení dojde vždy k vypsání potřebných práv aplikace. V tomto případě jsou oprávnění pro připojení k internetu a k lokalizačním službám. Po nainstalování je aplikace plně připravena k používání na zařízení.

5. Výsledky a diskuze

5.1. Dostupné mobilní platformy a jejich rozšířenost

Teoretická část práce byla zaměřena na zjištěný dostupných mobilních platforem.

Prvním pohledem bylo zjištění nejrozšířenějších mobilních platforem. Mezi nejrozšířenější platformy patří Android, iOS a Windows Phone s tím, že jsou zde i alternativní platformy určené pro rozvojové trhy. Alternativní platformy nejsou dostupné v rámci České republiky, a proto jsem se jimi nezabýval.

Nejrozšířenější mobilní platformou je Android následovaný iOS. Překvapivě je platforma Android nejrozšířenější v rámci celosvětového měřítka (prodejů). Ale v rámci lokálních trhů (v rámci zemí) jsou poměry platforem rozdílné podle kupní síly a ekonomické situace dané země. Například v Japonsku je více rozšířený iOS.

5.2. Způsoby vývoje mobilních aplikací

Dalším tématem pro teoretickou část aplikace bylo zjištění možností vývoje mobilních aplikací. Při zjišťování byly brány v potaz veškeré veřejně dostupné technologie vývoje aplikací. Bylo zjištěno, že lze aplikace vytvářet bez znalosti jakéhokoliv jazyka. K těmto účelům slouží například Microsoft Sienna. Dalšími plnohodnotnějšími způsoby jsou nativní aplikace, hybridní aplikace a webové aplikace.

Nativní aplikace jsou aplikace vytvořené speciálně pro jednu platformu. Pro každou platformu je potřeba psát nativní aplikaci v jiném jazyce. Hlavní výhodou nativní aplikace je rychlost a podpora veškerých technologií dané platformy.

Hybridní aplikace jsou aplikace vytvářené pro veškeré dostupné platformy. V rámci hybridní mobilních aplikací je využíváno HTML, CSS, JavaScriptu a jQuery. Lze také vytvářet z části hybridní aplikace. To znamená kombinace hybridních technologií spolu s nativním jazykem dané platformy.

Jako poslední byl zjištěn způsob vývoje webové aplikace. V tomto případě se jedná o aplikaci v rámci prohlížeče. Výhodou je, že uživatel vždy pracuje s nejaktuálnější verzí aplikace bez nutnosti stahovat novou verzi. Zde se využívá stejných technologií, jako v případě hybridní aplikace s tím rozdílem, že aplikace nemá takový přístup ke zdrojům zařízení.

Problémem při vývoji aplikace je především její náročnost. Vždy závisí na volbě lidí s příslušnými kompetencemi pro volbu způsoby vývoje. Ale měl by k tomu být přizván

člověk, který se v této problematice orientuje. Nevýhodou nativního způsobu vývoje je především cena a náročnost. V případě hybridní aplikace je naopak nižší cena, jak na koncové zařízení, tak i na vývojáře samotného.

5.3. Výběr způsobu vývoje a vytvoření aplikace

Cílem praktické části bylo navrhnout a implementovat mobilní aplikace univerzitní panel. Jako nejvýhodnější způsob vývoje na základě teoretické části byl zvolen hybridní způsob. Jako kritéria byla především cena zařízení. Hybridní způsob byl zvolen kvůli ceně zařízení, ale také kvůli případné budou obměně zařízení.

Důležitou částí návrhu bylo vytvoření rozložení prvků aplikace, aby splňovalo veškeré požadavky. Následně byl návrh grafické rozhraní vytvořen za pomocí HTML a CSS. Funkční prvky a celá funkcionalita aplikace byla implementována pomoci JavaScriptu a rozšiřující knihovny jQuery. Z Cordova pluginů byl navíc přidán plugin pro práci s geolokačnímu hardware zařízení.

5.4. Trendy v aplikacích a možnosti využití systémů

Jako nejvýznamnější trend v mobilních aplikací byla zjištěna orientace na cloud. Dále je následováno orientací na Inter of Things. Samozřejmě trendů je nespočet a v teoretické části práci nedošlo k pokrytí úplně všech z toho důvodu, že každý vidí trendy jinak.

Na univerzitě se pracuje s dvěma systémy UIS a Moodle. V teoretické části bylo zkoumáno možné využití v univerzitním informačním panelu. Vzhledem k porovnání dostupných informací byl vybrán pouze systém UIS. Samozřejmě je možné využít i externích systému, jako například služeb od společnosti Google. Zde ovšem nastává otázka, jestli je výhodné migrovat některá data mimo univerzitní systémy.

5.5. Testování aplikace

Výsledná vytvořená hybridní aplikace byla testována na třech zařízeních. Jedná se o dvě fyzická zařízení (smartphone a tablet) a jedno virtuální (Intel XDK emulátor pro iPad). Grafické rozhraní aplikace bylo testováno na třech zařízeních. Byly zde objeveny dvě chyby. Tyto chyby neovlivňovaly funkčnost aplikace, ale mohly dostat uživatele do nevhodné situace a způsobit obtíže pro práci v aplikaci. Chyby byly způsobeny špatným nastavením jednotek v CSS. Bylo navrženo a implementováno řešení pro opravu těchto chyb.

Dále jsem testoval aplikaci ve stavech, kdy aplikace nemá přístup k internetu a nemůže získávat aktuální informace o počasí a nemá přístup k údajům o pozici. V případě počasí došlo k nepříznivému stavu (nezobrazení obrázku ani textu). Tento problém byl ošetřen nastavením výchozího obrázku. Jelikož aplikace pracuje s lokálním JSON souborem tak zde nebyl problém se získáním vyučujících.

V rámci emulátoru jsem také testoval využití hardwaru zařízení přes záložku debug a profile v Intel XDK. K využití těchto dvou záložek je nutné připojit testovací zařízení. U CPU dochází k využití mezi 0.1 – 0.4%. Využití operační paměti je v rozmezí 156Kb po 338Kb (v případě načítání veškerých funkcí). Pokud dojde ke změně barevného schématu na modern (graficky náročnější) je využito 166Kb operační paměti. Testovacím zařízení u nároční na hardware byl Nvidia Shield tablet.

Aplikace byla testována na třech zařízeních s rozdílnými rozlišeními. Z hlediska funkcionality aplikace fungovala správně na všech zařízeních ve třech rozlišných rozlišeních. Nebyly nalezeny další chyby aplikace. Je vhodné tyto skutečnosti zohlednit během návrhu a implementace další aplikace, aby nenastaly podobné problémy.

5.6. Získané poznatky

Při vývoji hybridní aplikace jsem zjistil, že vývoj takovéto aplikace je jednodušší a rychlejší než v případě nativní aplikace. V teoretické části jsem zmiňoval, že je výhodou jednotné grafické prostředí aplikace pro veškeré platformy a to se také potvrdilo. Ovšem je potřeba si dát větší pozor na testování různých rozlišení. Potvrdilo se, že hybridní aplikace se z pohledu funkcionality zvládá vyrovnat nativní aplikaci bez větších problémů.

6. Závěr

Výsledkem této práce mělo být zjištění dostupných platforem a jejich rozšiřitelnosti. Dále zjištění možností vývoje aplikace a využití dostupných technologií. Následovalo zjištění trendů v mobilních aplikacích a v jejich vývoji. Jako poslední část teoretického východiska bylo zjištění využití stávajících informačních systémů na univerzitě v rámci aplikace. Vše výše uvedené se podařilo zjistit. Bylo zjištěno, že lze vyvíjet aplikace nativní, hybridní a webové. Dále bylo zjištěno, že nejvíce rozšířenou mobilní platformou je platforma Android následovaná platformou iOS. Po zjištění aktuálních trendů v mobilních aplikacích bylo zjištěno, že pro univerzitní panel je nejvhodnější využít univerzitní informační systém (UIS, is.czu.cz).

Výsledkem praktické části je mobilní aplikace pro tablet. Součástí praktické části je rozhodnutí pro způsob vývoje aplikace. Byla vybrána hybridní aplikace. Pro vývoj bylo využito vývojové prostředí Intel XDK. Dále bylo navrženo grafické rozhraní aplikace, které bylo následně implementováno. Po implementaci grafického rozhraní byla vytvořena aplikace včetně příslušné funkcionality. Aplikace byla následně testována na třech zařízeních (tablet, smartphone a virtuální emulátor). Po zjištění nedostatků aplikace bylo navrženo a implementováno řešení.

Největším problémem této hybridní aplikace se ukázalo být grafické rozhraní. Jelikož testovací zařízení mají rozdílné rozlišení tak docházelo k rozdílnému zobrazování na zařízeních. Tento problém je nezbytné zohlednit při vývoji hybridní mobilní aplikace.

7. Seznam použitých zdrojů

1. 3 Trends Shaping Mobile Development in 2016. *ADT Mag* [online]. 2016. https://adtmag.com/blogs/dev-watch/2016/01/mobile-dev-trends.aspx
2. 10 Mobile App Trends for 2016. *Enterprise APPS today* [online]. http://www.enterpriseappstoday.com/management-software/10-mobile-app-trends-for-2016.html
3. 15 Mobile App Development Trends To Look Out For In 2015. *Apptentive* [online]. 2015. http://www.apptentive.com/blog/15-mobile-app-development-trends-look-2015/
4. 15 Mobile App Development Trends To Look Out For In 2015. *July Rapid* [online]. http://julyrapid.com/15-mobile-app-development-trends-look-2015/
5. Android porn app secretly snapped users' photos and used them as ransom. *The Daily Dot* [online]. 2015. http://www.dailydot.com/technology/porn-app-android-photos-ransomware/
6. Application Development Trends 2015. *The App Solutions* [online]. http://theappsolutions.com/blog/marketing/app-trends-2015/
7. App Monetization: 6 Bankable Business Models That Help Mobile Apps Make Money. *Localytics* [online]. 2014. http://info.localytics.com/blog/app-monetization-6-bankable-business-models-that-help-mobile-apps-make-money
8. Application programming interface. *Wikipedia* [online]. https://en.wikipedia.org/wiki/Application_programming_interface
9. A Short History of JavaScript. *W3.org* [online]. https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript
10. Brand Guidelines. *JQuery Foundation* [online]. http://brand.jquery.org/
11. Cascading Style Sheet, designing for the Web - Chapter 20: The CSS saga. *W3.org* [online https://www.w3.org/Style/LieBos2e/history/
12. Cascading Style Sheets. *Wikipedi*a [online]. https://en.wikipedia.org/wiki/Cascading_Style_Sheets
13. Count of Active Applications in the App Store. *Pocket gamer* [online]. http://www.pocketgamer.biz/metrics/app-store/app-count/

14. Čínský středoškolák prodal ledvinu, aby měl na iPhone. *Kurzycz* [online]. 2012. http://www.kurzy.cz/tema/detail/cinsky-stredoskolak-prodal-ledvinu-aby-mel-na-iphone-917931.html
15. Definition of: mobile platform. *PC Mag* [online]. http://www.pcmag.com/encyclopedia/term/47144/mobile-platform
16. Gartner Says Emerging Markets Drove Worldwide Smartphone Sales to 15.5 Percent Growth in Third Quarter of 2015. *Gartner* [online]. 2015. http://www.gartner.com/newsroom/id/3169417
17. Historie Androidu v kostce aneb Od verze 1.0 až po Android M. *Svět Androida* [online]. 2015. http://www.svetandroida.cz/historie-androidu-201506
18. How Facebook Mobile Was Designed to Write Once, Run Everywhere. *Readwrite* [online]. 2011. http://readwrite.com/2011/09/29/how-facebook-mobile-was-design.
19. HOW TO: Determine the Right Price For Your Mobile App. *Mashable* [online]. http://mashable.com/2011/08/17/price-mobile-app/
20. HTML5 Canvas. *W3schools* [online]. http://www.w3schools.com/html/html5_canvas.asp
21. HTML5 Geolocation. *W3schools* [online]. http://www.w3schools.com/html/html5_geolocation.asp
22. HTML 5 Introduction. *W3schools.com* [online]. http://www.w3schools.com/html/html5_intro.asp
23. HTML5 versus nativní: debata o mobilních aplikacích. *Zdroják* [online]. 2011. https://www.zdrojak.cz/clanky/html5-versus-nativni-debata-o-mobilnich-aplikacich/
24. HTML. *Wikipedia* [online]. Dostupné z: https://en.wikipedia.org/wiki/HTML
25. Introducing JSON. *JSON* [online]. http://json.org/
26. Introduction. *Material design* [online] https://www.google.com/design/spec/material-design/introduction.html
27. IOS. *Wikipedia* [online]. https://en.wikipedia.org/wiki/IOS
28. Intel® XDK. *Intel Developer Zone* [online] https://software.intel.com/en-us/intel-xdk
29. JavaScript. *Wikipedia* [online]. https://en.wikipedia.org/wiki/JavaScript

30. JavaScript Date Methods. *W3schools* [online]. http://www.w3schools.com/js/js_date_methods.asp
31. JavaScript getHours() Methods. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_gethours.asp
32. JavaScript getMinutes() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_getminutes.asp
33. JavaScript getSeconds() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_getseconds.asp
34. JavaScript round() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_round.asp
35. JavaScript String replace() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_replace.asp
36. JavaScript String substr() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_substr.asp
37. JavaScript String toUpperCase() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_touppercase.asp
38. JavaScript toFixed() Method. *W3schools* [online]. http://www.w3schools.com/jsref/jsref_tofixed.asp
39. JavaScript Timing Events. *W3schools* [online]. http://www.w3schools.com/js/js_timing.asp
40. Je lepší nativní aplikace nebo mobilní web? *Marketing journal* [online]. 2012. http://www.m-journal.cz/cs/internet/je-lepsi-nativni-aplikace-nebo-mobilni-web-__s281x9241.html
41. JQuery. *Wikipedia* [online]. https://en.wikipedia.org/wiki/JQuery
42. LACKO, Ľuboslav. *Vývoj aplikací pro Windows 8.1 a Windows Phone*. 1. vyd. Brno: Computer Press, 2014. ISBN 9788025138229.
43. LACKO, Ľuboslav. *Vývoj aplikací pro Android*. 1. vyd. Brno: Computer Press, 2015. ISBN 9788025143476.
44. Manifest.permission. *Android Developers* [online]. http://developer.android.com/reference/android/Manifest.permission.html
45. Material icons. *Google Design* [online]. https://design.google.com/icons/
46. Mobile app. *Wikipedia* [online]. https://en.wikipedia.org/wiki/Mobile_app

47. Mobilní platforma. *Sun marketing* [online]. http://www.sunmarketing.cz/nastroje/slovnik/mobilni-platforma
48. Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. *Salesforce Developers* [online]. 2015.https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options
49. Number of Android applications. *AppBrain* [online]. http://www.appbrain.com/stats/number-of-android-apps
50. *Portal:Apple Inc* [online]. https://en.wikipedia.org/wiki/Portal:Apple_Inc.
51. Rethinking Mobile: 10 Trends in Enterprise Strategy and Development. *Oracle* [online]. http://www.oracle.com/us/corporate/features/rethinking-mobile/index.html
52. Smartphone OS sales market share. *Kantar Worldpanel ComTech* [online]. http://www.kantarworldpanel.com/global/smartphone-os-market-share/
53. Skycons. *Dark Sky App* [online]. https://darkskyapp.github.io/skycons/
54. Top 10 Mobile App Development Trends for 2015. *CODECONDO* [online]. 2015. http://codecondo.com/top-10-mobile-app-development-trends-for-2015/
55. VÁVRŮ, Jiří. *IPhone: vývoj aplikací*. 1. vyd. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-4457-5.
56. What is mobile app? *WhatIs* [online]. 2013. http://whatis.techtarget.com/definition/mobile-app
57. Windows 10 Mobile guide for IT pros - deply devices, help. *Windows* [online]. https://technet.microsoft.com/en-us/windows/mt631176
58. Windows Phone. *Wikipedia* [online]. https://cs.wikipedia.org/wiki/Windows_Phone
59. Windows Phones. *Microsoft* [online] https://www.microsoft.com/cs-cz/windows/phones
60. Windows Store Policies. *Microsoft* [online] https://msdn.microsoft.com/library/windows/apps/dn764944.aspx
61. Loga ČZU. *Student.czu.cz* [online]. https://student.czu.cz/index.php?id_menu=374&id_submenu=407&id_con_kind=5&id_con=3103
62. ČZU. *MenuPraha* [online]. http://menupraha.cz/restaurace/foto/9085/logo.jpg

63. Architecture overview of Crodova platform. *Cordova* [online]. https://cordova.apache.org/docs/en/latest/guide/overview/

Příloha 1: Barevná schémata simple

Schéma hnědá simple (zdroj: vlastní)

Schéma červená simple (zdroj: vlastní)

Schéma modrá simple (zdroj: vlastní)

Schéma zelená simple (zdroj: vlastní)

Příloha 2: Barevná schémata modern

Schéma červená modern (zdroj: vlastní)

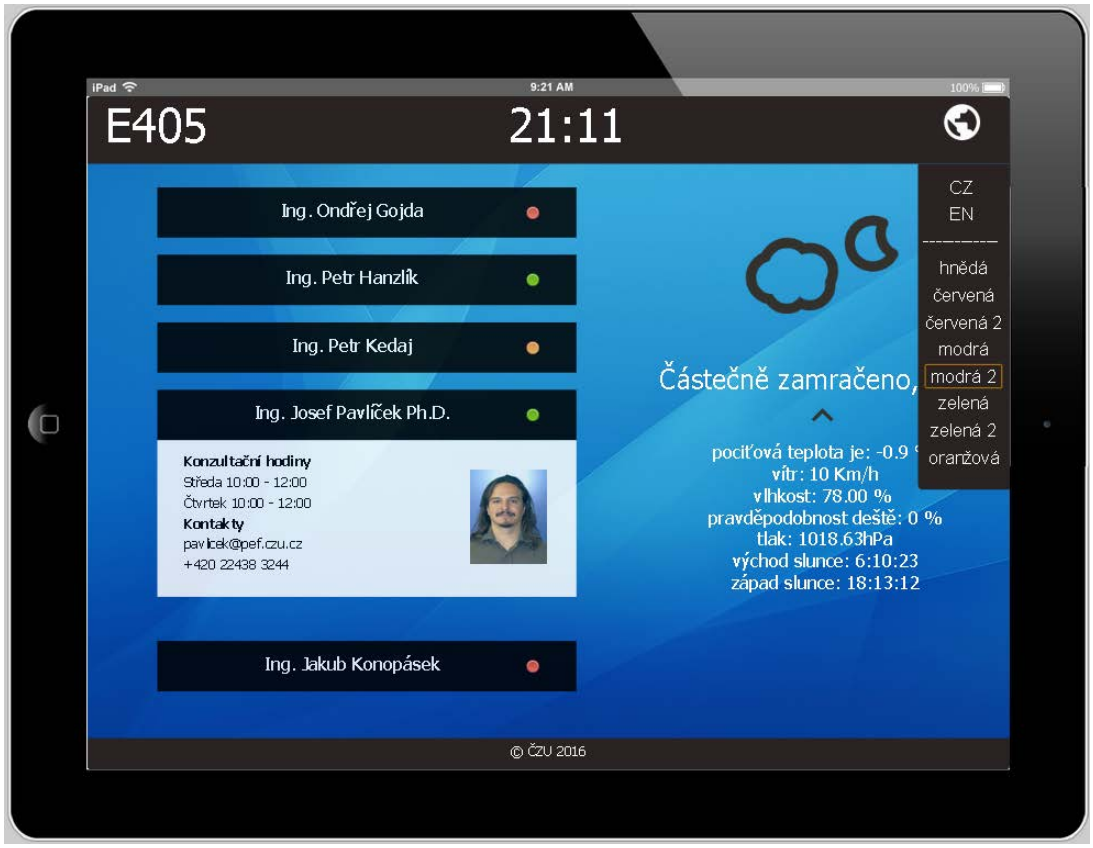


Schéma modrá modern (zdroj: vlastní)

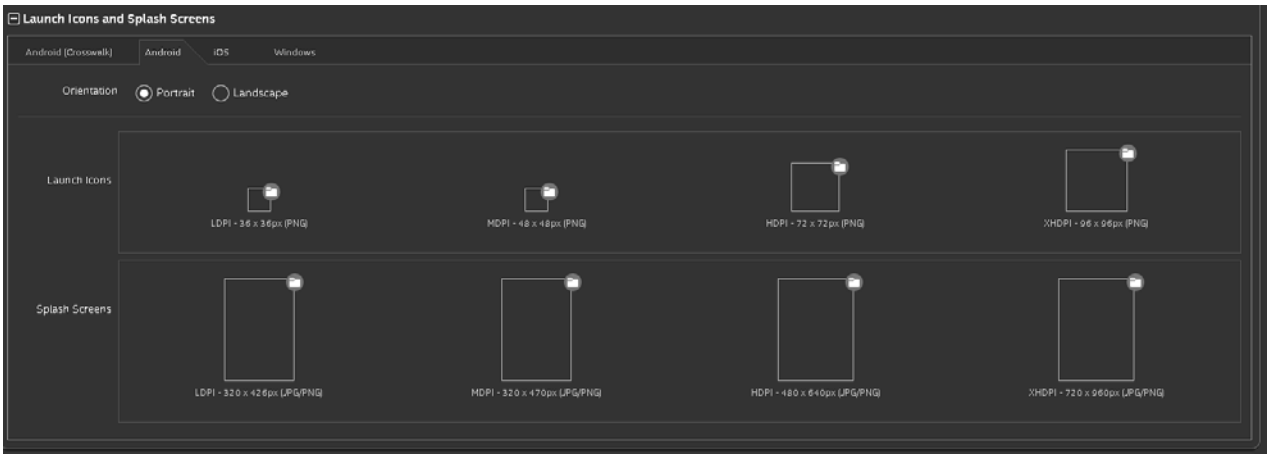
Schéma zelená modern (zdroj: vlastní)

Schéma oranžová modern (zdroj: vlastní)

Příloha 3: Nastavení aplikace

Nastavení geolokačního modulu (zdroj: vlastní)

Nastavení přístupových práv (zdroj: vlastní)



Nastavení ikony a spouštěcí obrazovky (zdroj: vlastní)

Nastavení fullscreen módu a zobrazení na šířku (zdroj: vlastní)

Příloha 4: Loga univerzity využití v aplikaci

Logo použité pro ikonu aplikace v systému (zdroj: 62)

Logo univerzity použité pro načítací obrazovku aplikace (zdroj: 61)