České vysoké učení technické v Praze
Fakulta stavební

Projekt Informatika 2
Akedemický rok 2012/2013

Mapa kamer – mobilní aplikace pro Android

Dokumentace

Autoři:
Martin Lžíčař
Dan Dluhoš
Michal Med

Vedoucí:
Ing. Martin Landa
Katedra mapování a kartografie
K153

Obsah

Zadání

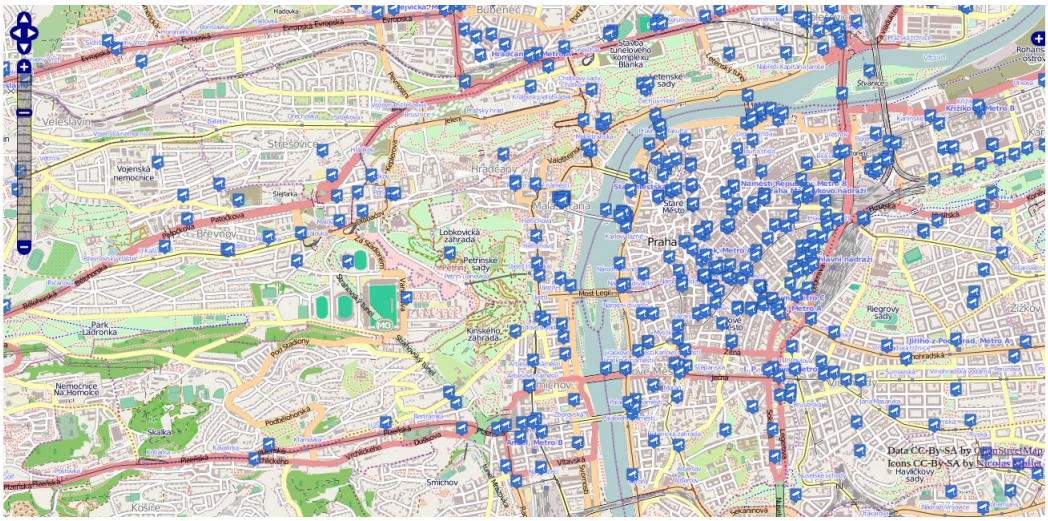
Mobilní aplikace pro mapování kamer ve veřejných prostorech
Skupina: B (2013)

Cílem projektu je tvorba aplikace pro Iuridicum Remedium, o. s., která by měla sloužit k mapování kamer ve veřejných prostorech. Aplikace bude určena pro mobilní zařízení s operačním systémem Android a bude napsána v programovacím jazyce Java. Aplikace by měla umožňovat zobrazení vrstvy s již zmapovanými kamerami ve veřejných prostorech nad OpenStreetMap a přidávání nových kamer pomocí předání souřadnic GPS a fotografie z mobilního zařízení do vrstvy OSM se zmapovanými kamerami. Vkládání bude probíhat přes zprostředkovatelský server, na který se budou ukládat fotografie a který bude obsahovat databázi kamer, do které budou vkládány kamery rovnou z aplikace. Na serveru zprovozníme databázi PostgreSQL a pohyb dat mezi aplikací, databází a OpenStreetMap bude umožněn pomocí PHP skriptů. Budeme vycházet z aplikace, která byla k tomuto účelu naprogramována na restartu iniciovaném občanským sdružením Iuridicum Remedium a upravíme ji podle našich potřeb. V současné aplikaci nefunguje komunikace se serverem, protože žádný server neexistuje, a prohlížečka mapy je založena na Google Maps. Google Maps chceme v aplikaci kompletně nahradit OpenStreetMap.

1 Úvod

Práce byla vytvořena v rámci projektu informatika 2 ve spolupráci s občanským sdružením IuRe – Iuridicum Remedium. Sdružení se zabývá ochranou soukromí a lidskými právy. Na svých stránkách[1] zveřejňují mimo jiné „výherce“ ankety `Big brother awards` nebo webovou aplikaci mapa kamer[2], ve které zobrazují veřejná místa pokrytá kamerovými systémy.

Obrázek 1.1: Logo občanského sdružení IuRe



Obrázek 1.2: Webová aplikace Mapa kamer

Sdružení se již v minulosti pokoušelo o tvorbu mobilní aplikace pro zaznámenávání a zobrazování kamer, ale potýkalo se s nedostatkem pracovních kapacit, proto naši bezplatnou nabídku ke spolupráci rádi přijali. Základ aplikace byl již připraven, ale nesplňoval ani jejich ani naše očekávání. Aplikace byla založena na `Google Maps` a neměla v pozadí ani databázi, ani server. I o to jsme se tedy museli postarat.

[1] www.iure.org
[2] www.mapakamer.cz

Aplikace je zacílena na mobilní operační systém `Android`. Jazykem běžně používaným pro tvorbu aplikací pro `Android` je `Java`. Použili jsme vývojové prostředí `Eclipse`, do kterého je však pro tvorbu mobilních aplikací potřeba doinstalovat `Android Developer Tools`.

Jako podkladová mapa byly použity rastry z opensourcového projektu `OpenStreetMap`[3]. Pro uchovávání informací o kamerách byla použita databáze `PostgreSQL` a dále byla, opět v jazyce `Java`, napsána webová aplikace umožňující komunikaci mezi mobilním zařízením a databází.

[3] www.openstreetmap.org

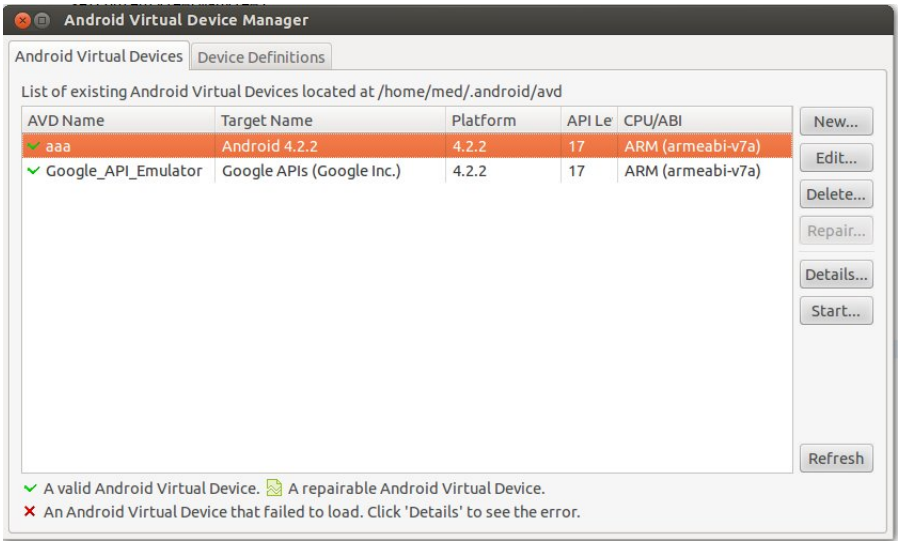
2 Mobilní aplikace Mapa Kamer

Tato kapitola se zabývá tvorbou a používáním mobilní aplikace. V jednotlivých sekcích se zmíníme o přípravě prostředí `Eclipse` pro vývoj mobilních aplikací pro systém `Android`, s používáním projektu `Osmdroid`[1], který slouží k používání map projektu `OpenStreetMap` v `Androidu` a o tom, jak do databáze přidat novou kameru. Na závěr kapitoly se zmíníme o specifikách vývoje aplikací pro systém `Android` jako takových.

2.1 Příprava prostředí Eclipse pro vývoj Android aplikací

Balíček `ADT Bundle` poskytuje vše, co je potřeba k vývoji aplikací pro Android, včetně `Eclipse IDE` s vestavěným ADT (`Android Developer Tools`). Je možné nainstalovat `Android STK` i do existujícího IDE[2], ale zde se budeme zabývat instalací celého balíčku `ADT Bundle`.

Po rozbalení do zvoleného adresáře stačí otevřít adresář adt-bundle-<os_platform>/eclipse/ a spustit z něj `eclipse`. IDE je rovnou načteno s pluginem `ADT` a SDK je připraveno k použití.



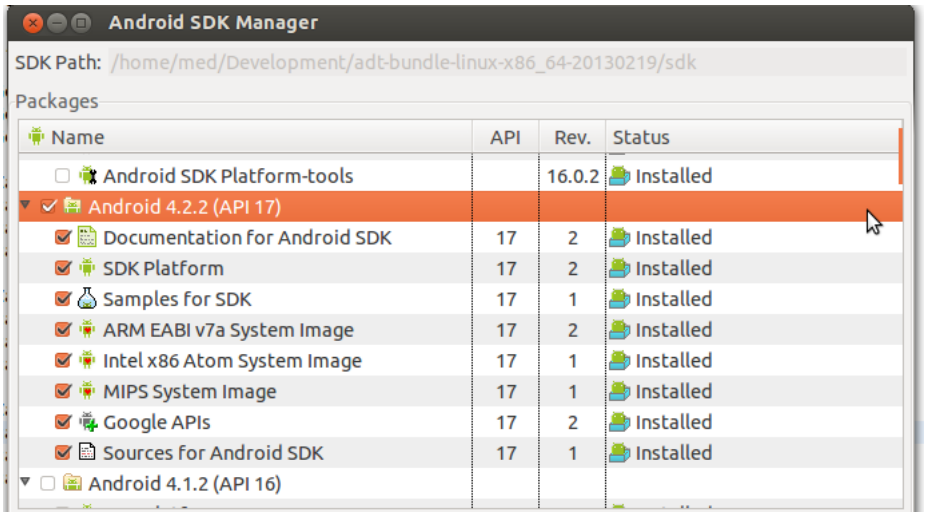
Obrázek 2.1: Správně nastavený emulátoru je na druhém řádku

Balíček obsahuje vše nezbytné včetně emulátoru. Ke správě, spuštění a nastavení SDK slouží položka v menu **Window** → **Android SDK Manager**, ke správě emulátoru potom **Window**

[1] http://code.google.com/p/osmdroid/
[2] http://developer.android.com/sdk/installing/index.html

→ **Android Virtual Device Manager**. K bezproblémovému chodu aplikace v emulátoru je potřeba nastavit jako cílové prostředí `Google APIs` v poslední verzi (17).

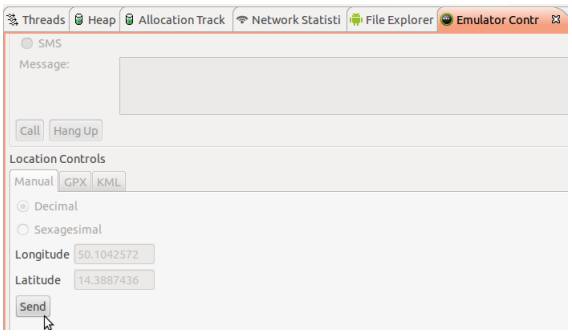
Nejnovější verze `Google API` v nabídce pravděpodobně nebude. K její instalaci je třeba otevřít **Window** → **Android SDK Manager** a doinstalovat ho.



Obrázek 2.2: Označené balíčky je potřeba doinstalovat

Pro spuštění aplikace `Mapa Kamer` v emulátoru je potřeba ještě nastavit souřadnice GPS. To se musí udělat ručně, protože emulátor nemá žádné fyzické GPS zařízení. ze to udělat dvěma způsoby:

- Ruční nastavení v DDMS:
  - **Window** → **Open Perspective** → **DDMS**,
  - označit emulátor a v záložce **Emulator Control** nastavit souřadnice,
  - potvrdit tlačítkem **Send**.



Obrázek 2.3: Označené balíčky je potřeba doinstalovat

- Pomocí telnetu:
  - otevřít konzoli,

– připojit zařízení: `telnet localhost 5554`,

– nastavit souřadnice: `geo fix 13.24 52.31`.

Je celkem jedno který z těchto způsobů je použit. Po správném nastavení souřadnic by se měl emulátor chovat jako mobilní zařízení v poloze nastavené těmito souřadnicemi. **Pozor**: nastavením GPS souřadnic není v zařízení "zapnut"GPS modul. GPS se zapíná v nastavení (**Menu** → **Settings** → **Location access** → **GPS satellites**).

2.2 Projekt Osmdroid

V `Android SDK` existuje mapová podpora pro `Google Maps`, protože `Android` patří také společnosti `Google`. Pro použití `OpenStreetMap` v aplikaci existuje více možností, kromě námi použitého `Osmdroid` lze použít například renderovací knihovnu `MapsForge`[3].

Balíček `Osmdroid` umožňuje používání nativních metod pro zobrazování `Google Maps` k zobrazování dat z projektu `OpenStreetMap`. V některých případech, jako je třeba zobrazení mapy, její vycentrování nebo nastavení hladiny zoom, se používají metody knihovny `Osmdroid` přesně jako ekvivalentní metody z balíčku `google.maps`. V některých jiných případech jsou třídy poněkud odlišné. Příkladem může být třeba zobrazení markerů (třída `ItemizedOverlay`). V některých případech nám možnosti hotových tříd nestačily a požadovanou funkčnost jsme museli dopsat sami. Stalo se nám to například u zobrazení obrázku v informačním okně vyskakujícím při poklepání na marker v mapě. Pro tento účel jsme vytvořili třídu `ImageOverlayItem`.

Od používání `OpenStreetMap` na úkor `Google Maps` jsme si slibovali možnost využití vrstvy `surveillance` z databáze `OpenStreetMap`, která obsahuje poměrně velké množství kamer sledujících veřejný prostor a která byla použita ve webové aplikaci Mapa Kamer a z velké míry i doplňována díky aktivitám sdružení IuRe. K zobrazování vrstvy jsou ve webové aplikaci použity `OpenLayers`, které ale neexistují ve verzi pro mobilní zařízení, jedině v prohlížeči. Pokoušeli jsme se nalézt nějaké řešení jak použít data z této vrstvy a nakonec jsme se rozhodli pro export dat z databáze `OpenStreetMap` do lokální databáze na našem serveru. Odtud je možné data normálně zobrazovat v mobilní aplikaci.

2.3 Používání aplikace

Při spuštění aplikace má uživatel na výběr mezi třemi tlačítky:

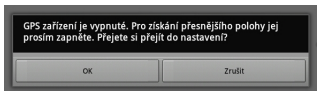
- Přidat kameru,
- Zobrazit mapu kamer,
- O projektu.

2.3.1 Přidat kameru

Nové kamery jsou přidávány přímo do databáze na serveru. Tato databáze se synchronizuje přímo s databází `OpenStreetMap`. O databázi i samotném nahrávání nové kamery po technické stránce se zmíníme ještě v dalších kapitolách. V této sekci popíšeme co a jak dělat v aplikaci aby byla nová kamera odeslána.

Při otevření aktivity aplikace nejprve zkontroluje, zda je zapnutá GPS. Pokud ne, dostane uživatel možnost GPS zapnout a zlepšit tak přesnost zaměření polohy.

[3] http://code.google.com/p/mapsforge/



Obrázek 2.4: Dialog vybízející k zapnutí GPS

Pokud GPS nezapne, bude poloha určena ze sítě. Poloha je zobrazena na řádku `GPS lokace`. Do řádku `popis kamery` uživatel napíše něco o kameře. Mělo by se to týkat faktických věcí, jako například kdo je provozovatelem kamery, upřesnění její polohy nebo například to, jestli kamera pouze zobrazuje, nebo i nahrává záznam. Poslední věc kterou je ke kameře možné přidat je fotografie. Po stisknutí tlačítka je uživatel přesměrován do aplikace fotoaparátu, kterou pořídí snímek. Poté už stačí celý formulář jenom potvrdit a kamera bude přidána do databáze.

2.3.2 Zobrazit mapu kamer

Při spuštění mapy aplikace nejprve zkontroluje, zda je zapnutá GPS. V případě že ano, je na aktuálních souřadnicích vycentrována mapa `OpenStreetMap`. Pokud GPS zapnutá není, zobrazí se stejný dialog jako v předchozí sekci a uživatel dostane možnost zapnout GPS.

Při volbě **Ano** je uživatel přesměrován do nastavení správy GPS. GPS může, ale nemusí být zapnutá. Pro určení polohy z GPS je použit objekt `gpsLocListener`. V případě, že se uživatel rozhodne nechat GPS vypnutou (a zvolí **Ne**), je poloha určována ze sítě pomocí objektu `networkLocListener`.

Obrázek 2.5: Mapa vycentrovaná na místě, na kterém jsem psal tuto dokumentaci

Podklad `OpenStreetMap` je zobrazen a vycentrován na aktuální pozici. Z databáze jsou poté pomocí příkazu `SELECT` vytaženy kamery a zobrazeny s markerem se symbolem cctv[4]. Při poklepání na kameru se otevře informační okno s podrobnostmi o kameře. Kromě popisu může obsahovat i fotografii.

2.3.3 O projektu

V této záložce se uživatel může dočíst jaké cíle má projekt Mapa kamer, na jakých principech je postaven a také něco o sdružení IuRe a tvůrcích projektu.

2.4 Základy vývoje aplikací pro Android

Tvorba aplikace pro systém `Android` je ve své podstatě hrozně jednoduchá. Každá aplikace má jeden xml soubor pojmenovaný `Manifest`. Ten obsahuje název balíčku, verze systému pro které

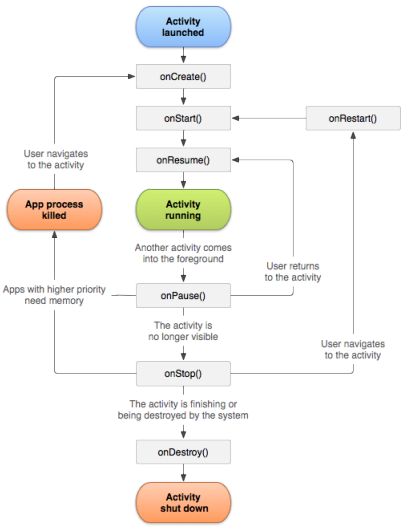
[4] http://mapicons.nicolasmollet.com/markers/offices/cctv/

Obrázek 2.6: Ukázka informačního okna kamery

Obrázek 2.7: Vzhled záložky O projektu (screenshot z tabletu)

je aplikace určená a dále velmi důležitý seznam použitých povolení (`permissions`) a vlastností (`features`). Další věci jsou obsaženy v elementu application. Jedná se především o seznam aktivit používaných v aplikaci. Co je to aktivita je vysvětleno dále.

Aktivitu si lze představit jako jednu obrazovku mobilního zařízení s tlačítky, nadpisy, menu a vším co k tomu patří. V aplikaci Mapa Kamer je například jednou z aktivit `HomeActivity`, která obsahuje domácí obrazovku aplikace se třemi tlačítky. Při poklepání na tlačítka se akorát spouští další aktivity, jmenovitě to jsou `NewCameraActivity`, `CameraMapActivity` a `AboutActivity`.



Obrázek 2.8: Životní cyklus aktivity

Každá aktivita může mít vzhled tvořen buď kódem v jazyce `Java`, nebo je možné vzhled nastavit pomocí xml souboru v `res` → `layout`. V prostředí `Eclipse` je možné xml vytvářet pomocí grafického editoru.

Každá aktivita má svůj životní cyklus. Aktivita může být v jednom ze čtyř základních stavů:

- **Aktivní** – aktivita běží v popředí.
- **Pozastavená** – aktivita je pořád viditelná, ale není aktivní. Chová se stejně jako aktivní aplikace, ale v případě malé paměti zařízení jí systém může zabít.
- **Zastavená** – aktivita je kompletně překryta jinou aktivitou. Když je paměť potřeba jinde, tak jí systém zpravidla zabije.
- **Zabitá** – pokud je pozastavená nebo zastavená aktivita zabita systémem a uživatel ji chce zobrazit, musí být znovu načtena úplně od začátku a uvedena zpět do předchozího stavu.

Každá aktivita má několik metod, kterými je možné ji uvést do některého z uvedených stavů. Metoda uvádějící aktivitu do určitého stavu má párovou metodu, která tento stav ukončí.

- `onCreate` a `onDestroy`
- `onStart` a `onStop`
- `onResume` a `onPaused`

3 Databáze

Jako databáze na které aplikace pojede byla zvolena databáze `PostgreSQL`. Důvody jsou dva:

- Jedná se o open source projekt.
- Projekt `OpenStreetMap` používá jako databázi také `PostgreSQL`.

Obrázek 3.1: Logo projektu PostgreSQL, na kterém slon reprezentuje dobrou paměť

Základem pro databázi bylo schéma, které nám laskavě poskytl pan inženýr Martin Landa. Schéma obsahovalo data `OpenStreetMap` pro Českou Republiku. Ze schématu jsme vytáhli pomocí následujícího příkazu vytáhli všechny body s tagem **surveillance** a uložili je do tabulky `b_kamery`.

```
CREATE TABLE b_kamery AS (SELECT osm_id,name,ST_X(ST_Transform(way::geometry,4326))
AS lat,ST_Y(ST_Transform(way::geometry,4326)) AS lon FROM osm.czech_point WHERE man_made
='surveillance');
```

Tabulka `b_kamery` obsahuje identifikátor, informace o kameře, souřadnice kamery, které byly naimportovány ze schématu `osm` a navíc jsme přidali atribut `image`, který zatím obsahuje adresu obrázku na serveru. Do budoucna plánujeme obrázky ukládat jako BLOB[1], protože při větším množství obrázků by pravděpodobně byl velký problém s pamětí. Z důvodů nevhodného kódování se v informačním okně kamery zobrazuje pouze obrázek a nikoliv popis kamery.

Pro zrychlení vyhledávacích a dotazovacích procesů v databázi bude záhodno databázi naindexovat. Indexy v databázi by ji mohli výrazně zrychlit a zvláště u pomalejších zařízeních a zařízeních s menší kapacitou paměti to dost pomůže rychlému chodu aplikace. Principem indexování je přenesení paměťové a kapacitní náročnosti ze zařízení na server, což by při porovnání výpočetní kapacity serveru a výpočetní kapacity mobilního zařízení nemělo vůbec vadit.

[1] Binary Large OBject

4 Připojení k databázi na serveru

Kapitola se věnuje řešením a problémům při práci na propojení mobilní aplikace s databází. Naším požadavkem bylo načítat data z databáze a zároveň do ní data i vkládat. Zkusili jsme několik způsobů, které krátce popíšeme a poté se podrobněji zaměříme na řešení, které bylo zvoleno.

Jako první jsme vymysleli, že z mobilní aplikace by se poslal požadavek `http post` na `php` skript na serveru. Ten by pracoval s databází a prováděl požadované operace nad daty. Tento způsob se nám nepodařilo implementovat a dozvěděli jsme se, že to není bezpečný způsob. Proto jsme další vývoj přerušili a zkusili něco jiného.

Implementovali jsme možnost, že by se samotná mobilní aplikace připojila přímo do databáze. Tento způsob fungoval a oba naše požadavky byly splněny. Ale při hledání řešení jsme objevili, že tento způsob je velmi náročný na výpočetní techniku, protože navázání spojení je to nejtěžší na celém procesu připojení aplikace k databázi. Po konzultaci s doktorem Janem Pytlem jsme se rozhodli použít `Java Servlety`.

4.1 Java Servlet

Servlet je třída v jazyce Java, která rozšiřuje možnosti serveru. Servlet umí zpracovat požadavek `request` a odpovědět `response`. Musí běžet na serveru, který zvládne zpracovat programovací jazyk Java. Použili jsme `Apache Tomcat 7`[1]. Ten implementuje verzi `Servlet 3.0`, jehož možnosti jsme využili a zmíníme je dále.

Webová aplikace se servlety si s použitím interfacu `DataSource` předpřipraví připojení do databáze a vyhne se tím problému náročného přímého přístupu do databáze. Tyto předpřipravené připojení pak „půjčuje“ servletům, které reagují na požadavky z mobilní aplikace. Nastavení připojení se nastavuje v souboru `context.xml`. V tomto souboru jsou přihlašovací údaje do databáze. K tomuto souboru má přístup pouze administrátor a jiná bezpečnostní opatření se zpravidla neaplikují[2].

Z důvodů popsaných výše jsme vytvořili jednoduchou webovou aplikaci obsahující následující dva servlety.

- Servlet **GetFromDB.java**, který se stará o získávání dat z databáze. Podle parametru získaného z `HttpServletRequest` je nastavena odpověď do `HttpServletResponse`. Parametrem je adresa umístění obrázku, který bude následně odeslán do mobilního zařízení. Pokud není žádný parametr uveden, pak jsou do mobilní aplikace poslány všechny kamery z databáze.
- Servlet **SaveToDB.java**, který z částí `HttpServletRequest` získá informace o nově přidávané kameře a vloží ji do databáze. V tomto servletu byly použity nové technologie z verze

[1] http://tomcat.apache.org/
[2] Konzultace s Ing. Janem Pytlem Ph.D.

`Servletu 3.0`. Pří sestavování požadavku o kameře v mobilní aplikaci byl použit objekt `MultipartEntity`. Ten umožňuje poslat jedním http postem jak samotný text, tak i soubory, což jsou v našem případě fotografie kamer. Ty jsou ukládány v adresáři pod jmény podle času vložení. to zajišťuje unikátní název souboru, který je uložen do databáze.

4.2 Vytváření požadavku v mobilním zařízením

Požadavek na server se vytváří v mobilním zařízením ve třech případech.

- **Poslání nové kamery do databáze** - pomocí výše zmíněné `MultipartEntity` se posílají všechna data (např. souřadnice, obrázek kamery) ke zpracovaní do webové aplikace.
- **Při zobrazení mapy kamer** - neposílá se žádný parametr, ale z webové aplikace se získávají data o kamerách. V databázi není uložen přímo obrázek, ale pouze cesta na místo na serveru kde je obrázek uložen. Aplikace vytáhne soubor ze serveru a použije ho v informačním okně kamery jako její obrázek. V současné době se načítají z databáze všechny kamery, což je velmi náročné na pameť mobilního zařízení. V dalším vývoji aplikace se tento problém vyřeší nahráním kamer pouze z okolí současné lokace přístroje.
- **Při zobrazení obrázku kamery** - jako parametr se nastavuje název souboru s obrázkem kamery, na kterou uživatel poklepal. Tento obrázek se zobrazí na obrazovce mobilního zařízení.

Při vytváření požadavků na server jsme narazili na zajímavý problém. Pokud byl požadavek vytvářen přímo v dané aktivitě, tak tento požadavek nešel poslat na server. Důvod byl ten, že od určité verze Android API nešel poslat požadavek v hlavním vlákně aplikace. Pomocí třídy `AsyncTask` [3] lze jednoduše provádět operace v jiném vlákně aplikace, ve kterém se provede odeslání požadavku na server.

[3] http://developer.android.com/reference/android/os/AsyncTask.html

5 Problémy při implementaci aplikace

S problémy jsme se setkávali téměř nepřetržitě. Verze, kterou teď prezentujeme, se od verze kterou odevzdáme sdružení IuRe ještě trochu liší právě proto, že jsme se některé z problémů zatím rozhodli ignorovat (pokud přímo nenarušují chod aplikace) a vyřešíme je před odevzdáním aplikace sdružení. Tyto problémy jsou většinou spojeny s vytížením serveru a případném zahlcení, pokud by se do databáze časem nahrálo hodně dat a obrázků. Řešení je nasnadě:

- Naindexování databáze.
- Cachování obrázků.
- Používání BLOBu pro obrázek místo adresy umístění na serveru.

Tyto úkony provedeme před finálním odevzdáním aplikace. Vzhledem k pokročilosti těchto technik si myslíme, že jsou nad rámec obsahu projektu informatika 2, ale máme chuť na aplikaci pracovat dál.

Samozřejmě jsme se při tvorbě aplikaci setkali s větším počtem problémů, ale ty byly buď popsány v jiné části této dokumentace nebo vyřešeny.

6 Závěr

Výsledkem naši práce je aplikace pro přístroje s operačním systémem Android, vyvinutá pro občanské sdružení IuRe.

Přestože jsme s vývojovým prostředím pro tuto platformu neměli žádné zkušenosti, bohatá dokumentace tématu a propracované prostředí IDE Eclipse umožňuje programování i nováčkům v oboru. Přesto je stále mnoho prostoru pro zlepšování a proto, že se vývoji aplikace budeme věnovat i nadále, není verze odevzdávaná v rámci předmětu finální. V budoucnosti bude předělána grafická stránka aplikace. Z důvodů snížení datové náročnosti bude implementována metoda, která stáhne umístění kamer pouze v bližším okolí. Dalším nezbytností je zabezpečení aplikace proti zneužitelnosti, aplikováním časového omezení vstupu nových kamer z přístroje, který bude možné dále identifikovat unikátním kódem. Nakonec bude pod technickým zázemím organizace spušťen server pro komunikaci.

Při práci na projektu jsme osobně ocenili praktický dopad aplikace, technickou náročnost a nové zkušenosti s vývojem mobilní aplikace. Zadání bylo splněno s dvěma výjimkami. Jednou z nich je komunikace mezi aplikací a databází pomocí php skriptů. Místo nich jsme použili `Apache Tomcat` a javovské servlety, protože se jedná o mnohem lepší řešení. Druhou výjimkou je nahrávání kamer z aplikace rovnou do databáze `OpenStreetMap`. To není z technického hlediska úplně možné a proto jsme se rozhodli tento problém obejít ukládáním kamer do databáze na našem serveru, do které jsou naimportovány i data z `OpenStreetMap`.

7 Seznam zdrojů

Web

- PostgreSQL 9.1.3 Documentation. POSTGRESQL GLOBAL DEVELOPMENT GROUP. *Dokumentace PostgreSQL* [online]. 2011 [cit. 2013-05-13].
  Dostupné z: http://www.postgresql.org/files/documentation/pdf/9.1/postgresql-9.1-A4.pdf
- STACK EXCHANGE INC. *Stack Overflow* [online]. [cit. 2013-05-13].
  Dostupné z: http://stackoverflow.com/
- REFSNES DATA. *W3Schools* [online]. 1999 [cit. 2013-05-13].
  Dostupné z: http://www.w3schools.com/
- GOOGLE INC. *Google Developers* [online]. 2011 [cit. 2013-05-13].
  Dostupné z: https://developers.google.com/
- POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL* [online]. 1996 [cit. 2013-05-13].
  Dostupné z: http://www.postgresql.org/
- WIKIMEDIA. *Wikipedia* [online]. 2001 [cit. 2013-05-13].
  Dostupné z: http://www.wikipedia.org/
- LARS VOGEL. *Vogella* [online]. [cit. 2013-05-13]
- ANDROID SNIPPETS. *AndroidSnippets.com* [online]. [cit. 2013-05-14].
  Dostupné z: http://www.androidsnippets.com/
- VIKAS PATEL *Vikas Patel's blog* [online]. [cit. 2013-05-14].
  Dostupné z: http://vikaskanani.wordpress.com/

Poděkování

Děkujeme za praktické rady k javě a serveru od pana doktora Pytla.