Masarykova univerzita
Fakulta informatiky



Ukážkové biznis procesy pre platformu Android

Bakalárska práca

Lenka Šišková

Brno, 2013

Prehlásenie

Prehlasujem, že táto práca je mojím pôvodným autorským dielom, ktoré som vypracovala samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovávaní používala alebo z nich čerpala, v práci riadne citujem s uvedením úplného odkazu na příslušný zdroj.

Poďakovanie

Moja vďaka patrí vedúcemu práce pánovi RNDr. Adamovi Rambouskovi a najmä technickému vedúcemu pánovi Jiřímu Svitákovi, bez ktorého by sa táto práca nedostala do finálnej podoby.

Zhrnutie

Teoretická časť tejto práce sa zaoberá obchodnými procesmi a správou obchodných procesov. Poskytuje prehľad nástrojov pre prácu s obchodnými procesmi a následne hlbšie skúma konkrétny nástroj jBPM5. Popisuje špeifikáciu BPMN2. ktorá je štandardom pri modelovaní obchodných procesov a základné komponenty tejto špecifikácie, ktoré podporuje jBPM5. Pozornosť je venovaná najmä obslužným úlohám, ktoré sú vytvárané v praktickej časti. Prieskum sa týka aj platformy Android, konkrétne jej architektúry a následne funkciám, ktoré môže vývojár na tejto platforme využiť.

Praktická časť pozostáva z vytvorenia troch obslužných úloh, ktoré bežia na platforme Android. Ich využitie v reálnych biznis procesoch demonštrujú ukážkové procesy. Funkcionalita vytvorených úloh bola otestovaná na reálnych zariadeniach aj emulátori. V práci sú využívané anglické termíny z dôvodu nezavedených slovenských ekvivalentov.

Kľúčové slová

jBPM5, Android, biznis proces, obslužná úloha, service task, BPMN2, handler, BPM, bussines process management

Obsah

Úvod **1**

1 Biznis procesy **3**

1.1 Špecifikácia biznis procesov – BPMN 2.0 4

1.2 Nástroje pre správu biznis procesov 5

1.2.1 Activiti 5

1.2.2 Bonita Open Solution 6

1.2.3 jBPM 6

1.2.4 IBM Websphere Lombardi Edition 6

2 Použitý nástroj pre správu biznis procesov **8**

2.1 jBPM API 8

2.2 Vytvorenie procesu 9

2.3 Základné podporované BPMN 2.0 komponenty 9

2.4 Obslužná úloha 11

2.4.1 Vlastné obslužné úlohy 11

2.4.2 Synchrónna interakcia 13

2.4.3 Asynchrónna interakcia 13

3 Mobilné platformy **15**

3.1 Symbian 15

3.2 iOS 15

3.3 Android 16

3.4 Windows Phone 16

3.5 BlackBerry 10 17

3.6 Platforma použitá v tejto práci 17

3.6.1 Architektúra platformy Android 17

3.6.1.1 Linuxové jadro 18

3.6.1.2 Knižnice 18

3.6.1.3 Android Runtime 19

3.6.1.4 Aplikačný rámec 19

3.6.1.5 Aplikácie 19

3.6.2 Android SDK ... 20
3.6.2.1 Rozhranie pre programovanie aplikácií – API ... 20
3.6.3 Základné definície Android aplikácie ... 21
4 Návrh riešenia ... 23
4.1 Proces doručovania balíka kuriérskou spoločnosťou ... 23
4.2 Proces plánovania stretnutia/porady ... 24
5 Implementácia obslužných úloh ... 25
5.1 Obslužné úloha GPS location ... 25
5.1.1 Konfiguračný súbor ... 25
5.1.2 Implementácia rozhrania WorkItemHandler – GpsHandler ... 26
5.1.2.1 Zisťovanie polohy zariadenia ... 26
5.1.2.2 Použitie v konkrétnej aplikácii ... 27
5.2 Obslužná úloha SMS location ... 28
5.2.1 Konfiguračný súbor ... 28
5.2.2 Implementácia rozhrania WorkItemHandler – SmsHandler ... 29
5.2.2.1 Posielanie SMS správy s aktuálnou polohou ... 29
5.2.2.2 Použitie v konkrétnej aplikácii ... 29
5.3 Obslužná úloha Add event ... 30
5.3.1 Konfiguračný súbor ... 30
5.3.2 Implementácia rozhrania WorkItemHandler – AddEventHandler ... 31
5.3.2.1 Vytváranie novej udalosti v kalendári ... 31
5.3.2.2 Použitie v konkrétnej aplikácii ... 31
5.4 Repozitár obslužných úloh ... 31
6 Testovanie funkčnosti obslužných úloh ... 33
6.1 Testovanie funkčnosti úloh GPS location a SMS location ... 33
6.2 Testovanie funkčnosti úlohy Add Event ... 34
Záver ... 35
Literatúra ... 36
Elektronicé prílohy ... 37

Úvod

V súčasnosti sa v modernej spoločnosti nájde málokto, kto by nevlastnil mobilný telefón. S rozmachom smartfónov, tzv. inteligených telefónov ľudia nemusia sedieť za počítačom, aby mali prístup k emailom či iným elektronickým službám. Vzhľadom na pokles predaja desktopových počítačov a notebookov v poslednej dobe v prospech prenosnejších zariadení ako sú tablety a smartfóny je migrácia softvérov rôznych typov logickým krokom. Z tohto dôvodu je moja práca zameraná na vytvorenie obslužných úloh a následne ukážkových biznis procesov, ktoré budú spustiteľné na platforme Android, ktorá je momentálne najrozšírenejším operačným systémom v mobilných zariadeniach.

Každá firma má zavedené obchodné procesy, ktoré určitým spôsobom definujú chod a ciele firmy. Mnohé z týchto procesov sú optimalizované pre dosiahnutie maximálneho zisku za minimálnych nákladov, či z hľadiska ľudskej a strojovej práce alebo z finančného, či časového pohľadu. Príkladom takéhoto obchodného procesu je vybavenie objednávky v internetovom obchode, postup pri prijímaní nového zamestnanca, priebeh založenia účtu v banke alebo proces výroby svetla do automobilu.

Na modelovanie, vykonávanie a monitorovanie týchto procesov počas ich životného cyklu sú používané špecifické nástroje. Medzi ne patrí aj open-source nástroj jBPM, momentálne vo verzii jBPM5, ktorý vznikol pod záštitou firmy Red Hat. S jeho použitím som vytvorila 3 obslužné úlohy, ktoré boli následne implementované v dvoch rôznych ukážkových biznis procesoch. Obslužná úloha zvyčajne vykonáva externé činnosti bez nutnosti ľudskej aktivity, napríklad odoslanie emailu. Obchodné procesy, ktoré implementujú tieto obslužné úlohy využívajú špecifikáciu BPMN2, ktorá je štandardom pri modelovaní obchodných procesov.

Moje obslužné úlohy využívajú zdroje mobilnej platformy Android a to konkrétne zistenie polohy zaradenia, službu telephony na odoslanie SMS správy a pridanie udalosti do preddefinovaného kalendára. Jednotlivé úlohy majú tieto funkcionality:

- GPS location: táto úloha na základe dostupnej služby na zistenie polohy určí zemepisnú šírku a zemepisnú dĺžku zariadenia a tieto údaje predá ako výstupný parameter nasledujúcemu uzlu

- SMS location: vstupnými parametrami sú zemepisná šírka a zemepisná dĺžka. Tieto údaje zašle vo forme „latitude longitude“ na zadané číslo.

- Add event: vstupným parametrom sú informácie o stretnutí ako názov, dátum, čas a miesto konania, prípadne detailnejší popis stretnutia. na základe týcho parametrov je do kalendára pridaná nová udalosť.

Implementované obslužné úlohy je možné využiť v navrhnutých ukážkových procesoch:

- Proces doručovania balíka kuriérom, kde sú obslužné úlohy GPS location a SMS location použité na zaslanie polohy kuriéra systému spoločnosti, ktorý overí pozíciu a naplánuje ďalšie doručovanie.

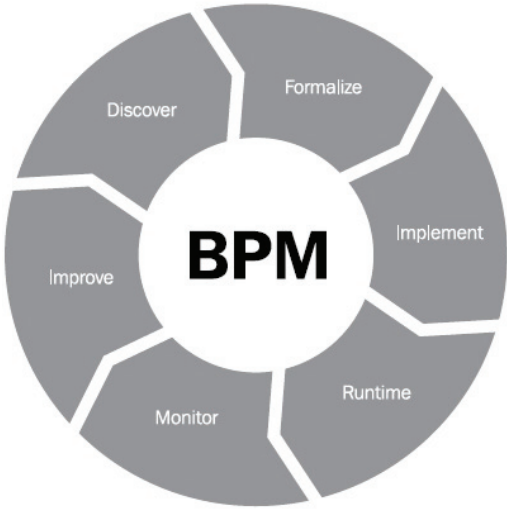
- Proces plánovania stretnutia/porady využíva obslužnú úlohu Add event, ktorá na základe naplánovaného stretnutia pridá toto stretnutie do kalendára.

Jedbým z cieľov práce je darovať vytvorené obslužné úlohy komunite jBPM, aby mohli byť používané a vylepšované ďalšími členmi tejo komunity.

1 Biznis procesy

Biznis proces zahŕňa skupinu aktivít, ktoré berú jeden alebo viac druhov vstupu a produkujú výstup, ktorý má hodnotu pre firmu. Popisuje jednotlivé kroky, ktoré musia byť vykonané pre dosiahnutie určitého cieľa.

Business process management (BPM) je činnosť vykonávaná podnikmi na identifikáciu, vyhodnotenie, a zlepšenie obchodných procesov. s rozvojom technológií, môže byť BPM teraz efektívne riadený softvérom, ktorý sa prispôsobuje na základe metrík a politík uvedených spoločnosťou. Má zásadný význam pre podniky snažiace sa zlepšiť problémy týkajúce sa výkonu či efektivity, aby mohli lepšie slúžiť svojim klientom.[1]



Obrázok 1: Fázy BPM[2]

BPM má 6 fáz, ktoré sa cyklicky opakujú. Najdôležitejšou a zároveň aj najťažšou fázou je definícia biznis procesov. Základom je vedieť, ako firma vykonáva svoju činnosť. Zvyčajne ich definujú obchodní analytici, ktorí potrebujú porozumieť jednotlivým činnostiam a cieľom v rozličných oblastiach firmy. Akonáhle je identifikovaný nový biznis proces, jeho účastníci a cieľ, je možné vytvoriť formálnu definíciu procesu, ktorá sa zvyčajne vytvára využitím preddefinovaného jazyka. v súčasnosti je štandardom modelovania procesov BPMN 2.0

(Business process model and notation). V tretej fáze spolupracujú technický pracovníci s obchodnými analytikmi na pridaní technických detailov potrebných pre vykonávanie procesu. V nasledujúcej fáze sú spúšťané zadefinované biznis procesy za účelom riadiť aktivity firmy. Činnosť procesov musí byť vopred otestovaná, aby sa zaistila korektnosť procesu a správne fungovanie interakcie s externými systémami. Účastníci týchto procesov pracujú s jednotným užívateľským rozhraním, ktoré zvyčajne nahradzuje niekoľko rôznych obrazoviek pre odlišné systémy. vo fáze päť, keď už je vykonaných niekoľko procesov, začína sa monitorovanie vykonávania procesov a výkonnosti. Takto získané informácie sú pre firmu nesmierne dôležité z hľadiska ďalšieho vývoja. V poslednej fáze sú na základe analýz predchádzajúcich spustení navrhované vylepšenia. Tieto analýzy sú vykonávané špeciálnymi nástrojmi, ktoré pri vhodnej konfigurácii dokážu odhaliť vzorce, ktoré by inak zostali skryté. Taktiež sa zohľadňujú aj obchodné zmeny, ktoré je potrebné zahrnúť do procesu. Výstup z fázy 6 je zvyčajne použitý ako vstup pre ďalšiu iteráciu.

1.1 Špecifikácia biznis procesov – BPMN 2.0

BPMN je štandard podporovaný Object Management Group. Cieľom tohto štandardu je poskytnúť normalizovanú grafickú podobu biznis procesov. BPMN 2.0 sa stal hlavným štandardom pre tvorbu procesov, pretože umožňuje vytvárať široké spektrum obchodných procesov a naviac ich grafická reprezentácia je ľahko zrozumiteľná a môžu byť spustiteľné. Toto prekonáva priepasť medzi záujmami obchodnej analýzy a IT obmedzeniami pre automatické spustenie procesov.

Od verzie 2.0 BPMN definuje aj serializáciu diagramov narozdiel od predchádzajúcej verzie, kde bola definícia čisto grafická. Toto spôsobilo určité problémy s interoperabilitou, pretože používatelia BPMN 1.x museli použiť XPDL, ktorý je štandardom pre zápis obchodných procesov na báze XML.(eXtensible Markup Language) na zaistenie interoperability každý BPMN 2.0 diagram zodpovedá jednej XML serializácii, ktorá obsahuje všetky sémanticky súvisiace informácie, ktoré súvisia s procesom samotným. Táto jedinečná

serializácia reprezentácie podnikových procesov umožňuje výmenu tohto procesu a jeho využitie v rôznych modelovacích nástrojoch.

BPMN 2.0 taktiež poskytuje štandard nazvaný BPMN Diagram Interchange (BPMNDI), ktorý je tiež založený na XML serializácii. Tento štandard sa používa na doplnenie BPMN 2.0 textovej notácie pridaním grafických informácií o diagrame do jadra XML serializácie. Tento zápis umožňuje importovať BPMN 2.0 diagramy priamo do grafického modelovacieho nástroja, ak sú v súlade s BPMN 2.0 špecifikáciou.[3]

BPMN 2.0 špecifikácia definuje nasledujúce typy uzlov:

- Udalosti sú používané na modelovanie výskytu určitého typu udalosti. Základnými typmi udalostí sú začiatok procesu a koniec procesu. Počas behu procesu sa môžu vyskytnúť
- Aktivity definujú rozličné akcie, ktoré sa vykonávajú počas behu procesu. V závislosti od typu vykonávanej akcie existujú rôzne typy úloh, ako napr. užívateľská úloha a obslužná úloha.
- Brány sú používané na definovanie rôznych vetiev procesu. na základe typu brán môže byť ich vykonávanie paralelné, alebo je vybraná konkrétna vetva na základe podmienky, atď.

1.2 Nástroje pre správu biznis procesov

Nástroje pre správu biznis procesov umožňujú modelovanie, vykonávanie a monitorovanie procesov počas ich životného cyklu. V súčasnosti je dostupných mnoho nástrojov pre prácu s biznis procesmi, či už voľne dostupné, alebo komerčné.

1.2.1 Activiti

Za projektom Activiti stoja bývalí zamestnanci firmy Red Hat pracujúci na projekte jBPM. Projekt Activiti je vyvíjaný pod záštitou firmy Alfresco a je licencovaný pod Apache licenciou.

Posledná stabilná verzia je 5.12. Poskytuje pokročilé webové rozhranie, podporu BPMN 2.0 špecifikácie, má zabudovanú podporu Springu a bežnú intergáciu Drools. Ako aplikačný server používa Apache Tomcat, ktorý vyniká svojou rýchlosťou.

1.2.2 Bonita Open Solution

Projekt Bonita Open Solution založil v roku 2001 Miguel Valdéz Foura z francúzskeho Národného inštitútu pre výskum v oblasti výpočtovej techniky a riadenia (INRIA). Bol sprístupnený vďaka OW2 Consorcium ako open-source alternatíva k existujúcim komerčným riešeniam v oblasti BPM.[4] Poslednou stabilnou verziou je 5.10. Bonita umožňuje grafické vytváranie procesov s BPMN 2.0 špecifikáciou. Sprístupňuje pripojenie k rôznym druhom bežne používaných komerčných aj open-source databáz, posielaniu správ, atď. Podporuje aj prispôsobovanie formulárov. Umožňuje prehľad priradených úloh a majiteľovi procesu správu jednotlivých úloh v rozhraní pripomínajúcom emailového klienta.

1.2.3 jBPM

Projekt jBPM je open-source systém pre správu biznis procesov, ktorý je vyvíjaný pod záštitou firmy Red Hat. Jadrom jBPM je ľahký, rozšíriteľný „workflow engine“ programovaný v čistej Jave, ktorý umožňuje vykonávanie biznis procesov používajúcich najnovšiu BPMN 2.0 (Business Process Model and Notation) špecifikáciu. Beží v akomkoľvek Java prostredí, vložený v aplikácii alebo ako služba. Výhodou pre programátorov je nízkoúrovňové API (Application Programming Interface) Súčasťami sú konzola na spúšťanie procesov a prehľad jednotlivých aktívnych úloh a Guvnor repozitár, ktorý slúži na skladovanie biznis procesov. V ňom je integrovaný aj Web Designer na vytváranie procesov spôsobom drag and drop.

1.2.4 IBM Websphere Lombardi Edition

Spoločnosť Lombardi, ktorej hlavnou činnosťou bol vývoj produktu Lombardi Teamwork slúžiaci ako nástroj na modelovanie biznis procesov, prevzala spoločnosť IBM v roku

2009 a začlenila produkt Lombardi Teamwork do svojho portfólia. Najnovšou verziou je IBM Websphere Lombardi Edition 7.2. Umožňuje grafickú tvorbu procesov, ich simuláciu krokovaním a na optimalizáciu procesu na základe výsledkov zozbieraných počas simulácie. Webové prostredie je určené na správu procesov, zber dát a vyhodnotenie. Viditeľné sú všetky nesplnené úlohy a je možné spúšťať nové procesy. Zaujímavým nástrojom je možnosť zobrazenia histórie a grafu plnenia úloh. IBM je v súčasnosti lídrom na trhu v oblasti BPM systémov.[5]

2 Použitý nástroj pre správu biznis procesov

Vzhľadom na to, že táto práca je vykonávaná pod záštitou firmy Red Hat, na implementáciu obslužných úloh a biznis procesov je použitý nástroj jBPM 5.3.

2.1 jBPM API

API ponúkané platformou jBPM5 je založená na koncepte vedomostí, tzv. knowledge-base. To znamená, že sa nezameriava len na procesy, ale potenciálne povoľuje načítanie aj iných typov znalostí. Tento princíp takmer vôbec neovplyvňuje používateľov, ktorých zámerom je len práca s procesmi, avšak pokiaľ je potrebné využívať v aplikácii aj obchodné pravidlá alebo komplexné spracovanie udalostí, toto API umožňuje pridať rôzne typy zdrojov do tej istej databázy znalostí (knowledge base). Základnými konceptmi sú:

Knowledge Builder – zapuzdruje množinu sémantických modulov a syntaktických analyzátorov, ktoré vedia ako spracovať jednotlivé typy zdrojov. Je zodpovedný za pridávanie a vytváranie binárnej reprezentácie knowledge modelov pripravených na spustenie. Definície môžu byť načítané z rôznych zdrojov, ako napr. z classpath, súborového systému, z URL, atď.

V príklade je načítavaný súbor process.bpmn z classpath aplikácie, a je špecifikované, že pridávaným zdrojom je BPMN2 model.

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
kbuilder.add(ResourceFactory.newClassPathResource("process.bpmn"), ResourceType.BPMN2);
```

Knowledge Base – je úložisko všetkých znalostí aplikácie. Môže obsahovať biznis procesy, obchodné pravidlá, atď. Objekt KnowledgeBase sa naplní získaním všetkých skompilovaných balíkov znalostí nachádzajúcich sa v KnowledgeBuilder ako je demonštrované v nasledujúcom kóde.

```
KnowledgeBasekbase = KnowledgeBaseFactory.newKnowledgeBase();
kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
```

Knowledge Session – využíva sa na komunikáciu s enginom. Musí mať odkaz na databázu znalostí s odkazmi na všetky potrebné definície procesu. Inštancia KnowledgeSession poskytuje procesom všetky služby, ktoré je potrebné spustiť. Je možné vytvoriť niekoľko knowledge session využívajúcich tú istú databázu znalostí, ktoré sú kontextovo oddelené. Získanie novej inštancie založenej na konkrétnej databáze znalostí ukazuje nasledujúci kód.

```
StatefulKnowledgeSession ksession =kbase.newStatefulKnowledgeSession();
```

2.2 Vytvorenie procesu

Biznis proces môže byť vytvorený jedným zo štyroch spôsobov:

- použitím grafického editoru v Eclipse plug-in
- využitím integrovaného Web Designeru
- ako XML súbor podľa XML procesného formátu definovaného v BPMN 2.0 špecifikácii
- priamym vytvorením procesu v Process API

2.3 Základné podporované BPMN 2.0 komponenty

- Počiatočný uzol : Označuje začiatok procesu. Každý proces obsahuje práve jeden počiatočný uzol, ktorý nesmie mať žiadne vstupné spojenie a práve jedno výstupné spojenie.
- Koncový uzol: Označuje koniec procesu. Každý proces má aspoň jeden koncový uzol majúci jedno vstupné spojenie a žiadne výstupné. Tento uzol môže buď ukončiť cestu, pričom paralelné cesty môžu stále pokračovať, alebo ukončiť celý proces, ktorý bude následne označený za hotový.
- Skriptová úloha: Reprezentuje skript, ktorý má byť vykonaný. Má jedno vstupné a jedno výstupné spojenie. Pripojená akcia určuje, čo sa má vykonať, v akom jazyku je kód a samotný kód akcie.

- Obslužná úloha: Reprezentuje interakciu s externým systémom. Všetky činnosti vykonávané externe by mali byť reprezentované s použitím obslužnej úlohy. Má jedno vstupné spojenie a jedno výstupné spojenie.

- Užívateľská úloha: Predstavuje úlohu, ktorá je vykonávaná ľudským účastníkom. Má jedno vstupné spojenie a jedno výstupné spojenie.

- Deliaca brána: Umožňuje vytváranie vetiev v rámci procesu. Má jedno vstupné spojenie a dve alebo viac výstupných spojení. Existujú tri typy deliacich brán:

    - paralelná – tok riadenia pokračuje vo všetkých vetvách simultánne

    - exkluzívna – vykoná sa len jedno z výstupných spojení

    - inkluzívna – vykonajú sa všetky výstupné spojenia ohodnotené ako true

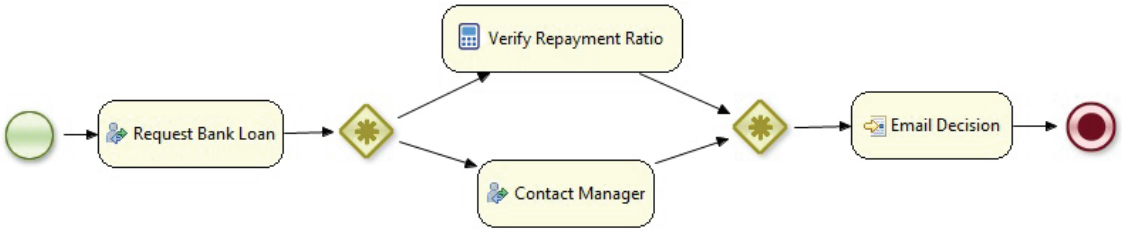
- Spájajúca brána: Umožňuje synchronizáciu viacerých vetiev. Má dve alebo viac vstupných spojení a jedno výstupné. Existujú dva typy spájajúcich brán:

    - paralelná – čaká, kým sa dokončia všetky vstupné vetvy a až potom sa pokračuje

    - exkluzívna – akonáhle je dokončená jedna zo vstupných vetiev, pokračuje sa do ďalšieho uzla.

Nasledujúci príklad zjednodušene modeluje proces žiadosti o pôžičku. Ako prvý je počiatočný uzol. Nasleduje užívateľská úloha Request Loan, ktorá reprezentuje zaznamenanie žiadosti o pôžičku pracovníkom banky. Ďalším uzlom je paralelná deliaca brána, z ktorej vychádzajú dve vetvy. V hornej vetve je obslužná úloha, v ktorej systém banky vyhodnotí schopnosť žiadateľa splácať pôžičku na základe údajov zadaných pracovníkom banky. V spodnej vetve sa nachádza užívateľská úloha Contact Manager, ktorá získa postoj manažéra. Tieto vetvy sa spájajú v spájajúcej bráne, ktorá čaká na dokončenie oboch úloh.

Nasledujúca obslužná úloha zašle žiadateľovi email s konečným rozhodnutím založeným na odpovedi systému a manažéra. Proces je ukončený koncovým uzlom.



Obrázok 2: Ukážkový proces vytvorený v BPMN editore

2.4 Obslužná úloha

Obslužná úloha definovaná BPMN 2.0 špecifikáciou reprezentuje akúkoľvek interakciu so službou mimo pôsobnosti procesu. Príkladom takejto interakcie je napríklad webový server, ktorý vykonáva funkciu vzdialeným volaním procedúry, senzorové snímanie informácie potrebné pre ďalší beh procesu, ovládanie vzdialeného prístroja, atď.

2.4.1 Vlastné obslužné úlohy

Pri vytváraní biznis procesov reálnej firmy je potrebné zachytiť, ako firma rieši konkrétne úlohy pre dosiahnutie želaného cieľa. Avšak nie je možné všeobecne definovať špecifické obchodné situácie a správanie bez kontextu tejto situácie. Preto je nevyhnutnosťou umožniť vytváranie vlastných špecifických obslužných úloh, ktoré reflektujú potreby konkrétnej firmy.

Každá špecifická obslužná úloha musí byť zadefinovaná v konfiguračnom súbore v classpath projektu, kde sú všetky vlastnosti definované dvojicami názov-hodnota. Konfiguračný súbor zvyčajne obsahuje názov, vstupné parametre, výstupy, zobrazený názov, stručný popis a ikonu zobrazovanú v grafickom editore. Ak ikona nie je v rovnakom priečniku ako konfiguračný súbor, je potrebné uviesť absolútnu cestu. Po definovaní

obslužnej úlohy (a v prípade grafického editoru v Eclipse aj zaregistrovaní v súbore drools.rulebase.conf) je dostupná v palete v kategórii Service Tasks a je ľahko použiteľná v procese. Bez ohľadu na parametre definované v konfiguračnom súbore, všetky jednotky práce majú ešte tieto vlastnosti:

- Parameter Mapping: umožňuje namapovať premenné procesu na vstupné parametre úlohy

```
import org.drools.process.core.datatype.impl.type.StringDataType;
[
  [
    "name" : "GPS location",
    "description" : "Locate GPS coordinates",
    "results" : [
       "latitude" : new StringDataType(),
       "longitude" : new StringDataType(),
     ],

    "displayName" : "GPS location",
    "icon" : "gps.gif",
    ]
  ]
]
```

- Result Mapping: umožňuje skopírovanie hodnôt výsledkov do procesných premenných

Zdrojový kód: ukážka konfiguračného súboru GpsDefinitions.wid

Zakaždým, keď je definovaná interakcia s externým subjektom, je potrebné implementovať rozhranie WorkItemHandler. Pokiaľ je vyžadované rovnaké správanie vo viacerých častiach procesu či v iných procesoch, je možné túto implementáciu opätovne využiť. Rozhranie WorkItemHandler definuje metódy executeWorkItem(), ktorá určuje logiku potrebnú na kontaktovanie externej služby, a abortWorkItem(), ktorá má na starosti zrušenie externej interakcie. Každá obslužná úloha musí mať zaregistrovaný handler. Registráciu je nutné spraviť ešte pred spustením procesu. Pri registrovaní implementácie handlera konkrétnej úlohe pristupujeme k manažérovi cez inštanciu triedy KnowledgeSession.

```
ksession.getWorkItemManager().registerWorkItemHandler("Service Task Name",myHandlerImpl);
```

Keď v rámci procesu príde na rad úloha, ktorá pristupuje k externému elementu, manažér WorkItemManager deleguje zodpovednosť za jej vykonávanie na príslušný zaregistrovaný handler, ktorý po dokončení úlohy informuje manažéra. V závislosti od toho, či manažér rozhodne, že interakcia skončila po ukončení metódy executeWorkItem(), alebo čaká na externý stimul aby považoval interakciu za ukončenú, rozdeľujú sa interakcie na synchrónne a asynchrónne.

2.4.2 Synchrónna interakcia

Synchrónna interakcia na konci metódy executeWorkItem() zavolá metódu completeWorkItem(long workItemId, Map results), čo znamená, že po ukončení externej interakcie je enginu poslaná notifikácia, aby pokračoval s vykonávaním procesu. Princípom synchrónnej interakcie je, že vo chvíli, keď je kód vykonaný, engine pokračuje na ďalšiu úlohu v rade. Pri pohľade na takéto správane z pohľadu procesného enginu, je viditeľné, že je blokované vlákno vykonávajúce procesy, kým skončia automatické a synchrónne akcie. To znamená, že je blokovaná celá aplikácia, ktorá zavolala startProcess(). Sú aj prípady, kedy nemôže byť aplikačné vlákno blokované, kým sa ukončia všetky synchronizované aplikácie. Napríklad ak je v jednom procese viacero synchronizovaných interakcií, aplikácia bude blokovaná kým sa nedokončia všetky, V takýchto prípadoch sú využívané asynchrónne interakcie.

2.4.3 Asynchrónna interakcia

Asynchrónna interakcia je realistickejší spôsob interakcie s externými systémami, pokiaľ je naším cieľom robustnosť. V reálnych procesoch nie je vhodné spoliehať sa na synchrónnu interakciu, pretože pokiaľ by volaná služba nebola dostupná, nevyhnutným krokom by bola implementácia rôznych druhov mechanizmov, ktoré by zvládali takúto situáciu. Množstvo týchto mechanizmov je už implementovaných v systéme asynchrónneho zasielania správ.

Asynchrónna interakcia môže byť riešená pridaním komponentu medzi procesný engine a externú aplikáciu. Tento nový komponent má na starosti prijímanie požiadaviek na interakciu s externou aplikáciou. Má sadu dostupných spracovateľov, ktorý

prevezmú žiadosti odoslaných procesným enginom a vykonáva skutočné interakcie s externými aplikáciami. po skončení tento komponent upovedomí žiadateľa, že vykonanie bolo úspešne dokončené.[6]

Od chvíle, keď spracovateľ prijal žiadosť od enginu a poslal späť potvrdenie o prijatí, do doby kým komponent informuje engine o dokončení interakcie, je vlákno, z ktorého bol spustený startProcess() voľné a môže zatiaľ pokračovať v o vykonávaní inej aplikačnej logiky. A akonáhle je proces engine informovaný o dokončení, je zavolaná metóda completeWorkItem() a pokračuje sa vo vykonávaní procesu. Čas, počas ktorého tento proces blokuje aplikačné vlákno je rovný času potrebnému na spustenie procesu a naplánovaniu novej externej interakcie spracúvajúcim komponentom.

3 Mobilné platformy

Mobilná platforma, označovaná aj ako mobilný operačný systém, je softvér, ktorý umožňuje na smartfónoch, tabletoch a ďalších zariadeniach spúšťanie aplikácií a programov. Taktiež spravuje mobilné a bezdrôtové pripojenia a prístup k telefónu. Najstarším operačným systémom pre smartfóny bol Symbian, ktorý sa už dnes v nových modeloch nepoužíva. V súčasnosti je na trhu niekoľko hlavných operačných systémov, medzi ktoré patrí iOS od Apple, Android firmy Google, Windows Phone od Microfost a BlackBerry 10.

3.1 Symbian

V roku 1998 sa sformovalo konzorcium Symbian Ltd. zo spoločností Ericsson, Nokia, Motorola a Psion, za účelom skúmania novej platformy, ktorá vznikla krížením PDA a mobilného telefónu.

Prvé zariadenie kombinujúce mobilný telefón a Symbian OS bolo vydané spoločnosťou Ericsson pod názvom R380. Svoj vrchol dosiahol OS Symbian v roku 2006, kedy bol najpoužívanejším operačným systémom pre smartfóny Odhadovaný podiel na trhu bol 73%.[7]

K rapídnemu poklesu používania OS Symbian došlo po tom ako jeho hlavný používateľ, spoločnosť Nokia, ohlásila 11. februára 2011 prechod na platformu Windows Phone.

3.2 iOS

Tento operačný systém je vyvíjaný a distribuovaný firmou Apple. po prvýkrát bol prezentovaný v roku 2006 na zariadeniach iPhone a iPod Touch. Licencia na iOS nie je poskytovaná iným zariadeniam, ako tým od Apple.

Z pohľadu používateľa je prívetivý a ľahko ovládateľný. Užívateľ si môže rozšíriť funkcionalitu zariadenia s týmto operačným systémom len zakúpením aplikácií v App Store, ktorý obsahuje viac než 700 000 aplikácií, ktoré pred zverejnením podliehajú kvalitatívnej kontrole.

Vyvíjané aplikácie musia byť písané a kompilované konkrétne pre iOS v jazyku Objective-C. Vývojovým prostredím pre iOS SDK (Software Development Kit) je Xcode. Vývojári môžu svoje aplikácie otestovať na simulátore, ale na nahratie aplikácie na zariadenie a zverejnenie v App Storeje potrebné zaplatiť ročný poplatok $99,00.

3.3 Android

Prvý mobilný telefón s operačným systémom Android bol HTC Dream, známy tiež ako T-Mobile G1, ktorý sa dostal na trh v októbri 2008. Vzhľadom na to, že tento operačný systém bol vydaný ako open-source, siaha po ňom mnoho výrobcov mobilných telefónov, pretože umožňuje prispôsobiť si jeho kód vlastným potrebám. Android pravidelne prichádza s novými aktualizáciami.

Používateľmi je obľúbený najmä z dôvodov dostupnosti v rôznych zariadeniach, ktoré sa neobmedzujú na jednu značku, možnostiam upravovania vzhľadu a funkcií a taktiež veľkým množstvom dostupných aplikácií v Google Play Store.

Z pohľadu vývojára je výhodou tejto platformy otvorenosť a rovnocenné postavenie pôvodných aplikácií a aplikácií vyvinutých treťou stranou. Vývoj aplikácií je zvyčajne v jazyku Java s použitím Android SDK, ktorý umožňuje využitie všetkých dostupných funkcionalít.

3.4 Windows Phone

Mobilný operačný systém Windows Phone je vyvíjaný firmou Microsoft. Najnovšia verzia Windows Phone 8 má podobné grafické užívateľské rozhranie ako desktopový operačný systém Windows 8. Nachádza sa v zariadeniach značky Nokia s ktorou Microsoft uzavrel partnerstvo, ale aj v smartfónoch značky HTC, Samsung a Huawei.

Unikátnou funkciou je komplexný Office balík, umožňujúci prácu s dokumentmi Micrososft Word, Excel, prezerať PowerPoint prezentácie a ukladať poznámky s aplikáciou

OneNote. Operačný systém má rýchlu odozvu. Windows Store umožňuje výber zo 120 000 aplikácií.

Pre vývoj vlastných aplikácií pre platformu Windows Phone je potrebné mať nainštalovaný Windows Phone SDK, ktorý je voľne dostupný. Aplikácie sú programované v jazyku C#, XAML (Extensible Application Markup Language) alebo C++. Testovanie na reálnom zariadení a pridanie aplikácie do Windows Store je podmienené zaregistrovaním tohto zariadenia vo Windows Phone Dev Center a zaplatením poplatku $99,00 na rok.

3.5 BlackBerry 10

BlackBerry 10 je špeciálny mobilný operačný systém vyvinutý spoločnosťou Research in Motion pre svoju sériu smartfónov a tabletov BlackBerry. Tento operačný systém má runtime vrstvu, ktorá umožňuje nahranie a spustenie Android aplikácií.

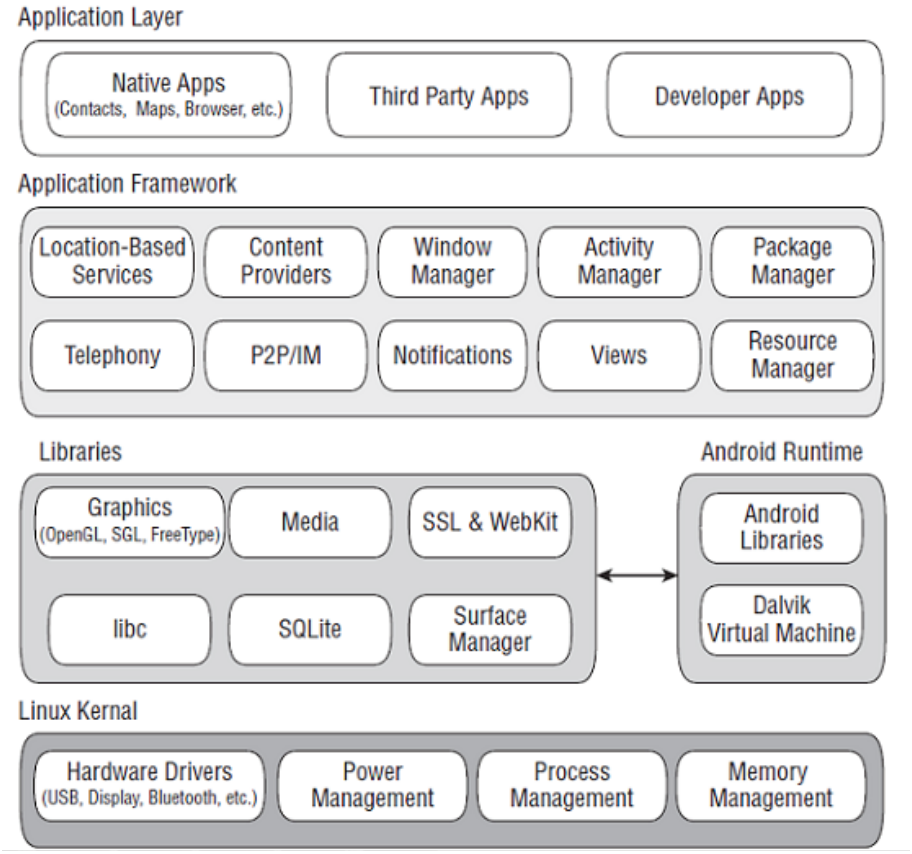
BlackBerry podporuje vývoj aplikácií vo viacerých programovacích jazykoch ako napr. C++, C, Java, HTML5 ai. na začatie vytvárania aplikácií je potrebné požiadať o podpisovacie kľúče, ktoré sú bezplatné. po publikovaní je aplikácia dostupná v BlackBerry World, odkiaľ môže byť nainštalovaná na zariadenie.

3.6 Platforma použitá v tejto práci

V tejto práci je využívaná platforma Android z dôvodu jej rozšírenosti a možností poskytujúcich pre vývoj vlastných aplikácií.

3.6.1 Architektúra platformy Android

Platformu Android tvorí päť vrstiev: linuxové jadro, knižnice, Android Runtime, aplikačný rámec a aplikácie. Architektúra je graficky znázornená na obrázku.



Obrázok 3: Architektúra platformy Android[8]

3.6.1.1 Linuxové jadro

Jadro predovšetkým riadi komunikáciu medzi hardvérom a softvérom. Jeho hlavnou súčasťou sú ovládače, ktoré túto komunikáciu zabezpečujú, napr. ovládač fotoaparátu, zvuku, pamäte, WiFi, obrazovky, atď. Taktiež zaisťuje správu procesov, napájania a pamäti, stará sa o bezpečnosť a pripojenie sietí. Android využíva mierne upravené linuxové jadro, momentálne vo verzii 3.0.31 pre najnovšiu verziu Androidu Jelly Bean.

3.6.1.2 Knižnice

Základné knižnice Androidu sú napísané v jazyku C/C++. Nachádzajú sa to Open GL knižnica pre prácu s 3D grafikou, SGL knižnica pre prácu s 2D grafikou, Media Framework pre prácu s audio a video médiami rôznych formátov, Surface manager ktorý má na starosti správu

zobrazení, SQLite pre základnú podporu databáz, SSL (Secure Socket Layer) a WebKit pre integráciu webového prehliadača a internetovú bezpečnosť a mnohé ďalšie.

3.6.1.3 Android Runtime

Táto vrstva slúži prevažne na beh aplikácií. Aplikácie na platforme Android používajú ako programovací jazyk zvyčajne Javu, takže je potrebný preklad do strojového kódu. na preklad nepoužíva Java Virtual Machine, ale vlastný virtuálny stroj Dalvik . Aplikácie naprogramované zvyčajne v Jave sú preložené do bytového kódu a následne z konvertované z .class súborov kompatibilných s Java Virtual Machine do súborov .dex (Dalvik Executable) kompatibilných so strojom Dalvik ešte pred inštaláciou na zariadenie. Súbory Dalvik Executable sú zostavené tak, aby boli vhodné pre systémy s nižšou operačnou pamäťou a rýchlosťou procesoru, ako sú napríklad smartfóny alebo tablety. Každá aplikácia beží v odlišnom procese s vlastnou inštanciou Dalvik. V tejto vrstve sa nachádzajú štandardné Java knižnice.

3.6.1.4 Aplikačný rámec

V tejto vrstve sa nachádzajú knižnice písané v Jave, ktoré sú tvoria systémové API a sú využívané pri tvorbe Android aplikácií. Zabezpečujú abstrakciu pre prístup k hardvéru, správe užívateľského rozhrania a zdrojom aplikácií.

3.6.1.5 Aplikácie

Aplikácie sú najvyššou vrstvou Androidu. na tejto vrstve užívateľ pracuje so základnými aplikáciami, ako napríklad SMS klient, webový prehliadač, správca kontaktov, ale aplikáciami vyvinutými treťou stranou, ktoré Android považuje za rovnocenné s originálnymi aplikáciami.

3.6.2 Android SDK

Android SDK (Software Development Kit) je súbor vývojových nástrojov, medzi ktoré patria knižnice, emulátor, plug-in pre vývoj aplikácií v prostredí Eclipse, nástroj na odladenie (debugger), dokumentácia, ukážky kódu a návody. Podporované sú taktiež aj staršie verzie Android platformy. Súčasťou SDK je aj SDK manažér, ktorý uľahčuje sťahovanie aktualizácií a doplnkov. Medzi podporované desktopové platformy patria Windows, Linux a Mac OS.

3.6.2.1 Rozhranie pre programovanie aplikácií – API

Možnosť využitia všetkých dostupných funkcionalít zariadenia je nepochybne jedným z dôvodov, prečo sa komunita využívajúca Android veľmi rýchlo rozrastá. Aplikácie pre túto platformu vytvárajú ako skúsení vývojári, tak aj nadšení amatéri práve preto, že na vytvorenie jednoduchej funkčnej aplikácie stačí pár riadkov kódu. Množstvo jednotlivých funkcionalít umožňuje široké spektrum rôznych využití.

- Geografické funkcie – Google Maps, Geocoding, zisťovanie polohy

Služby založené na polohe zariadenia spravujú technológie ako GPS a zisťovanie polohy na základe GSM sietí, vďaka ktorým určujú aktuálnu polohu zariadenia. Zaujímavou funkciou je Geocoding, ktorý na základe zemepisnej šírky a dĺžky určí najbližšiu adresu a taktiež z adresy zistí zemepisné koordináty. Táto služby vyžaduje pripojenie k internetu. Presnosť adresy je závislá na kvalite údajov v databáze.

- Prístup k hardvéru

Knižnice poskytované Androidom zjednodušujú používanie hardvérových komponentov zariadenia, ako je napr. fotoaparát, akcelometer, Bluetooth, Wi-Fi, dotyková obrazovka či GPS. Vďaka nim nie je nutné vytvárať odlišné implementácie pre jednotlivé zariadenia.

- Zdieľané dáta a medziprocesová komunikácia

Android využíva tri techniky na prenos informácií z aplikácií. Notifikáciami zariadenie zvyčajne upozorňuje používateľa. s využitím API je možné spustiť zvukový signál, vibrácie alebo rozsvietiť obrazovku. Ďalším typom sú tzv. Intenty, ktoré prenášajú správy v rámci aplikácie aj medzi aplikáciami. Posledným typom sú poskytovatelia obsahu, ktorý spravujú prístup k súkromným databázam aplikácií.

- Beh aplikácií na pozadí

Vzhľadom na to, že v jednej chvíli je zvyčajne na obrazovke len jedna konkrétna aplikácia, beh iných aplikácií na pozadí je užitočná funkcia. Aplikácie pracujú nenápadne na pozadí, kým sú používané ostatné aplikácie alebo kým nenastane situácia, ktorá si vyžaduje pozornosť užívateľa. Príkladom takýchto aplikácií je napríklad sledovanie akciových trhov, keď pri dosiahnutí nadstavenej sumy sledovanej je užívateľ upozornený, alebo sa na základe dňa a času zmení telefónne zvonenie.[9]

3.6.3 Základné definície Android aplikácie

Každý Android projekt obsahuje manifest súbor, AndroidManifest.xml, uložený v koreňovom priečinku projektu. Manifest definuje štruktúru a metadáta aplikácie a jeho jednotlivých komponentov. Základné informácie obsiahnuté v manifeste sú:

- Minimálnu úroveň Android API, ktorá je vyžadovaná aplikáciou.
- Povolenia nevyhnutné na to, aby aplikácia mohla pristupovať k chráneným častiam API a interakcii s ostatnými aplikáciami. Taktiež deklaruje aj povolenia vyžadovaných od ostatných aplikácií, ktoré by chceli pristupovať k jej komponentom.
- Popis Java balíka, ktorý je unikátnym identifikátorom aplikácie.
- Popis komponentov, z ktorých sa aplikácia skladá.

Vyslovene vyžadované sú len dve položky manifestu a to <manifest> a <aplication> a vyskytujú sa v ňom jen jedenkrát. Väčšina ostatných elementov sa môže v manifeste vyskytnúť niekoľkokrát, alebo sa tam vôbec nemusia nachádzať, avšak aspoň niektoré by sa tam mali nachádzať, aby bol výsledok zmysluplný.[10]

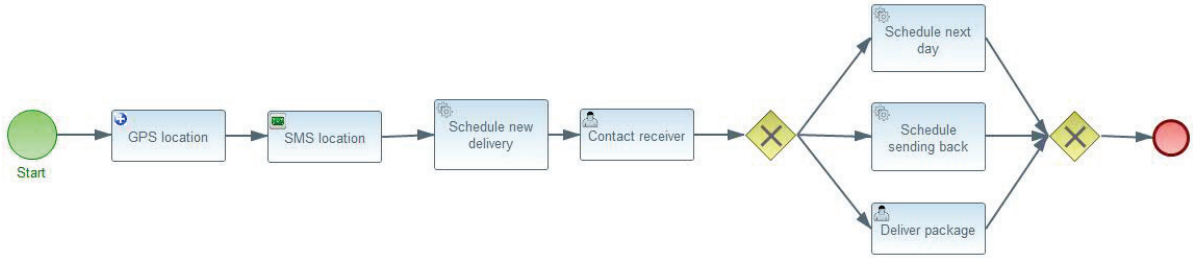
4 Návrh riešenia

V praktickej časti mojej práce navrhnem a implementujem tri obslužné úlohy pre platformu Android a vytvorím dva ukážkové biznis procesy, ktoré modelujú reálne procesy s využitím týchto úloh.

4.1 Proces doručovania balíka kuriérskou spoločnosťou

Prvým ukážkovým procesom, ktorý využíva obslužné úlohy pre Android je proces doručovania balíka kuriérom. Využívané sú v ňom úlohy GPS location, ktorá zistí aktuálnu polohu a získanú zemepisnú šírku a dĺžku uloží do procesných premenných, aby tieto hodnoty mohli byť využité neskôr v priebehu procesu, a úloha SMS location, ktorá na základe hodnôt získaných predchádzajúcou úlohou zistí adresu a odošle ju pomocou SMS na zadané telefónne číslo.

Proces v princípe funguje tak, že po jeho začatí sa zistí aktuálna poloha a koordináty tejto polohy sa odošlú SMS správou do interného systému v tvare „latitude longitude“ . Ten ich vyhodnotí a určí ďalšie miesto doručenia balíka na základe firemnej logiky. Kuriér následne kontaktuje zákazníka a na základe kontaktu je balík buď doručený alebo naplánovaný na rozvoz ďalší deň, prípadne je naplánované vrátenie balíka späť odosielateľovi v prípade odmietnutia prevzatia. Ak sa kuriérovi nepodarí kontaktovať zákazníka, táto úloha zostáva nedokončená až kým sa nepodarí nadviazať kontakt. Zasielanie aktuálne polohy je výhodné nielen z dôvodu plánovania poradia doručenia v priebehu rozvozu, ktoré sa môže dynamicky meniť, ale aj z dôvodu kontroly kuriérov, či sú skutočne na očakávanom mieste.

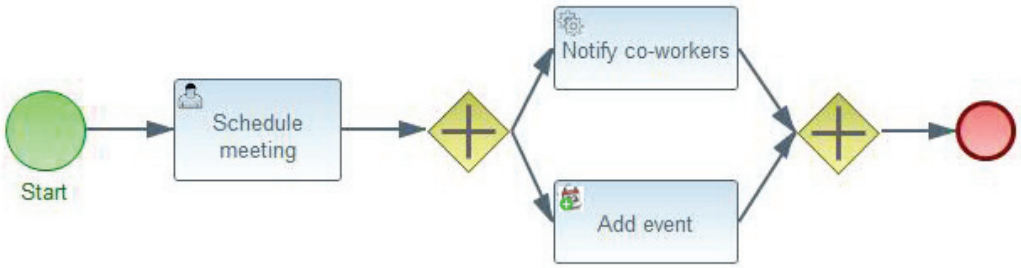


Obrázok 4: Ukážkový biznis proces doručenia balíka kuriérom

4.2 Proces plánovania stretnutia/porady

Druhým ukážkovým procesom využitia obslužných úloh pre Android je proces plánovania pracovného stretnutia. Využíva obslužnú úlohu Add event, ktorá pridá do kalendára naplánované stretnutie. Informácie o stretnutí poskytuje predchádzajúca časť procesu.

Proces funguje tak, že užívateľ naplánuje stretnutie zadaním názvu, dátumu, času, miesta konania, účastníkov, prípadne stručného popisu. Informácie o stretnutí sa pošlú zadaným užívateľom a na telefóne, z ktorého je proces spustený, sa do kalendára pridá toto stretnutie.



Obrázok 5: Ukážkový biznis proces plánovania stretnutia

5 Implementácia obslužných úloh

V implementačnej časti vysvetlím postup pri vytváraní jednotlivých obslužných úloh s dôrazom na implementovanie rozhrania WorkItemHandler.

Implementácia obslužných úloh sa skladá z týchto častí:

- vytvorenie konfiguračného súboru
- implementovanie rozhrania WorkItemHandler

5.1 Obslužné úloha GPS location

Táto obslužná úloha zistí polohu zariadenia a získané koordináty dá na výstup.

5.1.1 Konfiguračný súbor

Konfiguračný súbor tejto úlohy je napísaný v jazyku MVEL a má nasledovné položky:

- názov – ″name″ : ″GPS location″,
- stručný popis – ″description″ : ″Locate GPS coordinates″,
- výstupy typu String – ″latitude″ : new StringDataType(), ″longitude″: new StringDataType(),
- zobrazovaný názov – ″display name″ : ″GPS location″,
- zobrazovanú ikonu – ″icon″ : ″gps.gif″

Rozšírený konfiguračný súbor pre začlenenie obslužnej úlohy do repozitára obslužných úloh obsahuje ešte tieto položky:

- ″eclipse:customEditor″ : ″org.drools.eclipse.flow.common.editor.editpart.work.SampleCustomEditor″,
- kategória obslužnej úlohy – ″category″ : ″Android″,
- základnýý handler : ″defaultHandler″ : ″com.example.iamhere.GpsHandler″,
- dokumentácia/návod na použite – ″documentation″ : ″index.html″,
- závislosti – ″dependencies″ : ″file: ./lib/gpshandler.jar″,

5.1.2 Implementácia rozhrania WorkItemHandler – GpsHandler

Poloha Android zariadenia môže byť zistená na základe

- signálu od prevádzkovateľa mobilnej siete – poloha je zisťovaná na základe posielania dotazov Google location serveru s ID najbližšieho vysielača signálu. Odchýlka sa pohybuje od 100 do 1000 m v závislosti od pokrytia. Výhodou tejto metódy je, že nevyžaduje žiadne ďalšie náklady, napr. na energiu z batérie.
- GPS satelitov – poloha je zisťovaná na základe vzdialenosti zariadenia od minimálne troch rôznych satelitov. Odchýlka sa pohybuje od 5 do 20 m. Výhodou je vyššia presnosť ako pri zisťovaní polohy na základe signálu prevádzkovateľa mobilnej siete. Nevýhodou sú vyššie nároky na energiu batérie a problematické zisťovanie polohy v budovách.

5.1.2.1 Zisťovanie polohy zariadenia

Pre spustenie obslužnej úlohy je vytvorená trieda GpsHandler, ktorá implementuje rozhranie WorkItemHandler. Metóda executeWorkItemHandler() zodpovedná za obsluhu tejto úlohy vykonáva nasledujúci postup.

Na začiatku sú zadefinované dve lokálne premenné typu String, ktoré sú používané na prácu so zemepisnou šírkou a zemepisnou dĺžkou. Následne je inicializovaný manžér triedy

LocationManager, ktorý umožňuje prácu s lokalizačnými službami a listener, ktorý má na starosti sledovanie zmien polohy. po týchto úvodných nadstaveniach začína samotné zisťovanie polohy zariadenia. Vzhľadom na nároky GPS lokalizácie je na určenie polohy uprednostnený signál od prevádzkovateľa mobilnej siete. Overenie dostupnosti sa vykonáva zavolaním metódy isProviderEnabled(LocationManager.NETWORK_PROVIDER) na manažérovi. Ak je dostupný, vyžiada sa aktualizácie polohy a zistí sa posledná známa poloha, ktorú eviduje poskytovateľ siete. Získané hodnoty zemepisnej šírky a dĺžky sú prevedené do typu String vo formáte stupňov a uložené do lokálnych premenných definovaných na začiatku metódy executeWorkItemHandler().Pokiaľ nie je dostupný poskytovateľ siete, sú na zistenie polohy použité GPS satelity. Posledným krokom je vloženie hodnôt lokálnych premenných latitude a longitude do procesných premenných, aby mohli byť tieto hodnoty použité v ďalších častiach procesu. Následne je zavolaná metóda completeWorkItem(i.getId(), results) a proces pokračuje na ďalší uzol.

5.1.2.2 Použitie v konkrétnej aplikácii

Prístup k službám na zisťovanie polohy na zariadení Android vyžaduje určité povolenia, ktoré je potrebné uviesť v súbore AndroidManifest.xml. Povolenie pre prístup k zisteniu polohy na základe signálu od poskytovateľa mobilnej siete je:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Povolenie potrebné pre prístup k zisťovaniu polohy na základe signálu od poskytovateľa siete a zároveň aj GPS satelitmi je :

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Nie všetky mobilné zariadenia majú štandardne zapnuté zisťovanie polohy satelitmi, preto môže byť potrebné ho manuálne zapnúť.

5.2 Obslužná úloha SMS location

Táto obslužná úloha odošle SMS správu s aktuálnou pozíciou na prednadstavené telefónne číslo.

5.2.1 Konfiguračný súbor

Konfiguračný súbor je tejto úlohy napísaný v jazyku MVEL a má nasledovné položky:

- názov – ″name″ : ″SMS location″,
- stručný popis – ″description″ : ″Sending SMS with current location″,
- vstupné parametre typu String – ″latitude″ : new StringDataType(), ″longitude″: new StringDataType(),
- zobrazovaný názov – ″display name″ : ″SMS location″,
- zobrazovanú ikonu – ″icon″ :″sms.gif″

Konfiguračný súbor obslužnej úlohy pre začlenenie do repozitára obsahuje ešte tieto položky:

- ″eclipse:customEditor″ : ″org.drools.eclipse.flow.common.editor.editpart.work.SampleCustomEditor″,
- kategória obslužnej úlohy – ″category″ : ″Android″,
- základný handler : ″defaultHandler″ : ″com.example.iamhere.SmsHandler″,
- dokumentácia/návod na použite – „documentation“ : „index.html“,
- závislosti – ″dependencies″ : ″file: ./lib/smshandler.jar″,

5.2.2 Implementácia rozhrania WorkItemHandler – SmsHandler

Na naprogramovanie posielania SMS správy je využitá trieda SmsManager z knižnice android.telephony. Táto trieda spravuje SMS operácie ako zasielanie dát a textu. Podporuje GSM (Global System for Mobiles) aj CDMA (Code Division Multiple Access) siete.

5.2.2.1 Posielanie SMS správy s aktuálnou polohou

Trieda SmsHandler implementujúca rozhranie WorkItemManager v metóde executeWorkItem() najprv získa hodnoty z procesných premenných latitude a longitude a na základe týchto hodnôt zostaví správu pre systém v tvare „latitude longitude“. ˇˇDalej sa vytvorí manažér triedy SmsManager, pomocou ktorého je správa najprv rozdelená do častí s maximálnou dĺžkou na odosielanie SMS správy a následne odoslaná na preddefinované telefónne číslo metódou sendMultipartTextMessage(„defaultPhoneNumber“, null, parts, null, null) zavolanou na sms manažérovi. Správa je delená na časti z dôvodu zaručenia funkčnosti aj pri dlhších prednadstavených správach, hlavne pokiaľ je časťou správy hodnota niektorej premennej. Nakoniec je potvrdené dokončenie obslužnej úlohy.

Táto obslužná úloha je príkladom, ako zasielať automatizované SMS správy na preddefinované číslo bez umožnenia nadstavenia textu a čísla inak, ako v návrhu. Takýto prístup je výhodný pre uvedený ukážkový biznis proces doručovania balíka kuriérom, pretože nie je želané, aby kuriér mohol manipulovať obsahom správy alebo jej príjemcom.

5.2.2.2 Použitie v konkrétnej aplikácii

Pre prístup k zasielaniu SMS správ Android vyžaduje povolenie uvedené v súbore AndroidManifest.xml:

Odporúčam telefónne číslo uvádzať v tvare +(predvoľba štátu)xxx xxx xxx, z dôvodu fungovania v zahraničí.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

5.3 Obslužná úloha Add event

Táto obslužná úloha pridá do kalendára naplánovanú schôdzku.

5.3.1 Konfiguračný súbor

Konfiguračný súbor je tejto úlohy napísaný v jazyku MVEL a má nasledovné položky:

- názov – ″name″ : ″Add event″,
- stručný popis – ″description″ : ″Adding scheduled event to calendar″,
- vstupné parametre typu String – ″event″ : new StringDataType(), ″description″: new StringDataType(), ″date″ : new StringDataType(), ″time″ : new StringDataType(), ″place″ : new StringDataType(),
- zobrazovaný názov – ″display name″ : ″Add event″,
- zobrazovanú ikonu – ″icon″ : ″calendar.gif″

Konfiguračný súbor obslužnej úlohy pre začlenenie do repozitára obsahuje ešte tieto položky:

- ″eclipse:customEditor″ : ″org.drools.eclipse.flow.common.editor.editpart.work.SampleCustomEditor″,
- kategória obslužnej úlohy – ″category″ : ″Android″,
- základný handler : ″defaultHandler″ : ″com.examlple.addevent.AddEventHandler″,
- dokumentácia/návod na použite – ″documentation″ : ″index.html″,
- závislosti – ″dependencies″ : ″file: ./lib/addeventhandler.jar″,

5.3.2 Implementácia rozhrania WorkItemHandler – AddEventHandler

Od verzie Androidu 4.0 je podporovaná práca s kalendárom za pomoci Intentov, čo sú jednoduché objekty, ktoré umožňujú dosiahnutie určitého cieľa len popísaním zámeru. Pri práci s kalendárom je potrebné mať vytvorené aspoň jedno synchronizované konto.

5.3.2.1 Vytváranie novej udalosti v kalendári

Po spustení metódy executeWorkItemHandler() sú získané vstupné parametre z procesných premenných event, description, date, time a place typu String, ktoré obsahujú základné informácie o plánovanej udalosti. Ďalej je vytvorená inštancia triedy Calendar, na ktorej je zavolaná metóda set(), ktorej parametrami sú rok, mesiac, deň, hodina a minúta. Výsledná hodnota tejto metódy je počet milisekúnd od začiatku epochy po zadaný moment.

Vytvorí sa Intent, ktorý má parameter ACTION_INSERT, na základe čoho Android vie, že vykonávaná aktivita bude vkladanie. Nadstavené dáta Events.CONTENT_URI obsahujú URL pre interakciu s udalosťami. Následne sa nadstavia parametre vytváranej udalosti a spustí sa aktivita nadefinovaná Intentom. Nakoniec sa pošle notifikácia o ukončení obslužnej úlohy.

```
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
```

5.3.2.2 Použitie v konkrétnej aplikácii

Na zapisovanie dát do kalendára je potrebné uviesť povolenie do súboru AndroidManifest.xlm.

Pri využití tejto obslužnej úlohy je potrebné mať nadstavené aspoň jedno konto kalendára a minimálnu verziu Androidu 4.0. Formát dátumu musí byť v tvare RRRR/MM/DD a formát času HH:MM.

5.4 Repozitár obslužných úloh

Repozitár obslužných úloh umožňuje importovať do pracovného priestoru, ktoré by mohli byť potrebné v modelovanom procese. Importované obslužné úlohy sú automaticky pridané

do palety a môžu sa začať používať v procesoch. Je možné importovať ďalšie artefakty, ako napríklad ikony, akékoľvek potrebné závislosti, predvolený handler, ktorý bude použitý na spustenie služby. Importované úlohy môžu byť následne upravené podľa potrieb procesu.

Koreňový adresár musí okrem adresárov jednotlivých úloh obsahovať súbor index.conf, ktorý odkazuje na všetky repozitáre, ktoré by mali byť spracované pri hľadaní obslužných úloh. V adresároch jednotlivých obslužných úloh sa nachádzajú tieto súbory:

- rozšírený konfiguračný súbor s rovnakým názvom ako adresár
- ikona, na ktorú sa odkazuje konfiguračný súbor
- dokumentácia, na ktorú sa odkazuje konfiguračný súbor
- závislosti, ktoré sú uvedené v konfiguračnom súbore, napr. v podadresári lib[10]

Všetky vytvorené obslužné úlohy v elektronickej prílohe sú upravené podľa potrieb na vloženie do repozitára.

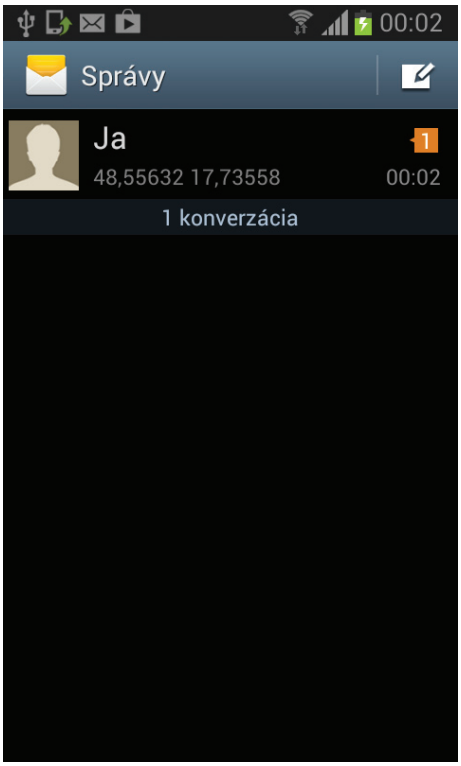
6 Testovanie funkčnosti obslužných úloh

Na otestovanie funkčnosti implementovaných obslužných úloh boli vytvorené jednoduché procesy nadefinované v Jave a spustené na reálnom zariadení aj na emulátori.

6.1 Testovanie funkčnosti úloh GPS location a SMS location

Tieto obslužné úlohy boli otestované v jednoduchom procese, ktorý využíva len tieto dve obslužné úlohy. Testy boli vykonané na emulátoroch s Androidom verzie 4.2.2 a reálnych zariadeniach s Androidom verzie 4.1.2 a 2.3.6. vo všetkých troch prípadoch aplikácia prebehla podľa očakávania a poslala SMS správu s aktuálnou polohou na zadané telefónne číslo. Pri testoch s emulátormi boli mechanicky nadstavené koordináty a prednadstavené číslo sa zhodovalo s číslom druhého emulátora.



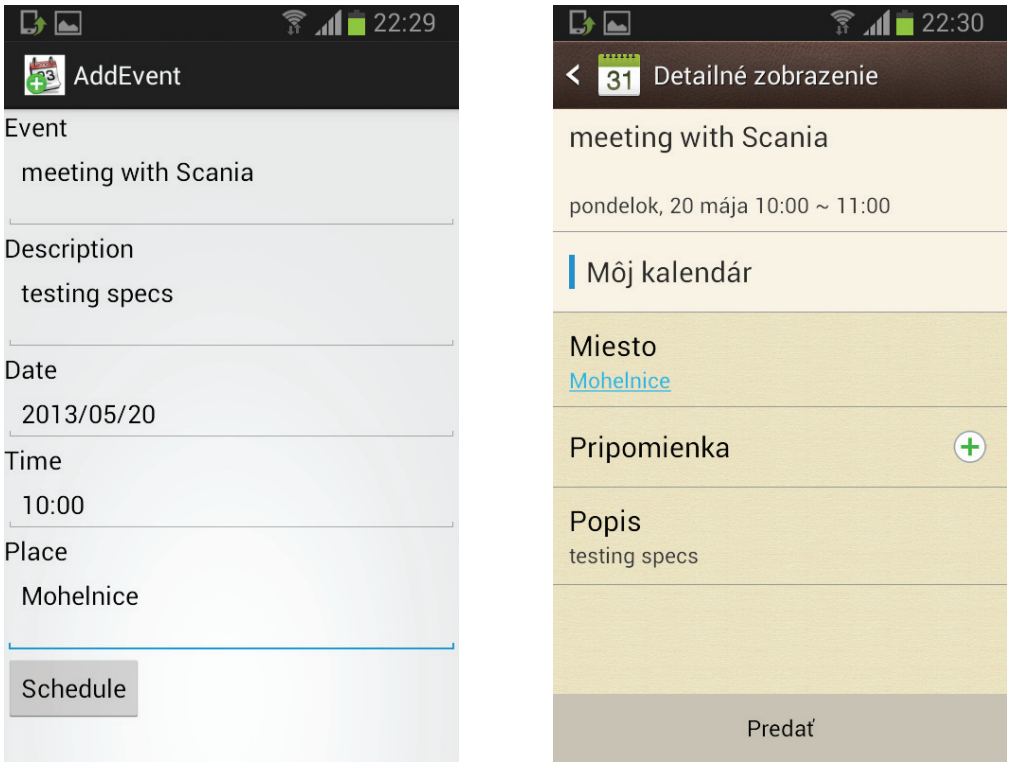


Obrázok 6 (vľavo): Importovanie testovacej aplikácie Send Location

Obrázok 7 (vpravo): Prijatá správa s aktuálnou polohou zariadenia

6.2 Testovanie funkčnosti úlohy Add Event

Pre potreby testovania tejto úlohy bola vytvorená ešte jedna úloha, ktorá údaje zadané do formulára posunie na spracovanie úlohe Add Event. Úloha Add Event bola otestovaná len na reálnom zariadení s Androidom verzie 4.1.2. Testovanie na zariadení zo staršou platformou nebolo možné, pretože práca s kalendárom pomocou Intentou je dostupná až od verzie 4.0. Testovanie na emulátori nebolo možné z dôvodu nutnosti synchronizácie kalendára. Vytvorená aplikácia podľa očakávania otvorila kalendár s predvyplnenými informáciami o novom stretnutí.



Obrázok 8 (vľavo): Zadávanie údajov, ktoré sú spracované obslužnou úlohou Add Event

Obrázok 9 (vpravo): Vytvorená udalosť v kalendári

Záver

Počas písania tejto práce som sa zoznámila s problematikou tvorby obchodných procesov, špecifikáciou BPMN2 a s nástrojom jBPM5, ktorý je používaný na modelovanie, správu a monitorovanie obchodných procesov. Hlbšie som preskúmala princípy a spôsob vytvárania obslužných úloh. Druhou časťou môjho prieskumu boli mobilné platformy, konkrétne platforma Android. Dozvedela som sa, aká je architektúra Androidu a ako možno využiť jeho knižnice na tvorbu aplikácií. Nadobudnuté znalosti som v praktickej časti demonštrovala vytvorením troch obslužných úloh, ktoré využívajú platformu Android. Ako demonštráciu praktického využitia týchto úloh som vytvorila dva ukážkové biznis procesy.

Implementácia mnou vytvorených úloh bola zverejnená na komunitnom fóre jBPM, aby ich mohol využívať ktokoľvek vo svojich procesoch. Ďalším smerom rozvoja môže byť vytvorenie repozitára s obslužnými úlohami pre platformu Android, ktoré by pokryli ďalšiu funkcionalitu. Iným smerom sa môže vybrať vývoj nástrojov, ktoré umožnia používateľom mobilnej platformy Android modelovať, spravovať a spúšťať svoje procesy priamo v mobilnom zariadení. V každom prípade vidím budúcnosť práce s biznis procesmi na platforme Android s využitím nástroja jBPM veľmi pozitívne.

Literatúra

[1] Business process management (BPM).Business Dictionary. [online]. [cit. 2013-05-19]. Dostupné z: http://www.businessdictionary.com/definition/business-process-management-BPM.html

[2] SALATINO, Mauricio a Esteban ALIVERTI. JBPM5 Developer Guide. Packt Publishing Ltd., 2012, 16. ISBN 978-1-84951-644-0.

[3] What is BPMN?. [online]. [cit. 2013-05-19]. Dostupné z: https://research.linagora.com/pages/viewpage.action?pageId=3050157#

[4] Bonitasoft.com. [online]. [cit. 2013-05-19]. Dostupné z: http://www.bonitasoft.com/company/bonitasoft-company-overviewefinition/business-process-management-BPM.html

[5] Business Process Management (BPM) Market Shares, Strategies, and Forecasts, Worldwide, 212 to 2018. BOMBOURG, Nicolas. [online]. [cit. 2013-05-19]. Dostupné z: http://www.reportlinker.com/p0963519-summary/Business-Process-Management-BPM-Market-Shares-Strategies-and-Forecasts-Worldwide-to.html

[6] SALATINO, Mauricio a Esteban ALIVERTI. JBPM5 Developer Guide. Packt Publishing Ltd., 2012, 176 – 178, 194. ISBN 978-1-84951-644-0.

[7] ABI RESEARCH. Nokia Leading Smartphone Market with 56%, While Symbian's Share of OS Market Is Set to Fall [online]. Londýn, 2007 [cit. 2013-05-19]. Dostupné z: http://web.archive.org/web/20120329111955/http://www.abiresearch.com/press/826

[8] MEIER, Reto. Professional Android™ Application Development. Indianapolis: Wiley Publishing, Inc., 2009, 13. ISBN 978-0-470-34471-2.

[9] MEIER, Reto. Professional Android™ Application Development. Indianapolis: Wiley Publishing, Inc., 2009, 6 – 7. ISBN 978-0-470-34471-2.

[10] The AndroidManifest.xml File. Android [online]. [cit. 2013-05-19]. Dostupné z: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[11] Service repository. jBPM User Guide: Version 5.3.0 Final [online]. [cit. 2013-05-19]. Dostupné z: http://docs.jboss.org/jbpm/v5.3/userguide/ch.domain-specific-processes.html#d0e7209

Elektronicé prílohy

Súčasťou práce sú tieto elektronické prílohy:

Priečinok Tasks v ktorom sa nachádzajú adresáre so súbormi jednotlivých úloh. Ku každej obslužnej úlohe je vytvorený konfiguračný súbor, pridsná ikona, súbor index.html so stručným popisom obslužnej úlohy a .jar súbor s implementáciou triedy WokrItemHandler v súboroch .java a class.

Testovacie aplikácie test_event a test_sms a taktiež zdrojové súbory týchto aplikácií, ktoré boli použité na otestovanie funkčnosti obslužných úloh.

Obrázky vytvorené v súvislosti s touto prácou, konkrétne diagram ukážkového procesu, proces doručenia balíka, proces, plánovania stretnutie a výstupné obrazovky z testovaní.