VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO PODPORU OBCHODNÍCH PROCESŮ NA MOBILNÍCH PLATFORMÁCH

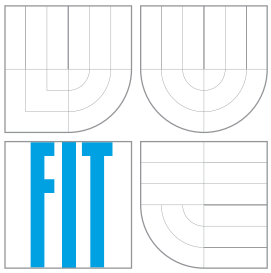
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ROMAN SVOBODA

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO PODPORU OBCHODNÍCH PROCESŮ NA MOBILNÍCH PLATFORMÁCH

TOOL FOR BUSINESS PROCESSES SUPPORT ON MOBILE PLATFORMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ROMAN SVOBODA
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. JITKA KRESLÍKOVÁ, CSc.
SUPERVISOR

BRNO 2014

Abstrakt

Náplní této diplomové práce je návrh nástroje pro podporu obchodních procesů na mobilních platformách. Práce se zabývá problematikou systémů Enterprise Resource Planning, dále synchronizací databází a možnostmi bezpečného přenosu dat mezi mobilním zařízením a serverem. Součástí práce je i seznámení s nejrozšířenějšími mobilními platformami a nástroji pro vývoj multiplatformních aplikací. V poslední části práce se nachází návrh aplikace pro podporu obchodních procesů na mobilních zařízeních, popis její implementace, použitých nástrojů a testování.

Abstract

The scope of this thesis is to design tools to support business processes on mobile platforms. The work deals with both the Enterprise Resource Planning systems issue, as well as the database synchronization and the possibilities of secure data transmission between the mobile device and the server. The work also includes familiarization with the most widely used mobile platforms and tools for multi-platform application development. The last part of the thesis comprises a model of an application meant to support business processes on mobile devices, including a description of its implementation, used tools and testing.

Klíčová slova

Podnikový informační systém, mobilní aplikace, multiplatformní vývoj, Xamarin, plánování podnikových zdrojů, Android, iOS, Windows Phone, MVVM, synchronizace, replikace databází.

Keywords

Company information system, mobile applications, cross-platform development, Xamarin, enterprise resource planning, Android, iOS, Windows Phone, MVVM, synchronization, database replication.

Citace

Roman Svoboda: Nástroj pro podporu obchodních procesů na mobilních platformách, diplomová práce, Brno, FIT VUT v Brně, 2014

Nástroj pro podporu obchodních procesů na mobilních platformách

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. RNDr. Jitky Kreslíkové, CSc. a Ing. Petra Křenka.

.....
Roman Svoboda
28. května 2014

Poděkování

Rád bych poděkoval své vedoucí práce doc. RNDr. Jitce Kreslíkové, CSc. za trpělivost i cenné rady. Děkuji také Ing. Petru Křenkovi za umožnění této práce.

Obsah

Kapitola 1

Úvod

Předmětem této diplomové práce je mobilní aplikace pro podporu obchodních procesů podnikového informačního systému typu Enterprise Resource Planning (dále jen ERP), která je vyvíjena s využitím nástrojů pro vývoj na více mobilních platforem současně pomocí jednoho zdrojového kódu sdíleného pro všechny platformy.

Kapitola 2 se zabývá jak obecným popisem ERP systémů, jejich možnostmi, přínosy a nevýhodami jejich využití, tak vybranými druhy konkrétních ERP systémů, oblastmi jejich využití a moduly, z nichž jsou sestaveny.

V kapitole 3 naleznete analýzu IT infrastruktury reálné společnosti a specifikace požadavků na nástroj pro podporu obchodních procesů na mobilních platformách.

O problematice synchronizace databází mezi různými klienty a centrální databází se dočtete v kapitole 4. Kapitola se zabývá i možnostmi přenosu dat mezi mobilní aplikací a centrálním úložištěm dat a jejich bezpečností.

Kapitola 5 obsahuje návrh řešení problému s distribucí, aktualizací a další údržbou jak aplikace, tak firemních mobilních zařízení pomocí systémů vzdálené správy.

Analýzou nejrozšířenějších mobilních platforem současnosti se zabývá kapitola 6. Naleznete v ní informace o architektuře platforem Android, iOS a Windows Phone i s přehlednými diagramy.

Kapitola 7 čtenáře nejdříve seznámí s obecným popisem druhů nástrojů pro vývoj multiplatformních aplikací, a poté se zabývá několika konkrétními vybranými významnými nástroji pro vývoj multiplatformních aplikací.

Návrhem konkrétního systému aplikace, jež je předmětem této diplomové práce, se zabývá kapitola 8. Dále se v ní seznámíte s architekturou návrhového vzoru MVVM (Model-View-ViewModel), jež je zvolen pro tuto aplikaci.

Popis použitých implemenentačních nástrojů a vlastní implementace se nachází v kapitole 9. Informace o způsobu testování obsahuje kapitola 10. Závěr, zhodnocení práce, získaných poznatků a zamyšlení nad možným pokračováním naleznete v kapitole 11.

Tato diplomová práce navazuje na semestrální projekt Nástroj pro podporu obchodních procesů na mobilních platformách. Ze semestrálního projektu jsou do této diplomové práce převzaty a případně rozšířeny kapitoly 2, 3, 6, 7 a 8.

1.1 Motivace

Motivací pro tuto diplomovou práci je snaha dozvědět se více o vývoji aplikací pro mobilní zařízení, možnostech usnadnění a urychlení jejich tvorby s využitím nástrojů pro multiplat-

formní vývoj, které dle mého názoru mají značný potenciál do budoucna pro tvorbu nejen mobilních aplikací.

K bližšímu seznámení s touto problematikou mi dopomohla možnost spolupráce s internetovou agenturou Q2 Interactive s.r.o. na vývoji mobilního klienta BYZNYS Mobile pro informační systém BYZNYS ERP v moderním multiplatformním vývojovém prostředí Xamarin od samého počátku vývoje.

Kapitola 2

Seznámení s Enterprise Resource Planning systémy

Enterprise Resource Planning (dále jen ERP) je podnikový informační systém, který integruje, automatizuje a zaznamenává množství interních podnikových procesů, jež souvisí s jeho činností. Obvykle je sestaven nebo doplněn o více různých integrovaných modulů zvolených či vytvořených dle potřeb konkrétního podniku. Umožňuje okamžitý („real-time“) přístup a správu dat uložených ve sdílené databázi uživatelům z různých útvarů podniku, jenž obvykle využívají jen některé z mnoha modulů systému, čímž předchází redundanci dat a snižuje prodlevy při komunikaci[26].

Například prodejce s „real-time“ přístupem ke zboží na skladě může okamžitě potvrdit dostupnost žádaného zboží. Jakmile prodejce uloží objednávku do systému, data z objednávky jsou ihned dostupná oddělení výroby či skladu, takže je možné okamžitě upravit výrobní plány nebo připravit expedici objednávky. ERP systémy jsou v dnešní době využívány jak velkými společnostmi, tak malými podniky, neboť v sobě integrují množství organizačních prostředků a umožňují bezchybné transakce a procesy.

2.1 Modely dodání a provozu ERP systémů

Lze rozlišit dva základní modely dodání a provozu ERP systému – „On premise“ a „On demand“[38].

2.1.1 „On premise“ model dodání a provozu

Základní charakteristikou tohoto modelu je instalace ERP systému na hardwaru klientské organizace a následná správa systému klientem s případnou spoluprácí dodavatele. „On premise“ ERP systémy jsou obvykle dodávány zprostředkující firmou, která systém případně upraví v rámci možností systému dle přání a potřeb klientské organizace. Jeho výhodou je možnost provést o mnoho větší přizpůsobení systému potřebám klientské organizace. Další výhodou je provoz na hardwaru organizace, a tudíž přístup přes lokální síť díky níž lze předpokládat lepší dostupnost a vyšší zabezpečení dat před zneužitím zvnějšku organizace.

Model „on premise“ byl původním a nejběžnějším modelem pro provoz ERP systémů, avšak postupně začíná ustupovat před modernějším „on demand“ modelem. Je však na zvážení každé organizace, jaké má finanční a technologické možnosti, a jaký model preferuje z hlediska bezpečnosti a dalších rizik.

2.1.2 „On demand“ model dodání a provozu

Též známý jako SaaS (Software as a Service) nebo ASP (Application Service Providing). Vyznačuje se „outsourcingem“ provozu ERP systému do „cloudu“ mimo organizaci a jeho dodáváním jako služby na základě smlouvy známé jako Service Level Agreement (SLA).

Hlavní výhody a důvody pro využití „on demand“ ERP jsou především celkově nižší náklady na zavedení, provoz, údržbu a aktualizace pro klientskou organizaci. O všechny tyto náležitosti se stará dodavatel ERP systému pro mnoho klientů současně, a díky tomu dosahuje mnohem lepšího využití všech zdrojů a nižší ceny poskytované služby. Mezi další nesporné výhody náleží možnost flexibilního zvyšování/snižování provozních prostředků v závislosti na růstu organizace, rychlost implementace systému, snadné predikování finančních nákladů na systém a dostupnost systému odkudkoli (za předpokladu připojení k internetu)[37].

Nevýhodou pak naopak může být z jistého pohledu vyšší riziko nedostupnosti či ohrožení dat. Toto riziko se však kvalitní poskytovatelé „on demand“ ERP řešení snaží minimalizovat a nezřídka díky své specializaci na poskytování a zabezpečení této služby pro mnoho klientských organizací zajistí oblast dostupnosti a bezpečnosti o mnoho lépe a levněji, než v problematice méně zkušený IT personál klientské organizace využívající „on premise“ systém[35].

Další nevýhodou je nižší možnost přizpůsobení „on demand“ ERP systému přímo na míru potřebám společnosti, což se dodavatelé snaží řešit větší škálou možných konfigurací systému, ale ani ty nemohou být nekonečné. A v poslední řadě, pokud je systém dostupný pouze po internetu, je tradičním rizikem i výpadek připojení k internetu[25].

„On demand“ model je v posledních letech stále oblíbenější zvláště u malých a středních organizací, které si nemohou dovolit velké investice do vlastního IT sektoru.

2.2 Stručná historie ERP systémů

Na vznik a vývoj ERP systémů, jak je známe v současnosti, měly nejzásadnější vliv následující společnosti – SAP AG (Systemanalyse und Programmentwicklung), International Business Machines Corporation (dále jen IBM), J.D. Edwards World Solution Company (dále jen JDE.), The Baan Corporation, PeopleSoft, Inc. a Oracle Corporation[26].

Prvopočátky vzniku ERP systémů lze vysledovat do 60. let minulého století, kdy velké výrobní společnosti využívaly tzv. Inventory Control Packages (IC) běžící na sálových počítačích pro monitorování jejich skladových zásob, bilancí a tvorbu zpráv o stavu.

V 70. letech vznikly systémy Material Requirements Planning (MRP), což byly nástroje pro plánování skladových zásob materiálu, požadavků a plánování výrobních procesů výrobních společnosti běžící na sálových počítačích. Taktéž došlo k vzniku společností JDE, Oracle Corporation, dále založení SAP AG a uvolnění jejich systémů SAP R/1 a SAP R/2. Světlo světa v tomto období spatřil i systém COPICS (Communications Oriented Production Information and Control System) od IBM.

V průběhu 80. let byly významným krokem ve vývoji středně velké („midrange“) počítače od IBM, jako například IBM System/38, které jsou přístupnější pro využití od malých a středních společností. Pro tyto počítače vznikly systémy typu MRP II (Manufacturing Resources Planning), které rozšiřují MRP o další funkce, jako plánování a řízení výrobních zdrojů, účtování a lidské zdroje. V tomto období také vznikla společnost PeopleSoft, která se soustředí na vývoj systémů pro správu lidských zdrojů, a její první systém Human Resource Management System (HRMS).

V 90. letech byl poprvé společností Gartner Group použit pojem Enterprise Resource Planning. Významným milníkem byl také roku 1992 ERP systém SAP R/3, který se vyznačoval využitím klient-server architektury, díky čemuž mohl běžet na různých operačních systémech a platformách. Současně se jeho návrh vyznačoval otevřenou architekturou, což umožňovalo společnostem třetích stran vyvíjet software, který by se integroval se SAP R/3.

Od roku 2000 pak došlo k definování Extended ERP, které přidružuje i systémy CRM (Customer Relationship Management) a SCM (Supply Chain Management)[22].

2.3 Základní části ERP systému

Typické moduly ERP systému mohou být následující:

- řízení prodeje, distribuce a logistiky,
- správa skladů,
- plánování výroby,
- kontrola kvality,
- údržba výrobních prostředků,
- management aktiv,
- řízení výkonu podniku, (Business Intelligence – BI),
- lidské zdroje (Human Resources Management – HRM),
- řízení vztahů se zákazníky (Customer Relationship Management – CRM),
- řízení vztahů s dodavateli (Supplier Relationship Management – SRM),
- řízení dodavatelského řetězce (Supply Chain Management – SCM),
- projektový systém,
- finanční management, fakturace a účetnictví.

Není však nezbytné, aby v každém ERP systému byly zahrnuty všechny tyto moduly. Mnohdy společnost vyžaduje více specifické řešení ERP systému, nebo naopak některé části vůbec nevyužije, například plánování výroby v případě obchodní společnosti.

2.3.1 Přínosy využití ERP systému

Nejzákladnější výhodou ERP systému je beze sporu fakt, že díky integraci velkého množství procesů do jednoho systému šetří čas a náklady. Navíc umožňuje dělat rychlejší rozhodnutí s méně chybami.

Mezi další možnosti a výhody využití ERP patří například:

- predikce prodeje,
- chronologické řazení transakcí,
- sledování objednávek od přijetí po naplnění,

- sledování příjmů od faktury po přijetí platby,
- řeší nutnost synchronizovat změny mezi několika systémy,
- snadnější delegování pravomocí a kontrola práce,
- odstranění redundantních dat a předcházení chybám v datech,
- vyšší bezpečnost díky integraci více systémů do jednoho,
- automatizované zpracovávání dat znamená větší efektivitu a menší časovou náročnost,
- statistické a analytické nástroje poskytují vedení společnosti okamžité podklady pro rozhodování,
- standardizace procesů,
- okamžité informace pro zákazníky, obchodní partnery i pracovníky[49].

2.3.2 Nevýhody a omezení ERP systémů

Jedním z nedostatků ERP systémů může být i přes veškerou snahu jejich malá modifikovatelnost, která nutí společnost přizpůsobovat své interní postupy ERP systému, což může být někdy značně obtížné.

Další nevýhodou je i nutnost proškolit na nový ERP systém zaměstnance, jež jej budou využívat. To v případě přechodu z jiných systémů může znamenat značnou dobu, než se zaběhnou nové postupy. Ve velkých organizacích může zavedení ERP systému trvat i několik let.

Samotné odlišné potřeby každé společnosti, kterým bývá ERP přizpůsobován, vytvářejí další nevýhodu, z níž plyne náročnější údržba a aktualizace.

V neposlední řadě bývá vnímán jako nedostatek i jejich hierarchická rigidita, centralizovaná kontrola a management. ERP systémy předpokládají, že informace by měly být spravovány centrálně a společnosti mají dobře definované hierarchické struktury, což však nemusí vždy platit.

Co se týče faktu, že ERP jsou systémy založené na relačních databázích, zde může vyvstat problém v případě, že databáze nebo její části jsou distribuovány na více strojích a je vyžadována synchronizace mezi nimi. V případě ERP aplikací pro mobilní zařízení nastává občasná nedostupnost připojení, a tím i nutnost odložené synchronizace offline změn dat.

Nakonec i fakt, že množství informací je ve společnosti ve formě nestrukturovaných dat, které se obtížně zpracovávají a nejsou příliš vhodné pro uchování v relačních databázích. Tudíž ERP systémy by se měly více soustředit i na efektivní zpracování nestrukturovaných dat a informace z nich pak spojovat s reálnými procesy[16][21].

2.4 Seznámení se systémem BYZNYS ERP

BYZNYS ERP je systém od české společnosti J.K.R. spol. s r.o, pro niž je určena i mobilní aplikace, jenž je předmětem této diplomové práce. Jedná se o B2B a B2C řešení s možností propojení s Microsoft Office a množstvím analytických nástrojů.

Systém je sestaven z více různých modulů, které pokrývají následující oblasti řízení:

- finanční oblast (bankovní operace, fakturace, finanční účetnictví, pokladna),
- evidence (evidence majetku, mzdy a personalistika, skladové hospodářství),
- zákazníci (CRM, iBYZNYS),
- inteligence (BYZNYS BI, Excellent, OLAP),
- produktivita (doprava, výroba),
- efektivita (prvky uživatelské volnosti, gadgety),
- procesy (projektové řízení, správa projektů, workflow),
- dokumenty (Document Management System DMS, BYZNYS Office),
- informace (BYZNYS Mobile, Informace, Manažer, Zakázky).

Systém je dodáván v několika variantách oborových řešení:

- univerzální řešení pro všechny typy organizací,
- účetní firmy,
- stavebnictví,
- výrobní firmy,
- obchodní společnosti,
- služby a zakázky,
- neziskové a příspěvkové organizace,
- dopravní společnosti[24].

Kapitola 3

Analýza IT infrastruktury reálné společnosti a specifikace požadavků

V této kapitole se podíváme na reálnou společnost, její IT infrastrukturu, a požadavky na nástroj pro podporu obchodních procesů. Naší modelovou společností bude firma J.K.R. spol. s r.o., jenž si u mého zaměstnavatele zadala zakázku na výrobu mobilního klienta pro jejich informační systém BYZNYS ERP.

3.1 IT infrastruktura reálné společnosti

Společnost J.K.R. se zabývá vývojem a dodáváním systému BYZNYS ERP různě velkým podnikům s využitím modelu „on premise“. Klientský podnik vlastní centrální databázový server (či více serverů), na němž běží systém BYZNYS ERP, a množství různých klientských pracovních stanic a mobilních zařízení vybavených aplikací pro připojení k systému, a samozřejmě i webovým rozhraním nebo aplikací pro přístup do systému pro externí klienty či partnery. Každý klient, partner, zaměstnanec či manažer firmy má přístup do různých modulů a částí systému a má udělena odlišná oprávnění provádět různé operace v systému.

3.2 Specifikace požadavků na nástroj

Požadavkem pro firmu Q2 Interactive bylo naprogramovat multiplatformního mobilního klienta pro informační systém BYZNYS ERP, který by sloužil obchodníkům klientských společností pro zobrazování, tvorbu a správu databáze objednávek, faktur, klientů, schůzek, a dalších základních funkcí systému BYZNYS ERP, a to i v případě nedostupnosti připojení k internetu.

Požadavky na funkčnost mobilního klienta jsou následující:

- Jádro aplikace – správa uživatelů, nastavení, synchronizace.
- Adresář (zákazníci) – seznam zákazníků (s vyhledáváním, tříděním, zobrazení seznamu), detail zákazníka obsahující adresy, osoby s možnost editace, historie objednávek zákazníka s rozpadem na doklad a položky, detail objednávky, historie vydané fakturace zákazníka s rozpadem na doklad, zobrazení informací o celkové dlužné částce, platební morálka, detail vydané faktury (pouze zobrazení), možnost zobrazení přílohy (pdf souboru).

- Produkty – seznam produktů zobrazený jako tabulka, mřížka nebo po jednom výrobku, detail produktu se zobrazením posledních 3 odběrů (datum, počet) z objednávek pro vybraného zákazníka, ceny k produktům dle vybraného zákazníka (vázané ceny a slevy), přidání produktu do objednávky/košíku.
- Objednávky – seznam objednávek, nová objednávka, úprava objednávky, zrušení objednávky.
- Aktivity – sety aktivit, které musí obchodník u zákazníka udělat (upomínky).

Kapitola 4

Možnosti synchronizace databází a bezpečnost přenosu dat

Součástí této diplomové práce je i zvážení možností synchronizace (replikace) databází. Proto se v této kapitole podíváme obecně na různé způsoby synchronizace dvou či více databází označované též jako replikace. Dále kapitola obsahuje analýzu možností přenosu dat a jejich bezpečnosti.

4.1 Důvody pro synchronizaci

Pro provádění synchronizace lze najít mnoho různých důvodů, z nichž některé jsou zmíněny níže.

- Zvýšení výkonu systému – množství operací lze rozdělit mezi různé databáze.
- Dočasná nedostupnost hlavní databáze – zejména u mobilních zařízení bez neustálého připojení k internetu bývá mnohdy vhodné databázi, nebo její část, uložit do lokálního databázového úložiště.
- Distribuce zátěže – např. využití jedné repliky databáze nebo její části pro složitější výpočty a druhé repliky pro jednoduché operace.
- Zabezpečení proti selhání systému – synchronizace z důvodu dostupnosti konzistentních úplných replik hlavní databáze, které okamžitě převezmou její roli v případě výpadku hlavní databáze.
- Dosažení místního zpracování – snížení vzdálenosti mezi daty a jejich uživatelem, díky čemuž data nemusí po síti cestovat tak daleko. Tím dojde i ke snížení zátěže sítě.

4.2 Dělení synchronizací obecně

Synchronizaci databází můžeme dělit z několika hledisek[44]:

- Dle modelu na:
  - jednosměrnou (též single-master, master-slave nebo pesimistická replikace),

  - obousměrnou (též multi-master, master-master, optimistická replikace, peer-to-peer, update-anywhere).

- Dle zpoždění propagace dat na synchronní („eager“) a asynchronní („lazy“).
- Dle způsobu replikace na fyzickou a logickou.

4.2.1 Dle modelu – jednosměrná replikace

Označovaná též pojmy single-master, master-slave nebo pesimistická synchronizace[13][19]. Tento model se vyznačuje jedinou hlavní (označována též angl. master) databází, v níž je uživatelům a aplikacím povoleno provádět čtení a aktualizace dat, a libovolného počtu replik master databáze (označovány anglicky slave), z nichž je uživatelům a aplikacím umožněno data pouze číst.

Změny provedené v master databázi jsou propagovány do slave databází synchronně nebo asynchronně. V případě synchronní propagace jsou změny do všech slave databází zapsány okamžitě v rámci jedné atomické transakce. Díky tomu synchronní propagace změn umožňuje udržovat všechny slave databáze vždy v konzistentním stavu s master databází, avšak snižuje rychlost provedení trasakce a zvyšuje odezvu master databáze. Více o synchronní replikaci naleznete v kapitole 4.2.3.

Při asynchronní propagaci jsou změny master databáze ukládány do transakčního žurnálu („log“) a do slave databází zapisovány se zpožděním společně s dalšími změnami provedenými v master databázi od poslední propagace změn do slave databází. Asynchronní propagace změn master databáze se vyznačuje její vyšší rychlostí provádění změn a rychlejší odezvou, avšak za cenu dočasné nekonzistence slave databází s master. Doba zpoždění propagace dat z master do slave databází pak záleží na konkrétních požadavcích na aktuálnost dat slave databází a nastavení systému. Více o synchronní replikaci naleznete v kapitole 4.2.4.

Díky svým vlastnostem je tento model vhodný zejména v případech, kdy čtecí operace výrazně převyšuje aktualizace. Hlavní výhodou modelu jednosměrné replikace je absence konfliktů způsobených aktualizací dat v různých databázích. V případě výpadku master databáze je pak možné zvolit za nový master jednu ze slave databází, obvykle tu s nejaktuálnějším stavem databáze a vhodnými technickými parametry[42].

4.2.2 Dle modelu – obousměrná replikace

Označovaná též pojmy multi-master, master-master, peer-to-peer, update-anywhere, optimistická synchronizace[13][19]. V případě obousměrného modelu replikace jsou si všechny databáze rovnocenné. To znamená, že čtení i aktualizace lze provádět ve všech databázích současně, neboť všechny jsou master.

Změny provedené v jedné dababázi mohou být do ostatních databází opět propagovány synchronně nebo asynchronně. V případě synchronní replikace jsou změny v jedné databázi do všech ostatních databází zapsány okamžitě v rámci jedné atomické transakce. Nedochází díky tomu tedy ke konfliktům, ale zvyšuje se odezva a snižuje rychlost operace. Více o synchronní replikaci naleznete v kapitole 4.2.3.

V případě asynchronní propagace naopak systém dosahuje lepší odezvy a vyšší rychlosti, avšak za cenu nutnosti dočasné nekonzistence s ostatními databázemi a nutnosti řešit konflikty, což je podrobněji popsáno v kapitole 4.2.4.

Tento model je vhodný zejména pokud potřebujeme provádět aktualizace na více databázích systému[42].

4.2.3 Dle zpoždění propagace dat – synchronní

Známá též jako „eager“ (včasná) replikace. Synchronní replikace spočívá v okamžité propagaci změn v jedné databázi do ostatních replik v rámci jedné atomické transakce. Synchronní replikace se implementuje dvěma způsoby.

V prvním případě je nastavena spoušť („trigger“), která při provedení aktualizace v primární databázi spustí tuto aktualizaci jako distribuovanou transakci ve všech replikách. Tato transakce uspěje pouze pokud uspěly transakce ve všech replikách.

Druhý způsob se nazývá dvoufázový potvrzovací protokol a spočívá v provedení jedné distribuované transakce paralelně na všech replikách. Při tomto způsobu se využívá mechanismu synchronizace procesů, který zajistí, že transakce na všech replikách proběhnou ve stejném pořadí.

Obecně však platí, že pro potvrzení distribuované transakce je třeba dílčí potvrzení transakce od všech databází, jinak je distribuovaná transakce považována za neúspěšnou[42].

Výhody:

- Předchází problémům s konflikty při synchronizaci, neboť všechny změny jsou promítnuty okamžitě v rámci transakce do všech databází.
- Zachovává vždy vzájemně konzistentní stav všech databází.

Nevýhody:

- Není možná při nedostupném spojení s ostatními databázemi.
- Vyšší časová prodleva pro dokončení transakce, neboť vyžaduje potvrzení transakce od všech databází, jež se synchronizují.
- Vyžaduje rychlou síť a vysokou dostupnost všech databází, což znamená vyšší náklady.

4.2.4 Dle zpoždění propagace dat – asynchronní

Známá též jako „lazy“ (líná) replikace. Asynchronní replikace, jak už název napovídá, provádí synchronizace databází nezávisle na prováděných transakcích. Při asynchronní replikaci dochází k ukládání úspěšných transakcí provedených v databázi do transakčního žurnálu („log“) a jejich souhrnné propagaci do dalších databází s určitým časovým zpožděním nebo na žádost uživatele, vzdáleného serveru, aplikace apod. Transakční žurnál obsahuje záznamy o provedených transakcích, případně může uchovávat i záznamy o provedených binární změnách bytů databáze v případě fyzické replikace[42].

Výhody:

- Je vhodná především v případech, kdy je potřeba rychlá odezva databáze bez nutnosti čekání na potvrzení distribuované transakce při synchronní replikaci.
- Nezbytná je pak v případech, kdy není neustále dostupné spojení s ostatními databázemi, jako tomu je u mobilních zařízení.

Nevýhody:

- V případě užití obousměrného modelu s asynchronní replikací nutnost řešit konflikty synchronizace.
- Z důvodu zpoždění synchronizace je nevýhodou i dočasná vzájemná nekonzistence databází.
- Pokud selže při obousměrném modelu zařízení s master databází, jež má neodeslané změny, celý systém je pak nekonzistentní vzhledem k těmto změnám.

4.2.5 Dle způsobu replikace – logická

Spočívá v replikaci změn master databáze pomocí aplikace logických operací. V případě jazyka SQL (Structured Query Language) se jedná o DML (Data Manipulation Language), případně i DDL (Data Definition Language), ale samozřejmě může databáze využívat i jiný jazyk nebo způsob záznamu operací.

Logická replikace se implementuje pomocí žurnálování („logování“) SQL dotazů, využitím databázových spouští („triggerů“) nebo pomocí agenta zachycujícího SQL dotazy[14].

Výhody:

- Umožňuje replikovat i pouhé části databáze, například jen některé tabulky nebo záznamy.
- Je nezávislá na použitém hardwaru, operačním systému, SŘBD[1] i verzi SŘBD. Čili umožňuje i replikaci mezi různými verzemi i typy databáze (např. z MySQL do MSSQL).
- Umožňuje obousměrnou replikaci.

Nevýhody:

- Logická replikace je náročnější na režii i nastavení.
- Nezaručuje konzistenci s replikovanou databází.
- V případě změn stejných záznamů v různých master databázích vyžaduje řešení těchto konfliktů ručně nebo pomocí předem připravených postupů, které obvykle nejsou triviální.

4.2.6 Dle způsobu replikace – fyzická

Fyzická replikace spočívá v tvorbě přesné binární kopie replikované databáze obvykle za účelem vytváření zálohy. Jde o operaci prováděnou na nejnižší úrovni SŘBD, která spočívá v žurnálování („logování“) binárních změn („diff“) bloků dat master databáze po jednotlivých bytech. Tyto změny jsou pak okamžitě nebo se zpožděním aplikovány na slave databázi[14].

[1] SŘBD – Systém řízení báze dat

Výhody:

- Jednodušší nastavení a údržba.
- Nižší režie a vyšší rychlost než logická replikace.
- Zaručená 100% konzistence s master databází.
- Větší rozšířenost.

Nevýhody:

- Je možné replikovat jen kompletní databáze, nikoli jejich části (např. jen určité tabulky).
- Umožňuje replikace výhradně mezi stejnými verzemi SŘBD na stejných operačních systémech.
- Master a slave databáze musejí mít stejné množství diskového prostoru, ideálně pak i stejný hardware, zejména architekturu procesoru.
- Slave databáze je přístupná nejvýše pro čtení („read-only“), není možné spouštět jiné dotazy.
- Možné využít jen pro jednosměrný model replikace, neboť neumožňuje řešit konflikty vzniklé při obousměrném modelu.

4.3 Metody přenosu dat a jejich zabezpečení

Vzhledem k tomu, že předmětem této diplomové práce je mobilní aplikace, je nezbytné zvážit i možnosti přenosu dat z hlavního úložiště (databázový server) do mobilního zařízení a zabezpečení přenášených dat.

U mobilních zařízení jsou dostupné pouze omezené možnosti pro získávání dat ze vzdálené databáze. První možnost, která některé nejspíše napadne, by bylo přímé připojení z aplikace ke vzdálené databázi. Záhy však zjistíme, že většina, ne-li všechny mobilní platformy, tuto možnost nativně nepodporují. Obvyklým způsobem pro získávání dat ze vzdálených databází jsou totiž u mobilních zařízení tzv. webové služby (web services).

Webová služba je metoda komunikace po síti mezi dvěma elektronickými zařízeními. Jde o neustále běžící softwarovou službu poskytovanou pod nějakou internetovou adresou[50].

Nejčastěji využívané specifikace pro komunikaci s webovou službou jsou SOAP a REST (REpresentational State Transfer), proto se na tyto specifikace, jejich výhody, nevýhody a především možnosti zabezpečení podíváme blíže.

4.3.1 Simple Object Access Protocol – SOAP

Jedná se o protokol pro komunikaci s webovými službami založený na formátu XML[2], který se soustředí na přístup k operacím, které obvykle reprezentují nějakou aplikační logiku. Přenos SOAP nejčastěji zajišťují protokoly aplikační vrstvy HTTP[3] a SMTP[4]. Typická

[2] XML – Extensible Markup Language
[3] HTTP – Hypertext Transfer Protocol
[4] SMTP – Simple Mail Transfer Protocol

komunikace pomocí SOAP protokolu se skládá ze zprávy zaslané webové službě jejím konzumentem (klientem) a odpovědi od webové služby na tuto zprávu zaslané zpět klientovi. Tělo SOAP zprávy může obsahovat požadavek na provedení nějaké akce, například zaslání dat, ale i samotná data s požadavkem na jejich uložení. Odpověď pak může v těle SOAP zprávy obsahovat požadovaná data, nebo informaci o úspěchu či neúspěchu akce[43].

Bezpečnost

SOAP protokol má definováno vlastní bezpečnostní rozšíření WS-Security, které zajišťuje zabezpečení na úrovni samotné zprávy. To znamená, že i když zpráva opustí transportní vrstvu, tak je stále zabezpečena a může ji rozšifrovat a přečíst jen její právoplatný příjemce.

Mechanismy, jimiž WS-Security umožňuje zabezpečit zprávu, jsou následující:

Autentizace – přiložením bezpečnostního tokenu pro ověření totožnosti odesílatele (Kerberos tikety, X.509 certifikáty, hesla).

Podepsání – podepsání zprávy zajišťuje její integritu, což znamená, že zpráva nebyla zfalšována.

Šifrování – zašifrování zprávy zabezpečí, že si ji nepřečte někdo neoprávněný[30].

Dalším rozšířením SOAP je WS-ReliableMessaging, které implementuje mechanismy potvrzení spolehlivého doručení SOAP zprávy od jejího odesílatele k příjemci.

Mimo tato bezpečnostní opatření SOAP podporuje samozřejmě i zabezpečení na úrovni přenosu zprávy pomocí SSL/TLS (Secure Sockets Layer/Transport Layer Security)[43].

Výhody

- Množství standardů pro podporu zabezpečení, spolehlivého přenosu, transakcí.
- Přesně definovaný protokol, možnost přesně specifikovat webovou službu pomocí WSDL[5].
- Podpora více různých transportních protokolů (SMTP, HTTP).
- Podporuje mechanismy zabezpečení i na úrovni zprávy, nejen na úrovni jejího přenosu.

Nevýhody

- Pro mnoho aplikací příliš složitý.
- Pomalý.
- Spousta nadbytečné režie.
- Odpovědi na zprávy nelze uložit ve vyrovnávací paměti.
- Formát zprávy pouze jako XML.
- Vyžaduje spoustu podpůrného softwaru (např. Windows Communication Foundation)[33].

[5] WSDL – Web Services Description Language

4.3.2 Representational State Transfer – REST

Hlavní rozdíl REST od SOAP je ten, že to není protokol, nýbrž návrhový vzor pro síťovou architekturu typu klient-server (konzument webové služby – webová služba). Přirozeně se používá společně s protokem HTTP[6]/HTTPS[7], tudíž je bezstavový, avšak je možné tuto architekturu využít i pro jiné protokoly. Dále budeme uvažovat pouze REST užitý společně s HTTP/HTTPS, takzvaný „RESTful“.

„RESTful“ definuje jednotné rozhraní pro přístup ke zdrojům, z nichž každý má jednoznačný identifikátor (URI – Unified Resource Identificator). Dále definuje 4 základní HTTP metody - GET, POST, PUT, DELETE pro práci se zdroji. Klient zasílá serveru požadavky pomocí URI zdroje a HTTP metody a server je vykonává a odpovídá. Nejčastější formát pro výměnu dat je JSON[8], avšak může využívat i XML[9] a jiné[9].

Bezpečnost

Veškerou bezpečnost, kterou REST umožňuje, získává od protokolu HTTP, či spíše jeho zabezpečené varianty HTTPS. Jedná se tedy o zabezpečení na úrovni přenosu zprávy s využitím SSL/TLS.

SSL/TSL poskytuje následující možnosti zabezpečení:

Autentizaci – ověření totožnosti odesílatele zprávy přiložením bezpečnostního tokenu. Obvykle svou totožnost prokazuje pouze server.

Šifrování – ochrání zprávy před odposloucháváním pomocí asymetrické šifry.

Zabezpečení na úrovni zprávy, pakliže nějaké vyžadujeme, musí být do webové služby i klienta implementováno dodatečně. Architektura REST ani protokol HTTPS k tomu neposkytuje žádné nástroje.

Výhody

- Množství různých formátů pro přenos zpráv – JSON, XML a další.
- Nezávislost REST architektury na protokolu.
- Jednotně definované rozhraní.
- Snadno škálovatelný.
- Jednoduchý a snadno implementovatelný.
- Méně nadbytečné režie zpráv, rychlejší než SOAP.
- Zdroje je možno uložit ve vyrovnávací paměti.

[6] HTTP – Hypertext Transfer Protocol
[7] HTTPS – Hypertext Transfer Protocol Secure
[8] JSON – JavaScript Object Notation
[9] XML – Extensible Markup Language

Nevýhody

- Bezstavovost při užití HTTP/HTTPS.
- Absence formální specifikace.
- Zabezpečení jen díky použitému protokolu[33].

Z výše uvedených možností pro implementaci webové služby bylo nakonec zvoleno užití REST architektury z důvodu její snadné implementovatelnosti, nižší přenosové režie zpráv a dostatečného zabezpečení pro potřeby aplikace, jelikož nebyly definovány žádné zvlášní bezpečnostní požadavky na integritu dat.

4.4 Případ aplikace – asynchronnní logická obousměrná

Synchronizace databází je znázorněná na diagramu 4.1 a bude probíhat mezi celkem třemi databázemi – databáze mobilního klienta, databáze synchronizačního Q2 serveru a produkční databáze systému BYZNYS ERP.

BYZNYS server je hlavní server, na který se připojují všechny aplikace, ať už jde o správce, zaměstnance, partnera, apod. Tyto servery se fyzicky nacházejí v sídle společnosti, jelikož jde o „on premise“ řešení.

Q2 server je mezistupeň, který bude obsahovat kopii dat na BYZNYS serveru a budou na něm probíhat synchronizace mezi klientskými mobilními aplikacemi a BYZNYS serverem. Fyzické umístění Q2 serveru bude na hostingu s garancí maximální dostupnosti a rychlosti připojení.

Databáze mobilní klientské aplikace si uchovává data potřebná pro svou offline práci, tedy veškeré faktury, objednávky, produkty, nabídky a schůzky pro dané partnery od určitého data, s nimiž bude obchodník (uživatel mobilní aplikace) pracovat při nedostupném připojení. V případě dostupného připojení si aplikace na žádost uživatele stahuje nová data dle zvolené operace nebo partnera, která ukládá do lokální databáze a poté pracuje s lokální instancí těchto dat. Tato databáze je omezena velikostí paměťové karty mobilního zařízení, avšak tato paměťová karta by měla být vždy zvolena v dostatečné velikosti, aby nedošlo k nedostatku místa.

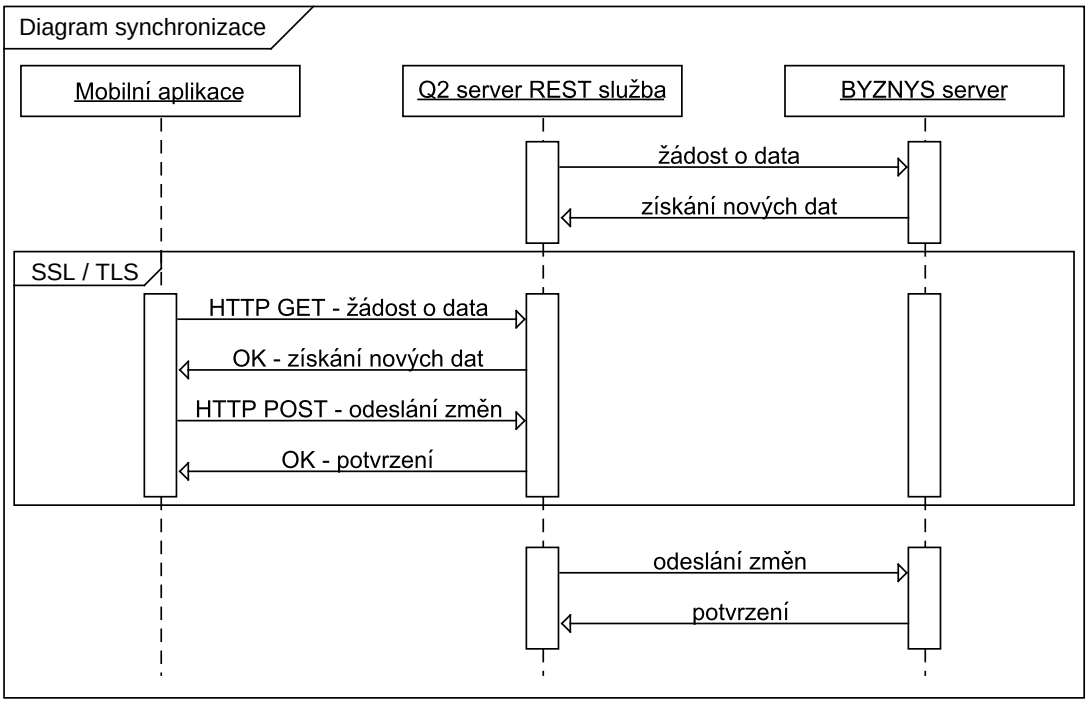
V případě vytvoření nové objednávky je nejdříve ověřena dostupnost připojení, a pokud je dostupné, je objednávka odeslána Q2 serveru, který ji dále zpracuje a předá BYZNYS serveru pro další zpracování. V případě nedostupnosti připojení aplikace uloží novou objednávku do databáze a odešle ji ke zpracování Q2 serveru automaticky po získání připojení.

Důvodem pro toto řešení je zajištění maximální dostupnosti dat pro mobilní klienty, neboť BYZNYS server nemusí mít vždy zajištěnu dostupnost.

V rámci synchronizace z BYZNYS serveru směrem do klienta budou dostupné dva rozsahy pro stažení dat – kompletní synchronizace a rozdílová synchronizace. Kompletní synchronizace, jak již název napovídá, provede stažení všech dat, což je pro hardware a rychlost připojení mobilního zařízení často zdlouhavá záležitost. Z tohoto důvodu bude k dispozici i rozdílová synchronizace, která stáhne pouze aktualizace za určité časové období, typicky od data posledního stažení.

4.4.1 Synchronizace BYZNYS server – Q2 server

Synchronizace mezi produkčním serverem a Q2 serverem probíhá periodicky po určitých časových intervalech. Jelikož jde o asynchronní logickou synchronizaci, je důležitým prvkem



Obrázek 4.1: Diagram synchronizace databází serverů a mobilních klientů [inspirováno interními dokumenty Q2 Interactive].

žurnál („log“) změn, který je v našem případě implementován pomocí tabulky obsahující informace o všech změnách od poslední synchronizace. Log změn je přítomen jak na Q2 serveru, tak na BYZNYS serveru a pro rozdílovou synchronizaci je nezbytný. Při kompletní synchronizaci se naopak log změn nevyužije, neboť tato synchronizace se využívá pouze pro počáteční naplnění vyprázdněné databáze Q2 serveru všemi dostupnými daty v tabulkách BYZNYS serveru.

Důvodem pro toto řešení je fakt, že Q2 a BYZNYS server využívají rozdílné typy databází a Q2 server má proti BYZNYS serveru i upravenou strukturu tabulek, například přidány sloupce pro příznaky typu změny záznamu nebo časové razítko poslední aktualizace záznamu. Není tedy možné použít synchronní ani fyzickou replikaci. Obousměrnost synchronizace je dána nutností provádět aktualizace BYZNYS databáze z mobilního klienta, například vytvořením objednávky a jejím odesláním do systému.

Rozdílový import změn z BYZNYS serveru na Q2 server

Na BYZNYS serveru se do tabulky změn ukládají informace o změnách záznamů všech tabulek určených k synchronizaci na Q2 server jako trojice: jméno tabulky, primární klíč změněného záznamu, typ změny (nový, upraven, smazán). Synchronizace z BYZNYS serveru na Q2 server (z pohledu Q2 serveru jde o import) pak probíhá tak, že Q2 server si nejdříve přečte všechny záznamy v tabulce změn na BYZNYS serveru. Poté si na základě informací z tabulky změn stáhne všechny změněné záznamy z patřičných tabulek BYZNYS serveru, vymaže je z tabulky změn na BYZNYS serveru a podle typu jejich příznaku změny aktualizuje patřičné záznamy v databázi Q2 serveru.

Nové záznamy přidá a jejich sloupec s příznakem změny nastaví na hodnotu pro nový záznam. Upravené záznamy aktualizuje dle záznamů z BYZNYS serveru a označí je příznakem změny pro upravení. Smazaným záznamům nastaví příznak změny pro smazání. U nových, upravených i smazaných záznamů na Q2 serveru je současně upraveno časové razítko jejich poslední aktualizace na aktuální čas. Časové razítko i příznak typu změny u každého záznamu v tabulkách Q2 serveru se využívá při rozdílové synchronizaci z Q2 serveru do mobilního klienta.

Rozdílový export změn z Q2 serveru na BYZNYS server

Při exportu z Q2 serveru na BYZNYS server se opět využívá tabulka se záznamem změn, která obsahuje všechny změny získané od klientů. Záznam v tabulce změn na Q2 serveru obsahuje:

- příznak, zda již byl záznam/změna exportována na BYZNYS server,
- jednoznačný identifikátor uživatele, který data vytvořil,
- jednoznačný identifikátor objektu, pakliže se jedná o úpravu nebo smazání,
- typ objektu,
- data serializovaná do formátu JSON[10],
- příznak typu změny objektu (nový, upravený, smazaný).

Během exportu jsou pak nahrány na BYZNYS server všechny záznamy o změnách, které nejsou označeny příznakem exportu, a následně jsou tímto příznakem označeny. BYZNYS server si je poté již zpracuje dále.

4.4.2 Synchronizace klient – Q2 server

Synchronizace mezi mobilním klientem a Q2 serverem probíhá pouze při dostupném připojení na vyžádání uživatele. Z tohoto samotného faktu tedy vyplývá, že je také asynchronní. Dále je logická, neboť uživatel by měl mít možnost stáhnout si pouze určitá data, která jej zajímají, např. faktury nebo objednávky. Obousměrnost synchronizace je opět dána nutností aktualizovat v mobilním klientovi data a nahrávat je prostřednictvím Q2 serveru zpět na BYZNYS server.

Klient má dostupné dva rozsahy synchronizace – kompletní a rozdílovou. Kompletní synchronizace provede vymazání lokální databáze a nové stažení veškerých dat z Q2 serveru. Slouží především pro prvotní naplnění klienta daty. Rozdílová synchronizace se využívá pro stažení pouze změn provedených v databázi od poslední synchronizace klienta s Q2 serverem. Důležitá je zejména pro snížení množství přenášených dat přes mobilní připojení a samozřejmě proběhne i rychleji než kompletní díky menšímu objemu přenášených dat.

Přenos dat během synchronizace probíhá pomocí protokolu HTTPS[11] s využitím REST webové služby Q2 serveru. Data jsou odesílána i získávána ve formátu JSON.

[10] JSON – JavaScript Object Notation
[11] HTTPS – Hypertext Transfer Protocol Secure

Rozdílový import změn z Q2 serveru do klienta

Klient vyžádá od Q2 serveru všechny záznamy od data posledního importu, které má uloženo lokálně, a provede zpracování obdržených dat na základě příznaku o typu změny u každého záznamu. Nové záznamy přidá do lokální databáze, upravené záznamy aktualizuje dle stažených dat, záznamy označené jako smazané vymaže z lokální databáze.

Rozdílový export změn z klienta na Q2 server

K exportu změn z klienta se opět využívá tabulka změn provedených od poslední synchronizace, která má stejný formát jako tabulka změn na Q2 serveru. Při jakékoli změně provedené v mobilním klientovi se informace o této změně uloží do lokální tabulky změn. Na žádost uživatele se při dostupném připojení provede nahrání dat v lokální tabulky změn do tabulky změn na Q2 serveru.

Kapitola 5

Distribuce a aktualizace aplikace, správa mobilních zařízení

Jelikož je pro mnohé společnosti užívající systém BYZNYS ERP důležité, aby jejich obchodníci měli na svých mobilních zařízeních vždy aktuální verzi mobilní aplikace pro systém BYZNYS ERP, je třeba vyřešit i aktualizace takovým způsobem, aby obchodník nebyl nucen provádět je sám.

K tomuto účelu lze využít rozličné systémy od různých výrobců pro vzdálenou správu mobilního zařízení (Mobile Device Management, dále jen MDM), které IT technikům firmy umožní kompletní vzdálenou správu mobilního zařízení nebo uzavřeného aplikačního prostředí. Dostupné jsou systémy nabízející jak „on demand“, tak „on premise“ řešení vzdálené správy[41]. Důvodů pro zavedení systémů pro vzdálenou správu zařízení je několik:

- Efektivnější práce díky možnosti hromadné i jednotlivé vzdálené správy, instalace, nastavení a aktualizace zařízení a aplikací. Například i znemožněním instalovat na zařízení hry.
- Zvýšení zabezpečení díky jednotnému profesionálnímu nastavení, možnosti vymazat nebo deaktivovat přístup při ztrátě zařízení, možnosti omezit využívání nedůvěryhodných aplikací a zdrojů pomocí černé listiny (blacklist).
- Znemožnit zaměstnanci měnit nastavení mobilního zařízení nebo manipulovat s nepovolenými funkcemi dle potřeb společnosti.
- Monitorovat činnost zaměstnance a způsob užívání zařízení.
- Zabezpečení dat na zařízení a komunikaci s organizací pomocí šifrování, autentizace a dalších bezpečnostních technik.
- Komunikace se zaměstnancem pomocí zpráv, notifikací[41].

K tématu vzdálené správy zařízení se vztahuje i pojem Bring Your Own Device (BYOD), což je využití soukromého zařízení zaměstnance pro účely vzdálené správy. Obvykle se však možnosti správy limitují na nějaký vyhrazený zabezpečný pracovní prostor společnosti, který zaměstnanec využívá pro práci, komunikaci s organizací, uchovávání dat, atd.

Z konkrétních MDM řešení můžeme zmínit třeba XenMobile, AirWatch, AppBlade, Appaloosa, které všechny podporují více platforem. Z nativních řešení je k dispozici například MDM od Apple pro iPad a iPhone, které bohužel není podporované pro Českou Republiku, nebo Windows Intune[10][18].

Kapitola 6

Možné cílové platformy a jejich architektura

V této části rozebereme nejdůležitější mobilní platformy Android, iOS a Windows Phone, jejich architekturu jádra a typické komponenty aplikací.

6.1 Android

Systém Android původem od společnosti Google je momentálně nejrozšířenější systém na trhu mobilních zařízení. Jedná se o open source operační systém, jehož jádro je od verze Android 4.0 postaveno na jádře Linuxu verze 3.4.10. Starší verze Android využívaly jádro založené na jádře Linuxu 2.6.x. v současnosti je nejnovější verzí Android 4.4, který nese označení KitKat. Android je naprogramován v jazycích C, C++ a pro vyšší úrovně systému a pro vývoj nativních aplikací se využívá Android SDK[1], Java a XML pro definici uživatelského rozhraní[20][47].

6.1.1 Architektura systému Android

Hlavní části architektury operačního systému Android znázorněné na obrázku 6.1 jsou následující:

Aplikace (Applications) Nejvyšší vrstva architektury obsahuje různé předinstalované aplikace, sms zprávy, kontakty, fotoaparát, prohlížeč atd., které ke svému běhu využívají aplikační framework.

Aplikační framework (Application framework) Tato vrstva obsahuje knihovny Java API[2], které umožňují aplikacím využívat služby knihoven v jazyce C/C++ z vrstvy knihoven. V tomto frameworku jsou vyvíjeny i předinstalované aplikace, tudíž je možné je případně nahradit i aplikacemi třetích stran. Nejdůležitější API zahrnují funkce telefonování, zdroje, lokace, UI content providers (dodavatelé obsahu uživatelského rozhraní – dat), správce balíčků, atd.

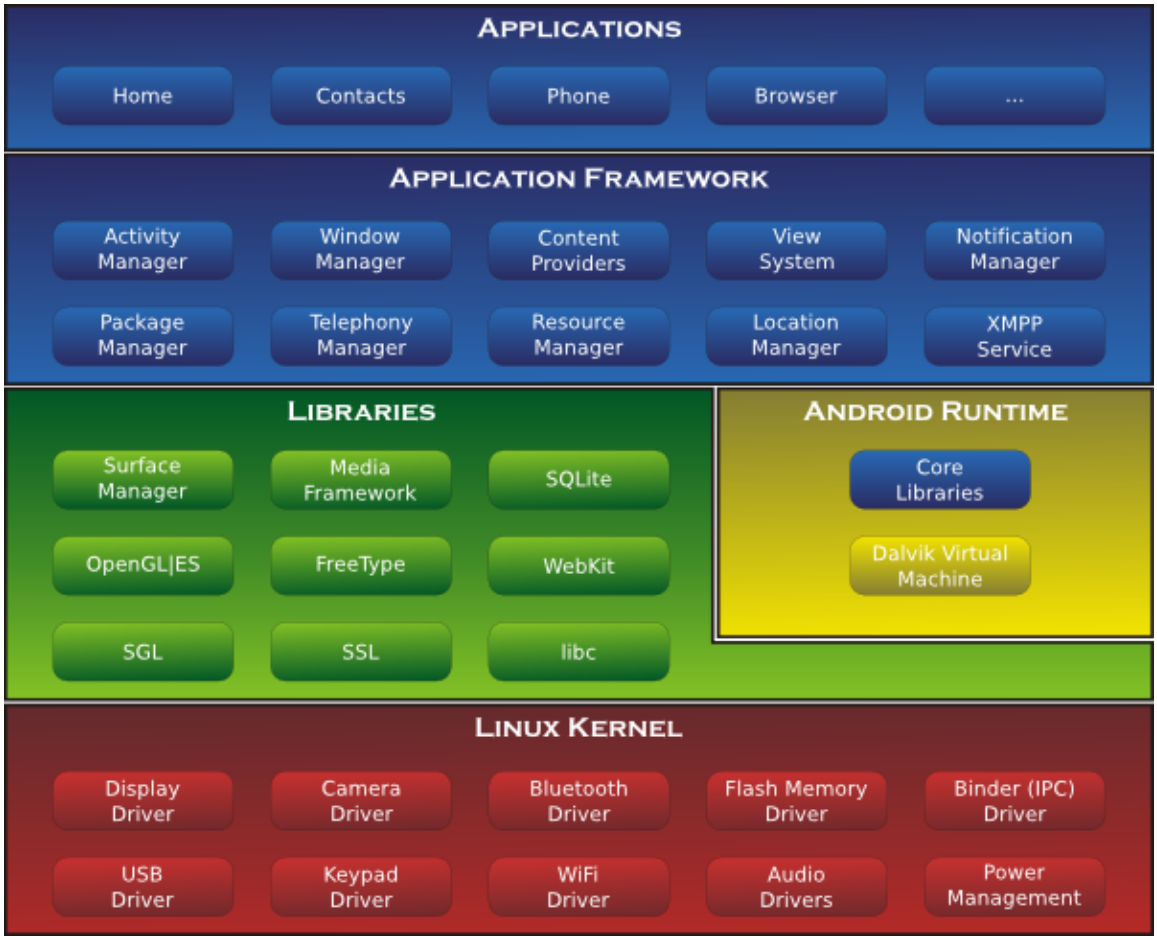
Běhové prostředí Android (Android runtime) Obsahuje knihovny jádra a Dalvik VM (virtual machine), který je optimalizován k běhu několika instancí virtuálních strojů.

[1] SDK – Software Development Kit
[2] API – Application Programming Interface

Když Java aplikace přistupují ke knihovnám jádra, každé je přidělena její vlastní instance virtuálního stroje, což zajišťuje izolaci od ostatních aplikací.

Knihovny (Libraries) Na předposlední úrovni je množství C/C++ knihoven, jako například OpenGL, WebKit, FreeType, SSL, libc, SQLite a Media framework. Tyto knihovny jsou programátorovi přístupné přes aplikační framework.

Linuxové jádro (Linux kernel) Na nejnižší úrovni je linuxové jádro zodpovědné za ovladače zařízení, přístup ke zdrojům, řízení napájení a ostatní povinnosti operačního systému. Standardní ovladače zahrnují displej, kameru, klávesnici, WiFi, flash paměť, audio, IPC (Inter Process Communication). Ačkoliv jádro je Linux, množství aplikací je napsáno v Javě a spouštěno skrze virtuální stroj Dalvik.



Obrázek 6.1: Diagram architektury operačního systému Android [převzato z [47]].

6.2 iOS

Druhým nejrozšířenějším systémem je operační systém iOS od společnosti Apple, Inc. Jádrem operačního systému iOS je open source operační systém Darwin, jenž slouží jako jádro i pro operační systémy OS X a náleží do rodiny systémů Unix. V současnosti je nejvyšší

verzí iOS 7. iOS je naprogramován v jazycích C, C++ a Objective-C, který je využíván především pro vyšší úrovně systému.

Objective-C je i jazykem pro programování nativních aplikací v prostředí Xcode a uživatelské rozhraní je obvykle definováno pomocí XML souboru generovaném v Xcode[45].

6.2.1 Architektura systému iOS

Hlavní vrstvy architektury operačního systému iOS, znázorněné na obrázku 6.2, jsou následující:

Aplikace (Application) Nejvyšší vrstva architektury obsahuje různé předinstalované aplikace, sms zprávy, kontakty, fotoaparát, prohlížeč atd,

Cocoa Touch Vrstva Cocoa Touch obsahuje klíčové frameworky pro tvorbu iOS aplikací. Nabízejí základní infrastrukturu aplikace a podporu pro technologie, jako je multitasking, dotykový vstup, rozpoznávače gest, rozvržení obrazovky (layout), notifikace a další vysokoúrovňové systémové služby. Náleží sem frameworky pro práci s adresářem (Address Book), událostmi (Event Kit), hrami (Game Kit), reklamami v aplikacích (iAd), mapami (Map Kit), zprávami a emaily (Message UI), Twitterem a rozličnými funkcemi uživatelského rozhraní (UIKit).

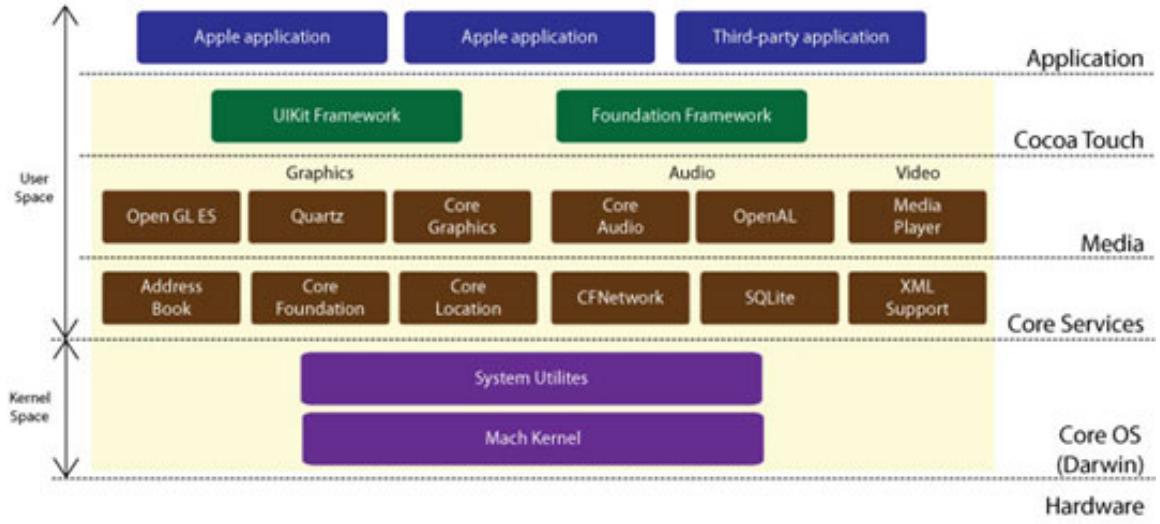
Media Vrstva Media obsahuje technologie pro práci s grafikou, zvukem a videem, např. OpenGL, OpenAL,

Služby jádra (Core Services) Obsahují základní systémové služby pro aplikace. Klíčové mezi těmito službami jsou Core Foundation a Foundation framework. které definují základní datové typy užívané všemi aplikacemi. Vrstva dále obsahuje technologie pro podporu vlastností jako lokalizace, iCloud, sociální média, SQLite, XML, vytváření sítí a další.

Jádro OS (Core OS) Na nejnižší úrovni je vrstva jádra, která krom samotného Darwin jádra obsahuje i nízkoúrovňové technologie, jako například Core Bluetooth pro práci s bluetooth, Accelerate pro práci s lineární algebrou, zpracováním obrazu, dále Security pro práci se zabezpečením a External Accessory pro práci s externím hardwarem. Samotné Darwin jádro je zodpovědné za ovladače zařízení, přístup ke zdrojům, řízení napájení, sockety, souborový systém, alokaci paměti a další povinnosti operačního systému.

6.3 Windows Phone 8

Třetím nejrozšířenějším na trhu je v současné době operační systém Windows Phone od Microsoftu (též zvaný Windows Store). Systém Windows Phone je poměrně novým nástupcem systému Windows Mobile a představil zcela nové uživatelské rozhraní známé jako Metro přizpůsobené moderním standardům dotykových obrazovek a mobilních zařízení. V současnosti je nejnovější verzí Windows Phone 8.1, který je jako první Windows pro mobilní telefony založen na upraveném jádru Windows NT, na němž běží i varianta Windows 8.1 pro stolní počítače.



Obrázek 6.2: Diagram architektury operačního systému iOS [převzato z [12]].

Windows (Phone) 8 představil i zcela nový vývojový model aplikací, který umožňuje programovat aplikace hned v několika jazycích dle preference programátora – C#, C++, VisualBasic, dokonce i JavaScript, případně tyto jazyky kombinovat. Vývojovým prostředím pro vývoj Windows 8 aplikací je dobře známé Visual Studio[32].

Tyto jazyky přitom využívají jednotné API společné pro všechny jazyky, přičemž veškeré uživatelské rozhraní aplikací psaných v C++, C# nebo VisualBasic je definováno pomocí jednotného značkovacího jazyka XAML. Využití JavaScriptu pak navíc přidává možnost uživatelské rozhraní Windows 8 aplikací definovat pomocí HTML5 a CSS3

6.3.1 Architektura systému Windows Phone 8.1

Architektura systému Windows Phone je znázorněna na obrázku 6.3. Můžeme poznamenat, že kromě možnosti vývoje v několika různých programovacích jazycích je i velice dobře přizpůsobena vývoji na více různých druhů zařízení (desktop, mobilní zařízení, konzole) s nutností minimálních změn kódu, ale také oplývá dobrou přenositelností aplikací na jiné mobilní platformy (iOS, Android). Architektura je blíže popsána níže.

Aplikace (Metro style apps) Nejvyšší vrstvu architektury zaujímají různé zabudované aplikace, ale i aplikace třetích stran.

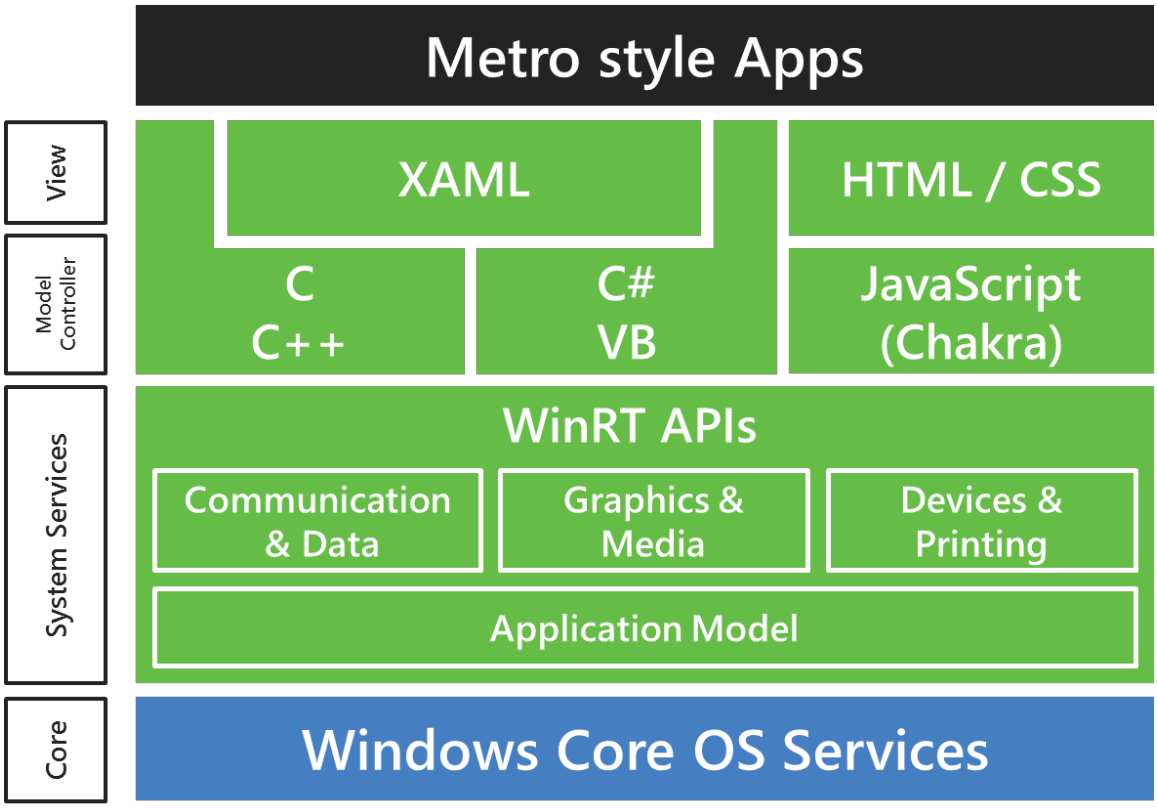
UI vrstva (View) Vrstva UI (User Interface) je v architektuře sytému Windows 8 zastoupena pomocí jazyků XAML[3] a HTML5[4]/CSS3[5], které slouží pro definici jednotného uživatelského rozhraní pro všechny možné jazyky nižších vrstev.

Model, Controller Na této vrstvě architektury je již samotný funkční kód aplikace psaný v jazycích C++, C#, VisualBasic nebo JavaScript, který využívá sdíleného jednotného API nižší úrovně.

[3]XAML – Extensible Application Markup Language
[4]HTML – HyperText Markup Language verze 5
[5]CSS3 – Cascading Style Sheets 3

Systémové služby (System services) Na této vrstvě nalezneme jednotné API – Windows Runtime – pro práci se systémovými službami jádra, které zahrnují např. grafiku, multimédia, tiskárny, externí zařízení, datové služby a komunikace.

Jádro (Core) Jádrem systému je upravené jádro systému Windows NT, které je společné všem druhům Windows 8 pro různá zařízení, a veškeré jeho služby jsou přístupné pomocí jednotného API Windows Runtime[32].



Obrázek 6.3: Diagram architektury systému Windows Phone 8 [převzato z [39]].

Kapitola 7

Multiplatformní vývojové nástroje a implementační prostředí

V této kapitole se budeme zabývat nástroji pro vývoj multiplatformních aplikací a jejich vlastnostmi. Na některé se podíváme i blíže a zvolíme ten nejvhodnější pro naši aplikaci. V současné době je na trhu velké množství různých frameworků a nástrojů pro vývoj multiplatformních aplikací, avšak mnoho z nich platí za svou univerzálnost nějakou daň v podobně horší optimalizace ať už uživatelského rozhraní, nebo funkčnosti.

7.1 Druhy multiplatformních nástrojů

Ve výstupu multiplatformních aplikací si můžeme všimnout dvou základních trendů – webové aplikace a nativní aplikace. Webové aplikace využívají interpretace webových technologií pro svou činnost, kdežto nativní aplikace jsou překládány do nativního kódu, případně nějakého univerzálního mezikódu. Webové aplikace jsou vhodné spíše pro aplikace malého rozsahu, neboť dosahují nižší míry optimalizace.

Z hlediska způsobu vývoje můžeme multiplatformní vývojové nástroje rozdělit na několik druhů (znázorněno na obrázku 7.1):

Webové nástroje (Web App Toolkits) Využívají pro vývoj aplikací různé webové technologie. Jejich výhodou je široká základna vývojářů obeznámená s těmito technologiemi a rychlý vývoj. Nevýhodou je z pravidla malá komplexnost a nativnost výstupní aplikace. Běží buď online v internetovém prohlížeči, nebo se tváří jako běžné aplikace a spouští se v offline prohlížeči. Mezi tyto nástroje patří např. JQuery Mobile, Kendo UI, Sencha Touch a další.

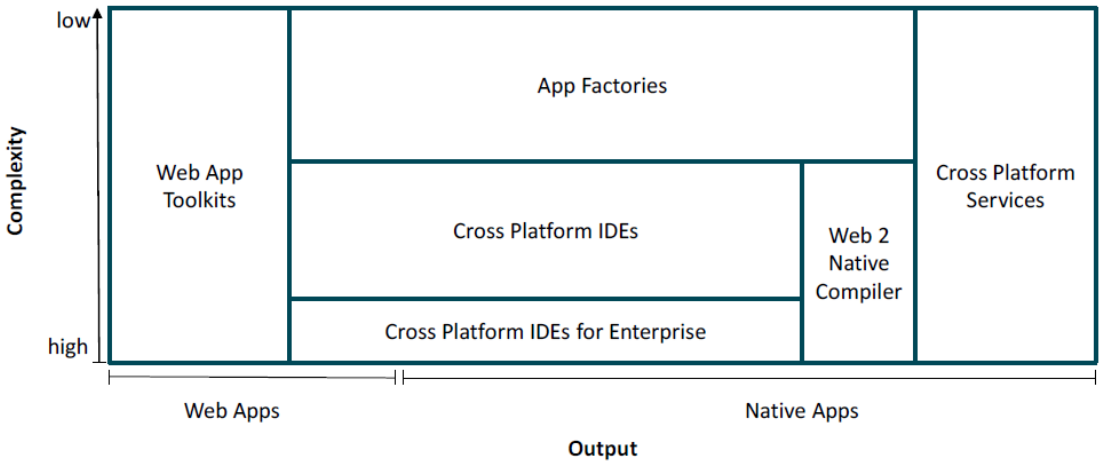
Továrny na aplikace (App Factories) Jsou nástroje umožňující vývoj aplikací i neprogramátorům na základě grafického uživatelského rozhraní. Představiteli jsou např. AppMachine, Mobile Roadie.

Multiplatformní vývojová prostředí (Cross platform Integrated Development Environment) Většina z nich se soustředí na vývoj nativních aplikací, ale najdou se i takové, které vytvářejí webové aplikace. Jejich cílem je napsat jeden kód, který se pak zkompiluje pro více různých platforem. Některé se soustředí na pokrytí co největšího množství platforem výměnou za horší kvalitu aplikace, jiné naopak volí cestu méně platforem a vyšší kvality aplikace. Příkladem jsou Xamarin, Marmalade, Titanium, PhoneGap, Monocross, Adobe Air.

Multiplatformní vývojová prostředí pro společnosti (Cross platform Integrated Development Environment for enterprises) Soustředí se na potřeby společností poskytováním většího množství API pro obvyklá ERP, CRM a e-shop systémy. Příkladem jsou Corona Enterprise, Rhomobile, Appscend, Any Presence, Service2Media.

Multiplatformní kompilátory (Cross platform compiler) Soustředí se na překládání jednoho kódu do nativních aplikací. Překlenují nativní API cílových zařízení do zvoleného programovacího jazyka pro zápis jednotného zdrojového kódu. Mnohá IDE je využívají místo tvorby svých vlastních řešení.

Multiplatformní služby (Cross platform services) Jsou „cloudové“ služby, které umožňují snadnou integraci různých funkcí do multiplatformních aplikací[36].



Obrázek 7.1: Diagram současného rozložení nástrojů pro multiplatformní vývoj [převzato z [36]].

Vhodnost multiplatformního vývojového nástroje lze hodnotit z několika hledisek:

- komplexnost nástroje,
- rychlost a optimalizace,
- nativnost výstupní aplikace
- podporované platformy, operační systémy,
- dostupnost podpory,
- množství dostupných hardwarových funkcí a API,
- množství dostupných předinstalovaných aplikací,
- cena a rychlost vývoje.

7.2 Adobe Air

Adobe Air je známé běhové prostředí pro aplikace psané v HTML[1]/JavaScript/CSS[2]/AJAX[3] nebo Adobe Flash/Adobe Flex/ActionScript. Je dostupné pro platformy Android, BlackBerry, iOS, OS X, Windows, ale i pro televize a další zařízení. Nenabízí však podporu pro Windows Phone. Aplikace Adobe Air jsou typu RIA (Rich Internet Application), což znamená, že nejsou přeloženy do nativního kódu zařízení, ale pro jejich běh je třeba mít nainstalováno běhové prostředí Adobe Air[8].

7.3 PhoneGap

PhoneGap je jedním z nejznámějších nástrojů využívajících pro vývoj webové technologie HTML5, CSS3, JavaScript a libovolné webové frameworky. Jedná se o open source nástroj s vlastním IDE[4], které při překladu obaluje kód webové aplikace dodatečným PhoneGap kódem a aplikace se poté tváří jako nativní. Nabízí plnou nebo alespoň částečnou podporou iOS, Android, Blackberry OS, WebOS, Windows Phone, Symbian a Bada. Avšak právě z důvodu neúplné podpory pro některá zařízení je pro nás nevhodný[34].

7.4 Embarcadero RAD Studio 5

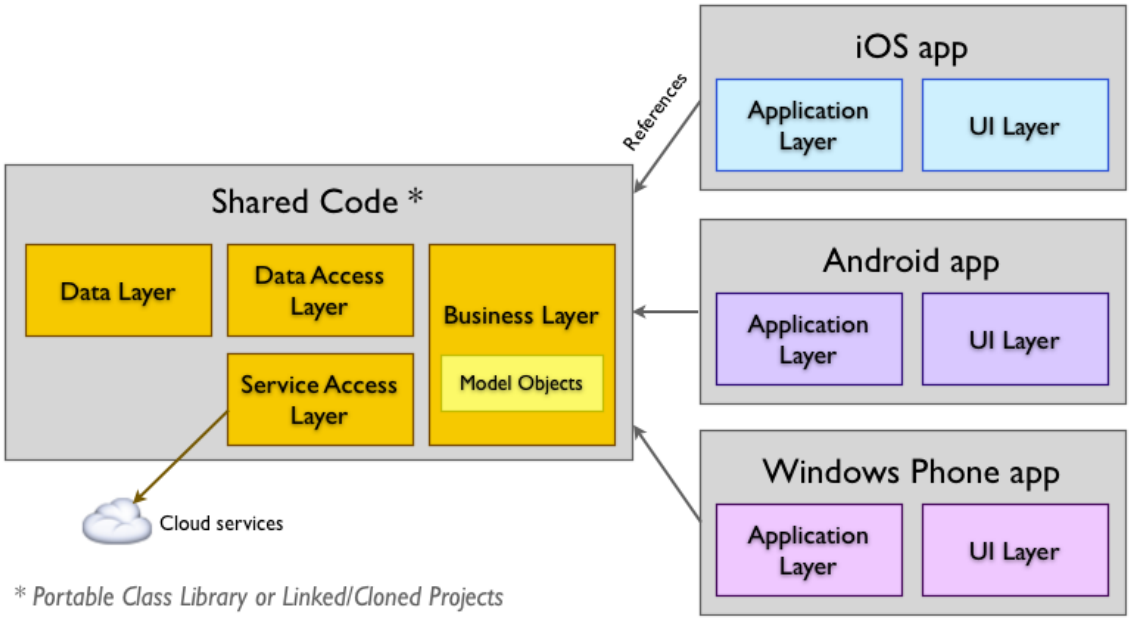
RAD Studio je komerční multiplatformní vývojové prostředí, které kompiluje aplikace do nativního kódu platformy. To přináší téměř 100% možnost sdílení kódu, avšak současně znamená jisté potíže při ladění uživatelského rozhraní na různé platformy, které díky sdílení kódu nikdy nebude dostatečně dokonalé, zvláště u složitějších UI. RAD Studio nabízí možnost programovat v C++ nebo Delphi a jeho cílové platformy jsou iOS, Android a Windows Phone, avšak Windows Phone podporuje jen pro C++[17]. Z těchto důvodů taktéž není vhodným nástrojem.

7.5 Xamarin

Xamarin je komerční platforma pro multiplatformní vývoj nejen mobilních aplikací nabízející za tímto účelem sadu integrovaných nástrojů. Platforma Xamarin nabízí:

- Mono .NET framework poskytující multiplatformní implementaci .NET a umožňující přístup k platformově specifickým nástrojům (Software Development Kit – SDK) a programovacím rozhraním (Application Programming Interface – API) pro vývoj nativního softwaru pro Windows, Android, iOS i Mac pomocí C# kódu.
- Kompilátory specifické pro každou z podporovaných platforem. v případě iOS překládají do nativního jazyka symbolických adres architektury ARM. V případě Android do mezikódu (Intermediate Language - IL), který se vykonává na aplikačním virtuálním stroji MonoVM s podporou Just-In-Time překladu pro urychlení běhu. Pro Windows se překlad provádí také do IL, který je poté spouštěn v zabudovaném běhovém prostředí (Common Language Runtime – CLR) též s podporou Just-In-Time kompilace.

[1] HTML – HyperText Markup Language
[2] CSS – Cascading Style Sheets
[3] AJAX – Asynchronous JavaScript and XML
[4] IDE – Integrated Development Environment



Obrázek 7.2: Diagram architektury multiplatformní aplikace v Xamarin [převzato z [52]].

- Samostatné vývojové prostředí (IDE) Xamarin Studio pro Mac i Windows a zásuvný modul (plugin) pro Microsoft Visual Studio, které umožňují vývoj nativních aplikací pro iOS, Android, Windows a Mac v jazyce C#.

Xamarin oplývá rozsáhlou komunitou vývojářů, jelikož pro abstrakci jednotlivých platforem vychází z frameworku Mono, který je multiplatformní open source implementací .NET frameworku od Microsoftu. Implementace Xamarin pro jednotlivé platformy vycházejí z jejich implementací od Mono. Jsou to:

- Xamarin.Android vycházející z Mono for Android,
- Xamarin.iOS, dříve známý jako MonoTouch,
- Xamarin.Mac, jež je variantou MonoMac[53].

Xamarin nabízí efektivní sdílení cca 60–70 % kódu aplikace, zbylou část je třeba vytvořit pro každou platformu zvlášť. Tento fakt, ačkoliv by se mohl zdát nevýhodou, je naopak výhodou Xamarinu. Je to právě plně nativní uživatelské rozhraní, které je třeba vytvořit pro každou platformu zvlášť, čímž se dosáhne nejlepšího možného výsledku, a právě proto je pro naši aplikaci nejvhodnější. Architektura aplikace v Xamarin je znázorněna na obrázku 7.2. Každá aplikace sdílí datovou vrstvu, vrstvu přístupu k datům, vrstvu služeb a logickou vrstvu. Prezentační vrstvu a uživatelské rozhraní již je potřeba pro každou platformu vytvořit zvlášť[51].

Kapitola 8

Návrh systému BYZNYS Mobile

Tato kapitola se zabývá návrhem samotného systému aplikace. Nejdříve se podíváme na architekturu návrhového vzoru Model-View-ViewModel (dále jen MVVM), která byla zvolena pro strukturu zdrojového kódu aplikace. Druhá část pojednává o struktuře samotné aplikace, o jejím logickém rozdělení na části.

8.1 Architektura MVVM

Jak již bylo zmíněno dříve, pro naši aplikaci budeme využívat prostředí Xamarin s návrhovým vzorem MVVM, jehož diagram je znázorněn na obrázku 8.1. Podíváme se na něj tedy blíže. Návrhový vzor MVVM pochází od Microsoftu a vychází z návrhového vzoru Model-View-Controller (MVC). MVVM umožňuje snadné a úplné oddělení vrstvy uživatelského rozhraní od logické a datové vrstvy, což nám v naší aplikaci umožní sdílet většinu zdrojového kódu mezi platformami[46].

Důležitou vlastností MVVM je „data binding“, což je vlastnost, která umožňuje snadno navázat prvky vrstvy uživatelského rozhraní (View) na data v logické vrstvě (ViewModel). Pokud je „data binding“ nadefinován, změna dat v logické vrstvě vyvolá automatickou propagaci změn do vrstvy uživatelského rozhraní a opačně[48].

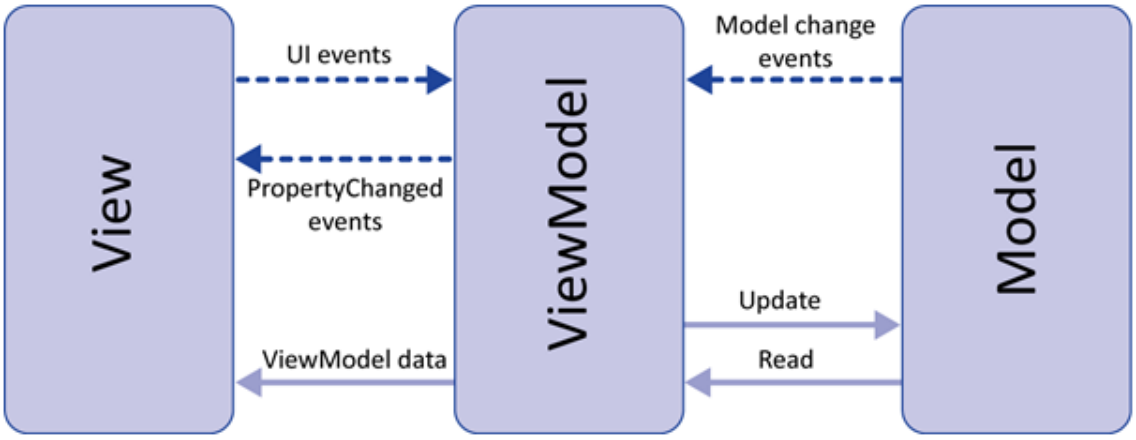
Vrstva uživatelského rozhraní (View) Obsahuje veškeré prvky grafického uživatelského rozhraní, definice jejich vzhledu, pozice. Jsou to všechna tlačítka, popisky, textová pole a další ovládací prvky, ale i obrázky atd. Generuje události v závislosti na akcích uživatele, které jsou zpracovávány korespondující třídou ViewModel.

Vrstva aplikační logiky (ViewModel) Obsahuje aplikační logiku, která slouží jako spojnice mezi View a Model. Získává data z modelu a zpracovává je do formátu vyžadovaného od View, také informuje View o změně dat v Modelu, nebo naopak upravuje data v Modelu na základě jejich změn nebo událostí v uživatelském rozhraní.

Vrstva datová/doménová (Model) Reprezentuje data v databázi nebo objekty modelující reálný stav. Informuje ViewModel, pokud došlo ke změně dat v datovém skladu[28][46].

8.2 Prvotní návrh aplikace

Prvotní optimistický návrh aplikace BYZNYS Mobile systému je znázorněn na diagramu 8.2. K tomuto návrhu by měla aplikace postupně směřovat, avšak bylo možné, že se spe-



Obrázek 8.1: Diagram obecné architektury MVVM [převzato z [28]].

cifikace bude časem měnit. Aplikaci lze rozdělit do několika logických částí navržených na základě specifikovaných požadavků na aplikaci. Jsou jimi následující části:

Jádro Jádrem aplikace budou různá její nastavení, správa uživatelů zahrnující role administrátora, manažera a obchodního zástupce, dále různá nastavení synchronizace a její ruční spuštění. Důležitou částí jádra je i přehledová stránka se statistikami (dashboard).

Adresář Druhá nejdůležitější součást aplikace je adresář, neboli seznam obchodních partnerů. V adresáři bude obchodní zástupce volit obchodního partnera, s jehož daty bude dále pracovat – vytvářet pro něj objednávky. Taktéž budou v adresáři k dispozici veškeré kontaktní údaje partnerů, kontaktní osoby, adresy, poznámky o schůzkách, objednávky daného partnera.

Katalog Katalog produktů bude obsahovat seznam všech produktů v nabídce pro daného partnera, různé možnosti filtrování a zobrazení produktů, individuální ceny pro partnera, detail produktu, fotografie produktu s možností zvětšení a samozřejmě možnost přidat produkty do košíku.

Objednávky Obsahuje seznam objednávek, který nabízí možnost filtrování neodeslaných a odeslaných objednávek, vytvoření nové objednávky a samozřejmě i její úpravu a zrušení. Důležitou součástí je i archiv objednávek.

Faktury Modul faktur spadá do části přehledů o zvoleném obchodním partnerovi. Tyto přehledy budou užitečné pro obchodního zástupce při rozhodování o prodeji partnerovi. Modul faktur bude obsahovat seznamy uhrazených a neuhrazených faktur, dále neuhrazených faktur po datu splatnosti a historii.

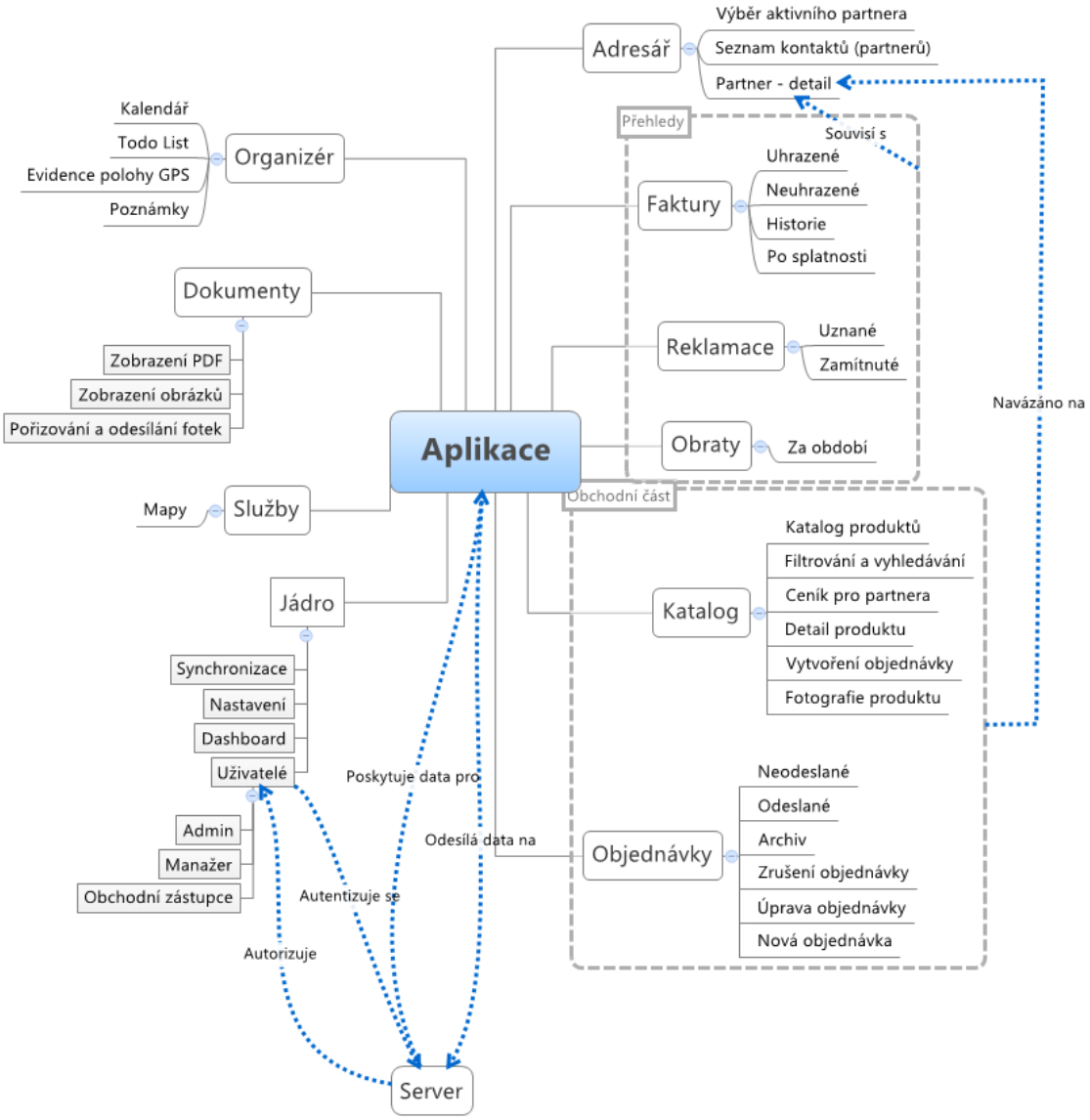
Reklamace Reklamace nabízejí přehledy o všech uznaných a zamítnutých reklamacích od zvoleného partnera.

Obraty Poslední částí přehledů bude statistika obratů s možností filtrování za určité časové období.

Organizér Organizér vztahující se vždy ke zvolenému partnerovi zahrnuje kalendář s poznámkami k různým datům, dále obecný seznam úkolů ke splnění pro partnera, případné poznámky a evidenci GPS poloh zaznamenaných při práci s partnerem.

Dokumenty Dokumentový modul zahrnuje možnost zobrazovat PDF, obrázky a možnost pořizovat a odesílat fotografie.

Služby Služby zahrnují dodatečné funkce aplikace, například mapy poloh různých partnerů.



Obrázek 8.2: Diagram prvotního návrhu logického členění modulů aplikace BYZNYS Mobile [inspirováno z interních dokumentů spol. Q2 Interactive].

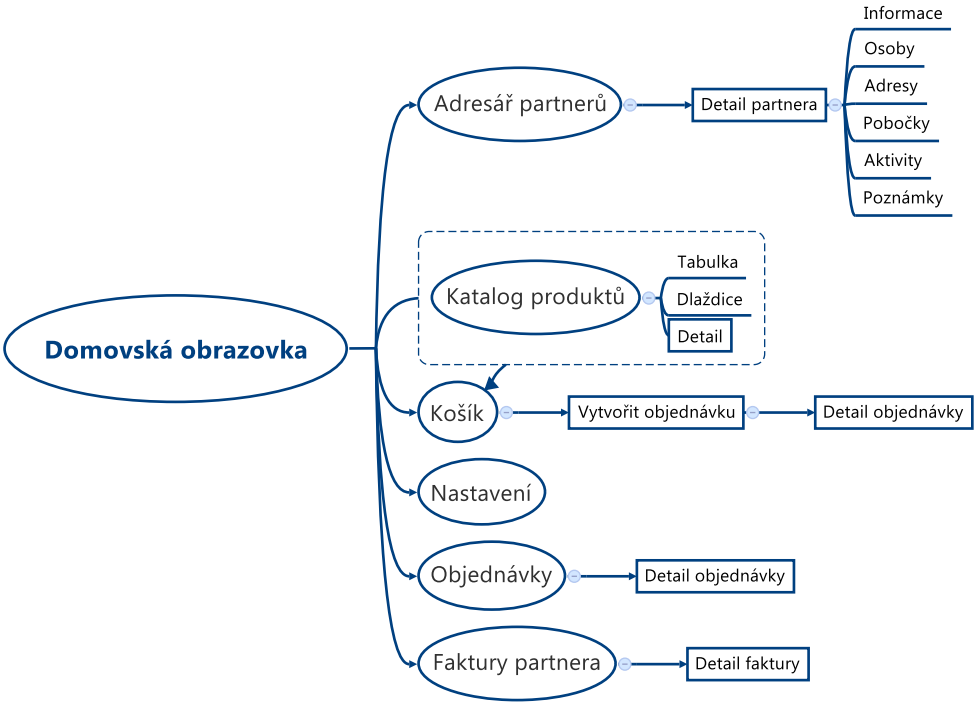
8.3 Návrh obrazovek a navigace aplikace BYZNYS Mobile

Dalším krokem k vývoji aplikace bylo navrhnout obrazovky aplikace, jejich přibližný vzhled a navigaci mezi nimi. Správně vyřešené – rychlé a intuitivní – ovládání a rozvržení aplikace je pro její uživatele klíčové, neboť nesprávně navržené ovládání by si vyžádalo složité zaškolování, nespokojené uživatele a neefektivní práci.

Následující návrh, který je znázorněný na obrázku 8.3, jsem provedl s ohledem na tato fakta a cílem, aby přibližně korespondoval s navigací budoucí první verze aplikace. Pro doplnění návrhu byly vytvořeny i koncepty obrazovek budoucí aplikace[1], které jsou uvedené v příloze B.

Diagram 8.3 ilustruje návrh navigace mezi jednotlivými obrazovkami aplikace. Jednotlivé obrazovky jsem pro přehlednost rozdělil do tří skupin:

- Obrazovky první úrovně navigace – přístupné přímo z domovské obrazovky nebo hlavního menu (na diagramu 8.3 zobrazeny v elipse). Cílem bylo, aby tyto obrazovky byly co nejsnáze přístupné z kteréhokoli místa aplikace.
- Obrazovky druhé úrovně navigace – přístupné z jiných obrazovek (na diagramu 8.3 zobrazeny v obdélníku).
- Záložky – přístupné pouze jako záložky nějaké obrazovky (na diagramu 8.3 zobrazeny podtržením). Slouží pro rozčlenění velkého množství dat na jednu obrazovku.



Obrázek 8.3: Diagram návrhu navigace mezi obrazovkami první verze aplikace BYZNYS Mobile [inspirováno obrázkem 8.2].

[1] Tvorbu konceptů budoucí aplikace jsem provedl ve spolupráci s Bc. Štěpánem Doubalem, zaměstnancem společnosti Q2 Interactive.

Obrazovky první úrovně jsou následující:

Domovská obrazovka – Domovská obrazovka by měla obsahovat hlavní menu celé aplikace, případně nějaké rychlé přehledy.

Adresář partnerů – Bude obsahovat všechny obchodní partnery a umožňovat mezi nimi vyhledávat, filtrovat a volit aktivního partnera. Z obrazovky adresáře partnerů by měl být po zvolení partnera zobrazen detail partnera. Návrh zobrazen na obrázku B.1a.

Katalog produktů – Katalog produktů by měl umožnovat dva druhy zobrazení produktů, dlaždicové a tabulkové. V katalogu produktů by mělo být možné přidávat položky do košíku a taktéž se do něj snadno navigovat. Dále by měl být z katalogu produktů přístupný detail produktu. Návrh je zobrazen na obrázku B.3a.

Košík – Košík bude umožňovat upravovat počty obsažených položek, případně zadávat slevy. Z košíku bude přístupná tvorba nové objednávky. Návrh je na obrázku B.3c.

Nastavení – Obrazovka nastavení by měla obsahovat veškerá možná nastavení aplikace a tlačítka pro manuální spuštění synchronizace.

Objednávky – Seznam objednávek by měl umožnit prohledávání a filtraci objednávek např. dle jejich stavu. Po zvolení objednávky bude přístupný detail objednávky. Návrh je zobrazen na obrázku B.2c.

Faktury – Seznam faktur by taktéž měl umožňovat vyhledávání a filtraci faktur. Ze seznamu faktur bude možno zobrazit detail faktury. Návrh je zobrazen na obrázku B.2a.

Obrazovky druhé úrovně jsou následující:

Detail partnera – Detail partnera bude obsahovat veškeré informace o partnerovi, možnost jeho editace a možnost zvolit jej jako aktivního. Z důvodu množství informací by měl být organizován s využitím záložek. Jednotlivé záložky by měly obsahovat – základní informace, kontaktní osoby, adresy partnera, pobočky, aktivity (schůzky, komunikace) a poznámky. Návrh je zobrazen na obrázku B.1b.

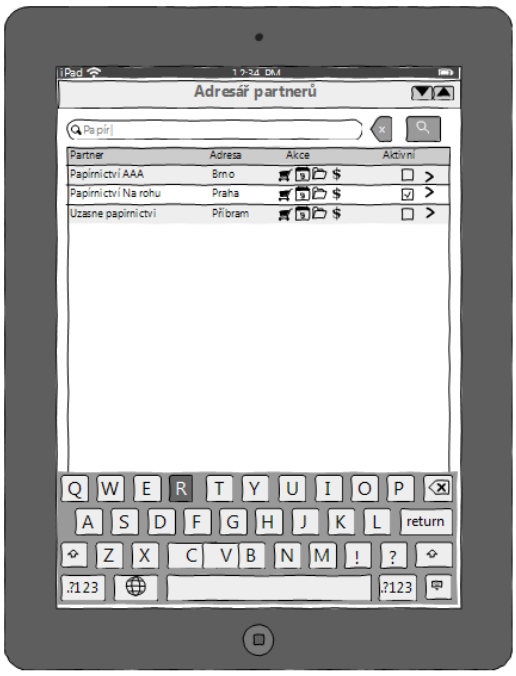
Detail produktu – Měl by obsahovat veškeré dostupné informace a obrázky k produktu, zobrazení posledních 3 objednávek produktu od vybraného partnera, možnost přidat produkt do košíku a samozřejmě tlačítko pro navigaci do košíku. Návrh je zobrazen na obrázku B.3b.

Tvorba nové objednávky – Tvorba nové objednávky umožní vyplnit veškeré možné dodatečné potřebné údaje k objednávce a poté přejít na obrazovku pro potvrzení nové objednávky.

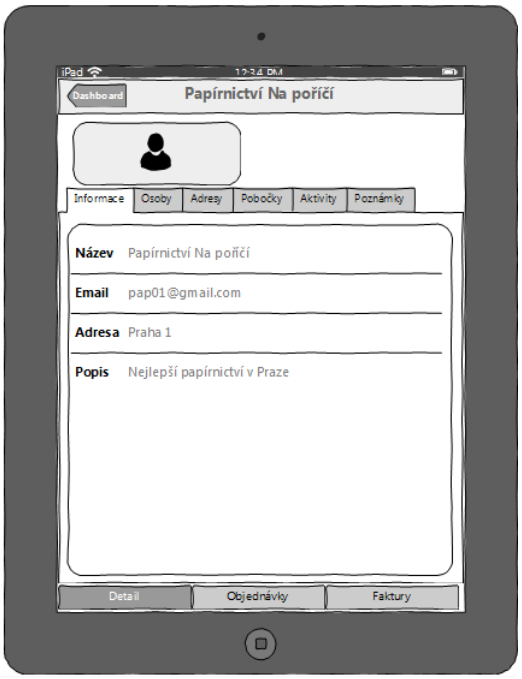
Potvrzení nové objednávky – Bude sloužit pro kontrolu všech údajů (včetně jednotlivých položek objednávky) vyplněné objednávky před jejím odesláním. Obsahuje tlačítko pro odeslání objednávky.

Detail objednávky – Bude zobrazovat veškeré dostupné informace o objednávce včetně seznamu všech jejích položek. Návrh je zobrazen na obrázku B.2d.

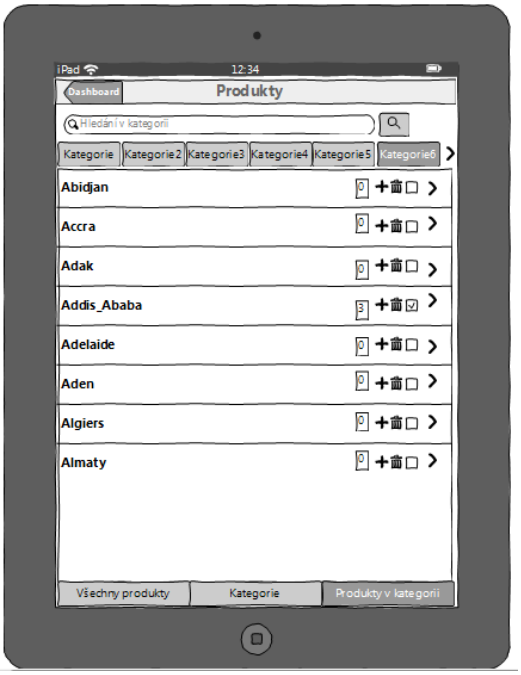
Detail faktury – Bude zobrazovat veškeré dostupné informace o faktuře včetně seznamu všech položek. Návrh je zobrazen na obrázku B.2b.



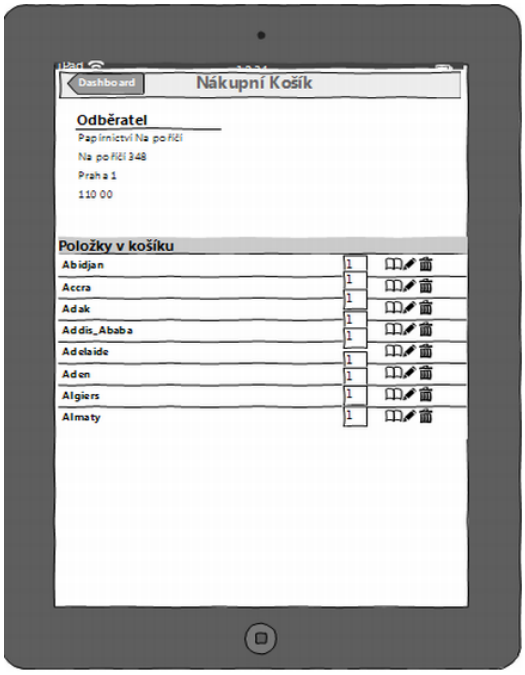
(a) Návrh obrazovky adresáře partnerů.



(b) Návrh obrazovky detailu partnera.



(c) Návrh obrazovky katalogu produktů.



(d) Návrh obrazovky nákupního košíku.

Kapitola 9

Popis implementace aplikace

V této kapitole naleznete popis implementace aplikace, popis při implementaci použitých vývojových nástrojů a jejich hodnocení. Taktéž je zde vymezen rozsah mé vlastní práce a uveden příklad implementovaného modulu pro práci s mapami. Na konci kapitoly se nachází snímky obrazovky výsledené aplikace.

Na základě analýzy dostupných prostředků pro vývoj multiplatformních mobilních aplikací bylo po dohodě s konzultantem zvoleno implementační prostředí Xamarin.

9.1 Použité vývojové nástroje

Tato část se zabývá nejvýznamnějšími nástroji a technologiemi, s nimiž jsem pracoval během implementace aplikace. Taktéž obsahuje jejich zhodnocení a popisuje problémy, s nimiž jsem se setkal při jejich užívání.

9.1.1 Apache Subversion

Pro správu a verzování zdrojových kódů byl zvolen nástroj Apache Subversion (SVN). Tento nástroj byl zvolen z důvodu tradičního využívání ve firmě Q2 Interactive a tudíž velmi dobré obeznámenosti s tímto nástrojem. Klienti pro systém SVN jsou dostupní pro různé operační systémy a v průběhu vývoje projektu byli užívání různí SVN klienti na operačních systémech Linux, OS X a Windows.

Nástroj SVN vyžaduje zřízení tzv. repositáře, kam jsou nahrávány změny ve zdrojových kódech projektu od všech vývojářů. Umožňuje kolaboraci více vývojářů na jednom projektu a dokonce i na stejném souboru. V případě konfliktu úprav více vývojářů na jednom souboru je SVN schopný vyřešit konflikt buď automaticky, nebo umožní vývojáři manuální vyřešení konfliktu.

SVN repositář obsahuje nejen aktuální zdrojové kódy a další zdroje (obrázky, dokumenty, libovolné soubory) projektu, ale i veškerou historii jejich změn od založení projektu a umožňuje návrat k libovolné revizi celého projektu i jednotlivých souborů. Taktéž umožnuje tvorbu různých vývojových větví projektu a jednoduché přepínání mezi nimi, což je funkce, která byla využita pro vývoj specifických větví projektu pro různé klienty[2].

9.1.2 Xamarin

Jako vývojový nástroj pro aplikaci byl po zvážení všech pro a proti byl po dohodě s konzultantem zvolen Xamarin, který byl nakonec upřednostněn před též zvažovaným Embar-

cadero Rad Studio 5. Velkou pomoc při rozhodování poskytl i dokument Cross Platform App Development Tool Benchmarking 2013[36], dále možnost vytvořit uživatelské rozhraní optimalizované pro každou platformu zvláště a taktéž využití moderního programovacího jazyka C#. Neocenitelná byla při vývoji i přehledná a rozsáhlá dokumentace s množstvím návodů a příkladů jak pro Xamarin, tak pro C#.

.NET/C#

Vývojovým jazykem užívaným v Xamarin (i v Mono) je C#. Vývoj mobilních aplikací v C# se později ukázal jako nedocenitelná výhoda, neboť díky platformě Xamarin je možné vytvářet v tomto jazyce aplikace i pro iOS a Android. Vývojář se tedy nemusí učit další programovací jazyky Objective-C a Java. Díky tomu, a samozřejmě i díky možnosti sdílet pro všechny platformy podstatnou část zdrojového kódu, bylo při vývoji ušetřeno mnoho času.

Z vlastností jazyka C#, které nám byly velice užitečné, mohu jmenovat:

- LINQ (Language Integrated Query),
- anonymní funkce, metody a lambda výrazy,
- klíčové slovo var,
- rozšiřující metody (Extensions),
- částečné (partial) třídy, které jsou často využívány v Xamarin.iOS,
- automaticky implementované vlastnosti (property),
- asynchronní metody,
- data binding.

MvvmCross

MvvmCross je multiplatformní framework pro Mono a Xamarin, který se soustředí na zavedení, usnadnění používání a maximální využití návrhového vzoru architektury MVVM (Model-View-ViewModel). Díky návrhovému vzoru MVVM umožňuje MvvmCross sdílet mezi platformami ještě větší množství zdrojového kódu než při běžném využití Xamarin nebo Mono. Jedná se přibližně o 80-90 % kódu.

Při MVVM achitektuře je možno v Xamarin sdílet všechny návrhové vrstvy aplikace kromě vrstvy uživatelského rozhranní, kterou v architektuře MVVM reprezentuje View. Uživatelské rozhraní je v aplikaci Xamarin.iOS, Xamarin.Android i ve Windows Store aplikaci vytvořeno pomocí speciální varianty XML formátu a případně jednoduché podkladové třídy, která může obsahovat složitější logiku uživatelského rozhraní.

Mimo to MvvmCross přidává rozšířenou podporu pro Data Binding, podporu vývoje multiplatformních aplikací pro Windows Presentation Foundation[40].

Hlavní cíle MvvmCross jsou:

- tvorba nativních aplikací,
- přenositelnost,
- znovupoužitelnost,

- nativní uživatelské rozhranní,
- podpora multiplatformního vývoje.

9.1.3 Vývojová prostředí a editory

Během vývoje jsme využili hned několik různých vývojových prostředí a editorů.

Visual Studio 2013

Visual Studio 2013 je nezbytné pro vývoj aplikací na Windows Store. Společně s návrhovým nástrojem pro uživatelské rozhraní Blend pro Visual Studio 2013 umožňuje efektivně oddělit práci programátora a designera aplikace, urychlit tak vývoj paralelní tvorbou oddělené aplikační logiky a uživatelského rozhranní a po dokončení tyto dvě části snadno a rychle provázat dohromady pomocí tzv.„data bindingu“.

Tato výhoda byla znát při implementaci aplikace pro Windows Store, neboť díky předem připravenému uživatelskému rozhraní byla implementace aplikace pro Windows Store v porovnání s iOS o poznání rychlejší[6].

Xamarin Studio

Xamarin Studio bylo využíváno pro vývoj iOS verze aplikace. Jedná se o poměrně užitečné vývojové prostředí s množstvím funkcí a nástrojů, avšak kvality Visual Studia doposud nedosahuje. Jako jeden z největších záporů využívání Xamarin Studia pro vývoj iOS vidím nutnost tvorby uživatelského rozhraní v Interface Builder Xcode. Jeho integrace s Xamarin Studiem není zcela dokonalá, což často přinášelo všelijaké vyjímky (exception) v provázání aplikace s uživatelským rozhraním, které Xcode generuje jako specifickou variantu XML souboru. Více bych ocenil integraci editoru pro návrh rozhraní přímo do Xamarin Studia[7].

Xcode

Xcode je vývojové prostředí od Apple určené pro vývoj aplikací na Mac OS a iOS v programovacím jazyce Objective-C. V Xcode je integrovaný i pokročilý editor uživatelského rozhraní umožňující navrhovat nejen jednotlivé obrazovky, ale i různé vazby mezi nimi. Například umožňuje navrhnout navigaci mezi obrazovkami pomocí grafického editoru.

Věřím, že pro vývoj nativních aplikací v Objective-C je Xcode i jeho editor uživatelského rozhraní skvělým nástrojem, avšak jeho využití společně s Xamarin Studio nebo Visual Studio se neobešlo bez obtíží v podobě hodin strávených u hledání chyby, jako například chybějící či přebytečné reference, nečekaného chování samotného editoru, apod[11].

Sublime Text

Editor Sublime Text 3 je moderní, rychlý editor zdrojových kódů, značkovacích jazyků s podporou zvýrazňovačů pro množství jazyků, nejrůznějších rozšíření a dalších funkcí. Při vývoji jsem jej použil pro tvorbu některých PHP skriptů webové REST služby běžící na Q2 serveru[5].

9.1.4 Simulátory použité pro testování

V případě, že nevlastníme pro testování žádné nebo alespoň dostatek mobilních zařízení pro které vyvíjíme aplikace, je možné využít simulátorů těchto zařízení. Během práce na aplikaci jsem takto využil celkem dva různé simulátory.

iOS Simulator

Originální simulátor iOS má možnost simulovat různé verze iPad a iPhone a mnohé jejich funkce. Simulátor je propracovaný a proti reálnému zařízení má jen velmi málo omezení. Zmínit mohu např. nemožnost testovat senzory rychlosti, polohy, přiblížení, kameru a mikrofon[15].

Při testování na iOS simulátoru mi chyběla především možnost simulovat reálnou rychlost skutečného iPadu. Simulátor byl vždy několikrát rychlejší než skutečný iPad. Taktéž bych ocenil lepší nebo snažší podporu simulování gest využívajících více prstů.

Visual Studio Windows Store simulator

Simulátor Windows Store aplikací se nainstaloval společně s Visual Studiem 2013. Umožňuje simulovat nejen tablety a mobilní telefony, ale i samotný systém Windows 8.1, který máme nainstalovaný, a všechny jeho funkce. Oceňuji na něm snadné a intuitivní simulování gest dotykového zařízení. Jeho nevýhodou je, že neběží v izolovaném prostředí, tudíž mohou vznikat chyby v systému[29].

9.1.5 Databáze

Data v mobilních zařízeních a na serveru je třeba uložit do databáze. Z těch jsme nakonec zvolili níže uvedené MySQL a SQLite, ačkoli se nabízely i jiné možnosti.

SQLite

SQLite je minimalistický vestavěný open source relační databázový systém implementující většinu standardu SQL-92. Zvolen byl pro jeho jednoduchost, minimalističnost, celosvětovou rozšířenost a dostupnost pod licencí Public Domain na všech cílových platformách projektu[1]. Taktéž iOS nativně využívá právě SQLite databázi. Celá databáze SQLite je obsažena v jednom souboru a k datům se přistupuje pomocí funkcí z přilinkované knihovny bez nutnosti komunikace s nějakým dalším procesem, jež by obsluhoval práci s databází.

Nevýhodou SQLite je uzamykání celé databáze během operací zápisu, což může zpomalovat systém při potřebě většího množství zápisových operací.

MySQL

Pro databázi Q2 serveru byla zvolena nejrozšířenější open source databáze MySQL. Důvodem pro její zvolení byla právě její rozšířenost, mnohé zkušenosti s ní ve firmě Q2 Interactive a její open source dostupnost. Ačkoli se nabízela i jiná řešení, jako například MSSQL, MySQL se ukázala jako dostatečná a plně vyhovující potřebám provést logickou synchronizaci s jistými úpravami struktury dat[3].

9.1.6 Další nástroje

Z dalších nástrojů jsem využil například jazyk PHP 5 a na něm založený framework Nette pro implementaci mapového modulu pro webovou službu Q2 serveru a formát JSON pro přenos dat mezi klientskou aplikací a Q2 serverem.

PHP 5

Pro implementaci webové služby na Q2 serveru byl zvolen skriptovací interpretovaný jazyk PHP 5 (Hypertext Preprocessor) z důvodu jeho běžného využívání pro webové stránky ve firmě Q2 Interactive a jelikož je pro tyto účely plně dostačující[4].

Nette

Nette je český open source framework, který byl zvolen pro implementaci REST architektury webové služby na Q2 serveru. Důvodem byla dobrá obeznámenost a časté využívání tohoto frameworku v Q2 Interactive, snadno nastavitelný „routing“ pro jednotlivé REST zdroje a MVP architektura frameworku[31].

JSON

JSON (JavaScript Object Notation) je datový formát užitý pro přenos dat mezi webovou službou a mobilní aplikací.

Důvodem pro jeho zvolení je možnost serializace objektů mobilní aplikace nebo dat z databáze, např. nových objednávek, do textového formátu, který může být snadno odeslán pomocí HTTP protokolu a u příjemce opět deserializován do struktury. Data ve formátu JSON se do REST zdroje webové služby odesílají pomocí metody POST HTTP protokolu a jsou získávána opět ve fomátu JSON pomocí metody GET.

9.2 Vlastní implementace

Na implementaci aplikace BYZNYS Mobile, webové služby pro synchronizaci, přípravu synchronizačního serveru, grafickém návrhu a dalších náležitostech se podílel tým programátorů, grafiků a dalších pracovníků společnosti Q2 Interactive.

Vývoj by se dal rozdělit na tři velké celky:

Vývoj iOS aplikace – Jedná se o největší a časově nejdelší celek, který zahrnoval vývoj kompletních vrstev View, Model a ViewModel pro iOS aplikaci. Vrstva Model byla dále rozdělena na vrstvy DataService, DataAccess a třídy entit. Vrstva DataService pracuje zejména s daty uloženými v operační paměti zařízení a urychluje tak běh aplikace. Vrstva DataAccess pracuje s daty uloženými v databázi SQLite, případně jiném úložišti. Entity představují třídy pro objektově relační mapování databáze a třídy pro abstrakci objektů reálného světa, s nimiž aplikace pracuje – objednávky, jejich položky, partneři, produkty, atd. Dále by se dala vyčlenit i vrstva pro komunikaci s webovou službou Q2 serveru a synchronizaci dat.

Vývoj synchronizačního serveru – Zahrnoval tvorbu MySQL databáze synchronizačního serveru, vývoj webové služby s REST rozhraním poskytujícím přístup k datům pro synchronizaci aplikaci BYZNYS Mobile, vývoj skriptů pro synchronizaci databází Q2 serveru a BYZNYS serveru. Do vývoje webové služby Q2 serveru jsem

přispěl vlastním modulem pro geokódování adres, návrhem tabulky databáze pro jejich uložení a REST rozhraním pro získání dat souřadnic do mobilního zařízení.

Vývoj Windows Store aplikace – Zahrnoval především grafický návrh, následnou tvorbu uživatelského rozhraní v nástroji Blend pro Visual Studio 2013, a poté provázání vzniklých XAML souborů s jádrem aplikace, které bylo vyvinuto již při vývoji iOS aplikace. V této části jsem prováděl implementaci uživatelského rozhraní v XAML do aplikace.

Celkově tedy byly pomocí Xamarin vyvinuty dvě verze aplikace – jedna pro iOS a druhá pro Windows Store.

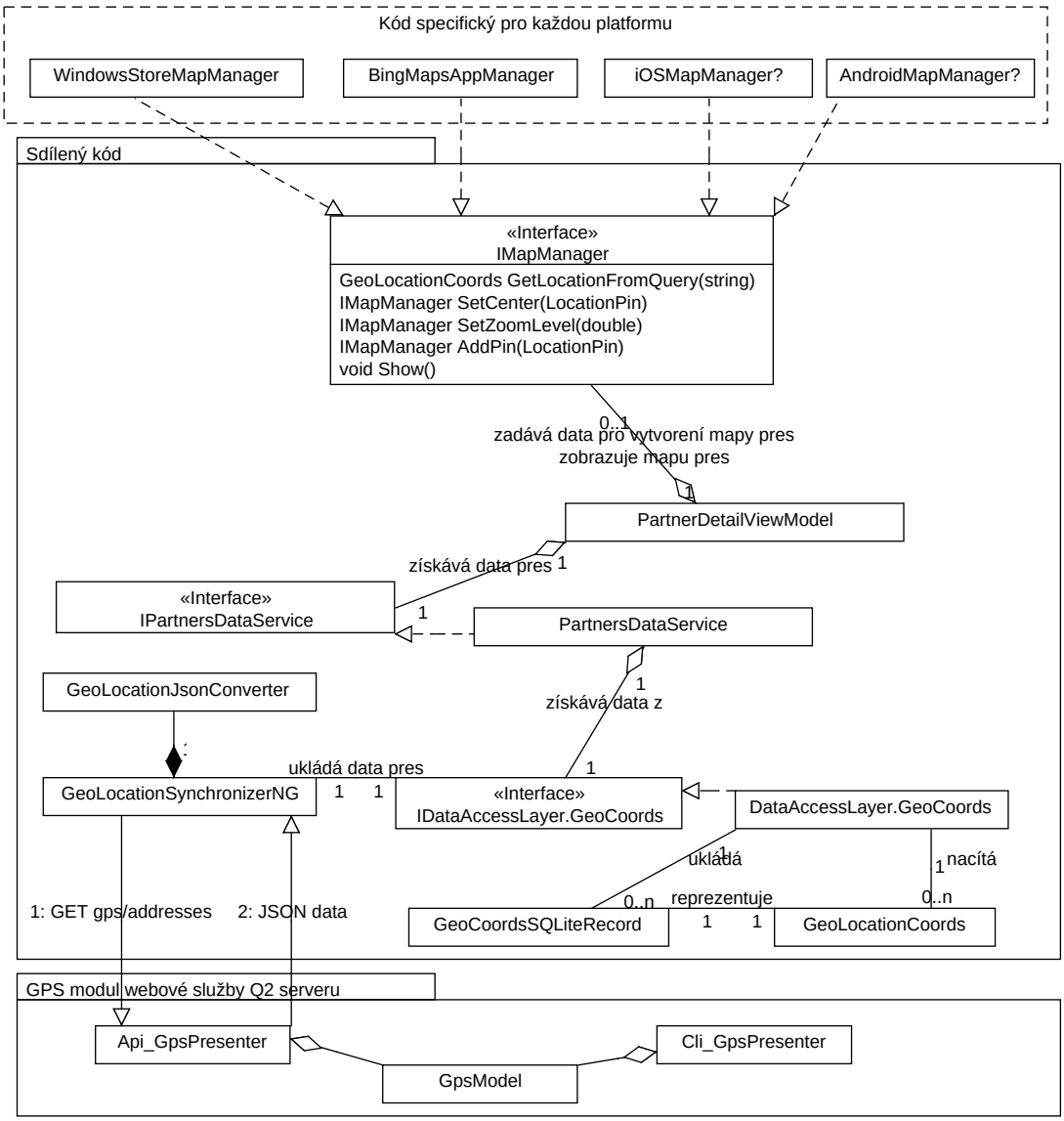
Jelikož se jednalo o projekt vyžadující značné množství člověkohodin, nebylo časově možné, abych jej v rámci diplomové práce implementoval celý sám. Proto uvádím seznam alespoň nejvýznamnějších částí, které jsem samostatně implementoval v rámci celého projektu. Můj díl práce v implementaci byl následující:

- obrazovka detailu partnera – implementace uživatelského rozhranní iOS a vrstvy aplikační logiky (ViewModel);
- obrazovka katalog produktů (částečně zobrazeno na obr. 9.2b) – implementace uživatelského rozhranní iOS, implementace nestandardní komponenty pro stromové menu iOS (View, ViewModel, Model);
- bublinové menu a dialogové filtry iOS;
- nestandardní třída pro snadné užití klouzavého gesta uživatelského rozraní iOS;
- práce s „on-screen“ klávesnicí – automatické posuvy obsahu okna při editaci textového pole;
- úpravy filtrování a vyhledávání v kolekcích dat pro podporu konkurentního a iterativního fultextového vyhledávání při rozšiřování vyhledávaného výrazu;
- optimalizace systému interních zpráv aplikace zasílaných při změně dat a obnovení uživatelského rozhraní při aktualizaci dat;
- optimalizace práce s daty ve vyrovnávací paměti ve vrstvě DataService;
- implementace automatické kontroly aktualizací;
- automatické skrývání předchozích modálních zpráv aplikace při zobrazení nové s rozlišením dle typu zprávy;
- uživatelské rozhraní pro modul katalogu produktů Windows Store aplikace (částečně zobrazeno na obr. 9.2a) – provázání vrstvy View a ViewModel a specifické úpravy funkčnosti pro platformu Windows;
- kompletní návrh a implementace modulu pro zobrazení partnerů na mapě zahrnující implementaci (znázorněno na diagramu 9.1):
  - REST rozhraní webové služby Q2 serveru,
  - služby Q2 serveru pro hromadné geokódování adres partnerů na GPS souřadnice,
  - tvorba databázových tabulek mobilní aplikace i Q2 serveru,

– potřebných tříd a metod na úrovni entit, vrstvy DataAccess, DataService, ViewModel a synchronizace,
– rozhraní (interface) pro jednotnou práci s mapami na všech platformách,
– třídy pro práci s mapami na platformu Windows Store,
– akcí pro zobrazení partnera a nejbližších partnerů v okolí na mapě pomocí aplikace Bing Mapy pro Windows Store (zobrazeno na obrázku 9.3).

9.3 Popis implementace mapového modulu

Pro představu architektury aplikace uvádím diagram tříd mapového modulu, který jsem samostatně naimplementoval.



Obrázek 9.1: Diagram tříd mapového modulu [vlastní tvorba].

Popis tříd diagramu 9.1:

GeoCoordsSQLiteRecord – entitní třída asociovaná se stejnojmennou tabulku SQLite databáze, v níž jsou uloženy zeměpisné souřadnice k adresám partnera. Zajišťuje objektově relační mapování (ORM).

GeoLocationCoords – třída reprezentující zeměpisné souřadnice v aplikaci.

IDataAccessLayer.GeoCoords – rozhraní poskytující metody pro uklání a získávání dat z obecného úložiště.

DataAccessLayer.GeoCoords – třída implementující rozhraní IDataAccessLayer.GeoCoords pro práci s konkrétním úložištěm dat – SQLite databázi.

IPartnersDataService – rozhraní poskytující metody pro práci s daty z vrstvy IDataAccessLayer. Taktéž může poskytovat metody pro práci s daty načtenými do vyrovnávací paměti zařízení („cache“). Jelikož souřadnice náleží k adresám partnerů, zahrnuje toto rozhraní i práci se souřadnicemi.

PartnersDataService – třída obsahující konkrétní implementaci metod rozhraní IPartnersDataService.

PartnerDetailViewModel – třída obsahující metody akcí uživatelského rozhraní obrazovky detailu partnera. Slouží jako prostředník mezi uživatelským rozhraním a nižšími vrstvami aplikace. Taktéž pracuje s třídou implementující rozhraní IMapManager, když reaguje na vyvolání akce pro zobrazení mapy – obvykle klik na příslušné tlačítko.

IMapManager – rozhraní sloužící pro sjednocení práce s mapami na všechny možné platformy. Poskytuje za tímto účelem sadu metod.

BingMapsAppManager, WindowsStoreMapManager – konkrétní implementace rozhraní IMapManager specifické pro každou platformu, případně pro každý mapový systém. BingMapsAppManager zobrazuje mapu v aplikaci Bing Mapy. WindowsStoreMapManager slouží pro zobrazení mapy přímo v mapové komponentě aplikace.

iOSMapManager, AndroidMapManager – totéž, co BingMapsAppManager, WindowsStoreMapManager, avšak prozatím nejsou implementovány, pouze v plánu.

GeoLocationJsonConverter – třída zajišťující konverzi („deserializaci“) JSON dat získaných od webové služby Q2 serveru na objekty typu GeoLocationCoords.

GeoLocationSynchronizerNG – třída implementující metody pro získání zeměpisných souřadnic z webové služby Q2 serveru a jejich uložení pomocí třídy s rozhraním IDataAccessLayer.GeoCoords.

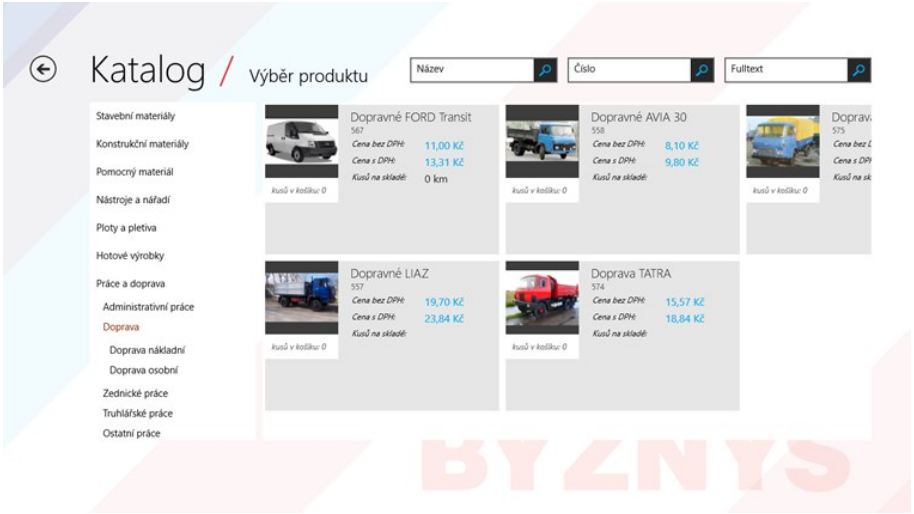
Api_GpsPresenter – třída poskytující zdroje pro REST rozhraní webové služby Q2 serveru, s nimiž komunikuje třída GeoLocationSynchronizerNG. Poskytuje data získaná pomocí třídy GpsModel.

Cli_GpsPresenter – třída poskytující metody pro akce spouštěné pomocí příkazového řádku. Například spuštění procesu geokódování adres partnerů.

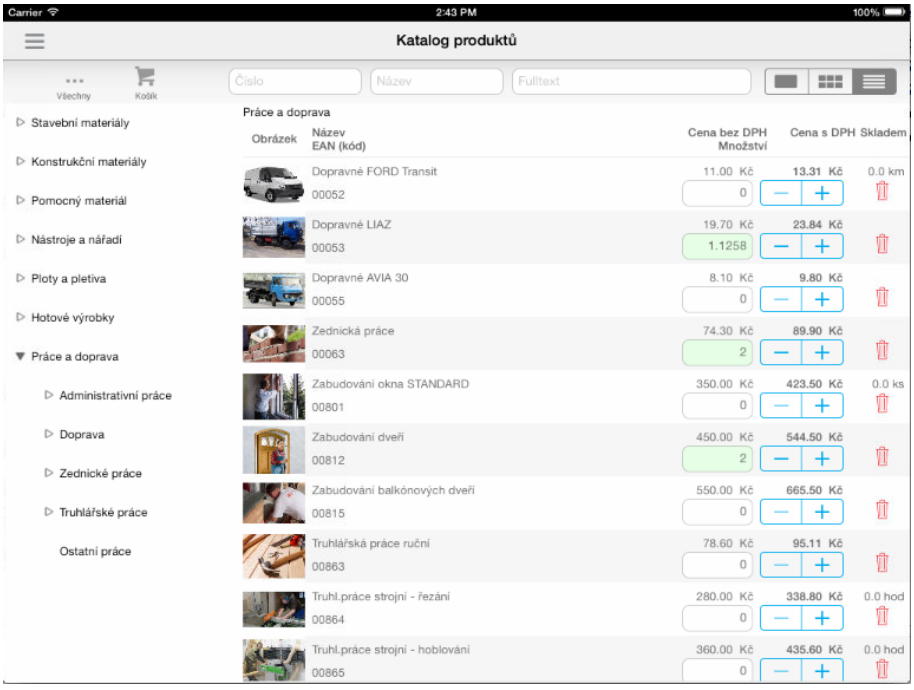
GpsModel – modelová třída zajišťující načítání dat z databáze Q2 serveru a taktéž metody pro geokódování adres partnerů.

9.4 Výsledná aplikace BYZNYS Mobile

Pro srovnání výsledků implementace aplikace pro Windows Store a pro iOS uvádím snímky obrazovky katalogu produktů pro obě platformy.

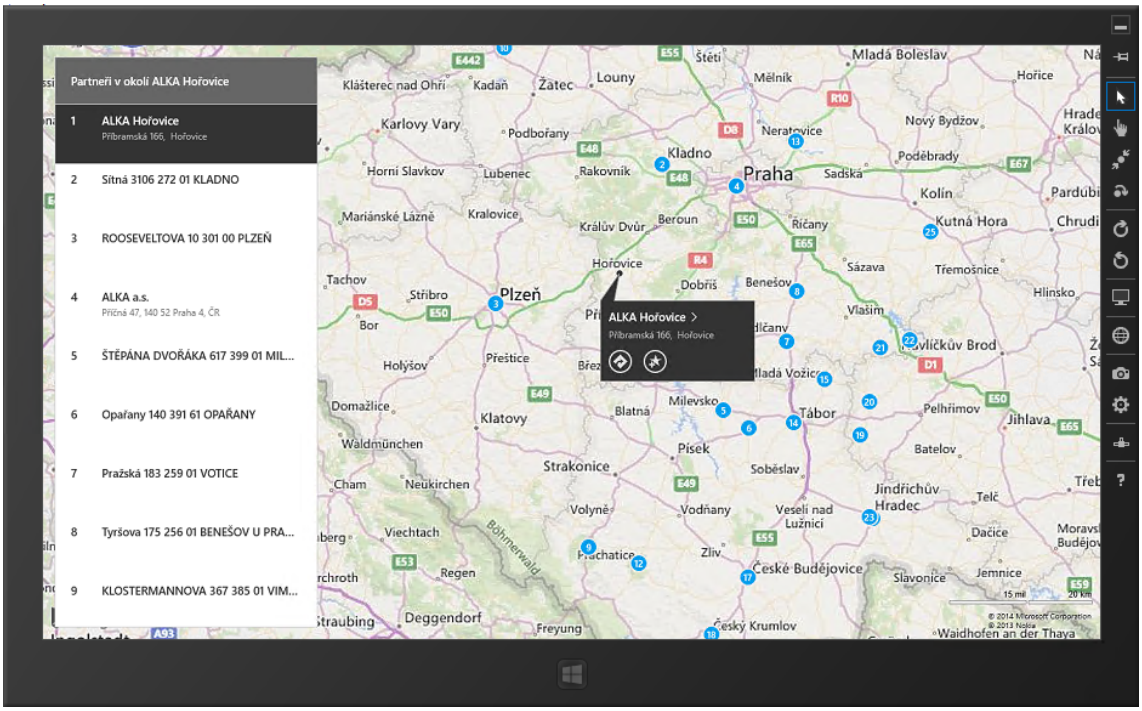


(a) Obrazovka katalogu produktů aplikace BYZNYS Mobile pro Windows Store [převzato z [23]].



(b) Obrazovka katalogu produktů aplikace BYZNYS Mobile pro iOS [vlastní tvorba].

Výsledek implementace modulu pro práci s mapami je zobrazen na obrázku 9.3. Další snímky aplikace jsou v příloze C.



Obrázek 9.3: Obrazovka Windows Store simulátoru s aplikací Bing Mapy, v níž je zobrazena poloha partnera zvoleného v aplikaci BYZNYS Mobile a také polohy nejbližších partnerů v okolí [vlastní tvorba].

Kapitola 10

Testování a ověření funkčnosti aplikace

Během životního cyklu aplikace probíhalo několik druhů testování:

Testování programátory – Zpravidla manuální testování čerstvě implementované funkcionality a případně souvisejícíh částí metodou bílé skříňky (se znalostí detailů implementace), kterého jsem se též účastnil. Náležely sem i zátěžové testy, například testování rychlosti načtení přibližně 70 tisíc faktur a rychlosti kompletní synchronizace za účely optimalizace rychlosti.

Testování funkcionalit – Testy aplikace manuálně prováděné zaměstnanci společnosti Q2 Interactive bez znalosti implementačních detailů metodou černé (bez znalosti implementačních detailů) nebo šedé skříňky (se základní znalostí implementačních detailů). Při tomto druhu testování jsem mohl testovat pouze součásti, které jsem osobně neimplementoval. Jednalo se především o ověřování nejrůznějších scénářů užívání aplikace jejími uživateli, např. procesu tvorby objednávky, editace partnera.

Akceptační/pilotní testování – Manuální testování aplikace klientem metodou černé skříňky, zda aplikace splňuje požadovanou funkčnost, prováděné před jejím převzetím. Do této kategorie náleží i certifikace způsobilosti aplikace pro zveřejnění na Windows Store, k čemuž slouží automatický nástroj Windows App Certification Kit a následně i manuální certifikace testery společnosti Microsoft[27].

Pro své vlastní testování aplikace programátorem a testování funkcionalit jsem volil různé sady testovacích dat – pro zátěžové testy šlo o skutečná provozní data od klienta, pro testy funkcionality naopak o základní předpřipravená testovací data, kterých bylo menší množství a testování díky tomu probíhalo rychleji.

Testování funkčnosti probíhalo především v simulátorech iOS simulator a Visual Studio Windows Store simulator, ale i na reálných zařízeních. Testy zátěže probíhaly pouze na reálných zařízeních, neboť simulátory vykazovaly značně vyšší rychlost, než reálná zařízení.

Kapitola 11

Závěr

Zadáním této diplomové práce bylo seznámit se systémy ERP a zaměřit se na jejich aktuální omezení vycházející z jejich samotné technologické podstaty a faktu, že jde o systémy založené na databázích. Dále bylo zadáno seznámit se s principy multiplatformních aplikací, provést analýzu IT infrastruktury reálné společnosti, specifikovat požadavky na nástroj pro podporu obchodních procesů a uvažovat možnosti synchronizace databází. Zvážit možnosti zabezpečení přenosu dat, způsobu distribuce, aktualizace a centrálního ovládání mobilních zařízení. Posledním úkolem bylo zmíněný systém navrhnout, implementovat a otestovat.

Ze zadání se podařilo splnit všechny body.

V úvodu práce jsem provedl seznámení s ERP systémy, jejich výhodami a omezeními. Dále jsem provedl analýzu infrastruktury reálné společnosti a specifikoval požadavky na nástroj pro podporu obchodních procesů. Byly zváženy možnosti synchronizace databází, bezpečnost přenosu dat mezi nimi a zvolen nejvhodnější způsob realizace této problematiky i z hlediska možností společnosti Q2 Interactive. V práci bylo provedeno seznámení s principy multiplatformních aplikací. Dále byl představen návrh aplikace BYZNYS Mobile a popsána implementace této aplikace, která je již používána v praxi obchodníky společností, které využívají systém BYZNYS ERP. Funkčnost vytvořených aplikací byla pomocí testování ověřována průběžně během vývoje, také při dokončování aplikace pomocí uživatelských testů prováděných klientem, a nakonec i procesem certifikace aplikace pro Windows Store.

Jako nejvýznamnější osobní přínos vidím proniknutí do vývoje multiplatformních mobilních aplikací, získání mnoha cenných poznatků a zkušeností s touto stále se rozvíjející technologií. Taktéž oceňuji získání mnoha zkušeností s programovacím jazykem C#, který mne poměrně nadchnul, seznámení se s problematikou ERP systémů a prohloubení vědomostí o problematice synchronizace databází a bezpečnosti přenosu dat mezi nimi.

Za největší přínos pro vývoj multiplatformní aplikace považuji vývojové prostředí Xamarin, které umožňuje využívat moderního programovacího jazyka C# pro programování mobilních aplikací na všechny hlavní mobilní platformy, sdílet při architektuře MVVM až 90 % kódu, a současně zajišťuje díky nutnosti implementovat vrstvu uživatelského rozhraní pro každou platformu zvlášť i plnou nativnost výsledné aplikace. Tento přínos potvrdila i implementace Windows Store verze aplikace BYZNYS Mobile, která byla připravena za pouhý zlomek času proti vývoji sdíleného jádra a iOS verze aplikace.

Do budoucna by bylo vhodné aplikaci implementovat i pro Android a rozšířit ji o další funkce, například nástroje pro zvyšování prodeje typu cross-selling, up-selling, nebo rozšířit možnosti aplikace o další oblasti práce s ERP systémem.

Jako další vhodné rozšíření se mi jeví implementace role zákazníka. Momentálně je

mobilní aplikace BYZNYS Mobile vhodná pouze pro využití obchodníky společností vlastnících systém BYZNYS ERP. Jistě by však stálo za zvážení vložit do aplikace roli zákazníka (v aplikaci jde o jednotlivé partnery), který by měl pomocí aplikace možnost zobrazovat si nabídku produktů, vytvářet objednávky, zobrazovat vlastní faktury, objednávky a případně upravovat výhradně své vlastní údaje. Toto rozšíření by mohlo ušetřit čas obchodníkům a tím snížit náklady společností užívajících systém BYZNYS ERP. Taktéž by mohlo více zpřístupnit nabídku těchto společností maloodběratelům, na které nemusejí mít obchodníci vždy čas.

Literatura

[1] SQLite. [Online; cit. 2014-05-14].
URL http://http://www.sqlite.org/

[2] Apache Subversion. 2014, [Online; cit. 20-05-2014].
URL http://subversion.apache.org/

[3] MySQL :: The world's most popular open source database. 2014, [Online; cit. 20-05-2014].
URL http://www.mysql.com/

[4] PHP: Hypertext Preprocessor. 2014, [Online; cit. 20-05-2014].
URL http://php.net/

[5] Sublime Text. 2014, [Online; cit. 20-05-2014].
URL http://www.sublimetext.com/3

[6] Visual Studio - Domovská stránka. 2014, [Online; cit. 20-05-2014].
URL http://www.visualstudio.com/

[7] Xamarin Studio - The best IDE for cross-platform mobile development. 2014, [Online; cit. 20-05-2014].
URL http://xamarin.com/studio

[8] Adobe: Adobe roadmap for the Flash runtimes. 2013, [Online; cit. 2014-01-14].
URL http://www.adobe.com/devnet/flashplatform/whitepapers/roadmap.html

[9] ALSHAHWAN, F.; MOESSNER, K.; CARREZ, F.: Evaluation of Distributed SOAP and RESTful Mobile Web Services. *International Journal on Advances in Networks and Services*, ročník 3, č. 3 & 4, 2010.
URL http://epubs.surrey.ac.uk/125638/1/netser_v3_n34_2010_11.pdf

[10] Apple Inc.: Mobile Device Management in iOS. 2014, [Online; cit. 2014-01-14].
URL http://www.apple.com/iphone/business/it/management.html

[11] Apple Inc.: Xcode - What's New - Apple Developer. 2014, [Online; cit. 20-05-2014].
URL https://developer.apple.com/xcode/

[12] bada Developers: The Basic Architecture and UI comparisons between bada and iOS. 2011, [Online; cit. 2014-01-14].
URL http://developer.bada.com/article/
The-Basic-Architecture-and-UI-comparisons-between-bada-and-iOS

[13] BARCAL, J.: *Replikace v relační databázi*. Diplomová práce, Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Praha, 2008.
URL http://www.vse.cz/vskp/show_file.php?soubor_id=1223763

[14] Dbvisit Software Limited, Auckland, New Zealand: *Using Physical Replication and Oracle Database Standard Edition for Disaster Recovery*. March 2012.
URL http://www.dbvisit.com/content/pdfs/Dbvisit_Whitepaper_Physical_Standby_for_Oracle_DR.pdf

[15] iOS Developer Library: iOS Simulator User Guide. [Online; cit. 2014-05-14].
URL https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/TestingontheiOSSimulator/TestingontheiOSSimulator.html#//apple_ref/doc/uid/TP40012848-CH4-SW19/

[16] Economia, a.s.: Stručná historie systémů ERP — HN.IHNED.CZ - systémy crm a erp. 2014, [Online; cit. 2014-01-14].
URL http://hn.ihned.cz/c1-18324610-strucna-historie-systemu-erp

[17] Embarcadero Technologies: RAD Studio XE5 — Android, iOS, Windows and OS X App Development. [Online; cit. 2014-01-14].
URL http://www.embarcadero.com/products/rad-studio

[18] FieldAware: *IS BYOD (BRING YOUR OWN DEVICE) PART OF YOUR COMPANY'S FIELD SERVICE AUTOMATION PLAN?*
URL http://www.aciconsulting.com/assets/is-byod-part-of-your-field-service-automation-plan-fieldaware-white-paper.pdf

[19] HAMMARBERG, E.; GUSTAFSSON, T.: *A Mobile Unit Synchronization Algorithm, A Partial Database Synchronization Scheme between a Centralized Server and Mobile Units*. Diplomová práce, Chalmers University of Technology, University of Gothenburg, March 2011.

[20] HASHIMI, S.; KOMATINENI, S.; MACLEAN, D.: *Pro Android 2*. Apress, 2010, ISBN 13-978-1-4302-2659-8.

[21] HOSEINI, L.: *Advantages and Disadvantages of Adopting ERP Systems Served as SaaS from the Perspective of SaaS Users*. Diplomová práce, School of Information and Communication Technology, 2012.
URL http://www.diva-portal.org/smash/get/diva2:647780/FULLTEXT01.pdf

[22] JACOBS, F. R.; Jr., F. W.: Enterprise resource planning (ERP) – A brief history. *Journal of Operations Management 25*, 2007.
URL http://orennahum.dyndns.org/PDFs/ERP/Enterprise%20resource%20planning%20(ERP)%20-%20A%20brief%20history.pdf

[23] J.K.R. spol. s r.o.: BYZNYS mobile — JKR - dodavatel BYZNYS ERP. 2014, [Online; cit. 2014-05-14].
URL http://www.jkr.cz/byznys-erp/popis-systemu/oblasti-rizeni/informace/byznys-mobile

[24] J.K.R. spol. s r.o.: Popis systému — JKR - dodavatel BYZNYS ERP. 2014, [Online; cit. 2014-01-14].
URL http://www.jkr.cz/byznys-erp/popis-systemu

[25] MIJAČ, M.; PICEK, R.; STAPIC, Z.: Cloud ERP System Customization Challenges. In *Central European Conference on Information and Intelligent Systems*, Varaždin, Croatia: Faculty of Organization and Informatics, September 18-20 2013.
URL
http://www.ceciis.foi.hr/app/public/conferences/1/papers2013/637.pdf

[26] MONK, E.; WAGNER, B.: *Concept in Enterprise Resource Planning*. Course Technology Cengage Learning, 2009, ISBN 13-978-1-4239-0179-2.

[27] MSDN – the Microsoft Developer Network: Certifikace aplikace. [Online; cit. 2014-05-20].
URL http://msdn.microsoft.com/cs-cz/library/windows/apps/hh694079.aspx

[28] MSDN – the Microsoft Developer Network: Implementing the Model-View-ViewModel Pattern. [Online; cit. 2014-01-14].
URL http://msdn.microsoft.com/en-us/library/ff798384.aspx

[29] MSDN – the Microsoft Developer Network: Run Windows Store apps in the simulator. [Online; cit. 2014-05-20].
URL http://msdn.microsoft.com/cs-cz/library/windows/apps/hh694079.aspx

[30] MSDN – the Microsoft Developer Network: Understanding WS-Security. [Online; cit. 2014-05-14].
URL http://msdn.microsoft.com/en-us/library/ms977327.aspx

[31] Nette Foundation: Rychlý a pohodlný vývoj webových aplikací v PHP — Nette Framework. 2014, [Online; cit. 20-05-2014].
URL http://nette.org/

[32] NOVÁK, I.; BALÁSSY, G.; ARVAI, Z.; aj.: *Beginning Windows 8 Application Development*. John Wiley & Sons, Inc., 2012, ISBN 978-1- 118-01268-0.

[33] PAUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F.: RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In *Proceedings of the 17th International World Wide Web Conference*, Beijing, China, April 21–25 2008.
URL http://www2008.wwwconference.org/papers/pdf/p805-pautassoA.pdf

[34] PhoneGap: PhoneGap — FAQs. [Online; cit. 2014-01-14].
URL http://phonegap.com/about/faq/

[35] PLATT, D.: SaaS Adoption Breeds Compliance Challenges. *ISSA Journal*, Dec 2008.
URL http://www.symplified.com/articles/ISSA-SaaS%20Adoption-1208.pdf

[36] research2guidance.com: Cross Platform Tool Benchmarking 2013. 2013, [Online; cit. 2014-01-14].
URL
http://research2guidance.com/cross-platform-tool-benchmarking-2013/

[37] SAINI, S. L.; SAINI, D. K.; YOUSIF, J. H.; aj.: Cloud Computing and Enterprise Resource Planning Systems. In *Proceedings of the World Congress on Engineering 2011 Vol I*, London, U.K.: WCE 2011, July 6 - 8 2011.
URL http://www.iaeng.org/publication/WCE2011/WCE2011_pp681-684.pdf

[38] SCHOLZ, M.: *Komparace modelů on-demand a on-premise pro dodávku, nasazení a provoz ERP systémů v malých podnicích*. Diplomová práce, Vysoké učení technické, Fakulta podnikatelská, Brno, 2012.

[39] SINOFSKY, S.: Building Windows for the ARM processor architecture. 2012, [Online; cit. 2014-01-14].
URL http://blogs.msdn.com/b/b8/archive/2012/02/09/building-windows-for-the-arm-processor-architecture.aspx

[40] SLODGE, S.: MvvmCross. [Online; cit. 2014-05-14].
URL https://github.com/MvvmCross/MvvmCross

[41] SOUPPAYA, M.; SCARFONE, K.: *Guidelines for Managing the Security of Mobile Devices in the Enterprise*. National Institute of Standards and Technology, první vydání, June 2013.
URL http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-124r1.pdf

[42] STAGE, A.: Synchronization and replication in the context of mobile applications. March - April 2005.
URL http://www14.in.tum.de/konferenzen/Jass05/courses/6/Papers/11.pdf

[43] URBAN, J.: *Specifikace „Web Services Security“*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2005.
URL https://is.muni.cz/th/50769/fi_m/DP_Jan_Urban.pdf

[44] VONDRA, T.: Replikace v PostgreSQL. Praha, 2011.

[45] VÁVRŮ, J.: *iPhone vývoj aplikací*. Grada Publishing, a.s., 2012, ISBN 978-80-247-4457-5.

[46] Wikipedia: Model View ViewModel — Wikipedia, The Free Encyclopedia. 2013, [Online; cit. 14-01-2014].
URL http://en.wikipedia.org/w/index.php?title=Model_View_ViewModel&oldid=581807335

[47] Wikipedia: Android (operating system) — Wikipedia, The Free Encyclopedia. 2014, [Online; cit. 14-01-2014].
URL http://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=590588088

[48] Wikipedia: Data binding — Wikipedia, The Free Encyclopedia. 2014, [Online; cit. 13-01-2014].
URL http://en.wikipedia.org/w/index.php?title=Data_binding&oldid=589722210

[49] Wikipedia: Enterprise resource planning — Wikipedia, The Free Encyclopedia. 2014, [Online; cit. 13-01-2014].
URL http://en.wikipedia.org/w/index.php?title=Enterprise_resource_planning&oldid=589992229

[50] Wikipedia: Web service — Wikipedia, The Free Encyclopedia. 2014, [Online; cit. 26-05-2014].
URL
http://en.wikipedia.org/w/index.php?title=Web_service&oldid=609540911

[51] Xamarin: Build apps with C# and .NET for iOS, Android, Mac and Windows. [Online; cit. 2014-01-14].
URL http://www.xamarin.com

[52] Xamarin: Part 3 - Setting Up A Xamarin Cross Platform Solution. [Online; cit. 2014-01-14].
URL http://docs.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_3_-_setting_up_a_xamarin_cross_platform_solution/

[53] Xamarin: Understanding the Xamarin Mobile Platform. [Online; cit. 2014-05-14].
URL http://docs.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_1_-_understanding_the_xamarin_mobile_platform/

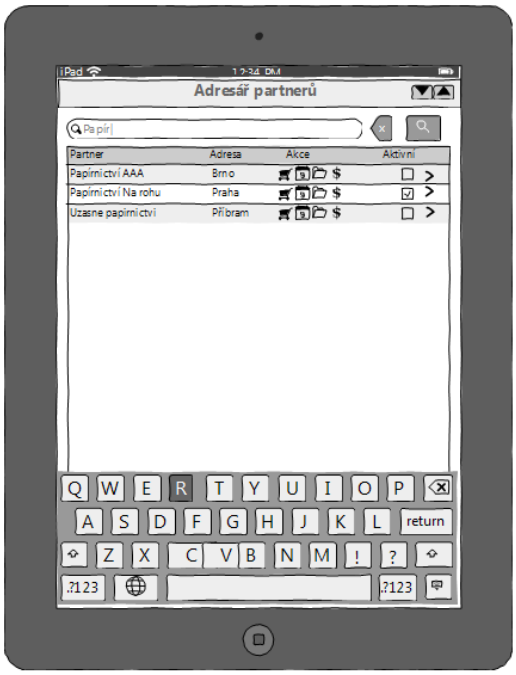
Příloha A

Obsah CD

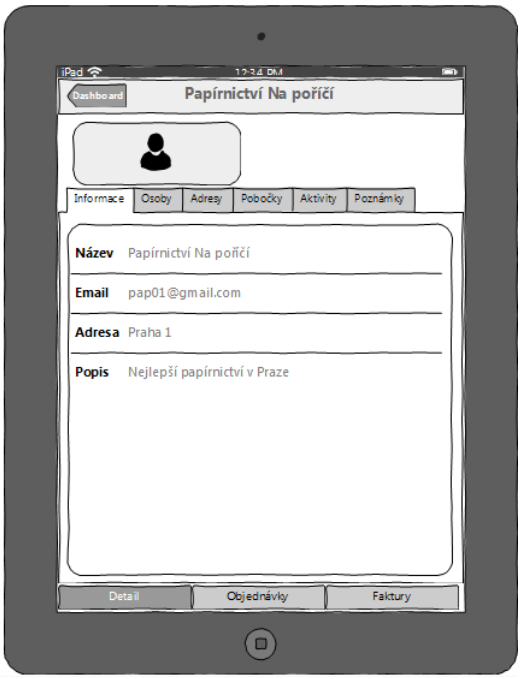
- pdf verze diplomové práce,
- zdrojové texty diplomové práce.

Příloha B

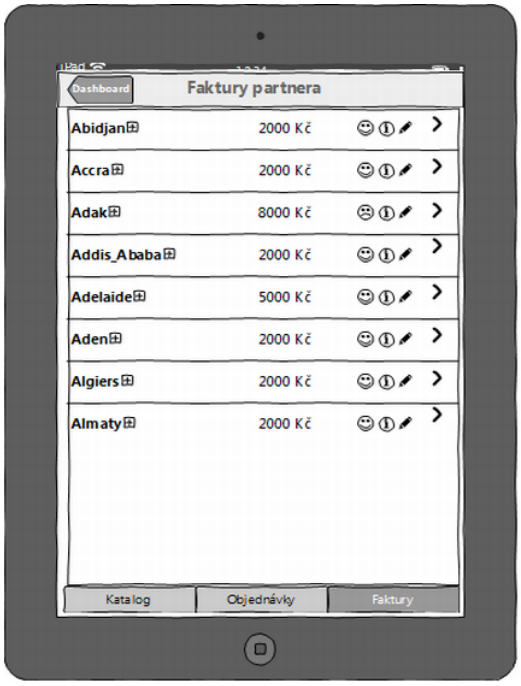
Návrhy obrazovek budoucí aplikace



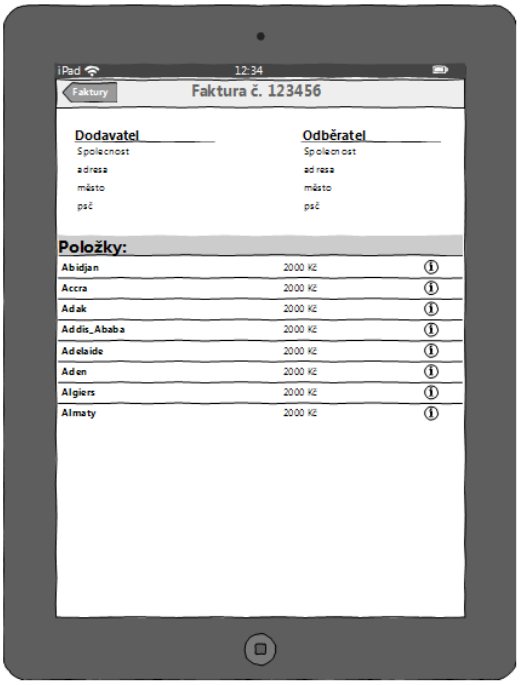
(a) Návrh obrazovky adresáře partnerů.



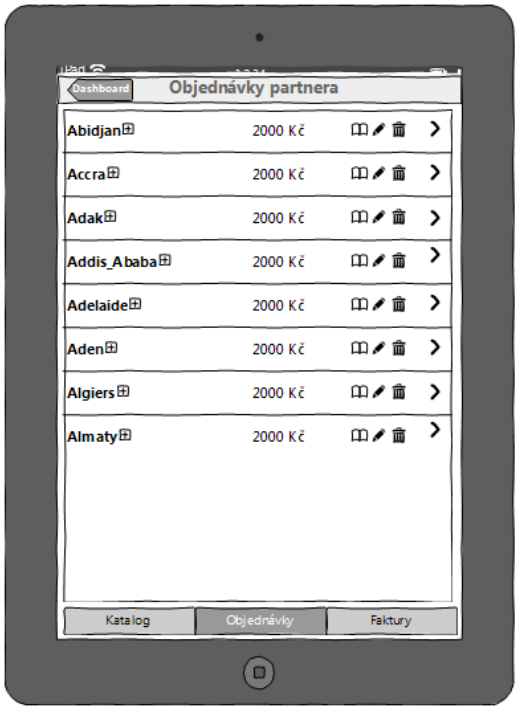
(b) Návrh obrazovky detailu partnera.



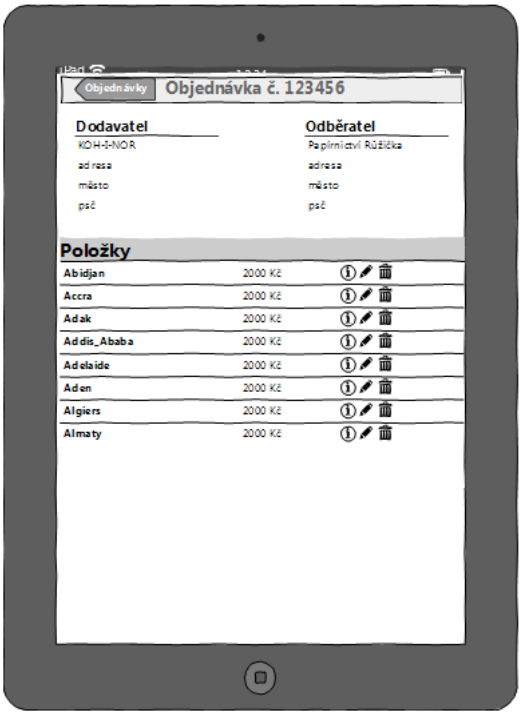
(a) Návrh obrazovky faktur partnera.



(b) Návrh obrazovky detailu faktury.



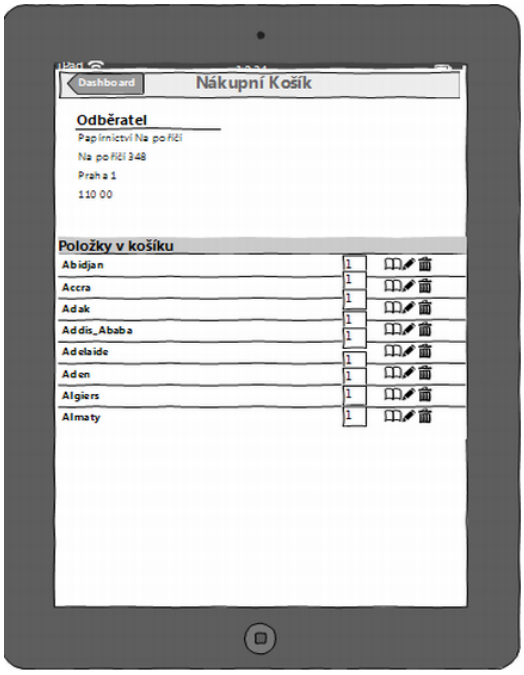
(c) Návrh obrazovky objednávek partnera.



(d) Návrh obrazovky detailu objednávky.

(a) Návrh obrazovky katalogu produktů.

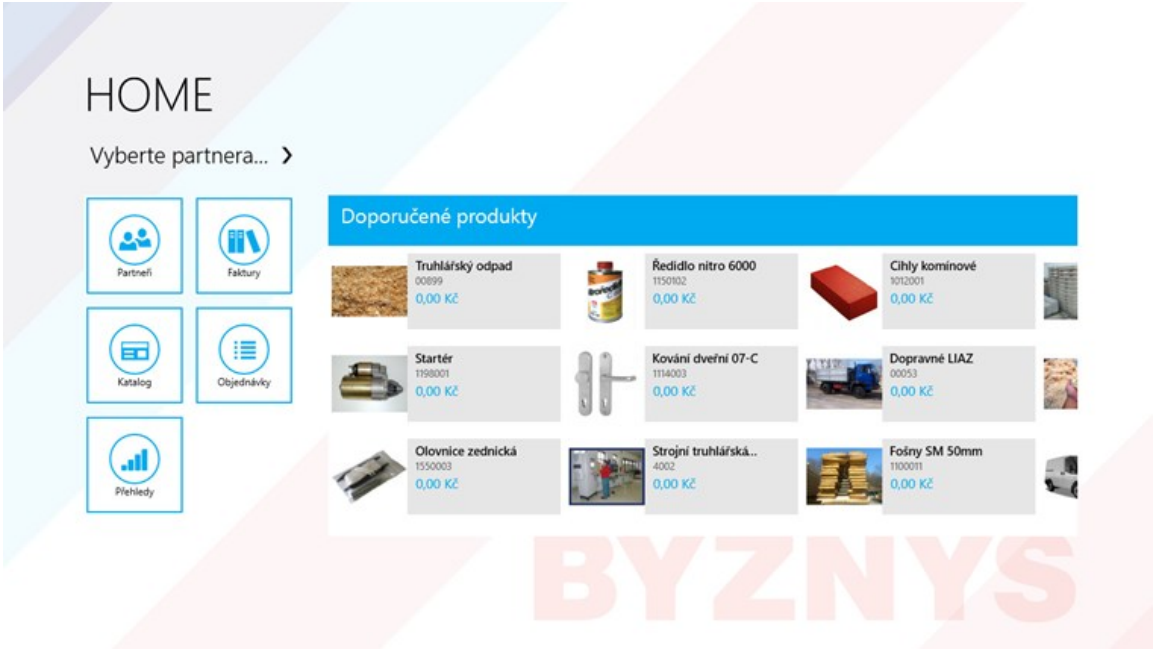
(b) Návrh obrazovky detailu produktu.



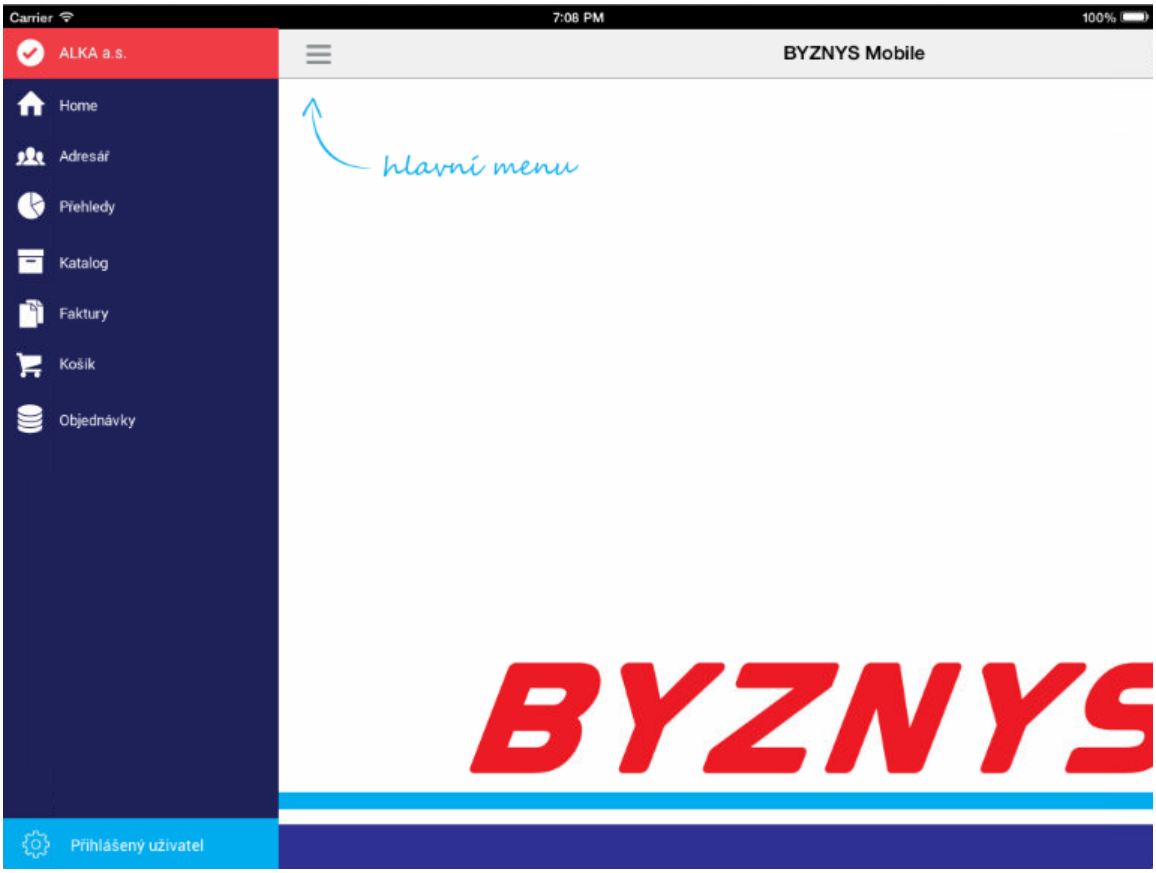
(c) Návrh obrazovky nákupního košíku.

Příloha C

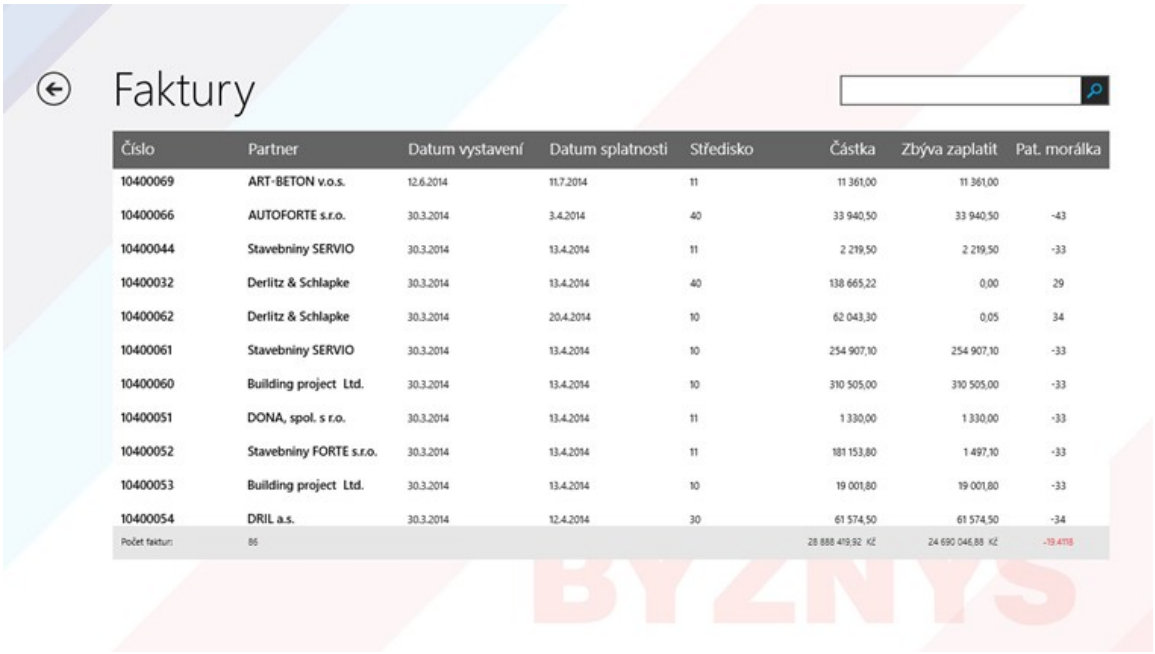
Snímky obrazovky výsledné aplikace pro iOS a Windows Store



Obrázek C.1: Obrazovka domovské stránky aplikace BYZNYS Mobile pro Windows Store [převzato z [23]].

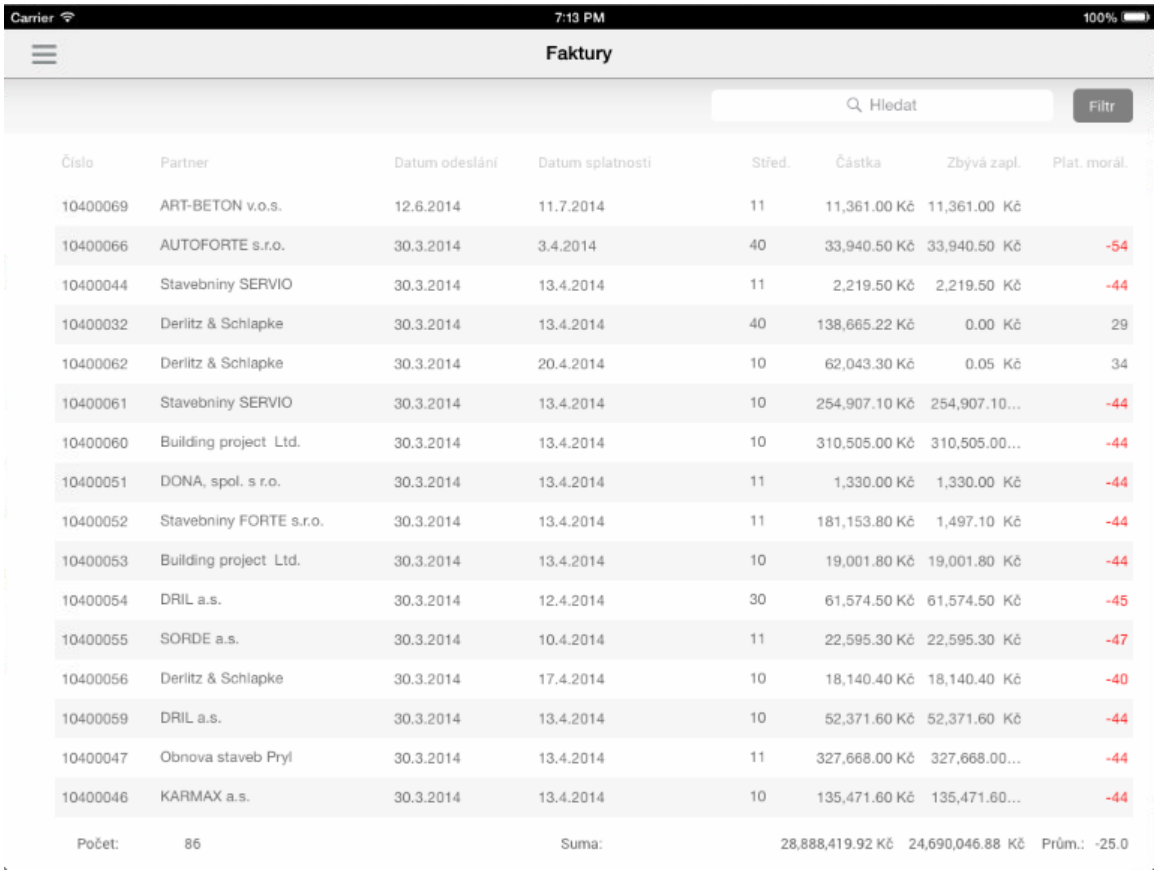


Obrázek C.2: Obrazovka domovské stránky aplikace BYZNYS Mobile pro iOS [vlastní tvorba].



Číslo	Partner	Datum vystavení	Datum splatnosti	Středisko	Částka	Zbýva zaplatit	Pat. morálka
10400069	ART-BETON v.o.s.	12.6.2014	11.7.2014	11	11 361,00	11 361,00	
10400066	AUTOFORTE s.r.o.	30.3.2014	3.4.2014	40	33 940,50	33 940,50	-43
10400044	Stavebniny SERVIO	30.3.2014	13.4.2014	11	2 219,50	2 219,50	-33
10400032	Derlitz & Schlapke	30.3.2014	13.4.2014	40	138 665,22	0,00	29
10400062	Derlitz & Schlapke	30.3.2014	20.4.2014	10	62 043,30	0,05	34
10400061	Stavebniny SERVIO	30.3.2014	13.4.2014	10	254 907,10	254 907,10	-33
10400060	Building project Ltd.	30.3.2014	13.4.2014	10	310 505,00	310 505,00	-33
10400051	DONA, spol. s r.o.	30.3.2014	13.4.2014	11	1 330,00	1 330,00	-33
10400052	Stavebniny FORTE s.r.o.	30.3.2014	13.4.2014	11	181 153,80	1 497,10	-33
10400053	Building project Ltd.	30.3.2014	13.4.2014	10	19 001,80	19 001,80	-33
10400054	DRIL a.s.	30.3.2014	12.4.2014	30	61 574,50	61 574,50	-34
Počet faktur:	86				28 888 419,92 Kč	24 690 046,88 Kč	-19,4118

Obrázek C.3: Obrazovka seznamu faktur aplikace BYZNYS Mobile pro Windows Store [převzato z [23]].



Obrázek C.4: Obrazovka seznamu faktur aplikace BYZNYS Mobile pro iOS [vlastní tvorba].