TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Vývoj hybridních mobilních aplikací a jejich využití v informačním systému OR-SYSTEM Open

Bakalářská práce

Studijní program: B6209 – Systémové inženýrství a informatika
Studijní obor: 6209R021 – Manažerská informatika

Autor práce: **Michal Dostál**
Vedoucí práce: Ing. Dana Nejedlová, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Economics

The development of hybrid mobile applications and their usage in OR-SYSTEM Open information system

Bachelor thesis

Study programme: B6209 – System Engineering and Informatics
Study branch: 6209R021 – Managerial Informatics

Author: **Michal Dostál**
Supervisor: Ing. Dana Nejedlová, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Ekonomická fakulta
Akademický rok: 2016/2017

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal Dostál**
Osobní číslo:	**E14000640**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Manažerská informatika**
Název tématu:	**Vývoj hybridních mobilních aplikací a jejich využití v informačním systému OR-SYSTEM Open**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

1. Současné trendy ve vývoji mobilních aplikací
2. Frameworky pro vývoj hybridních aplikací
3. Jazyk Javascript a Angular JS
4. Vývoj konkrétní hybridní aplikace

Rozsah grafických prací:

Rozsah pracovní zprávy: **30 normostran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

WILKEN, Jeremy a Adam BRADLEY. Ionic in action: hybrid mobile apps with Ionic and AngularJS. Shelter Island, NY: Manning Publications, 2015. ISBN 9781633430082.
DAYLEY, Brad a Brendan DAYLEY. AngularJS, JavaScript, and jQuery All in One, Sams Teach Yourself. Sams Publishing, 2015. ISBN 9780672337420.
Elektronická databáze článků ProQuest (knihovna.tul.cz).

Vedoucí bakalářské práce: **Ing. Dana Nejedlová, Ph.D.**
Katedra informatiky

Konzultant bakalářské práce: **Ing. Petr Motl**
OR-CZ, spol. s r. o.

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **31. května 2018**

prof. Ing. Miroslav Žižka, Ph.D.
děkan

L.S.

doc. Ing. Jan Skrbek, Dr.
vedoucí katedry

V Liberci dne 31. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych tímto poděkoval mé vedoucí bakalářské práce Ing. Daně Nejedlové, Ph.D. a Ing. Petru Motlovi za jejich čas, ochotu a cenné rady poskytnuté při zpracování této bakalářské práce.

Anotace

Bakalářská práce Vývoj hybridních mobilních aplikací a jejich využití v informačním systému OR-SYSTEM Open se zabývá porovnáním dostupných metod tvorby mobilních aplikací s konkrétním zaměřením na hybridní mobilní aplikace a popisem vývoje konkrétní hybridní mobilní aplikace.

V práci jsou popsány současné trendy ve vývoji mobilních aplikací a základní přístupy k jejich vývoji. Práce dále popisuje jazyk JavaScript a další webové technologie jakožto nejpoužívanější jazyky pro tvorbu hybridních mobilních aplikací.

V praktické části se práce zaměřuje na vývoj konkrétní mobilní hybridní aplikace komunikující s informačním systémem OR-SYSTEM Open.

Klíčová slova

Hybridní mobilní aplikace, mobilní webové stránky, nativní mobilní aplikace, framework Ionic, JavaScript, AngularJS, datový terminál, SOAP.

Annotation

This bachelor thesis The development of hybrid mobile applications and their usage in OR-SYSTEM Open information system deals with comparison of available methods of creating mobile applications with in-depth look into hybrid mobile applications and description of development of specific hybrid mobile application.

First part of this thesis describes trends in mobile app development and approaches to their development – native mobile apps, mobile web pages and hybrid mobile apps. This thesis also describes JavaScript and other web development languages most used in hybrid mobile app development.

Second part of this thesis describes the development of hybrid mobile application which communicates with the OR-SYSTEM Open information system.

Key Words

Hybrid mobile apps, mobile web pages, native mobile apps, Ionic framework, JavaScript, AngularJS, data terminal, SOAP.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
AOL	America OnLine
API	Application Programming Interface
BYOD	Bring Your Own Device
CLI	Command Line Interface
CSS	Cascading Style Sheets
DIČ	Daňové identifikační číslo
DOM	Document Object Model
DVM	Dalvik Virtual Machine
GPS	Global Positioning System
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICE	Integrated Cloud Environment
IDE	Integrated Development Environment
IoT	Internet of Things
JSON	JavaScript Object Notation
MVC	Model-View-Controller
QR	Quick Response
RFID	Radio Frequency IDentification
SDK	Software Development Kit
SOAP	Simple Object Access Protocol

UI	User Interface
URL	Uniform Resource Locator
UWP	Universal Windows Platform
UX	User eXperience
W3C	World Wide Web Consorcium
XML	Extensible Markup Language
XAML	Extensible Application Markup Language

Pojmy související s tvorbou mobilních aplikací

API – volně přeloženo jako rozhraní pro programování aplikací. Jedná se o abstrakci, jež popisuje interakci s různými funkcemi. Může se jednat o funkce a metody operačního systému přístupné pro ostatní programy a aplikace pracující v tomto operačním systému od výrobců nezávislých na výrobci operačního systému, který přitom může být proprietární, tj. nemusí mít otevřený zdrojový kód. Zároveň se může jednat i o API programovacích jazyků, které své funkce a metody zpřístupňují pomocí knihoven.

CLI nástroje – nástroje, které jsou ovládány skrze příkazový řádek

DX – nástroj působící při spouštění aplikace v OS Android, který překládá soubory .class do souborů s příponou .dex, protože jen tomuto typu souborů virtuální stroj Dalvik Virtual Machine rozumí.

Framework – sada knihoven a nástrojů. Může obsahovat například set kaskádových stylů nebo speciální sadu funkcí, které jsou při programování využívány.

Front-end – to, co uživatel na obrazovce a vidí a pomocí čehož komunikuje se zařízením.

Kompilovaný programovací jazyk – programovací jazyk, jehož zdrojový kód se překládá (kompiluje) za pomoci překladače do strojového kódu.

Komponenta – v souvislosti s frameworky, vnímáno jako jednotlivé grafické a funkční části frameworku, které dohromady tvoří aplikaci. Za komponentu lze například považovat tlačítko, zobrazované v aplikaci.

Nativní mobilní aplikace – aplikace vytvořená v programovacím jazyce, který podporuje cílová platforma – tedy operační systém daného zařízení. Zpravidla má každý operační systém (platforma) jiný programovací jazyk a nástroje SDK, které při sestavení instalačního balíčku vytvoří ze zdrojového kódu bytecode, který je pak interpretován operačním systémem cílového zařízení.

Platforma – pracovní prostředí, ve kterém aplikace pracuje a funguje – v případě mobilních aplikací se jedná o operační systém zařízení.

Plugin – v této práci zmiňováno v souvislosti s frameworky pro vývoj hybridních aplikací. Jedná se přídavnou funkčnost frameworku, umožňující komunikaci se službami a hardwarem mobilního zařízení.

Interpretovaný programovací jazyk – programovací jazyk, jehož instrukce ve zdrojovém kódu jsou interpretovány (prováděny) přímo, programem zvaným interpret bez vzniku strojového kódu. Zdrojový kód může být před interpretací přeložen do takzvaného bytecode. V každém bytecodu je vyhrazen jeden byte právě pro instrukci.

Garbage collector – speciální algoritmus vyhledávající objekty, které již nejsou pro běh programu potřeba, či na ně již neodkazuje nějaká proměnná nebo objekt.

Webová služba – jakákoli část softwaru, která zpřístupňuje své funkce v síti internet. Běží například na aplikačním serveru a komunikuje s klienty s pomocí XML. To znamená, že komunikace není závislá na operačním systému klienta ani serveru a ani na programovacím jazyce, ve kterém jsou obě strany napsány.

Úvod

Mobilní zařízení jsou dnes mnohdy používanější než klasické stolní počítače. Z mobilního telefonu jeho uživatel vyřídí emaily, domluví schůzky, spojí se se svými přáteli, či si dokonce zahraje hry pro oddech. Takové zařízení máme v dnešní době téměř všichni, a tak není divu, že se trend mobilních zařízení dostává i do firemního prostředí.

Vývojářské firmy tak přicházejí na trh se svými mobilními aplikacemi, které dnes nejčastěji běží na platformě Android, iOS nebo Windows. Přestože existují i další, tyto tři platformy jsou nejvyužívanější. Pokud se na problematiku podíváme z pohledu vývojářské firmy, ta si přeje mít aplikace, které využije co nejvíce uživatelů, tedy jejích zákazníků, a proto se bude snažit o vývoj dané aplikace pro všechny běžně využívané platformy.

Pro takový vývoj, je třeba práce několika týmů, kde se každý tým zaměří na vývoj na jedné platformě. Vývoj dané aplikace tak probíhá v oddělených projektech pro různé platformy. Proto na scénu přicházejí mobilní hybridní aplikace, jejichž hlavní předností je fakt, že jsou vytvořeny jako jediný projekt, ale jsou schopny běžet na všech podporovaných platformách, a tak ušetří spoustu práce. Navíc jsou vyvíjeny v HTML, CSS a Javascriptu, jež jsou jazyky, které ovládá snad každý web programátor.

Autor této bakalářské práce si klade za cíl seznámit čtenáře s technologií vývoje hybridních mobilních aplikací a jejím využitím ve firmě OR-CZ, spol. s r. o. při tvorbě mobilní aplikace pro informační systém OR-SYSTEM Open.

Pojmy, které jsou v této práci označeny modrou barvou, jsou vysvětleny v kapitole Pojmy související s tvorbou mobilních aplikací

1 Současné trendy ve vývoji mobilních aplikací

1.1 Přístupy k vývoji mobilních aplikací

V oboru vývoje mobilních aplikací existuje několik přístupů. Za 3 základní přístupy jsou považovány klasické nativní aplikace (viz **Nativní mobilní aplikace**), mobilní webové stránky a hybridní mobilní aplikace, které spojují výhody prvních dvou zmíněných.

1.1.1 Nativní mobilní aplikace

Za nativní mobilní aplikaci považujeme takovou aplikaci, která byla napsána v nativním programovacím jazyku operačního systému, pro který byla vyvíjena. Nativní aplikace využívá služeb a metod, které jí poskytuje operační systém, nad kterým je postavena a na kterém bude fungovat. Nativní kód je většinou kompilovaný (viz **Kompilovaný programovací jazyk**) a v mnoha případech rychlejší než interpretovaný (viz **Interpretovaný programovací jazyk**), jako je např. Javascript [1]. Tabulka č. 1 popisuje příklady nativních programovacích jazyků tří největších mobilních platforem.

Platforma	**Programovací jazyk**
Android	Java
iOS	Swift, Objective C
Windows	Visual C++

Tabulka 1: Nativní programovací jazyky vybraných platforem
Zdroj: [2, s. 3]

Základní charakteristikou nativních mobilních aplikací je nepřenositelnost na platformu jinou. To znamená, že pokud vytvoříme aplikaci pro operační systém Android, nelze ji pak nainstalovat např. na zařízení s operačním systémem iOS [3].

Jak uvádí Wilken [2], výhodou nativních mobilních aplikací je jejich přímý přístup k nativnímu aplikačnímu programovému rozhraní, dále jen **API**, což umožňuje nejužší spojení s platformou. Dále je to výkon aplikací a nativní programovací jazyk. Jako nevýhodu Wilken označuje potřebu programátora být plně zběhlý v daném nativním programovacím jazyce. Nativní aplikace nejsou multiplatformní, proto je třeba vyvíjet aplikaci pro každou platformu zvlášť.

1.1.2 Mobilní webové stránky

Mobilní webové stránky, nebo mobilní webové aplikace jsou tvořeny pro mobilní zařízení a je k nim přistupováno pomocí webových prohlížečů. Jsou vyvíjeny webovými technologiemi, hostovány na vzdálených serverech, předávány k uživateli pomocí

standardních protokolů (např. HTTP nebo HTTPS) a je k nim přistupováno pomocí URL [3]. Vývoj je koncipován tak, aby byly dobře zobrazitelné na malém displeji např. mobilního telefonu. Některé webové stránky mají pro tento účel speciální verzi – toho si lze povšimnout, pokud webová stránka zobrazovaná na mobilním zařízení má subdoménu „m." nebo „t.", což lze v současnosti vidět např. na webu www.idnes.cz.

Za výhody mobilních webových stránek jsou dle Wilkena [2, s. 6] považovány především udržitelnost, multiplatformovost a eliminace nutnosti aplikaci instalovat. Aplikace je kdykoli přístupná z mobilního webového prohlížeče. Jako největší nevýhoda je považován nulový přístup k nativnímu API, kdy webová stránka může maximálně reagovat na události prohlížeče, ale „hlouběji" do zařízení přístup nemá.

1.1.3 Hybridní mobilní aplikace

Hybridní mobilní aplikace spojuje výhody obou předchozích přístupů, protože je schopna pomocí pluginů (viz **Plugin**), které lze získat z repositáře frameworku Apache Cordova (viz **Framework**), přistupovat k nativnímu API zařízení. Taková aplikace je zároveň multiplatformní, a tak je možné ji spustit na téměř všech mobilních operačních systémech. V současné době jsou, dle výpisu dostupných platforem generované utilitou Ionic ve verzi 2.1.1, použitelné následující platformy: Android, Windows, iOS, Blackberry OS 10 nebo Firefox OS.

1.2 Obecné trendy ve vývoji mobilních aplikací

Podle Klubnikina [4] je největším trendem vývoj aplikací pro tzv. IoT – Internet of Things – přeloženo jako internet věcí. Tyto aplikace ve většině případů neběží na mobilních telefonech, ale na běžných domácích spotřebičích, které mohou být připojeny k internetu.

Dnes již existuje mnoho přístrojů v domácnosti, které jsou připojeny k internetu a je možné je ovládat pomocí mobilních zařízení. Vzniká tak tzv. chytrá domácnost, kde běžné přístroje v ní mají připojení k internetu a dokáží komunikovat s mobilními telefony a s ostatními zařízeními v domácnosti. Do takové chytré domácnosti dnes mohou patřit například chytré televize, chytré lednice nebo chytré pračky, které se snaží jejich uživatelům usnadnit užívání. Do systému lze napojit například i světla, topení a zabezpečení domácnosti.

Pokud chce vývojářská firma dosáhnout toho, aby její aplikace měla co nejvyšší počet uživatelů, musí být její aplikace pro uživatele užitečná a snadná k ovládání. User Experience

– neboli tzv. UX, je důležitým prvkem při tvorbě mobilní aplikace. Dle J. Goveové [5] až 25 % uživatelů spustí aplikaci jen jednou a už se k ní nikdy nevrátí. Je tedy třeba se držet kroků ke správnému UX – uživatelské zkušenosti. Takovým krokem může být tvorba přehledné a intuitivní navigace v aplikaci, která cílového uživatele dovede tam, kam potřebuje. Podle [5] je též výhodné, pokud jsou hlavní funkce aplikace viditelné již při spuštění. Uživatel se tak ihned může dostat k činnosti, kvůli které aplikaci spustil.

Dalším významným trendem ve vývoji mobilních aplikací je oddělení dat pracovních od těch osobních. Pro uživatele je dnes nejpříjemnější využívat jen jedno zpravidla soukromé zařízení též k práci pro zaměstnavatele a mnoho firem podporuje či vyžaduje politiku BYOD (= přines si své zařízení). V takovém telefonu jsou jak data osobní, tak data pracovní, která je třeba jistým způsobem chránit před neautorizovaným užitím či napadením telefonu virem.

Tzv. wearables – neboli nositelná zařízení nebo nositelná elektronika, jako jsou například chytré hodinky, se stávají stále více populárními. Hlavní výhodou je propojení s mobilním telefonem, a tak je možné na takových hodinkách provést nejběžnější operace jako přečíst SMS zprávu nebo zjistit číslo příchozího volajícího bez potřeby vytažení telefonu z kapsy. Tato zařízení bývají také často využívána při sportu, kdy pomocí aplikací dokáží sledovat výkony nebo díky hardwaru měřit tep či počítat kroky.

Co se týče vývojových prostředí, dle [6] existuje nový trend v oblasti vývoje. Jedná se o tzv. ICE – integrované cloudové prostředí, kdy je nahrazeno klasické IDE jeho cloudovou variantou. Autor článku [6] spatřuje výhody tohoto řešení v tom, že vývojář není omezen výkonem svého stroje a má k dispozici více platforem, možností a nástrojů. Dále je to dobrá spolupráce v týmu díky komponentám (viz **Komponenta**) pro komunikaci a spolupráci. Nevýhody mohou tvořit nákladnost cloudového řešení a případné možné problémy s internetovým připojením, kterém není vždy a všude a nemusí být dostatečně rychlé.

1.2.1 Trendy v nativních mobilních aplikacích

Trendem poslední doby je hlasová komunikace s virtuálním osobním asistentem v mobilním zařízení. Společnost Siri, Inc., která byla v roce 2010 koupena společností Apple [7], začala s vývojem dnes velice známého virtuálního osobního asistenta Siri, který je součástí mobilních zařízení Apple a dokáže komunikovat se svým uživatelem. Siri provádí základní úkoly jako sestavení SMS zprávy dle hlasového pokynu či vyhledávání informací na internetu a přednesení výsledku zpět uživateli v hlasové formě.

Dle článku [8] se dnes stává oblíbeným využití autentizace pomocí otisků prstů. Na trh s tímto řešením přišla společnost Apple, která uvedla funkci pro zařízení iPhone 5S, zvanou TouchID [9]. Dnes již technologii autentizace využívají i další výrobci, a přestože se nejedná o nejbezpečnější formu zabezpečení (hesla se dají měnit, ale otisky prstů ne), je stále více instalována do nových zařízení.

Jak uvádí článek [10], dnešní zařízení začínají být vybavena různými senzory, které mohou aplikace využívat. Mezi nové typy senzorů v mobilních zařízeních lze zařadit např. teploměr, barometr či dokonce měřič vlhkosti. V budoucnu to pak mohou být i 3D senzory.

Na několika zařízeních lze dnes najít i duální fotoaparáty nebo 3D kamery.

1.2.2 Trendy v mobilních webových stránkách

Základním požadavkem pro mobilní web je to, aby tato verze webu byla co nejvíce zjednodušená, ale zároveň stejně použitelná jako verze desktopová. Existuje výbor pro standardy spadající pod World Wide Web Consorcium (W3C), který doporučuje dodržování tzv. principu jednoho webu, v originále „One Web“ princip. Tento princip říká, že uživateli by měly být prezentovány stejné informace a služby bez ohledu na zařízení, na kterém koncový uživatel operuje [11]. Například společnosti Google a Twitter se tohoto pokynu drží [12].

Základní strategií pro vývojáře je vytvořit rozdílné front-end komponenty (viz **Front-end**). Vývojář tak vytvoří rozdílné komponenty pro desktopovou a mobilní verzi, zatímco je snaha zachovat implementaci na straně serveru stejnou. S tvorbou mobilní verze webu mohou pomoci různé knihovny a frameworky (např. Twitter Bootstrap) nebo dokonce nástroje, které dokážou migrovat webovou stránku do podoby, která je pro mobilní zařízení vhodná.

Další strategií pro vývojáře je se zaměřit na vývoj webů, pro které není třeba vyvíjet samostatnou mobilní verzi, nýbrž jejich webová stránka se automaticky přizpůsobí možnostem zařízení, na kterém je zobrazována. Je využíváno skriptovacích jazyků, které napomáhají k jednoduššímu a přesnějšímu zpracování zdrojového souboru webové stránky a jeho následnému zobrazení.

Důležitou součástí dnešních mobilních webů je jazyk HTML ve verzi 5, který přináší novou funkcionalitu a několik nových tagů – formátovacích značek. Nová funkcionalita se dotkla například formulářů, kdy lze rozeznávat různé formáty vstupů a ovlivnit tak uživatelský

vstup. V mobilním zařízení se tak díky tomu například zobrazí virtuální klávesnice s relevantními znaky. Mezi nové typy uživatelského vstupu patří například telefonní číslo, email, datum nebo URL. Jak uvádí [13], HTML 5 není jen sada formátovacích značek, ale díky nově implementovaným API a mechanismům lze ukládat data do lokálního úložiště, využívat video prezentací a komplexního formátování obsahu.

1.2.3 Trendy ve vývoji hybridních aplikací

Za velký trend ve vývoji hybridních aplikací lze považovat užití různých frameworků, které jeho uživateli (programátorovi) usnadní práci při vývoji programu. Většina z nich je open source, takže za nimi existuje velká komunita uživatelů, která pomáhá tvůrcům frameworku ve zlepšování a kontrole jejich práce. Programátor se tak může při řešení svého problému obrátit na ostatní uživatele a existuje velká šance, že takový problém již někdo řešil, a v nejlepším případě i vyřešil.

Frameworky jsou dále charakteristické svými grafickými prvky. Využívané jsou hlavně webové technologie, takže se tvůrci snaží těmito nástroji přiblížit vzhledu nativní aplikace, tak, aby běžný uživatel nepocítil jakýkoli rozdíl při jejím užívání. Nativní aplikace mají specifické grafické prvky spojené s platformou – může se jednat o vzhled tlačítek nebo styl menu. S nativními technologiemi je tak těžké vytvořit aplikaci vzhledově identickou pro všechny platformy. Tento problém řeší právě frameworky pro vývoj hybridních aplikací, které svými nástroji dokáží vygenerovat pro každou platformu aplikaci vzhledově identickou.

K vývoji hybridních aplikací dnes existují 2 přístupy. Prvním je využití webových technologií, jako jsou např. HTML, CSS a Javascript. Tohoto přístupu využívají např. framework Ionic a Apache Cordova. Druhou možností je napsat aplikaci v jazyce C# nebo Javascript a nechat výslednou aplikaci zkompilovat do nativního kódu. Tento přístup využívají mimo jiné i frameworky Xamarin nebo Appcelerator Titanium [14].

V roce 2016 provedli autoři článku [14] studii hybridních aplikací, které jsou k dispozici ke stažení skrze „obchody s aplikacemi“. Z 80 000 zkoumaných aplikací bylo 15 512 aplikací hybridních. Tuto část pak podrobili výzkumu, na jakém frameworku byla aplikace postavena. Výsledky ukazují, že největší podíl má Adobe PhoneGap a to konkrétně 10 562 aplikací. Zbytek aplikací bylo postaveno na frameworcích Appcelerator Titanium nebo Adobe Air. Frameworkům pro vývoj hybridních aplikací se v detailu věnuje kapitola č. 2.

1.3 Specifika nejrozšířenějších mobilních platforem

Tato kapitola se zaměřuje na specifické vlastnosti nejrozšířenějších mobilních platforem, kam řadíme Android, iOS a Windows, zejména pak na životní cykly aktivity aplikací cílených na jmenované mobilní platformy.

1.3.1 Platforma Android

Android jako takový se skládá ze čtyř vrstev: Linux Kernel, Libraries, Android Runtime a Application Framework. Nad těmito čtyřmi vrstvami pak stojí samotná aplikace.

Linux Kernel neboli linuxové jádro, je považováno za základní pilíř Androidu. Jedná se o nejnižší vrstvu architektury. Toto jádro je upraveným jádrem linuxovým, což znamená, že došlo k redukci funkcí a přizpůsobení možnostem mobilních zařízení. Hlavním smyslem linuxového jádra je přímá komunikace s hardwarem zařízení. Dále spravuje paměť zařízení, procesy a základní síťovou vrstvu spolu s ovladači. Na této úrovni jsou pak implementovány funkce jako zabezpečení systému, základní grafika, ovladače pro Bluetooth, 3G, Wi-Fi, kompas, GPS nebo fotoaparát.

Druhou vrstvu tvoří tzv. libraries, v překladu knihovny. Tyto knihovny jsou napsány v jazycích C nebo C++. Tvoří spojení mezi vyššími vrstvami a jádrem systému. Mezi tyto knihovny řadíme mimo jiné i Webkit, který slouží k renderování a zobrazování webových stránek, nebo Surface Manager, který zajišťuje funkcionalitu multidotykového displeje a zabezpečuje konečnou kompozici grafického výstupu do souvislého toku dat, směřujícího do grafické vyrovnávací paměti, odkud je pak zajištěno vykreslení na obrazovku.

Vrstva Android Runtime v sobě obsahuje sadu základních knihoven. Každá aplikace spuštěna na Androidu je samostatný proces, využívající svoji instanci DVM – virtuálního stroje Dalvik. DVM zabezpečuje běh spustitelných souborů (přípona .DEX), jež vznikly kompilací CLASS a JAR souborů. Díky optimalizaci bere virtuální stroj Dalvik v úvahu omezené možnosti zařízení jako napájení nebo paměť.

Proces spuštění aplikace je následující: Java kompilátor přeloží soubory zdrojového kódu aplikace do více binárních souborů Javy. Následně nástroj **DX** tyto binární soubory transformuje do samostatného souboru. DVM – virtuální stroj Dalvik – pak tento soubor začne interpretovat – vykonávat.

Application Framework neboli aplikační framework obsahuje ovládací prvky, ikony a další software, opakovaně použitelný v aplikacích mobilního zařízení. Také poskytuje základní služby systému jako jsou např. Package Manager, který působí jako databáze s aktuálně instalovanými aplikacemi na daném zařízení. Window Manager zase spravuje okna tvořící aplikace [15].

Typická aplikace pro Android sestává ze čtyř komponent: Activity, Service, Content Provider a Broadcast Receiver. Activity (aktivita) reprezentuje, co může uživatel s aplikací udělat. Service (služba) je komponenta běžící na pozadí, Content Provider spravuje sdílený set dat aplikace. Broadcast Receiver reaguje (odpovídá) na systémové zprávy a oznámení, jako např. „baterie je vybitá“ [16].

Životní cyklus aktivity Android aplikace je následující. Po spuštění aplikace (aktivity) se zavolá metoda onCreate(), která inicializuje aktivitu a například vytvoří uživatelské rozhraní. Po ní je spuštěna metoda onResume(), která je použita k inicializaci polí, naslouchání událostí a vázání služeb. Tato metoda je také volána vždy, když je aktivita znovu viditelná, například po navrácení se do aplikace. Poté se aktivita dostane do stavu „běží“.

Při stavu, kdy aplikace již není na popředí je zavolána metoda onPause(), která je používána k uvolnění zdrojů a dat. Jakmile už aplikace není viditelná, je zavolána metoda onStop(). V případě ukončení aplikace je volána metoda onDestroy() a následně je aktivita ukončena [17].

1.3.2 Platforma iOS

Pokud chce vývojář vytvořit mobilní aplikaci pro platformu iOS, potřebuje k tomu počítač Macintosh s procesorem Intel a nainstalovaným iOS SDK.

Mobilní aplikace pro iOS se píší v jazyce Objective-C nebo jazyce Swift, což je programovací jazyk postaven na jazycích C a Objective-C. Jazyk Swift byl vyvinut společností Apple pro platformy (operační systémy) iOS, macOS, watchOS a tvOS.

Základem aplikace psané v Objective-C, ostatně jako ve všech na C založených jazycích, je funkce *main*. Programátor však tuto funkci nepíše. Ta je vytvořena vývojovým prostředím Xcode, které by se dalo přirovnat k Android Studiu jako k nejvíce používanému a podporovanému IDE pro konkrétní platformu.

Aplikace na iOS může existovat v celkem 5 stavech. V prvním z nich – "Neběží" – ještě nebyla aplikace spuštěna, nebo byla terminována systémem. Při stavu "Neaktivní" aplikace běží v popředí, ale nepřijímá žádné události, jako dotyk, vzdálené ovládání, gyroskop, akcelerometr aj. Ve stavu "Aktivní" pak běží na popředí a události přijímá. Jedná se o normální stav aplikací běžících na popředí. Dalším ze stavů je "Na pozadí". V tomto stavu se většina aplikací nachází těsně před suspendací, ale pokud operace aplikací prováděná potřebuje delší čas na své provedení, v tomto stavu zůstane po potřebnou dobu. Pro aplikace vytvořené za účelem běhu na pozadí, je toto jejich hlavní stav, do kterého se dostane po spuštění. Posledním z možných stavů iOS aplikace je "Suspendována", kdy je aplikace na pozadí, ale neprovádí žádný kód. Pokud je v tomto stavu, aplikace zůstává uložena v paměti a pokud nastane situace, kdy je nízká úroveň baterie nízká, dojde k odstranění aplikace z paměti a k uvolnění prostředků [18].

1.3.3 Platforma Windows

Na mobilních zařízeních, používající Windows jako svůj operační systém, dnes najdeme především tyto verze: Windows Phone 8, 8.1 a Windows 10. Zařízení běžící na prvních dvou mají většinou nárok na upgrade na nejnovější verzi Windows 10, pokud jim to dovolují jejich technické specifikace. Každá z těchto verzí Windows má jiný nativní programovací jazyk.

Nejnovější, tzv. Windows Universal Apps, jsou vyvíjeny ve zvláštním jazyce WinJS. Aplikace vytvořená na platformě Universal App Platform je spustitelná na mobilním zařízení s operačním systémem Windows 10 Mobile i na klasickém desktopu s operačním systémem Windows 10. Lze tak na desktopovém počítači spouštět aplikace určené pro mobilní zařízení.

Dalšími programovacími jazyky, které je možné využít k naprogramování mobilní aplikace pro Windows 10, jsou Visual C++, C#, Visual Basic nebo JavaScript. Pro každý z těchto jazyků je používaný jiný systém definování rozložení grafických prvků uživatelského rozhraní. U Visual C++, C# a Visual Basic lze využít tzv. XAML – XAML je značkovací jazyk postavený na XML, který byl vytvořen společností Microsoft a jedná se o jazyk stojící za vizuální prezentací vyvíjené aplikace [19]. U JavaScriptu se pak klasicky využívá jazyka HTML. Tato druhá možnost zajišťuje výsledné aplikaci možnost běžet na velké škále zařízení, díky univerzálnosti jazyka HTML.

S Windows 10 přišla tzv. Universal Windows Platform (univerzální platforma Windows), která nabízí stejné prostředí pro běh aplikací na všech zařízeních, na kterých běží Windows 10. Díky tomu mohou aplikace na platformě UWP volat WinRT API, ale i další API, která nejsou specifická pro typ zařízení (device family), na kterém aplikace běží. UWP garantuje stejnou vrstvu API operačních systémů napříč zařízeními. To znamená, že lze vytvořit aplikaci, která může být nainstalována na široké množství zařízení [20].

Co se týče životního cyklu aktivity Windows aplikací, změna od klasického modelu „běží“ nebo „neběží“ známého od Win32 nebo .NET aplikací, přišla s nástupem Windows 8, kdy byla přidán nový stav aplikace, a to „suspendována“. Tato změna nastoupila v tzv. Windows Store Apps, které Windows 8 představilo. Windows Store aplikace je suspendována krátce potom, co ji uživatel minimalizuje nebo přepne do jiné aplikace. To znamená, že vlákno aplikace je zastaveno a aplikace je uložena do paměti. Toto umožňuje rychlé spuštění, když se uživatel do aplikace vrátí.

Windows 10 pak představily další dva stavy a to „běžící na pozadí“ a „běžící na popředí“. Vypnutá aplikace je ve stavu „neběží“. Po spuštění se aplikace dostane do stavu „běžící na pozadí“, poté tento stav opustí a je ve stavu „běžící na popředí“ – v tomto stavu s aplikací uživatel pracuje. V případě, že uživatel aplikaci minimalizuje, Windows čeká několik sekund, jestli se uživatel k aplikaci vrátí. Pokud se v tomto časovém okně nevátí a pokud neběží nějaký úkol na pozadí, Windows tuto aplikaci suspenduje. Systém se jí snaží udržet v paměti, ale pokud nemá dostatek zdrojů, musí je uvolnit, takže aplikaci ukončí (terminuje). Po návratu ze stavu „suspendována“ se aplikace vrací do stavu „běžící na pozadí“, a to v případě, že na ní uživatel přepne, nebo se zařízení dostane ze stavu kdy je nízký stav nabití baterie. Pokud se totiž zařízení dostane do stavu nízké úrovně baterie, začne ukončovat aplikace v paměti, aby prodloužilo dobu funkčnosti [21].

2 Frameworky pro vývoj hybridních mobilních aplikací

Tato kapitola by měla čtenáře seznámit s dnes nejznámějšími frameworky pro vývoj mobilních hybridních aplikací.

2.1 Apache Cordova

Apache Cordova je open source framework obsahující řadu pluginů. Tyto pluginy zprostředkovávají přístup webového prohlížeče hybridní aplikace k nativnímu API zařízení a umožňují tak například ovládat fotoaparát, získat GPS polohu zařízení, pracovat s kontakty nebo s kalendářem. Aby takový plugin fungoval, je potřeba napsat kód obsluhující každou platformu, na kterou má být plugin cílený. To znamená, že plugin fungující na platformě Android byl vytvořen v jazyce Java, viz Tabulka 1. Funkcionalita zprostředkovaná nativním kódem je následně předána do hybridní aplikace pomocí JavaScriptu.

V souvislosti s Apache Cordova se také mluví o PhoneGap. Jak uvádí Wilken [2, s. 11], společnost Adobe přispěla svým frameworkem do Apache Software Foundation právě pod jménem Cordova. Dnes je PhoneGap distribucí Apache Cordova, nebo jinými slovy se jedná o Apache Cordova s přídavnými komerčními funkcemi, které jsou přímo podporovány společností Adobe.

Framework Apache Cordova v sobě obsahuje řadu CLI nástrojů (viz **CLI nástroje**), díky kterým jsou spravovány pluginy, projekty, ale hlavně platformy, na kterých by výsledná aplikace měla běžet (viz kapitola 1.1.3). Pro každou zvolenou platformu lze pak pomocí těchto nástrojů sestavit instalační balíček, který je možno distribuovat, a to buď manuálně uložením na cílové zařízení, nebo uložením do „obchodů“ s aplikacemi jednotlivých platforem. Pro Android je to Google Play Store, pro Windows jím je Windows Store, a v případě iOS se jedná o App Store. Následuje již jen instalace daného balíčku aplikace.

2.2 Mobile Angular UI

Dle Mobile Angular UI dokumentace [22] je tento framework „blízkým příbuzným“ tzv. Twitter Bootstrap, což je front-end framework pro rychlou a jednoduchou tvorbu webových stránek. Mobile Angular UI tento framework vytvořený společností Twitter obohatil o několik dalších prvků pro použití v mobilním zařízení. Tak jako ostatní frameworky je tento postaven na JavaScriptu a HTML, a tak vývoj probíhá právě v těchto jazycích.

Samozřejmostí je, že výsledná aplikace je multiplatformní, protože její finální sestavení probíhá za pomoci Apache Cordova nebo Adobe PhoneGap, viz kapitola 2.1.

2.3 Sencha Touch

Autor článku [23] uvádí, že framework Sencha Touch je založený na technologiích HTML a JavaScript. Ve svém jádru skrývá JavScriptový framework ExtJS, který je dle autora článku jeden z nejpopulárnějších. Sencha Touch je velice oblíbený především v komerční sféře a tomu také odpovídají ceny některých produktů, které pod společnost Sencha patří a jsou vytvořeny tak, aby velmi dobře spolupracovaly s frameworkem Sencha Touch.

Na webu společnosti, stojící za Sencha Touch [24], je uvedeno, že framework využívá techniky hardwarové akcelerace, aby bylo možné docílit vysoko výkonných UI komponent. Svou technologií se snaží uživatelům přinést co nejvíce nativní zážitek, a proto je framework také plně integrován s frameworky PhoneGap a Cordova. Díky integrovanému balíku pro tvorbu grafů a přehledů dokáže vytvořit mnoho druhů grafických znázornění dat.

2.4 Framework Ionic

Ionic framework je framework primárně vyvíjený týmem společnosti Ionic. Dle oficiálního webu společnosti Ionic [25] byl vytvořen v roce 2012 a od doby svého vzniku jeho popularita rychle vzrůstá. Stal se primární volbou pro vývoj mobilních hybridních aplikací a, jak uvádí Wilken [2, s. 9], každý měsíc vznikne 20 000 aplikací postavených právě na tomto frameworku.

Ionic je jakousi nadstavbou frameworku Apache Cordova. Funguje tak na stejném principu a vývojářům přináší vlastní grafické prvky, styly a různé knihovny.

Základní vlastností frameworku Ionic je to, že obsahuje sadu komponent, které chybí v HTML, ale jsou běžné v nativních aplikacích. Tyto komponenty Ionicu jsou postaveny na HTML, CSS a JavaScriptu a chovají se stejně jako komponenty nativní [2].

Pro práci s tímto frameworkem je zapotřebí mít nainstalované CLI nástroje Ionic a Apache Cordova, kterými se provádí nejdůležitější operace s projektem. Při tvorbě projektu pomocí těchto nástrojů se automaticky do aplikace naimportují odkazy na potřebné knihovny Ionicu, AngularJS a Apache Cordova. Postup tvorby projektu je uveden v kapitole 4.4.1 této práce. Společně fungují způsobem, jakým je naznačeno v ukázce procesu spuštění a užití fotoaparátu telefonu [2, s. 7-8].

1. Uživatel klepne na tlačítko v mobilní aplikaci, což je grafická komponenta Ionicu.
2. Tlačítko zavolá tzv. controller – kód, který má na starosti Javascriptovou část aktuálního pohledu (view) –, který zavolá komponentu Cordova pomocí JavaScript API.
3. Cordova komunikuje se zařízením pomocí nativního SDK a vyžádá spuštění aplikace Fotoaparát.
4. Operační systém zařízení otevře aplikaci Fotoaparát a uživatel je schopen pořídit fotografii.
5. Když uživatel fotografii potvrdí, aplikace Fotoaparát se zavře a vrátí vytvořená obrazová data do Cordovy.
6. Cordova předá data zpět do Angular controlleru.
7. Vizuální zobrazení dat je provedeno zpět v komponentě Ionicu.

CLI nástroje frameworku Ionic zprostředkovávají ovládání projektu. Pomocí těchto nástrojů lze projekt vytvořit, zobrazit jeho náhled, sestavit jej (zkompilovat) nebo odeslat do zařízení a spustit. Neméně zajímavou součástí je taky knihovna ikon, které lze v aplikaci volně využít. Tyto ikony jsou designově stylizovány buď do platformy iOS nebo Android.

3 Jazyky JavaScript a AngularJS

JavaScript, AngularJS a jQuery patří k dnes nejvyužívanějším technologiím pro tvorbu dynamických webových stránek. Z počátku byly webové stránky pouze statické – soubor zpracovaný webovým prohlížečem je identický s tím, který je uložen na serveru a je stažen do prohlížeče ke zpracování. V případě potřeby komplexnější webové stránky se toto řešení stává nevýhodným, protože je potřeba míti na webovém serveru velké množství HTML souborů. V dnešní době jako vhodnější řešení přichází v potaz využití skriptů, které používají data ze serveru a dynamicky sestaví HTML soubory, které jsou následně renderovány neboli zobrazovány v cílovém prohlížeči.

Existují v zásadě dva způsoby využití skriptování, a to tzv. „client-side“ nebo „server-side“ skriptování. V prvním případě se jedná o proces zaslání javascriptového kódu spolu s webovou stránkou z webového serveru do prohlížeče. Tento kód se provede prohlížečem (klientem) buď v době načítání, nebo po dokončení načítání webové stránky. Výhodou skriptování na straně klienta je jeho mnohem rychlejší interakce s uživatelem. JavaScript, AngularJS a jQuery jsou dnes nejběžnějšími formami skriptování na straně klienta – „client-side“ skriptování.

Skriptování na straně serveru v sobě zahrnuje 2 typy – tzv. server-side šablony a AJAX request handlers, což je volně přeloženo jako zpracovávače AJAX požadavků. Server-side šablony využívají PHP, .NET nebo Java jako aplikace běžící na serveru, které generují celou HTML stránku nebo jen její části, a to dynamicky dle toho, jako jsou od uživatele vyžadovány. Výhodou je, že veškeré zpracování dat je prováděno na serveru a zdrojová data webových stránek nejsou přenášena skrze Internet. Nevýhodou je fakt, že toto řešení vyžaduje mnohem více zpracování a snižuje se tím škálovatelnost aplikace.

AJAX handler je aplikace vracející tzv. raw data ve formě JSON (JavaScript Object Notation) nebo XML (Extensible Markup Language) do prohlížeče, a to v odpovědi na AJAX požadavek, kde AJAX znamená Asynchronní JavaScript plus XML [26].

3.1 JavaScript

Jak uvádí web Mozilla Developers [29], první JavaScript byl vytvořen panem Brendanem Eichem ze společnosti Netscape Communications Corporation (dnešní AOL). Jedná se

o multiplatformní, na prototypech založený skriptovací jazyk, který může fungovat jako procedurální i objektově orientovaný jazyk.

Díky JavaScriptu lze psát komplexní aplikace s přímým přístupem k událostem prohlížeče a DOM objektům – lze přidat, změnit nebo odstranit části webu bez potřeby nového načtení stránky.

Základy jazyka JavaScript formuje skriptovací jazyk ECMAScript, který zároveň působí jako jeho standard. Tento standard je vydáván společností ECMA International zabývající se tvorbou standardů. Autoři webu Mozilla Developers uvádí, že *"Všechny moderní prohlížeče od roku 2012 plně podporují ECMAScript 5.1. Starší prohlížeče podporují alespoň ECMAScript 3"* [27].

3.1.1 JavaScript a DOM

Díky DOM má JavaScript přehled o webové stránce, HTML dokumentech, XML dokumentech a jejich součástech (elementech). Každý element (dokument jako takový, hlavička, tabulka, nadpis, vstupní pole, …) je součástí DOM a díky tomu je přístupný a může s ním být manipulováno skriptovacím jazykem, jako je JavaScript. To znamená, že lze obsah stránky přidat nebo upravit programově. Ve zdrojovém souboru skriptu jsou přístupné jednotlivé datové typy objektového modelu dokumentu jako document, element, nodeList, attribute [28].

- document – jedná se o kořenový objekt v DOM
- element – odkazuje na element nebo na vlákno typu element. Je vráceno například při volání funkce `document.createElement()`, která vytvoří nový element v DOM.
- nodeList – pole elementů. Je vráceno například při volání funkce `document.getElementsById()`, kdy se funkcí dotazujeme na vrácení všech elementů se specifickým id. Navráceno je pak pole elementů, jež je přístupné jako klasické pole pomocí indexů.
- attribute – atribut, vlastnost elementu. Je vracen například při volání výrazu `document.getElementById("div1").style.width`, kdy je navrácena reference na atribut "width" (šířka) elementu identifikovaného jako "div1". Atribut lze buď jen vypsat, nebo i nastavit jeho hodnotu.

Vztah DOM a skriptovacího jazyka znázorňuje následující rovnice (1) [28]:

$$\text{API (HTML nebo XML)} = \text{DOM} + \text{JS (skriptovací jazyk)} \quad (1)$$

3.2 AngularJS

AngularJS je open source projekt od společnosti Google, který byl založen v roce 2009. Přináší programátorům možnost psát aplikace bez potřeby používat „server-side“ jazyky jako PHP, Ruby nebo Java [2].

Využívá MVC framework, což znamená, že rozkládá aplikaci na 3 logické celky [30]:

- Model – představuje veškerou logiku spojenou s daty, se kterými uživatel pracuje – například data přenášená mezi View a Controllerem.
- View (pohled) – představuje veškerou grafickou logiku.
- Controller – působí jako rozhraní mezi modelem a pohledem.

Angular řeší uživatelský vstup, manipuluje s daty na straně klienta a kontroluje, jak jsou elementy v prohlížeči zobrazeny. Mezi jeho hlavní výhody lze zařadit jeho kompatibilitu, a to díky tomu, že je postaven na JavaScriptu. Dále je to tzv. data-binding neboli provázání dat s aplikací, které je řešeno například tzv. službami, jako je „$scope“ nebo „$rootScope“, díky kterým má pohled přístup k datům [26].

3.3 jQuery

Dle úvodních slov z webu jQuery.com, je jQuery rychlá, malá a na schopnosti bohatá JavaScript knihovna. Udávají, že zjednodušuje manipulaci s HTML dokumentem, reakce na události a animace, a to díky jednoduchému API, které dokáže pracovat napříč velkým množstvím webových prohlížečů. Autor jQuery John Resig, se o myšlence o existenci JavaScript knihovny využívající selektory k svázání JavaScript funkcí s různými HTML elementy, poprvé zmínil v článku [31] "Selectors in JavaScript" na svém blogu, kde popsal své návrhy, jak by tato knihovna měla fungovat.

Knihovna jQuery zpřístupňuje své metody a vlastnosti pomocí dvou vlastností objektu window, a to jQuery a $. Znak $ je alias pro jQuery. Je více využíván, protože je kratší a rychleji se píše. Příkladem může být reakce na kliknutí na tlačítko:

```
$("tlacitko1").click(function( event ) {
  console.log("Tlačítko bylo stisknuto");
})
```

Výše uvedený kód provede následující: u elementu s id "tlacitko1" naslouchá na události click() - kliknutí. Po této události do konzole zapíše hlášení "Tlačítko bylo stisknuto".

Mezi další události, na které lze s pomocí jQuery reagovat, jsou např. hover(), která je vyvolána v případě, kdy ukazatel myši "přejede" nad elementem nebo událost keydown(), kdy uživatel stiskne klávesu na klávesnici či klávesu drží stisknutou delší dobu. Událost keyup() je naopak vyvolána v momentě, kdy uživatel přestane klávesu tisknout [32].

3.4 Node.js

V kapitole o JavaScriptu se nelze nezmínit o Node.js, protože se jedná o nedílnou součást při vývoji mobilních hybridních aplikací, a to zejména při použití frameworku Ionic. Node.js je serverový framework založený na vysoce výkonném open-source JavaScript enginu V8 od Google Chrome, který je napsán v jazyce C++. Implementuje standard ECMAScript a je schopen běžet na všech platformách včetně Windows XP. V8 je využíván především v prohlížeči Chromium, Node.js a dalších vestavěných (embedded) aplikacích. Kompiluje a provádí JavaScript kód, stará se o alokaci paměti a vyznačuje se výkonným a přesným garbage collectorem (viz **Garbage collector**) [33].

Důležitou součástí Node.js je *npm* – node package manager, který byl v roce 2009 založen jako open-source projekt pro vývojáře v JavaScriptu na sdílení balíčků modulů jejich kódu. Tzv. *npm Registry* je veřejná kolekce balíčků s open-source kódem pro Node.js, webové a mobilní aplikace a mnoho dalšího. Program *npm* je zároveň CLI klientem pro instalaci a publikování těchto balíčků [34].

4 Tvorba konkrétní mobilní aplikace

V rámci praktické části této bakalářské práce byla vyvinuta mobilní aplikace s názvem „Příprava expedice zakázek“, která má za úkol pomáhat pracovníkům zákaznické firmy v urychlení přenosů informací mezi nimi a informačním systémem OR-SYSTEM Open, a to za pomoci přenosných datových terminálů.

4.1 OR-SYSTEM Open

Současný informační systém OR-SYSTEM Open je nástupcem informačního systému OR-SYSTEM, jehož kořeny sahají do programového systému ORFERT. Původní německý systém ORFERT od společnosti Ott&Rehorst Computersysteme byl na počátku 90. let minulého století přeložen do českého jazyka a stávající moduly pro výrobu a logistiku byly rozšířeny o finančně-ekonomické moduly. Takto vznikl informační systém OR-SYSTEM, který se vyznačoval znakovým uživatelským prostředím neboli CLI (které bylo podporováno až do roku 2010) a na svou dobu velkými konfiguračními možnostmi. V roce 1998 došlo ke změně aplikačního principu a bylo vytvořeno grafické uživatelské prostředí na straně klienta za pomoci systému ISA Dialog Manager, které bylo plně kompatibilní s původním prostředím znakovým.

Kolem roku 2010 započala tvorba jádra nového systému v technologii Java. V roce 2012 byla podepsána smlouva o spolupráci se společností ORTEX Hradec Králové na vývoji produktu ORCore, což je technologické jádro a zároveň framework, na jehož základech je budován nový informační systém OR-SYSTEM Open. Grafické uživatelské prostředí je nyní řešeno v technologii Java. I nadále je systém složen ze dvou částí, a to části logisticko-výrobní a části ekonomické.

Pomocí jeho webových služeb s ním komunikuje již několik mobilních aplikací, včetně mobilní aplikace „Příprava expedice zakázek“. Tyto aplikace slouží pouze jako klientské aplikace, a tak veškerá manipulace s daty je prováděna v OR-SYSTEMu Open.

4.2 Analýza

Tato podkapitola popisuje analýzu stavu před zavedením mobilní aplikace a analýzu požadované mobilní aplikace.

4.2.1 Stávající řešení

Před zavedením mobilní aplikace „Příprava expedice zakázek“ probíhal proces přípravy seznamu zakázek k expedici následujícím způsobem.

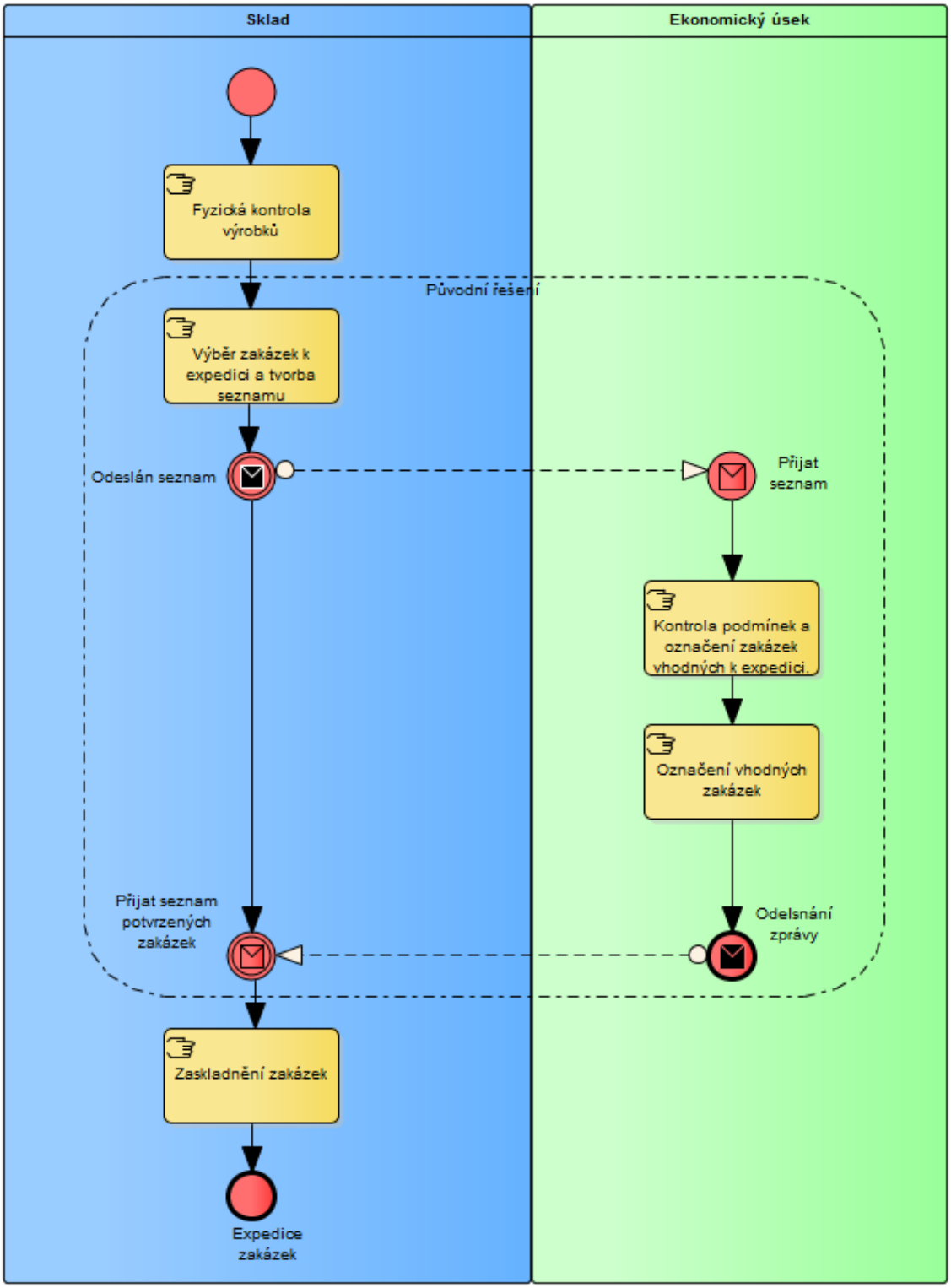
Pracovník skladu kontroluje výrobky ve výrobním procesu a zjišťuje potenciální datum dostupnosti k expedici. Výrobky, které jsou vhodné k expedici pro daný den, zaznamená do seznamu v papírové podobě. Tento seznam výrobků je předán do ekonomického úseku, kde jsou pracovníkem ověřeny podmínky, a to, zda je zaplacena zálohová faktura a zdá má zahraniční odběratel platné DIČ. Pokud zakázka vyhovuje podmínkám pro expedici, označí ji v seznamu a tento seznam předá zpět pracovníkovi skladu. Pracovník skladu tyto výrobky zaskladní a v době plánované expedice provede hromadné vyskladnění s možností tvorby dodacích listů. Toto řešení je naznačeno Business Process diagramem níže (viz Obrázek 1).

4.2.2 Požadovaný stav

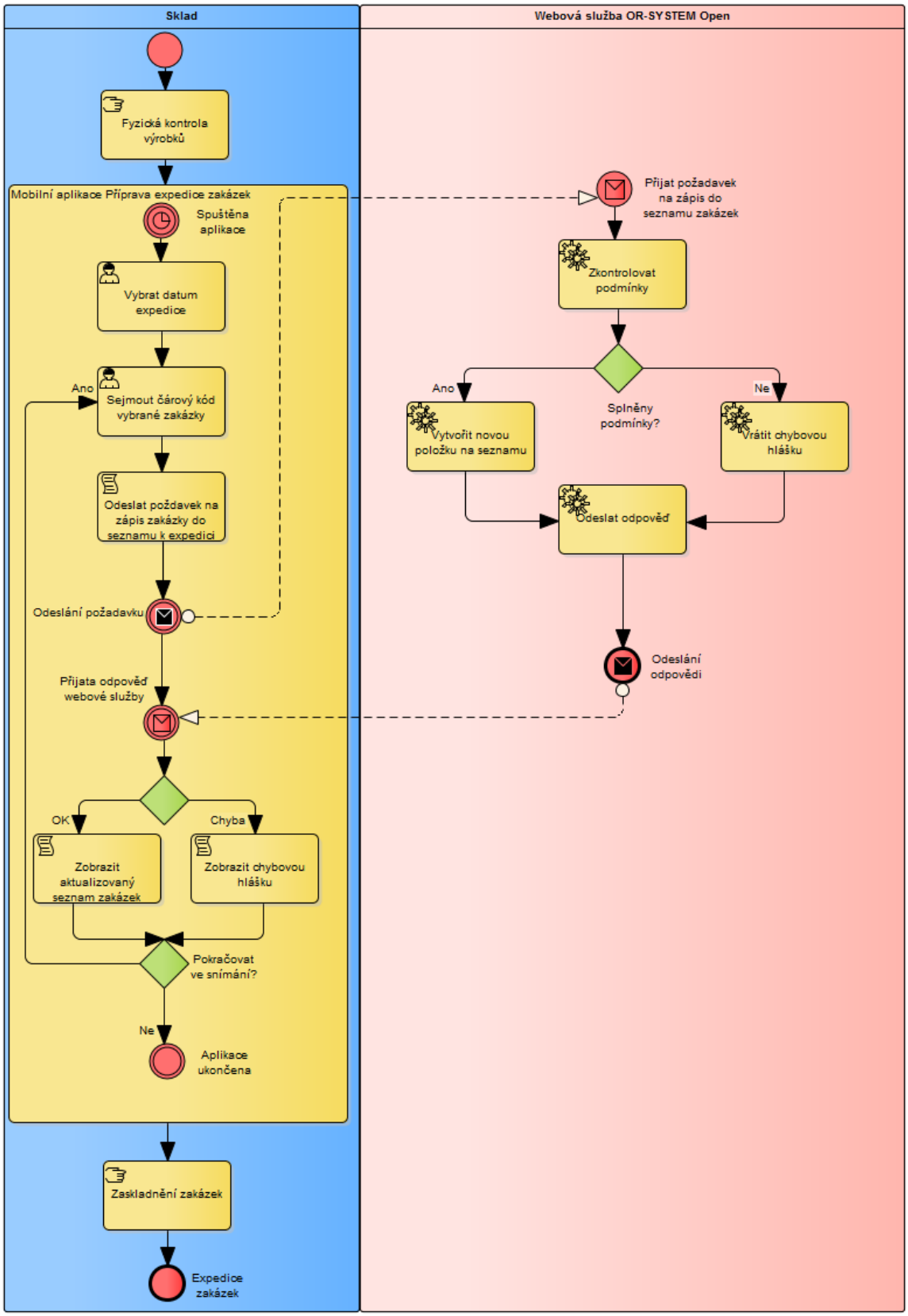
Požadavkem zákaznické firmy bylo, aby tvorba seznamu zakázek k expedici probíhala elektronicky. Tvorba seznamu by měla probíhat za pomoci mobilního zařízení se skenerem čárových kódů, a to snímáním čárových kódů zastupujících identifikační číslo zakázky na průvodním listě zakázky. Mobilním zařízením využívaným ke snímání čárových kódů by měl být datový terminál s operačním systémem Android, a to především kvůli mobilitě. Vytvořený seznam by měl být administrován pracovníkem skladu a pracovníkem ekonomického úseku s možností využití pro další akce, jako je například hromadný příjem na sklad z výroby a hromadná tvorba dodacích listů, které jsou k expedované zakázce přiloženy.

Aplikace by pak měla fungovat následovně. Poté, co pracovník spustí aplikaci, zobrazí se mu obrazovka s aktuálně nastaveným datem expedice, se kterou se pracovalo naposledy, kdy toto datum je v paměti uloženo jako poslední použité. Uživatel může toto datum změnit nebo jen potvrdit. Po potvrzení se uživateli zobrazí obrazovka s polem pro zadání krátkého čísla řádku výrobní zakázky a zároveň seznam zakázek, které jsou již na seznamu pro aktuálně vybrané datum expedice. Dále uživatel snímačem sejme čárový kód z průvodního listu zakázky. Systémem je následně zkontrolováno, zda jsou splněny podmínky: zda již není zakázka hotova, zda existuje vazba na otevřenou obchodní zakázku a zda již nebylo přiděleno jiné datum expedice. Pokud jsou splněny tyto podmínky, systém založí novou položku v seznamu zakázek k expedici.

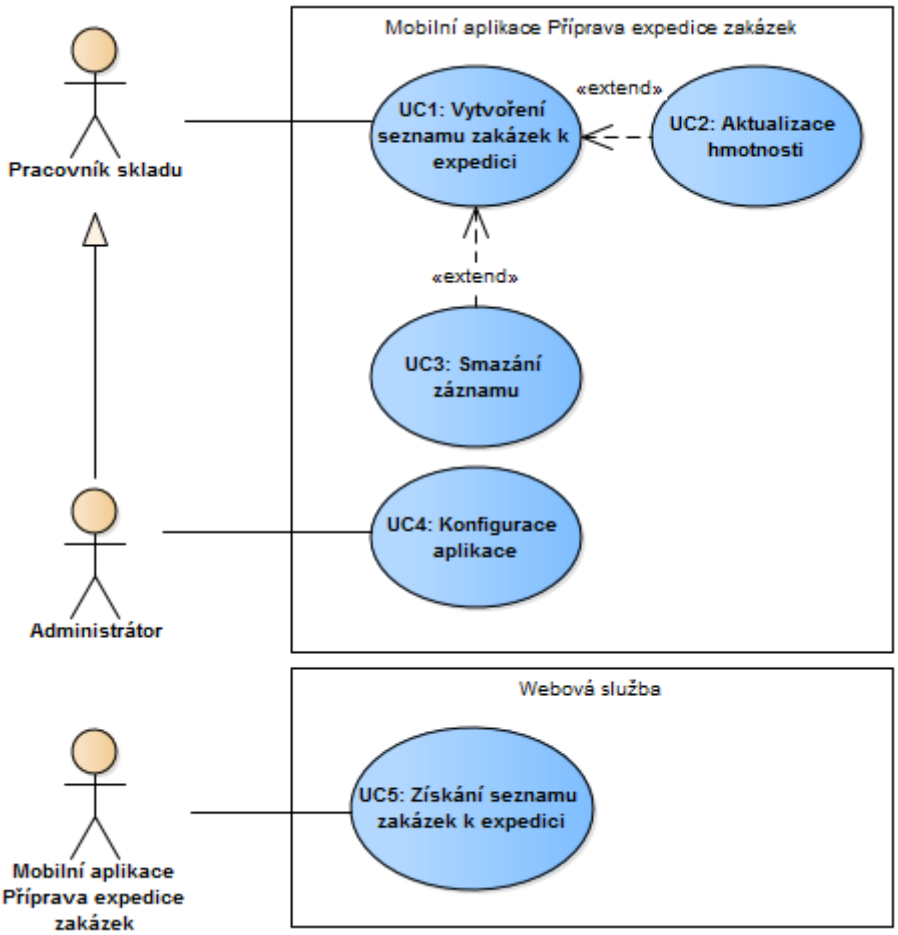
Řešení pomocí mobilní aplikace a datového terminálu je znázorněno na Business Process diagramu níže (viz Obrázek 2) spolu s diagramem případů užití (Obrázek 3).



Obrázek 1: Business proces původního řešení přípravy expedice zakázek
Zdroj: vlastní



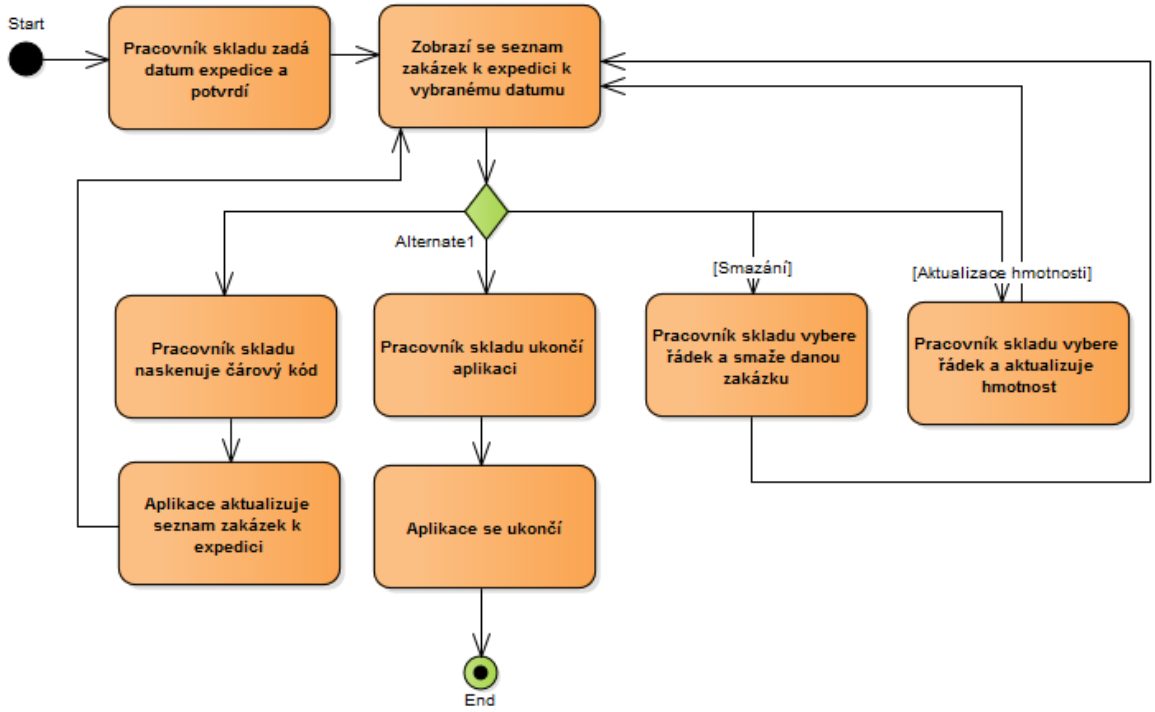
Obrázek 2: Business proces řešení s mobilní aplikací
Zdroj: vlastní



Obrázek 3: Diagram případů užití mobilní aplikace „Příprava expedice zakázek“
Zdroj: vlastní

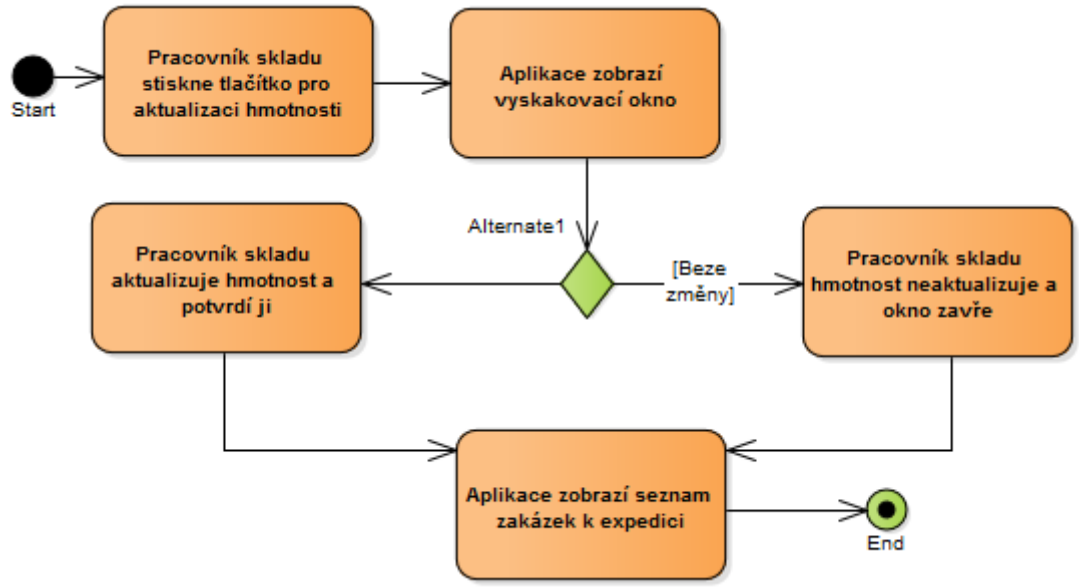
4.2.3 Rozbor jednotlivých případů užití

Prvním případem užití je *UC1 Vytvoření seznamu zakázek k expedici*. Pracovník skladu nebo jiná obsluha chce mobilní aplikaci „Příprava zakázek k expedici“ využít k vytvoření seznamu zakázek k expedici. Jedná se o hlavní případ užití, protože se jedná o činnost, pro kterou byla aplikace vytvořena a jejíž proces se snaží optimalizovat. Diagram aktivit tohoto případu užití (Obrázek 4) popisuje proces tvorby tohoto seznamu.



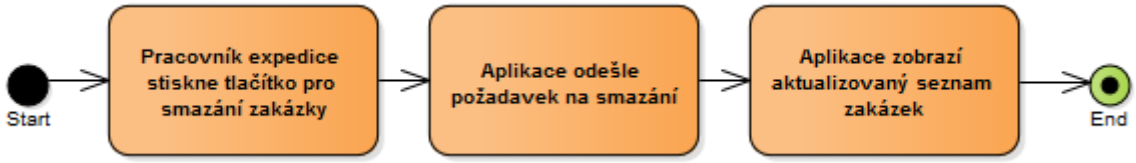
Obrázek 4: Diagram aktivity pro UC1 Vytvoření seznamu zakázek k expedici
Zdroj: vlastní

Rozšířením prvního případu užití jsou UC2 a UC3, kdy v prvním z nich (*UC2 Aktualizace hmotnosti* – viz Obrázek 5) pracovník skladu aktualizuje hmotnost kusu zakázky.



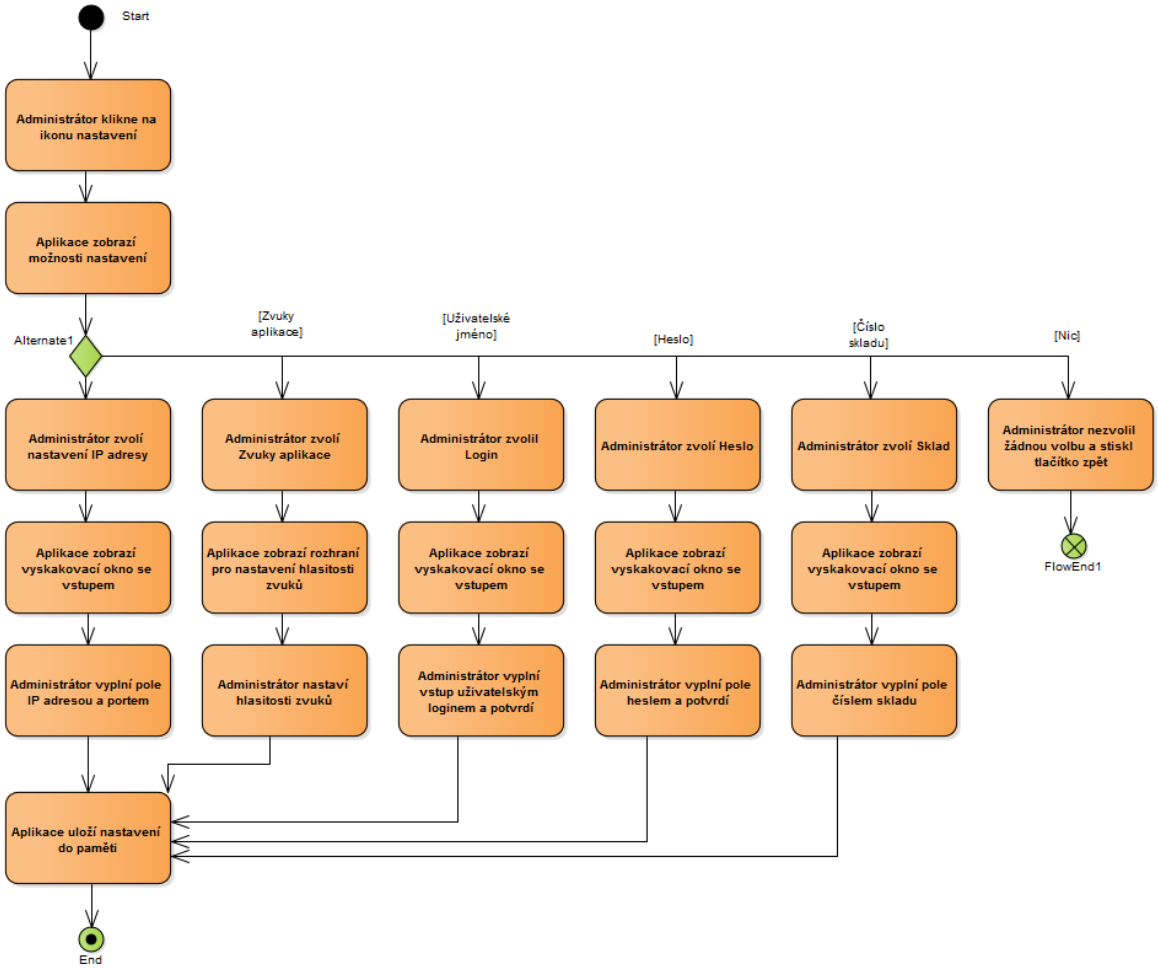
Obrázek 5: Diagram aktivity pro UC2 Aktualizace hmotnosti
Zdroj: vlastní

V druhém případu užití rozšiřujícím případ UC1 je případ užití *UC3 Smazání záznamu* (viz Obrázek 6), kdy pracovník skladu či jiná obsluha vymaže zakázku ze seznamu zakázek k expedici.



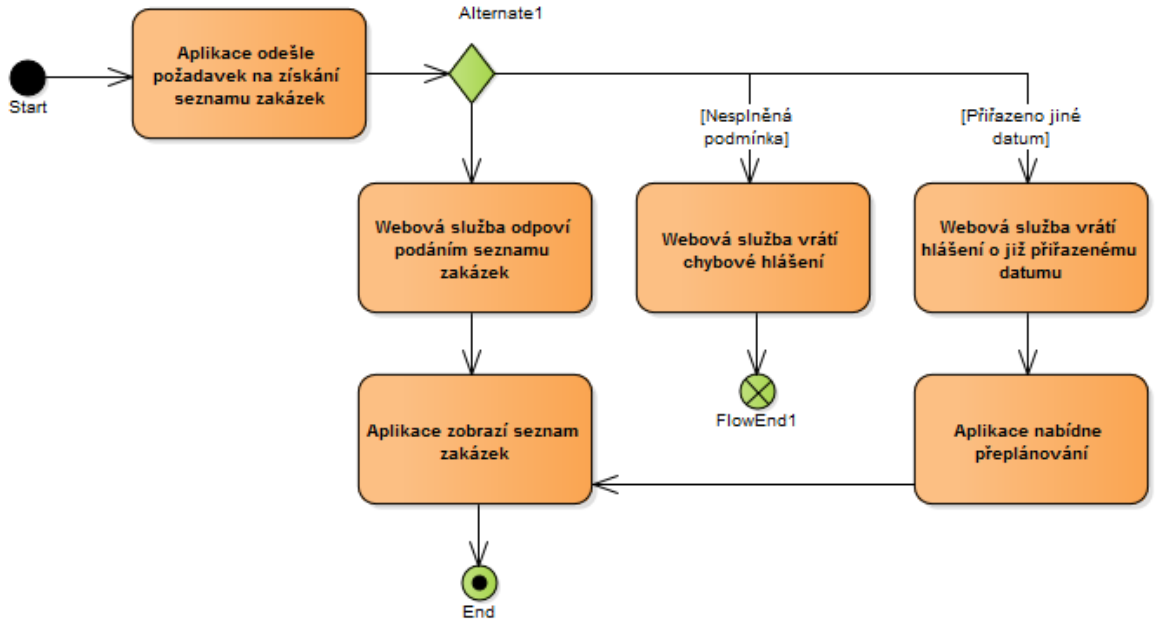
Obrázek 6: Diagram aktivity pro UC3 Smazání záznamu
Zdroj: vlastní

Případem užití pro Administrátora je *UC4 Konfigurace aplikace*. Konfigurovat může například IP adresu a port webové služby nebo přihlašovací údaje.



Obrázek 7: Diagram aktivity pro UC4 Konfigurace aplikace
Zdroj: vlastní

Poslední případ užití *UC5 Získání seznamu zakázek k expedici* se týká webové služby, kdy mobilní aplikace působí jako aktér „Příprava expedice zakázek“ a kontaktuje webovou službu.



Obrázek 8: Diagram aktivity pro UC5 Získání seznamu zakázek k expedici
Zdroj: vlastní

4.3 Prostředky pro vývoj

Prostředky pro vývoj mobilní aplikace „Příprava expedice zakázek“ lze rozdělit na softwarové a hardwarové. Mezi softwarové řadíme v prvé řádě framework Ionic. Dále je to vývojové prostředí NetBeans a technologie SOAP jako prostředek pro komunikaci s webovou službou. Do hardwarových prostředků patří mobilní datový terminál, na kterém má aplikace fungovat.

4.3.1 Framework Ionic

Vývoj mobilní aplikace probíhal za použití frameworku Ionic. Proto, aby mohl být použit Ionic, je zapotřebí na počítač, na kterém probíhá vývoj, nainstalovat důležité součásti. Základem je Node.js a jeho správce balíčků *npm* (viz kap. 3.4). Pomocí něho se nainstalují frameworky Ionic a Apache Cordova. Jak již bylo uvedeno v kapitole 2.4, framework Ionic je nadstavbou frameworku Apache Cordova, a tak je ke správnému chodu potřebná jeho přítomnost v systému. Instalace frameworků se provádí jednoduchým příkazem:

```
C:\> npm -g install cordova ionic
```

Přepínač `-g` správci balíčků *npm* říká, aby oba frameworky nainstaloval globálně. Díky tomu lze utilitu ionic nebo cordova spustit v kterémkoli adresáři, protože jsou nainstalovány do adresáře, který je uložen v systémové proměnné PATH. Pokud by příkaz nebyl volán s přepínačem `-g`, frameworky by se nainstalovaly do aktuálního adresáře, ve kterém je příkazový řádek aktuálně nastaven. Správce balíčků *npm* provede instalaci frameworků v pořadí, v jakém jsou uvedeny v příkazu. Dalším krokem je vytvořit nový projekt, což je popsáno v kapitole 4.4.1.

4.3.2 NetBeans

Jelikož je ve firmě OR-CZ, spol. s r.o., konkrétně v divizi OR-SYSTEM, využíváno jako primární vývojové prostředí NetBeans, probíhal vývoj aplikace právě v něm. Integrované vývojové prostředí NetBeans je na vývoj hybridních aplikací připraveno, a tak lze s projektem pracovat ihned.

Značným pomocníkem při vývoji je webový prohlížeč Google Chrome, který se s pomocí doplňku NetBeans Connector dokáže napojit na vývojové prostředí NetBeans, kde pak lze zobrazit výstup na konzoli, který by byl jinak přístupný pouze skrze samotný webový prohlížeč. Mimo přenosu výstupu na konzoli umožňuje toto propojení i efektivní ladění (debugging).

Díky přítomnosti instalace frameworku Apache Cordova je možné provést spuštění na mobilním zařízení, které je připojeno pomocí USB rozhraní, rovnou z prostředí NetBeans.

4.3.3 SOAP

Aplikace byla navržena tak, aby sloužila jako klientský terminál, který komunikuje s webovou službou pomocí SOAP. World Wide Web Consorcium ve svém doporučení [35] popisuje SOAP jako lehký a nenáročný protokol pro výměnu strukturovaných informací v decentralizovaném a distribuovaném prostředí. Pro komunikaci využívá technologií XML.

Komunikace s webovou službou v aplikaci „Příprava expedice zakázek“ je řešena pomocí speciálního modulu pro AngularJS, který do aplikace zavádí potřebnou funkcionalitu. Tento modul byl speciálně upraven dle cílové webové služby.

Základem SOAP zprávy (message), které je komunikačním prostředkem mezi webovou službou a klientem, je obyčejný XML soubor, který má následující strukturu [36]:

- Envelope element (obálka) – díky tomuto elementu je XML dokument identifikován jako SOAP zpráva.
- Header element (hlavička) – obsahuje hlavičkové informace – například verze nebo kódování znaků.
- Body element (tělo) – obsahuje obsah volání služby nebo její odpověď.
- Fault element (chyby) – obsahuje stavové informace a chybová hlášení.

4.3.4 Datový terminál Chainway C4000

Jako zařízení pro čtení čárových kódů byl vybrán datový terminál Chainway C4000, a to především díky předinstalovanému operačnímu systému Android. Model C4000 existuje v několika variantách. Vybrána byla varianta s lineárním skenerem čárových kódů. Mezi další varianty řadíme variantu se skenerem 2D kódů, známých také jako QR kódy, a variantu se čtečkou RFID čipů.

Jak ukazuje Obrázek 9, datový terminál Chainway C4000 je také dodáván s pistolovým držákem, který v sobě obsahuje externí baterii a tlačítko pro spuštění skeneru čárových kódů. Díky tomuto řešení je obsluha zařízení pro uživatele velmi pohodlná.

Softwarově je terminál vybaven operačním systémem Android ve verzi 4.4 (dle [37]) a obsahuje základní softwarovou výbavu, kterou lze u operačního systému očekávat. Mimo to, lze na zařízení najít aplikace pro konfiguraci skeneru čárových kódů.

Zařízení také nabízí možnost vložit SIM kartu. Není tedy omezeno pouze na WiFi připojení, ale lze jej používat i v místech, kde je dostupné pouze datové připojení mobilních operátorů.

Obrázek 9: Datový terminál Chainway C4000
Zdroj: [37]

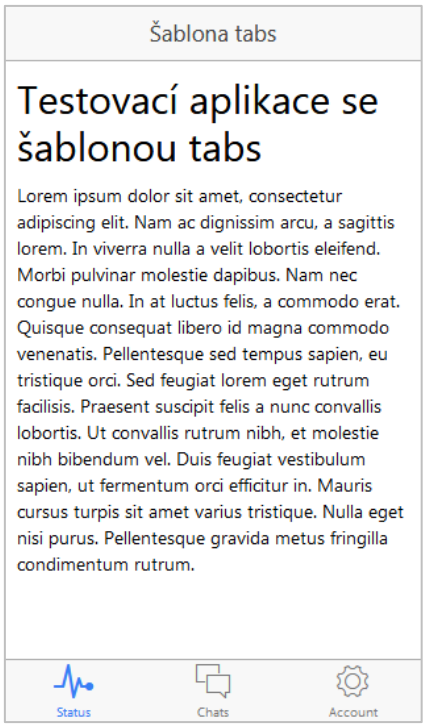
4.4 Tvorba aplikace a její struktura

V následující kapitole jsou popsány možnosti tvorby nového projektu pracujícího s frameworkem Ionic. Dále je zde popsána struktura projektu tvořené aplikace, její uživatelské rozhraní a webová služba, se kterou mobilní aplikace spolupracuje.

4.4.1 Tvorba nového projektu

K vývoji mobilní aplikace „Příprava expedice zakázek“ byly zvoleny prostředky uvedené v kapitole 4.3. K tvorbě nového projektu mobilní aplikace ve frameworku Ionic existuje několik přístupů. V této práci budou popsány 2 způsoby. Prvním způsobem, jak založit nový projekt, je využití CLI nástrojů, který autor této bakalářské práce preferuje. Druhým způsobem je využití online nástroje od tvůrců frameworku Ionic zvaný Ionic Creator.

Pro vytvoření nového Ionic projektu pomocí příkazové řádky (tedy prvním způsobem) je potřeba se nejprve rozmyslet, jaký typ šablony chce uživatel využít. Na výběr jsou tři základní šablony vzhledu mobilní aplikace: tabs, sidemenu a blank. Šablona tabs využívá modelu tzv. záložek (viz Obrázek 10). Stručně řečeno, stránky (pohledy) aplikace jsou tříděny do záložek, které tvoří textový popisek, ikona či kombinace obojího.



Obrázek 10: Příklad použití šablony tabs
Zdroj: vlastní demo aplikace

Šablona sidemenu vytváří klasický model s postranním vysouvacím menu, které se vyvolává kliknutím na ikonu menu v levém horním rohu (viz Obrázek 11). Poslední předdefinovaná šablona blank vytvoří prázdnou aplikaci a je na uživateli, aby si vytvořil svůj model navigace či použil některý z výše uvedených.

Obrázek 11: Příklad tlačítka menu v šabloně sidemenu
Zdroj: vlastní demo aplikace

Příkazem, který spustí tvorbu nového projektu, je:

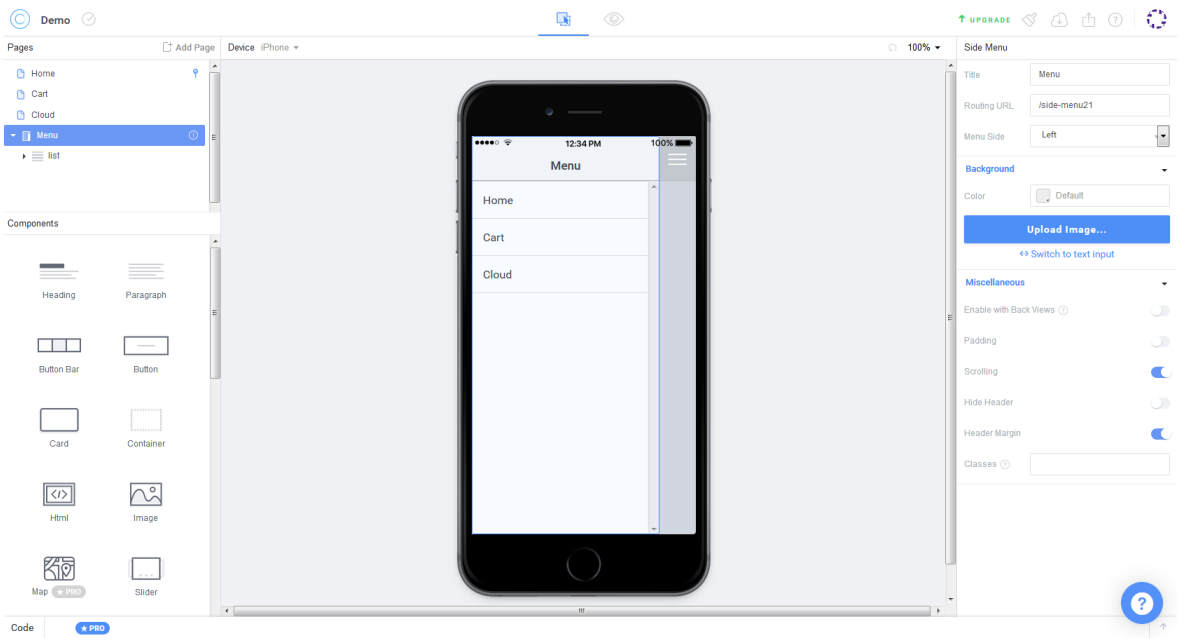
```
C:\Vyvoj> ionic start pripravaExpedice blank
```

Část `C:\Vyvoj>` v příkazovém řádku vyjadřuje, v jaké složce se uživatel právě nachází. Je to tedy cílová složka, ve které se projekt vytvoří. Příkazem říkáme programu `ionic`, aby vytvořil nový projekt s názvem `pripravaExpedice` a aby použil šablonu `blank`. Šablona `blank` byla vybrána z důvodu, že plánovaná aplikace by měla mít 2 základní, po sobě jdoucí pohledy a jeden samostatný pohled pro konfiguraci aplikace. Není tedy potřeba vytvářet aplikaci se záložkami nebo postranním vysunovacím menu.

Po spuštění výše uvedeného příkazu započne program Ionic provádět tvorbu nového projektu, kdy stáhne vybranou šablonu ze svého repositáře a provede instalaci potřebných node.js modulů pro správnou funkci projektu a práci s ním.

Takto se vytvoří nový projekt pomocí příkazového řádku. Výhodou je, že je plně na uživateli, jak si aplikaci a její rozložení naprogramuje a jaké prvky k tomu použije. Nevýhodou tohoto řešení je, že pokud ještě nemá programování těchto aplikací tzv. v krvi, je potřeba více využívat referenčních materiálů a čas programování se může prodloužit.

Pokud chce uživatel využít možnosti druhé, nástroje Ionic Creator, jeho prvním krokem musí být vytvoření účtu Ionic ID. K tomu je ostatně naveden při navštívení webové stránky nástroje Ionic Creator na adrese https://creator.ionic.io/. Po registraci účtu a přihlášení do služby se zobrazí rozhraní (viz Obrázek 12), kde je možné si nadefinovat rozložení grafických prvků a jednotlivých pohledů své aplikace. V základní verzi (zdarma) lze takto vytvořit pouze jeden projekt. Možnosti jsou omezené, a tak si lze aplikaci pouze předpřipravit pro další vývoj ve vývojovém prostředí. V plné verzi je pak možné aplikaci upravit mnohem více a téměř celou ji skrze tento nástroj naprogramovat.



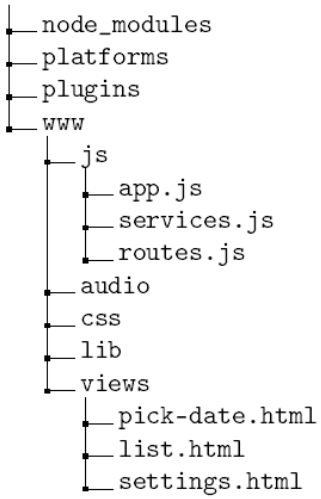
Obrázek 12: Webové rozhraní nástroje Ionic Creator
Zdroj: http://creator.ionic.io/

Pokud je návrh rozložení dokončen, je možné si celou aplikaci stáhnout v archivu ZIP a dále s ním pracovat. To znamená, že celý tento projekt si lze po extrahování z archivu

naimportovat do vybraného vývojového prostředí. V prostředí NetBeans je to jednoduché. Stačí zvolit možnost *Otevřít projekt* a následně jen vybrat adresář se zdrojovými soubory z místa jeho uložení.

4.4.2 Struktura projektu

Základní adresářová struktura projektu je znázorněna níže (Obrázek 13):



Obrázek 13: Adresářová struktura projektu
Zdroj: vlastní

Adresář *node_modules* obsahuje lokální kopie JavaScriptových modulů, které jsou pro běh aplikace nezbytné anebo rozšiřují její funkcionalitu.

V adresáři *plugins* jsou uloženy pluginy od Apache Cordova, které aplikace využívá. Jak již bylo zmíněno v kap. 2.1, framework Apache Cordova umožnuje instalaci pluginů. Tyto pluginy jsou po jejich instalaci uloženy do adresáře *plugins*, odkud jsou při sestavení instalačního balíčku mobilní aplikace převzaty a zkompilovány.

Adresář *platforms* v sobě obsahuje podadresáře se soubory jednotlivých platforem, které jsou k projektu přiřazeny. V těchto podadresářích se tak například uloží zkompilovaný instalační balíček aplikace. Obsah podadresářů je dán cílovou platformou.

Pro programátora mobilní aplikace v Ionicu je nejdůležitějším adresářem adresář *www*. Ten obsahuje veškeré HTML, CSS a JavaScript soubory, se kterými se pracuje. V podadresáři *lib* se skrývá jádro frameworku Ionic se všemi svými zdrojovými soubory a další důležité součásti.

V podadresáři *views* jsou uloženy HTML soubory, které určují rozložení UI aplikace. Mobilní aplikace „Příprava expedice zakázek“ má v této složce 3 zdrojové soubory HTML:

- *pick-date.html* – definuje rozložení UI prvků prvního pohledu, který se po spuštění aplikace spustí. Obsahuje tlačítko pro otevření kalendáře, který je řešen externí JavaScriptovou komponentou.
- *list.html* – definuje rozložení UI pohledu druhého. Obsahuje pole pro vstup – konkrétně naskenované krátké číslo zakázky a seznam zapsaných zakázek pro expedici ve zvolený den.
- *settings.html* – definuje rozložení UI pohledu pro konfiguraci aplikace. V tomto pohledu je seznam veškerých konfiguračních položek, jako nastavení IP adresy, se kterou má aplikace komunikovat aj.

Podadresář *js* obsahuje zdrojové JavaScript soubory programovou logikou. Je plně na uživateli, jaké soubory zde vytvoří a jak je pojmenuje. Všechny JS soubory musí být definovány v *index.html*. Best practice je takové, že pro přehlednost se zdrojový JS kód rozdělí na několik souborů.

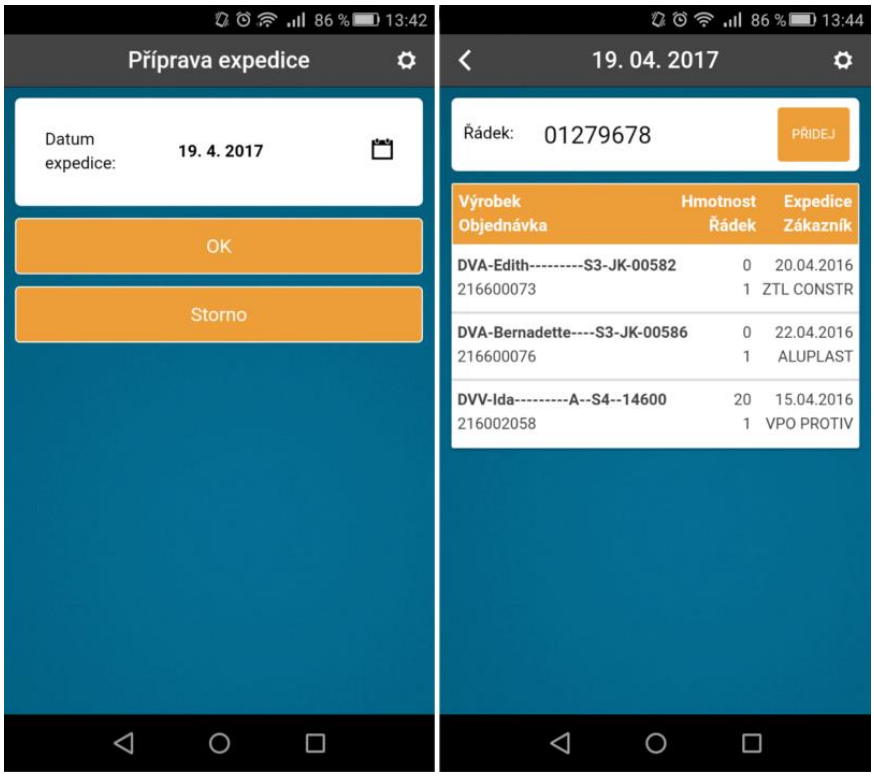
Tímto způsobem je řešena i mobilní aplikace „Příprava expedice zakázek“:

- *app.js* – v tomto souboru je definován modul, který mobilní aplikaci ovládá, a tzv. controllery (viz kap. 3.2). Při větším množství controllerů, je z důvodu přehlednosti lepší dát každý do zvláštního souboru. Je třeba také zmínit, že pokud si tak programátor přeje, lze mít celou JS část aplikace definovanou v jednom zdrojovém souboru.
- *routes.js* – v tomto souboru se definují pohledy (views), se kterými aplikace pracuje. Pokud je přítomen zdrojový HTML soubor pohledu, ale není o něm zmínka v router.js, nelze na něj v aplikaci přistoupit.
- *services.js* – tento soubor obsahuje definice dvou velice důležitých součástí mobilní aplikace „Příprava expedice zakázek“. Jsou zde 2 tzv. factory (lze přeložit jako továrny), které při zavolání vrací sadu metod, které lze volat. První z nich je factory „mojestorage“, které má na starosti ukládání a načítání dat z lokálního úložiště. Druhým factory je „SoapFactory“, které zprostředkovává napojení aplikace na

modul komunikující s webovou službou pomocí SOAP, který je definován v adresáři *lib*.

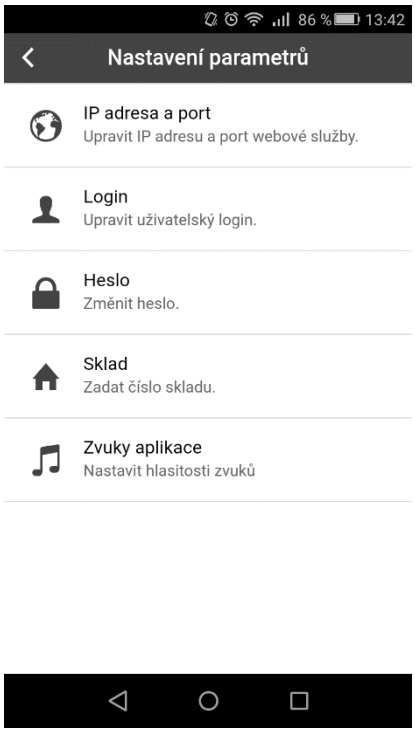
4.4.3 Uživatelské rozhraní aplikace

Uživatelské rozhraní mobilní aplikace „Příprava expedice zakázek“ tvoří 3 základní pohledy. První dva z nich lze vidět na obrázku níže (Obrázek 14). První pohled slouží k výběru data expedice. Toto datum se po výběru uloží do paměti a je tam uloženo, dokud není znovu změněno. To znamená, že obsluha nemusí datum při každém spuštění aplikace znovu zadávat. Při stisku tlačítka Storno se mobilní aplikace ukončí. Po kliknutí na tlačítko OK aplikace přejde na pohled druhý. Ten slouží k samotnému zadávání čísel, které je řešeno skenerem čárových kódů, který je nastaven tak, aby za každým nasnímaným číslem emuloval stisk klávesy pro odřádkování, a tak obsluha nemusí klikat na tlačítko Přidej, protože je to provedeno automaticky. Na druhém pohledu lze také vidět seznam zakázek. Jedná se o seznam zakázek, které jsou plánovány k expedici ke zvolenému datu. Při posunutí zvolené zakázky směrem vlevo se zobrazí 2 tlačítka. První slouží pro aktualizaci hmotnosti kusu, to druhé je pro smazání zakázky ze seznamu zakázek.



Obrázek 14: Základní pohledy aplikace
Zdroj: aplikace „Příprava expedice zakázek“

Třetím pohledem je konfigurace aplikace (Obrázek 15). V něm je možné nakonfigurovat atributy, se kterými aplikace pracuje. Základním nastavením je IP adresa a port webové služby. Na tuto adresu a port se bude aplikace odkazovat při komunikaci s webovou službou OR-SYSTEM Open, protože na této adrese a portu služba naslouchá. Dalšími atributy jsou uživatelské jméno a heslo, které jsou potřebné pro autentizaci při provádění úkonů s aplikací. Tyto údaje jsou uloženy do paměti, takže je nemusí obsluha při každém použití aplikace zadávat. Neméně významným atributem je také číslo skladu, které koresponduje s tím, jak jsou jednotlivé sklady v OR-SYSTEMu Open nadefinovány. Poslední položkou konfigurace je nastavení hlasitosti jednotlivých zvukových hlášení (OK a chyba). Veškerá provedená nastavení v konfiguraci aplikace jsou ukládána do lokálního úložiště mobilního zařízení.



Obrázek 15: Pohled konfigurace aplikace
Zdroj: aplikace „Příprava expedice zakázek“

4.4.4 Webová služba

Autorem webové služby informačního systému OR-SYSTEM Open je Ing. Petr Motl z firmy OR-CZ spol. s r.o., kde působí jako vedoucí systémový programátor a zároveň je konzultantem této bakalářské práce.

Webová služba, se kterou mobilní aplikace Příprava expedice zakázek komunikuje, funguje díky GlassFish serveru, což je open-source aplikační server, na kterém je webová služba spuštěna a s klientem komunikuje pomocí SOAP (viz kap. 4.3.3).

Body element SOAP zprávy, kterou zasílá mobilní aplikace webové službě, obsahuje informace o třídě a metodě, kterou volá. První metodou, která je volána je *getPripraveneRadky()*. Volá se po stisku tlačítka OK na prvním pohledu aplikace (viz Obrázek 14) – v požadavku (requestu) se odesílají atributy *datumExpedice, login, heslo* a *sklad*. Heslo je samozřejmě kvůli bezpečnosti zašifrováno. V odpovědi webové služby pak mobilní aplikace dostává pole objektů v JSON formátu obsahující veškeré důležité informace, které budou v aplikace zobrazeny nebo se s nimi pracuje. Mobilní aplikace zpracuje JSON řetězec zpět do pole objektů a zobrazí je v přehledné tabulce (viz Obrázek 14 - druhý pohled).

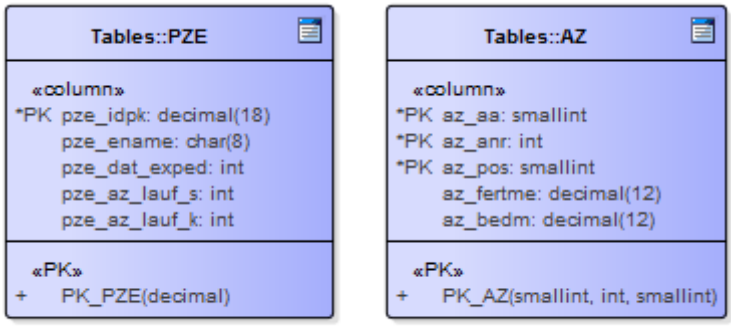
Druhý metoda, která je mobilní aplikací volána, je *zapisPze()*. Je zavolána po naskenování čárového kódu v druhém pohledu aplikace. V požadavku jsou volány stejné atributy jako u předchozí metody s doplňkem atributu *azLauf*, který představuje identifikační číslo zakázky. Webová služba pak prověří všechny podmínky a v závislosti na vyhodnocení podmínek vrátí aktualizovaný seznam zakázek ve formátu JSON (stejně jako u předchozí metody) nebo zprávu s chybovým hlášením, které aplikace zobrazí.

Zvláštním případem je, když webová služba vrátí hlášení, že zakázce je již přiřazeno jiné datum expedice. V tomto případě vrátí speciální typ zprávy, na kterou aplikace zareaguje zobrazením dotazu, zda chce uživatel expedici přeplánovat. Pokud ano, je webové službě odeslána další zpráva, která volá metodu *prepisPze()*. Opět je vrácen aktualizovaný seznam zakázek.

Další metoda webové služby, *zrusPze(),* je volána v případě, že si uživatel přeje zakázku ze seznamu odstranit. Webovou službou je pak mobilní aplikaci vráceno hlášení, zda se požadavek podařilo splnit.

4.4.5 Datový model

Mobilní aplikace „Příprava expedice zakázek“ nemanipuluje s daty přímo. Operace s nimi jsou prováděny pomocí webové služby, která je popsána v předchozí kapitole (viz kap. 4.4.4).



Obrázek 16: Zjednodušený datový model
Zdroj: vlastní

Obrázek 16 zjednodušeně popisuje dvě nejdůležitější tabulky, se kterými webová služba OR-SYSTEM Open pracuje. Model obsahuje pouze nejdůležitější atributy. V tabulce PZE jsou to:

- *pze_idpk* – primární klíč tabulky PZE a ID záznamu,
- *pze_ename* – uživatelský login pořizovatele záznamu,
- *pze_dat_exped* – datum expedice zakázky,
- *pze_az_lauf_s* – ID řádku výrobního příkazu,
- *pze_az_lauf_k* – ID řádku prodejní objednávky (přejímáno z tabulky AZ).

V tabulce AZ jsou to pak:

- *az_aa* – druh zakázky (zákaznická, sériová, výrobní, …) a PK tabulky,
- *az_anr* – číslo zakázky a PK tabulky,
- *az_pos* – číslo řádku zakázky a PK tabulky,
- *az_fertme* – skutečné množství,
- *az_bedm* – plánované množství.

Tabulka AZ slouží k uchovávání dat o všech zakázkách podniku. Tyto zakázky mohou být zákaznické (prodejní), sériové, výrobní aj. Po spuštění mobilní aplikace „Příprava expedice zakázek“ obsluha nastavuje datum expedice. Toto datum je uloženo v lokální paměti zařízení. Když obsluha datového terminálu do aplikace naskenuje čárový kód z průvodního listu zakázky, jedná se o ID výrobního příkazu. Po odeslání dat skrze SOAP, webová služba provede založení nového řádku v tabulce PZE. Atribut *pze_az_lauf_s* je vyplněn právě naskenovaným číslem – tedy ID výrobního příkazu. Atribut *pze_az_lauf_k* je vyhledán a předán z tabulky AZ, kde se nachází řádek příslušné prodejní zakázky (objednávky). Atribut *pze_dat_exped* je vyplněn datem, které bylo zvoleno obsluhou po spuštění a *pze_ename* je vyplněno uživatelským loginem, který je nastaven v konfiguraci mobilní aplikace.

4.5 Zhodnocení přínosu řešení

Aplikace je v době práce na této bakalářské práci ve stádiu testování u zákazníka, takže zatím nefunguje v plném provozu. Aplikace by měla výrazně usnadnit a urychlit proces přípravy expedice zakázek, protože se zaměřuje na nejslabší článek procesu přípravy expedice zakázek, a to na proces tvorby potenciálního seznamu zakázek, kontroly plnění podmínek pro úspěšnou expedici a následnou selekci do finálního seznamu zakázek určených k expedici.

Díky mobilní aplikaci „Příprava expedice zakázek“ odpadá potřeba tvořit papírové seznamy, které putují mezi jednotlivými články procesu přípravy expedice, protože se tyto operace provádí automaticky a v návaznosti na OR-SYSTEM Open, který automaticky zpracovává podmínky a rozhodne, zda je zakázka pro vybrané datum vhodná k expedici. Díky automatizaci se proces výrazně zrychlí a zvýší se tak jeho efektivita.

4.6 Ekonomické zhodnocení

Z ekonomického hlediska je vývoj hybridních aplikací výhodný především v dlouhodobém období, a to jak pro zákaznickou firmu, tak pro řešitele. Pokud se zákaznická firma rozhodne přejít na jinou platformu, není potřeba znovu programovat aplikaci pro tuto nově zvolenou platformu. Díky hybridnímu vývoji jen stačí vygenerovat další instalační balíček. Je to časově i finančně výhodnější.

Navíc v dlouhodobém pohledu se mohou technologie vývoje nativních aplikací měnit s tím, jak se modernizují zařízení a inovují technologie. Je tak možné, že nativní aplikaci by bylo nutno v budoucnu znovu vyvinout v aktuální technologii. Pokud je však využito hybridního vývoje, struktura aplikace zůstává stejná, jen by se změnil způsob kompilace a tvorby nového instalačního balíčku, což však vývojáře mobilní aplikace nemusí zajímat – stačí, když si aktualizuje potřebné nástroje.

Při ekonomickém hodnocení mobilní aplikace „Příprava expedice zakázek“ se náklady skládají především ze dvou složek – hardware a softwarový vývoj. V současné době se cena datového terminálu Chainway C4000 při dokoupení vhodného příslušenství pohybuje kolem 15 tis. Kč [38]. Do nákladů je také nutné připočítat cenu práce na vývoji mobilní aplikace, vývoji modulu a úpravy webové služby pro OR-SYSTEM Open, instalaci zařízení a servis.

S ohledem na obě firmy zde nejsou uvedeny přesné částky zahrnuté do nákladů.

Závěr

Cílem této bakalářské práce bylo seznámit její čtenáře s vývojem mobilních hybridních aplikací a jejich využitím ve firmě OR-CZ spol. s r.o. při vývoji mobilních aplikací spolupracujících s informačním systémem OR-SYSTEM Open.

První část práce se zabývala vývojem mobilních aplikací. Zaměřila se na současné trendy a technologie, které lze k vývoji mobilních aplikací využít. Dotkla se všech tří směrů pro vývoji mobilních aplikací – nativní mobilní aplikace, hybridní mobilní aplikace a mobilní webové stránky. Popsány byly také nejčastěji používané frameworky pro vývoj hybridních aplikací. První část stanoveného cíle této bakalářské práce lze tedy považovat za splněnou – čtenáři neznalému v oboru vývoje mobilních hybridních aplikací by měla tato bakalářská práce přinést alespoň základní náhled do problematiky.

V druhé části se práce zabývala popisem vývoje mobilní aplikace „Příprava expedice zakázek“, jejíž hlavním cílem bylo optimalizovat proces přípravy expedice zakázek v zákaznické firmě. Aplikace byla vytvořena v hybridní technologii Ionic a bude používána na datovém terminálu Chainway C4000 s operačním systémem Android, který má v sobě integrovaný hardwarový skener čárových kódů. Pracovníci tak mohou přestat používat papírové seznamy a zefektivnit tak svou činnost. Díky mobilní aplikace je nyní tato činnost přímo napojena na informační systém OR-SYSTEM Open a díky tomu je i docíleno automatizace procesu kontroly podmínek vhodnosti zakázky k expedici. Byl tedy splněn i druhý stanovený cíl této bakalářské práce – seznámit čtenáře s vývojem konkrétní mobilní aplikace.

Podle autora této bakalářské práce budou v budoucnu tvořit hybridní mobilní aplikace většinovou část trhu s mobilními aplikacemi. Jejich výhody jsou nesporné – jsou multiplatformní, jsou vyvíjeny ve webových technologiích, které nejsou náročné k naučení a jsou nenáročné na údržbu a rozšíření. Díky neustále vzrůstající výkonnosti mobilních zařízení by tak, co se týče výkonu aplikace, mohl pominout rozdíl mezi aplikacemi nativními a hybridními.

Použité zdroje

Citace

[1] CHARLAND, Andre a Brian LEROUX. Mobile application development. *Communications of the ACM* [online]. 2011, **54**(5), 49-53. ISSN 00010782. DOI: 10.1145/1941487.1941504.

[2] WILKEN, Jeremy a Adam BRADLEY. *Ionic in action: hybrid mobile apps with Ionic and AngularJS*. Shelter Island, NY: Manning Publications, 2015. ISBN 9781633430082.

[3] MALAVOLTA, Ivano. Beyond native apps: web technologies to the rescue! (keynote). In: *Proceedings of the 1st International Workshop on Mobile Development – Mobile! 2016* [online]. New York, New York, USA: ACM Press, 2016, s. 1-2. ISBN 9781450346436. DOI: 10.1145/3001854.3001863.

[4] KLUBNIKIN, Andrei. Mobile App Development Trends for 2017 & Beyond. In: *R-Style Lab* [online]. 2016 [cit. 2016-12-18]. Dostupné z: http://r-stylelab.com/company/blog/mobile-technologies/mobile-app-development-trends-for-2017-beyond

[5] GOVE, Jenny. Principles of Mobile App Design. In: *Think with Google* [online]. 2016 [cit. 2016-12-18]. Dostupné z: https://www.thinkwithgoogle.com/articles/principles-of-mobile-app-design-introduction.html

[6] POPA, Marius. Considerations Regarding the Cross-Platform Mobile Application Development Process. *Academy of Economic Studies. Economy Informatics*. 2013, **2013**(1), 41-42. ISSN 15827941.

[7] RAO, Leena. Confirmed: Apple Buys Virtual Personal Assistant Startup Siri. In: *TechCrunch.com* [online]. 2010 [cit. 2016-12-21]. Dostupné z: https://techcrunch.com/2010/04/28/apple-buys-virtual-personal-assistant-startup-siri/

[8] HAN, Jinho. Fingerprint Authentication Schemes for Mobile Devices. *International Journal of Electrical and Computer Engineering*. 2015, **5**(3), 579-585. ISSN: 2088-8708.

[9] ROSENBLATT, Seth. IPhone 5S comes with Touch ID fingerprint scanner. In: *CNET.com* [online]. 2013 [cit. 2016-12-21]. Dostupné z: https://www.cnet.com/news/iphone-5s-comes-with-touch-id-fingerprint-scanner/

[10] PASTORE, Serena. Approaches and methodologies for mobile software engineering. *Advances in Computer Science : an International Journal* [online]. 2015, vol. 4, issue 5, No. 17, 14-22 [cit. 2017-01-13]. ISSN 2322-5157.

[11] World Wide Web Consorcium (W3C). *Mobile Web Best Practices 1.0*. [online]. 2008. [cit. 2017-01-27]. Dostupné z: https://www.w3.org/TR/mobile-bp/#OneWeb

[12] ROY CHOUDHARY, Shauvik, Mukul R. PRASAD a Alessandro ORSO. Cross-platform feature matching for web applications. In: *Proceedings of the 2014 International Symposium on Software Testing and Analysis - ISSTA 2014* [online]. New York, New York, USA: ACM Press, 2014, s. 82-92. ISBN 9781450326452. DOI: 10.1145/2610384.2610409.

[13] XANTHOPOULOS, Spyros a Stelios XINOGALOS. A comparative analysis of cross-platform development approaches for mobile applications. In: *Proceedings of the 6th Balkan Conference in Informatics on - BCI '13* [online]. New York, New York, USA: ACM Press, 2013, s. 213-220. ISBN 9781450318518. DOI: 10.1145/2490257.2490292.

[14] ALI, Mohamed a Ali MESBAH. Mining and characterizing hybrid apps. In: *Proceedings of the International Workshop on App Market Analytics - WAMA 2016* [online]. New York, New York, USA: ACM Press, 2016, s. 50-56. ISBN 9781450343985. DOI: 10.1145/2993259.2993263.

[15] LACKO, Ľuboslav. *Vývoj aplikací pro Android.* 1. Přeložil: M. HERODEK. Brno: Computer Press, 2015. 472 s. ISBN 978-80-251-4347-6.

[16] KHANMOHAMMADI, Kobra, Mohammad R. REJALI a Abdelwahab HAMOU-LHADJ. Understanding the Service Life Cycle of Android Apps: An Eploratory Study. In: *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM '15)* [online]. New York, NY, USA. ACM, 2015, s. 81-86. ISBN: 978-1-4503-3819-6. DOI: 10.1145/2808117.2808123.

[17] VOGEL, Lars. Android application and activity life cycle. *Vogella.com*. [online] [cit. 2017-03-24]. Dostupné z: http://www.vogella.com/tutorials/AndroidLifeCycle/article.html

[18] The App Life Cycle. *App Programming Guide for iOS* [online] [cit. 2017-03-21] Dostupné z: https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

[19] What is XAML? *Microsoft Developer Network.* [online] [cit. 2017-03-19]. Dostupné z: https://msdn.microsoft.com/en-us/library/cc295302.aspx

[20] Intro to the Universal Windows Platform. *Windows Dev Center* [online] [cit. 2017-03-19]. Dostupné z: https://docs.microsoft.com/cs-cz/windows/uwp/get-started/universal-application-platform-guide

[21] Windows 10 universal Windows platform (UWP) app lifecycle. *Windows Dev Center* [online] [cit. 2017-03-19]. Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/launch-resume/app-lifecycle

[22] Dokumentace Mobile Angular UI. *Mobile Angular UI* [online]. [cit. 2016-12-18]. Dostupné z: http://mobileangularui.com/docs/#main-concepts

[23] ARORA, Sunil. 10 Best Hybrid Mobile App UI Frameworks: HTML, CSS and JavaScript. *Noeticforce.com* [online]. [cit. 2017-04-12]. Dostupné z: http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5-js-css

[24] Sencha Touch – Overview. *Sencha.com* [online]. [cit. 2017-04-12]. Dostupné z: https://www.sencha.com/products/touch/#overview

[25] All About Ionic - Company About. *Ionic.io* [online]. [cit. 2016-12-20]. Dostupné z: https://ionic.io/about

[26] DAYLEY, Brad a Brendan DAYLEY. *Sams Teach Yourself AngularJS, JavaScript and JQuery*. 1. vyd. United States of America: Sams Publishing, 2015. ISBN 978-0-672-33742-0.

[27] JavaScript. *MDN – Mozilla Developer Network* [online]. [cit. 2017-04-04]. Dostupné z: https://developer.mozilla.org/cs/docs/Web/JavaScript

[28] Introduction to the DOM – Web APIs. *MDN – Mozilla Developer Network* [online]. [cit. 2017-04-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction#DOM_and_JavaScript

[29] Introduction - JavaScript. *MDN - Mozilla Developer Network* [online]. [cit. 2016-12-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction

[30] MVC Framework - Introduction. *Tutorialspoint.com* [online]. [cit. 2016-12-10]. Dostupné z: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

[31] RESIG, John. Selectors in Javascript. In: *Ejohn.org* [online]. 22.8.2005 [cit. 2017-04-04]. Dostupné z: http://ejohn.org/blog/selectors-in-javascript/

[32] How jQuery Works. *About jQuery* [online]. [cit. 2017-04-04]. Dostupné z: http://learn.jquery.com/about-jquery/how-jquery-works/

[33] Introduction. *V8 – Wiki* [online]. [cit. 2017-04-10]. Dostupné z: https://github.com/v8/v8/wiki/Introduction

[34] About npm. *Npmjs.org* [online]. [cit. 2017-04-10]. Dostupné z: https://www.npmjs.com/about

[35] Introduction. *SOAP Version 1.2 Part 1: Messaging Framework* [online]. [cit. 2017-04-14]. Dostupné z: https://www.w3.org/TR/soap12/#intro

[36] XML Soap. *W3Schools.com* [online]. [cit. 2017-04-21]. Dostupné z: https://www.w3schools.com/xml/xml_soap.asp

[37] Chainway C4000. *Chainway.cz* [online]. [cit. 2017-04-11]. Dostupné z: http://www.chainway.cz/produkty-cz/c4000-cz

[38] Mobilní datový terminál Chainway C4000 / 1D laser. *ITFutuRe.cz* [online]. [cit. 2017-05-02]. Dostupné z: http://obchod.itfuture.cz/chainway-c4000-1d-laser-p2882/

Bibliografie

ARLOW, Jim a Ila NEUSTADT. *UML a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

Cambridge Dictionary [online]. 2017. Dostupné z: http://dictionary.cambridge.org