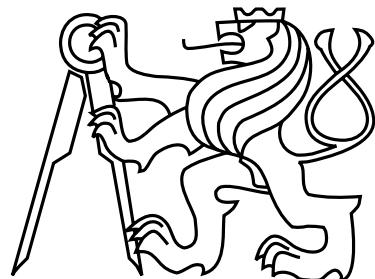


České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

**Mobilní část systému pro evidenci dopravních přestupků a
pohybu policistů**

Martin Štajner

Vedoucí práce: Ing. Martin Komárek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

13. května 2012

Poděkování

Chtěl bych poděkovat svému vedoucímu práce panu Ing. Martinovi Komárkovi za pozitivní přístup, rady a pomoc při tvorbě bakalářské práce. Dále bych chtěl také poděkovat kolegovi Pavlovi Brožovi za ochotnou a příjemnou spolupráci během vývoje.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 13. 5. 2012

.....

Abstract

SMART-FINE project deals with creation of a client (mobile applications) for record keeping of traffic offenses through mobile phones. This application should simplify the work of police officers with issuing parking tickets and controlling car parking. Elimination of paper pads, options of selecting frequent values and automatic pre-filling will accelerate the work and the availability of information in digital form will simplify the subsequent administration. This paper deals with analyzing the problem, designing the application, the actual development of the application and testing it. Java programming language was used for the development and the target devices are phones running Android. Application interacts with the server part of the system, which is dealt with by Pavel Brož in his thesis [12].

Abstrakt

Práce se zabývá tvorbou klienta (mobilní aplikace) systému pro evidenci dopravních přestupek pomocí mobilních telefonů. Ten by měl usnadnit policistům práci s udělováním parkovacích lístků a kontrolou parkování. Eliminace papírových bločků, možnosti rychlých voleb a automatické před-vyplňování práci urychlí a dostupnost informací v digitální podobě usnadní následnou administrativu.

Práce se zabývá analýzou problému, návrhem aplikace, jejím samotným vývojem a testováním. Při vývoji byl použit programovací jazyk Java, cílová zařízení jsou mobilní telefony s operačním systémem Android. Aplikace spolupracuje se serverovou částí systému, kterou se zabývá ve své bakalářské práci kolega Pavel Brož [12].

Obsah

1	Úvod	1
1.1	Problém a cíl projektu	1
1.2	Motivace	1
1.3	Podobné projekty	1
1.4	Struktura práce	2
2	Iterace 1	3
2.1	Úvod, cíle iterace	3
2.1.1	Funkční a nefunkční požadavky	3
2.2	Vymezení projektu, problém, cíle, přínosy	4
2.3	Případy užití	4
2.4	Doménový model, datové entity v systému	7
2.4.1	Položky papírového parkovacího lístku	7
2.5	Platforma, vývojové prostředí	8
2.6	Uživatelské prostředí	9
2.7	Implementace	10
2.7.1	Tvorba parkovacích lístků	10
2.7.2	Seznam a náhled parkovacích lístků	11
2.7.3	Struktura aplikace, seznam balíčků	11
2.8	Testování	13
2.8.1	Heuristická evaluace	14
2.8.2	Uživatelské testování	15
2.9	Závěr	16
2.9.1	Splněné funkční požadavky:	16
2.9.2	Nesplněné funkční požadavky:	17
3	Iterace 2	19
3.1	Úvod, cíle iterace	19
3.2	Případy a scénáře užití	19
3.3	Uživatelské prostředí	20
3.4	Implementace	21
3.4.1	Pořízení fotodokumentace	21
3.4.2	Určení aktuální adresy	21
3.5	Automatické rozpoznávání SPZ z fotografie	22
3.6	Závěr	22

3.6.1	Splněné funkční požadavky:	23
3.6.2	Nesplněné funkční požadavky:	23
4 Iterace 3		25
4.1	Úvod, cíle iterace	25
4.2	Případy a scénáře užití	25
4.3	Uživatelské prostředí	30
4.4	Implementace	31
4.4.1	Oblíbené položky v návodě	35
4.4.2	Komunikace se serverem	35
4.4.3	Kontrola SMS parkování, kontrola parkovací karty, tisk parkovacích lístků	36
4.4.4	Přihlašování	36
4.4.5	Sledování aktuální polohy	37
4.4.6	Tisk parkovacího lístku	38
4.5	Testování	39
4.6	Závěr	39
4.6.1	Splněné funkční požadavky:	39
5 Závěr		41
5.1	Zhodnocení provedené práce	41
5.2	Osobní zhodnocení	41
5.3	Budoucí vývoj	42
A Přehled o projektu		45
A.1	Požadavky na systém	45
A.1.1	Obecné požadavky	45
A.1.2	Funkční požadavky	45
A.2	Spolehlivost	48
A.3	Možné překážky a rizika	48
A.3.1	Při vývoji	48
A.3.2	Při nasazení	48
B Analýza		49
B.1	Případy užití	49
B.2	Scénáře případů užití	50
B.3	Doménový model	53
B.3.1	Parkovací lístek - ParkingTicket	53
B.3.2	Odkaz na zákon - Law	53
B.3.3	Častá hodnota - FrequentValue	54
B.3.4	Geolokační bod - Waypoint	54
C Návrh		55
D Seznam použitých zkratek		57
E Instalační a uživatelská příručka		59

Seznam obrázků

2.1	Model případů užití	5
2.2	Model závislostí požadavků na případech užití	6
2.3	Doménový model	8
2.4	První návrh podoby obrazovek	9
2.5	Balíček tříd - smartfine	12
2.6	Balíček tříd - dao	13
2.7	Balíček tříd - model	13
3.1	Změněné a přidané obrazovky	21
4.1	Model případů užití	28
4.2	Model závislostí požadavků na případech užití	29
4.3	Obrazovky přihlášení	30
4.4	Obrazovky s přidaným action barem	31
4.5	Obrazovky výběru oblíbených položek	31
4.6	Balíček tříd - model	32
4.7	Balíček tříd - android	33
4.8	Balíček tříd - dao	34
B.1	Doménový model	53

Kapitola 1

Úvod

1.1 Problém a cíl projektu

Kdykoliv v současnosti dojde k přestupku při parkování, policista na místě vyplňuje parkovací lístek v podobě papírového formuláře. Musí si pamatovat čísla paragrafů zákona, zjišťovat adresu, kde se událost odehrála, několikrát opisovat různé informace dokola atd. Papírové formuláře se mohou kdykoliv ztratit, jsou nepraktické na přenášení a zvyšují administrativu. Kontrola parkování se také může v dnešní době stát obtížnější, vzhledem k tomu, že lze platit pomocí SMS. Policista si na místě nemůže být okamžitě jistý, zda má vozidlo parkování povolené. Na některých místech, označených jako modrá zóna lze v Praze parkovat, pokud řidič vlastní speciální parkovací kartu. Některé tyto karty jsou přenosné, neobsahují tedy SPZ vozidla, a pokud dojde k jejímu odcizení, může pachatel kartu bez problému použít. Policista nemůže hned na místě ověřit, zda se zrovna nejedná o odcizenou kartu.

Cílem projektu je vytvoření systému pro mobilní zařízení (mobilní telefon), které bude usnadňovat kontrolu parkování zaplaceného pomocí SMS, bude nahrazovat manuální psaní parkovacích lístků a bude umožňovat ověření, zda parkovací karta nebyla odcizena a není tedy používána neprávem.

Více v kapitole 2.

1.2 Motivace

Téma „Evidence dopravních prostředků a pohybu policistů pomocí mobilních telefonů“ jsem si vybral ze seznamu nabízených témat pro předmět A7B36SI2 – Řízení softwarových projektů [1], kde jsme na projektu pracovali v týmu. Téma nebylo před tím zpracováno jiným týmem, na který bychom museli navazovat, tudíž jsme začínali úplně od začátku. Po ukončení tohoto předmětu jsem se rozhodl aplikaci dále vyvíjet v rámci bakalářské práce, neboť shledávám téma jako zajímavé.

1.3 Podobné projekty

V České republice i v zahraničí existují systémy, částečně řešící problematiku, kterou se projekt Smart-Fine zabývá. V nalezených případech se ale vždy jedná pouze o určitou funk-

cionalitu, nikdy nejde o celistvé řešení všech výše popsaných problémů, ze kterého by se dalo výrazně čerpat.

V České republice se SMS parkovným se komerčně zabývá např. firma Erika a.s. [11]. Nabízí službu kontroly parkování zaplaceného přes SMS řešenou online přístupem k aplikaci skrze mobilní zařízení, kde pracovníci kontroly zadají SPZ vozidla.

V zahraničí např. firma Zebra Technologies [19], která se zabývá distribucí nejrůznějších druhů mobilních tiskáren, nabízí i možnost tisku z mobilních zařízení. Ve jejich ukázkové aplikaci, která demonstruje bezdrátový tisk z mobilních telefonů, je možnost tisku jakéhosi pokutového lístku. Toto řešení má ovšem poměrně daleko k opravdovému nasazení, lze se však inspirovat.

1.4 Struktura práce

Bakalářská práce je psána obdobou iterativního způsobu [14]. Každá iterace má určité cíle, díky jimž dochází ke změnám v systému a dokumentaci. Může se jednat o úpravu čehokoliv, co již bylo předtím v projektu provedeno, přidávání nových cílů, nové funkčnosti systému, nebo dokončování cílů nesplněných v předešlých iteracích. Při přidávání nové funkčnosti se znova provede analýza, návrh, implementace a testování daných změn. Většina kapitol tedy obsahuje popis a důvod všech provedených změn v projektu. Výsledkem spojení výstupů všech iterací dohromady pak vzniká finální projekt, tato bakalářská práce.

I přes jasná pravidla vývoje iterativním způsobem, docházelo bohužel při psaní k jejich porušování. Hlavním problémem bylo ukončování iterací uprostřed vývoje nějaké funkčnosti. To znamená, že daný funkční požadavek byl splněn pouze z části, a dokončen v iteraci další. Způsob vývoje se také liší od klasického iterativního způsobu tím, že v případě projektu Smart-fine je většina požadavků na funkcionalitu stanovena hned na začátku, a samotné iterace tedy spíše představují jakési časové úseky. Ty popisují průběh vývoje projektu a stanovují, co přesně bylo za daný časový úsek splněno, a co zůstalo do dalšího vývoje.

Veškeré diagramy jsou modelovány v Enterprise Architect [5] pomocí jazyka UML.

Kapitola 2

Iterace 1

2.1 Úvod, cíle iterace

Text této kapitoly se zabývá veškerou prací udělanou v rámci předmětu SI2 [1], který trval jeden semestr. Právě toto období jednoho semestru představuje 1. iteraci. Jak již bylo zmíněno v úvodní kapitole, na projektu se začalo pracovat úplně od začátku, neboť se doposud tímto tématem nikdo jiný nezabýval.

Jelikož se jedná o 1. iteraci, jejím cílem je hlavně vymezení samotného projektu a jeho zahájení. Po vymezení projektu a ujasnění si, čím se má vlastně zabývat je dalším důležitým cílem stanovení funkčních a nefunkčních požadavků na systém - sepsání jednotlivých funkcí, kterými má systém disponovat a co má vlastně umět.

2.1.1 Funkční a nefunkční požadavky

Sběr požadavků byl bezproblémový, bylo na první pohled jasné, co by asi aplikace tohoto typu měla zvládat a co za funkce nabízet. Detailnější popis funkčních a nefunkčních požadavků představovaný vypsáním „pod-požadavků“ je v příloze [A]. Systém bude umožňovat/poskytovat:

F.Req.1.: Vyplnění parkovacího lístku

F.Req.2.: Ná pověda při vyplňování čísel paragrafů zákonů nejběžnějších přestupků

F.Req.3.: Prohlížení lokálně uložených parkovacích lístků

F.Req.4.: Úprava lokálně uložených parkovacích lístků

F.Req.5.: Odeslání parkovacích lístků z mobilního zařízení na server

F.Req.6.: Tisk parkovacích lístků pomocí mobilní tiskárny

F.Req.7.: Pořízení fotodokumentace přestupku

F.Req.8.: Rozpoznání státní poznávací značky z fotografie

F.Req.9.: Automatické vyhledání adresy pomocí GPS

F.Req.10.: Kontrola odcizení přenosné parkovací karty opravňující parkovat v modré zóně

F.Req.11.: Přístup k funkcím aplikace po zadání identifikačního čísla

Základní předpoklady pro plné využití a správný chod aplikace (nefunkční požadavky) jsou:

N.Req.1.: Mobilní zařízení (mobilní telefon)

N.Req.2.: Mobilní tiskárna s datovým připojením přes Bluetooth

N.Req.3.: Přístup k serveru Smart-Fine

V rámci iterace by také mělo dojít k analýze a implementaci co nejvíce požadavků, počínaje těmi nejzákladnějšími. Zaměřena bude tedy hlavně na tvorbu parkovacích lístků (F.Req.1) včetně předvyplňování a náповědy v podobě nejběžnějších voleb (F.Req.2), přidávání fotodokumentace (F.Req.7), prohlížení vytvořených lístků a jejich editaci (F.Req.3, F.Req.4). Cílem iterace je také testování hotové implementované funkcionality.

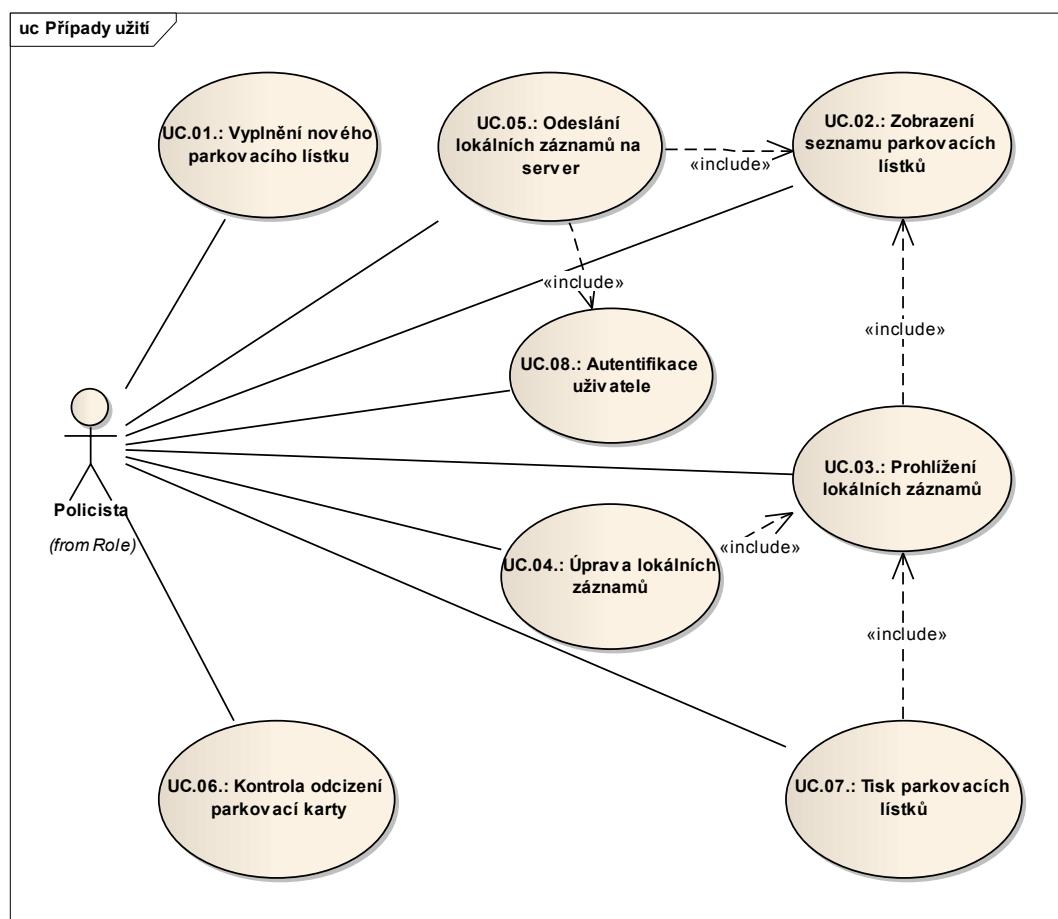
2.2 Vymezení projektu, problém, cíle, přínosy

Jak již bylo popsáno v 1. kapitole, systém Smart-Fine se zabývá problematikou kontroly parkování a vypisování parkovacích lístků. Místo ručního vypisování se informace budou zapisovat do mobilní aplikace, s možností archivace a zálohy na serveru. Tím se ulehčí práce s papírováním, opisováním věcí a celkově administrací. Aplikace si může za uživatele pamatoval, nebo zjišťovat informace a předvyplňovat je. Např. automatické vyplnění času, aktuální polohy v podobě adresy atd. Další urychlení při vyplnění parkovacího lístku spočívá v použití nabídky nejčastějších voleb. Mobilní telefon v dnešní době většinou dokáže fotografovat, situaci lze tím pádem zdokumentovat i takovýmto způsobem. Všechny informace se odešlou a uloží na server, odkud budou snadno získatelné k další práci. Pomocí adres uložených v parkovacích lístcích bude možné na serveru sledovat pohyb policistů. Aplikace bude také umožňovat okamžitou kontrolu parkování zaplaceného přes SMS ověřením SPZ vozidla, a ve stejném duchu i kontrolu přenosné parkovací karty opravňující parkovat v modré zóně v Praze. K ovládání nebude potřeba žádných zvláštních dovedností, jen lehké zaškolení v ovládání, vše by mělo být intuitivní. Práce se systémem tak bude jednoduchá a efektivní zároveň.

Možné překážky vývoje jsou v příloze [A].

2.3 Případy užití

Po vypsání požadavků na systém v úvodu iterace došlo k určení tzv. případů užití [10]. Jde o definování, jakým způsobem bude systém používán a jak s ním budou uživatelé (policisté) manipulovat. Detailnější popis případů užití je v příloze [B].



Obrázek 2.1: Model případů užití

UC.1.: Vyplnění nového parkovacího lístku

UC.2.: Zobrazení seznamu parkovacích lístků

UC.3.: Prohlížení lokálních záznamů

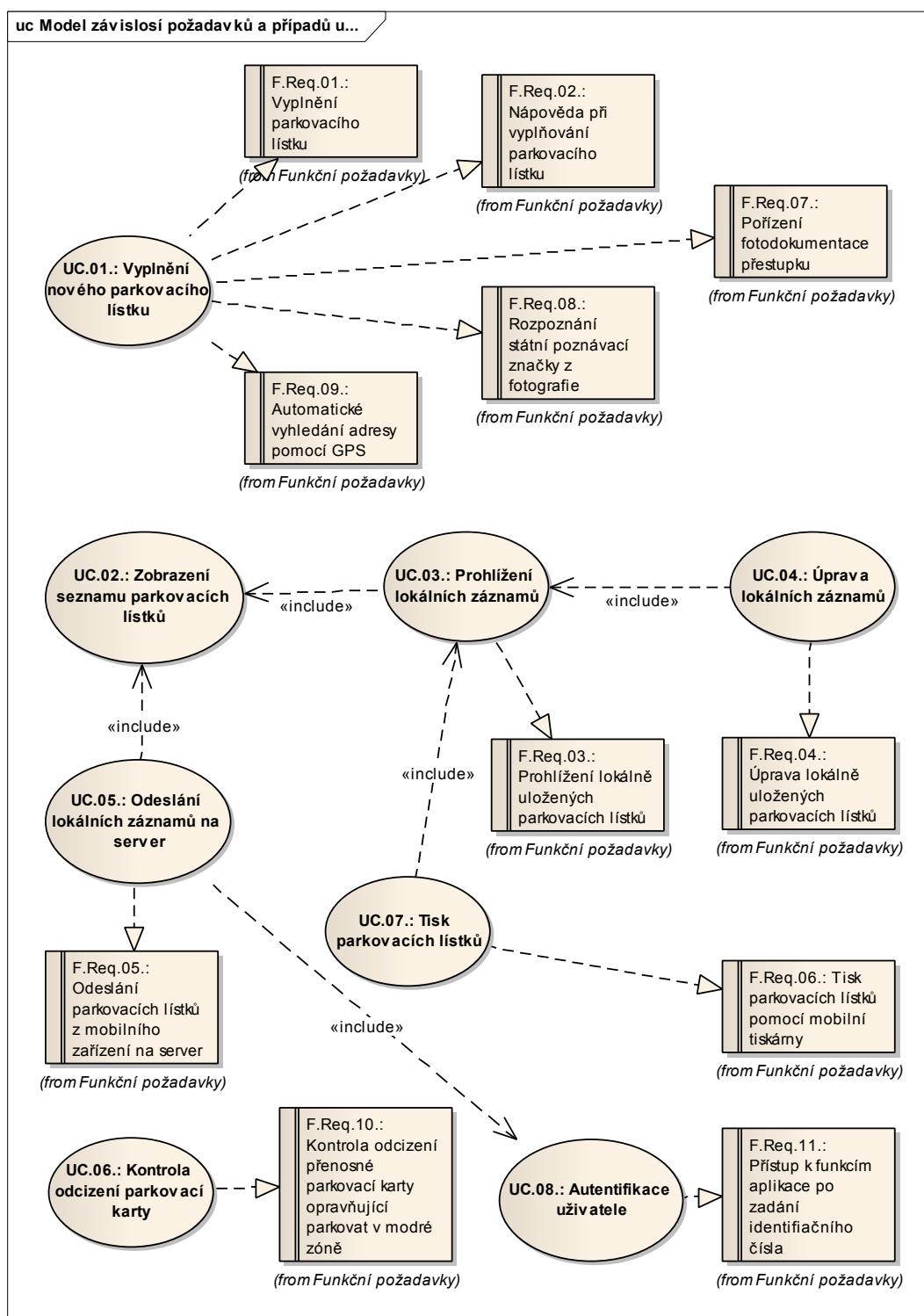
UC.4.: Úprava lokálních záznamů

UC.5.: Odeslání lokálních záznamů na server

UC.6.: Kontrola odcizení parkovací karty

UC.7.: Tisk parkovacích lístků

UC.8.: Autentifikace uživatele



Obrázek 2.2: Model závislostí požadavků na případech užití

Případy užití lze znázornit diagramem 2.1, který ukazuje uživatelské role, a souvislosti mezi nimi a případy užití. Uživatelská role je v tomto případě pouze jedna, policista v terénu, což znamená, že všechny případy užití, veškerou manipulaci s aplikací, bude prováděn pouze on.

S případy užití souvisí tzv. scénáře užití popisující posloupnost kroků, které musí být vykonány za účelem provedení dané činnosti. Scénář zahrnuje jak kroky uživatele, tak kroky systému. Jejich seznam je vypsán a jednotlivé scénáře popsány v příloze [B].

Požadavky na systém a scénáře užití spolu také úzce souvisí. Každý požadavek musí být zahrnut v nějakém případu užití, aby došlo k jeho realizaci. Jde vlastně říci, že případy užití využívají funkční požadavky, neboli pokud bude chtít uživatel provést určitou činnost, tak aplikace musí obsahovat příslušnou funkčnost. Diagram 2.2 ukazuje mapování funkčních požadavků na jednotlivé případy užití.

2.4 Doménový model, datové entity v systému

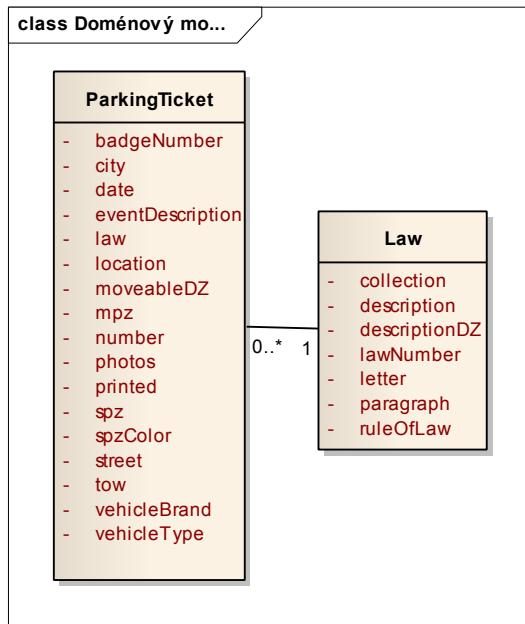
Doménový model je konceptuální model, který popisuje různé entity, datové domény, jejich atributy a vztahy mezi nimi. V systému jsou zatím pouze 2 hlavní datové domény, se kterými je potřeba pracovat. Jedná se o entity, které v sobě ukládají nějaká data. Jsou jimi parkovací lístek (parking ticket) a odkaz na zákon (law). Každý parkovací lístek musí obsahovat právě 1 odkaz na příslušný zákon. Další informace, které charakterizují parkovací lístek, jsou vypsány v podobě atributů objektu. Např. parkovací lístek musí obsahovat datum, adresu, informace o vozidle apod. Odkaz na zákon musí mít např. číslo zákona, paragraf atd. Při zjišťování a sepisování správných atributů se vycházelo z parkovacího lístku, který momentálně používá policie města Prahy (k dispozici na CD).

2.4.1 Položky papírového parkovacího lístku

- Datum a čas
- SPZ – státní poznávací značka
- MPZ – mezinárodní poznávací značka
- Barva SPZ – např.: bílá, oranžová
- Druh vozidla - např.: osobní, nákladní
- Tovární značka vozidla - např.: Škoda, Ford
- Adresa - město, ulice, č. popisné
- Místo - určení místa, kde došlo k přestupku, např.: číslo lampy veřejného osvětlení
- Popis jednání (část pro policistu) - kód přestupku
- Označení, zda je přítomná fotodokumentace
- Označení, zda byl přestupek na pohyblivém dopravním značení

- Identifikační číslo policisty
- Popis jednání (část pro řidiče) – popis dopravního přestupku
- Podpis policisty

Diagram 2.3 znázorňuje jednoduchý doménový model. Ten obsahuje objekty daných entit a popisuje jejich vztahy. Objekty jsou reprezentovány třídami s atributy, prozatím bez datových typů.



Obrázek 2.3: Doménový model

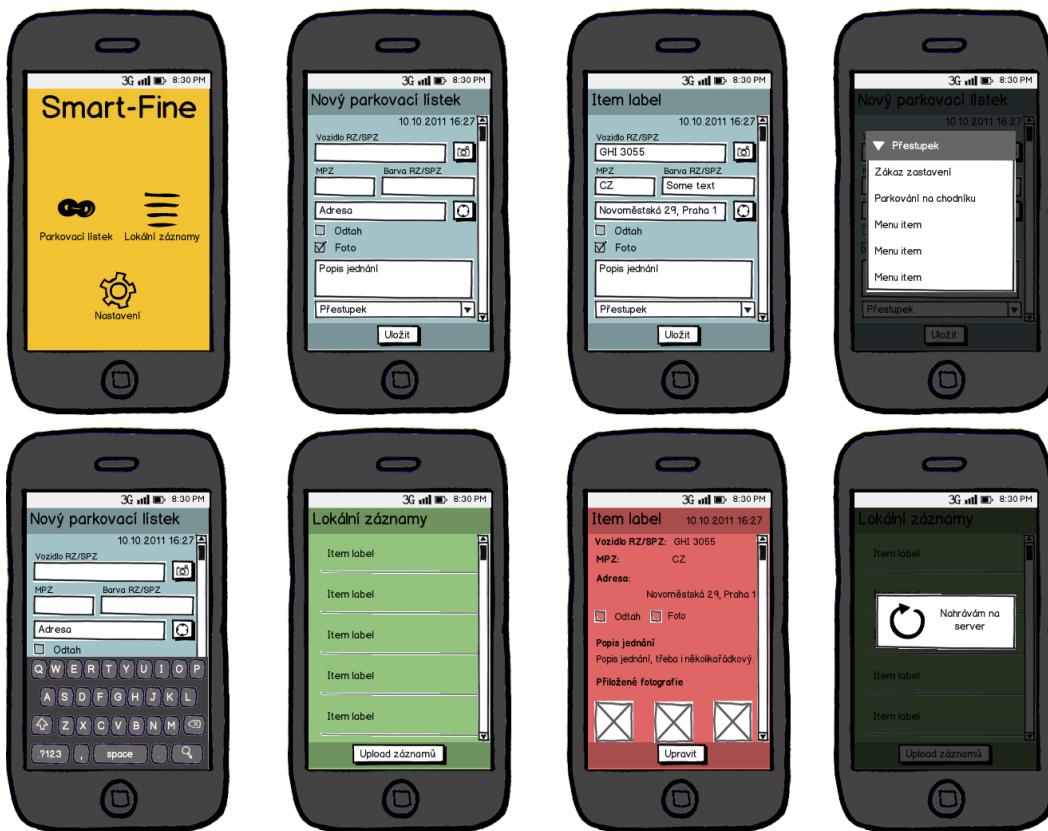
Seznam a význam všech atributů jednotlivých objektů je vypsán v příloze [B].

2.5 Platforma, vývojové prostředí

Při výběru cílové platformy mobilních zařízení byl kladen důraz na velikou rozšířenost. Z toho důvodu byl vybrán Android OS, který se v posledních málo letech velmi rozrostl. Minimální verzí pro tento projekt byla zvolena verze 2.1. Aplikace pro tento operační systém jsou vyvíjeny v jazyku Java a pomocí speciálních knihoven Android SDK [2], a tak tomu tak je i v tomto projektu. Jako vývojové prostředí bylo použito IDE Eclipse [6], které přímo podporuje programování pro Android. Testovat lze kromě vlastního mobilního zařízení také na vestavěném emulátoru, který je součástí Android SDK. Na emulátoru však nejsou přístupné všechny funkce, z toho důvodu je doporučeno testovat software přímo na mobilním zařízení. Při vývoji byl k dispozici mobilní telefon HTC Desire, vývoj i testování tedy probíhaly na něm.

2.6 Uživatelské prostředí

Návrh uživatelského prostředí byl vytvořen pomocí online nástroje Balsamiq Mockup [3]. Byly navrženy jednotlivé obrazovky, jak by asi mohly vypadat ve finální aplikaci. Jde pouze o orientační znázornění, naznačuje spíše strukturu, neboť tento nástroj zdaleka neumí vyjádřit pravý design aplikace. Návrh je včetně popisů jednotlivých obrazovek k dispozici na CD. Obrázek 2.4 ukazuje obrazovky hlavního menu, vytváření nového parkovacího lístku, seznamu uložených parkovacích lístků, zobrazení detailu již vyplněného parkovacího lístku a obrazovky odesílání lístků na server.



Obrázek 2.4: První návrh podoby obrazovek

Jak je vidět z obrázku, hlavní menu obsahuje pouze navigaci po aplikaci v podobě tlačítek. Vytváření nového parkovacího lístku je prakticky pouhé vyplnění několika formulářů. To lze usnadnit výběrem položky z nabídky nejčastějších voleb, která se vyvolá stiskem postranních tlačítek, nebo použitím automatického před-vyplnění. Celý lístek se uloží stiskem tlačítka ve spodní části obrazovky. Seznam záznamů obsahuje všechny neodeslané parkovací lístky. Po kliknutí na položku se zobrazí její detailní. Parkovací lístek se dá vytisknout, upravit, či smazat stiskem tlačítka ve spodní části obrazovky. Všechny záznamy lze naráz nahrát na server pomocí stisku tlačítka ve spodní části na obrazovce seznamu všech lístků.

Z hotových návrhů obrazovek vznikl první prototyp (k dispozici na CD) v podobě kli-

kacího PDF souboru, který ve velmi zjednodušené podobě ukazuje a popisuje, jak by asi aplikace mohla fungovat. V prototypu není zahrnuta žádná funkčnost, opět se jedná pouze o primitivní návrh.

Jak celá aplikace vypadá po grafické stránce, je ukázáno v příloze [C].

2.7 Implementace

Struktura aplikace bude dodržuje návrhový vzor MVC (Model, View, Controller). Funkci části Controller zajišťují třídy typu Activity (dále aktivita). Ty představují jednotlivé obrazovky aplikace, každá aktivita tedy bude mít svoje grafické rozhraní GUI (View). Většina případů užití je implementována právě pomocí jedné aktivity. Většina View je pro lepší přehlednost řešena pomocí XML souborů obsahující popis grafického rozhraní, nikoliv napsaná přímo v kódu aplikace. Modelem jsou datové třídy, představující jednotlivé entity, jak bylo dříve zmíněno v sekci Doménový model.

Při vývoji je použito návrhových vzorů. Konkrétně návrhový vzor DAO (Data access object), který odstíňuje zdroje dat a umožňuje tak lehké použití různých implementací ukládání dat, např. do souboru, nebo do databáze. Implementace tříd DAO v sobě po celou dobu běhu aplikace uchovávají reference na data modelů uložená v paměti. Dalším použitým návrhovým vzorem je Singleton (Jedináček), který umožňuje znova použití určitých objektů.

2.7.1 Tvorba parkovacích lístků

Parkovací lístek se vytvoří vyplněním sady formulářových polí, které představují jednotlivé položky papírového parkovacího lístku. K vyplnění může uživatel použít nabídku nejčastějších voleb, které jsou definovány jako pole v souboru XML, odkud jdou lehce načíst do příslušných nabídkových seznamů. Při vybrání položky ze seznamu se hodnota vloží do políčka a to se zablokuje. Pokud chce uživatel vložit vlastní hodnotu, musí v seznamu vybrat položku „Vlastní“. Při stisku tlačítka „Uložit“ se nejdříve z vyplněných údajů vytvoří objekt parkingTicket, který pak projde validací. Např. SPZ vozidla by neměla mít méně než 5 znaků (5 znaků mohou mít některé americké SPZ) a více než 10 znaků. Některá políčka jsou označena jako povinná, ta musí být neprázdná. Např. SPZ a MPZ vozidla, adresa, odkaz na zákon atd. Při chybě během validace se zobrazí varování.

Součástí vytváření parkovacích lístků je také pořízení fotodokumentace. Uživateli se po stisku tlačítka pro přidání zobrazí dialog, kde si může vybrat jestli chce pořídit novou fotografi, nebo vybrat již vyfocenou z galerie. Během první iterace došlo k implementaci pouze první možnosti. V rámci „zjednodušení“ práce aplikace k pořizování fotek využívá nativní systémovou aplikaci fotoaparátu, kterou v základu obsahuje každá verze systému Android. Ta novou fotografi uloží na námi definované místo, v našem případě do rootu paměťové karty. Při implementaci zobrazení náhledu fotografií v parkovacím lístku nastal problém, kdy aplikace po pořízení 2 - 3 fotografií z ničeho nic spadla. Bylo zjištěno, že se tak děje z důvodu nedostatku operační paměti, jelikož fotografie ve své plné kvalitě zabírají kolem 2,5 mb, a aplikace nedokázala takové množství pojmit. Problém byl vyřešen překódováním obrázků na menší rozlišení (čím se zmenší i velikost) pro účely zobrazení náhledu.

Dalším problém bylo mizení náhledů fotografií a vyplněných dat ve formulářích při otočení telefonu, kdy se automaticky obrazovka přetočí ze svíslého na vodorovný způsob zobrazení. Přitom se totiž aktivita, která drží všechna data v průběhu vytváření, restartuje, aby se mohlo překreslit uživatelské prostředí. Problém byl vyřešen ukládáním dat při ničení aktivity do objektu Bundle, který je třída schopna si před restartem uložit a znova při vytváření načíst. Do objektu tohoto typu lze ukládat jednoduché datové typy, ale i serializovatelné objekty.

Pokud objekt parkovacího lístku projde validací, uloží se do seznamu k ostatním lístkům a následně se celý seznam uloží do souboru. K tomu slouží třídy DAO, o kterých byla zmínka dříve.

2.7.2 Seznam a náhled parkovacích lístků

Další části aplikace je seznam vytvořených neodeslaných parkovacích lístků. Položky (každý řádek) mají nestandardní view. To znamená, že obsahují více než jenom 1 řádek textu. V našem případě má každá položka seznamu popis dopravního přestupku, datum a čas a SPZ vozidla. K naplnění takového seznamu daty je potřeba použít objekt typu ArrayAdapter. Ten v cyklu z každého objektu parkovacího lístku vytáhne potřebné informace a zobrazí je do příslušných kolonek položky seznamu.

Při kliknutí na položku v seznamu se spustí aktivita zobrazující detail parkovacího lístku. Té se při spouštění předá index vybraného lístku. Jelikož se zobrazují stejné informace, které se vyplňují při vytváření, bylo k zobrazení použito stejně view. Zde se ale všechna formulárová pole zablokují, aby nešla upravovat.

Jak v seznamu, tak v zobrazení detailu se pro účely demonstrace a testování implementovaly dialogy ukazující stav tisku parkovacího lístku a stav nahrávání lístků na server. Ty se aktivují po stisku příslušných tlačítek. Dialog je aktivní pouze pár vteřin a automaticky oznámí úspěch operace. To je provedeno pomocí objektu typu Handler, který dokáže spustit objekt typu Runnable se zpožděním. Opravdové nahrání dat na server a tisk parkovacího lístku bude implementováno v dalších iteracích.

2.7.3 Struktura aplikace, seznam balíčků

Třídy aplikace jsou pro lepší orientaci rozděleny do jednotlivých balíčků, podle jejich účelu.

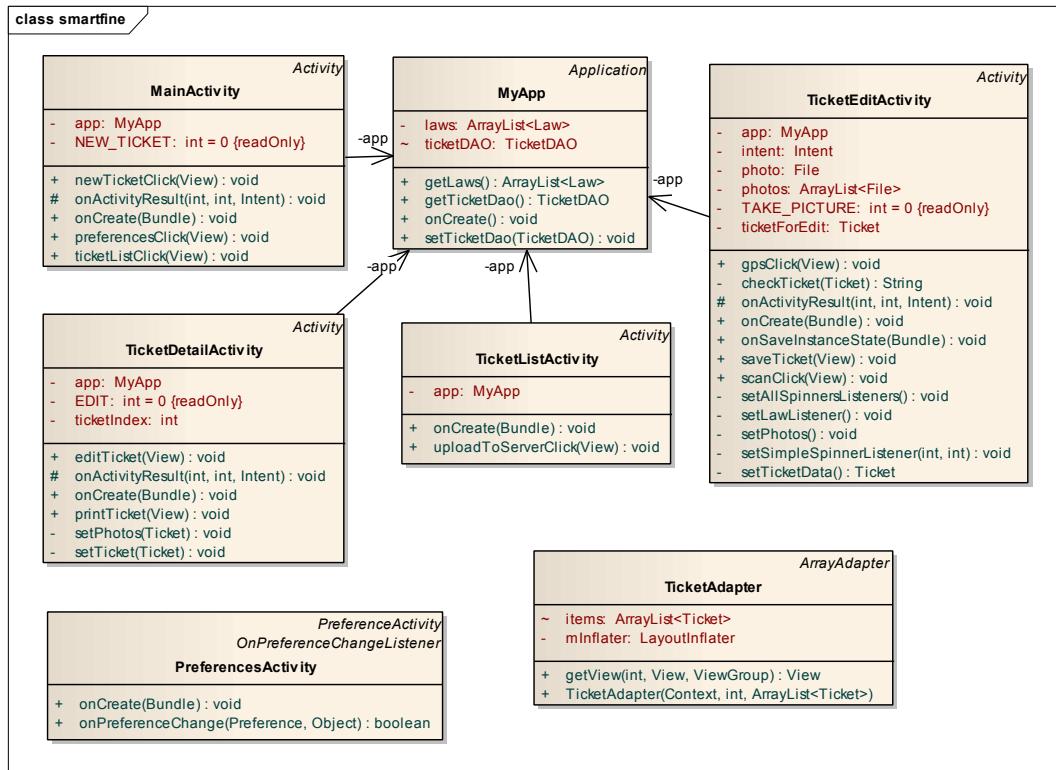
- cz.smartfine - obsahuje hlavně třídy aktivit
- dao - poskytuje přístup k souborové vrstvě a slouží jako uložiště objektů
- model - obsahuje datové třídy entit
- model.util - obsahuje pomocné třídy aplikace, převážně se statickými metodami
- model.validator - obsahuje validační třídu
- res - obsahuje obrázky a XML soubory s grafickým prostředím a texty použitými v aplikaci

Diagram 2.5 ukazuje strukturu balíčku cz.smartfine, který má funkci části Controller. Jednotlivé třídy již obsahují datové typy svých proměnných a metody.

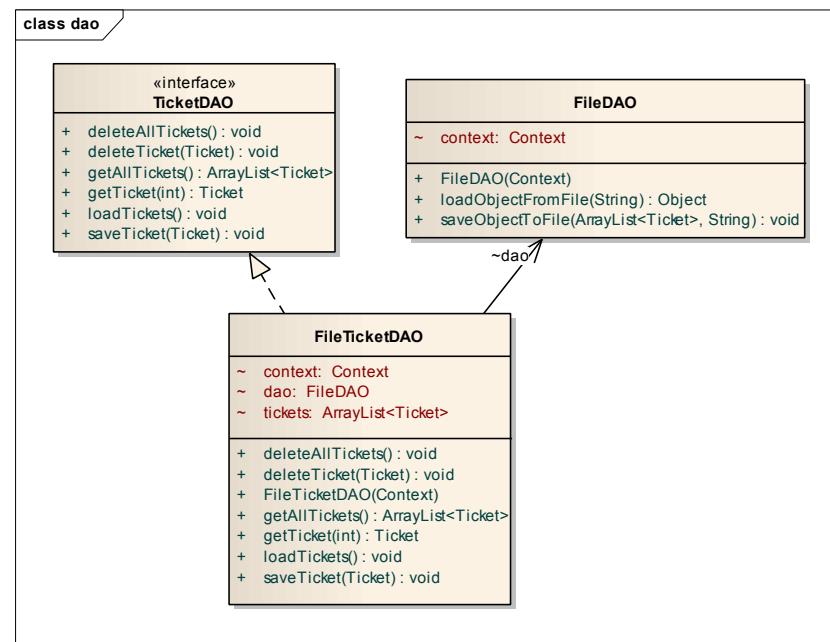
Diagram 2.6 ukazuje strukturu balíčku dao.

Diagram 2.7 ukazuje strukturu balíčku model, který má funkci části Model. V diagramu je oproti doménovému modelu nová třída Settings. Jde o preferenze, která si uživatel může nastavit, zatím lze pouze přednastavit město a adresu serveru Smart-Fine. Třída není obsažena v doménovém modelu, protože její objekt vyloženě neukládá data jako třídy parkovacího lístku a odkazu na zákon. Je však zařazena do modelu, jelikož zajišťuje přístup k datům uloženým v preferencích aplikace. Preference slouží k persistentnímu uložení dat primitivních datových typů. Širší využití třídy Setting nastane až v dalších iteracích. U třídy Ticket nejsou znázorněny metody GET a SET z důvodu jejich většího počtu.

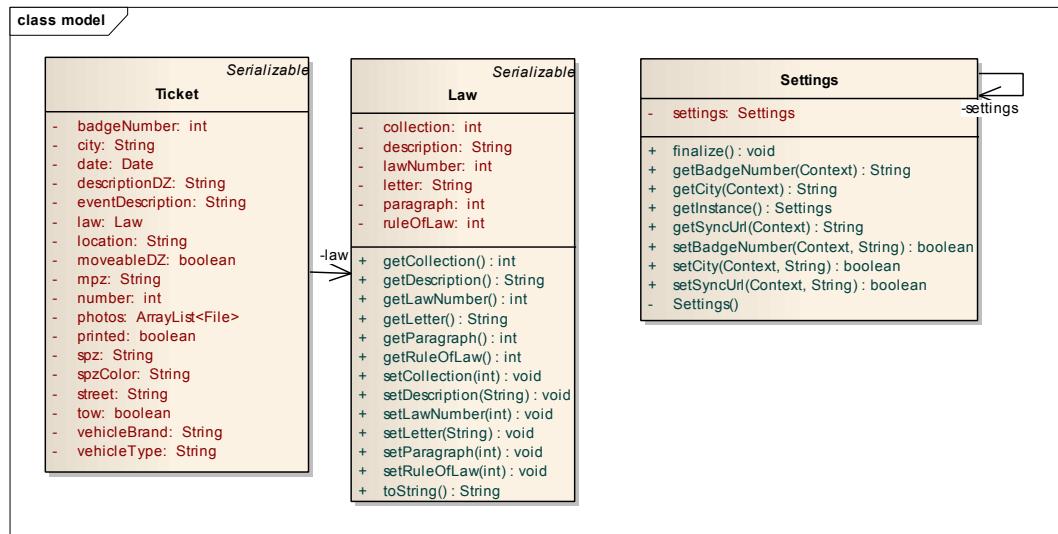
Diagramy ostatních balíčků jsou v příloze [C]. Kompletní popis implementace je zpracován za pomocí JavaDoc (k dispozici na CD).



Obrázek 2.5: Balíček tříd - smartfine



Obrázek 2.6: Balíček tříd - dao



Obrázek 2.7: Balíček tříd - model

2.8 Testování

Základní testování programátorem probíhá prakticky po celou dobu vývoje. Hned jak dojde k implementaci jakékoliv části systému, dojde k jejímu otestování. Ta však nejsou nikde uvedena, jelikož jsou drobná, samozřejmá a jsou součástí vývoje. Během první iterace však

došlo ke dvěma větším testováním, která provedli uživatelé, jež se na vývoji nepodílí. Tak bylo možné získat informace od nezainteresovaných osob.

2.8.1 Heuristická evaluace

V počátcích vývoje, kdy byl k dispozici pro testování pouze PDF prototyp (k dispozici na CD), proběhlo testování podobné tzv. Heuristické evaluaci [9]. Dvou testerům byly zadány úkoly pro práci s prototypem, které provedli, a výsledkem jsou hodnocení obou zúčastněných.

Pro testování byla použita tato kritéria:

1. Vидitelnost statusu systému
2. Uživatelské prostředí a svoboda pohybu
3. Prevence chyb
4. Design a struktura

Pro testování byly použity tyto úkoly:

1. Vytvořte nový lístek a uložte ho.
2. Najděte uložený lístek a upravte ho. Následně ho uložte.
3. Nahrajte všechny záznamy na server.

Evaluace - Tester 1 - student ČVUT (Softwarové inženýrství)

V prototypu se lze zatím prakticky pouze přesouvat mezi obrazovkami, veškerá funkčnost aplikace je v tomto typu prototypu nemožná. To samozřejmě značně omezuje zadáne úkoly.

Ačkoliv je přesně vidět, kde se člověk zrovna nachází, není zcela jasné, jak se dostat zpět. Znalý uživatel androidu ví, že přes tlačítko „Zpět“ na zařízení, neznalý tohoto však může mít problém. (Kritérium 1. Úkol 1, 2, 3)

Důležitou částí navigace jsou tlačítka ve spodní části obrazovky (při vytváření lístku, prohlížení záznamů, prohlížení a editaci lístku). Tlačítko je vidět neustále, což by mohl být problém, v případě, že si uživatel neuvědomí, že může obsahem nad tlačítkem pohybovat. (Kritérium 2, Úkol 1, 2, 3)

Chybí potvrzení pro uživatele, o nějaké vykonané akci. Úspěšné uložení záznamu atd. (Kritérium 3. Úkol 1, 2, 3)

Evaluace - Tester 2 - ČVUT (Komunikace a multimédia)

Systém nekontroluje, zda jsou zadaná veškerá data potřebná k identifikaci. (Kritérium 3, Úkol 1, 2)

Klást důraz na popisky, důležité, aby opravdu vystihovaly to, co mají. Např.: „Parkovací lístek“ -> „Nový parkovací lístek“.

V aplikaci neexistuje jakékoliv potvrzování rozhodnutí typu „Opravdu chcete smazat lístek?“. Mohl bych omylem udělat něco, co bych nechtěl. (Kritérium 3. Úkol 1, 2, 3)

2.8.2 Uživatelské testování

Podobně probíhalo také uživatelské testování v závěru iterace. Mělo by ukázat, zda je aplikace intuitivní na ovládání, přehledná a logicky strukturována. Testuje se celkově celá aplikace, uživatel provádí jednotlivé úkoly a komentuje je. Opět byli přítomni 2 testeři, kteří prováděli sadu úkolů, tentokrát však měli k dispozici funkční prototyp přímo na mobilních zařízeních. Celý průběh byl monitorován a je k dispozici na CD v podobě videozáznamu.

Seznam úkolů pro testování:

1. Vytvořte a uložte nový lístek, pouze povinné údaje.
2. Vytvořte a uložte nový lístek, opatřete ho fotodokumentací a využijte nápovedy vyplňení.
3. Upravte první parkovací lístek, přidejte mu fotodokumentaci.
4. Vytiskněte druhý lístek.
5. Změňte v nastavení Město, a vytvořte nový lístek.
6. Odešlete všechny lístky na server.

Výsledek testování:

1. video - Tester 1 - student ČVUT - m. telefon: Samsung Galaxy S
2. video - Tester 2 - student VŠE - m. telefon: HTC Desire

Oba uživatelé si vedli při testování dobře. Práce byla přímočará a prakticky bez zádrhelů. Uživatelé vždy věděli, kam přesně kliknou a kde hledat tlačítka a formuláře. To poukazuje na intuitivní a přehlednou strukturu aplikace i pro ty, co s Androidem běžně nepracují. Design přišel oběma uživatelům ucházející, ale chtěli by ho nějakým blíže nespecifikovaným způsobem vylepšit.

2.9 Závěr

Jak bylo zmíněno v úvodu této kapitoly, 1. iterace probíhala přibližně dobu jednoho semestru, cca čtyři měsíce. Během této doby byla z ničeho vytvořena nová aplikace se základním uživatelským rozhraním a funkčností, která bude v dalších přibývat. Byla sepsána dokumentace a vytvořen systém, kterým se vývoj aplikace bude nadále ubírat.

Iterace se dá považovat za úspěšnou, funkční požadavky F.Req.3 a F.Req.4 byly implementovány úplně, požadavky F.Req. 1, F.Req. 2 a F.Req.7 částečně. Aplikace umí vytvořit nový parkovací lístek i s pomocí návodů a přidat k němu fotodokumentaci. Parkovací lístky lze zobrazit v seznamu, nebo v detailu, a lze je upravit.

Seznam uvádí, co bylo a nebylo splněno, a co zůstalo do dalšího vývoje. V dalších iteracích bude také kladen důraz na výsledky testování a na komentáře uživatelů.

2.9.1 Splněné funkční požadavky:

F.Req.1.: Vyplnění parkovacího lístku

1. Vyplnění informací o datu a času přestupku, SPZ, MPZ, barvě SPZ, druhu vozidla, tovární značky, města, ulice, čísla, místa, identifikačního čísla, přestupku (paragraf, odstavec, písmeno, zákon, sbírka, popis) a možnost výběru odtahu, přenosné DZ a fotodokumentace.
2. Všechny vyjmenované informace (kromě výběrových) bude možné zadat za pomocí softwarové klávesnice nebo hardwarové klávesnice integrované do mobilního zařízení.
3. Systém bude standardně předvyplňovat údaje: datum, čas, město, identifikační číslo policisty.
5. Systém bude vyplněné parkovací lístky ukládat do paměti mobilního zařízení

F.Req.2.: Návod při vyplňování čísel paragrafů zákonů nejběžnějších přestupků

1. Systém bude uživateli nabízet výčet nejběžnějších přestupků a po vybrání uživatelem sám vyplní příslušná čísla zákonů a popis do parkovacího lístku
3. Systém bude uživateli nabízet výčet nejběžnějších voleb v položkách: MPZ, Barva SPZ, Druh vozidla, Tovární značka

F.Req.3.: Prohlížení lokálně uložených parkovacích lístků

F.Req.4.: Úprava lokálně uložených parkovacích lístků

F.Req.7.: Pořízení fotodokumentace přestupku

1. Systém bude umožňovat při vyplňování parkovacího lístku vytvořit fotodokumentaci přestupku
2. Fotodokumentaci bude možné pořídit pomocí před-instalované aplikace fotoaparátu v systému mobilního telefonu

2.9.2 Nesplněné funkční požadavky:

F.Req.1.: Vyplnění parkovacího lístku

4. Systém bude volitelně a podle aktuálních možností mobilního zařízení předvyplňovat údaje: adresa, SPZ

F.Req.2.: Návod pro vyplňování čísel paragrafů zákonů nejběžnějších přestupků

2. Uživatel si bude moci přidat vlastní přestupky
4. Uživatel si bude moci přidat vlastní volby v položkách: MPZ, Barva SPZ, Druh vozidla, Tovární značka

F.Req.5.: Odeslání parkovacích lístků z mobilního zařízení na server

F.Req.6.: Tisk parkovacích lístků pomocí mobilní tiskárny

F.Req.7.: Pořízení fotodokumentace přestupku

3. Fotodokumentaci bude možné přidat vybráním předem nařízených fotografií ze před-instalované aplikace galerie v systému mobilního telefonu
4. Systém bude fotodokumentaci ukládat do externí paměti (na paměťovou kartu)

F.Req.8.: Rozpoznání státní poznávací značky z fotografie

F.Req.9.: Automatické vyhledání adresy pomocí GPS

F.Req.10.: Kontrola odcizení přenosné parkovací karty opravňující parkovat v modré zóně

F.Req.11.: Přístup k funkcím aplikace po zadání identifikačního čísla

Kapitola 3

Iterace 2

3.1 Úvod, cíle iterace

Text této kapitoly se zabývá veškerou prací udělanou během 2. iterace. Hlavním cílem je pokračování v implementování funkcionality stanovené v iteraci 1. Jde o funkční požadavky, které byly dokončeny jen z části, případně jejich vývoj zatím vůbec nezačal. Nově se od této iterace objevuje nový požadavek na funkčnost, a to:

F.Req.12.: Kontrola parkovaní zaplaceného přes SMS

1. Vyplnění SPZ vozidla.
2. Systém bude umožňovat online ověření vozidla pomocí SPZ, zda dané vozidlo zaplatilo parkování. Pokud ano, v jakém časovém intervalu má povolení parkovat

3.2 Případy a scénáře užití

S novým funkčním požadavkem vznikl také nový případ užití:

UC.9.: Kontrola parkovaní zaplaceného přes SMS

- Systém bude umožňovat online ověření vozidla pomocí SPZ, zda dané vozidlo zaplatilo parkování. Pokud ano, v jakém časovém intervalu má povolení parkovat

Došlo k doplnění scénářů pro případy užití UC.06 a nový UC.09:

S.6.: Kontrola odcizení parkovací karty

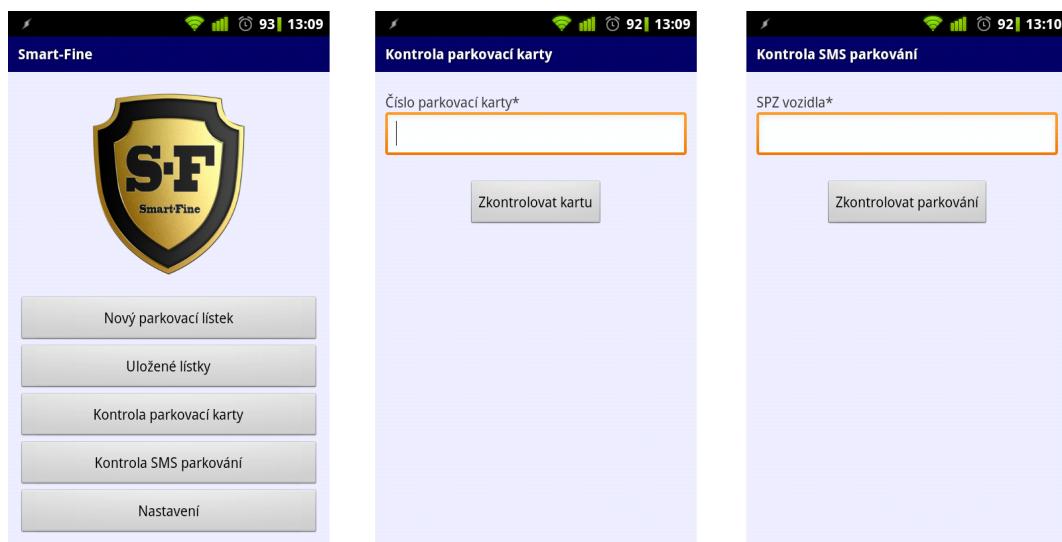
1. Uživatel chce ověřit, zda parkovací karta souhlasí s SPZ vozidla.
2. Systém zobrazí zadávací formulář na číslo parkovací karty.
3. Uživatel vyplní číslo parkovací karty.
4. Uživatel požádá systém o ověření čísla parkovací karty.
5. Systém ověří připojení k internetu.

- (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
6. Systém získá informace ze serveru a zobrazí je.
 - (a) Pokud se získání informací nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
- S.9.: Kontrola parkovaní zaplacného přes SMS**
1. Uživatel chce ověřit, zda má vozidlo zaplacné parkování pomocí SMS.
 2. Systém zobrazí zadávací formulář na SPZ vozidla.
 3. Uživatel vyplní SPZ vozidla.
 4. Uživatel požádá systém o parkování.
 5. Systém ověří připojení k internetu.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
6. Systém získá informace ze serveru a zobrazí je.
 - (a) Pokud se získání informací nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.

Změněný diagram modelu případů užití a diagram modelu závislostí mezi případy užití a požadavky na systém jsou k dispozici v příloze [B]. Nejsou uvedeny zde, jelikož se v tomto momentě tolik neliší od diagramů v první iteraci.

3.3 Uživatelské prostředí

Vzhledem k tomu, že je již část aplikace hotová, a vzhledem k jednoduchosti, nebyl vytvořen návrh uživatelského prostředí pro nové funkce v návrhovém editoru, jako tomu bylo v 1. iteraci, ale byl realizován přímo v aplikaci Eclipse [6]. Obrázek 3.1. Automatické určení polohy pomocí GPS se vyvolá tlačítkem při vytváření nového parkovacího lístku, dané tlačítka bylo navrženo a vytvořeno již v 1. iteraci. Naopak kontrola parkovací karty a kontrola parkování zaplacného přes SMS vyžadovaly nové aktivity a tudíž i nové grafické rozhraní. Pro kontrolu se zadá číslo parkovací karty, případně SPZ do příslušných formulářů a stiskem tlačítka se provede kontrola. Výsledné informace o parkování, či parkovací kartě, se zobrazí v prázdném prostoru pod tlačítky. Z důvodu přibývající funkcionality se musela upravit také úvodní obrazovka s menu, aby nedošlo k znepřehlednění.



Obrázek 3.1: Změněné a přidané obrazovky

3.4 Implementace

Diagramy ukazující nové třídy v jednotlivých balíčcích jsou k dispozici v příloze[C]. Kompletní popis implementace je zpracován za pomocí JavaDoc (k dispozici na CD).

3.4.1 Pořízení fotodokumentace

Z 1. iterace zbyla rozdělaná práce implementace přidávání fotodokumentace při vytváření nového parkovacího lístku, případně jeho editace. Fotografie šly fotit pomocí systémové aplikace fotoaparátu, která je standardní výbavou operačního systému Android. Nyní bude moci uživatel vybrat předem vyfocenou fotografií z galerie telefonu. Ta zůstane uložena na svém původním místě a k parkovacímu lístku se uloží pouze odkaz na ní v podobě objektu typu File. Při pořizování nových fotografií skrze aplikaci Smart-Fine se fotografie uloží na externí, nebo interní kartu, kde nebudou zabírat spoustu paměti, do složky „smartfine“. Po odeslání lístků na server se fotografie z této složky vymažou, aby nezabíraly zbytečně místo.

3.4.2 Určení aktuální adresy

Dalším předmětem implementace byla funkce určení adresy pomocí GPS, při vytváření parkovacího lístku. To nastane po stisknutí tlačítka s kompasem u formuláře s adresou. Jelikož určení polohy není přímočará záležitost, je potřeba čekat na dobrý signál a GPS modulu chvíli trvá, než polohu nalezne, není řešení zcela primitivní.

Pro získání polohy musí mít uživatel na mobilním telefonu zapnutého buď internetového, nebo GPS poskytovatele polohy (dále NET provider, GPS provider). V nejlepším případě oba. GPS provider má z důvodu větší přesnosti přednost před NET providerem. Ten je ale zase mnohem rychlejší. Po požádání o aktuální polohu se nejdříve zkонтroluje dostupnost

providerů, a pokud je alespoň jeden z nich aktivní, přiřadí se mu posluchač změny polohy. Ten se zavolá ve chvíli, kdy byla nová poloha nalezena. Hledání běží v jiném vlákně, takže, takže se UI aplikace „nezasekne“ a uživatel může mezikádem dále vyplňovat parkovací lístek. V kódu je také nastaven časovač pomocí objektu Handler, který se spustí po 30 vteřinách a pokud GPS provider polohu do té doby nenašel, hledání polohy se ukončí. Pokud toto nastane, uživatel bude muset znovu o polohu požádat stiskem tlačítka s kompasem.

Pro získání adresy z GPS souřadnic je potřeba připojení k internetu. Adresa lze získat pomocí objektu Geocoder. Po úspěšném získání adresy se automaticky vyplní do formulářů. To je možné právě díky objektu třídy Handler, který spouští objekt typu Runnable v UI vlákně. Více o vláknech a UI ve 4. kapitole v sekci Komunikace se serverem.

3.5 Automatické rozpoznávání SPZ z fotografie

Jedním z funkčních požadavků na systém (konkrétně F.Req.1.4) mělo být automatické rozpoznání SPZ vozidla z fotografie a předvyplnění formuláře při vytváření nového parkovacího lístku. Implementace problému však není vůbec jednoduchá, jelikož Android, ani jazyk Java samy o sobě nenabízí žádný „jednoduchý“ způsob implementace.

K dispozici nejsou téměř žádné volně dostupné knihovny na OCR pro Android. Existuje verze OCR knihovny Tesseract [8] (psána bohužel v C++), která se jmenuje Tesjeract [7] a dá se použít pro Android, jelikož se už jedná o Javu, ale po vyzkoušení aplikace OCR Test [13] založené na této knihovně nakonec použita nebyla. Aplikace sice dokázala rozpoznat text, dokonce i SPZ, ale bylo potřeba namířit mobilní telefon přesně na určité místo, aby rozpoznání zafungovalo. I přes to nebyl výsledek vždy stoprocentní. Policista získá SPZ vozidla daleko snadněji a rychleji obyčejným opsáním do formulářového pole.

Další aplikací pro Android, která dokáže již daleko lépe rozpoznat SPZ vozidla, je LPlateEU ANPR [20]. I přes to, že autorem je Čech, nepodařilo se s ním navázat kontakt.

Nakonec byla nalezena bakalářská práce Ondřeje Martinského [15], který se zabýval právě tématem rozpoznání SPZ vozidla z fotografie. Bohužel, také se mi s ním nepodařilo navázat kontakt.

Na ČVUT jsem zkontoval pana profesora Ing. Jiřího Matase, Ph.D, který se tématem (i když ne pro Android) zabývá již zhruba 10 let. Po krátké konverzaci mi doporučil, abych úmyslu vytvořit rozpoznávání SPZ vozidla úplně od začátku zanechal, jelikož je téma velmi rozsáhlé a časově velmi náročné. Po jeho doporučení a po prohlédnutí bakalářské práce a zdrojového kódu Ondřeje Martinského jsem tak nakonec učinil. Rozpoznání SPZ vozidla z fotografie by představovalo v systému Smart-Fine pouze malou část funkčnosti a vzhledem k náročnosti byl i z časových důvodů tento funkční požadavek odstraněn.

3.6 Závěr

Během 2. iterace, která trvala asi měsíc, bylo splněno několik dalších funkčních požadavků. Iterace se obecně nedá považovat za zcela úspěšnou, jelikož za dobu trvání bylo vypracováno poměrně málo implementace. Dokončen byl požadavek F.Req.9 (F.Req.1.4) a požadavek F.Req.7. Došlo k rozpracování požadavku F.Req.10 a F.Req.12. Aplikace nyní umí nalézt

aktuální polohu a předvyplnit adresu. Přidávání fotodokumentace ne nyní možné i ze systémové aplikace galerie.

Seznam uvádí, co bylo a nebylo splněno, a co zůstalo do dalšího vývoje.

3.6.1 Splněné funkční požadavky:

F.Req.1.: Vyplnění parkovacího lístku

4. Systém bude volitelně a podle aktuálních možností mobilního zařízení předvyplňovat údaje: adresa

F.Req.7.: Pořízení fotodokumentace přestupku

F.Req.9.: Automatické vyhledání adresy pomocí GPS

F.Req.10.: Kontrola odcizení přenosné parkovací karty opravňující parkovat v modré zóně

1. Vyplnění čísla karty.

F.Req.12.: Kontrola parkovaní zaplacенного přes SMS

1. Vyplnění SPZ vozidla.

3.6.2 Nesplněné funkční požadavky:

F.Req.2.: Návod při vyplňování čísel paragrafů zákonů nejběžnějších přestupků

2. Uživatel si bude moci přidat vlastní přestupky
4. Uživatel si bude moci přidat vlastní volby v položkách: MPZ, Barva SPZ, Druh vozidla, Tovární značka

F.Req.5.: Odeslání parkovacích lístků z mobilního zařízení na server

F.Req.6.: Tisk parkovacích lístků pomocí mobilní tiskárny

F.Req.10.: Kontrola odcizení přenosné parkovací karty opravňující parkovat v modré zóně

2. Systém bude umožňovat online porovnání čísla zadáné parkovací karty se seznamem odcizených přenosných parkovacích karet

F.Req.11.: Přístup k funkcím aplikace po zadání identifikačního čísla

F.Req.12.: Kontrola parkovaní zaplacенного přes SMS

2. Systém bude umožňovat online ověření vozidla pomocí SPZ, zda dané vozidlo zaplatilo parkování. Pokud ano, v jakém časovém intervalu má povolení parkovat

Kapitola 4

Iterace 3

4.1 Úvod, cíle iterace

3. iterace je poslední iterací, kterou se zabývá tato bakalářská práce. Jejím cílem je dokončit veškerou nedodělanou funkcionality, definovanou funkčními požadavky v kapitole 2. Jelikož jde o hodně nedokončené práce, nastane v této iteraci velké množství změn. Nově se od této iterace objevuje nový funkční požadavek, a to:

F.Req.13.: Sledování a evidence aktuální polohy mobilního zařízení

1. Systém bude od přihlášení až do odhlášení snímat aktuální polohu mobilního zařízení.
2. Geolokační data se odešlou spolu s parkovacími lístky na server najednou.

Ačkoliv je evidence pohybu policistů přímo v zadání bakalářské práce, původně se jednalo o sledování adres odeslaných parkovacích lístků. Nyní v rámci nového funkčního požadavku bude poloha sledována kontinuálně po celou dobu chodu aplikace.

4.2 Případy a scénáře užití

S novým funkčním požadavkem také vznikl nový případ užití:

UC.10.: Evidence pohybu policistů

- Systém bude od přihlášení do aplikace až do odhlášení z aplikace snímat a ukládat aktuální polohu mobilního zařízení.
- Data o poloze se odešlou na server spolu se seznamem parkovacích lístků.

Vzhledem k tomu, že nový případ užití nevykonává uživatel, ale systém na pozadí, není přiřazen k aktéru policisty a nemá scénář užití.

Došlo také k úpravě funkčního požadavku F.Req.2. Původně definoval, že uživatel bude mít možnost vytvářet a přidávat vlastní nejčastější volby u položek MPZ vozidla, barva

SPZ vozidla, druh vozidla a tovární značka vozidla. Ty by následně využil jako návodů při vyplňování parkovacího lístku. Seznamy položek jsou ale víceméně jasně dané (např. existuje přesný seznam všech světových MPZ, atd.) a proto je možnost přidávání vlastních položek zbytečná. Kdyby nastala situace, že musí policista použít jinou hodnotu, než je v nabídce, může ji vyplnit přímo při vytváření parkovacího lístku. Seznamy všech MPZ a továrních značek vozidel jsou velmi rozsáhlé (kolem 300 záznamů) a při vytváření parkovacího lístku by trvalo dlouho, než by policista našel správnou hodnotu. V nastavení aplikace má tedy uživatel místo vytváření nových položek volbu vybrat si tzv. oblíbené položky, které se následně zobrazí při vytváření nového lístku místo celého dlouhého seznamu. Položky barva SPZ vozidla a druh vozidla tuto možnost nemají, jelikož seznamy těchto voleb jsou velmi krátké.

Funkční požadavek také doposud stanovoval, že uživatel bude mít obdobně přidávat i vlastní přestupky. Opět, ze stejných důvodů jako výše, tato možnost v aplikaci nakonec nebude. Jelikož je seznam také krátký, nebude mít policista ani možnost vybrat si „oblíbené“ přestupky.

Dále došlo ke změně scénářů užití S.1 případu užití UC.1 a S.4 případu užití UC.4. Jde o vytváření nových parkovacích lístků a editaci již vytvořených. Uživatel se bude muset nyní lokálně (míněno offline) autentizovat před vytvořením nebo editací parkovacího lístku. Opatření zabezpečí, že tyto činnosti bude mít možnost provádět pouze policista, který je zrovna přihlášen v aplikaci. To zamezí, aby nikdo jiný nemohl vytvářet či měnit parkovací lístky pod služebním číslem svého kolegy. Při přihlašování se automaticky vyplní služební číslo policisty, ten tedy potřebuje vyplnit pouze PIN. To alespoň trochu redukuje zpoždění způsobené samotným přihlašováním. Změněné scénáře užití jsou tedy:

S.1.: Vyplnění nového parkovacího lístku

1. Uživatel chce vyplnit nový parkovací lístek.
2. INCLUDE (Autentifikace uživatele)
3. Systém zobrazí zadávací formulář.
4. Uživatel vyplní všechna textová pole a vybere druh přestupku z nabídky.
5. Uživatel volitelně nafotí přestupek.
6. Systém zkонтroluje validitu všech dat.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
7. Systém uloží parkovací lístek.

S.4.: Úprava lokálních záznamů

1. Uživatel chce upravit parkovací lístek.
2. INCLUDE (Prohlížení lokálních záznamů)
3. Uživatel požádá o zobrazení editačního formuláře.
4. INCLUDE (Autentifikace uživatele)
5. Systém zobrazí editační formulář.

6. Uživatel upraví požadované informace.
7. Uživatel požádá o uložení parkovacího lístku.
8. Systém zkонтroluje validitu všech dat.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 6.
9. Systém uloží upravený parkovací lístek.
10. Návrat do Prohlížení lokálních záznamů.

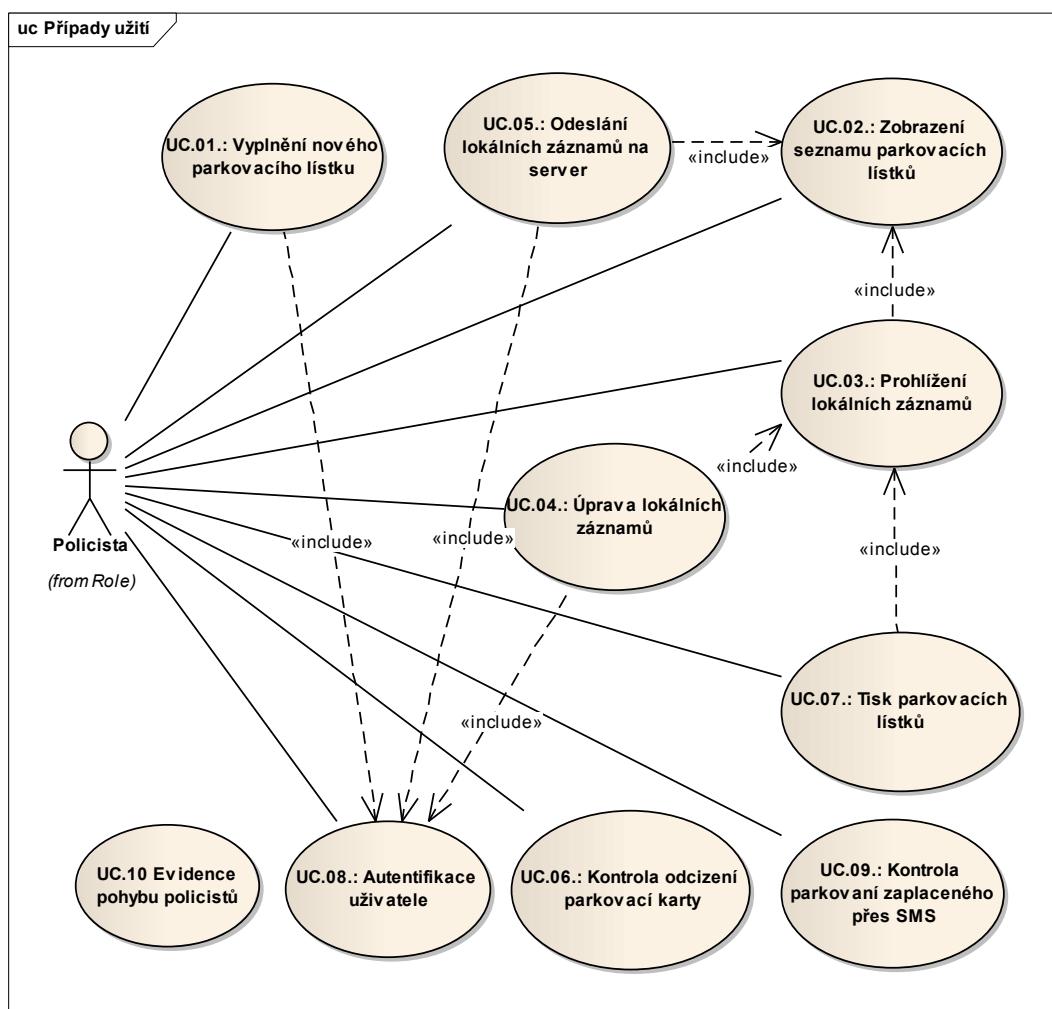
Nakonec došlo také na doplnění scénáře S.7 pro případ užití UC.7, zabývající se tiskem parkovacího lístku na mobilní tiskárně, který nebyl doposud zpracován.

S.7.: Tisk parkovacích lístků

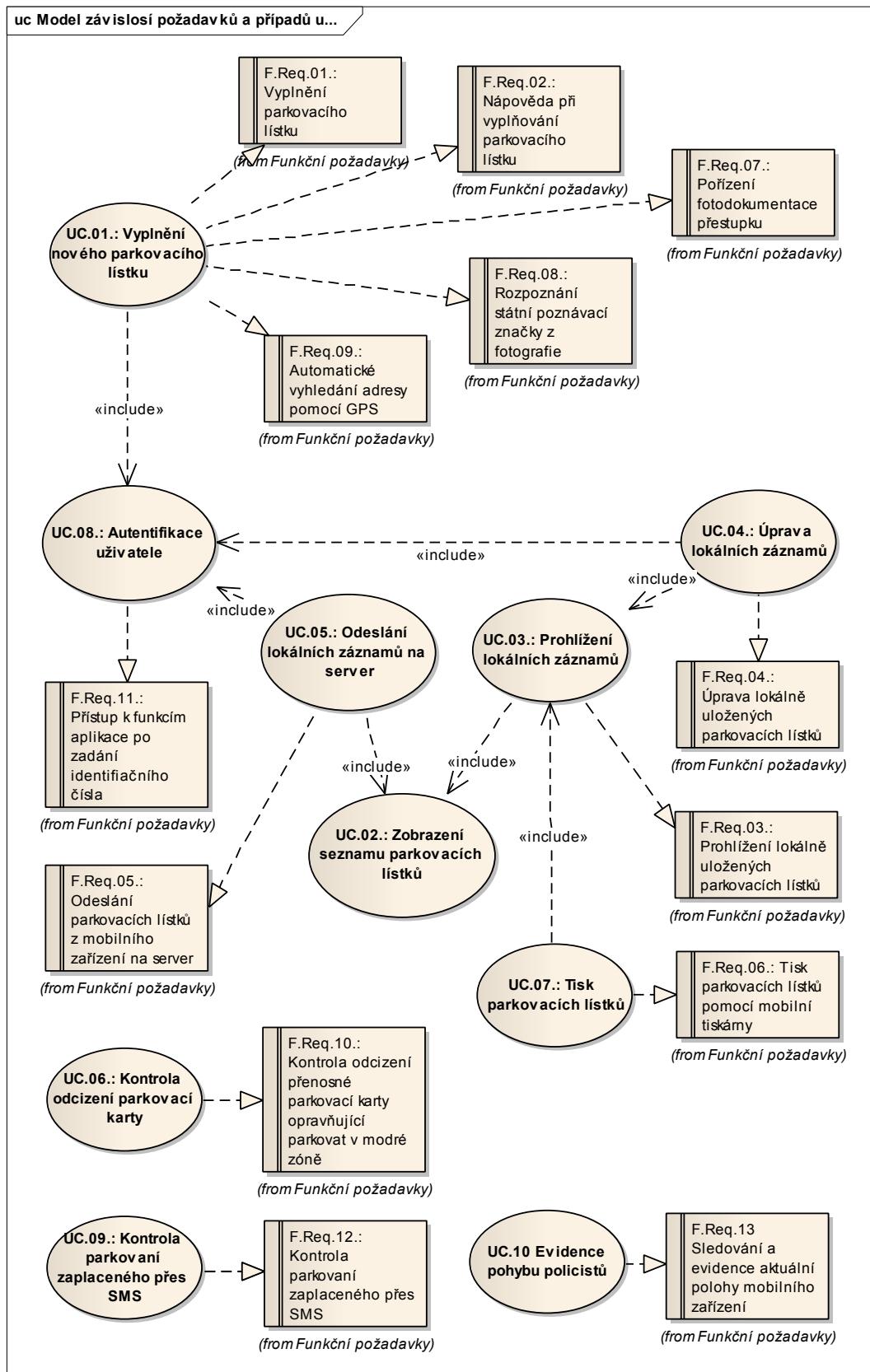
1. Uživatel chce vytisknout parkovací lístek.
2. Uživatel požádá systém o tisk parkovacího lístku.
3. Systém požádá uživatele o potvrzení akce.
4. Uživatel potvrdí akci.
5. Systém zkonzroluje, zdali je zapnutý Bluetooth.
 - (a) Pokud není, vyzve uživatele aby Bluetooth zapnul.
 - (b) Pokračuje se bodem 5.
6. Systém vytiskne parkovací lístek.
7. Systém zobrazí hlášku, jak operace dopadla.
 - (a) Pokud se tisk nepodaří, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 2.

Kompletní výpis všech případů a scénářů užití je k dispozici v příloze [B].

Vzhledem ke spoustě změnám, které byly s případy a scénáři užití provedeny, se změnily diagramy ukázané v kapitole 2. Diagram 4.1 ukazuje model případů užití po provedených změnách. Diagram 4.2 ukazuje model závislostí požadavků na případech užití po provedených změnách.



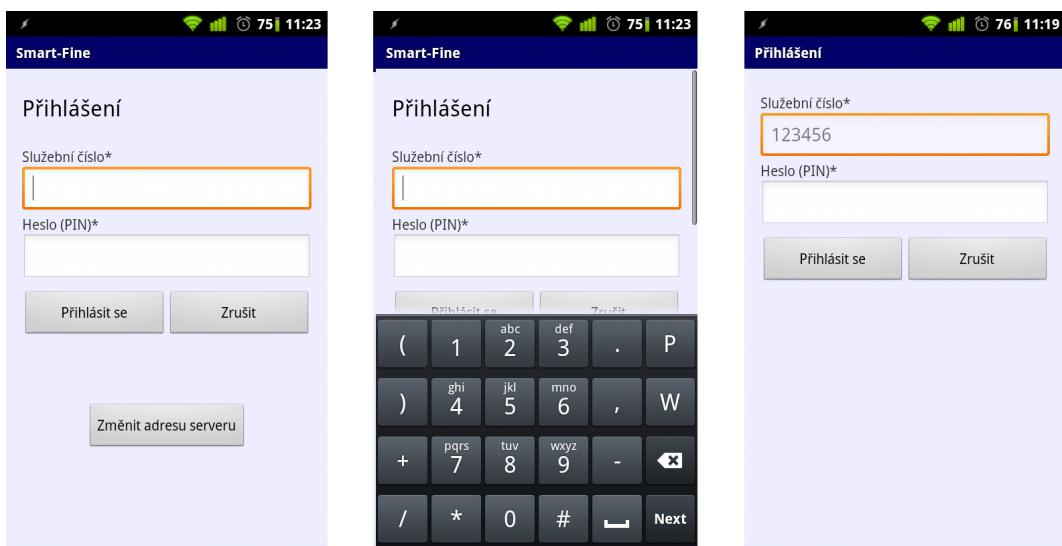
Obrázek 4.1: Model případů užití



Obrázek 4.2: Model závislostí požadavků na případech užití

4.3 Uživatelské prostředí

Uživatelské prostředí se od minulých iterací už příliš nezměnilo, spíše přibyly nové obrazovky. Veškeré změny, které nastaly, byly opět, z důvodu jednoduchosti, navrženy přímo v aplikaci, nikoliv v nějakém návrhovém editoru. Přibyla nová obrazovka přihlášení 4.3. Ta obsahuje 2 formuláře na vyplnění služebního čísla a pinu policisty a přihlašovací tlačítko. Po zadání údajů a stisknutí tlačítka se provede přihlášení. Přihlašovací obrazovka, která se ukáže ihned po spuštění aplikace má navíc formulář na změnu adresy serveru, se kterým aplikace komunikuje. Ten se vyvolá stiskem tlačítka ve spodní části přihlašovací obrazovky.

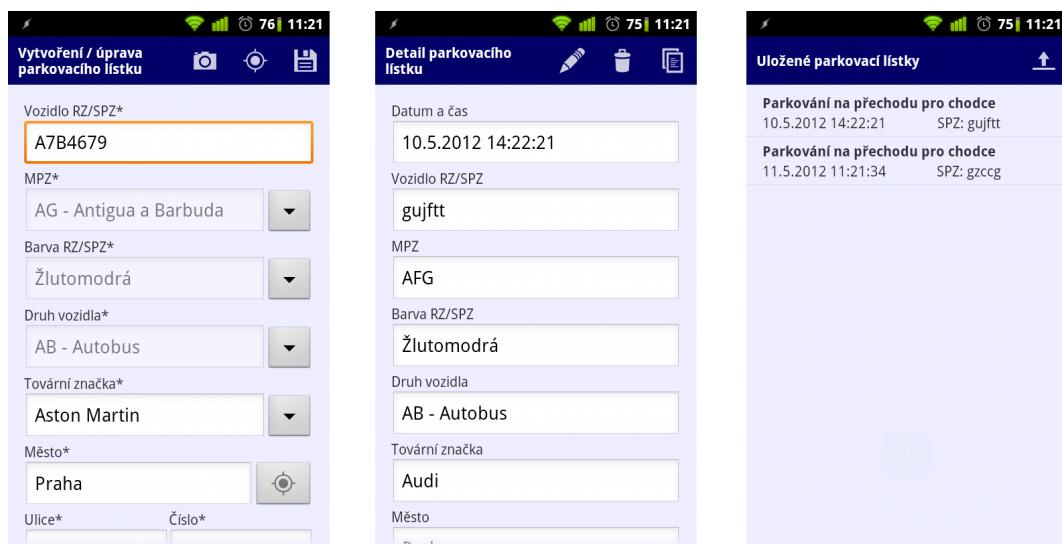


Obrázek 4.3: Obrazovky přihlášení

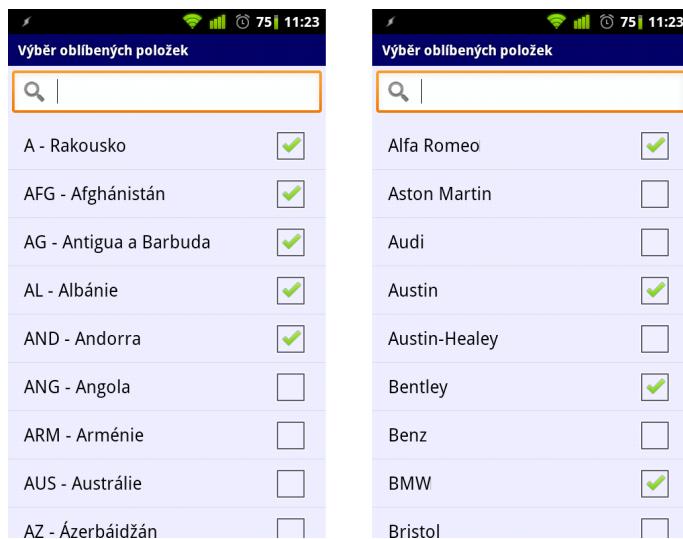
Další větší změnou v grafickém prostředí aplikace bylo vytvoření tzv. action baru na obrazovkách vytváření nového parkovacího lístku (nebo jeho úpravy), zobrazení detailu parkovacího lístku a zobrazení seznamu uložených lístků. Viz obrázek 4.4. Na těchto obrazovkách byla funkcionality dříve spouštěna přes tlačítka ve spodní části obrazovky, nyní je ale přesunuta do lišty v horní části. To má za následek zvětšení prostoru, ve kterém jsou zobrazovány informace, lepší přehlednost a dostupnost navigačních prvků. Action bar je hojně používán v novějších verzích operačního systému Android a je velmi oblíbený, jde o „modernější“ vzhled.

Přidána byla také podpora pro hardwarové tlačítko „Menu“, které má každý mobilní telefon se systémem Android. Je jedním ze základních navigačních prvků. Po stisknutí tohoto tlačítka vyjede ve spodní části obrazovky menu, které představuje další způsob ovládání aplikace.

Z grafického hlediska změnou prošla ještě obrazovka nastavení aplikace, kde přibylo pár nových položek, a vznikly nové obrazovky výběru oblíbených položek. Viz obrázek 4.5. V horní části obrazovky se nachází formulář, přes který lze v seznamu, zobrazeném pod formulářem, vyhledávat. Každá položka pak obsahuje políčko, jehož zaškrtnutím se položka označí jako oblíbená.



Obrázek 4.4: Obrazovky s přidaným action barem



Obrázek 4.5: Obrazovky výběru oblíbených položek

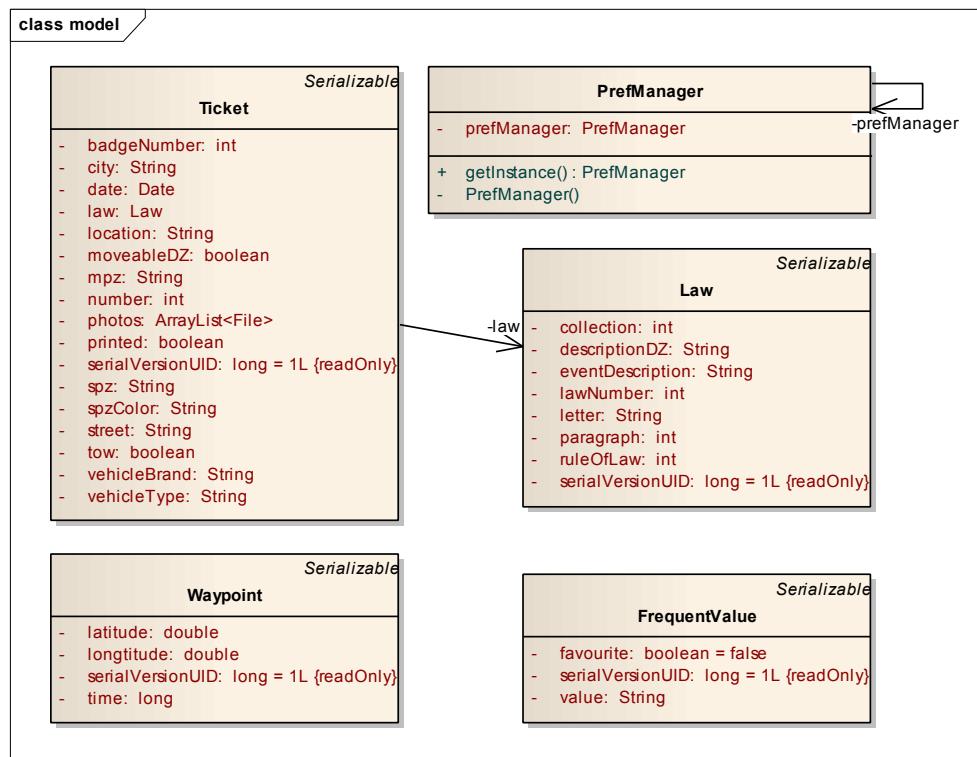
Všechny obrazovky aplikace včetně různých variací jsou k dispozici v příloze [C].

4.4 Implementace

Během 3. iterace došlo k rozsáhlé implementaci funkcionality aplikace. To mělo za následek změny v diagramech tříd a ve struktuře balíčků, ukázaných v iteraci 1. Všechny balíčky mají nyní nový prefix android, tzn. ve, co bylo dříve v balíčku cz.smartfine.* je nyní v

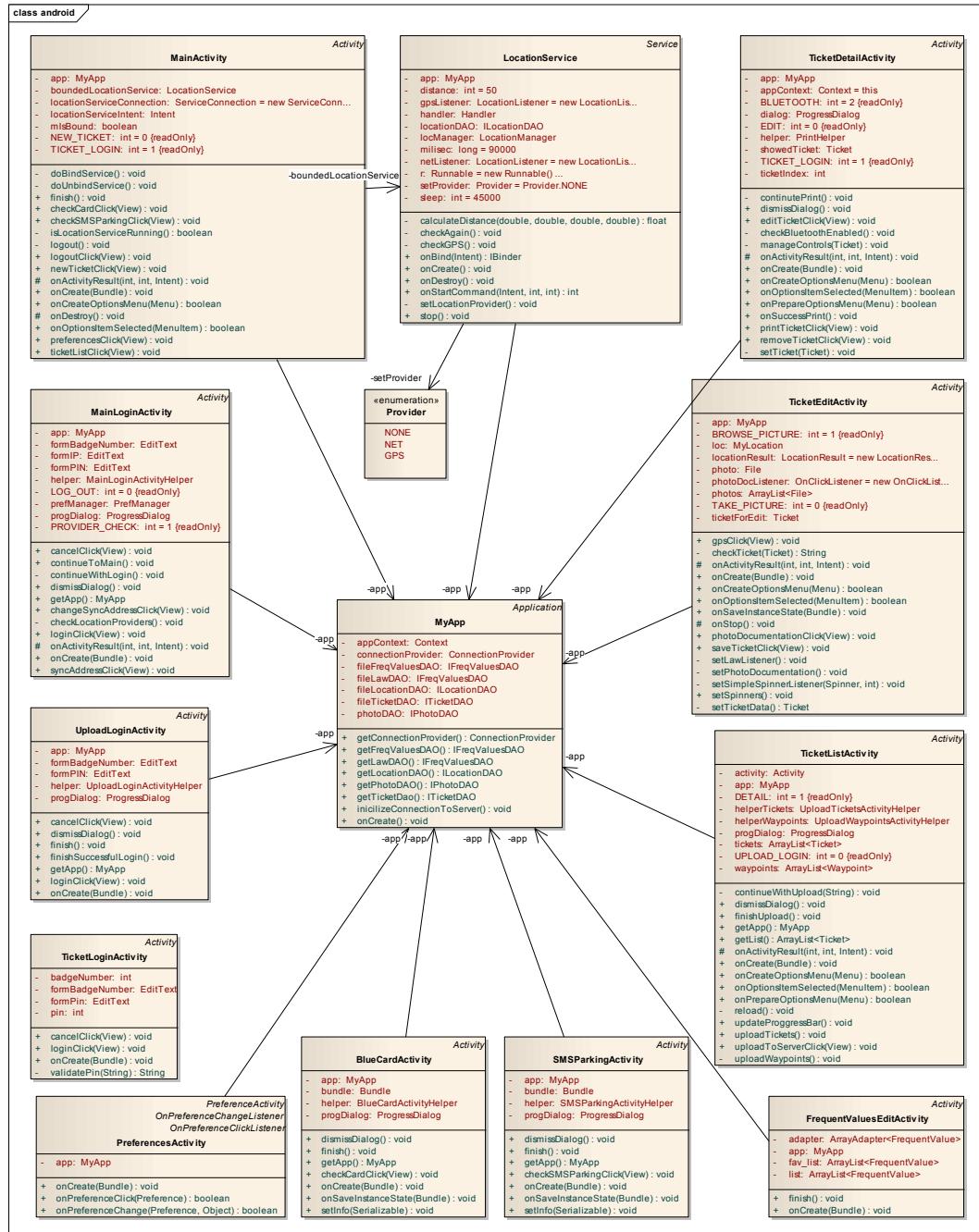
cz.smartfine.android.*. K této změně došlo z důvodu potřeby odlišit balíčky klientské aplikace od obecné síťové vrstvy, balíčků serveru atd. Struktura balíčků nyní vypadá takto:

- cz.smartfine.android - obsahuje třídy aktivit a třídu služby (Service)
- adapters - obsahuje třídy adaptérů, které slouží k naplnění seznamů
- dao - poskytuje přístup k souborové vrstvě a slouží jako uložiště objektů
- dao.interfaces - obsahuje interface pro objekty DAO
- helpers - obsahuje pomocné třídy pro komunikaci se serverem
- model - obsahuje datové třídy entit
- model.util - obsahuje pomocné třídy aplikace, převážně se statickými metodami
- printlayer - obsahuje třídy, které se starají o tisk parkovacích lístků
- res - obsahuje obrázky a XML soubory s grafickým prostředím a texty použitými v aplikaci

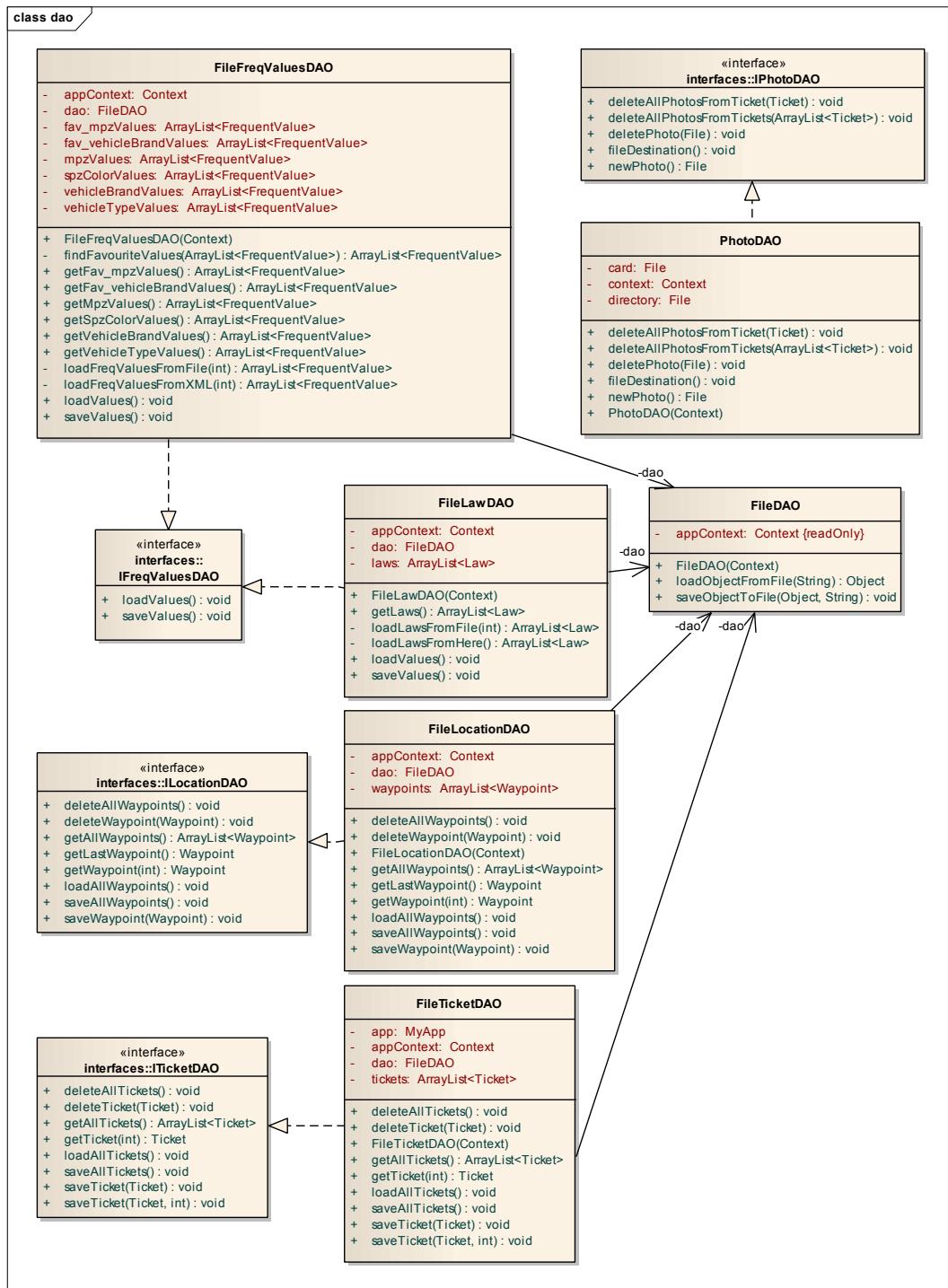


Obrázek 4.6: Balíček tříd - model

Pro zachování přehlednosti jsou zde uvedeny pouze nejzákladnější změněné diagramy, stejně jako tomu bylo v iteraci 1. Diagram 4.7 ukazuje strukturu balíčku android. Obsahuje třídy typu Activity, Application a Service, které mají funkci controlleru. Nově vznikla třída LocationService typu Service, která slouží ke kontinuálnímu snímání polohy zařízení na pozadí.



Obrázek 4.7: Balíček tříd - android



Obrázek 4.8: Balíček tříd - dao

Diagram 4.8 ukazuje aktuální strukturu balíčku dao. Byly přidány rozhraní a jejich implementace pro kontrolu nad daty zákonů, nejčastějších hodnot a fotografií. Všechny (až

na zvláštní případ fotografií) využívají jako uložiště disk telefonu, kde se data persistentně ukládají. Struktura tříd je navržena tak, že lze kdykoliv snadno změnit uložiště např. na databázi pouze novou implementací daného rozhraní.

Změny nastaly i ve struktuře balíčku model. Diagram 4.6 ukazuje jeho aktuální podobu. Nově vznikly datové entity FrequentValue, která reprezentuje častou hodnotu, položku v seznamech nápovedy a Waypoint, která představuje geologační bod, aktuální pozici ve formátu GPS souřadnic. V diagramu jsou kvůli přehlednosti vynechány všechny nedůležité GET a SET metody.

4.4.1 Oblíbené položky v nápovědě

Jak již bylo zmíněno výše, došlo ke změně funkčního požadavku ohledně položek nápovedy při vytváření nových parkovacích lístků. Seznamy jsou jasně určeny v XML souborech. Při první inicializaci se z nich data načtou a už jako objekty se uloží do souboru, ze kterého jsou při dalším spuštění aplikace načítány. Přes nastavení aplikace se uživatel dostane k seznamu všech nejčastějších hodnot, kde může položky označovat za oblíbené a naopak. Data jsou zobrazena v listu, kde každá položka (řádek) je opět vytvořena pomocí vlastního View (text a zaškrťávací políčko), podobně jako tomu je v seznamu všech parkovacích lístků. Proto musel být k naplnění listu použit objekt adaptéra, který z objektu časté hodnoty vezme její textovou reprezentaci a napiše jí do položky seznamu. Problém nastal v momentě implementace filtrování seznamu z formuláře v horní části obrazovky. Jelikož mají položky seznamu nestandardní tvar, nešel použít standartní filtr a musel být implementován vlastní. Ten funguje následovně. Po zadání znaků do formuláře se zavolá posluchač, který překontroluje celý seznam. Nejdříve porovnává textovou reprezentaci každé položky jako celku, a v případě neúspěchu rozdělí text na slova (podle mezer) a kontroluje jednotlivá slova. Když celá položka, nebo alespoň nějaké slovo z položky, začíná stejně, jako text ve formuláři, tak se položka zobrazí ve filtrovaném seznamu.

4.4.2 Komunikace se serverem

Aplikace potřebuje pro správný chod často komunikovat se serverem. Parkovací lístky lze sice vytvářet, upravovat a tisknout bez připojení k internetu a bez přístupu na server, kontrola parkovaní zaplaceného pomocí SMS, nebo odesílání lístků na server se bez toho ale neobejde.

Pro účely komunikace se serverem je vytvořena speciální síťová vrstva, která nabízí aplikaci prostředky pro odesílání a stahování dat ze/na server. Síťovou vrstvu klientské aplikace poskytuje kolega Pavel Brož v rámci své bakalářské práce [12]. Komunikovat se serverem je potřeba při přihlašování a odhlašování z/do aplikace, při odesílání geolokačních dat a parkovacích lístků, při ověřování stavu parkovací karty opravňující parkovat v modré zóně a kontrole parkování zaplaceného pomocí SMS.

Vlastní komunikace se serverem funguje ve všech zmíněných případech téměř totožně. Aplikace nejdříve ze všeho ověří, zda je připojena k internetu. Pokud ano, dojde v případě přihlášení k validaci vstupu. Dalším krokem je ověření, zda již aplikace není k serveru připojena. Když ano, využije se aktuální připojení. V případě, že aplikace z nějakého důvodu k serveru připojena není, provede se automaticky připojení a přihlášení s údaji, kterými se

policista přihlašoval do aplikace. Kontrola připojení je nutná z toho důvodu, že když policista aplikaci dlouho nepoužije, nebo pokud jí vypne a zapne, spojení se ukončí. Ukončí se samozřejmě i v případě ztráty internetového připojení. Samotné připojení (v případě ztráty spojení i znova-přihlášení) se vždy provádí v samostatném vlákně, jelikož jde o blokující metody. Aplikace by se „zasekla“ na určitou dobu, což by bylo pro uživatele matoucí. Komunikace v jiném vlákně má také tu výhodu, že během celé operace lze na obrazovce zobrazit dialog informující uživatele o stavu dění a o tom, co má dělat. V našem případě čekat, než se požadavek dokončí. Při aktivním připojení se vytvoří komunikační protokol a odešle požadavky na server. Odpovědi ze serveru chodí skrze posluchače, navržené v síťové vrstvě. Volají se různé metody posluchače podle typu zprávy, např. v případě přerušení spojení se ozve jiná metoda, než v případě potvrzení přihlášení. Při ukončování komunikace se serverem ještě potřeba odpojit protokol. Připojení k serveru jako takové se neruší, lze ho použít příště, pokud nenastane nějaký důvod jeho přerušení, jak bylo již zmíněno výše.

Při použití jiných vláken ale nastává problém, kdy je potřeba upravovat UI obrazovky podle stavu požadavku, u kterého nevíme, kdy přesně skončí. Jde o úpravu dialogu takzvané za běhu, nebo zobrazování upozornění apod. Nová vlákna totiž v systému Android nemohou upravovat UI, pouze ta, která ho sama vytvořila. K tomuto účelu se používá objekt třídy Handler, přes který dokáže nové vlákno posílat zprávy typu Message do vlákna, ve kterém je objekt hanleru vytvořen. Tímto způsobem lze posílat jednoduché datové typy ale i serializovatelné objekty. Jednoduše lze tak vlákno, ve kterém byl objekt Handler vytvořen, upozornit, že proběhla nějaká akce a že může zobrazit notifikaci.

4.4.3 Kontrola SMS parkování, kontrola parkovací karty, tisk parkovacích lístků

Kontrola SMS parkování a kontrola karty opravňující parkovat v modré zóně mají prakticky totožnou implementaci. Po připojení na server se odešle požadavek na informaci o vozidle/kartě, která se po přijetí zobrazí na obrazovce telefonu. Policista tedy okamžitě ví, zda má vozidlo povolení parkovat, nebo jestli byla parkovací karta odcizena.

Tisk parkovacích lístků probíhá velmi podobně. Před samotným odesíláním lístků se nejdříve odešlou geolokační data. Po úspěšném odeslání se odpojí protokol pro geolokační data a otevře se protokol na posílání parkovacích lístků. Ty se posílají po jednom. další lístek se pošle vždy až v momentě, kdy přijde potvrzení od serveru o přijetí toho předešlého. Kdyby se v průběhu posílání dat přerušilo spojení, tak neodeslané parkovací lístky zůstanou bez problému uloženy v telefonu do dalšího odeslání. Během odesílání parkovacích lístků je zobrazen dialog, který uživatele informuje, kolik lístků již bylo úspěšně odesláno a kolik jich ještě zbývá.

4.4.4 Přihlašování

Přihlašování je trojího typu. Přihlášení do aplikace (přihlášení k serveru), ověření identity při vytváření nebo editaci parkovacích lístků a ověření identity při nahrávání dat na server.

Když chce policista použít aplikací, musí se na začátku své služby přihlásit k serveru. Bez přihlášení se k funkcím aplikace nelze dostat. Po přihlášení už dále může aplikaci bez problému používat i bez připojení k internetu (samořejmě pouze funkce, které nevyžadují

komunikaci se serverem), bude fungovat zcela normálně. Na konci služby by neměl zapomenout se odhlásit, což opět znepřístupní funkce aplikace.

Podobně funguje i ověření identity uživatele při nahrávání dat na server. To může učinit i jiný policista, než ten, co je momentálně přihlášen k aplikaci. K k odeslaným datům se přidá služební číslo policisty, který nahrání provedl. Opět se zkонтroluje, zda je telefon připojen k internetu, zvaliduje se vstup a v případě bezproblémového postupu se aplikace pokusí přihlásit.

Třetí způsob přihlašování je autentifikace při práci s parkovacími lístky. Vytvářet nové a upravovat stávající může pouze ten policista, který je přihlášen k aplikaci. Ověření proběhne bez použití serveru, pouze s údaji uloženými v preferencích aplikace.

4.4.5 Sledování aktuální polohy

Nový požadavek na kontinuální sledování polohy vyžaduje trošku jiný přístup, než implementace dosavadních obrazovek, jelikož poloha musí být snímána po celou dobu služby, tedy i když je aplikace vypnutá. Řešením bylo vytvoření objektu třídy Service, což je prakticky služba, která provádí určitý kód a nemá přitom žádné uživatelské prostředí (na rozdíl od tříd typu Activity).

Systém Android má možnost v rámci „úklidu“ při nedostatku paměti možnost kdykoliv ukončit jakoukoliv aplikaci, která běží na pozadí. Služba kontinuálního zjišťování polohy se spustí na pozadí v novém vlákně, aby nepřekážela při běžné práci s aplikací. Spuštění probíhá v takzvaném sticky módu, který zajistí, že pokud dojde k ukončení služby systémem, bude po chvilce opět spuštěna. To zajistí prakticky nepřetržitý běh služby, dokud ji aplikace sama neukončí, nebo nedojde k restartování telefonu. Při každém spuštění aplikace se zkонтroluje, zda je služba aktivní. Pokud není, spustí se dodatečně. Tím je zajistěno, že nikdy nemůže nastat situace, kdy by běžela aplikace Smart-Fine, ale lokální služba nikoliv.

Ke zjištění aktuální polohy je potřeba mít aktivní alespoň jednoho poskytovatele polohy (dále NET nebo GPS provider). Při přihlašování aplikace zkонтroluje, zda je nějaký povolen. V případě, že žádný aktivní není, vyzve uživatele a přesměruje ho do nastavení, aby alespoň jeden (nejlépe oba) povolil. Do aplikace se nelze přihlásit bez alespoň jednoho aktivního providera. Problém ale nastane, když uživatel vypne prividera až po tom, co se do aplikace přihlásí. Tomu bohužel nejde zabránit, softwarově nelze nijak nařídit povolení poskytovatele, to musí učinit sám uživatel v nastavení systému. Takto je Android OS navržen. Celý problém povolování a blokování providerů je poněkud komplexní, a je vyřešen dále.

Pro získání nové polohy se musí pomocí objektu typu LocationManager požádat u jednotlivých providerů o odběr. Nová poloha je vrácena pomocí posluchače, který se zavolá v případě, že je nalezena. Odebírání aktuální polohy trvá do doby, než se u objektu LocationManager požádá o zrušení odběru. Při registraci odběru polohy se nastaví, že chceme odebírat vždy cca po minutě a půl, z důvodu šetření baterie. Vedle posluchače pro novou polohu existuje také posluchač změny statusu providera. Ten se zavolá v případě, že se provider, u kterého je registrován odběr polohy, vypne, nebo zase zapne. Bohužel, posluchač změny statusu providera přestane fungovat v momentě, kdy se u něj zruší odběr, což je jediný způsob, jak zamezit jeho používání.

Je nutné podotknout, že pro získání polohy z NET providere, je potřeba mít připojení na internet! Provider totiž může být aktivní i bez něj, v tom případě ale neustále vrací poslední zjištěnou polohu, která už je v momentě obdržení neplatná.

Chceme, aby telefon využíval providera následujícím způsobem. Pokud je aktivní jak NET (internetový), tak GPS provider, a zároveň je k dispozici připojení na internet, použijeme NET. Ten je energeticky prakticky nenáročný, na rozdíl od GPS, která spotřebuje spousty energie. NET poloha sice není zdaleka tak přesná, jako poloha z GPS, nicméně v našem případě nepotřebujeme zkoumat exaktní polohu. Pokud je aktivní pouze NET provider, také není problém. Pokud je aktivován pouze GPS, nebo je aktivován i NET ale není dostupné připojení k internetu, musí se přepnout na GPS providera. To znamená odhlásit odběr od NET, aby provider nevracel neplatnou polohu, a zaregistrovat odběr u GPS. Jakmile se opět zpřístupní NET provider a obnoví internetové připojení, odregistruje se odběr u GPS a přihlásí opět na NET, aby se šetřila baterie. V případě, že není aktivní ani jeden provider, služba nedostává žádné nové polohy.

Postup je naprostě jasný, ale jeho implementace je složitá. Jakmile se vypne provider, u kterého je přihlášen odběr, zavolá se skrze posluchač změny statusu nastavovací metoda, která zjistí, jak jsou na tom celkově provideri, a nastaví správného podle postupu uvedeného výše. Přitom také odregistruje vypnutého providera, bychom v případě NET, nedostávali špatné polohy, a v případě GPS, šetřili baterii. Např. Když je aktivní NET i GPS, a uživatel vypne NET, posluchač zavolá nastavovací metodu, která odregistruje NET a zaregistruje GPS. Pokud ale uživatel znova zapne NET, posluchač změny statusu se už nezavolá, protože byl odregistrován. To by musel nejdříve zapnout NET a pak vypnout GPS, kde se z posluchače změny statusu GPS providera zavolá nastavovací metoda a opět nastaví NET.

Po několikadenním řešení a ladění tohoto a podobných problémů byl nakonec problém kontroly providerů vyřešen následujícím způsobem. Nastavovací metoda se zavolá vždy, když dojde ke změně statusu registrovaného posluchače a vždy, když služba získá novou aktuální polohu. V případě, že se služba dostane do bodu, kdy není aktivní ani jeden provider polohy, spustí se Runnable se zpožděním, který po vypršení časového intervalu znova zavolá nastavovací metodu. Tak pořád do kola, dokud uživatel nepovolí alespoň jednoho poskytovatele. Zpožděný Runnable se spustí také v případě, že je aktivní GPS provider. To je z důvodu sledování, zda uživatel nezapne ještě NET. V tom případě služba přepne raději na NET poskytovatele.

4.4.6 Tisk parkovacího lístku

Na trhu existuje spousta mobilních tiskáren, které by se hodily pro účel tisku parkovacích lístků. Bohužel, ne všechny jsou dostupné v České republice včetně podpory, a jsou také poměrně drahé. Ceny se pohybují (v době psaní práce - duben/květen 2012) od 11000 - 20000 Kč. K dispozici jsou i levnější tiskárny, ty ale nedisponují možností bezdrátové komunikace, což je pro naše účely důležitý faktor.

Pro tisk parkovacích lístků byla nakonec vybrána mobilní tiskárna SM-T300 od firmy Star [16], která je schopná s mobilním telefonem komunikovat přes Bluetooth, nepatří k těm nejdražším, a lze k ní v České republice bez problému dokoupit spotřební materiál - termo papír. Firma Star navíc ke svým tiskárnám nabízí k volnému stažení a použití SDK [17],

které obsahuje ukázkovou aplikaci a knihovnu určenou pro Android, která slouží přímo ke komunikaci s tiskárnou.

Při implementaci tisku parkovacích lístků vznikl obrovský problém. Aplikace, dokonce i ta ukázková ze Star SDK, neustále padala a nefungovala. Po několika dnech zkoumání bylo zjištěno, že problém je v importované knihovně. Projekt knihovnu vidí a bez problémů využívá její funkcionality, avšak po nahrání aplikace do telefonu, jakoby knihovna přestala fungovat. Problém nastal vždy, když se program dostal v kódu do části, kde měl využít importovanou knihovnu. Nakonec stačilo pouze knihovnu vložit do projektu způsobem popsaným zde [4], a vše fungovalo jak má.

Samotný tisk probíhá následovně. Policista při zobrazeném detailu parkovacího lístku stiskne tlačítko tisku. Aplikace zkонтroluje, zda má mobilní telefon vestavěný a aktivní Bluetooth. Pokud je Bluetooth vypnutý, aplikace vyzve uživatele, aby ho zapnul a přesměruje ho do nastavení mobilního telefonu. V momentě, kdy je Bluetooth aktivní aplikace otevře port pro komunikaci s tiskárnou a pokusí se parkovací lístek vytisknout. Aplikace se automaticky připojí k poslední připojené tiskárně. Pokud bylo k telefonu přes Bluetooth připojeno více tiskáren, lze si vybrat tím, že v nastavení aplikace zadáme MAC adresu vybrané tiskárny.

Tiskárna ovládá několik způsobů tisku. Kromě celé řady čárkových kódů tiskne i obrázky a text. Zde ovšem nastal problém, tiskárna při tisku obyčejného textu neovládá českou diakritiku. Řešením je vytisknout parkovací lístek jako obrázek. Tisk je sice pomalejší a složitější, nicméně je to jediná možnost, jak donutit tiskárnu tisknout české znaky. Parkovací lístek je tedy před každým tiskem doslova „nakreslen“ pomocí objektů Bitmap a Canvas. V průběhu tisku se na obrazovce zobrazí dialog, který upozorňuje uživatele, že má počkat. Tisk je volán z jiného vlákna, jelikož metoda tisku je blokující. Kdyby tomu tak nebylo, docházelo by k „zamrzání“ aplikace, uživatelského prostředí.

4.5 Testování

TODO

4.6 Závěr

Poslední iterace, která trvala zhruba 1 měsíc, byla nejnáročnější ze všech. Během poměrně krátkého úseku musely být implementovány nejobtížnější části systému a doladěna celá aplikace. Celý kód byl rádně zdokumentován pomocí JavaDoc. Během iterace se podařilo splnit všechny zbylé funkční požadavky, které se nestihly v předešlých iteracích, je tedy označena za úspěšnou. Jediný požadavek, který nebyl splněn, je automatické rozpoznání SPZ vozidla z fotografie, který byl zrušen v kapitole 3. Tento požadavek bude doporučen do dalšího vývoje, již mimo tuto bakalářskou práci.

4.6.1 Splněné funkční požadavky:

F.Req.2.: Návod při vyplňování čísel paragrafů zákonů nejběžnějších přestupků

2. Uživatel si bude moci přidat vlastní přestupky

4. Uživatel si bude moci přidat vlastní volby v položkách: MPZ, Barva SPZ, Druh vozidla, Tovární značka
5. Uživatel si bude moci vybrat oblíbené hodnoty k zobrazení v nápovědě z položek MPZ vozidla a tovární značka vozidla.

F.Req.5.: Odeslání parkovacích lístků z mobilního zařízení na server

F.Req.6.: Tisk parkovacích lístků pomocí mobilní tiskárny

F.Req.10.: Kontrola odcizení přenosné parkovací karty opravňující parkovat v modré zóně

F.Req.11.: Přístup k funkcím aplikace po zadání identifikačního čísla

F.Req.12.: Kontrola parkovaní zaplaceného přes SMS

F.Req.13.: Sledování a evidence aktuální polohy mobilního zařízení

Kapitola 5

Závěr

5.1 Zhodnocení provedené práce

Cílem bakalářské práce bylo vytvořit fungující systém, který by policistům usnadňoval každodenní práci a zjednodušoval rutinní úkony. Toho mělo být dosaženo za pomocí mobilních technologií a komunikačních služeb, které jsou v dnešní době přístupné široké veřejnosti.

Podařilo se vyvinout systém, který toto umožňuje. Skládá se z klientské aplikace, nahrané na mobilním telefonu, která je schopná bezdrátově komunikovat se serverem na centrále, a může tak okamžitě poskytovat informace o tom, co se děje ve službě policisty. Další částí je server, který naopak může policistovi v terénu ihned poslat informace, potřebné k výkonu jeho služby, aniž by policista musel jezdit na centrálu, či mít zdroj informací celou dobu u sebe. Poslední částí je PC klient na centrále, který policistům usnadní práci s papírováním, skrze informace zaslané z mobilního klienta.

Obsahem této bakalářské práce byl však pouze mobilní klient, jehož vývoj byl úspěšný. Podařilo se, až na jeden požadavek, splnit veškerou očekávanou funkcionality. Klient bez problému dokáže vytvářet parkovací lístky, upravovat je a tisknout na mobilní tiskárnu. Úspěšně komunikuje se serverem, na který dokáže parkovací lístky odeslat, takže jsou hned k dispozici. Umožňuje zkontolovat platnost parkování, které řidič zaplatil pomocí SMS. Aplikace také umí ověřit, zda není parkovací karta, kterou řidič používá pro parkování v modré zóně, označena za odcizenou.

Síťová vrstva a serverová část byly vytvořeny v rámci bakalářské práce kolegy Pavla Brože, který byl ve splnění svých cílů rovněž úspěšný.

5.2 Osobní zhodnocení

Práce byla poměrně rozsáhlá ne zrovna triviální co se implementace týče. Byla také velmi náročná na čas, jelikož je to vůbec poprvé, co jsem vyvíjel aplikaci pro operační systém Android. I přes to, že jde o programovací jazyk Java, se kterým jsem se v minulosti mnohokrát setkal, byla pro mne spousta věcí naprostě nová, z toho důvodu, že aplikace na tomto operačním systému fungují trošku jinak, než např. na Windows.

I přes to, že jsem častokrát narazil při vývoji na četná úskalí, kvůli kterým mě práce nebavila, a nechtěl jsem jí dělat, jsem ve výsledku s prací spokojen. Byla pro mne velkým

přínosem. Naučil jsem se nové technologie, použití Javy na jiném zařízení, než je desktopový počítač, či notebook. Lépe jsem porozuměl použití technologií jako např. GPS a naučil se brát v potaz dotykové prostředí při vývoji uživatelského prostředí. Jelikož jsem dlouhodobým majitelem telefonu se systémem Android, práce pro mne byla pohodlná a dokázal jsem se na aplikaci dívat očima „obyčejného uživatele“.

5.3 Budoucí vývoj

Splněním cílů bakalářské práce ale vývoj systému Smart-Fine nekončí. K tomu aby byla aplikace nasazení-schopná, je potřeba ještě udělat kus práce.

V budoucím vývoji by měla být doladěna implementace stávající části. Zcela jistě existuje mnoho bugů, které zatím nebyly objeveny. Další vývoj by se měl také zaměřit na vylepšení grafického uživatelského prostředí. Aplikaci by neškodil ucelený design, který nemusí využívat systémových prvků (např. vlastní vzhled tlačítek apod.).

Měla by přibývat další funkcionality. Aplikace by měla umožňovat rozpoznávání SPZ značek vozidel z fotografie, případně přímo z kamery a napomáhat tak při vyplňování parkovacích lístků. Jde o nesplněný požadavek z minulého vývoje. Aplikace se nakonec může zabývat i vypisováním blokových pokut a řešením složitějších dopravních přestupků. Ověřování nemusí končit u SMS parkování a parkovacích karet, policista by mohl mít skrz mobilní telefon přístup k seznamu hledaných vozidel, klidně i lidí. Rychlý přístup k telefonům čísel např. odtahové služby, rychlý přístup k článcům zákona, které se zabývají dopravními přestupky. Dotykový display telefonu může sloužit k elektronickým podpisům, atd.

Jistě je existuje spousta možností, jak systém zdokonalit a učinit užitečnějším.

Literatura

- [1] A7B36SI2 – Řízení softwarových projektů.
<https://edux.feld.cvut.cz/courses/A7B36SI2/start>, stav z 1. 5. 2012.
- [2] Android SDK – Knihovny pro vývoj pro Android os.
<http://developer.android.com/sdk/index.html>, stav z 1. 5. 2012.
- [3] Balsamiq Mockup – Návrh uživatelského prostředí.
<http://www.balsamiq.com/products/mockups>, stav z 1. 5. 2012.
- [4] Řešení importu knihovny JAR do projektu Eclipse.
<http://stackoverflow.com/questions/10031628/android-zip4j-1-3-1-jar-throws-verifyerr>
stav z 10. 5. 2012.
- [5] Enterprise Architect – Modelování v UML.
<http://www.sparxsystems.com.au>, stav z 1. 5. 2012.
- [6] Ide eclipse – Vývojový nástroj programování.
<http://www.eclipse.org>, stav z 1. 5. 2012.
- [7] Tesjeract - JNI wrapper for Tessaract OCR.
<http://code.google.com/p/tesjeract/>, stav z 10. 5. 2012.
- [8] Tesseract-ocr - Optické rozpoznání znaků.
<http://code.google.com/p/tesseract-ocr/>, stav z 10. 5. 2012.
- [9] Testování a hodnocení - Heuristická analýza.
<http://human-computer-interaction.webnode.cz/testovani-a-hodnoceni-rozhrani/metody-t>
stav z 10. 5. 2012.
- [10] Use case diagram - diagram případů užití.
<http://mpavus.wz.cz/uml/uml-b-use-case-3-2-1.php>, stav z 10. 5. 2012.
- [11] ERIKA a.s. SMS parkovné.
<http://www.erika-as.cz/index.php/cz/sms-municipal/sms-parkovne>, stav z 10. 5. 2012.
- [12] Pavel Brož. Bakalářská práce - Systém pro evidenci přestupků pomocí mobilních telefonů - serverová část.
<http://code.google.com/p/smart-fine>, stav z 1. 5. 2012.

- [13] CzechDroid. OCR test.
<https://play.google.com/store/apps/details?id=edu.sfsu.cs.orange.ocr&hl=cs>, stav z 10. 5. 2012.
- [14] Martin Dráb. Iterativní vývoj.
<http://dev.goshoom.net/2011/09/iterativni-vyvoj/>, 24. 9. 2011.
- [15] Ondřej Martinský. JAVA Anpr - Automatické rozpoznání poznávací značky.
<http://javaanpr.sourceforge.net/>, stav z 10. 5. 2012.
- [16] Star Micronics. Mobilní tiskárna SM-T300.
<http://www.starmicronics.com/printer/PrinterDesc.aspx?PageId=14&PrinterId=105>, stav z 12. 5. 2012.
- [17] Star Micronics. Sdk pro operační systém Android pro komunikaci s mobilní tiskárnou.
<http://www.starasia.com/MobileDriverdownload.asp>, stav z 10. 5. 2012.
- [18] Martin Štajner; Pavel Brož. Smart-Fine - systém evidence dopravních dopravních přestupků a pohybu policistů pomocí mobilních telefonů.
<http://code.google.com/p/smart-fine>, stav z 1. 5. 2012.
- [19] ZEBRA Technologies. Mobilní tiskárny.
<http://www.zebra.com>, stav z 10. 5. 2012.
- [20] Robert Theis. LPlateEU ANPR - Automatické rozpoznání poznávací značky.
<https://play.google.com/store/apps/details?id=com.phosil.LP.EU&hl=cs>, stav z 10. 5. 2012.

Příloha A

Přehled o projektu

Verze přílohy z iterace 1 a 2 jsou k dispozici na CD a na stránkách projektu [18]

A.1 Požadavky na systém

A.1.1 Obecné požadavky

N.Req.1.: Mobilní zařízení (mobilní telefon)

1. Operační systém Android ve verzi min. 2.1
2. Integrovaný fotoaparát
3. Schopnost připojení na internet - WiFi nebo jiné mobilní datové přenosy nebo kombinace obou
4. Integrované nebo externí GPS (v případě externího připojení nezajišťuje Smart-Fine)
5. Integrované Bluetooth

N.Req.2.: Mobilní tiskárna s datovým připojením přes Bluetooth

N.Req.3.: Přístup k serveru Smart-Fine

A.1.2 Funkční požadavky

F.Req.1.: Vyplnění parkovacího lístku

1. Vyplnění informací o datu a času přestupku, SPZ, MPZ, barvě SPZ, druhu vozidla, tovární značky, města, ulice, čísla, místa, identifikačního čísla, přestupku (paragraf, odstavec, písmeno, zákon, sbírka, popis) a možnost výběru odtahu, přenosné DZ a fotodokumentace.
2. Všechny vyjmenované informace (kromě výběrových) bude možné zadat za pomocí softwarové klávesnice nebo hardwarové klávesnice integrované do mobilního zařízení.

3. Systém bude standardně předvyplňovat údaje: datum, čas, město, identifikační číslo policisty.
4. Systém bude volitelně a podle aktuálních možností mobilního zařízení před-vyplňovat údaje: adresa, SPZ.
5. Systém bude vyplněné parkovací lístky ukládat do paměti mobilního zařízení.

F.Req.2.: Návod pro vyplňování čísel paragrafů zákonů nejběžnějších přestupků

1. Systém bude uživateli nabízet výčet nejběžnějších přestupků a po vybrání uživatelem sám vyplní příslušná čísla zákonů a popis do parkovacího lístku.
2. Uživatel si bude moci přidat vlastní přestupky.
3. Systém bude uživateli nabízet výčet nejběžnějších voleb v položkách: MPZ, Barva SPZ, Druh vozidla, Tovární značka.
4. Uživatel si bude moci přidat vlastní volby v položkách: MPZ, Barva SPZ, Druh vozidla, Tovární značka.
5. Uživatel si bude moci vybrat oblíbené hodnoty k zobrazení v návodě z položek MPZ vozidla a tovární značka vozidla.

F.Req.3.: Prohlížení lokálně uložených parkovacích lístků

1. Systém bude umožňovat prohlížet lokálně uložené parkovací lístky od posledního odeslání na server.

F.Req.4.: Úprava lokálně uložených parkovacích lístků

1. Systém bude umožňovat úpravu lokálně uložených parkovacích lístků.

F.Req.5.: Odeslání parkovacích lístků z mobilního zařízení na server

1. Systém bude umožňovat odeslání vyplněných a vytisknutých parkovacích lístků na určený server přes Internet.
2. Odesílaná data budou chráněna proti zneužití šifrováním.
3. Pro odeslání parkovacích lístků bude nutná autorizace uživatele za pomocí služebního čísla a hesla.
4. Systém bude odeslané parkovací lístky odstraňovat z mobilního zařízení.

F.Req.6.: Tisk parkovacích lístků pomocí mobilní tiskárny

1. Systém bude umožňovat tisk vyplněného a uloženého parkovacího lístku na mobilní tiskárně.
2. Tisk bude probíhat přímo z aplikace Smart-Fine
3. Mobilní tiskárna bude moci být připojena bezdrátově (WiFi nebo Bluetooth).

F.Req.7.: Pořízení fotodokumentace přestupku

1. Systém bude umožňovat při vyplňování parkovacího lístku vytvořit fotodokumentaci přestupku.

2. Fotodokumentaci bude možné pořídit pomocí před-instalované aplikace fotoaparátu v systému mobilního telefonu.
3. Fotodokumentaci bude možné přidat vybráním předem nařízených fotografií ze před-instalované aplikace galerie v systému mobilního telefonu.
4. Systém bude fotodokumentaci ukládat do externí paměti (na paměťovou kartu).

F.Req.8.: Rozpoznání státní poznávací značky z fotografie

1. Systém bude umožňovat vyplnit SPZ jejím vyfocením.
2. Systém bude převádět obrazovou informaci o SPZ na text.

F.Req.9.: Automatické vyhledání adresy pomocí GPS

1. Systém bude umožňovat automaticky vyhledávat a předvyplňovat adresu - město, ulici, číslo popisné, kde se nachází mobilní zařízení se systémem Smart-Fine.
2. K vyhledání bude použito systému GPS, nebo seznamu BTS (určení polohy pomocí internetu).
3. Podmínkou pro fungování této funkce je dostupnost internetového připojení na mobilním zařízení.

F.Req.10.: Kontrola odcizení přenosné parkovací karty opravňující parkovat v modré zóně

1. Vyplnění čísla karty.
2. Systém bude umožňovat online porovnání čísla zadané parkovací karty se seznamem odcizených přenosných parkovacích karet.

F.Req.11.: Přístup k funkcím aplikace po zadání identifikačního čísla

1. Systém bude umožňovat přístup k funkcím aplikace pouze po přihlášení pomocí identifikačního čísla a hesla (PIN).
2. Některé funkce budou vyžadovat dodatečné ověření pomocí identifikačního čísla.

F.Req.12.: Kontrola parkovaní zaplaceného přes SMS

1. Vyplnění SPZ vozidla.
2. Systém bude umožňovat online ověření vozidla pomocí SPZ, zda dané vozidlo zaplatilo parkování. Pokud ano, v jakém časovém intervalu má povolení parkovat.

F.Req.13.: Geolokační data se odešlou spolu s parkovacími lístky na server na jednou

1. Sledování a evidence aktuální polohy mobilního zařízení
2. Systém bude od přihlášení až do odhlášení snímat aktuální polohu mobilního zařízení.

A.2 Spolehlivost

Aplikace nevyžaduje žádná speciální opatření v oblasti spolehlivosti, protože kdykoliv během výpadku je možno opět použít papírový parkovací lístek. Při selhání systému tak nevzniká žádná větší škoda. Během vývoje však bude kladen důraz na odladění aplikace tak, aby nedocházelo k selhání systému, a dále bude zajištěno zálohování serveru pro ukládání parkovacích lístků pro případ nečekaného selhání.

A.3 Možné překážky a rizika

A.3.1 Při vývoji

Riziko:	Ztráta dat na počítači
Předcházení:	Zálohování dat nahráním na SVN
Řešení:	Obnova dat z SVN

A.3.2 Při nasazení

Riziko:	Ztráta dat v důsledku selhání serverových datových úložišť
Předcházení:	Pravidelná záloha dat a kontrola stavu serveru (stav disků skrze S.M.A.R.T., HW stav) a případná výměna komponent
Řešení:	Obnova dat ze zálohy

Riziko:	Ztráta dat v důsledku selhání serverových datových úložišť
Předcházení:	Časté nahrání lokálně uložených parkovacích lístků na server
Řešení:	Vzhledem k náročnosti a ceně obnovy dat z poškozeného zařízení u specializované firmy jsou data na poškozeném mobilním zařízení považována za ztracená.

Riziko:	Ztráta mobilního zařízení
Předcházení:	
Řešení:	Považovat data na mobilním zařízení za ztracená a donutit uživatele ke změně přístupového hesla pro nahrávání parkovacích lístků na server.

Příloha B

Analýza

Verze přílohy z iterace 1 a 2 jsou k dispozici na CD a na stránkách projektu [18]

B.1 Případy užití

UC.1.: Vyplnění nového parkovacího lístku

- Uživatel bude schopen vypsat nový parkovací lístek systémem vyplňování formulářových polí. Tímto způsobem bude moct zadat veškeré potřebné informace.

UC.2.: Zobrazení seznamu parkovacích lístků

- Uživatel bude schopen zobrazit seznam všech vyplněných parkovacích lístků od posledního úspěšného nahrání na server.

UC.3.: Prohlížení lokálních záznamů

- Uživatel bude schopen prohlédnout všechny informace již vyplněných parkovacích lístků.

UC.4.: Úprava lokálních záznamů

- Uživatel bude schopen upravit všechny informace již vyplněných parkovacích lístků.
- Po vytisknutí parkovacího lístku ho již nebude možno dále upravovat.

UC.5.: Odeslání lokálních záznamů na server

- Uživatel bude schopen, pokud bude připojen k internetu, odeslat všechny lokální záznamy do databáze na server.
- Odeslat bude možné pouze všechny parkovací lístky na jednou, nikoliv po jednom, z důvodu zamezení možnosti zadržování lístků v zařízení. Jde tedy o způsob jak zajistit, že nedojde ke korupčnímu jednání.

UC.6.: Kontrola odcizení parkovací karty

- Systém bude umožňovat zkontolovat, zda je přenosná parkovací karta odcizená.

UC.7.: Tisk parkovacích lístků

- Uživatel bude schopen vytisknout všechny lokální záznamy na mobilní tiskárně.
- Po vytištění již dále nebude možné parkovací lístek upravit, nebo smazat.

UC.8.: Autentifikace uživatele

- Uživatel se bude muset při přístupu do aplikace a některých funkcí přihlásit pomocí identifikačního čísla a číselného hesla (PIN). Jedná se o bezpečnostní a monitorovací prvek.
- K přihlášení je potřeba internetové připojení. Některé funkce budou umožňovat přihlášení i bez připojení k internetu.

UC.9.: Kontrola parkovaní zaplaceného přes SMS

- Systém bude umožňovat online ověření vozidla pomocí SPZ, zda dané vozidlo zaplatilo parkování. Pokud ano, v jakém časovém intervalu má povolení parkovat

UC.10.: Evidence pohybu policistů

- Systém bude od přihlášení do aplikace až do odhlášení z aplikace snímat a ukládat aktuální polohu mobilního zařízení.
- Data o poloze se odešlou na server spolu se seznamem parkovacích lístků.

B.2 Scénáře případů užití

S.1.: Vyplnění nového parkovacího lístku

1. Uživatel chce vyplnit nový parkovací lístek.
2. INCLUDE (Autentifikace uživatele)
3. Systém zobrazí zadávací formulář.
4. Uživatel vyplní všechna textová pole a vybere druh přestupku z nabídky.
5. Uživatel volitelně nafotí přestupek.
6. Uživatel požádá systém o uložení parkovacího lístku.
7. Systém zkontovaluje validitu všech dat.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
8. Systém uloží parkovací lístek.

S.2.: Zobrazení seznamu parkovacích lístků

1. Uživatel chce zobrazit seznam lokálně uložených parkovacích lístků.
2. Uživatel požádá systém o zobrazení seznamu parkovacích lístků.

3. Systém zobrazí seznam vyplněných parkovacích lístků.

S.3.: Prohlížení lokálních záznamů

1. Uživatel si chce prohlédnout konkrétní parkovací lístek.
2. INCLUDE (Zobrazení seznamu parkovacích lístků)
3. Uživatel požádá systém o zobrazení vybraného parkovacího lístku.
4. Systém zobrazí parkovací lístek v detailu (všechny informace, které při vytváření zadal).

S.4.: Úprava lokálních záznamů

1. Uživatel chce upravit parkovací lístek.
2. INCLUDE (Prohlížení lokálních záznamů)
3. Uživatel požádá o zobrazení editačního formuláře.
4. INCLUDE (Autentifikace uživatele)
5. Systém zobrazí editační formulář.
6. Uživatel upraví požadované informace.
7. Uživatel požádá o uložení parkovacího lístku.
8. Systém zkонтroluje validitu všech dat.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 6.
9. Systém uloží upravený parkovací lístek.
10. Návrat do Prohlížení lokálních záznamů.

S.5.: Odeslání lokálních záznamů na server

1. Uživatel chce odeslat vyplněné parkovací lístky na server.
2. INCLUDE (Zobrazení seznamu parkovacích lístků)
3. Uživatel požádá systém o nahraní všech záznamů na server.
4. Systém zkonzoluje, zdali je zařízení připojeno k internetu.
 - (a) Pokud není zařízení online, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 3.
5. INCLUDE (Autentifikace uživatele)
6. Systém provede nahraní na server.
7. Systém zobrazí hlášku, jak operace dopadla.
 - (a) Pokud se nahraní dat na server nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 3.

S.6.: Kontrola odcizení parkovací karty

1. Uživatel chce ověřit, zda parkovací karta souhlasí s SPZ vozidla.

2. Systém zobrazí zadávací formulář na číslo parkovací karty.
3. Uživatel vyplní číslo parkovací karty.
4. Uživatel požádá systém o ověření čísla parkovací karty.
5. Systém ověří připojení k internetu.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
6. Systém získá informace ze serveru a zobrazí je.
 - (a) Pokud se získání informací nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.

S.7.: Tisk parkovacích lístků

1. Uživatel chce vytisknout parkovací lístek.
2. Uživatel požádá systém o tisk parkovacího lístku.
3. Systém požádá uživatele o potvrzení akce.
4. Uživatel potvrdí akci.
5. Systém zkонтroluje, zdali je zapnutý Bluetooth.
 - (a) Pokud není, vyzve uživatele aby Bluetooth zapnul.
 - (b) Pokračuje se bodem 5.
6. Systém vytiskne parkovací lístek.
7. Systém zobrazí hlášku, jak operace dopadla.
 - (a) Pokud se tisk nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 2.

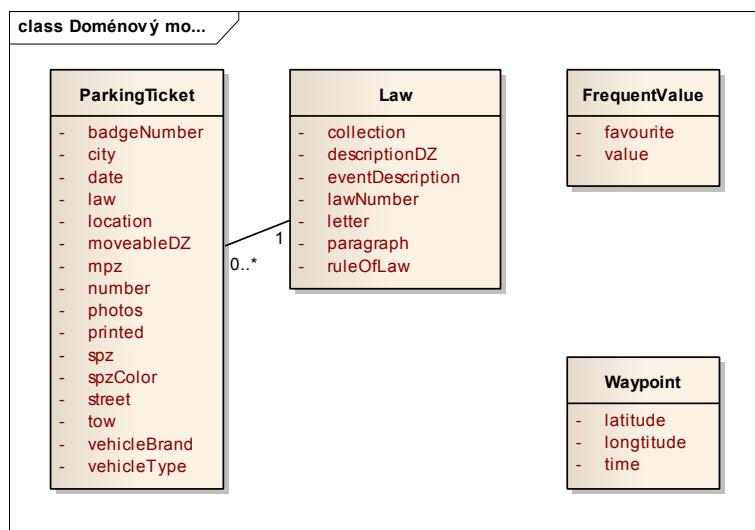
S.8.: Autentifikace uživatele

1. Uživatel se chce přihlásit.
2. Systém uživateli zobrazí přihlašovací formulář.
3. Uživatel se autentifikuje pomocí služebního čísla a hesla.
 - (a) Pokud se autentizace nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 2.

S.9.: Kontrola parkování zaplacенного přes SMS

1. Uživatel chce ověřit, zda má vozidlo zaplacené parkování pomocí SMS.
2. Systém zobrazí zadávací formulář na SPZ vozidla.
3. Uživatel vyplní SPZ vozidla.
4. Uživatel požádá systém o parkování.
5. Systém ověří připojení k internetu.
 - (a) Pokud nalezne chybu, zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.
6. Systém získá informace ze serveru a zobrazí je.
 - (a) Pokud se získání informací nepodaří, systém zobrazí chybovou hlášku.
 - (b) Pokračuje se bodem 4.

B.3 Doménový model



Obrázek B.1: Doménový model

B.3.1 Parkovací lístek - ParkingTicket

Parkovací lístek se všemi potřebnými informacemi.

Atribut	Popis
badgeNumber	Služební číslo - číslo odznaku policisty
city	Město lístků na server.
date	Datum a čas
eventDescription	Popis události
location	Místo události (např.: na chodníku)
moveableDZ	Zda je přenosné DZ
mpz	Mezinárodní poznávací značka
number	Císto ulice
photos	Fotografie události
spz	Státní poznávací značka
spzColor	Barva SPZ
street	Ulice
printed	Zda byl parkovací lístek vytiskněn
tow	Zda odtah vozidla
vehicleBrand	Značka automobilu
vehicleType	Typ automobilu

B.3.2 Odkaz na zákon - Law

Informace, kde lze daný přestupek najít v zákoně.

Atribut	Popis
collection	Sbírka
description	Odstavec
lawNumber	Číslo zákona
letter	Písmeno
paragraph	Odstavec zákona (Nikoliv §)
ruleOfLaw	Paragraf zákona, článek zákona (§)
descriptionDZ	Popis jednání DZ

B.3.3 Častá hodnota - FrequentValue

Položka seznamu nejčastějších hodnot pro nápovědu při vytváření nového parkovacího lístku.

Atribut	Popis
favourite	Zda je hodnota označená jako oblíbená
value	Textová reprezentace hodnoty

B.3.4 Geolokační bod - Waypoint

Geolokační bod nalezený při snímání aktuální polohy zařízení.

Atribut	Popis
latitude	Zeměpisná šířka
longitude	Zeměpisná délka
time	Čas pořízení geobodu

Příloha C

Návrh

Verze přílohy z iterace 1 a 2 jsou k dispozici na CD a na stránkách projektu [18] TODO viz dotaz v emailu.

Příloha D

Seznam použitých zkratek

SI2 Předmět Řízení softwarových projektů A7B36SI2

SDK Software Developer's Kit - Soubor vývojových nástrojů

IDE Integrated Development Environment - Integrované vývojové prostředí

POS Project Overview Statement - Přehled o projektu

PL Parkovací lístek

UML Unified modeling language - Unifikovaný modelovací jazyk

DAO Data Access Object

OCR Optical Character Recognition - Optické rozpoznávání znaků

UI User Interface - Uživatelské rozhraní

GUI Graphic User Interface - Grafické uživatelské prostředí

:

Příloha E

Instalační a uživatelská příručka

TODO

Příloha F

Obsah přiloženého CD

TODO