Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informační systémy a technologie

Využití frameworku IONIC pro vývoj multiplatformní mobilní aplikace

DIPLOMOVÁ PRÁCE

Student : Bc. Pavel Gruda

Vedoucí : doc. Ing. Alena Buchalcevová, Ph. D.

Oponent : Ing. Jaroslav Zapadlo

2016

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 12. 12. 2016 . . . . . .

Pavel Gruda

Poděkování

Rád bych na tomto místě poděkoval vedoucí práce doc. Ing. Aleně Buchalcevové, Ph.D. za cenné rady, připomínky a za čas strávený konzultacemi. Dále děkuji zástupcům firmy Notix s.r.o., kteří mi poskytli konzultace během analýzy a implementace projektu PowerFLOW mobile. Poděkování patří také mé rodině a přátelům, kteří mě po celou dobu studia podporovali.

Abstrakt

Cílem diplomové práce je analýza frameworku *IONIC* a zjištění vhodnosti tohoto frameworku pro multiplatformní mobilní vývoj. Teoretická část diplomové práce je zaměřena na analýzu současného trhu, úvodu do mobilního vývoje a popisu frameworku IONIC a s ním spojených technologií. Praktická část se zabývá analýzou a návrhem mobilní aplikace PowerFLOW pomocí *Metodiky pro Malé Softwarové projekty*, popisem vytvořené mobilní aplikace a ukázkami kódu. V poslední části práce je framework IONIC zhodnocen na základě stanovených kritérií. Hodnocení je provedeno na základě zkušeností nabytých během implementace mobilní aplikace PowerFLOW. V práci jsou uvedeny také osobní zkušenosti z implementace pomocí frameworku *IONIC*. Na základě této práce lze framework IONIC doporučit pro multiplatformní mobilní vývoj.

Klíčová slova

mobilní vývoj, multiplatformní aplikace, IONIC framework, JavaScript, mobilní aplikace, MMSP

Abstract

The aim of the thesis is to analyse framework IONIC and determine the suitability of this framework for cross-platform mobile development. The intent of the theoretical part is to analyse the current market, introduction to the mobile development and a description of the framework IONIC and related technologies. The practical part deals with the analysis and design mobile application PowerFLOW, describing the implemented mobile application and sample code. The last part is about framework IONIC evaluation based on set criteria. The evaluation rating was determined on the experience gained during the implementation of mobile application PowerFLOW. There are also listed personal experience of the implementation. Based on this thesis the IONIC can be recommended as the framework for cross-platform mobile development.

Keywords

mobile development, cross-platform applications, IONIC framework, JavaScript, MMSP, mobile application

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Mobilní telefony se neustále zdokonalují a postupně přejímají více a více funkcí od stolních počítačů. Také značně pokročil jejich výkon. Pro uživatele je bezesporu výhoda mít u sebe malé přenosné zařízení, které umožňuje využívat programy prakticky kdekoliv. Proto je vhodné mít mimo desktopových aplikaci i mobilní aplikace, umožňující přesunout funkcionalitu z desktopových počítačů do mobilního zařízení. Je možné využívat i funkcionalitu, která na stolním počítači není k dispozici. Jako například fotoaparát, GPS, gyroskop, dotykové ovládání a další funkce, které jsou pro chytrý telefon běžné.

Stojíme před rozhodnutím, pro jaké platformy mobilních zařízení aplikace vyvíjet. Podle článku *Smartphone OS Market Share, 2016 Q2* [6] mají operační systémy *Android* a *iOS* v druhém kvartále roku 2016 99,3 % prodejů z celého trhu. Zbylých 0,7 % je rozděleno mezi ostatní operační systémy, ze kterých má největší zastoupení *Windows Phone* 0,4 %. Z těchto faktů vyplývá, že většinu trhu mobilních telefonů ovládli dva operační systémy, pro které se vyplatí mobilní aplikace vyvíjet, pokud je má používat široká veřejnost. Ostatní mobilní operační systémy podle trendu zanikají.

Další rozhodnutí, které je potřeba před vývojem mobilních aplikací udělat je, zda vytvořit nativní mobilní aplikace nebo použít framework umožňující následné nasazení na výše zmíněné operační systémy. Nativní mobilní aplikace sice umožňují použití všech specifických vlastností daných operačních systému, ale na druhou stranu její vývoj a udržitelnost verzí pro více operačních systémů vyžaduje oddělený vývoj aplikací pro jednotlivé platformy a programátory s různými specializacemi. S rychle se měnícím trhem a technologickým vývojem je potřeba vytvářet aplikace s nízkými náklady, co nejrychleji.

Jedním z vhodných frameworků pro multiplatformní vývoj mobilních aplikací je framework *IONIC*. Tento framework využívá komponenty *CSS*, *HTML* a *Javascript*, které jsou přizpůsobené pro mobilní vývoj. Aplikace v něm vyvíjené jsou dobře udržovatelné, protože mají předdefinovanou strukturu. Framework je open-source[1]. Jeho komunita tedy vytváří různé návody a vylepšení pro jeho používání. Má také rozsáhlou webovou dokumentaci [3].

Tato diplomová práce se zabývá prozkoumáním frameworku IONIC a jeho použitím při vývoji multiplatformní mobilní aplikace. Rozebírá framework, jak po teoretické, tak i po praktické stránce. Praktická část popisuje funkcionalitu a problémy, na které jsem narazil při implementaci mobilní aplikace. Pro analýzu a návrh mobilní aplikace je použita *Metodika pro menší softwarové projekty* [1].

[1] Open-source je něco, co lidé mohou libovolně modifikovat a sdílet, protože jeho návrh je veřejně dostupný. [7]

1.1 Vymezení tématu práce a důvod výběru tématu

Téma jsem si vybral, protože pracuji v oblasti vývoje aplikací. Momentálně mám podrobné znalosti funkcionality jen na backend straně aplikací. Vývoj mě zajímá jako celek, tak bych rád pochopil i funkcionalitu frontend části aplikace. Framework IONIC považuji za dobrý z toho důvodu, že ho používá více než 1,2 milionu mobilních aplikací [3]. Vypadá to, že bude i nadále pokračovat jeho využívání, jako jednoho z předních frameworků pro vývoj multiplatformních aplikací. Napovídá tomu i fakt, že momentálně je vydaný *IONIC 2* ve verzi beta. Také mě motivuje i skutečnost, že v rámci práce vznikne aplikace, která se bude reálně využívat a na které budu popisovat vlastnosti použitého frameworku.

1.2 Cíle práce

Hlavním cílem diplomové práce je analýza frameworku IONIC a zjištění vhodnosti tohoto frameworku pro multiplatformní mobilní vývoj. Cíl je složený z dílčích cílů, které jsou definovány níže.

K dosažení hlavního cíle jsou určeny dílčí cíle:

- Popis frameworku IONIC.
- Analýza aplikace vyvíjené ve frameworku IONIC metodikou MMSP [1].
- Popis části kódu aplikace vytvořené ve frameworku IONIC.
- Zhodnocení frameworku IONIC.

1.3 Cílové skupiny

První cílovou skupinou této práce jsou začínající frontend programátoři, kteří hledají vhodný nástroj pro vývoj multiplatformních mobilních aplikací. Zde se mohou dozvědět, základní informace o frameworku IONIC včetně okomentovaných částí kódu. Druhou vhodnou cílovou skupinou jsou analytici, kteří v práci mohou vidět aplikovanou metodiku MMSP na vývoji mobilní aplikace. Práce je vhodná i pro projektové managery, kteří rozhodují, který framework pro vývoj mobilní aplikace použít.

1.4 Použité metody

Na analýzu a návrh aplikace je použita v diplomové práci *Metodika pro Malé softwarové projekty* [1]. Tato metodika byla vytvořena lokalizací a customizací metodiky *OpenUP* [8]. Je inspirována i mnohými dalšími, především agilními metodikami. Slouží také k efektivní spolupráci v rámci projektových týmů do 10 členů. Využívá grafický jazyk *UML* [2] pro vizualizaci, specifikaci a dokumentaci softwarových projektů. Pro analýzu rizik jsem použil *Winterlingovu krizovou matici* [37], která umožňuje detekovat účinky rizik na organizaci/projekt. Během vývoje byly prováděny systémové testy pro ověření nově implementované funkcionality. Na již fungujících systémech se prováděly regresní testy pro zjištění konzistence vstupů do vyvíjené aplikace. Během implementace byla použita metoda folder-by-feature, která koncertuje funkcionalitu jedné stránky do jedné složky. Tento způsob umožnuje vyšší přehlednost projektu a možnost kooperace více lidí na projektu, protože nedochází ke kolizím s úpravou kódu více lidmi v jednom souboru. Pro dokumentaci REST API byl využit framework Swagger.

1.5 Struktura práce

Jednotlivé kapitoly práce postupně přecházejí od obecnějšího popisu multiplatformního mobilního vývoje k praktické části analýzy, implementace a následnému zhodnocení.

V první části diplomové práce je kapitola *úvod*, která slouží uvedení do problematiky a popisuje důvody ke vzniku této práce. Po přečtení této kapitoly má čtenář jasnou představu o čem se ve zbytku práce jedná a pro koho je práce určena.

Druhou kapitolou je *komentovaná rešerše*. Ta seznamuje čtenáře s pracemi zabývajícími se problematikou, kterou je vhodné znát při analýze a vývoji mobilních aplikací.

Třetí kapitola je *Úvod do mobilního vývoje*. V první časti je čtenář seznámen se současnými trendy technických zařízení a důvodech pro vývoj mobilních aplikací. V druhé části je popsán přístup vývoje mobilních aplikací.

Čtvrtá kapitola rozebírá obecné principy fungování frameworku IONIC a přidružených technologií. Popisuje prvky využité ve frameworku jako např. javascriptový framework AngularJS, kaskádové styly, využité HTML, aj.

V páté kapitole je aplikována *Metodika pro Malé softwarové projekty* na vývoj mobilní aplikace *PowerFLOW mobile*. V této kapitole jsou diagramy popisující fungování systémů souvisejících s aplikacemi a další části, které jsou podle metodiky spojeny s vývojovým projektem. V první části kapitoly je analyzován současný stav již běžících komponent a ve druhé části se analyzuje vznikající mobilní aplikace.

V šesté kapitole jsou popsány části vyvinuté aplikace PowerFLOW mobile. Je zde nastíněna struktura projektu, struktura zakládání stránek a praktické ukázky kódu.

V sedmé kapitole se využívá načerpaných zkušeností z analýzy frameworku IONIC a vývoje mobilní aplikace ve frameworku. Je zde provedeno zhodnocení frameworku z různých pohledů. Osmá kapitola je závěr práce.

1.6 Předpoklady a omezení práce

Během praktického vývoje aplikace budu mentorovaný zástupci firmy Notix, s.r.o., kteří jsou v rolích zadavatelů. Tento způsob mi umožňuje získávat cenné rady při analýze i vývoji aplikace. Nicméně to může projekt vývoje mobilní aplikace i zdržet, pokud nebudu mít včas akceptované jednotlivé výstupy projektu.

Aplikace vzniklá během psaní diplomové práce se bude používat v komerčním prostředí firmy Notix, s.r.o. a předpokládá se, že bude nadále pokračovat její vývoj. Proto bude kladen důraz na udržitelnou strukturu projektu.

Oblast multiplatformního vývoje mobilních aplikací je velmi rychle se rozvíjející odvětví a framework *IONIC* má velkou základnu uživatelů. Vzhledem k faktu, že je framework opensource, je možné, že již během psaní práce budou vznikat novější postupy. Proto není jedno, na kterých verzích použitých technologií se bude aplikace vyvíjet a spouštět.

Pro porozumění práce je třeba alespoň základních znalostí programovacích jazyků a principu fungování internetových stránek.

1.7 Přínosy práce

Práce ukáže výhody a nevýhody vývoje mobilní aplikace ve frameworku *IONIC* na reálném projektu. Na konci práce je framework IONIC zhodnocen.

Práce ukáže vývoj mobilní aplikace na reálném produktu *PowerFLOW mobile*. Je to mobilní klient pro aplikaci, která slouží k modelování, správu procesů a úkolů v organizaci. V rámci práce se tedy vytvoří mobilní aplikace, která se bude reálně používat a na které se bude pokračovat v dalším vývoji nových funkcionalit. Součástí práce jsou okomentované části kódu.

V rámci práce bude použita *Metodika pro Malé softwarové projekty* [1] a budou vidět její výstupy na reálném projektu vývoje mobilní aplikace.

Práce zkoumá framework IONIC, který vznikl v roce 2013. Je to tedy moderní technologie, o níž není dostatek materiálů v akademických zdrojích. DP může sloužit jako pomůcka pro práci s frameworkem.

Práce pomůže studentům, vývojářům nebo manažerům, kteří budou váhat, jaký framework pro vývoj mobilních aplikací použít.

2 Komentovaná rešerše informačních zdrojů

Během rešerše zdrojů jsem našel knihy psané v anglickém jazyce. Jedná se o oblast, která se velmi rychle vyvíjí, proto jsem nenašel žádné české zdroje. Překlad by pravděpodobně trval dlouho a kniha by již byla zastaralá. Tématem se zabývají i akademické práce, které většinou porovnávají jednotlivé frameworky. I akademické práce jsou v současné době spíše zastaralé a v pouze v jedné jsem našel zmínku o frameworku IONIC, který tam není rozebíraný, protože byl v té době velmi mladý. Poslední částí jsou elektronické zdroje. Tato část obsahuje nejaktuálnější informace. Je to většinou dokumentace použitých frameworků.

2.1 Knihy

Kniha *Learning Ionic* [12] obsahuje popis framework IONIC a přidružených technologií. V první části se seznamujeme s frameworkem IONIC samotným a technologií AngularJS, následuje stylování, vývoj praktické aplikace, popis ostatních vhodných technologií a na závěr kniha informuje o způsobu vytvoření aplikace spustitelné na více platformách.

Podobnou strukturu má i druhá kniha *Ionic in action* [13], kde je na začátku seznámení s frameworkem Ionic a multiplatformními aplikacemi. V další části popisuje propojení IONIC frameworku a AngularJS, navigační prvky, ukázkovou aplikaci. Nakonec předvádí pokročilé techniky programování, debugování[2] a publikování aplikací.

Kniha *Příklady modelů analýzy a návrhu aplikace v UML* [1] v první části popisuje *Metodiku pro Malé softwarové projekty*. Tato metodika vznikla z metodiky OpenUI, používá UML a lze podle ní řídit malé softwarové projekty do 10 členů. V druhé části knihy jsou praktické ukázky aplikace metodiky na dvou projektech.

Kniha *UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky* [2]. V první části seznamuje kniha s UML standardem, dále popisuje sběr požadavků, analýzu a návrh aplikace. V poslední částí je pracovní postup implementace a doplňkový materiál.

Další knihy jsou o frameworku AngularJS, na kterém je Ionic postavený. Kniha *AngularJS* [21] nejdříve uvede čtenáře do problematiky, popisuje strukturu a vývoj aplikací v AngularuJS, direktivy, komunikaci se serverem, a nakonec zmiňuje best-practises. Kniha *AngularJS in Action* [23] oproti předchozí popisuje navíc formuláře, validace a animace v AngularuJS. Kniha Beginnin AngularJS [24] popisuje podobné prvky, jako dvě

[2] Debugování je způsob odstraňování chyb aplikace [autor]

předchozí knihy s tím rozdílem, že klade větší důraz na základy. Kniha, nebo spíše příručka *AngularJS in 60 Minutes* [22] obsahuje základní rychlý přehled pro práci s AngularemJS.

2.2 Akademické práce

Akademická práce *Srovnání přístupů multiplatformního vývoje mobilních aplikací* [5] porovnává jednotlivé frameworky pro vývoj multiplatformních aplikací. V rámci práce jsou určena kritéria porovnávání frameworků a dle kritérií se určuje jejich vhodnost.

Akademická práce *Analýza frameworků pro vývoj multiplatformních mobilních aplikací využívající HTML technologie* [14] porovnává oproti předchozí práci jen frameworky založené na technologii HTML.

Akademická práce *Návrh a vývoj multiplatformních mobilních aplikací* [15] z Univerzity Hradec Králové opět srovnává multiplatformní frameworky, ale v rámci práce je řešena implementace aplikace Worktastic, která slouží k zaznamenávání docházky.

Akademická práce *Vývoj multiplatformních mobilních aplikací pro prodej zboží* [16] je z Masarykovi univerzity a v rámci práce se řeší výběr vhodného mobilního multiplatformního frameworku a implementace ukázkové aplikace ve frameworku PhoneGap.

Akademická práce *Multiplatformní vývoj mobilních aplikací pomocí webových technologií* [17] porovnává jednotlivé frameworky a v druhé části diplomové práce vytvořil jednoduchou aplikaci ve frameworku PhoneGap. V rámci této práce se zmiňuje o frameworku IONIC, který byl v té době novinkou, a proto ho ani nezahrnul do průzkumu.

Akademická práce *Nástroje pro podporu vývoje nativních multiplatformních mobilních aplikací* [18] definuje kritéria vhodného výběru frameworku. V práci vybírá framework Xamarin a demonstruje v něm vývoj jednoduché multiplatformní aplikace.

Akademická práce *Přístupy k řešení mobilních webových aplikací* [19] ukazuje charakteristiky uživatelů používající osobní počítač, tablet a chytrý telefon. Práce se zabývá i rozložením uživatelského prostředí a vlivem psychologie v aplikacích. Porovnává také historii vývoje. Neřeší nativní multiplatformní aplikace.

Akademická práce *Moderní metody tvorby nativních multiplatformních mobilních aplikací* [20] z Univerzita Tomáše Bati ve Zlíně rozebírá hlavní mobilní platformy. Shledává za nejvhodnější vývojový nástroj multiplatformní mobilní aplikace nástroj Xamarin a ten také ve většině práce popisuje. Ukazuje i návrhové vzory, které se dají využívat při vývoji s nástrojem Xamarin.

2.3 Elektronické zdroje

Elektronický zdroj *IONIC Documentation* [3] popisuje jakým způsobem založit projekt ve frameworku IONIC, jaké lze používat multiplatformní komponenty, obsahuje připravené kódy v Javascriptu a také popisuje, jak používat CLI[3] .

Elektronický zdroj *NgCordova Documentation* [4] slouží jako elektronický portál k dokumentaci funkcí, které lze u telefonu použít. NgCordova je framework používaný k rozhraní mezi AngularemJS a mobilními telefony.

Elektronický zdroj *Guide to Angular 1 Documentation* [25] slouží k přehledu operací knihoven Javascriptu. Jsou zde uloženy ukázkové kódy pro vytvoření formulářů, filtrů, použití stylů a mnoho dalších funkcí, které lze ve frameworku AngularJS vytvořit.

Elektronický zdroj *Swagger* [32] obsahuje dokumentaci a zdrojové kódy ke frameworku, který umožňuje popis REST API

Ostatní elektronické zdroje dokumentací frameworků, které souvisejí s frameworkem IONIC. Stránky obsahují dokumentaci, zdrojové kódy a návody pro použití. Guide to Angular 1 Documentation [25], Bower [26], Node.js [27], Git [28], Sass [29], Gulp [30], Apache Cordova [31].

[3] CLI neboli command line utility je přikazová řádka Ionic frameworku

3 Úvod do mobilního vývoje

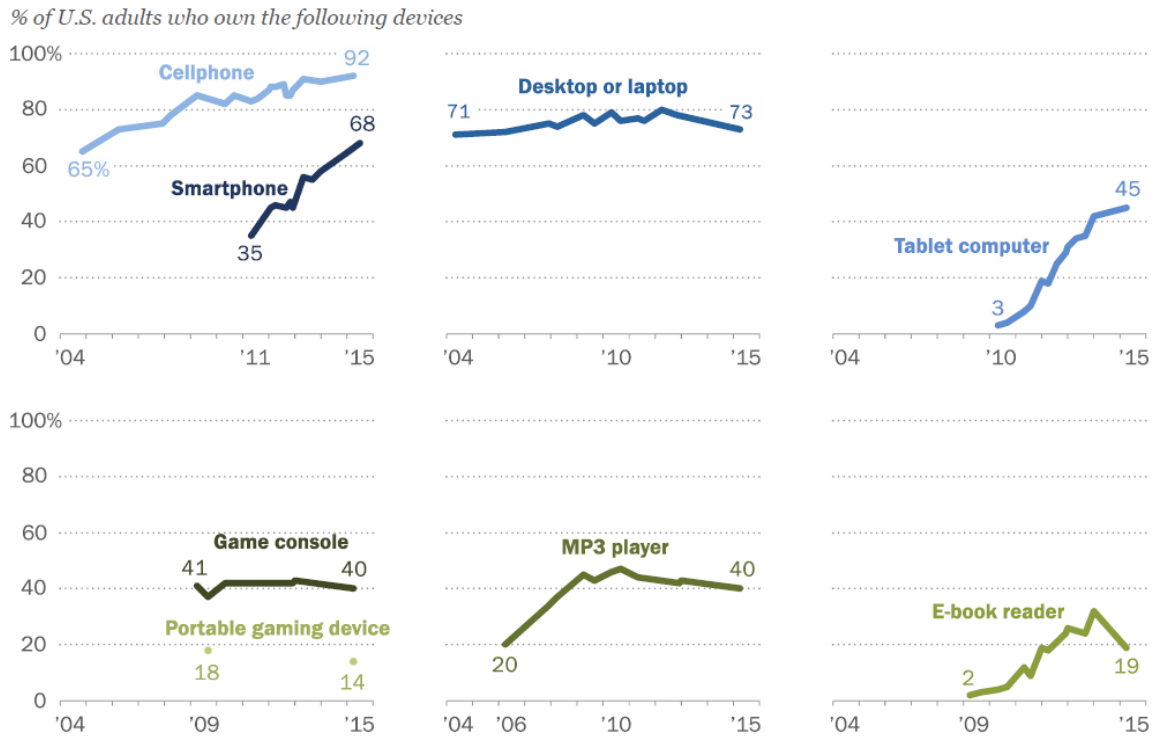
V první části kapitoly jsou popsány trendy používaných technických zařízení během několika minulých let. Je zde objasněno, proč je vývoj programů pro mobilní operační systémy důležitý. V druhé části se kapitola věnuje vymezením pojmů mobilního vývoje a definuje typy aplikací, které se dají používat na různých mobilních platformách.

3.1 Charakteristika současného stavu problémové oblasti

V této části je přiblížena problematika vývoje mobilních aplikací. Je zde rozebrán důvod, proč je vývoj mobilních aplikací v dnešní době tak důležitý a zároveň je objasněno, že není jednoduché zvolit platformu ve které budeme aplikaci vyvíjet.

Trendy technických zařízeních

Z odborného článku *Technology Device Ownership: 2015* [9], který provádí průzkum amerického trhu v oblasti vlastnictví technologických zařízení, lze vypozorovat, že od roku 2011 rapidně stoupá procento dospělých, kteří vlastní chytrý telefon. Podobný nárůst je i v oblasti vlastnictví tabletů, nicméně mají v roce 2015 o 23 procent méně podílu než chytré telefony. Vše je patrné z obrázku 1. Z grafů na obrázku lze dále vyčíst, že se zvyšuje procento lidí vlastnící jakýkoliv telefon. Na druhou stranu procenta vlastnictví osobních počítačů, herních konzolí, hudebních přehrávačů a elektronických čteček buď stagnují nebo se zmenšují. Z těchto údajů lze usoudit, že bude stále více podporovat vývoj aplikací na mobilní zařízení, protože mají čím dál tím větší podíl na trhu.

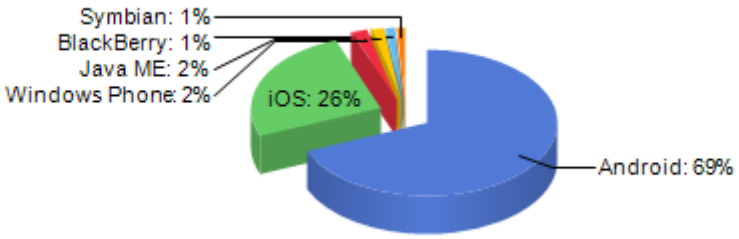


Obrázek 1:
Smartphones, Tablets Grew in Recent Years; Other Devices Declined or Stayed Flat [9]

Podíl operačních systémů chytrých telefonů na trhu

Při vývoji mobilní aplikace pro širokou veřejnost je pro nás jedním z rozhodujících faktorů podíl operačních systémů chytrých mobilních telefonů na trhu. Obrázek 2 zobrazuje celkový podíl operačních systémů mobilních telefonů a tabletů v roce 2016. Je z něj patrné, že nadpoloviční většinu trhu dominuje platforma Android. Jako druhý je nejčastěji používaným operačním systémem iOS se čtvrtinovým podílem na trhu. Ostatní operační systémy mají minoritní podíl na trhu – dvě procenta a méně.

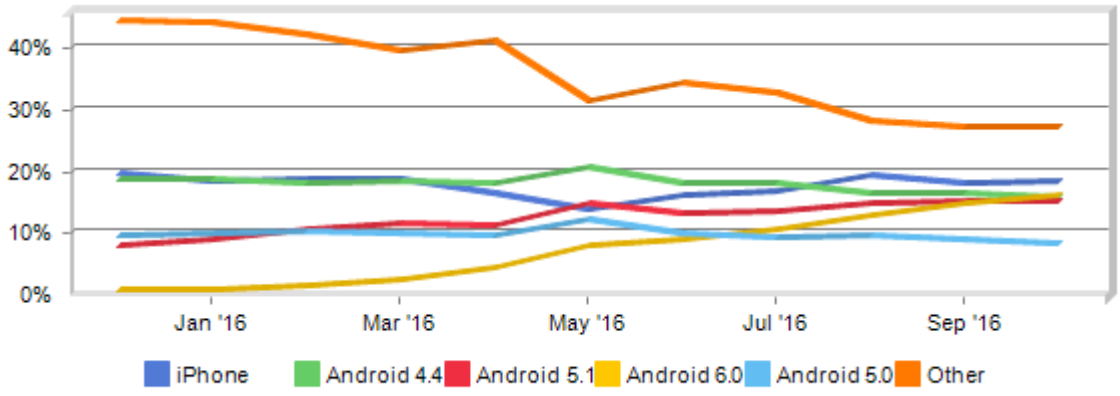
Pokud chceme, aby vyvíjená mobilní aplikace mohla být používána na většině zařízeních na trhu, pak musíme počítat se zastoupením minimálně dvou operačních systémů a nelze určit, zda se v budoucnu neprosadí ještě další operační systémy. Kandidáty jsou zatím Windows Phone a Java ME s podílem dvou procent z celkového trhu.



Obrázek 2:
Mobile/Tablet Operating System Market Share[4] [10]

Podíl verzí operačních systémů chytrých telefonů na trhu

Není důležité znát pouze podíl operačních systémů na trhu, ale také jejich verze, protože i to může ovlivnit v jakém frameworku nebo programu mobilní aplikaci vytvoříme. Na obrázku 3 vidíme pět nejčastěji používaných verzí operačních systémů v roce 2016 od ledna do května. Největší podíl má suverénně skupina ostatní, ve které se skrývají všechny verze operačních systémů, které na grafu nejsou zobrazeny. Z tohoto faktu můžeme usoudit, že podíl verzí operačních systémů na trhu je značně různorodý. Ostatně i u zbývajících verzí operačních systémů v tomto půlročním intervalu není jasné, která je nebo bude dominantní.



Obrázek 3:
Mobile/Tablet Top Operating System Share Trend [11]

Souhrn

Z výše uvedených studií a grafů je patrné, že trend chytrých mobilních telefonů narůstá a není zcela zřejmé, které operační systémy a jejich verze budou na trhu domino-

[4] *Market share* neboli podíl na trhu reprezentuje procento z celkových prodejů v dané oblasti za určitý čas [autor]

vat. Spíše to vypadá, že se bude trh s chytrými mobilními telefony bude ubírat cestou vytváření nových verzí operačních systémů a staré verze budou postupně zanikat. To pro vývojáře mobilních aplikací znamená, že musí podporovat širokou škálu verzí operačních systémů. Práce se zabývá řešením frameworkem IONIC, které by mohlo pomoci vyvíjet aplikace na sjednoceném kódu. Kód se následně jde spustit na různých operačních systémech a jejich verzích.

3.2 Vymezení pojmů vývoje aplikací pro více operačních systémů

Tato část se věnuje vymezením a popsáním pojmů nativní, multiplatformní a webová aplikace. Pojmy se mohou používat i v dalších odvětvích, ale práce se zaměřuje na vývoj mobilních aplikací, takže je téma rozebráno výhradně z tohoto pohledu. Použité zdroje v této části byly: článek *Native, HTML5, or Hybrid: Understanding Your Mobile Application Deve-lopment Options* [33] a článek z oficiálního blogu Ionic – *Where does the Ionic Framework fit in?* [34]

Vymezení pojmu multiplatformní aplikace

Multiplatformní aplikace je počítačový program, operační systém či výstup programovacího jazyku, který může běžet na více počítačových platformách. Pojem ovšem neznamená, že aplikace musí vždy běžet na všech dostupných platformách, ale znamená, že bez použití úpravy kódu je program spustitelný minimálně na dvou platformách. Pojem platforma znamená kombinace operačního systému s konkrétním hardwarovým vybavení. Například nainstalovaný operační systém Microsoft Windows na hardwarové architektuře x86. Je možné, že ke správné funkci multiplatformní aplikace je potřeba doinstalovat na operační systém program, který zabezpečuje správný chod aplikace.

Příklad: Program napsaný v jazyce Java

Java je multiplatformní programovací jazyk. Jeho výstupy lze spustit na operačních systémech na kterých je možné nainstalovat prostředí Java Virtual Machine (JVM). Úkolem modulu JVM je zpracovat Java bytecode, který je uložený v souborech typu *.jar a *.class. Soubory *jar a *.class jsou vygenerované prostředím, kde se aplikace psaná jazyce Java vyvíjí. JVM lze nainstalovat na mnoho operačních systémů a prostředí. Např. na Microsoft Windows, MAC OS, Linux a další...

Příklad: Program napsaný s použitím frameworku Cordova

Cordova umožňuje zabalit webovou aplikaci do různých prostředí mobilních operačních systémů tak, aby se pro mobilní telefon jevila jako nativní. Zároveň vystaví aplikaci API, která jí umožní přístup k nativním funkcím mobilního operačního systému, takže s nimi může aplikace pracovat. Vyvíjí se tedy jeden stejný kód, který je po „zabalení“ spustitelný na více mobilních operačních systémech

Výhody mobilní multiplatformní aplikace

- Kód programu se napíše jednou a lze spouštět na více operačních systémech.
- Program může pracovat s nativními funkci operačního systému.
- Vývojáři mohou pracovat v jednom programovacím kódu, který znají a nemusejí se učit nativní programovací jazyky operačních systémů.
- Je možné použít stejné prostředí pro vývoj, jako pro webové stránky.

Nevýhody mobilní multiplatformní aplikace

- Přístup k nativním funkcím operačního systému může být omezené nebo může být potřeba doprogramovat plugin pro přístup.
- Programátor musí vytvořit všechny UI komponenty, protože nemá přístup k nativním komponentám.
- Aplikace využívá nativní internetový prohlížeč, proto je výkon aplikace závislý na kvalitě internetového prohlížeče.

Vymezení pojmu mobilní webové stránky

Webové stránky se píší programovacími jazyky a technologiemi, které podporují internetové prohlížeče nainstalované na různých operačních systémech. Tyto stránky jsou zobrazeny stejně nebo podobně. Důležité je, že webové stránky jsou psané pro všechny operační systémy, potažmo jejich prohlížeče, stejně. Technologie využívané pro jejich psaní a používání jsou například: HTLM, CSS, Ajax, XML, HTTP, SOAP, WSDL a mnoho dalších. Existují frameworky, kterými lze optimalizovat grafické rozložení komponent na stránce. Stránky tak mohou být přizpůsobené na daný display zobrazovacího zařízení platformy. Tento způsob umožňuje lépe přizpůsobit stránku pro obrazovku počítače, telefonu nebo tabletu. Stránky se při použití této technologie nazývají responzivní.

Výhody

- Dobře se udržují, protože není nutná instalace ani aktualizace.
- Není nutná instalace, protože se pokaždé načítají ze serveru.
- Jsou přístupná ze všech zařízení, které mají internetový prohlížeč.

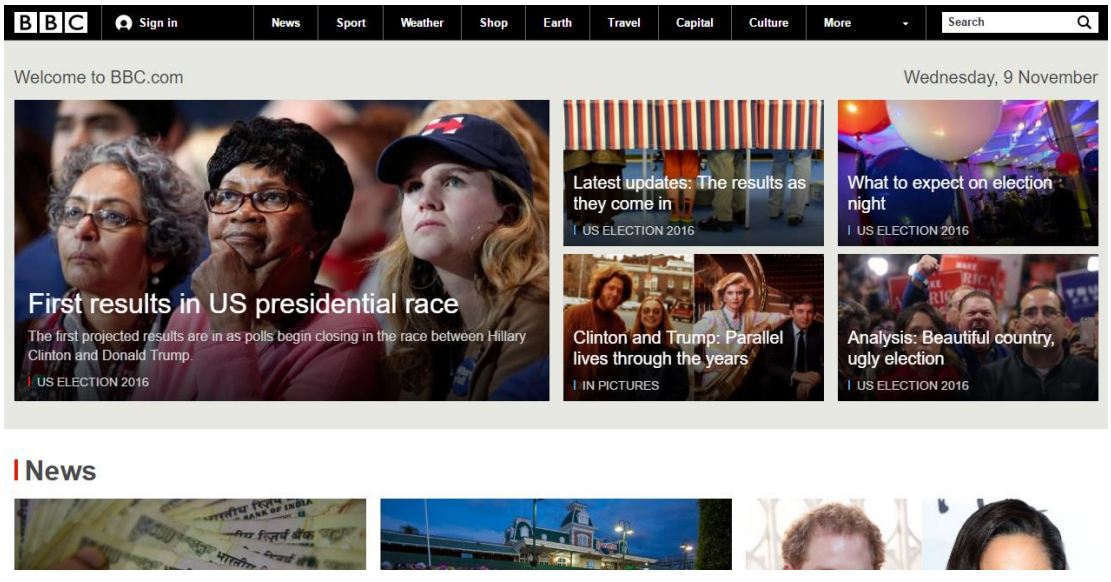
Nevýhody

- Nemají přístup k nativním funkcím operačního systému.
- Při zatížení serveru nebo špatném přístupu k internetu se aplikace déle načítá nebo nenačte vůbec.
- Je nutné zadat URL adresu, což je na mobilním telefonu komplikovanější než kliknout na ikonu aplikace.
- Jsou dostupné pouze omezené prostředky a nástroje na zobrazování ve webovém prohlížeči.

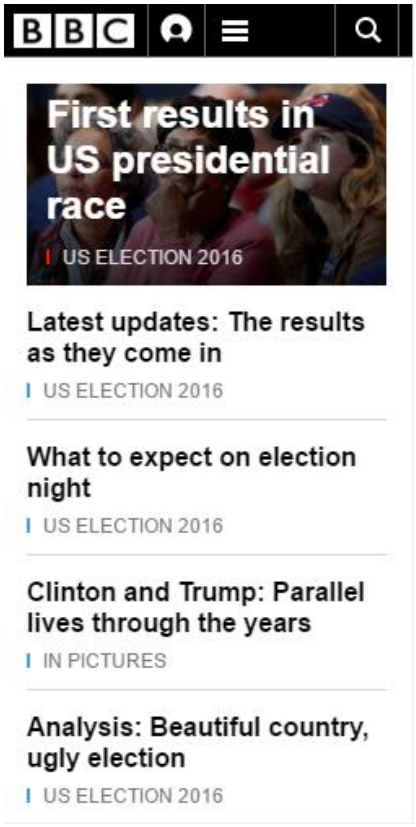
Ukázka responzivního zobrazení webové stránky BBC

Zdroj: http://www.bbc.com/

Na obrázku 4 a 5 lze vidět rozdíly mezi zobrazením stejné stránky na různých zařízeních s různými rozlišeními zobrazovacího displeje. Stránka je zobrazena pomocí internetových prohlížečů nainstalovaných na jednotlivých zařízeních. Jedná se o stejný zdrojový kód.



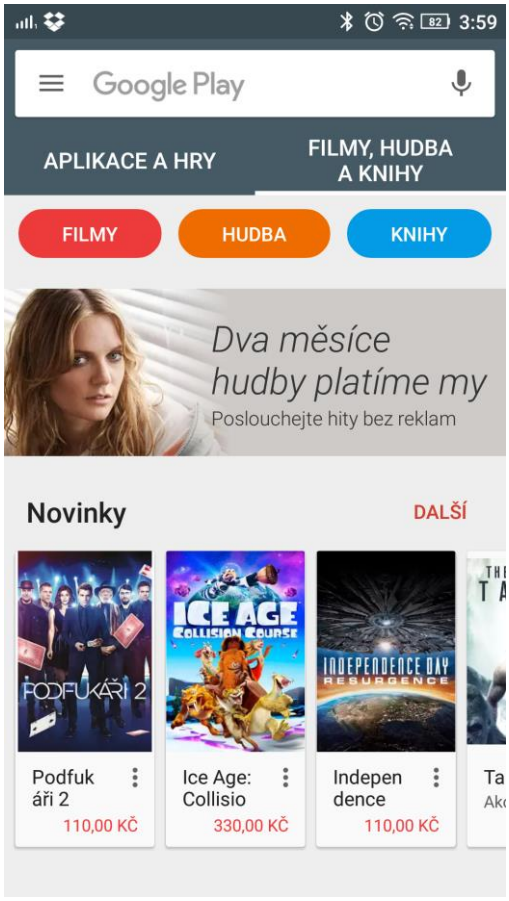
Obrázek 4:
Ukázka zobrazení webové stránky BBC na velkém rozlišení – např. stolní počítač nebo notebook [35]



Obrázek 5:
Zobrazení na menším rozlišení – např. mobilní telefon [35]

Vymezení pojmu nativní aplikace

Nativní aplikace jsou programy vyvíjené pro konkrétní operační systém, platformu a konkrétní hardware. Jsou rychlé, spolehlivé a umí využívat hardwarových schopností daného zařízení. Díky tomu že se nativní aplikace vyvíjecí v programovacím jazyce, který je určený pro konkrétní platformu, tak tyto aplikace mohou využívat všechny služby, které jsou systémem dostupné. Vede to také k lepší správě a využití systémových a hardwarových prostředků. Příklad nativní mobilní aplikace je zobrazen na obrázku 6. Program je integrovaný do systému, využívá jeho prostředky jako např.: push notifikace, ikona, jeho úložiště, umožňuje instalaci ostatních aplikací a další…



Obrázek 6:
Nativní aplikace Androidu Google Play [Dostupné z: https://play.google.com]

Výhody

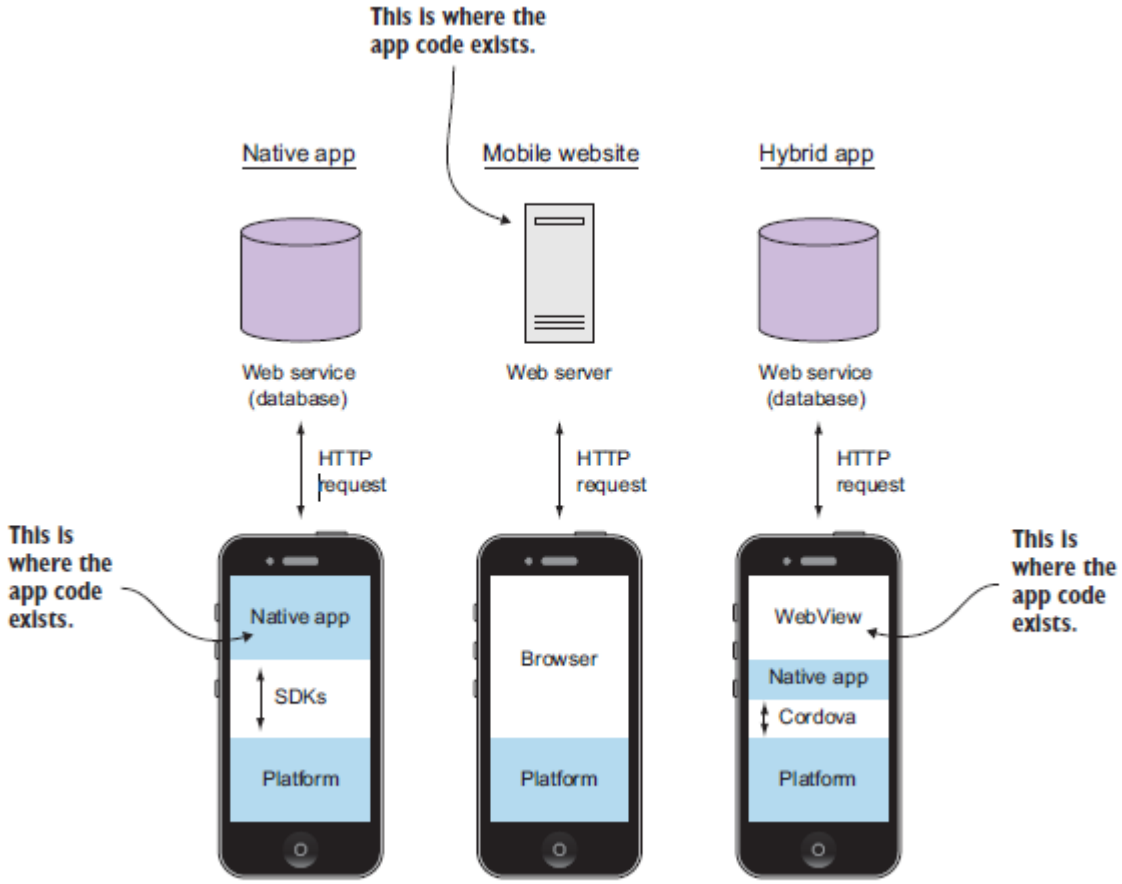
- Nativní aplikace vyžívá napřímo API operačního systému, takže je úzce spojená s platformou.
- Využívá maximálně zdroje operačního systému, protože tam není žádný mezičlánek.
- Vždy je dostupná aktuální verze API operačního systému, takže umožňuje využívat při vývoji ihned nejnovější funkce.

Nevýhody

- Jazyk, ve kterém se aplikace píše je spjatý platformou, pro kterou se aplikace vyvíjí. Programátor tedy musí umět specifický programovací jazyk.
- Aplikace je možné vyvíjet v jednom čase pouze pro jednu platformu.
- Aplikace vyžaduje více času pro vývoj, takže se musí počítat s vyššími náklady.

Porovnání architektur

Na obrázku 7 lze vidět porovnání architektur různě vytvořených aplikací pro mobilní telefony. V případě nativní aplikace se používá pro komunikaci rovnou SDK[5] dané platformy. Mobilní webová aplikace používá internetový prohlížeč dané platformy a u multiplatformní aplikace se používá malá instance internetového prohlížeče, která je úzce svázaná s vygenerovanou nativní aplikací a Apache Cordovou [31], která zprostředkovává komunikaci přes její API s danou platformou.



Obrázek 7:
Porovnání architektury webové, nativní a multiplatformní mobilní aplikace [13 str.4]

Představení frameworku Ionic

Framework IONIC byl chybějící článek u multiplatformních mobilních aplikací, protože jednou z jeho hlavních předností je zprostředkování prvků UI, které se chovají a vypadají podobně jako prvky z nativních mobilních aplikací. Tento framework tedy umož-

[5] SDK (software development kit) – je soubor knihoven pro určité prostředí nutných na vývoj softwaru [autor]

ňuje připravit podobné UI, jako při vývoji nativní aplikace. Jedná se třeba o komponenty postranního menu, spodních tabů, tlačítek, předpřipravených stylů a ikon a dalších prvků. Proto framework Ionic na svém blogu [34] označují jako framework pro vývoj nativních multiplatformních aplikací. Ionic spojuje významné technologie pro vývoj webových stránek. Umožňuje používat pro vývoj CSS, HTML, and JavaScript.

Výhody frameworku Ionic

Ionic umožňuje pomocí jeho předpřipravených prvků a používaných frameworků vytvořit aplikace, která jsou velmi podobná nativním aplikacím. Jak chováním, tak rychlostí. Plní očekávání uživatelů aplikací, kteří ji chtějí mít rychlou a intuitivní. Díky předpřipraveným komponentám lze vytvořit aplikaci rychle a s řadou funkcionalit. Framework Ionic má následující vlastnosti [13]:

- Umožňuje vytvoření mobilní aplikace, která se chová jako nativní aplikace pomocí prvků webových stránek HTML, CSS, and JavaScript.
- Funkcionalita frameworku Ionic se vytváří pomocí frameworku AngularJS, který je společný i pro webové aplikace. AngularJS obsahuje množství připravené funkcionality, která zjednodušuje práci a umožňuje používat moduly třetích stran. AngularJS je používán pro vývoj webových aplikací framework Ionic ho upravuje pro použití v mobilních aplikacích.
- Využívá kaskádové styly CSS3, které upravují zobrazování ve webovém prohlížeči. Umožňují takové věci jako stínování, animace, zaoblené rohy, přechody a další.
- Rychlost a výkon aplikace napsané v IONIC je srovnatelná s nativní aplikací.
- Kvůli připraveným grafickým komponentám vypadají aplikace jako nativní.
- Protože se Ionic hojně využívá, tak jeho vývojový tým neustále zlepšuje jeho funkcionalitu. Napovídá tomu i fakt, že vyjde Ionic 2, který je v době psaní této práce ve stavu bety.
- Protože je Ionic open source a je hojně používaný, tak má rozsáhlou základnu uživatelů. Ti vytvářejí různé návody a diskutují na internetu, takže tímto způsobem lze získat mnoho rad.
- IONIC poskytuje servis, kterým se aplikace vyvíjí pohodlněji. Jedna z takových funkcí je IONIC Creator [3], který umožňuje vytvoření uživatelského rozhraní za pomocí funkcí drag-and-drop. Tímto nástrojem lze exportovat základ aplikace pro vývoj nebo vytvořit aplikaci ve verzi beta, která je vhodná k testování uživatelského rozhraní. Podobné funkce zlepšují efektivitu vývoje mobilních aplikací.

4 Framework IONIC a s ním spojené technologie

Tato kapitola obsahuje obecný popis frameworku IONIC. V první části jsou popsány používané technologie s frameworkem. Dále je zde popsána instalace, struktura projektu a způsoby testování.

4.1 Technologie využívané s frameworkem Ionic

Ionic je občas označován jako framework frameworků a má to svůj dobrý důvod. Sám o sobě zajišťuje pouze zlomek funkcionality, a ostatní funkce deleguje na jiné frameworky, které využívá. V této části jsou tyto technologie vysvětlené.

AngularJS

AngularJS je framework webových aplikací postavený na technologii Javascript. Je jedním z nejpoužívanějších frameworků využívající Javascript. Javascript je skriptovací jazyk umožňující změnu stránky nebo pouze části stránky bez načtení nové URL. Tato část čerpá informace z knih *AngularJS in action* [23] a *Beginning AngularJS* [24].

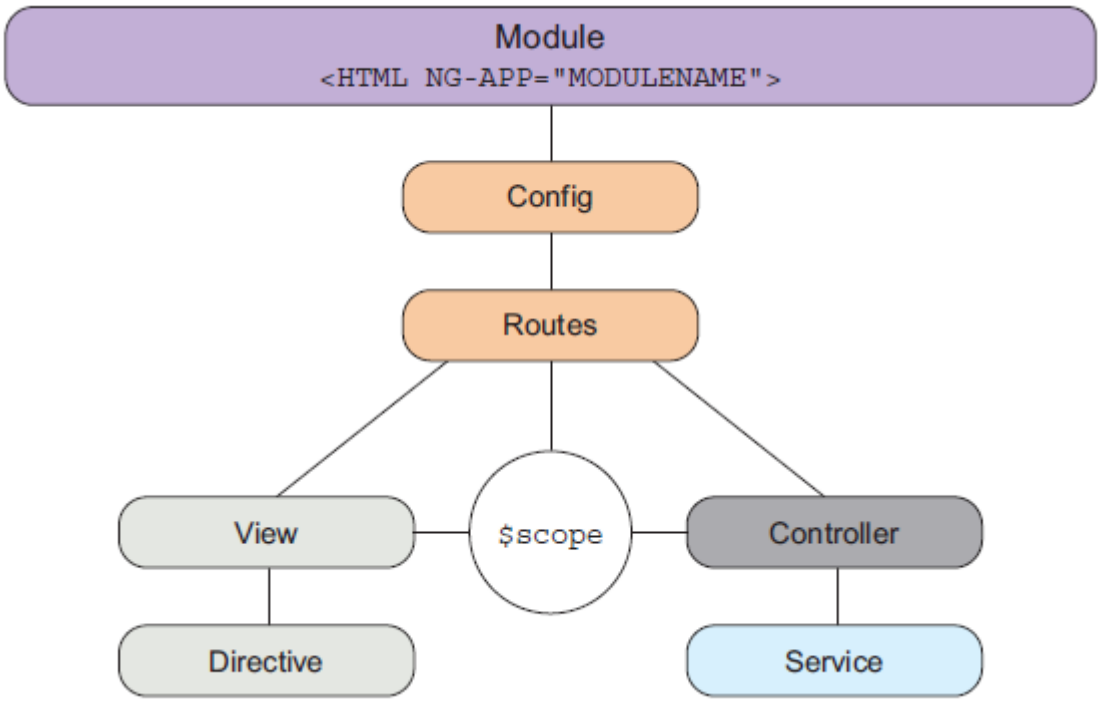
Výhody a vlastnosti

- Umožňuje organizovat kód aplikace tak, aby byl udržitelný, čitelný, bylo na něm možné lépe pracovat v týmu a snadno ho rozšiřovat. Využívá k tomu návrhový vzor model-view-controller.
- Kód napsaný ve frameworku AngularJS je testovatelný.
- Nemusí se definovat jQuery k vyhledání elementu na stránce, vytvářet lisenery pro události a parsovat hodnoty. Tento způsob se využívá u jQuery aplikací. Ve frameworku AngularJS se jednoduše definuje Javascript proměnná a naváže se na HTML tagy. S proměnnou lze dále pracovat v částech aplikace, které zajišťují interaktivní funkcionalitu. Tím odpadá množství kódu pro zprostředkování přenesení hodnoty.
- K vytvoření obsahu webové stránky využívá jazyk HTML, který je obecně známý. Tento jazyk umožňuje vytvoření datové struktury webové stránky a AngularJS k němu přidává způsob, jak udělat tento obsah interaktivní.

Architektura

Na obrázku 8 vidíme architekturu aplikací, které jsou psané ve frameworku AngularJS. Z obrázku je zřejmé, že je zde využíváno známého návrhového vzoru model-view-controller. Návrhový vzor umožňuje psát přehlednější kód. V horní části obrázku je vyznačen modul. Modul slouží jako kontejner pro části ucelené funkcionality, které spolu souvisí. Dobré rozdělení do modulů umožňuje znovupoužití kódu a jde z nich poskládat základ pro novou aplikaci. Konfigurační blok frameworku AngularJS je určen pro zapsání funkcionality, která se provede před samotným spuštění aplikace. V této části je možné konfigurovat dynamické servisy, nastavovat cesty apod. Soubory Routes nastavují cesty. Ze souborů routes se aplikace dozví URL cesty a přesměrování při různých stavech aplikace. Také se zde zapisuje, jaké kontrolery a servisy se mají použít pro jakou stránku. Část View se vytvoří poté co se zkompiluje a přeloží DOM[6], ve kterém je již přidaná Javascript funkcionalita. View slouží k zobrazování prvků UI. $scope se stará o přenos informací mezi views a kontrolery. V kontroleru se definují metody a proměnné, které se využívají v časti views pro zobrazování. Mělo by tam být co nejméně kódu a starat se pouze o stránky, ke kterým patří. Direktivy jsou rozšíření frameworku AngularJS umožňující vytvoření opakovatelně používaných prvků ve views. Jedna část kódu se tedy tímto způsobem dá použít víckrát napříč aplikací ve views. Jako poslední je část servis, zajišťující společnou funkcionalitu. Na příklad, když jsou v aplikaci data, která se používají ve více kontrolerech, tak je dobré k datům přistupovat pomocí kódu, který je napsaný zvlášť.

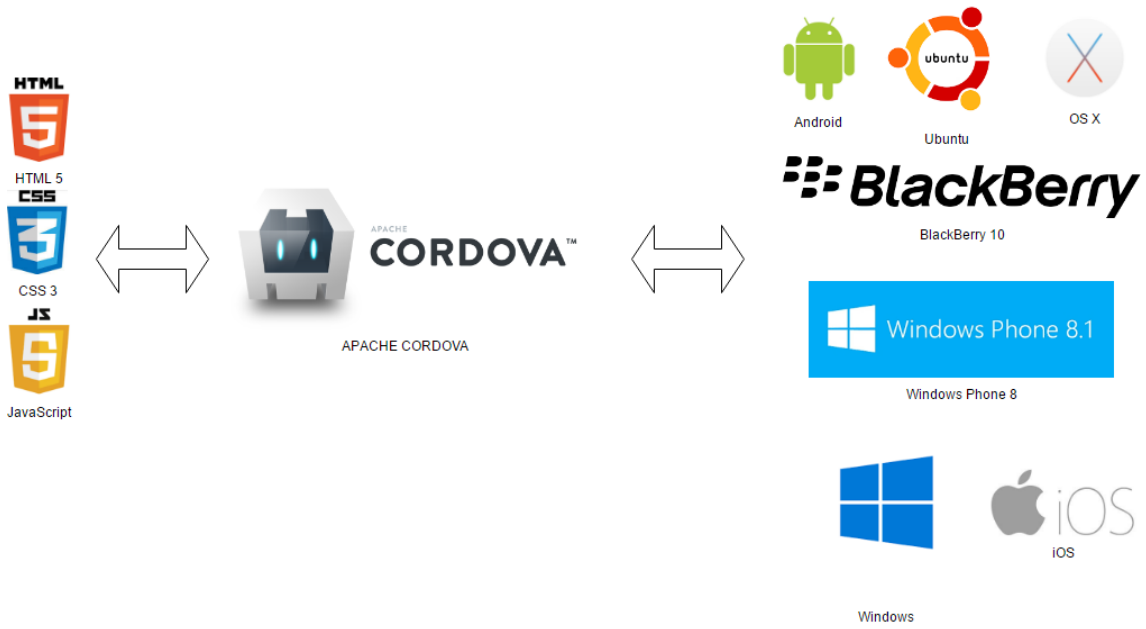
[6] DOM: Document Object Model – objektový model dokumentu je objektově orientovaná reprezentace XML nebo HTML dokumentu.



Obrázek 8:
Architektura frameworku AngularJS [13]

Apache Cordova

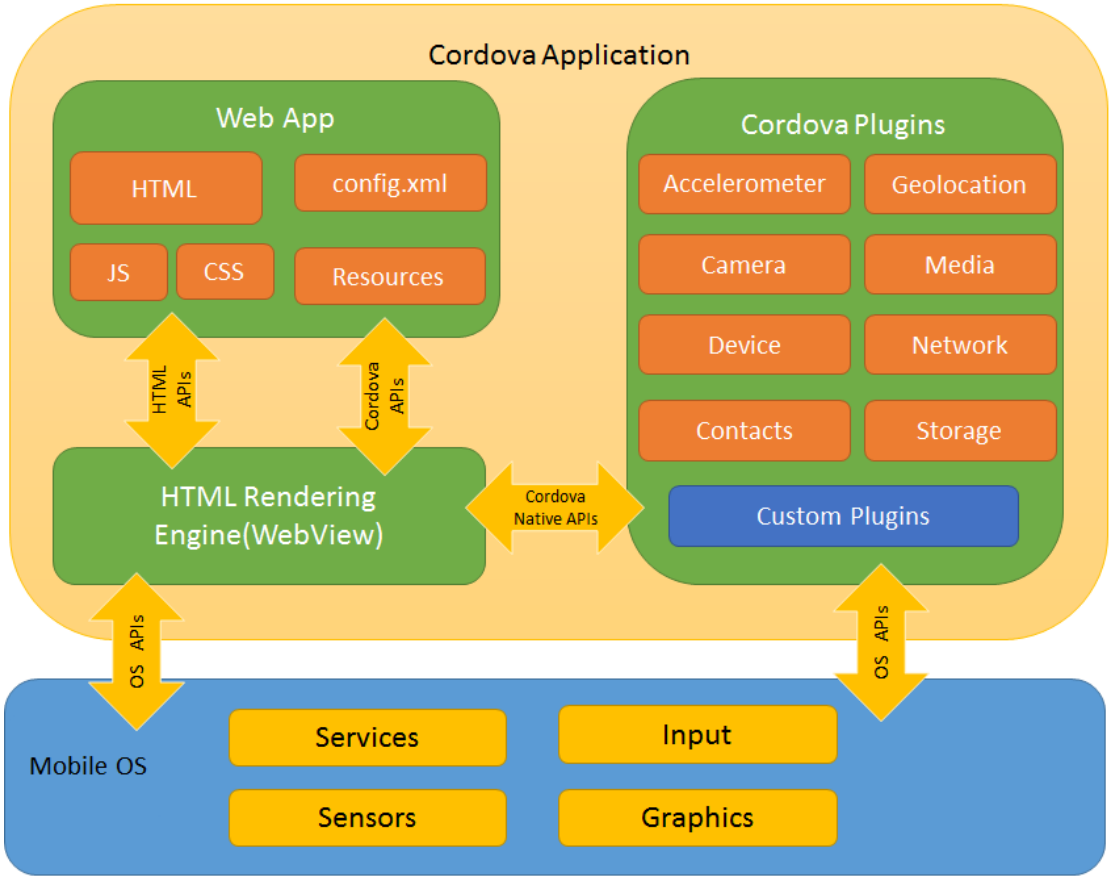
Apache Cordova je open source framework pro vývoj mobilních aplikací. Umožňuje využívat standartní webové technologie: HTML5, CSS3 a JavaScript. Apache Cordova vnoří vyvinutou aplikaci do obalu (wrapper), tak aby aplikace vypadala, že byla vyvíjená pro konkrétní operační systém. Obal komunikuje se specifickým rozhraním operačního systému a pro aplikaci vystavuje vždy stejné API, takže je vyvinutý jeden kód a následně lze pomocí frameworku Apache Cordova převést na aplikaci specifickou pro danou platformu. Použité technologie a operační systémy lze vidět na obrázku 9. Popisuji verzi Apache Cordova 6.X .



Obrázek 9:
Apache Cordova: frameworky a operační systémy [autor]

Architektura

Na obrázku 10 jsou zobrazeny moduly Cordova aplikace. Kód vyvinuté aplikace je v komponentě Web App. Je složena z částí HTML, CSS, Javascriptu, konfiguračního souboru aplikace a ostatních zdrojů. Aplikace komunikuje s komponentou WebView, která pracuje s nativním minimalistickým webovým prohlížečem dané platformy a zprostředkovává potřebné API aplikaci. WebView také komunikuje s částí Cordova pluginů, které zprostředkovávají informace z nativních funkcí operačního systému. Mimo výchozích pluginů Cordovi lze doprogramovat specifické pluginy pro různé platformy. Tento způsob umožňuje využít nativní funkcionality, která ještě není ve frameworku Apache Cordova zahrnuta.



Obrázek 10:
Architektura Cordova aplikací [4]

Cordova pluginy

V tabulce 1 vidíme přehled základních pluginů frameworku Apache Cordova. Tyto pluginy se nainstalují s instalací frameworku. V případě, že je potřeba do aplikace i další nativní funkcionalita, která zde není uvedena, tak na oficiální stránce Apache Cordova [4] je záložka Plugins, kde je možné stáhnout další pluginy. Mezi pluginy tam lze vyhledávat fulltextovým polem a vybráním příslušných operačních systému, které chceme, aby plugin podporoval. Pokud ještě plugin není vyvinutý nebo dostupný, je v dokumentaci popis, jakým způsobem lze vytvořit plugin vlastní.

Tabulka 1: Přehled základních pluginů Apache Cordova

Název pluginu	**Popis**
Battery Status	Plugin vystavuje API, které umožňuje sledovat stav baterie a nabíječky telefonu.
Camera	Plugin vystavuje API, které umožňuje pořizovat obrázky fotoaparátem nebo vybírat obrázky z knihovny operačního systému.

Console	Plugin vystavuje API, které umožňuje vypisovat hlášky v konzoly operačního systému.
Contacts	Plugin vystavuje API, které umožňuje číst v databázi kontaktů zařízení. Uživatel aplikace musí potvrdit přístup. Přístup ke kontaktům podléhá zákonu o ochraně osobních údajů.
Device	Plugin vystavuje API, které umožňuje získat informace o hardwaru a softwaru zařízení.
Device Motion	Plugin vystavuje API, které umožňuje získat informace z akcelerometru zařízení. Vrací souřadnice polohy x, y a z.
Device Orientation	Plugin vystavuje API, které umožňuje získat informace z kompasu zařízení. Vrací směr, jakým je zařízení natočeno: 0-359,99 stupňů. Nultý stupeň je sever.
Dialogs	Plugin vystavuje API, které umožňuje přístup k některým z nativních dialogů UI. Jsou to například dialogové zprávy, potvrzovací dialogy, pípání..
File	Plugin vystavuje API, které umožňuje číst a zapisovat do souborů. Každý operační systém má jinou stromovou strukturu složek.
File Transfer	Plugin vystavuje API, které umožňuje posílat a stahovat soubory.
Geolocation	Plugin vystavuje API, které umožňuje získat polohu zařízení. API nezaručuje, že poloha, kterou vrátí je pravdivá. Poloha se získává z GPS signálu, ip adresy, GSM/CDMA, RFID nebo MAC adresy přes WiFi a Bluetooth. Poloha podléhá zákonu o ochraně osobních údajů, proto uživatel aplikace musí potvrdit přístup.
Globalization	Plugin vystavuje API, které umožňuje získat zemi používání telefonu, jazyk a časovou zónu.
Inappbrowser	Plugin vystavuje API, které umožňuje zobrazit články, videa a jiné zdroje z internetu přímo v aplikaci.
Media	Plugin vystavuje API, které umožňuje nahrávat a přehrát zvuk na zařízení.
Media Capture	Plugin vystavuje API, které umožňuje nahrávání audia, videa a focení. Zachytávání obrazu podléhá zákonu o ochraně osobních údajů, proto uživatel aplikace musí potvrdit přístup.

Network Information	Plugin vystavuje API, které umožňuje sledovat, zda má zařízení přístup k internetu a přes jaký zdroj.
Splashscreen	Plugin vystavuje API, které umožňuje nastavení uvítací obrazovky aplikace.
Vibration	Plugin vystavuje API, které umožňuje používání vibrací telefonu. Nastavuje se délka trvání vibrace v ms.
Statusbar	Plugin vystavuje API, které umožňuje přizpůsobení vrchního stavového řádku na zařízeních Android a iOS.
Whitelist	Plugin podporuje nastavení konfiguračního souboru, který určuje, na které webové stránky má uživatel přístup v aplikaci.

Ostatní technologie

V této části jsou popsány další nástroje a technologie, které framework IONIC využívá.

Bower

Bower [26] je nástroj optimalizovaný pro frontend vývoj aplikací. Stará se o instalaci a správu balíků a závislostí v projektu. Jedním příkazem lze stáhnout všechny závislosti v projektu. Frontend vývoj je založený na skládání a znovu používání komponent a jejich provázanost a závislosti nemusí být vždy jednoduché. Díky Boweru může být rychle připraveno prostředí pro nového člena vývojového týmu.

Node.js

Nástroj Node.js [27] umožňuje spuštění Javascript kódu mimo okno prohlížeče. To do aplikace přidává možnost asynchronních http requestů. Dokonce pomocí tohoto nástroje je možné vytvořit backend aplikaci. Tedy aplikaci, která nezobrazuje uživatelské rozhraní, ale je schopna zpracovávat data.

Sass

Saas (Syntactically Awesome StyleSheets) [29] je framework, který rozšiřuje funkcionalitu kaskádových stylů. Je to nástroj, pomocí kterého lze efektivně upravovat způsob zobrazování uživatelského rozhraní aplikace. Výhody jsou například podpora proměnných, funkce pro změnu barev a proměnných, podpora directive.

Gulp

Gulp [30] je nástroj starající se o prekompilaci kódu. Umožňuje složit kód z více souborů do jednoho a jeho minimalizaci. Tento přístup zrychluje načítání a minimalizuje počet importovaných souborů. Gulp také umožnuje sledovat změny v projektu a ihned na ně reagovat. Proto je možné vidět po změně kódu okamžitě výsledek bez nutného restartu aplikace.

Git

Git [28] je nástroj na verzování souborů projektu. Umožňuje kooperaci týmu při vývoji. Stará se také o zálohování souborů. V případě, že dva programátoři pracují na jednom souboru, tak git zprostředkovává způsob výběru změn, které na něm provedli a uživatele upozorní na konflikt. Umožňuje také rozvětvit vývoj do dílčích celků, které se navzájem nemohou ovlivnit.

4.2 Instalace, nastavení vývojového prostředí

Protože IONIC využívá hodně dalších technologií, tak instalace a nastavení prostředí není jednoduchá. Proto se tato část věnuje postupem, jak nainstalovat všechny potřebné programy, aby šel framework využívat.

Postup instalace:

1. **Instalace NODE.JS**

Node.js je knihovna open source projektů. Instalátor je na internetových stránkách [27]. Z knihovny lze tyto projekty instalovat na lokální disk pomocí příkazové řádky.

2. **npm install –g cordova**

Příkaz uvedený výše po zadání do příkazového řádku stáhne a nainstaluje open source framework Apache Cordova [31], umožňující vyvíjet mobilní aplikace pro různé operační systémy v jednom vytvořeném kódu. Přepínač „-g“ znamená „global“. Instalace se provede pro celé prostředí operačního systému. Ověření správné instalace lze provést zadním příkazu „cordova –v“ do příkazového řádku. V případě, že je program právně nainstalovaný se vypíše jeho aktuální verze.

3. **npm install -g bower**

Příkaz uvedený výše po zadání do příkazového řádku stáhne a nainstaluje program Bower [26], který slouží k řízení balíčků a závislostí internetových aplikací. Přepínač „-g“ znamená „global“. Instalace se provede pro celé prostředí operačního systému. Ověření správné instalace lze provést zadním příkazu „bower –v“ do příkazového řádku. V případě, že je program právně nainstalovaný se vypíše jeho aktuální verze.

4. **npm install -g gulp**

Příkaz uvedený výše po zadání do příkazového řádku stáhne a nainstaluje open source framework Gulp [30]. Gulp je framework, který pomáhá při webovém vývoji. Umožňuje např. používat preprocesory zdrojového kódu nebo živé načítání webové stránky v případě uložení zdrojové stránky. Přepínač „-g“ znamená „global“. Instalace se provede pro celé prostředí operačního systému. Ověření správné instalace lze provést zadním příkazu „gulp –v“ do příkazového řádku. V případě, že je program právně nainstalovaný se vypíše jeho aktuální verze.

5. **npm install -g ionic@1.7.16**

Příkaz uvedený výše po zadání do příkazového řádku stáhne a nainstaluje open source framework IONIC, který slučuje frameworky používané pro vývoj multiplatformních mobilních aplikací a umožňuje využívat prvky, které jsou velice podobné prvkům nativních mobilních aplikací. Přepínač „-g“ znamená „global“. Instalace se provede pro celé prostředí operačního systému a nainstaluje IONIC verze 1.7.16, kterou využíváme v této diplomové práci. Ověření správné instalace lze provést zadním příkazu „ionic –v“ do příkazového řádku. V případě, že je program právně nainstalovaný se vypíše jeho aktuální verze.

6. **Instalace GITu**

Git je open source na verzování a sdílení kódu. Instaluje se instalačním balíčkem dostupným na oficiální internetové stránce Gitu [28].

7. **npm install**

Příkaz uvedený výše po zadání do příkazového řádku stáhne a nainstaluje závislosti konkrétního projektu. Příkaz se musí spustit v kořenovém adresáři projektu.

8. **bower install**

Příkaz uvedený výše po zadání do příkazového řádku stáhne a nainstaluje závislosti ze souboru bower.json konkrétního projektu. Příkaz se musí spustit v kořenovém adresáři projektu. Doinstalují se jím závislosti z použitých bower knihoven.

4.3 Inicializace projektu

IONIC umožňuje inicializovat projekt příkazovou řádkou Ionic CLI. Příkaz vytvoří strukturu projektu a založí všechny potřebné soubory, aby aplikaci rovnou šlo spustit ve webovém prohlížeči.

Příkaz pro inicializaci projektu: *ionic start [directoryName] [template]*

Místo *[directoryName]* uživatel vloží název adresáře, ve kterém chce projekt vytvořit a místo *[template]* vloží název šablony.

Šablony lze využít 3:

- tabs – základní šablona pro vytvoření 3 záložek
- sidemenu – šablona pro vytvoření postranního menu
- blank – šablona, která nemá žádné view

4.4 Testování

Testovaní je nedílnou součástí vývoje softwaru. V této kapitole jsou uvedeny způsoby testování, popsány jejich výhody, nevýhody a vhodnost kdy které testy aplikovat.

Testování prohlížečem stolního počítače

Příkaz „ionic serve“ je nejrychlejší způsob testování a náhledu vyvíjené aplikace. Stačí využít Ionic CLI a v kořenovém adresáři projektu ho spustit. Po malé chvilce se otevře okno webového prohlížeče, kde aplikace vypadá, jako by běžela na mobilním telefonu. Je to živý náhled s celou funkcionalitou mimo nativní funkce. Pokud se někde v projektu používá knihovna nebo plugin Cordovi, tak v tomto náhledu nebude fungovat. Tento způsob je doporučeným způsobem testování na oficiálních stránkách. Je možné, že se zobrazování na jednotlivých platformách bude mírně odlišovat, nicméně funkcionalita je stejná. Příkaz „ionic serve“ je vhodný používat hlavně během vývoje, kde je možné vidět okamžitý náhled a funkci napsaného kódu. Nevýhodou je také to, že si uživatel/tester nevyzkouší ovládání aplikace na mobilním zařízení. Některé hodnoty nebo formuláře se ve webovém prohlížeči na desktopovém počítači mohou vyplňovat daleko lépe než následně na mobilním zařízení. V prohlížeči je možné sledovat chybové hlášky, provoz sítě, requesty na server, uložené proměnné. Je to hlavní nástroj, přes který lze debugovat.

Testování vzdáleným přístupem

Chrome umožňuje přeposílání lokálního portu do mobilního zařízení prostřednictvím USB kabelu. Je to způsob testování, který nevyžaduje moc velkou přípravu, ale uživatel/vývojář si může vyzkoušet aplikaci na skutečné obrazovce mobilního zařízení. Je potřeba na telefonu a zapnout debugging. Postup je popsán níže

Postup spuštění vzdáleného přístupu:

1. Nastavit na mobilním telefonu mód pro vývojáře.
2. Spustit debugging na mobilním zařízení.
3. Připojit telefon k PC.
4. Nainstalovat ovladače mobilního telefonu do PC.
5. Zapnout IONIC aplikaci příkazem „ionic serve“.
6. V prohlížeči Chrome zapnout vývojářskou konzoli (F12).
7. Rozkliknout menu/Další nástroje/Vzdálená zařízení.
8. Nastavit přesměrování portu do mobilního zařízení.
9. Otevřít webový prohlížeč Chrome v telefonu.
10. Otevřít odkaz localhost:[port] na mobilním zařízení v Chromu.

Tímto způsobem lze přesměrovat spuštěnou lokální aplikaci do webového prohlížeče v telefonu a vyzkoušet si vzhled a přívětivost ovládání prstem.

Testování v aplikaci IONIC view

Aplikace *IONIC view* je volně dostupná v internetových obchodech Google Play a Apple Store. Oficiální stránka je zde: http://view.ionic.io/. IONIC view umožňuje nahrát testované aplikace a spustit je v nativním módu, takže se mohou projevit chyby, které nebylo možné předchozími způsoby detekovat. Tento způsob testování je vhodný uskutečnit před uvedením aplikace do internetových obchodů. Testované aplikace lze distribuovat do ostatních mobilních telefonů s nainstalovaným IONIC view sdílením klíče k aplikaci. Metoda lze použít k testování nebo ukázce aplikace.

Testování pomocí emulátoru

K tomuto druhu testování je třeba doplnit software development kit, který je specifický pro danou platformu. iOS nelze testovat na OS Windows. Emulátory jsou většinou pomalé. Trvá dlouhou dobu, než se nastartují a neumožňují jednoduše otestovat nativní funkce telefonu. Na stránkách frameworku IONIC je uvedeno, že není doporučeno používat Android SDK Emulátor. Existují také alternativní nástroje pro emulaci mobilních operačních systémů.

Testování jako nativní aplikaci

Přestože je vývoj aplikace prováděn nad jedním kódem je více než potřebné testovat aplikaci před distribucí mezi uživatele na jednotlivých platformách. Pro iOS je pro deploy mobilní aplikace nutné si založit zpoplatněný developerský účet za 99 dolarů ročně a mít notebook s iOS. Pro Android je nutné si naistalovat Android studio včetně Android SDK. Poté stačí napsat „*ionic run android*“ a na připojené zařízení k počítači se nahraje a spustí aplikace. Mobilní zařízení musí mít aktivovaný debugging.

Jednotkové testy

Z dlouhodobého hlediska a udržitelnosti aplikace je vhodné psát jednotkové testy, které testují části aplikace a pomáhají zjišťovat chyby v implementaci. Protože IONIC využívá jako programovací jazyk AngularJS, tak je možné využívat komponenty pro testování AngularuJS.

Příklady frameworků vhodných pro jednotkové testy:

- https://jasmine.github.io/ - jednoduchá knihovna bez dalších závislostí
- https://mochajs.org/ - funguje asynchronně nad Node.js
- https://qunitjs.com/

5 Analýza a návrh mobilní aplikace využívající framework IONIC

V této kapitole je řešena analýza a návrh mobilní aplikace PowerFLOW metodikou pro Malé softwarové projekty. K zakreslení diagramů je použité ULM 2.0, které je i metodikou využívané. UML je univerzální jazyk pro vizuální modelování systému, sám o sobě není metodikou.

5.1 Analýza zavedeného systému PowerFLOW

Mobilní aplikace vzniká pro již fungující systém. Pro navržení funkcionality mobilní aplikace je nutné vytvořit základní popis zavedeného systému. V této části jsou popsány vlastnosti a principy zavedeného systému, stavové diagramy jeho komponent, analytický model pro pochopení vztahů mezi třídami, diagram komponent pro pochopení komunikace částí systému a diagram nasazení.

Seznámení s aplikací PowerFLOW

PowerFLOW slouží k vytváření pracovních procesů pro podporu týmové práce. Na trhu jsou podobné nástroje, ale jsou velice robustní, náročné na údržbu a jejich konfigurace a správa vyžaduje zaškoleného odborníka. V PowerFLOW lze vytvořit workflow bez programování a speciálních znalostí.

Vlastnosti PowerFLOW:

- Rychlá příprava jednoduchých procesů bez programování.
- Nastavení a měření SLA pro celý proces a pro každou fázi.
- Notifikace pro řešitele úkolů v případě nedodržení SLA.
- Zveřejnění procesů lze provést až ve čtyřech krocích: vývoj, test, akceptace, produkce.
- Předpřipravené fáze procesů, které se často využívají: úkol, doporučení, schválení.
- Přehledná fronta jak svých úkolů, tak týmových.
- Ukládá se detailní historie a auditní stopa o průběhu procesů.
- Komentáře a diskuze k procesů.

- Práce s dokumenty a přílohami.
- Sledování procesů, reporting.
- Korelace procesů dle obchodních případů.
- Procesy lze vyhledávat dle business atributů a identifikátoru případu.
- Uživatelé si mohou uložit často používané typy procesů do oblíbených a pak je rychleji spustit znovu.
- Řešitel úkolu může předat úkol na jiného řešitele, navrhnout storno nebo si vyžádat doplnění zadání.
- Zadavatel může v každé úkolovací fázi provádět akceptaci řešení a případně si vyžádat doplnění řešení.

Princip zavedené aplikace

Základním stavebním kamenem aplikace PowerFLOW je vytvoření a spouštění procesů v administračním rozhraní. V klientské části poté aplikace umožňuje plnit a spravovat úkoly ve spuštěných procesech. Zjednodušený sled kroků použití aplikace PowerFLOW je následující:

1. Manager zmapuje procesy ve firmě a vytvoří v management části PowerFLOW požadavek a nakonfiguruje ho. Přiřadí k jednotlivým fázím řešitele a managery.
2. Požadavek je spuštěn.
3. Fáze, která je na řadě, se zařadí do týmové fronty řešitelů.
4. Jeden z řešitelů začne na fázi pracovat a změní ji stav na claimed. Tím se odstraní ostatním z týmové fronty.
5. Řešitel fázi po jejím dokončení označí za vyřešenou a pokud je další fáze ve spuštěném požadavku, tak je spuštěna.
6. Opakovaně se provádějí kroky 3-5 dokud nejsou vyřešeny všechny fáze požadavku.
7. Po zpracování všech fází systém nastaví požadavek za dokončený.

Požadavek

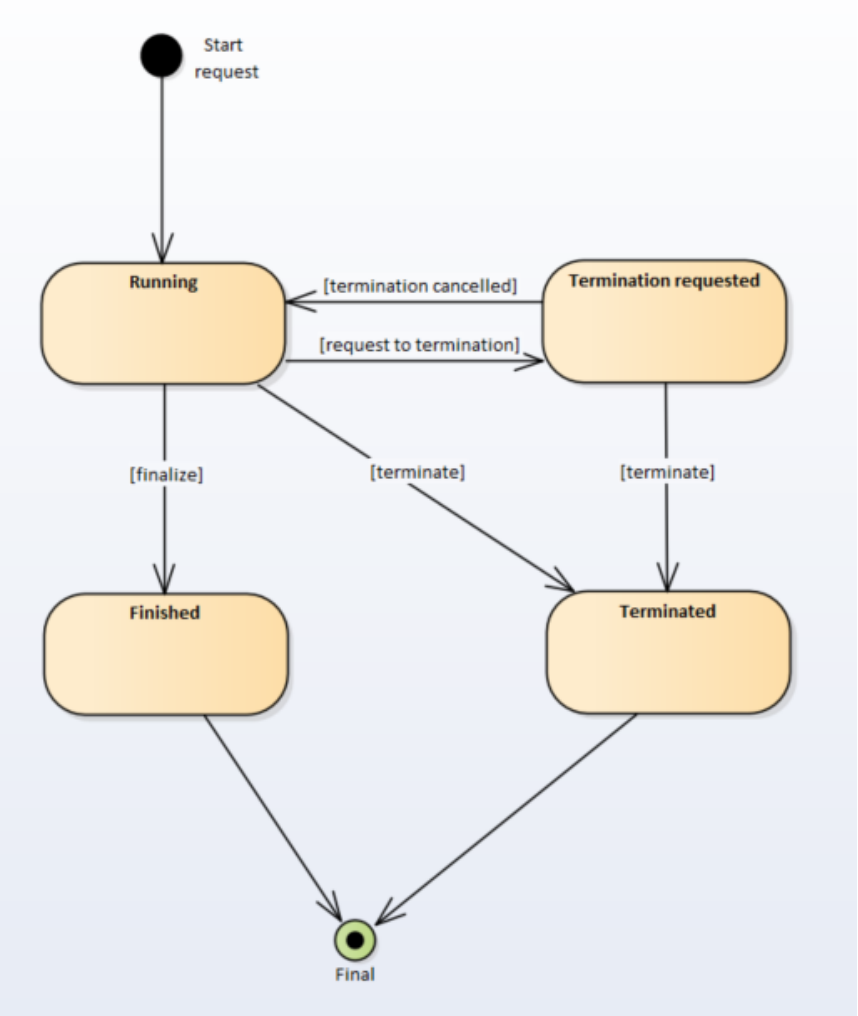
Požadavek je namodelovaný proces v management části PowerFLOW. Na obrázku 11 vidíme tři základní fáze požadavku:

1. doporučovací fáze
2. úkolovací fáze
3. schvalovací fáze

Fáze jsou základní, protože aplikace je připravena na přidávání dalších fází podle potřeb zákazníků. Fáze jsou na jedné čáře, protože požadavek nelze větvit mimo jednotlivé fáze. Jednotlivé fáze jsou popsány dále.

Obrázek 11:
Ukázka namodelovaného požadavku [autor]

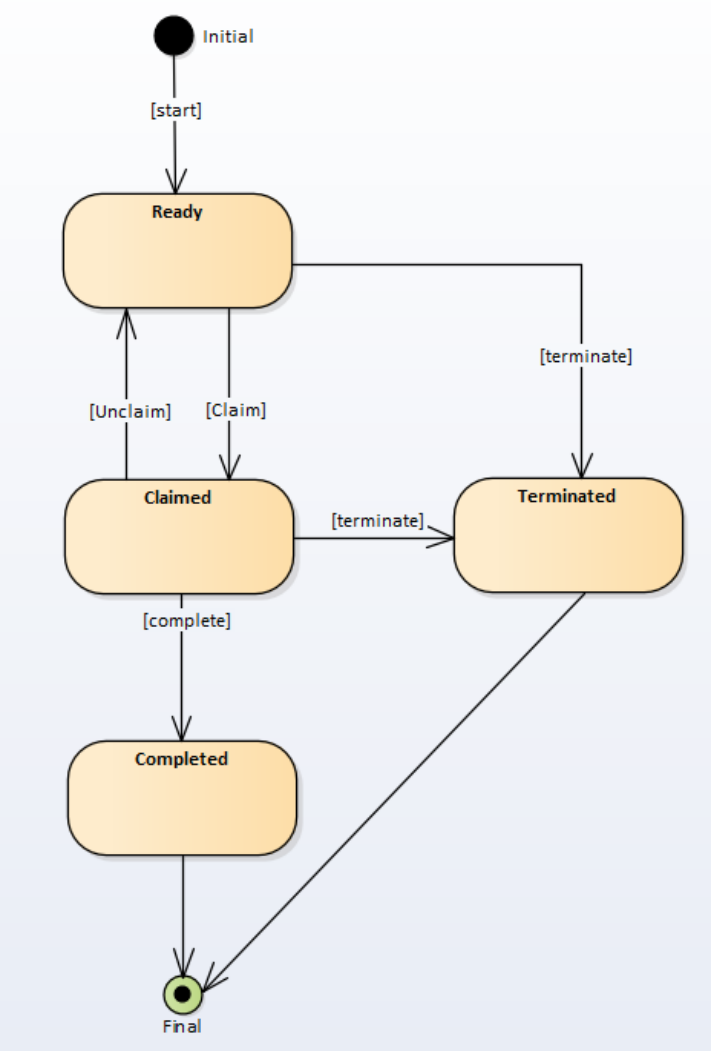
Na obrázku 12 vidíme jednotlivé stavy, do kterých se může požadavek dostat. Ihned po spuštění požadavku je ve stavu „running“. V tomto stavu se vykonávají jednotlivé fáze. Během stavu „running“ může dojít k žádosti o terminaci požadavku. V takovém případě je buď žádosti vyhověno a požadavek je terminován a jeho činnost je ukončena nebo je žádost zrušena a vrátí se zpět do stavu „running“. Může dojít také k přímé terminaci požadavku vlivem výstupu fáze (např. neschválení, nemožnost dokončení úkolu...). Po dokončení všech fází v požadavku systém změní jeho stav na „finished“.



Obrázek 12:
Stavový diagram požadavku [autor]

Fáze PowerFLOW

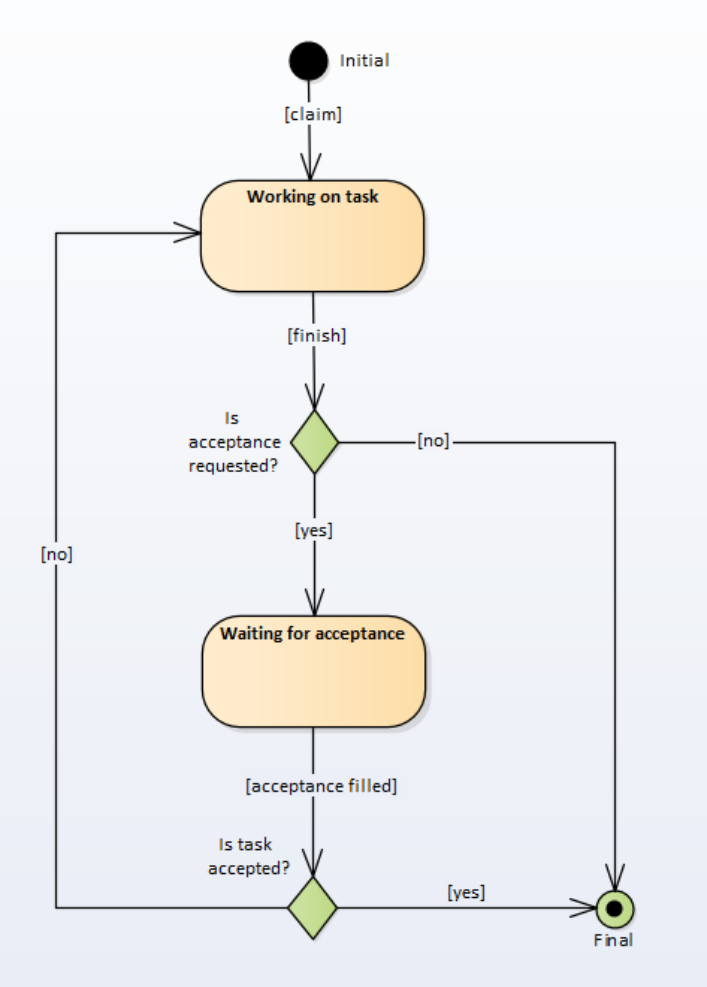
V této části jsou popsány fáze požadavku. Fáze jsou prozatím pouze tři, protože jsou velmi univerzální a jde jimi namodelovat velké množství procesů. Na obrázku 13 vidíme do jakých stavů se může fáze dostat. Po spuštění fáze se dostane do stavu „ready“. V tomto stavu přetrvává, dokud se někdo z řešitelů neujme jejím vypracováním. Ve stavu „ready“ je fáze v týmové frontě. V případě, že na fázi začne někdo z řešitelů pracovat, dostane se do stavu „clamed“. Po dokončení práce řešitel označí fázi za dokončenou a v případě, že jsou splněny všechny náležitosti se její stav uzavře. Pokud během stavu „ready a „clamed“ dojde k terminaci požadavku, je fáze také terminována před jejím vyřešením.



Obrázek 13:
Stavový diagram fáze [autor]

Úkolovací fáze

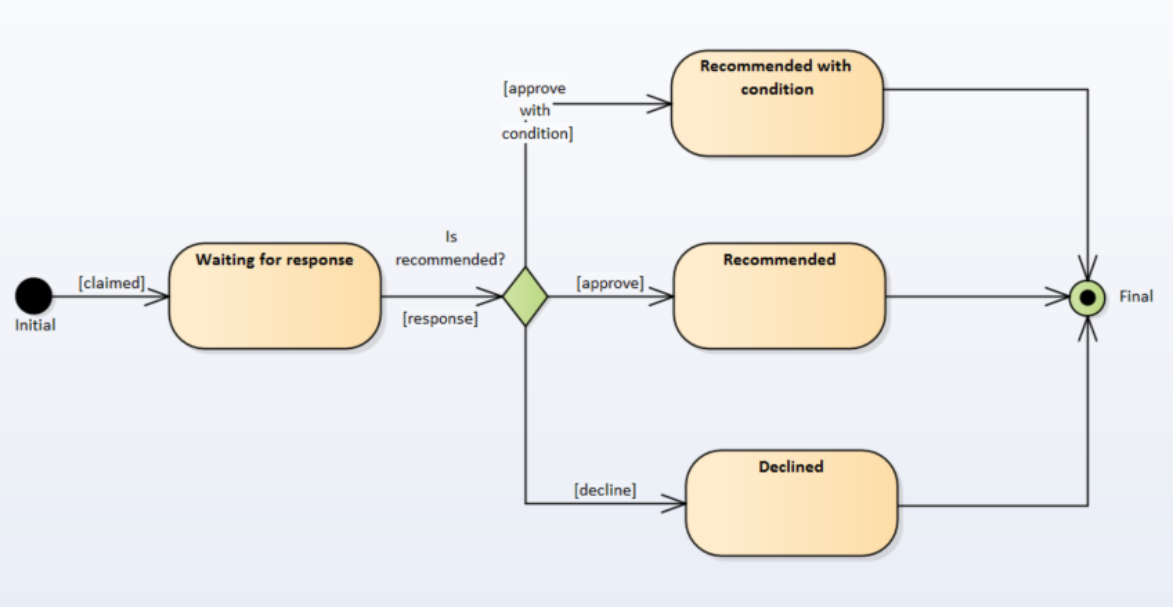
Úkolovací fáze slouží k zadání úkolu řešitelům. Po dokončení úkolu řešitel informuje prostřednictvím formuláře o jeho stavu. Formulář obsahuje výběr způsobu řešení a komentář. Na obrázku 14 vidíme jednotlivé stavy, do kterých se úkolovací fáze může dostat. Po přiřazení řešitelem „claim“ se fáze dostane do stavu „working on task“. V případě, že není vyžadována akceptace, tak se fáze dokončí. V opačném případě je ve stavu „waiting for acceptance“. Při pozitivním výstupu akceptace fáze končí a při negativním se fáze vrací do stavu „working on task“ a proces se znovu opakuje.



Obrázek 14:
Stavový diagram úkolovací fáze [autor]

Doporučovací fáze

Doporučovací fáze je vhodná pro schválení jedním řešitelem. Na obrázku 15 vidíme stavy, do kterých se doporučovací fáze může dostat. V této fázi lze, aby řešitel doporučil, doporučil s podmínkou nebo odmítnul řešení.



Obrázek 15:
Stavový diagram doporučovací fáze [autor]

Schvalovací fáze

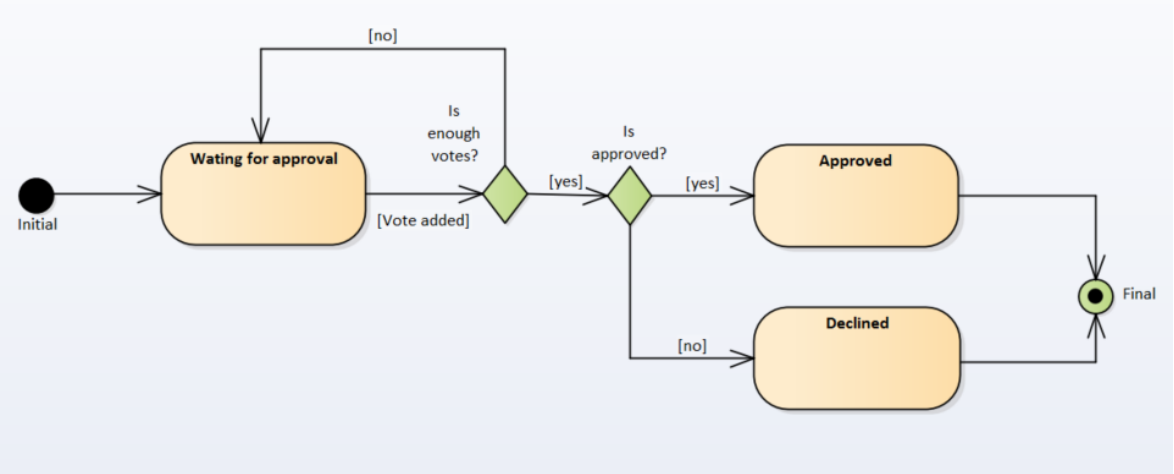
Schvalovací fáze slouží ke schválení více řešiteli. Fáze umožňuje nastavit způsob schválení více řešiteli následujícím způsobem vyhodnocení:

- Alespoň jeden řešitel schválí.
- Více než polovina řešitelů schválí.
- Všichni řešitelé schválí.
- Lze zadat procentuálně.

Fáze je vyhodnocena ve chvíli, když:

- Je jasné, že již nelze splnit podmínka hodnocení – přejde do stavu „Declined“.
- Je jasné, že podmínka hodnocení je již splněná – přejde do stavu „Approved“.

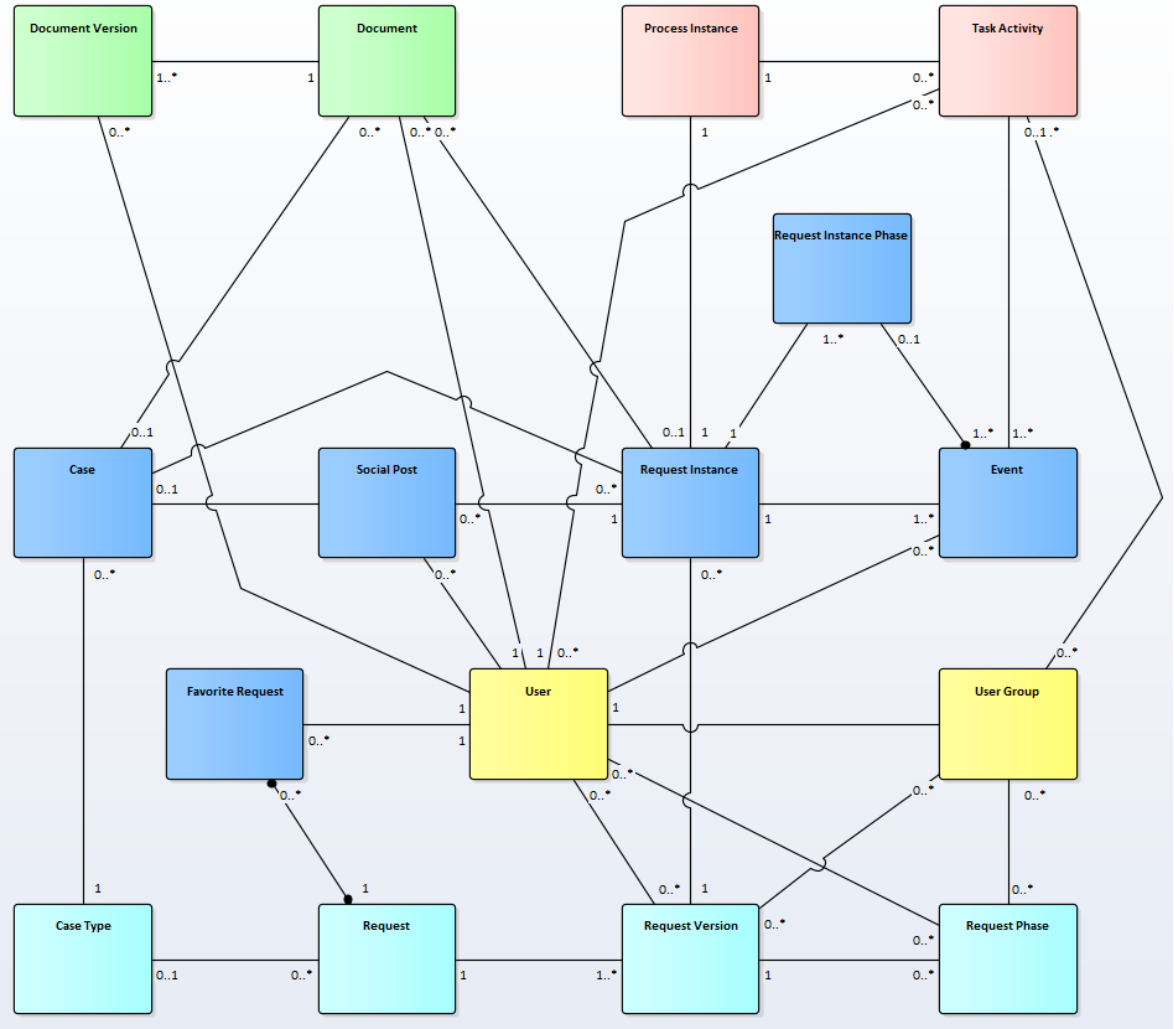
Na obrázku 16 lze vidět stavový diagram schvalovací fáze.



Obrázek 16:
Stavový diagram schvalovací fáze [autor]

Analytický model tříd

Na obrázku 17 je zobrazen analytický model tříd aplikace PowerFLOW. Lze vidět hlavní propojení entit a kardinality vztahů. Z obrázku je zřejmé, že logický datový model je hodně složitý. Logický datový model pro vývoj mobilní aplikace nepotřebujeme. Stačí dokumentace vystaveného REST API. Nicméně nám tento model poskytuje nahlédnutí do databázové struktury PowerFLOW a tím můžeme více pochopit princip jeho fungování. Třídy *Document* a *DocumentVersion* jsou označeny zeleně, protože o jejich uložení se stará napojená komponenta *Document management system*. Třídy *User* a *Group* jsou označené žlutě. Jejich informace se získávají z napojené komponenty *User repository*. Třídy označené červeně a tmavě modře jsou již spuštěné požadavky, které se ukládají v do systému PowerFLOW. Světle modré třídy jsou vytvářeny před spuštěním požadavku, jako šablony pro spouštění. Podrobnější popis tříd a jejich dopad na různé systémy je v tabulce 2.



Obrázek 17:
Analytický model tříd backendu PowerFLOW [autor]

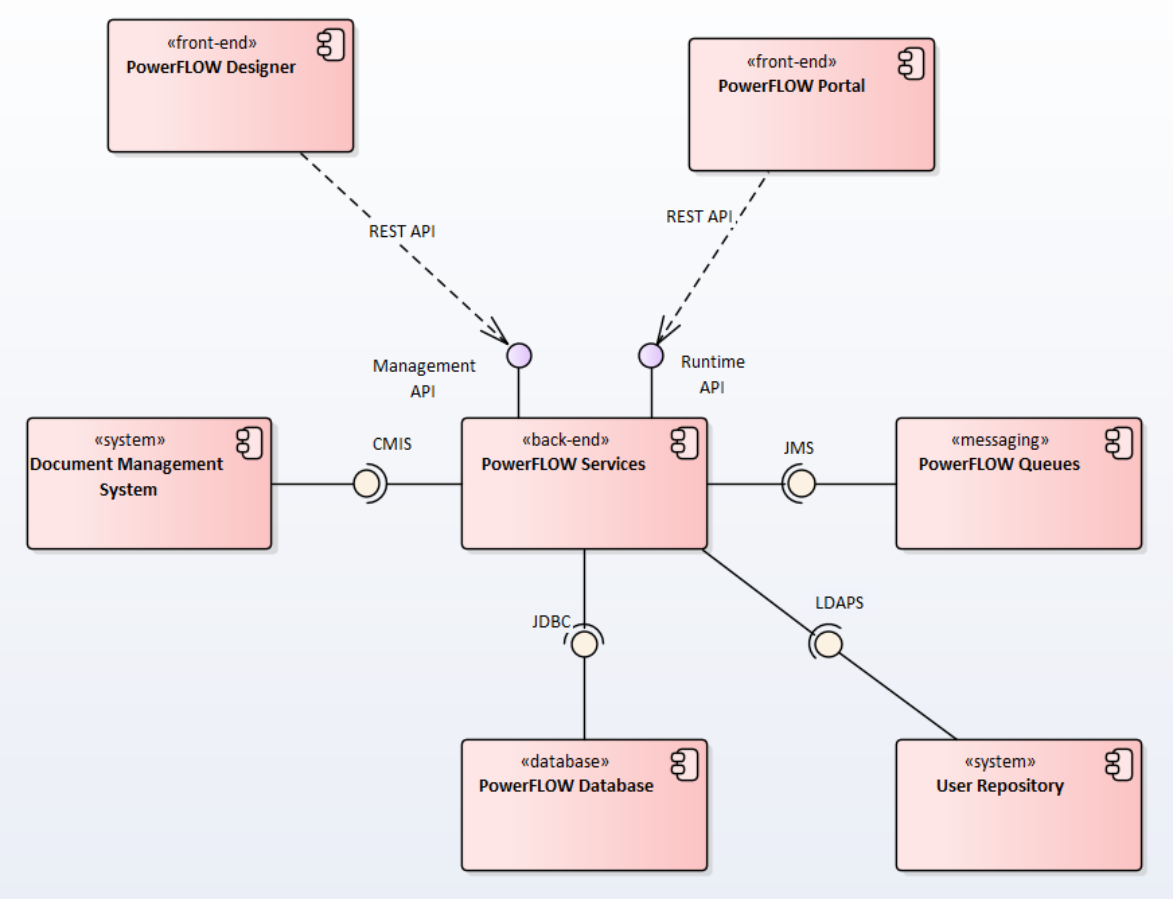
Tabulka 2: Popis tříd analytického modelu

Třída	**Popis třídy**	**Systém**
User	Reprezentuje uživatele aplikace	User Repository
User Group	Skupina uživatelů. Např.: management	User Repository
Request	Šablona požadavku	PowerFLOW Services
Request Version	Verze šablony požadavku	PowerFLOW Services
Request Phase	Šablona fáze požadavku	PowerFLOW Services
Case Type	Typ případu. Např.: Půjčka, vklad, ...	PowerFLOW Services

Favorite Request	Oblíbený požadavek uživatele	PowerFLOW Services
Case	Konkrétní případ.	PowerFLOW Services
Social Post	Komentář	PowerFLOW Services
Request Instance	Spuštěný požadavek	PowerFLOW Services
Request Instance Phase	Fáze spuštěného požadavku	PowerFLOW Services
Event	Událost požadavku	PowerFLOW Services
Proces Instance	Třída, reprezentující spuštěný požadavek v Activity BPM.	Activiti BPM
Task Activity	Třída, reprezentující fázi/úkol v Activity BPM.	Activiti BPM
Document	Dokument	Document Management System
Document Version	Verze dokumentu	Document Management System

Diagram komponent

Na obrázku 18 jsou zachyceny komponenty systému PowerFLOW. Všechny uvedené systémy jsou již vytvořené a spuštěné pro testovací účely. V tabulce 3 jsou popsány jednotlivé komponenty aplikace a v tabulce 4 jsou vypsány zkratky komunikačních protokolů.



Obrázek 18:
Diagram komponent systému PowerFLOW [autor]

Tabulka 3: *Popis komponent*

Název komponenty	**Popis**
PowerFLOW Services	Klíčový modul pro správu dat z celého systému. Integruje jednotlivé moduly a potřebné informace ukládá do databáze.
PowerFLOW Designer	Frontend, ve kterém je možné modelovat a spouštět požadavky.
PowerFLOW Portal	Frontend, ve kterém jsou zobrazeny fronty úkolů. Je zde možná jejich správa. Také pouštět požadavky a sledovat reporty.
PowerFLOW Database	Databáze systému.
PowerFLOW Queues	Komponenta starající se o asynchronní odesílání zpráv.
Document management system	Komponenta na správu souborů.

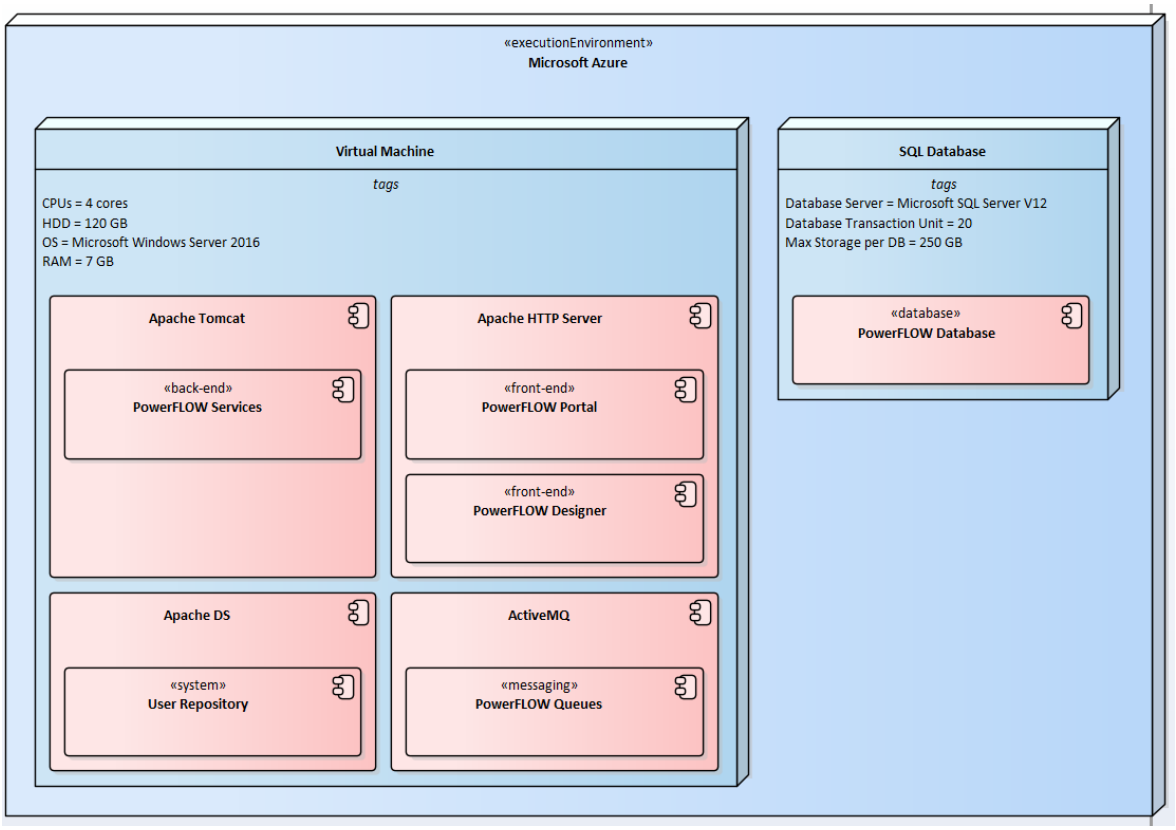
User repository	Komponenta na správu uživatelů a přístupů.
Apache ActiveMQ	Systém na posílání zpráv

Tabulka 4: *Popis rozhraní komunikace*

Zkratka	**Celý název**
API	Application Programming Interface
REST	Representational State Transfer
JDBC	Java Database Connectivity
CMIS	Content Management Interoperability Services
LDAPS	Lightweight Directory Access Protocol
JMS	Java Messaging Services

Diagram nasazení

Na obrázku 19 lze vidět, kde se provozují komponenty aplikace PowerFLOW. Pro hosting je využívána služba Microsoft Azure, kde běží jak databáze aplikace, tak její moduly. Pro databázi je využíván Microsoft SQL server a pro ostatní služby je spuštěn virtuální počítač s operačním systémem Microsoft Windows Server 2016.



Obrázek 19:
Diagram nasazení PowerFLOW [autor]

5.2 Analýza aplikace PowerFLOW mobile

V této části je analýza a návrh mobilní aplikace PowerFLOW mobile, která vznikala při psaní diplomové práce. K analýze a návrhu se využívá metodika MMSP. V úvodu podkapitoly jsou obecnější popisy a postupně se více specifikují. Na začátku je zkrácená vize projektu, následuje vymezení problému a produktu, popis zúčastněných subjektů, přehled produktu s obecnými požadavky a plán projektu. Podkapitola pokračuje seznamem rizik a z ní je vytvořena matice rizik, podle které lze vyhodnotit nejvýznamnější rizika projektu. Následuje slovník pojmů, který specifikuje pojmy použité při analýze a návrhu aplikace. Z obecných požadavků byly vytvořeny již konkrétní funkční požadavky a zadavatelem byly definovány nefunkční požadavky. Funkční požadavky jsou znázorněny v případech užití aplikace. Pro mobilní aplikaci je velice důležitý návrh uživatelských obrazovek, aby se s aplikací dobře pracovalo. Obrazovky jsou navržené přímo pomocí komponent frameworku IONIC a podle funkčních požadavků. Podkapitola neobsahuje diagram tříd, protože aplikace využívá pro data služeb REST API systému PowerFLOW services. Služby REST API jsou popsány po návrhu obrazovek. Pro zobrazení komunikace mobilní aplikace je uveden diagram komponent a na konci části je diagram nasazení.

Vize

Vizí projektu je realizovat multiplatformní mobilní aplikaci pro systém Power-FLOW ve frameworku IONIC. Aplikace má na starosti plnění a administraci úkolů v pracovních procesech. Mobilní aplikace pomůže uživatelům, kteří momentálně nemohou obsluhovat počítač a chtějí s úkoly pracovat.

Vymezení problému a produktu

Tabulka 5: *Vymezení problému a produktu*

Definice problému	Firma vlastní informační systém PowerFLOW a narazila na možnost vhodného rozšíření aplikace na mobilní platformy pro koncové zákazníky.
Přínosy úspěšného řešení problému	Koncoví uživatelé systému PowerFLOW budou moci řešit část funkcionality přes mobilní klienty, takže prakticky od-kudkoliv.
Pro	Uživatele systému PowerFLOW
Název vyvíjeného IS/ICT	PowerFLOW mobile

Popis všech zúčastněných subjektů

Tabulka 6: *Popis všech zúčastněných subjektů*

Název	Popis	Odpovědnost
Notix s.r.o.	It společnost, pro kterou je nový IS vyvíjen	Hlavní článek projektu, investor projektu
Koncový uživatel Power-FLOW	Osoba využívající IS/ICT systém PowerFLOW	Má přístup do systému a využívá funkce Power-FLOW

Přehled produktu

V tabulce 7 jsou uvedeny obecné požadavky na IS/ICT z hlediska funkcionality. Požadavky jsou detailněji rozpracovány v časti *Požadavky* a *Případy užití*.

Tabulka 7: *Seznam obecných požadavků*

Požadavek (potřeba)	Priorita (1-5)	Podrobný popis
Mé úkoly	1	Zobrazení detailu mého úkolu. Možnost dokončení.
Komentáře	2	Zobrazení komentářů, přidání komentáře.
Pod-úkoly	2	Zobrazení pod-úkolů
Dokumenty	3	Zobrazení dokumentů a jejich stažení.
Business atributy	2	Zobrazení business atributů a
Push notifikace	2	Přijímání notifikací z jiné aplikace a jejich zobrazování na stavovém řádku mobilní aplikace.

Plán projektu

V tabulce 8 lze vidět milníky jednotlivých částí projektu. Vždy, když fáze začíná, tak fáze před tím končí. Ve fázi zahájení projektu se řešili administrativní záležitosti s vedoucí DP a zadavateli. Proběhla základní analýza. Zahájení je klíčovým prvkem od-startování projektu, definování záměru projektu a sbírání požadavků. Ve fázi Rozpracování se klade důraz na upřesnění požadavků a definuje se architektura a fungování systému. Následuje fáze Konstrukce, kde se provádí samotná implementace zdrojového kódu. Jako poslední je fáze nasazení, v tomto případě na dva mobilní operační systémy. Během této fáze se provádí akceptace zákazníkem. Zákazníkem je na tomto projektu zástupce firmy Notix s.r.o.

Tabulka 8: *Plán projektu*

Milník	Termín
Zahájení projektu	20.6.2016
Fáze Zahájení	28.6.2016
Fáze Rozpracování	28.7.2016

Fáze Konstrukce	28.8.2016
Fáze Nasazení	30.11.2016
Konec první iterace	10.12.2016

Seznam rizik

V tabulce 9 jsou uvedena rizika projektu a jejich protiopatření. Ke každému z rizik byla také přiřazena závažnost a pravděpodobnost z množiny {malá, střední, vysoká}. Díky zvolené množině možností jsme schopni vytvořit Winterlingovu krizovou matici a určit největší rizika.

Tabulka 9: *Seznam rizik*

ID	Název	Popis	Závaž-nost	Pravdě-podob-nost	Preventivní opatření
1	Neznalost technolo-gií	Neznalost frameworku IONIC a s ním sdruže-ných technologií.	vysoká	střední	Pořízení knížek, rešerše zdrojů, konzultace s vývojáři firmy.
2	Málo času	Vývoj aplikace, učení nových technologií a psaní diplomové práce zabere dohromady hodně času.	vysoká	vysoká	Přesun osobních aktivit. Vyšší alokace na projekt ve firmě.
3	Nedosta-tečná analýza	Nedostatečné informace od zadavatele během analýzy.	střední	střední	Vyhrazení alespoň hodinu týdně od vývojářů Power-FLOW.
4	Testovací data	Testovací data jsou potře-ba ke kontrole správné funkcionality.	nízká	nízká	Včas zažádat o testovací data, aby byla dodána před jejich potřebou.
5	Schvalo-vání	Schvalování je nutné během vývoje, aby nedo-šlo k odchýlení od zadání.	střední	střední	Průběžně ukazovat vývoj aplikace zodpovědnému pracovníkovi.

Matice rizik

Z tabulky 10 lze vypozorovat, že největší riziko představuje riziko s ID 2: Málo času. Proto musím věnovat zvláštní pozornost jeho protiopatřením. Dalším významným rizikem je ID 1: Neznalost technologií. Tabulka ukazuje nejmenší rizika, která jsou vlevo dole po ty nejhorší umístěná vpravo nahoře.

Tabulka 10: *Matice rizik: Winterlingova krizová matice [37]*

Vysoká		1	2
Střední	3	5	
Nízká	4		
Závažnost P(X)	Nízká	Střední	Vysoká

Slovník pojmů

V tabulce 11 jsou výrazy, které jsou používané v rámci aplikace. Proto je k nim správné přiřadit vysvětlení.

Tabulka 11: *Slovník pojmů aplikace*

Pojem	Vysvětlení
Požadavek	Požadavek je proces, který je namodelovaný v management části aplikace. Je složený z fází.
Fáze	Fáze jsou jednotlivé úkoly v požadavku. Základní fáze jsou tři: schvalovací, doporučovací a úkolová.
Business historie	Historie událostí, které se staly na požadavku.
Případ (case)	Požadavky se mohou vázat na konkrétní případ.
Řešitelé	Zaměstnanci firmy, kteří mají zodpovědnost za splnění fáze požadavku.
Manageři	Zaměstnanci firmy, kteří mají na starost dohlížení požadavků a fází. Také mají možnost změn fází a požadavku.
Týmová fronta	Fáze, která se spustí, je ihned zařazena do týmové fronty. V týmové frontě jej vidí všichni řešitelé.
Claimed	Fáze požadavku, která je z týmové fronty odebrána, protože ji jeden z řešitelů začal zpracovávat.
Úkol	Termín, který používá zákazník pro fázi požadavku.
Business atributy	Jsou proměnné, které lze editovat jak u případu, tak u fáze.
Zákazník	Pověřený zástupce firmy Notix s.r.o.
Wireframe	Návrh obrazovky, který slouží k ujasnění pozic elementů na stránce. Vytváří se před vývojem aplikace a jsou schváleny zadavatelem.

Požadavky

Úkolem požadavků je poskytnout přehled funkcionalit, který má být vyvinut. Požadavky byly se zadavatelem projednány a odsouhlaseny.

Funkční požadavky

Tabulka 12: *Funkční požadavky*

ID požadavku	Název požadavku	Podrobný popis	Priorita 1:nejméně 5:nejvíce
F001	Prohlédnutí vstupních obrazovek	Aktér si nainstaluje mobilní aplikaci a před samotným přihlášením má možnost prohlédnutí informačních obrazovek.	2
F002	Zadání URL adresy backendu	PowerFLOW obsahuje citlivá data, proto každý zákazník má svou instanci backend aplikace. Je tedy nutné, aby do mobilní aplikace mohl aktér zadat, kam se má připojit.	5
F003	Změna URL adresy backendu	Umožnuje aktérovi změnit již zadanou URL adresu backendu.	5
F004	Přihlášení do systému	Aby se aktér dostal k zabezpečeným informacím a funkcionalitě aplikace, tak je nutné, aby se aktér přihlásil do systému.	5
F005	Odhlášení	Umožnuje se aktérovi odhlásit z aplikace.	5
F006	Zobrazení seznamu „Mé úkoly“	Aktér si zobrazí přehled úkolů, které jsou na něm přiřazené (claim). Uvidí tam název úkolu a požadavku.	5
F007	Zobrazení seznamu „Týmové úkoly“	Aktér si zobrazí přehled úkolů, ve kterých je jedním z řešitelů, nicméně je ještě nikdo nemá přiřazené na sobě. Uvidí tam název úkolu a požadavku.	4
F008	Zobrazení informací přihlášeného uživatele	Aktér má možnost se podívat, pod jakým profilem je přihlášený a jaké jsou u něj uvedeny informace (Jméno, příjmení, telefonní kontakt, fotografie)	2
F009	Zobrazit	Zobrazí seznam dokumentů k požadavku, kde je	4

	seznam dokumentů k požadavku	vidět název dokumentu, kdo dokument vložil a čas vložení.	
F010	Stažení poslední verze dokumentu u požadavku	Aktér má možnost stáhnout si poslední verzi nahraného dokumentu k požadavku.	3
F011	Zobrazit seznam komentářů k požadavku	Zobrazení seznamu komentářů k požadavku. U komentářů je uvedeno jméno a text komentáře.	5
F012	Přidání komentáře k požadavku	Aktér přidá komentář k požadavku.	5
F013	Zobrazení informací o úkolu	Zobrazení informací o úkolu: Název úkolu, popis úkolu, název požadavku, spuštěno na prostředí.	5
F014	Zobrazení business atributů požadavku	Zobrazí seznam business atributů požadavku. Zobrazené informace: název, hodnota.	4
F015	Zobrazení business atributů případu	Zobrazí seznam business atributů případu. Zobrazené informace: název, hodnota.	4
F016	Zobrazení seznamu pod-úkolu úkolu	Zobrazení seznamu pod-úkolu úkolu, kde je uvedeno: název pod-úkolu, popis, termín, řešitel, status a priorita.	4
F017	Zobrazení historie požadavku	Zobrazí seznam událostí historie požadavku, kde je uvedeno v každém úkolu: název úkolu, jméno uživatele/systém, datum a čas, název události.	3
F018	Přihlášení k řešení úkolu (claim)	Aktér může úkol z týmové fronty odebrat tím, že se přihlásí k jeho řešení. Změní tedy stavu úkolu ze stavu „ready“ na „claimed“.	5

F019	Odhlášení se z řešení úkolu (unclaimed)	Aktér se může odhlásit z řešení úkolu a změnit jeho stav z „claimed“ zpátky na „ready“. Tím úkol odebere ze seznamu „Mých úkolů“ a přeřadí ho do seznamu „Týmové úkoly“.	5
F020	Editace business atributů požadavku	Aktér je schopen změnit hodnotu business atributů požadavku. Hodnoty business atributů mohou nabývat typů: STRING, TEXT, DATE, DATETIME, BOOLEAN, URL, EMAIL, DECIMAL.	2
F021	Editace business atributů případu	Aktér je schopen změnit hodnotu business atributů případu. Hodnoty business atributů mohou nabývat typů: STRING, TEXT, DATE, DATETIME, BOOLEAN, URL, EMAIL, DECIMAL.	2
F022	Dokončení úkolu	Aktér může úkol, ke kterému je přihlášený dokončit. Úkol mohou být tři základní fáze a každá z nich má rozdílný způsob řešení.	5

Nefunkční požadavky

Tabulka 13: *Nefunkční požadavky*

ID požadavku	**Název požadavku**	**Podrobný popis**	**Priorita** 1:nejméně 5:nejvíce
N001	Framework IONIC	Na vývoj mobilní aplikace se použije framework IONIC, protože je to moderní framework a firma má již zkušenost jak s vývojem, s nasazením a dlouhodobou podporou.	5
N002	Intuitivnost ovládání	UI komponenty v aplikaci budou použity v rámci „best practice“ pro mobilní vývoj. Aktér se tedy bude snadno moci zorientovat v prostředí aplikace.	4
N003	Doba odezvy	Průměrná doba odezvy nepřesáhne 2 sec. pro načítání dat do aplikace a 4 sec. pro odesílání formulářů. Tato odezva platí pro připojení k internetu o rychlosti minimálně 10Mb/s.	3
N004	Konzistence terminologie	Terminologie použitá v mobilní aplikaci musí korespondovat s terminologiemi ostatních částí	5

		PowerFLOW systému.	
N005	Programovací jazyk	Mobilní aplikace bude vyvíjena v programovacím jazyku AngularJS.	5
N006	Připojení	Mobilní aplikace nebude umožňovat offline mód. Pro její správnou funkčnost tedy musí být za běhu neustále připojena k internetu. Kvůli častým změnám a tedy konzistenci dat.	5
N007	Šifrování	Přihlášení uživatelé mobilní aplikace budou komunikovat se serverem prostřednictvím šifrovaného připojení HTTPS.	5
N008	Struktura aplikace	Struktura aplikace musí být vytvořena tak, aby aplikace mohla být udržitelná a rozšiřitelná.	5
N009	REST volání	Backend aplikace již má definované REST API pro klientskou část PowerFLOW. Požadavkem je, aby se tyto volání využívali i v rámci mobilní aplikace a došlo tak k minimálním úpravám. Tohle opatření povede dlouhodobě k lepší udržitelnosti kódu, protože se bude využívat jedna část backendu pro více frontend aplikací.	5

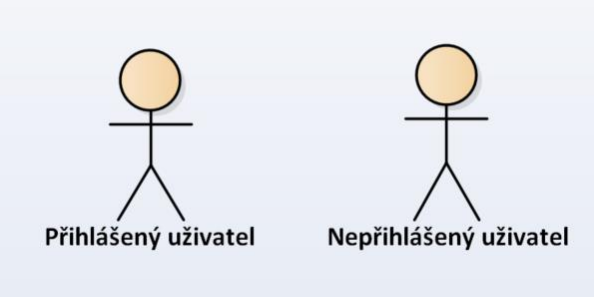
Model případů užití

V této části je zachycen model případů užití navrhované mobilní aplikace PowerFLOW. Aplikace PowerFLOW je mnohem rozsáhlejší, ale v této první iteraci se vývoj bude zabývat pouze případy užití zobrazenými níže.

Aplikační role

Na obrázku 17 jsou zobrazeny aplikační role, které budou s mobilní aplikací PowerFLOW interagovat. Níže si jednotlivé role rozebereme:

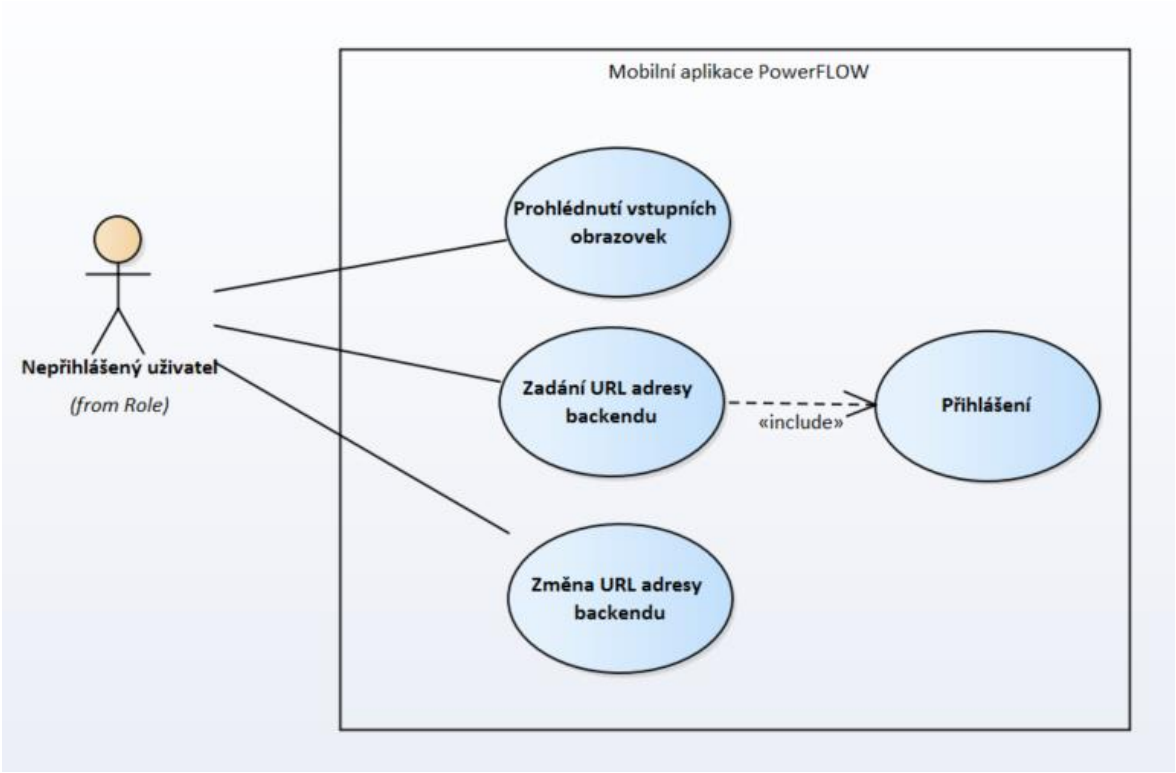
- Nepřihlášený uživatel může být koncový uživatel systému PowerFLOW, ale také uživatel, který si stáhl volně dostupnou mobilní aplikaci PowerFLOW z internetových portálů a se systémem PowerFLOW se ještě nesetkal.
- Přihlášený uživatel je koncový uživatel systému PowerFLOW, který má přihlašovací údaje ke svému účtu.



Obrázek 20:
Aplikační role [autor]

Nepřihlášený uživatel

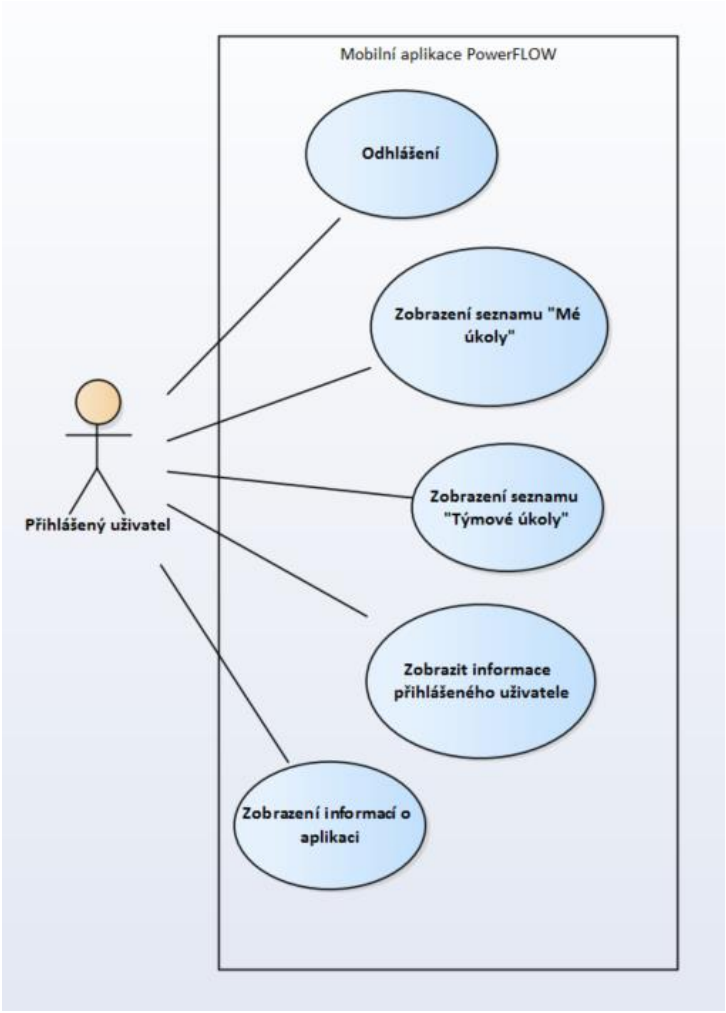
Na obrázku 21 jsou zachyceny případy užití aktéra v roli Nepřihlášeného uživatele. Protože mobilní aplikace neumožňuje registraci do systému, tak je potřeba seznámit se základními vlastnostmi PowerFLOW prostřednictví případu užití *Prohlédnutí vstupních obrazovek.* PowerFLOW má koncept takový, že pro každého klienta vytvoří vlastní prostředí systému. To zahrnuje nasazení vlastního backendu. Z tohoto důvodu je nutné před přihlášením nastavit na jaké URL se volání budou provádět. Proto je mezi UC *Přihlášení* a *Zadání URL adresy backendu* vazba *include*. Backend aplikace je možné přesunout na jiný hosting nebo změnit doménové jméno. UC, který se stará o zmíněnou variantu je *Změna URL adresy backendu*.



Obrázek 21:
Model případů užití: Nepřihlášený uživatel [autor]

Případy užití přihlášeného uživatele – základní

Na obrázku 22 jsou zachyceny základní případy užití přihlášeného uživatele. Aplikace mu umožňuje *Odhlášení*, *Zobrazení informací o přihlášeném uživateli* nebo *Zobrazení informací o aplikaci*. Klíčovými prvky aplikace je také zobrazení fronty úkolů. Ty se rozdělují na dvě: moje úkoly a týmové úkoly.

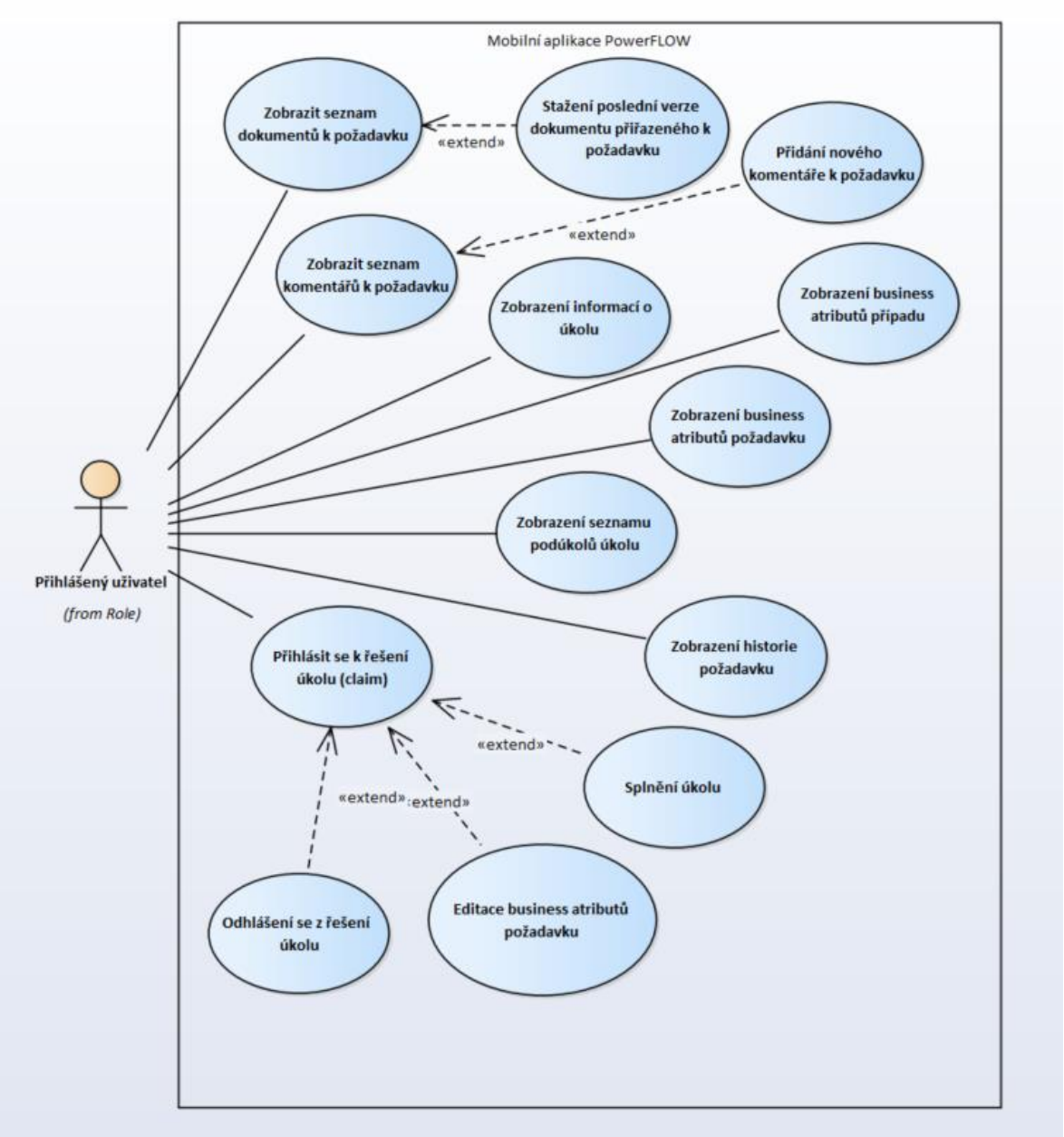


Obrázek 22:
Model případů užití: Přihlášený uživatel – základní případy užití [autor]

Případy užití přihlášeného uživatele – práce s úkoly

Na obrázku 23 vidíme případy užití týkající se práce s úkoly. Jsou zde použité vazby <<*extend*>>, které mohou rozšířit případ užití o další scénář. Cesta to však není nutná.

Každý spuštěný úkol se váže na spuštěný požadavek. Proto se i některé použité proměnné na obrázku případu užití váží k požadavku. Jsou to dokumenty, komentáře a business atributy požadavku.

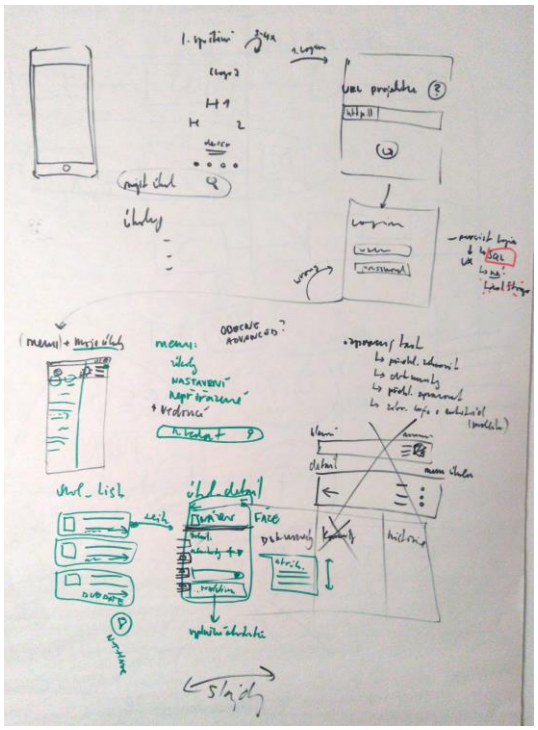


Obrázek 23:
Model případů užití: Přihlášený uživatel – práce s úkoly [autor]

Návrh uživatelského rozhraní

Návrh uživatelského rozhraní není bráno na lehkou váhu, protože špatné navržení obrazovek a způsobu ovládání může odradit uživatele od používání aplikace. Nejdříve bylo nutné si ujasnit, jaké obrazovky v aplikaci budou. K tomu posloužila schůzka se zákazníkem. Výstupem schůzky bylo vyjasnění obrazovek a jejich náplní. Také hlavní ovládání aplikace. Rozhodli jsme se pro hlavní průchody aplikací a ovládacími prvky. Na obrázku 24 jsou vidět poznámky psané na flipchart během schůzky ohledně obrazovek. Následoval výběr programu pro vytvoření wireframů. IONIC poskytuje službu, která se jmenuje IONIC Creator. Lze v ní vytvořit kostru aplikace včetně cest přesouváním drag-and-drop

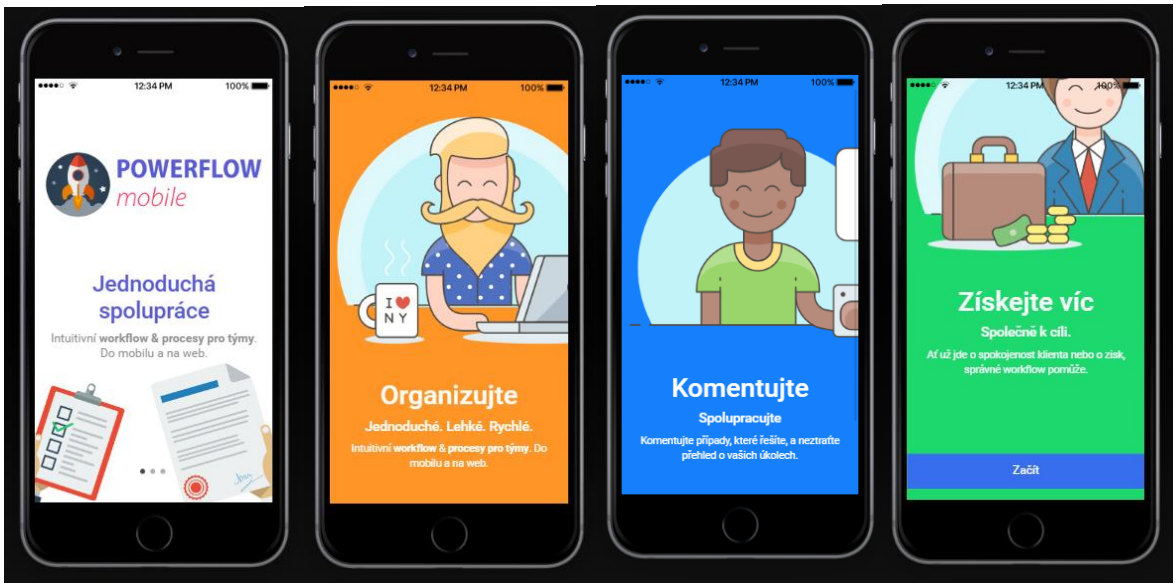
elementů. Poměrně jednoduchým způsobem lze vytvořit prototyp aplikace, která se dá ukázat koncovým uživatelům. Wireframy obrazovek lze vidět níže. Následně lze v Ionic creatoru přidat odkazy na tlačítka a vytvořit tak prototyp aplikace, který lze prezentovat zákazníkovi.



Obrázek 24:
Skica úvodního rozboru aplikace [autor]

Úvodní obrazovky

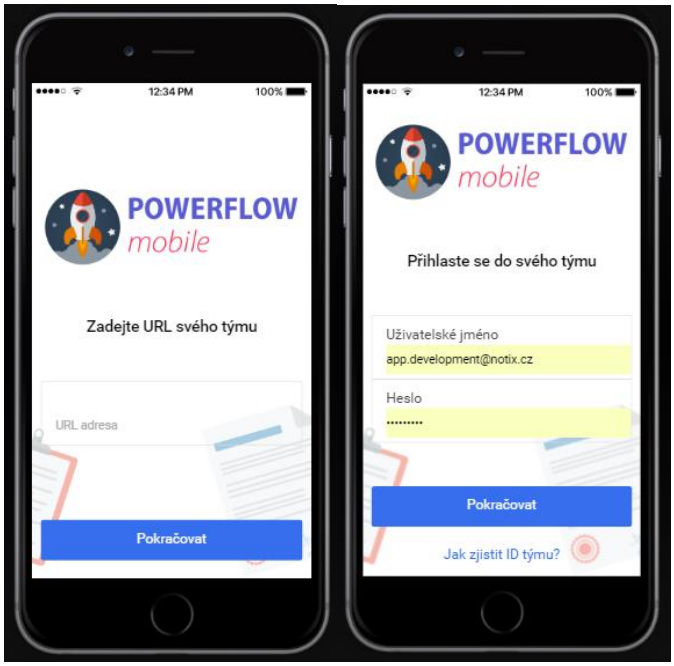
Úvodní obrazovky slouží k informování aktéra, který si aplikaci nainstaloval, a ještě není přihlášený. Jsou zobrazeny na obrázku 25. Bude zde použita komponenta slider, která umožňuje přejetím prstem do strany změnit obsah stránky. Obrazovky se týkají funkčního požadavku F001.



Obrázek 25:
Wireframes: Úvodní obrazovky [autor]

Obrazovky s přihlášením

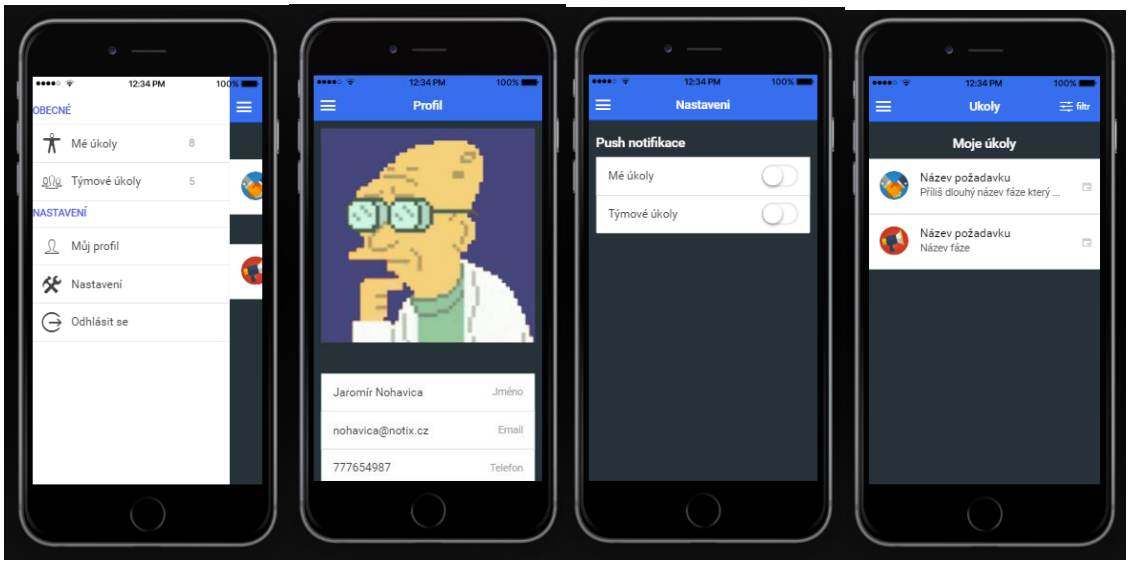
Na první obrazovce je vidět pole pro vyplnění URL adresy, abychom věděli, proti jakému serveru aktéra autentizovat. Na druhé obrazovce je vidět prostor pro vyplnění uživatelského jména a hesla. Obrazovky se týkají funkčních požadavků F002, F003. Jsou na obrázku 26.



Obrázek 26:
Wireframes: Přihlašovací obrazovky [autor]

Obrazovky přístupné z menu

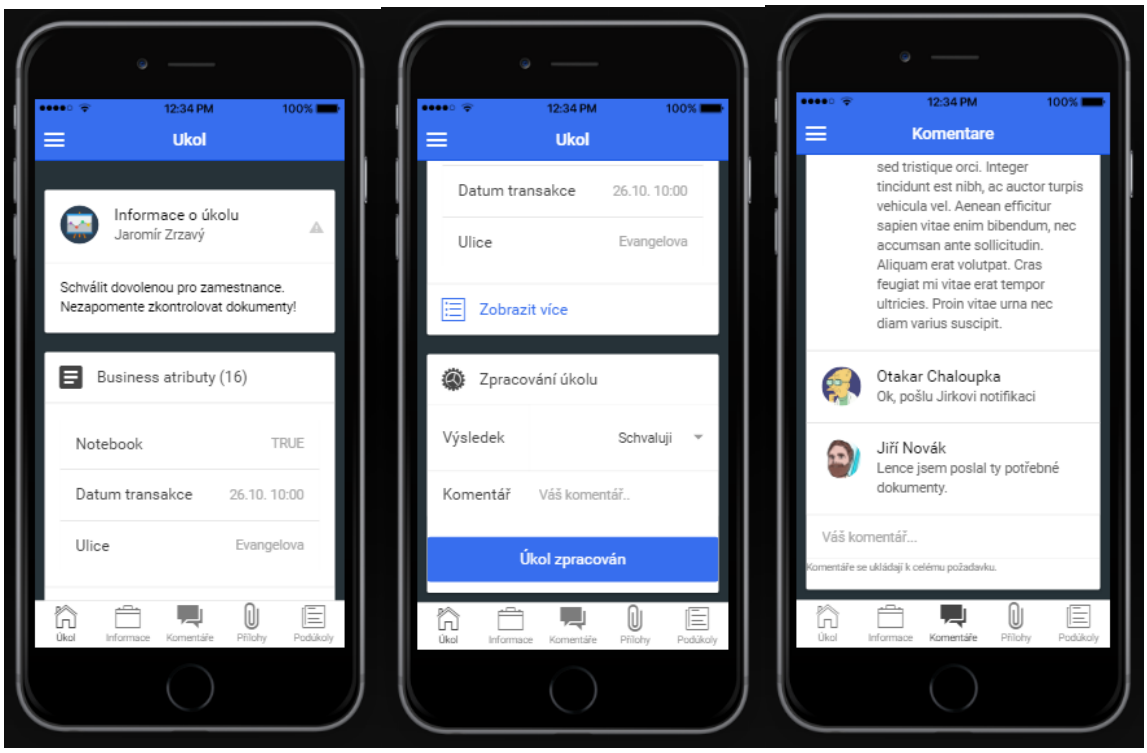
Na první obrazovce z obrázku 27 vidíme hlavní menu aplikace. Je dostupné skoro ze všech míst aplikace. Vysune se z levé strany, pokud aktér klikne do levého horního rohu na tři vodorovné čáry: hamburger menu. Tento prvek menu je v aplikacích iOS, tak Android hojně používán. Proto ho Ionic integroval jako jeden ze dvou podporovaných hlavních menu. Na druhé stránce je zobrazen profil přihlášeného uživatele. Na stránce je fotka, celé jméno, emailová adresa a telefonní číslo. Na stránku se dostaneme z menu po kliknutí na tlačítko „Můj profil“. Třetí stránka je s nastavením. Opět se na ní lze dostat z menu kliknutím na tlačítko nastavení. Čtvrtá obrazovka z obrázku 26 je fronta *Mých úkolů.* U jednotlivých úkolů je zobrazen *Název fáze* a *Název požadavku*. Obrazovky se týkají funkčních požadavků F005: Odhlášení, F006: Zobrazení seznamu „Mé úkoly“, F007: Zobrazení seznamu „Týmové úkoly“, F008: Zobrazení informací přihlášeného uživatele. Obrazovka nastavení se v této iteraci projektu neřeší, ale je jíž navržená v menu.



Obrázek 27:
Wireframes: Menu, profil, nastavení a fronta úkolů [autor]

Obrazovky detailu úkolu 1

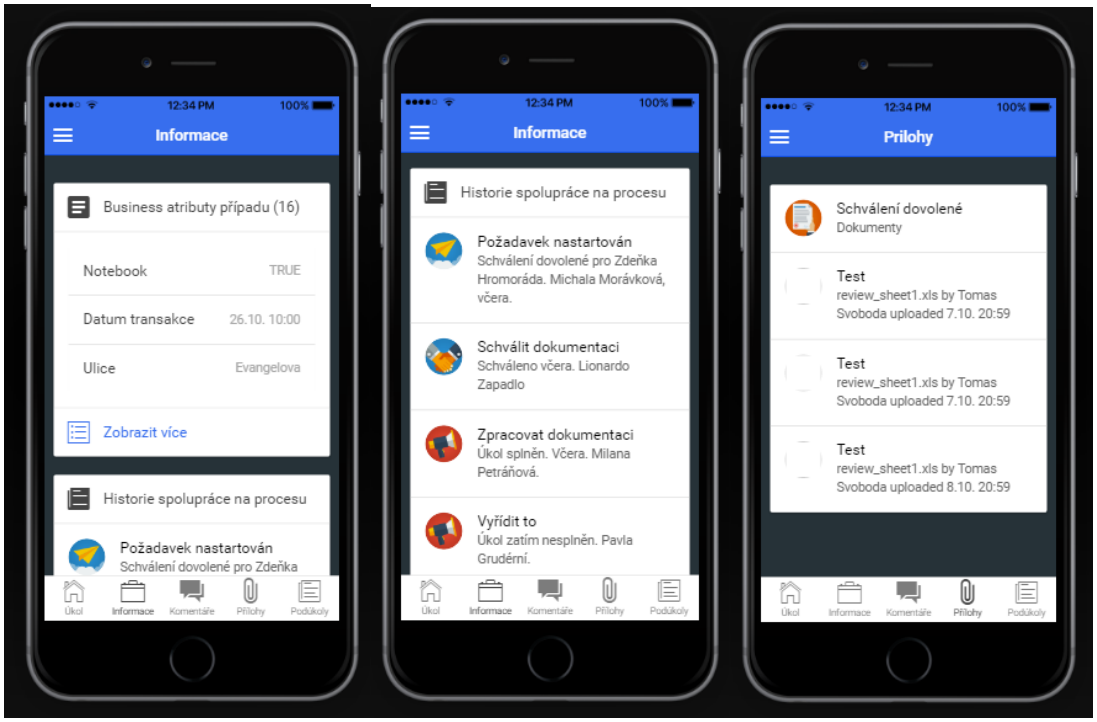
Na obrazovkách z obrázku 28 je vidět obsah jednotlivých záložek. Některé obrazovky jsou rozdělené na dva obrázky, protože byly tak dlouhé, že se na jednu obrazovku nevešli. První záložka *Úkol* zobrazuje informace o úkolu F013, zobrazuje business atributy požadavku F014 a úkol na ní lze dokončit F022 tlačítek “úkol zpracován”. Druhá záložka „Komentáře“ zobrazuje komentáře k požadavku F011 a Lze přidat komentář F012.



Obrázek 28:
Wireframes: Úkol – záložky detail, komentáře [autor]

Obrazovky detailu úkolu 2

Na obrazovkách z obrázku 29 jsou vidět další záložky úkolu. Na první záložce informace lze vidět business atributy případu F015. Prostřední obrazovka také zobrazuje záložku informace, ale je posunutá níže, kde je vidět historie požadavku F017. Obrazovka vpravo zobrazuje přílohy. Souvisí tedy s funkčními požadavky dokumentů F009, F010.



Obrázek 29:
Wireframes: Úkol – záložky informace, přílohy [autor]

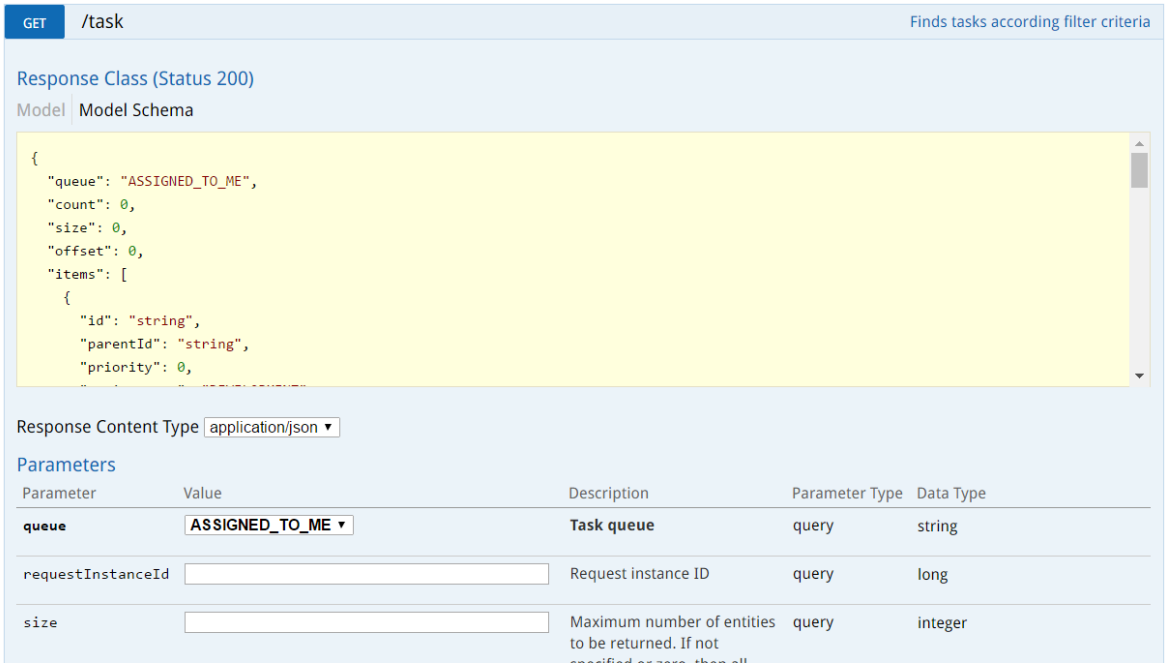
Obrazovky detailu úkolu 3

Na obrazovce z obrázku 30 je vidět záložka “Pod-úkoly“, která souvisí s funkčním požadavkem F016 a zobrazuje status, prioritu, řešitele, datum splnění, název pod-úkolu a popis.

Obrázek 30:
Wireframes: Úkol – záložka Podúkoly [autor]

REST API

Pro komunikaci s backendem aplikace je připraveno rozhraní REST API, které je namodelované v programu Swagger [39]. Bohužel volání osahují tolik informací, že se do diplomové práce z důvodu rozsahu nevejdou. Proto je zde pouze ukázka na obrázku 31.



Obrázek 31:
Ukázka: JSON REST API PowerFLOW services - Swagger [autor]

Využité REST API PowerFLOW services

V aplikaci PowerFLOW mobile jsou využívány následující REST API služby z PowerFLOW services:

- POST /auth – přihlásí uživatele
- DELETE /auth – odhlásí uživatele
- GET /system – services vrátí data o systému
- GET /task – services vrátí fronty úkolů
- POST /task/{id}/claim – uživatel se přihlásí k řešení úkolu
- POST /task/{id}/unclaim – uživatel se odhlásí od řešení úkolu
- GET task/{id} – services vrátí detail úkolu
- GET /user – services vrátí detail přihlášeného uživatele
- GET /task/{parentTaskId}/subtask – services vrátí seznam podúkolů úkolu
- GET /request-instance/{requestInstanceId}/post – services vrátí seznam komentářů k běžícímu požadavku

- POST /request-instance/{requestInstanceId}/post – přidá nový komentář k požadavku
- GET /document/{documentId}/version – services vrátí verze dokumentů
- GET /document/{documentId}/version/{version} – services vrátí soubor ke stažení
- GET /case-type/{id} – services vrátí detal případu, ve kterém jsou uloženy business atributy případu
- GET /request-instance/{requestInstanceId}/history – vrátí historii běžícího požadavku
- POST /phase/task/task/resolve/{id} – označí fázi úkol za splněnou
- POST /phase/recommending/task/recommending/{id} - označí fázi doporučení za splněnou
- POST /phase/approval/task/approval/{id} - označí fázi schvalování za splněnou

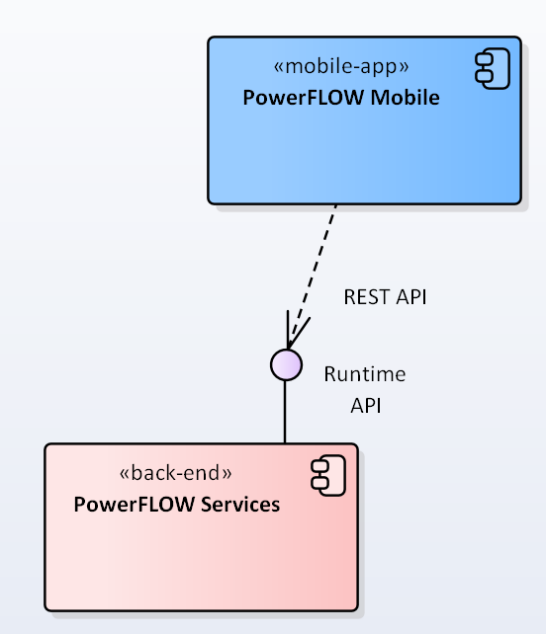
Použité REST API metody

HTTP REST API využívá specifické operace k příkazům s podobnou funkcionalitou. Zde jsou vedeny metody využívané aplikací PowerFLOW mobile.

- GET – nedestruktivně vrátí data ze serveru
- POST – vytvoří nový záznam do databáze nebo provede významnou změnu
- DELETE – smaže záznam z databáze, v této aplikaci je příkaz využíván pro odhlášení

Diagram komponent mobilní aplikace

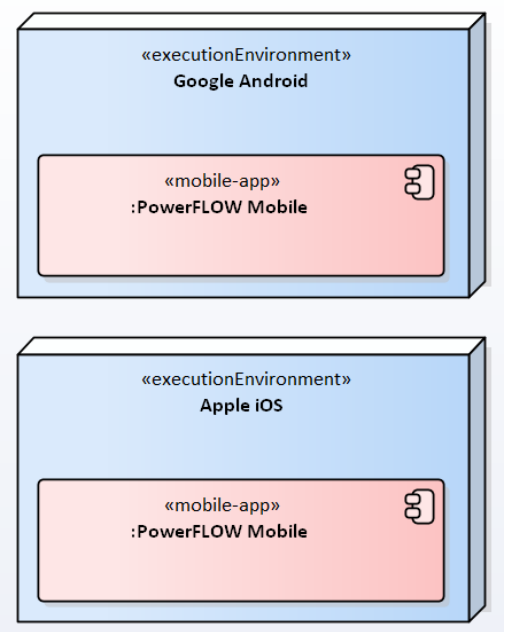
Na obrázku 32 je vidět diagram komponent mobilní aplikace PowerFLOW mobile. Mobilní aplikace komunikuje přes REST API zabezpečenou komunikací https s backend částí systému PowerFLOW Services. Podrobnější obrázek diagramu komponent lze nalézt v předcházející části. Byl by rozšířen o komponentu PowerFLOW mobile – viz níže.



Obrázek 32:
Diagram komponent aplikace PowerFLOW mobile [autor]

Diagram nasazení mobilní aplikace

Na obrázku 33 lze vidět diagram nasazení mobilní aplikace na zařízení Android a iOS. Podrobnější obrázek diagramu nasazení lze nalézt v předcházející části. Byl by rozšířen o dva mobilní přístroje – viz níže.



Obrázek 33:
Diagram nasazení aplikace PowerFLOW mobile [autor]

6 Popis vytvořené mobilní aplikace PowerFLOW mobile

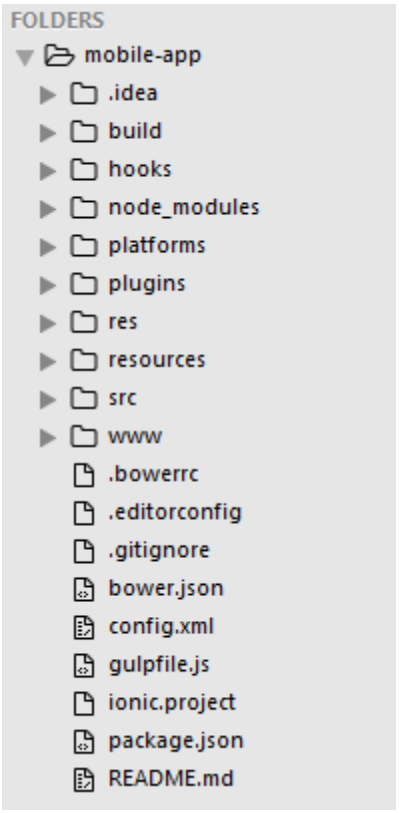
Tato část popisuje vytvořenou aplikaci PowerFLOW mobile. Je zde rozebrána struktura projektu a ukázky kódu.

6.1 Struktura projektu

Na obrázku 34 je zobrazena struktura projektu aplikace PowerFLOW mobile vyvíjené ve frameworku IONIC. Jsou zde rozebírány složky v kořenovém adresáři projektu.

Popis důležitých složek projektu:

- Složka *build* – v této složce je uložený soubor *config-defaults.js*, ve kterém jsou zaznamenány soubory používané v projektu. Soubor obsahuje cesty ke knihovnám, skriptům, sass stylům a šablonám.
- Složka hooks – obsahuje uživatelské příkazy pro ovlivnění vývojového procesu Cordovi.
- Složka *node_modules* – obsahuje knihovny projektu.
- Složka *platforms* – obsahuje zdrojové soubory jednotlivých operačních systému, lze zde přizpůsobovat aplikaci pro jednotlivé platformy před uvedením do produkce.
- Složka *plugins* – obsahuje uložené moduly Cordovi.
- Složky *res* a *resources* – obsahuje obrázky IONIC frameworku.
- Složka *src* – obsahuje samotný kód aplikace, v této složce tráví programátor nejvíce času při vývoji. Jsou zde uloženy kontrolery, šablony, cesty, obrázky a styly aplikace.
- Složka *www* – obsahuje projekt po aplikování gulpu, tedy minimalizovaný a připravený pro nasazení.



Obrázek 34:
Základní struktura projektu PowerFLOW mobile [autor]

6.2 Moduly v aplikaci PowerFLOW mobile

Moduly jsou v aplikaci důležité, protože nám projekt rozdělují do celků s ucelenou funkcionalitou. Na obrázku 35 je obsah souboru app.module.js, který propojuje moduly využívané v aplikaci. V první části jsou moduly Ionic, ngCordova a angular-jwt. Jsou to základní stavební kameny pro vývoj aplikace v IONIC frameworku.

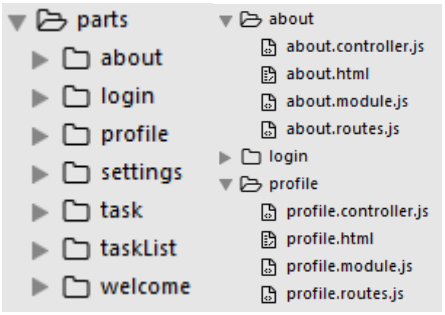
- Modul ionic přidává funkcionalitu pro zobrazování mobilních elementů v UI a přizpůsobuje javascript pro mobilní vývoj.
- Modul ngCordova se stará o komunikaci mezi aplikací a mobilním zařízení a umožňuje využívat nativní funkce zařízení.
- Modul angular-jwt přidává javascriptové knihovny frameworku Angular
- myApp.core je modul, vytvořený inicializací projektu.
- ntx.powerflow-client-services je modul zprostředkovávající komunikaci s PowerFLOW services.
- Další moduly jsou rozděleny podle navrhnutých obrazovek.

```
(function() {
 'use strict';

 angular
  .module('myApp', [
      'ionic',
      'ngCordova',
      'angular-jwt',

      'myApp.core',

      //NTX modules
      'ntx.powerflow-client-services',

      // parts
      'myApp.login',
      'myApp.profile',
      'myApp.settings',
      'myApp.taskList',
      'myApp.welcome',
      'myApp.task',
      'myApp.about'
  ]);
})();
```

Obrázek 35:
Moduly v aplikaci PowerFLOW mobile [autor]

6.3 Struktura stránek

Každá stránka v aplikaci musí být složena nejméně ze čtyř částí. Aby byl projekt přehlednější, tak jsou stránky se samostatnou funkcionalitou a jejich součásti umístěny do složek, podle metody folder-by-feature. Struktura složek je zobrazena na obrázku 36.



Obrázek 36:
Základní struktura projektu PowerFLOW mobile [autor]

Obsah stránky

Každá stránka s funkčností obsahuje:

- Controller – je psaný v javascriptu a stará se o funkcionalitu na stránce.
- Šablona – je napsaná v html a stará se o zobrazování stránek uživateli.
- Modul – určuje, ve kterém modulu se stránka nachází.
- Routes – soubor definující URL cestu ke stránce a propojuje soubory stránky.

Struktura folder-by-feature

Aplikace využívá struktury folder-by-feature, která má následující výhody:

- Je jen pár souborů ve složce
- Hledání ve stromové struktuře je lehké
- Vývojáři mohou pracovat odděleně na různých částech kódu
- Nová stránka se dá vytvořit zkopírováním podobné stránky
- Název složky vypovídá o tom, co složka obsahuje za funkcionalitu
- Je dobře rozšiřitelná i pro větší tým

6.4 Ukázka přihlášení

Před samotným používáním mobilní aplikace je potřeba uživatele autentifikovat. K tomuto účelu je v aplikaci přihlašovací obrazovka. V této části jsou popsány jednotlivé části kódu, které jsou k přihlašování použity.

Šablona

Na obrázku 37 je zobrazena část kódu z přihlašovací obrazovky. Jedná se o formulář pro vyplnění uživatelského jména a hesla. Tato část kódu je uložena v šabloně s názvem *login.html*. Jedná se tedy o HTML soubor, ve kterém jsou prvky zobrazující se uživateli. Na stránce jsou dvě vstupní pole. Jsou to tagy input. Aby se hodnoty, které uživatel zadal dostali do kontroleru je potřeba definovat jejich názvy. To zajišťuje atribut *ng-model*.

Názvy proměnných přenesených do kontroleru:

- credentials.username
- credentials.password

Metoda, která se zavolá po kliknutí je definována atributem *ng-click.* Jedná se o metodu *authenticate*. V ukázce lze ještě nalézt atribut *class*, který definuje styly zobrazování.

Vysvětlení stylování tlačítka „Pokračovat“

Styl zobrazování je definován atributem class="button button-positive“. Tyto styly jsou součástí IONIC frameworku. V případě, že by u tlačítka nebyl parametr class, tak by se tlačítko zobrazovalo pouze jako hypertextový odkaz. Vysvětlení jednotlivých parametrů:

- button – určuje, že to je tlačítko
- button-positive – určuje, že tlačítko bude mít modrou barvu
- button-block – určuje, že tlačítko bude v šířce od levé strany až po pravou

Ukázka vzhledu obrazovky je v části wireframes s názvem *Wireframes: Přihlašovací obrazovky*

```
<form class="list">
  <ion-list>
    <label class="item item-input item-floating-label">
      <span class="input-label">Uživatelské jméno</span>
      <input type="text" placeholder="Uživatelské jméno" ng-
model="credentials.username">
    </label>
    <label class="item item-input item-floating-label">
      <span class="input-label">Heslo</span>
      <input type="password" placeholder="Heslo" ng-
model="credentials.password">
    </label>
  </ion-list>
  <div class="spacer" style="height: 40px;"></div>
  <a ng-click="authenticate()"  class="button button-positive  button-
block">Pokračovat</a>
</form>
```

Obrázek 37:
Část šablony ze souboru login.html pro přihlašování [autor]

Kontroler

Na obrázku 38 je zobrazena funkce authenticate, která se provede v případě stisku tlačítka „Pokračovat“ v přihlašovací obrazovce. Na začátku se provede načtení přihlašovacích údajů z formuláře do proměnných *username* a *password.* Poté se připraví hlavička requestu headers pro Basic autentifikaci. Využívá se v ní převod uživatelského jména a hesla do Base64 kódování. Následně je zavolána funkce *authenticate*, která má jako vstupní parametr vytvořenou hlavičku. Funkce zavolá přes REST API metodu POST /auth na PowerFLOW services. V případě, že se vrátí z PowerFLOW services status 200 kód po-

kračuje následujícím řádkem, kde se získá z odpovědí autorizační token a přihlášený uživatel, který se uloží do Local Storage. Local Storage je služba zpřístupněná s HTML 5, která umožňuje ukládat hodnoty do webového prohlížeče. Díky tomuto uložení se v dalších requestech už nemusí uživatel přihlašovat. V rámci každého requestu se posílá autorizační token, takže PowerFLOW services z něj je schopné určit, zda je uživatel pro přístup autorizovaný. V další části se nastaví přesměrování na obrazovku zobrazující „Moje úkoly“. Když se volání funkce authenticate vrátí s jiným statusem než 200 provede se zalogování do konzole s výpisem chyby.

```
$scope.authenticate = function() {

  var username = $scope.credentials.username;
  var password = $scope.credentials.password;

  window.localStorage.setItem("credentials.username", username);

  var headers = {
    'Authorization' : 'Basic ' + Base64.encode(username + ':' + password)
  };

  Authentication.authenticate(headers, function(response, status) {

    var token = response.data.token;
    var user = response.data.user;

    if (token != null && user != null) {

      window.localStorage.setItem('credentials.token', token);
      window.localStorage.setItem('credentials.user', user);

      $state.go("myTaskList");
    }
  }, function(data, status) {
        $log.debug("Error: %s)", data);
  });

};
```

Obrázek 38:
Část souboru login.controller.js pro přihlašování [autor]

Modul

Každá část kódu musí být v nějakém modulu. Proto je v souboru login.module.js následující část kódu – viz obrázek 39. Hlavní modul aplikace je 'myApp', proto všechny moduly, které jsou vytvořené specificky pro tuto aplikaci musí být vnořené.

```
(function() {
 'use strict';

 angular
  .module('myApp.login', ['templates-cache']);
})();
```

Obrázek 39:
Část souboru login.module.js pro přihlašování [autor]

Cesta

Cesty v aplikaci jsou definovány v souborech routes pomocí stavů. Na obrázku 40 je vidět zápis v souboru login.routes.js. Každý stav musí obsahovat šablonu, definovanou URL adresu a jaký kontroler se má použít.

```
(function() {
 'use strict';

 angular
  .module('myApp.login')
  .config(config);

 config.$inject = ['$stateProvider'];

 function config($stateProvider) {

    $stateProvider.state('login', {
      url: '/login',
      templateUrl: 'parts/login/login.html',
      controller: 'LoginCtrl'
    });

 }
})();
```

Obrázek 40:
Část souboru login.routes.js pro přihlašování [autor]

6.5 Ukázka využití komponenty slider

IONIC komponenta slider umožňuje do obrazovky vložit okno, ve kterém se dá přepínat mezi obrázky přejetím prstu na levou nebo pravou stranu. Komponenta je využita před přihlášení do aplikace. Na obrázku 41 lze vidět její použití. Vnější html tag je *ion-slides* a vnitřní tagy jsou *ion-slide-page*, které nemají žádný obsah, ale mají vyplněné atributy. IONIC pro komponenty využívá AngularJS direktivy.

```
<ion-view title="Slider"  hide-nav-bar="true" hide-back-button="true"
id="page22">
  <ion-content padding="true">


    <ion-slides disable-side-menu-drag="" options="{'loop': false}"
slider="slider1" id="slider-slider1" style="width:100%;height:500px;">
      <ion-slide-page id="slider-slide25" sty-
le="background:url(/assets/img/slide1.PNG) no-repeat center;background-
size:cover;"></ion-slide-page>
      <ion-slide-page id="slider-slide26" sty-
le="background:url(/assets/img/slide2.PNG) no-repeat center;background-
size:cover;"></ion-slide-page>
      <ion-slide-page id="slider-slide24" sty-
le="background:url(/assets/img/slide3.PNG) no-repeat center;background-
size:cover;"></ion-slide-page>
      <ion-slide-page id="slider-slide23" sty-
le="background:url(/assets/img/slide4.PNG) no-repeat center;background-
size:cover;"></ion-slide-page>
    </ion-slides>



    <button class="button button-full button-positive" ui-
sref="loginUrl">
      Pokračovat k přihlášení
    </button>

  </ion-content>
</ion-view>
```

Obrázek 41:
HTML kód obrazovky s komponentou IONIC slider [autor]

6.6 Testování mobilní aplikace

Aplikace byla testována v živém náhledu webového prohlížeče a vzdáleným přístupem na mobilním telefonu. K požadavků byly vytvořeny testovací scénáře, které nejsou součástí diplomové práce. Jejich exekuce probíhala na několika testovacích prostředích. První prostředí bylo v okně vývojářské konzole webového prohlížeče Chrome, které umožňuje sledovat konzoly, REST volání a také uložené proměnné v *Local Storage* prohlížeče. Tento způsob testování je vhodný při vývoji aplikace. Je rychlý, změny jsou vidět na spuštěné aplikaci v řádu vteřin. Druhé prostředí bylo na mobilním telefonu Android přesměrováním portu běžící aplikace ze stolního počítače. Testování tímto způsobem je vhodné pro zjištění uživatelské přívětivosti aplikace, umožnuje si vyzkoušet komfort zadávání textu a ovládání telefonem nicméně neodhalí chyby, které se se projeví až po spuštění aplikace na specifických platformách. Proto se provádělo ještě akceptační testování. Při něm se využívala aplikace *IONIC view* pro telefony Android a iOS [Dostupná z http://view.ionic.io/]. Pomocí aplikace *IONIC view* lze otestovat vytvořenou aplikaci bez nutnosti vytvoření instalačních balíčků pro jednotlivá zařízení.

Díky tomuto testování se odhalily následující chyby, které nejsou zjistitelné pomocí předchozích testů:

- Špatná práce se stavy aplikace – nefungující odkaz.
- Špatně použitá cesta k obrázkům – obrázky se nenačítaly.
- Nepovolené URL adresy do internetu – nefungovala komunikace REST API.
- Nepovolené načítání souborů ze systému – obrázky se nenačítaly.

K závěrečnému testování je vhodné vytvořit instalační balíčky pro jednotlivá zařízení nebo použít IONIC view. Obě varianty se chovají stejně.

Tabulka 14 zobrazuje funkční požadavky a hodnoty výstupů akceptačních testů po opravě předchozích chyb. Výstupy jsou z množiny {Prošel, Neprošel, N/A}, kde N/A je označení zkratky " not available", tedy že test nebyl uskutečněn.

Tabulka 14: *Záznam výsledků testů aplikace PowerFLOW mobile*

ID požadavku	**Název požadavku**	**Implementováno**	**Test: Android**	**Test: iOS**
F001	Prohlédnutí vstupních obrazovek	Ano	Prošel	Prošel
F002	Zadání URL adresy backendu	Ano	Prošel	Prošel
F003	Změna URL adresy backendu	Ano	Prošel	Prošel
F004	Přihlášení do systému	Ano	Prošel	Prošel
F005	Odhlášení	Ano	Prošel	Prošel
F006	Zobrazení seznamu „Mé úkoly“	Ano	Prošel	Prošel
F007	Zobrazení seznamu „Týmové úkoly“	Ano	Prošel	Prošel
F008	Zobrazení informací přihlášeného uživatele	Ano	Prošel	Prošel
F009	Zobrazit seznam dokumentů k požadavku	Ano	Prošel	Prošel
F010	Stažení poslední verze dokumentu u požadavku	Ne	N/A	N/A
F011	Zobrazit seznam komentářů k požadavku	Ano	Prošel	Prošel
F012	Přidání komentáře k požadavku	Ano	Prošel	Prošel
F013	Zobrazení informací o úkolu	Ano	Prošel	Prošel
F014	Zobrazení business atributů požadavku	Ano	Prošel	Prošel
F015	Zobrazení business atributů případu	Ano	Prošel	Prošel

F016	Zobrazení seznamu pod-úkolu úkolu	Ano	Prošel	Prošel
F017	Zobrazení historie požadav-ku	Ne	N/A	N/A
F018	Přihlášení k řešení úkolu (claim)	Ano	Prošel	Prošel
F019	Odhlášení se z řešení úkolu (unclaimed)	Ano	Prošel	Prošel
F020	Editace business atributů požadavku	Ne	N/A	N/A
F021	Editace business atributů případu	Ne	N/A	N/A
F022	Dokončení úkolu	Ano	Prošel	Prošel

6.7 Ukázka obrazovek nasazené aplikace

V příloze A*: Ukázka obrazovek* lze vidět ukázky tří obrazovek nasazené aplikace PowerFLOW mobile na mobilních operačních systémech Android a iOS. Během testování proběhla i analýza vzhledu prvků uživatelského rozhraní pro oba operační systémy. Jediný prvek, který se výrazně odlišoval je menu vytvořené v detailu úkolu. V mobilním operačním systému iOS je zmíněné menu ve spodní části obrazovky. U operačního systému Android, se menu nachází v horní části obrazovky. Ostatní grafické prvky uživatelského rozhraní jsou umístěny na stejných pozicích. Rozdílné je zadávání textu. Každé zařízení má svojí klávesnici, která je závislá i na modelu zařízení.

7 Zhodnocení frameworku IONIC

Kapitola má tři části. V první části provádím hodnocení frameworku IONIC podle kritérií, které definuji. V druhé části jsou uvedeny zkušenosti z praktické implementace projektu a třetí část je celkové shrnutí této kapitoly.

7.1 Hodnocení podle kritérií

V této části jsem stanovil kritéria hodnocení podle nabytých zkušeností z analýzy frameworku IONIC a vlastních poznatků z vývoje end-to-end projektu PowerFLOW mobile ve frameworku IONIC. Ke stanovení kritérií mi také pomohli jak vlastní zkušenosti, tak práce zabývající se porovnávání frameworků. Čerpal jsem z prací *Nástroje pro podporu vývoje nativních multiplatformních mobilních aplikací* [18], *Multiplatformní vývoj mobilních aplikací pomocí webových technologií* [17] a *Analýza frameworků pro vývoj multiplatformních mobilních aplikací využívající HTML technologie* [14]. Nejdříve kritéria definuji a stanovuji způsob jejich hodnocení. Následně jsou jednotlivá kritéria vyhodnocena a na konec je uveden celkový přehled kritérií.

Definice kritérií

Zde jsou definována kritéria. Kritéria pomůžou rozhodnout, zda je projekt vhodný realizovat v IONIC frameworku. Kritéria jsou samo vysvětlující a pokud není nějaké jasné, tak se lze podívat do jeho vyhodnocení, kde je popsáno, jakým způsobem proběhlo.

Stanovená kritéria:

- cena
- vhodnost pro různé druhy aplikací
- podporované nativní funkce
- budoucnost frameworku
- vzhled UI frameworku
- pokrytí mobilních operačních systémů
- dostupnost vývojářů z hlediska software vendor firmy
- rychlost UI mobilní aplikace
- rychlost adaptace vývojáře
- náročnost instalace a přípravy systému na vývoj

- testování
- publikace aplikace
- dokumentace
- komunita a podpora
- znovu použitelnost komponent
- pluginy a moduly třetích stran
- integrace frameworku do IDE

Hodnocení kritérií

Kritéria jsou hodnocena ze stupnice jedna až pět. Hodnocení probíhá jako ve škole:

- 1 – výborný
- 2 – chvalitebný
- 3 – dobrý
- 4 – dostatečný
- 5 – nedostatečný

Vyhodnocení kritérií

Zde jsem popsal a subjektivně vyhodnotil všechna definovaná kritéria na základě vlastních zkušeností získaných z analýzy frameworku IONIC, implementace aplikace PowerFLOW mobile ve frameworku IONIC, mé vývojářské praxe a zmíněných zdrojů. Vždy je nejdříve název kritéria, na dalším řádku následuje jeho ohodnocení a potom je uveden důvod a popis hodnocení.

Cena

Hodnocení: 1

V tabulce 15 jsou zobrazeny licence použitých frameworků a technologií při implementaci ve frameworku IONIC. Všechny licence jsou volně k používání. IONIC je tedy zdarma. Ohodnotil jsem toto kritérium číslem jedna.

Tabulka 15: *Technologie a licence frameworku IONIC*

Název frameworku	**Licence**
IONIC	MIT Licence

AngularJS	MIT Licence
Apache Cordova	Apache License
Git	Open source
Bower	MIT Licence
Node.js	Expat License, MIT licence
Sass	MIT Licence
Gulp	MIT Licence

Vhodnost pro různé druhy aplikací

Hodnocení: 2

Podle článku *Different Types Of Apps That App Developers Need To Know About* [40] je 7 druhů aplikací publikovaných na Google Play a Apple storu. V následující tabulce je zobrazeno, zda jde pro vývoj těchto kategorií aplikací využít framework IONIC. Aplikace vytvořené ve frameworku IONIC jsou psané programovacím jazykem javasript a frameworkem AngularJS. Tyto technologie využívá velká komunita a existuje mnoho knihoven třetích stran, které se dají v projektu využít. Framework IONIC je spíše vhodný pro aplikace, které mají složité a strukturované UI. Není přímo vhodný pro vytváření 3D her a práci s grafikou, ale díky velké základně uživatelů Javascriptu, existují frameworky, které se dají pro práci s grafikou použít. Například framework Phaser dostupný z: https://phaser.io/. IONIC neumožňuje sám o sobě vytvořit widget na plochu mobilního OS, ale na internetu jsou návody, jak využít Apache Cordova pro vytvoření widgetu. Nicméně díky těmto komplikacím dávám známku 2. Na obrázku 16 je vyhodnocena vhodnost využití frameworku IONIC pro vývoj různých kategoriích aplikací.

Definované kategorie aplikací a jejich příklady:

- Nástroje: kalkulačka, poznámkový blok, komunikační aplikace, aplikace s počasím
- Zábava: aplikace na prohození obličeje, aplikace na výuku angličtiny, streamování videa
- Hry: střílení ptáků, závody aut, kolo štěstí
- Noviny: magazín fotek, aplikace na čtení zpráv
- Produktivita: finanční aplikace, sdílení souborů, překladače
- Životní styl: fitness aplikace, cestovatelské, hodnocení jídla
- Sociální sítě: sdílení obsahu, komunikace, Instagram, Google+

Tabulka 16: *Kategorie aplikací a jejich vhodnost pro vývoj ve frameworku IONIC*

Kategorie aplikace	**Vhodnost vývoje aplikace v IONIC frameworku**	**Omezení**
Nástroje	Ano s omezením	Bez widgetů, nutná implementace pro specifickou platformu.
Zábava	Ano	
Hry	Ano s omezením	IONIC framework samotný nepodporuje grafiku a 3D.
Noviny	Ano	
Produktivita	Ano	
Životní styl	Ano	
Sociální sítě	Ano	

Podporované nativní funkce

Hodnocení: 1

Přehled podporovaných nativních funkcí je uvedený v tabulce *Přehled základních pluginů Apache Cordova* ve čtvrté kapitole. Apache Cordova zprostředkovává komunikaci mezi vyvíjenou mobilní aplikací a specifickým operačním systémem a umožňuje využívat jeho nativní funkce. Je k dispozici 20 pluginů obsluhující oblasti nativních funkcí. V případě nutnosti lze vytvořit plugin pro customizaci používání frameworku Apache Cordova. Proto hodnotím kritérium známkou jedna.

Budoucnost frameworku

Hodnocení: 1

Framework IONIC vznikl v roce 2013, od té doby má širokou základnu uživatelů, kteří ho používají. Pro jeho chod využívá zisk z placených funkcí, které usnadňují jeho užití. Nyní je IONIC 2 ve verzi beta. Aplikace napsané ve frameworku IONIC 2 lze publikovat i pro mobilní operační systém Windows 10. Z dostupných informací je zřejmé, že framework jen tak nezanikne, a proto dávám známku 1.

Vzhled UI frameworku

Hodnocení: 1

Jedna ze silných stránek frameworku IONIC je v jednoduchém používání jeho CSS komponent. Komponenty jsou již nastylovány tak, aby vypadali jako nativní komponenty mobilních operačních systémů. Pokud není potřeba vytvořit obrazovky se speciálními prvky na které je nutné vytvoření nových stylů, tak lze vytvořit aplikace bez toho, aniž by programátor přepisoval CSS styly nebo používal SASS. Hodnotím tedy za jedna.

Pokrytí mobilních operačních systémů

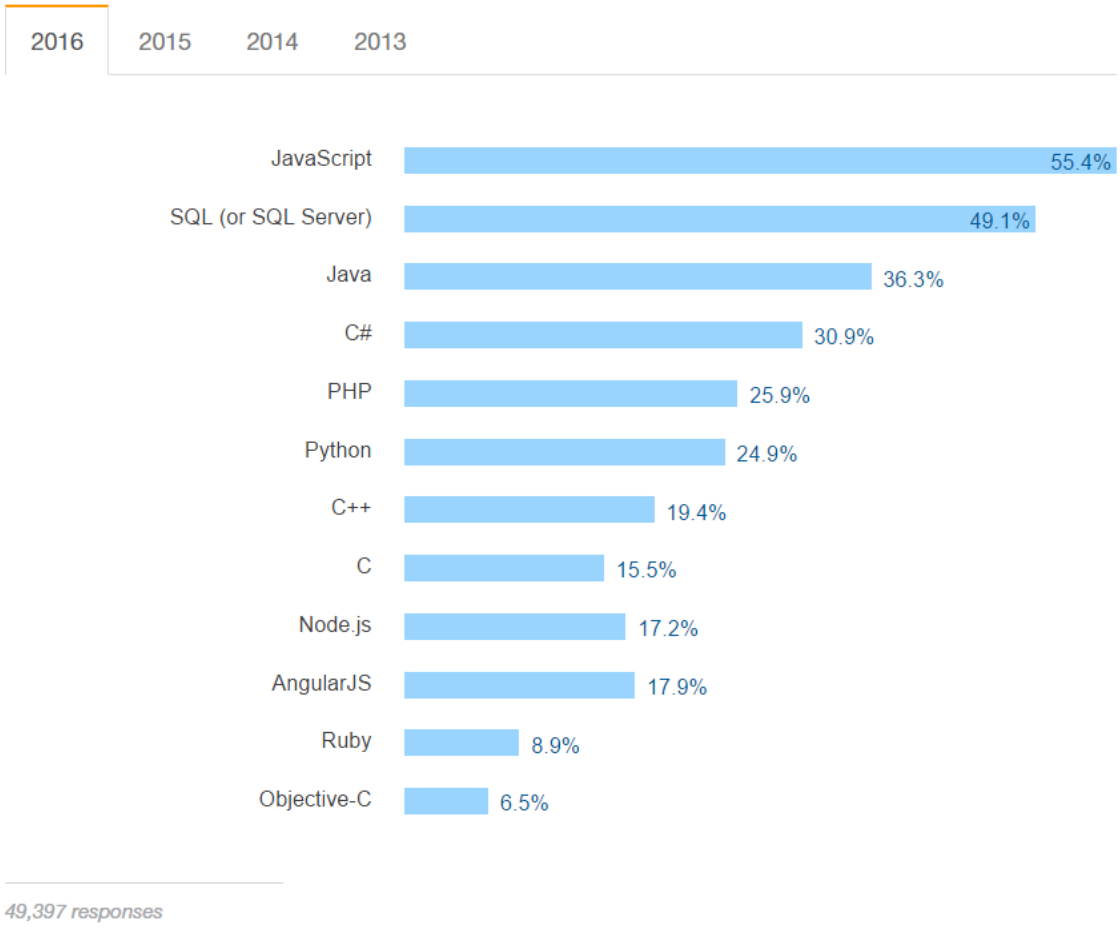
Hodnocení: 1

Na obrázku *Mobile/Tablet Operating System Market Share* v třetí kapitole je graf, který jasně vypovídá o majoritních mobilních operačních systémech na trhu. Je zde vidět, že v roce 2016 převažuje platforma Android s 69 procenty a platforma iOS s 26 procenty. Ostatní mobilní operační systémy mají minoritní podíl na trhu – dvě procenta a méně. IONIC 1 podporuje mobilní OS Android a iOS. Nastupující IONIC 2, který je momentálně ve verzi beta již podporuje operační systém Windows 10, který má podle průzkumu 2 procenta na trhu. Z tohoto příkladu je vidět, že IONIC reaguje na mobilní trh a přidává podporu pro významné mobilní platformy. Proto hodnotí známkou jedna.

Dostupnost vývojářů z hlediska software vendor firmy

Hodnocení: 1

Pro firmu vyvíjející software je důležitým faktorem výběru frameworku pro implementaci i dostupnost vývojářů na trhu. Na obrázku 42 je zobrazen graf s nejvíce populárními vývojovými technologiemi z průzkumu z roku 2016, ve kterém odpovědělo padesát tisíc respondentů z celého světa. Hlavní technologie použité ve frameworku IONIC jsou na tomto žebříčku, takže by neměl být problém najít vhodné kandidáty na pozici IONIC vývojáře. Podle široké základny uživatelů lze také usoudit, že na internetu bude dostatek materiálů. Hodnotím tedy známkou 1.



Obrázek 42:
Nejvíce populární vývojové technologie v roce 2016 [38]

Rychlost UI mobilní aplikace

Hodnocení: 1

Rychlost uživatelského rozhraní aplikace lze velmi těžko měřit, ale myslím si, že by zde toto kritérium nemělo chybět. Při testovaní mobilní aplikace PowerFLOW mobile jsem nenarazil na problémy s rychlostí uživatelského rozhraní, nicméně lze usoudit, že nativní aplikace vyvíjené pro každý operační systém zvlášť budou nepatrně rychlejší. Jiné mobilní aplikace vyvíjené v HTML 5 mají zpoždění 300ms od kliknutí uživatelem. Je to proto, že prohlížeč čeká, jestli uživatel provede dlouhý dotek na prvku UI. Dlouhý dotek slouží na příklad pro rozbalení menu, otevření nabídky apod. 300ms zpoždění není u mobilních aplikací žádoucí a knihovny IONIC přepisují komponentu pro sledování dotyků a tento neduh odstraňují. Zdroj: *Hybrid Apps and the Curse of the 300ms Delay* [41]. Proto hodnotím známkou 1.

Rychlost adaptace vývojáře

Hodnocení: 2

Framework IONIC využívá mnoho technologií, které se obecně používají pro frontend vývoj aplikací. Pokud je s těmito technologiemi vývojář seznámen, tak se adaptuje snadno. Technologie, které je vhodné znát jsou definované v tabulce *Znalosti technologií pro vývoj v IONIC frameworku*. Adaptaci pomůže rozsáhlá a přehledná dokumentace frameworku IONIC včetně návodů.

V následující tabulce jsou vypsány technologie, se kterými se vývojář, pracující s frameworkem IONIC setká na základě mých vlastních zkušeností projektu. Je zde definována náročnost technologie z množiny {malá, střední, velká}, dopad na vývoj z množiny {malý, střední, velký} a znovu použitelnost technologie mimo IONIC framework.

V tabulce 17 je popsáno, jaký vliv může mít dopad na vývoj, pokud vývojář není seznámen s určitou technologií.

IONIC používá velké množství technologií. Proto je ze začátku pro vývojáře, který ještě neimplementoval frontend řešení obtížné se v technologiích zorientovat. Nicméně stejné nebo podobné technologie se používají ve většině frontend řešeních i pro desktopové aplikace. IONIC představuje ucelený ekosystém, který má všechny základní technologie již integrované. Je tedy jednodušší technologie začít používat. Adaptaci pomáhá i vytvoření projektů základními šablonami. Viz *Inicializace projektu* ve čtvrté kapitole. Vývojář má možnost vytvořit si jeden ze tří ukázkových a plně funkčních projektů, na kterých si může vyzkoušet a prozkoumat, jak IONIC funguje. Hodnotím tedy známkou 2.

Tabulka 17: *Znalosti technologií pro vývoj v IONIC frameworku*

Technologie	**Náročnost**	**Dopad na vývoj**	**Znovupoužitelnost**
Javascript	velká	velký	Frontend, backend
AngularJS	velká	velký	Frontend, backend
HTML	malá	velký	Frontend
Gulp	malá	malý	Frontend, backend
Bower	malá	střední	Frontend, backend
Apache Cordova	velká	střední	Pouze na projektech využívající multiplatformní vývoj
Sass	střední	střední	Frontend
CSS	střední	Střední	Frontend

Git	malá	střední	Frontend, backend
Node.js	malá	malý	Frontend, backend

V tabulce 18 jsou popsány technologie a ovlivnění vývoje, pokud vývojář technologii neovládá.

Tabulka 18: *Znalosti technologií a jejich dopad pro vývoj v IONIC frameworku*

Technologie	**Možný dopad na vývoj ve frameworku IONIC, pokud vývojář technologii neovládá**
Javascript	Nepochopení struktury projektu. Problém s implementací funkcionality.
AngularJS	Nepochopení struktury projektu. Problém s implementací funkcionality.
HTML	Problém s vytvářením šablon obrazovek.
Gulp	Problém s vygenerováním specifického spustitelného kódu.
Bower	Problém s instalací modulů a balíků třetích stran.
Cordova	Problém s implementací nativních funkcí.
Sass	Problém s vytvořením specifických komponent UI.
CSS	Problém s vytvořením specifických komponent UI.
Git	Problém s verzováním kódu. Problém s kooperací v týmu.
Node.js	Není

Náročnost instalace a přípravy systému na vývoj

Hodnocení: 1

Kroky instalace a inicializace projektu jsou uvedeny v části *4.2 Instalace, nastavení vývojového prostředí*. Jedná se o postup sedmi kroků, které je možné realizovat do 30 minut. Hodnotím tedy známkou jedna.

Testování

Hodnocení: 1

Testování je popsáno ve čtvrté kapitole. Během vývoje je možné testovat aplikaci živým náhledem v prohlížeči. Pro zobrazení aplikace na telefonu lze použít USB kabel. Tyto způsoby testování jsou vhodné pro vývoj. Velkým plusem je testování pomocí aplikace *IONIC view*. Tato aplikace umožňuje nahrání implementované aplikace do *IONIC cloudu* a je možné spustit a testovat aplikaci v telefonu bez nutnosti vytvářet instalační balíčky pro jednotlivé systémy. Během testování principe *IONIC view* jsem odhalil chyby, které testováním v prohlížeči nebyly zjistitelné. Chyby se projevily až po spuštění aplikace na mobilním telefonu. Chyby jsou popsány také ve čtvrté kapitole. Pro dlouhodobou údržbu aplikace jsou podstatné kromě uživatelských testů i jednotkové. Funkcionalita framework IONIC je postavená na Javascriptu a frameworku AngularJS. S těmito technologiemi jde provádět jednotkové testy. Hodnotím testování známkou jedna.

Publikace aplikace

Hodnocení: 1

Kritérium vyhodnocuji podle článku Publishing your app [42] z dokumentace frameworku, protože jsem sám ještě neměl příležitost publikaci vyzkoušet. Pro publikaci do internetových obchodů Google Play a Apple store je nutná placená registrace. Pro Google Play stojí registrace vývojáře 25$ za rok a registrace do Apple store stojí 99$ za rok. Pro Android je potřeba vytvořit Apachem Cordovou instanční balíček, ten podepsat svým certifikátem, Zazipovat aplikaci programem zipalign a nahrát do Android Marketu. Pro iOS je postup složitější ve vyplňování množstvím dalších popisů aplikace. Z hlediska frameworku IONIC nevidím problém v publikování aplikace, spíše v následném nahrání do příslušného obchodu. Hodnotím známkou jedna.

Dokumentace

Hodnocení: 1

IONIC má rozsáhlou a praktickou dokumentaci. V dokumentaci jsou uvedeny jednotlivé komponenty včetně grafických ukázek a ukázek kódu. Jak šablon, tak kontrolerů. Na internetu jsou dostupná i oficiální videa. Níže je vyjmenovaná dostupná dokumentace.

Dokumentace:

- Ionic Documentation – dokumentace Javascript komponent, CSS komponent a ostatních nezbytných částí nutných pro vývoj
- Ionic Cloud Documentation – dokumentace podpůrných služeb
- The Ionic Book – provádí zjednodušeným procesem životního cyklu IONIC aplikace

Hodnotím známkou jedna.

Komunita a podpora

Hodnocení: 1

Vývoj ve frameworku IONIC je podporován rozsáhlou komunitou uživatelů. IONIC spustil svoje vlastní fórum, kde řeší dotazy vývojářů. Funguje IONIC Worldwide Slack, na kterém probíhá živá komunikace v několika vláknech. Když jsem se přihlásil byly nejpozdější příspěvky staré v řádech hodin. Také má svůj kanál na známém Stackoverflow. Hodnotím známkou jedna.

IONIC komunita:

- Ionic Forum – technické dotazy na webové stránce
- IONIC Worldwide Slack – Slack je komunikační nástroj
- Stackoverflow – známá webová stránka s technickými dotazy

Znuvupoužitelnost komponent

Hodnocení: 2

Framework samotný má definovanou strukturu model-view-controller, nemá jasně definovanou strukturu, takže pokud vývojář nezná best-practises, tak může vytvořit nepřehledný „Špagetový kód“. Javascript nemá definovaný typ dat proměnných, takže je vhodné je dobře pojmenovat. AngularJS umožňuje používat direktivy, které zajišťují, aby se kód neopakoval. Dají se použít vícekrát na různých místech aplikace. Pokud jsou během psaní kódu dodržovány best-practises, pak lze komponenty použít znovu jak v rámci jednoho, tak více projektů. Kvůli skutečnosti, že samotný framework IONIC a jeho nástroje neupozorňuje na nevhodně použitý kód dávám známku dvě.

Pluginy a moduly třetích stran

Hodnocení: 1

Javascript, AngularJS a už i IONIC framework mají širokou základnu uživatelů. Proto je na internetu k dispozici velké množství knihoven, modulů a materiálů, které lze do projektu přidat a používat. Na příklad v kritériu *Vhodnost pro různé druhy aplikací* zmiňuji framework, který umožňuje vytvoření 3D grafiky. Jako další příklad uvedu framework Material, který umožnuje použít v IONICu nové prvky UI. Je dostupný z http://ionicmaterial.com/. Díky velkému množství knihoven, pluginů a modulů dávám hodnocení jedna.

Integrace frameworku do IDE

Hodnocení: 3

Framework IONIC je poměrně nový, proto některá jeho klíčová slova vývojové prostředí nenapovídá. Je důležité použité prvky testovat živým náhledem. Na druhou stranu prostředí IDE podporují funkce a syntaxi Javascriptu, HTML a CSS stylů. Integrace do IDE byla vyzkoušena během vývoje aplikace PowerFLOW mobile v nástrojích *WebStorm* a *Sublime Text*. Hodnotím známkou 3.

Souhrn hodnocení všech kritérií

Během analýzy frameworku IONIC, implementace end-to-end aplikace a vyhledávání zdrojů jsem se vyloženě nesetkal s problémem s požitím tohoto frameworku. V tabulce 19 lze vidět všechna má hodnocení.

Tabulka 19: *Souhrn hodnocení všech kritérií*

Název	Hodnocení
cena	1
vhodnost pro různé druhy aplikací	2
podporované nativní funkce	1
budoucnost frameworku	1
vzhled UI frameworku	1
pokrytí mobilních operačních systémů	1
dostupnost vývojářů z hlediska software vendor firmy	1

rychlost UI mobilní aplikace	1
rychlost adaptace vývojáře	2
náročnost instalace a přípravy systému na vývoj	1
testování	1
publikace aplikace	2
dokumentace	1
komunita a podpora	1
znovu použitelnost komponent	2
pluginy a moduly třetích stran	1
integrace frameworku do IDE	3

7.2 Zkušenosti z praktické implementace

V této části shrnuji praktické zkušenosti, které jsem nabyl při implementaci aplikace PowerFLOW mobile. Část začíná praktickými osobními zkušenostmi během vývoje a pokračuje doporučenými nástroji.

Osobní zkušenosti

Na jednu stranu je velice hezké mít okamžitý náhled, jak aplikace funguje příkazem *ionic serve*. Na druhou stranu je v prohlížeči cache, která některé změny nezobrazuje okamžitě. Proto je vhodné během vývoje používat zkratku ctrl + F5 pro refresh stránky a načtení celé stránky znovu.

Pokud něco v náhledu aplikace příkazem *ionic serve* nefunguje, tak vývojářská konzole internetového prohlížeče Chrome poskytuje informace o chybě. Je to tedy první nástroj, do kterého by se měl vývojář podívat, v případě, že se aplikace nechová tak jak je předpokládáno. K lepší orientaci v konzoli lze v kódu používat výpisy do konzole a breakpointy[7].

K vytvoření UI je vhodné použít IONIC Creator. Je to drag-and-drop editor, který umožňuje používat IONIC komponenty a zobrazuje uživatelské UI. V nástroji lze také

[7] Breakpoint je místo, kde se kód zastaví a vývojář se může podívat na stav aplikace

vytvořit prototyp aplikace včetně prokliků. Příklady obrazovek, které lze v IONIC Creatoru vytvořit je možná shlédnou v analýze PowerFLOW mobile aplikace v části práce *Návrh uživatelského rozhraní*. Prototyp aplikace neslouží pouze k ukázce zobrazení a prokliků stránek, ale lze z něj vyexportovat základ projektu IONIC.

Doporučené nástroje pro vývoj

Pro samotnou implementaci je vhodné použít nástroj, který kontroluje Javascript, HTML, a CSS syntaxi. Doporučuji na základě vlastní zkušenosti následující nástroje. Jsou použity i v různých video tutoriálech, které jsem absolvoval.

Vhodné nástroje pro implementaci:

- WebStorm: The Smartest JavaScript IDE, JetBrains – Propracovaný placený nástroj pro frontend vývoj, umožňuje nainstalovat velké množství pluginů, integruje GIT
- Sublime Text – Zobrazuje strukturu projektu, kontroluje syntaxi a lze ho použít k rychlému doplňování kódu.

Nástroj na verzování kódu:

- Git – Umožnuje týmovou práci na zdrojovém kódu, vytvoření větví pro oddělený vývoj, řeší kolize kódu

Vhodný nástroj na debugování aplikace:

- Chrome debugger konzole – Umožnuje vypisování do konzole, popis a délku requestů, zobrazuje proměnné v Local Storage prohlížeči a lze přesměrovat port do mobilního telefonu pro vizuální dojem z aplikace.

7.3 Celkové shrnutí: Vhodnost IONIC frameworku pro multiplatformní mobilní vývoj

Na základě praktické implementace projektu PowerFLOW mobile, vyhodnocených kritérií a osobních zkušeností považuji framework IONIC za nástroj vhodný pro vývoj aplikace pro více mobilních operačních systémů. Technologie, které se ve frameworku vyskytují se hojně používají při vývoji frontend aplikací, proto je i velká základna potenciálních vývojářů. To platí i na druhou stranu. Vývojář, který se naučí pracovat s frameworkem IONIC je schopný využít nabyté znalosti pro implementaci jiných frontend projektů. Všechny technologie jsou bezplatně dostupné a jsou zdokumentované online. Komunita je aktivní prostřednictvím nástroje Slack, Ionic fóra nebo na Stackoverflow. Dokumentace je online a aktuální. Framework sám o sobě není vhodný pro hry a nelze v něm vytvářet widgety, ale to vše lze doplnit vhodným modulem nebo pluginem.

8 Závěr

Ve své práci, jejíž hlavní cíl je analýza frameworku IONIC, jsem dospěl k mnoha zajímavým poznatkům.

Chytré telefony se stávají součástí každodenního života velké části populace. Od roku 2011 do roku 2015 se počet prodaných chytrých telefonů podle průzkumu v kapitole tři zdvojnásobil. Proto je nutné při vytváření aplikací zohlednit i platformy mobilních telefonů.

Hlavní operační systémy na trhu jsou zatím dva: Android a iOS, ale může se stát, že v budoucnu bude mít výrazný podíl na trhu i další platforma. Nejblíže k tomu má Windows Phone. Vývoj aplikací pro jednotlivé platformy je sice optimalizovaný pro danou platformu, ale přináší řadu nevýhod. Hlavní nevýhodou jsou náklady. Další nevýhody jsou udržování aplikací, vývojový tým pro každou platformu, nevhodnost paralelního vývoje jedním týmem pro více platforem atd. Proto na trh nastupují frameworky ulehčující vývoj mobilní aplikace pro více operačních systémů. Práce se nezabývá porovnání frameworků, to je již v práci Srovnání přístupů multiplatformního vývoje mobilních aplikací [5].

Současným velkým hráčem multiplatformního vývoje mobilních aplikací je framework IONIC. Framework využívá množství technologií používaných obecně u frontend vývoje aplikací pro internetový prohlížeč. IONIC přizpůsobuje tyto technologie pro mobilní vývoj a přidává navíc komponenty UI pro mobilní zařízení. IONIC tedy vytváří vhodné prostředí pro multiplatformní mobilní vývoj. Dalším prvkem, který IONIC obsahuje je Apache Cordova. Apache Cordova umožňuje používání nativních funkcí v mobilní aplikací.

Využité technologie frameworkem IONIC jsou popsány v kapitole *Framework IONIC a s ním spojené technologie*. Ideální prostředek k načerpání vlastních praktických zkušeností, na jejichž základě lze zhodnotit vhodnost IONIC, je vývoj konkrétní aplikace nad tímto frameworkem. Rozhodl jsem se proto realizovat formou malého, ale uceleného end-to-end projektu mobilní aplikaci „PowerFLOW mobile“.

Součástí vývoje SW je analýza, která je v práci zahrnuta v kapitole *Analýza a návrh mobilní aplikace využívající framework IONIC.* V této části je aplikována *Metodika pro Malé softwarové projekty*. Nejdříve je analyzován a popsán současný stav vývoje a následně je analyzováno vznikající mobilní řešení pro aplikaci. Přestože je metodika v knize *Příklady modelů analýzy a návrhu aplikace v UML* [1] popisována pro desktopové aplikace, šla bez problému aplikovat na mobilní vývoj. Mobilní aplikace využívá služeb PowerFLOW services, takže jsou některé modely vynechány, protože se pro popis této mobilní aplikace nehodí. Například diagram tříd a sekvenční diagramy. Kdyby se mobilní aplikace vyvíjela včetně backendu, tak by tyto diagramy opět dávali smysl.

Po analýze mobilní aplikace následoval praktický vývoj ve frameworku IONIC. Přestože jsem neměl žádné zkušenosti s frontend vývojem, tak jsem se dokázal v poměrně krátké době s frameworkem seznámit a implementovat velkou část navržené funkcionality. Nejvíce mě překvapilo množství knihoven, pluginů a technologií, které musí začínající frontend vývojář zvládnout. Jedná se o velice komplexní systém.

V šesté kapitole je praktická ukázka vytvořeného projektu PowerFLOW mobile, kde je popsána struktura projektu, používané moduly v aplikaci, struktura stránek a ukázka zdrojového kódu pro přihlášení.

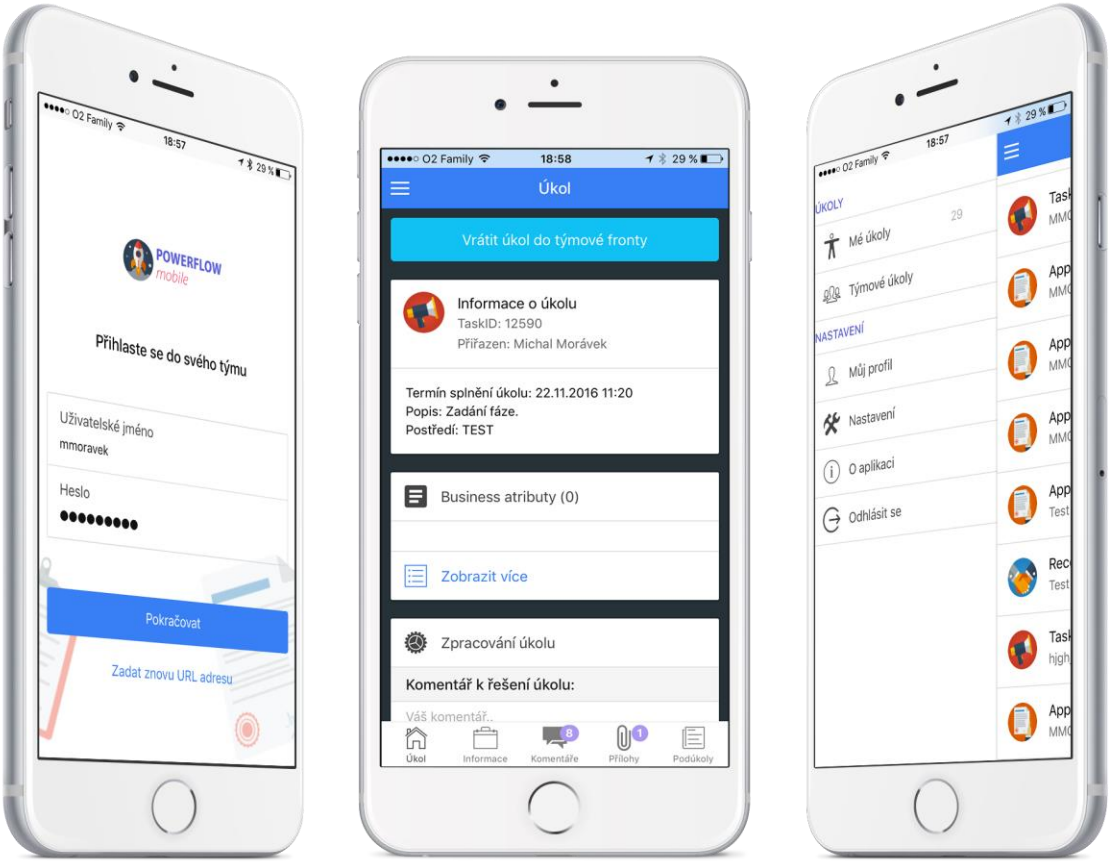
Po analýze mobilní aplikace a jejím vývoji jsem nabyl zkušenosti, které jsem využil v sedmé kapitole *Zhodnocení frameworku IONIC*. V kapitole jsou definovaná kritéria, podle kterých lze zhodnotit, zda je framework vhodný pro vývoj či nikoliv Framework lze využít pro vývoj překvapivě velkému množství typů mobilních aplikací. Jediný typ mobilních aplikací, pro které není vhodný jsou hry, protože obsahují renderování obsahu. Podpora frameworku z hlediska komunity i dokumentace je více než dobrá. Dokumentace je již připravena na stránkách pro beta verzi Ionic 2 a příspěvky ve fóru a na komunikačním nástroji Slack přibývají v rámci minut.

Ve výsledku jsem analyzoval framework IONIC různými pohledy. Od teoretického, který popisuje využívané technologie frameworkem, po praktický, kterým je samotný vývoj end-to-end mobilní aplikace. Následně jsem tyto dva pohledy využil v části, kde framework hodnotím.

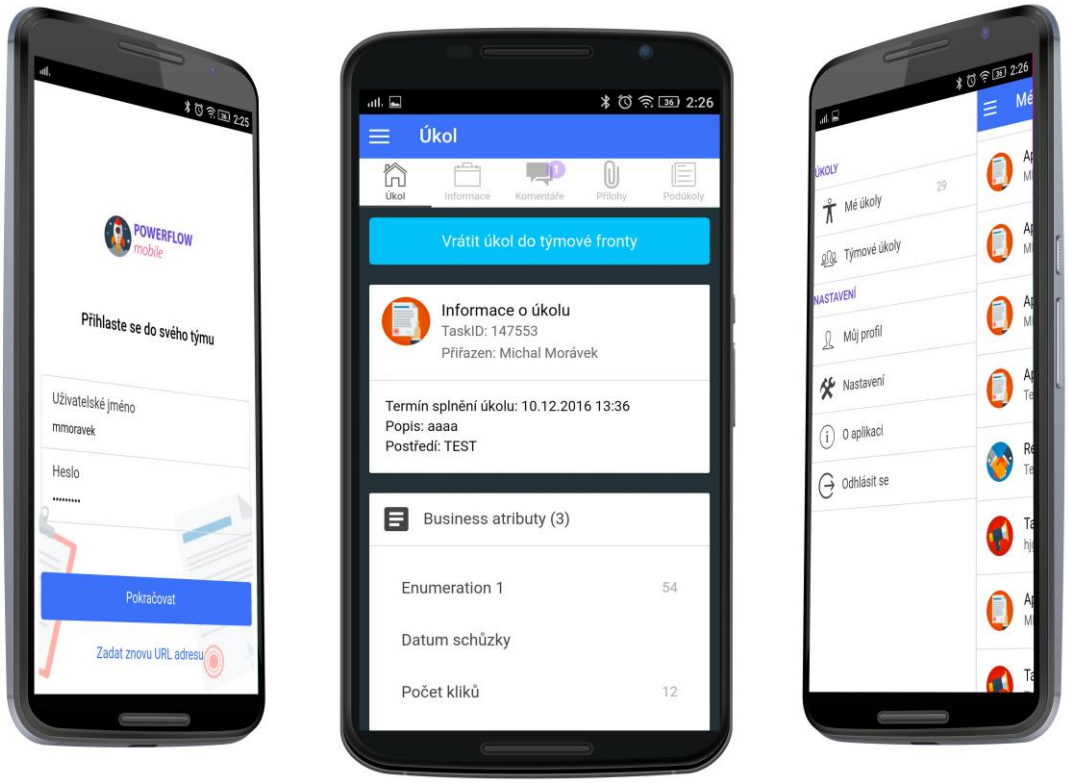
Dosažení vytčených cílů

Hlavním cílem této diplomové práce je analýza frameworku IONIC a zjištění vhodnosti tohoto frameworku pro multiplatformní mobilní vývoj. Cíl se skládá z dílčích cílů. První dílčí cíl *Popis frameworku IONIC* je realizován ve čtvrté kapitole. Druhý dílčí cíl Analýza aplikace vyvíjené ve frameworku IONIC metodikou MMSP [1] je realizován v páté kapitole na mobilní aplikaci PowerFLOW mobile. Třetí dílčí cíl *Popis části kódu aplikace vytvořené ve frameworku IONIC* je řešen v šesté kapitole a poslední dílčí cíl *Zhodnocení frameworku IONIC* je zahrnut v sedmé kapitole.

Příloha A: Ukázka obrazovek



Obrázek 43:
Ukázka obrazovek nasazené aplikace na telefonu s iOS [autor]



Obrázek 44:
Ukázka obrazovek nasazené aplikace na telefonu s OS Android [autor]

Terminologický slovník

Termín	Význam [zdroj]
MMSP	Metodika pro malé softwarové projekty [1]
UML	Unified Modeling Language je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. [https://cs.wikipedia.org/wiki/Unified_Modeling_Language]
CSS	Kaskádové styly
framework	Framework (aplikační rámec) je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji. [wikipedie]
Javascript	Je multiplatformní, objektově orientovaný skriptovací jazyk. [wikipedie]
REST	Representational State Transfer je architektura rozhraní, navržená pro distribuované prostředí. [Wikipedie]
API	Application Programming Interface označuje v informatice rozhraní pro programování aplikací. [Wikipedia]
UI	User interface
SDK	Software development kit
hybrid	Anglický termín pro slovo multiplatformní
Backend	Označuje se tak část aplikace, která slouží k administraci webových stránek a zpracování dat. Většinou komunikuje s frontendem, kde data zobrazuje.
Front-end	Část webu, která zajišťuje uživatelské rozhraní. Upravuje data tak, aby byla pro uživatele lépe čitelná.
IDE	Vývojové prostředí

Použité informační zdroje

[1] BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. Příklady modelů analýzy a návrhu aplikace v UML. Praha: Oeconomica, 2013. ISBN 978-80-245-1922-7

[2] ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací: Objektově orientovaná analýza a návrh prakticky. Dotisk prvního vydání. Brno: Computer Press, a.s., 2011. ISBN 978-80-251-1503-9.

[3] Ionic Documentation [online]. 2016 [cit. 2016-09-14]. Dostupné z: http://ionicframework.com/docs/

[4] NgCordova Documentation [online]. 2016 [cit. 2016-09-14]. Dostupné z: http://ngcordova.com/docs/

[5] ZIKMUND, Jan. Srovnání přístupů multiplatformního vývoje mobilních aplikací. Praha, 2015. Vysoká škola ekonomická v Praze. Vedoucí práce Alena Buchalcevová.

[6] Smartphone OS Market Share, 2016 Q2. IDC Research, Inc. [online]. Framingham: IDC Research, Inc., 2016 [cit. 2016-11-06]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[7] What is open source software? In: Opensource.com [online]. USA: Red Hat, Inc., 2016 [cit. 2016-11-07]. Dostupné z: https://opensource.com/resources/what-open-source

[8] OpenUP [online]. EPF 1.5.1.4 - Licence, 2012 [cit. 2016-11-07]. Dostupné z: http://epf.eclipse.org/wikis/openup/

[9] ANDERSON, Monica. Technology Device Ownership. Pew Research Center [online]. 2015, 26 [cit. 2016-11-08]. Dostupné z: http://www.pewinternet.org/2015/10/29/technology-device-ownership-2015

[10] Mobile/Tablet Operating System Market Share. NETMARKETSHARE [online]. Market Share Statistics for Internet Technologies, 2016 [cit. 2016-11-08]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1#

[11] Mobile/Tablet Top Operating System Share Trend. NETMARKETSHARE [online]. Market Share Statistics for Internet Technologies, 2016 [cit. 2016-11-08]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=11&qpcustomb=1

[12] RAVULAVARU, Arvind. Learning Ionic: Build real-time and hybrid mobile applications with Ionic. Birmingham, UK: Packt Publishing Ltd., 2015. ISBN 978-1-78355-260-3.

[13] WILKEN, Jeremy a Adam BRADLEY. Ionic in action: hybrid mobile apps with Ionic and AngularJS. Shelter Island, NY: Manning Publications, 2015. ISBN 978-163-3430-082.

[14] VOLDŘICH, Martin. Analýza frameworků pro vývoj multiplatformních mobilních aplikací využívající HTML technologie. Praha, 2015. Vysoká škola ekonomická v Praze. Vedoucí práce Jarmila Pavlíčková.

[15] RUMPLI, Marko. Návrh a vývoj multiplatformních mobilních aplikací. Hradec Králové, 2015. Univerzita Hradec Králové. Vedoucí práce Tomáš Kozel.
[16] KADĚRA, Miroslav. Vývoj multiplatformních mobilních aplikací pro prodej zboží. Brno, 2014. MASARYKOVA UNIVERZITA. Vedoucí práce Vlastislav Dohnal.
[17] KLOS, Roman. Multiplatformní vývoj mobilních aplikací pomocí webových technologií. Praha, 2014. Vysoká škola ekonomická v Praze. Vedoucí práce Jarmila Pavlíčková.
[18] BULVAS, Vít. Nástroje pro podporu vývoje nativních multiplatformních mobilních aplikací. Praha, 2016. Vysoká škola ekonomická v Praze. Vedoucí práce Jarmila Pavlíčková.
[19] NADRCHAL, Tomáš. Přístupy k řešení mobilních webových aplikací. Praha, 2012. Vysoká škola ekonomická v Praze. Vedoucí práce Zuzana Šedivá.
[20] ČÁPEK, Petr. Moderní metody tvorby nativních multiplatformních mobilních aplikací. Zlín, 2014. Univerzita Tomáše BAti ve Zlíně. Vedoucí práce Erik Král.
[21] GREEN, Brad a Shyam SESHADRI. AngularJS. Sebastopol, CA: O'Reilly Media, Inc., 2013. ISBN 978-144-9344-856.
[22] WAHLIN, Dan a Ian SMITH. AngularJS in 60 Minutes. Wahlin Consulting, 2014. Stránka autora: https://weblogs.asp.net/dwahlin
[23] RUEBBELKE, Lukas a Brian FORD. AngularJS in action. Shelter Island, NY: Manning, 2015. ISBN 16-172-9133-1.
[24] GRANT, Andrew. Beginning AngularJS. Berkeley, CA: Apress, 2014. Expert's voice in Web development. ISBN 14-842-0161-2.
[25] Guide to Angular 1 Documentation. AngularJS [online]. Google, 2016 [cit. 2016-11-08]. Dostupné z: https://docs.angularjs.org/guide
[26] Bower: A package manager for the web [online]. 2012 [cit. 2016-11-09]. Dostupné z: https://bower.io/
[27] Node.js [online]. Node.js Foundation [cit. 2016-11-09]. Dostupné z: https://nodejs.org
[28] Git: --local-branching-on-the-cheap [online]. [cit. 2016-11-09]. Dostupné z: https://git-scm.com
[29] Sass: CSS with superpowers [online]. [cit. 2016-11-09]. Dostupné z: http://sass-lang.com/
[30] Gulp: Automate and enhance your workflow [online]. [cit. 2016-11-09]. Dostupné z: http://gulpjs.com/
[31] Apache Cordova: Target multiple platforms with one code base [online]. 2012 [cit. 2016-11-09]. Dostupné z: https://cordova.apache.org/
[32] Swagger: The World's Most Popular Framework for APIs. [online]. USA [cit. 2016-11-09]. Dostupné z: http://swagger.io/
[33] Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. In: Salesforce developers [online]. 2016 [cit. 2016-11-09]. Dostupné z: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[34] Where does the Ionic Framework fit in? In: THE OFFICIAL IONIC BLOG: The next generation HTML5 hybrid app development SDK [online]. 2013 [cit. 2016-11-09]. Dostupné z: http://blog.ionic.io/where-does-the-ionic-framework-fit-in/

[35] BBC [online]. 2016 [cit. 2016-11-09]. Dostupné z: http://www.bbc.com/

[36] MMSP: Metodika pro malé softwarové projekty [online]. Praha: Vysoká škola ekonomická v Praze, 2011 [cit. 2016-11-14]. Dostupné z: http://mmsp.czweb.org/

[37] Winterlingova krizová matice. Management Mania [online]. 2015 [cit. 2016-11-19]. Dostupné z: http://managementmania.com/index.php/component/content/article/62-ostatni/381-winterlingova-krizova-matice

[38] Developer Survey Results. In: Stackoverflow [online]. 2016 [cit. 2016-12-04]. Dostupné z: http://stackoverflow.com/research/developer-survey-2016

[39] SWAGGER: THE WORLD'S MOST POPULAR API FRAMEWORK [online]. Somerville, 2016 [cit. 2016-12-05]. Dostupné z: http://swagger.io/

[40] Different Types Of Apps That App Developers Need To Know About. *Buzinga: app development* [online]. 2016 [cit. 2016-12-06]. Dostupné z: http://www.buzinga.com.au/buzz/how-many-app-types-are-there/

[41] Hybrid Apps and the Curse of the 300ms Delay: Build amazing native and progressive web apps with HTML5. THE OFFICIAL IONIC BLOG [online]. 2014 [cit. 2016-12-10]. Dostupné z: http://blog.ionic.io/hybrid-apps-and-the-curse-of-the-300ms-delay/

[42] Publishing your app: How to get the most out of Ionic. Ionic documentation [online]. [cit. 2016-12-10]. Dostupné z: https://ionicframework.com/docs/guide/publishing.html