ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

Katedra technologií a měření

DIPLOMOVÁ PRÁCE

Mobilní aplikace určená pro inteligentní parkovací systém

Bc. Libor Peleška

Plzeň 2018

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok 2017/2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Bc. Libor PELEŠKA
Osobní číslo:	T15N0038P
Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Komerční elektrotechnika
Název tématu:	Mobilní aplikace určená pro inteligentní parkovací systém
Zadávající katedra:	Katedra technologií a měření

Zásady pro vypracování:

1. Proveďte analýzu možností mobilní aplikace určené pro motoristy využívající parkoviště FEL/ZČU a posuďte statistiku vytíženosti daného parkoviště v průběhu roku
2. Vytvořte serverovou aplikaci schopnou dodávat data do mobilní aplikace v chytrých telefonech prostřednictvím datového a WiFi připojení
3. Implementujte navrženou strukturu dat volných parkovacích míst
4. Navrhněte a realizujte mobilní aplikaci (OS Android) se kterou bude moci data ze serveru přijímat a zobrazit na mobilním telefonu (např.pouze prostřednictvím textu)
5. Dosažené výsledky zhodnoťte, diskutujte klady, zápory, případnou nevýhodu navrženého řešení a popište možnosti dalšího vylepšení a využití

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah kvalifikační práce:	**40 - 60 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Vladimír Pavlíček, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**10. října 2017**
Termín odevzdání diplomové práce:	**7. června 2018**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Ing. Aleš Hamáček, Ph.D.
vedoucí katedry

V Plzni dne 10. října 2017

Abstrakt

Tato diplomová práce se zabývá inteligentními parkovacími systémy, konkrétně na část předání informace o situaci na parkovišti uživateli. V práci je popsána problematika spolu s přístupy k jejímu řešení. Je zde představeno řešení pracující na indows Server 2012 spolu s aplikací operačního systému Android pro parkoviště před budovou FEL/ZČU. Obsluha serveru byla naprogramována za použití Visual Studio Community 2017 spolu s pluginem pro mobilní aplikace Xamarin, obojí za použití jazyku C#. Právě díky Xamarinu lze vytvořenou aplikaci snadno převést pro další operační systémy mobilních zařízení.

Klíčová slova

Inteligentní parkovací systém, mobilní aplikace, Android, Xamarin, Windows Server, Windows Service,, webová služba, MySQL, XML

Abstract

This master thesis is focused on the topic of intelligent parking systems, specifically on the part of transmit information about situation on the parking lot to the parking user. In this paper are described intelligent parking systems with their issues and approaches to their realization. A solution for server side on Windows Server 2012 is presented together with application for Android operating system which covers the situation on the parking lot in front of the building of Faculty of Electrical Engineering at University of West Bohemia. Server management service was made with Visual Studio Community 2017 containing plugin Xamarin, both with usage of language C#. Thanks to Xamarin the application is portable for other most common operation systems for mobile devices.

Keywords

Intelligent parking system, smartphone application, Android, Xamarin, Windows Server, Windows Service, webservice, MySQL, XML

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 23.5.2018 Libor Peleška

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce panu Ing. Vladimíru Pavlíčkovi Ph.D. za jeho užitečné příspěvky, návrhy a metodické vedení práce. Dále bych chtěl poděkovat panu Ing. Petru Weissarovi Ph.D. a Ing. Ladislavu Peleškovi za odborné konzultace a cenné rady v oblasti NET Frameworku a programování v jazyce C#.

Seznam obrázků

Seznam symbolů a zkratek

AI ..... Automatic Incrementation
FEL ZČU ..... Fakulta Elektrotechnická Západočeské Univerzity v Plzni
FTP..... File Transfer Protocol
HTML ..... HyperText Markup Language
HTTP ..... Hypertext Transfer Protocol
HTTPS ..... Hypertext Transfer Protocol Secure
IEEE..... Institute of Electrical and Electronics Engineers
IIS..... Internet Information Services
JSON..... JavaScript Object Notation
NN..... Not Null
PCL ..... Portable Class Library
PDF ..... Portable Document Format
PIR ..... Pasiv Infra Red detector
PK ..... Primary key
PDF ..... Portable Document Format
RFID ..... Radio Frequency Identification
SGML ..... Standard Generalized Markup Language
SOAP ..... Simple Object Access Protocol
SQL..... Structured Query Language
WCF..... Windows Communication Framework
WSDL ..... Web Services Description Language

Obsah

1 Úvod

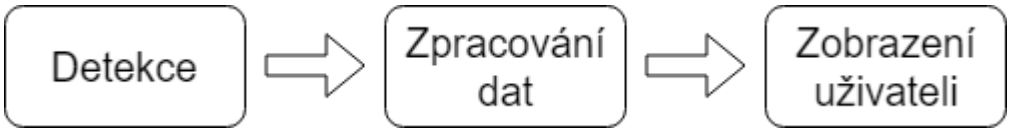
Spolu s rozvojem každodenního využití automobilů k přepravě osob za poslední století se začíná stále více projevovat problematika parkování. Pokud vlastníte automobil, zajisté máte s problémem hledání místa k zaparkování vlastní zkušenost. Z posledních dostupných dat [1] Svazu dovozců automobilů bylo do 31. 12. 2016 v Česku registrováno celkem 5 368 660 osobních automobilů, takže průměrně skoro každý druhý občan vlastní automobil. V místech s větší hustotou populace (zejména v městech) se tedy stává aktuální informace o dostupnosti parkovacích míst velmi cennou, nejen z hlediska ušetřeného času, ale i pro možnost snížení spotřeby paliva, znečištění a dopravního vytížení. Za tímto účelem vznikají různé druhy inteligentních parkovacích systémů. Systémy se liší v přístupu ke snímání situace na parkovišti, uživatelských rozhraních nebo rozlehlosti oblastí pro parkování. Existují systémy pro celá města, ve kterých jsou vyobrazeny parkoviště, nebo i systémy řešící parkování v menších areálech například univerzitách.

Úkolem inteligentních parkovacích systémů je dát uživateli informaci o aktuální situaci na parkovišti. K získávání těchto dat lze přistupovat více způsoby. Nejpraktičtějším a v budoucnu patrně nejvyužívanějším se zdá být použití senzorů nebo videodetekce. Informační rozhraní pro uživatele pak bývají nejčastěji v podobě mobilních či webových aplikací, nebo informačních tabulí. V současné době již existují různá komerční řešení a byly vytvořeny jiná koncepční v rámci vědeckých konferencí, která jsou zmíněna v první kapitole. Tato práce má za úkol navrhnout mobilní aplikaci určenou pro inteligentní parkovací systém obsluhující parkovací stání před budovou FEL/ZČU. Je zde navržena datová struktura pro získávání dat ze snímacího stupně, vytvořena serverová strana pro jejich zpracování, uchování a předání uživateli mobilní aplikace. Uživateli se na obrazovce telefonu zobrazí mapa parkoviště s parkovacími pozicemi zabarvenými do třech možných stavů.

Práce je součástí projektu věnujícímu se detekování obsazenosti parkovacích stání pomocí kamerových dat. V rámci tohoto projektu již byla vytvořena diplomová práce *Obrazová analýza kamerových dat parkovacích stání,* ve které byl představen proces detekce a vytvoření výstupních dat. Tato data jsou pak vstupem pro předloženou práci.

2 Inteligentní parkovací systémy

Najít volné místo k zaparkování v zalidněných bývá často těžkým údělem nejen pro řidiče, ale i pro společnost. Podle výzkumu [2] Rudgers University z roku 2010 v jedné obchodní čtvrti Los Angeles bylo zjištěno, že při hledání místa k zaparkování vozidla ujedou za jeden rok vzdálenost srovnatelnou s 38 cestami kolem světa, vyprodukují 730 tun oxidu uhličitého a spálí přibližně 1,8 milionu litrů paliva. V posledních dvou dekádách je ve městech směřováno více pozornosti k tomuto problému a jsou vytvářeny informační parkovací systémy. Jejich úkolem je zjistit informace (počet, poloha) o volných místech na parkovišti a zobrazit je uživateli skrz uživatelská rozhraní. Systémy se vzájemně se odlišují v přístupech ke snímání volných míst a způsobů jejich zobrazení, často také bývají připojeny k účtům pro placení parkování.



Obrázek 1: Funkční bloky inteligentních parkovacích systémů

2.1 Způsoby detekce volných míst

2.1.1 Senzory

Ke zjištění přítomnosti vozidla jsou předně využívány různé typy senzorů, které se liší dle jejich fyzikálního principu. Senzory lze umístit nad nebo pod povrch vozovky. Umístěním senzoru pod povrch lze předejít jeho úmyslnému i neúmyslnému poškození vlivem dění na vozovce a v jejím okolí, avšak náklady na jeho instalaci převyšují povrchové detektory. Nejčastějším typem těchto senzorů jsou senzory indukční smyčky, která je umístěna pod povrchem. Smyčka je napájena harmonickým napětím vytvářejícím homogenní harmonické pole. Kovová karoserie automobilu pak vyvolá změnu tohoto pole, kterou vyhodnotí detektor připojený do okruhu [3]. Levnějším přístupem je sledování změn magnetického pole Země, vyvolaných přítomností automobilu v okolí pomocí magnetometrů [4]. Další možností je umístit na povrch síť optických vláken a sledovat změnu délky vlny světla odraženého do snímače [5]. Nad povrch vozovky pak umisťujeme detektory optické, ultrazvukové a PIR. První dva mají společný princip vysílání média (laserový paprsek/ ultrazvukové vlny) a následné vyhodnocení změn v jeho odrazech. Snímače PIR detekují změnu teploty při příjezdu vozidla na základě pyroelektrického jevu[6]. Nedílnou součástí každého detekčnho systému je pak

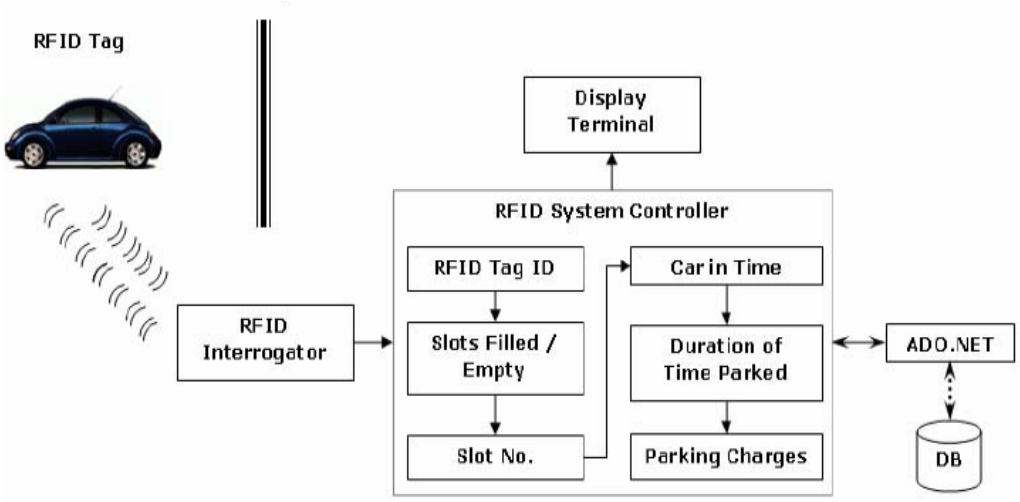
jednotka vyhodnocování získaných dat.

2.1.2 Využití technologií v mobilních telefonech

S nástupem chytrých telefonů se stalo jednodušší sdílet informace. Valná většina z nich je vybavena GPS, a tak data z jejich pohybu přispívají k získání informací o provozu, dopravních zácpách nebo parkování. Protože tato technologie pro svou funkci nepotřebuje žádnou konkrétní fyzickou infrastrukturu, mohou tedy být nejlevnějším řešením. S rostoucí podporou crowdsourcing technologií se mohou stát řešením budoucnosti. Pomocí akcelerometru a gyroskopu v telefonu lze vyhodnotit, zda jedete v autě nebo jdete pěšky a pohyb lze mapovat. Takových informací můžeme využít i pro analýzu parkování, pro které je směrodatná hlavně změna módu chůze/jízda. Při výzkumu [7] vědci zašli ještě o něco dál a nashromážděním dat do statistik vytvořili modely pro pravděpodobnosti obsazenosti míst. Hlavní nevýhodou tohoto přístupu je částečná ztráta soukromí odevzdáváním dat o své poloze třetí straně. Další nevýhodou je nižší spolehlivost získaných dat oproti senzorovému přístupu.

2.1.3 RFID technologie

Také technologie identifikace na rádiové frekvenci nachází své využití při detekci volných parkovacích míst. Toho je dosaženo uložením informací o vozidle na mikročip s anténou, pomocí RFID značek se identifikují příjezdy a odjezdy vozide [8]. Existují dva druhy RFID čipů – aktivní, pasivní. Aktivní mají své vlastní napájení a mohou do okolí vysílat signál s informacemi. Jsou také považovány za lépe fungující. Oproti tomu pasivní čipy vlastní napájení nemají. Snímač pasivních čipů do svého okolí periodicky vysílá pulzy a teprve přiblížením pasivního čipu ke snímači dojde k nabití jeho napájecího kondenzátoru a odeslání odpovědi. Většinou dokáží odeslat jen číslo dané při jejich výrobě [9]. Pro svou nízkou cenu nacházejí využití správy parkování. Technologický institut v indickém městě Vellore přišel v roce 2007 s řešením [10] přidělování míst a správy parkoviště pomocí pasivních čipů. Každé auto má svůj chip, na vjezdech a výjezdech z parkoviště jsou umístěny RFID čtečky připojené k počítači, ve který vyhodnocuje obsazenost parkoviště ukládá data informace o délce parkování a parkovacího poplatku do databáze, ve které jsou také informace o vozidle.



Obrázek 2: Architektura automatického přiřazování míst pomocí RFID čipu (převzato z [10])

V počítači běží čtyři služby: komunikátor sériového portu, kontrola obsazenosti, kalkulátor parkovacího poplatku a služba pro komunikaci s uživateli pomocí SMS. Parkující se tak může dotázat, zda jsou na parkovišti volná místa, nebo si na určitý čas místo rezervovat.

2.1.4 Videodetekce

Dalším možným způsobem analýzy dění na parkovišti je videodetekce. Tento přístup spočívá ve snímání prostoru parkoviště kamerou, následnou aplikací masek a detekčních metod. V rámci projektu řešení parkování před budovou FEL/ZČU byla již vytvořena diplomová práce [11] na téma zpracování dat z IP kamery.

2.2 Uživatelská rozhraní

Parkující může být informován o možnostech parkování více způsoby. Nejzákladnějším je informační LED tabule, dokáží pouze informovat o aktuálním počtu volných míst větších parkovišť ve městech.

Obrázek 3: Ukázka informační tabule v Plzni

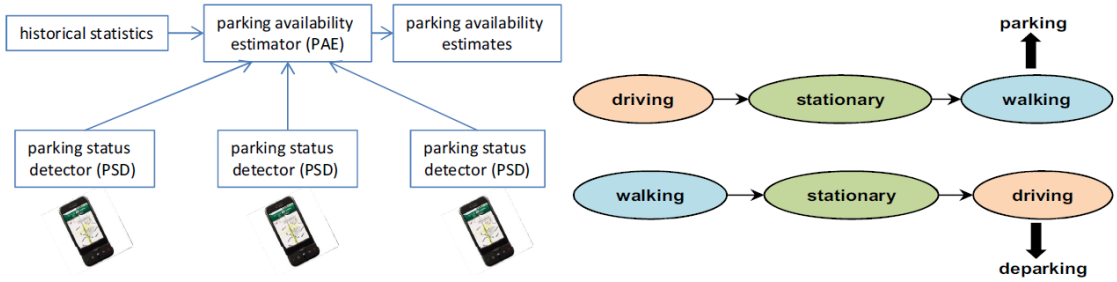
Vzájemná komunikace mezi uživatelem a parkovištěm bývá také zprostředkována pomocí webových, nebo mobilních aplikací či může být realizována přes SMS pomocí GSM modulu v řídící jednotce parkoviště. Pomocí těchto rozhraní lze za parkování platit, rezervovat místa na určitou dobu, nebo uživatele navést na konkrétní volné místo.

2.3 Současný stav a vývoj

Již v roce 2013 bylo na mezinárodní konferenci IEEE představeno velmi sofistikované řešení [13] systému inteligentního parkování. Systém obsahuje webové i mobilní aplikace pro správce i uživatele parkovišť v rámci většího objektu uzavřeného (univerzity) a umožňuje komunikaci s již existujícími službami jako Google Maps či FourSquare. Oproti dále zmiňovaným aplikacím se velmi liší v přístupu ke zjišťování obsazenosti. Uživatel aplikace si musí vytvořit účet a následně do něj zaregistrovat své automobily, vyplnit osobní a případně platební údaje. Vyhodnocení obsazenosti pak vychází ze znalosti kapacity jednotlivých parkovišť v objektu a počtu vozidel přihlášených k tomuto parkovišti. Pro navádění a zobrazování map aplikace využívá Google Maps API. Aplikace pro operátory parkoviště umožňuje spravovat a přidávat nová parkovací místa v objektu či zobrazovat různé statistiky z dat získaných provozem.

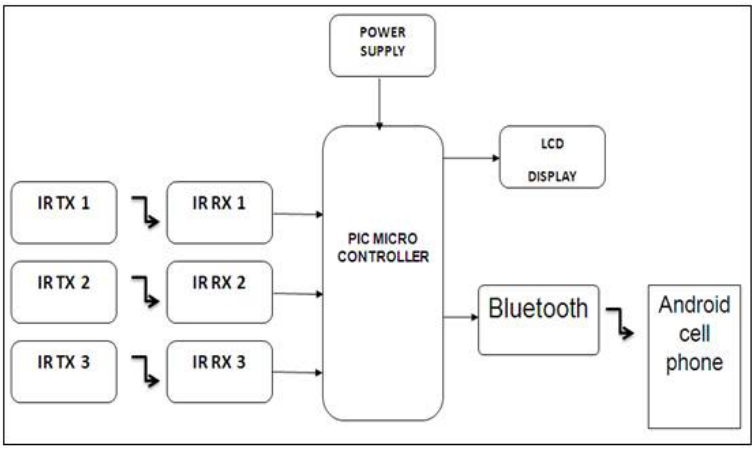
Obrázek 4: Architektura systému, webová aplikace pro správu parkoviště a mobilní aplikace pro uživatele (převzato z [13])

Na stejné konferenci bylo představeno ještě řešení nazvané PhonePark [7] vědci z Illinoiské univerzity v Chicagu. PhonePark představuje metodu detekce stavu parkujícího, vytváření historického profilu stavu parkoviště a algoritmy pro vyhodnocení stavu parkoviště. Algoritmy byly ověřeny na základě dostupných reálných dat. K detekci je navrženo použití senzorů v mobilních telefonech (GPS a akcelerometr), což činní systém cenově dostupným. Algoritmus detekce také umožňuje platbu za parkování. Obecně se toto řešení zdá být ekonomické a flexibilní. V reálném čase poskytuje dobrý odhad a zaměřuje se na pouliční parkování.



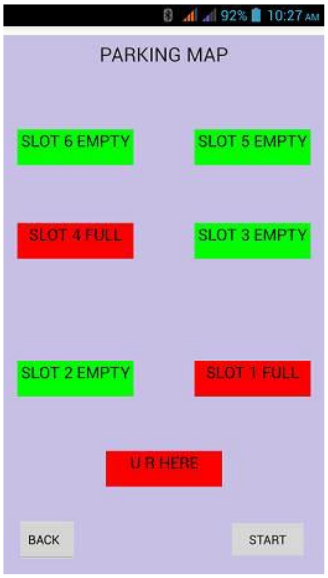
Obrázek 5: Komponenty systému PhonePark a schéma detekce dopravního režimu (převzato z [7])

V roce 2016 byl a indické univerzitě Savitribai Phule Pune ve městě Maharashtra zpracován projekt [14] parkovacího systému řízeného mikrokontrolerem řady PIC16XX. Zařízení snímá obsazenost míst pomocí IR čidel, data vyhodnocená čipem pak zobrazuje na informační tabuli a pomocí bluetooth do Android aplikace, která pak uživatele navede na místo.



Obrázek 6: Blokový diagram systému řízeného mikrokontrolerem PIC16XX (převzato z [14])

Algoritmus přidělení místa pak probíhá následovně: Kontrolér zjistí stav volných míst na základě čidel. Z mobilního zařízení skrze bluetooth přijde dotaz, řídící jednotka zkouší obsazenost míst na základě jejich ID. V případě, že je místo volné přiřadí se uživateli a zobrazí nejkratší cestu. Pokud je místo obsazené, pokračuje v prohledávání. Když nenajde žádná volná místa vrátí se k prvnímu kroku.



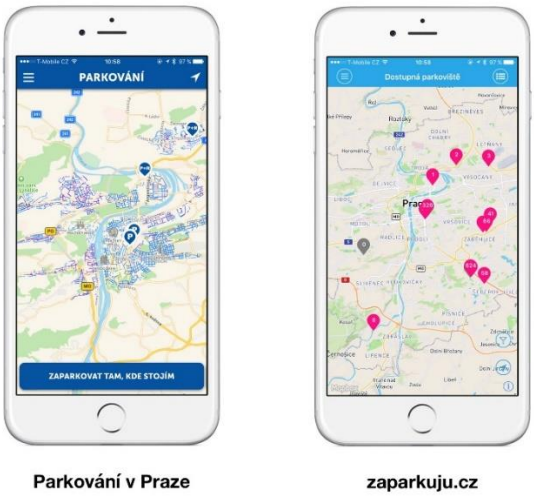
Obrázek 7: Android aplikace k projektu Savitribai Phule Pune University (převzato z [15])

Jiným přístupem, který lze zvolit pro řízení provozu parkování v areálu je pomocí QR kódů. Do kódu lze uložit identifikační číslo místa, jeho zeměpisnou šířku a délku. Kód se poté vytiskne a umístí k parkovacímu místu a vytvoří se databáze míst. Pro přístup do databáze pak slouží webová nebo opět mobilní aplikace. Uživatel si musí vytvořit účet a přihlásit se do

systému. Při příjezdu nebo odjezdu pak naskenuje QR kód a informace se přidají do databáze, do které pak přistupují uživatelské aplikace a zobrazují je [15]. To ovšem není velmi praktické.

Nabídka komerčních mobilních aplikací pomáhajících s parkováním stále roste. Trh ještě není zasycen, ale projevuje se snaha investorů nepromarnit příležitost v tomto segmentu. Některé aplikace, jako například Parking Mate, mají pouze informativní charakter. Zobrazují na mapě parkoviště poblíž naší polohy, jejich otevírací dobu a hodnocení. Aplikace jako Parkmobile, SecureParking nebo ParkMate už jsou značně sofistikovanější, a kromě obsazenosti míst umožňují za parkování platit, rezervaci míst v různých režimech (v závislosti na denní době nebo lokaci – hotely, sportovní stadiony) nebo různá dlouhodobější předplatné. U některých aplikací lze platit pomocí peněženek, kam si uživatel nahraje zvolenou částku jako kredit.

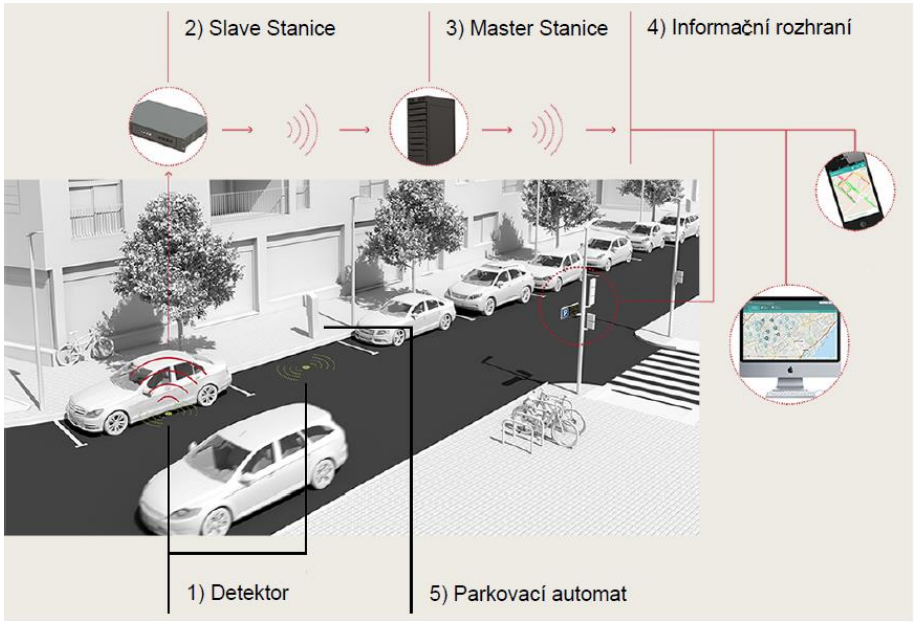
V Česku existují zatím dvě takové aplikace, a to pouze pro Prahu. Jmenují se zaparkuju.cz a Parkování v Praze. Po spuštění se vám zobrazí mapa města s ulicemi zbarvenými podle parkovacích zón. Aplikace zaparkuju.cz nabízí detaily o otevírací době, ceně parkování, hodnocení a výhody ve smysli dostupného MHD nebo zabezpečení, v této aplikaci je rovněž plánována možnost rezervace míst. Obě aplikace nabízejí možnost uložit i více SPZ a platební nebo CSS karty. K samotnému placení ale dochází pomocí webové aplikace „virtuální parkovací hodiny“. Vývojáři zaparkuju.cz do budoucna plánují zavést vlastní platební portál. Obě tyto aplikace jsou zdarma [16].



Obrázek 8: Aplikace pro parkování v Praze

3 Serverová strana

Inteligentní parkovací systém je složen z jednotlivých funkčních bloků, některé z těchto bloků jsou pro funkci systému nezbytné a jiné jsou volitelné v závislosti na typu aplikace. Prvním nepostradatelným blokem je detektor poskytující formu dat za jejíž pomocí lze vyhodnotit obsazenost parkovacího místa v našem případě IP kamera. Druhým blokem systému je Slave stanice komunikující s jedním, nebo více detektorů a vyhodnocující data. V našem případě má každá kamera vlastní Slave stanici s podobě jednotky pro zpracování signálu. Jedná se o mikropočítač s nahraným operačním systémem a algoritmem pro zpracování obrazu. Tento algoritmus je detailněji popsán v práci [11]. Příjem a zpracování dat slave stanic obsluhuje master stanice. Rovněž zajišťuje předání dat do zobrazovacího rozhraní. Volitelným prvkem je parkovací automat sloužící k výběru parkovného.

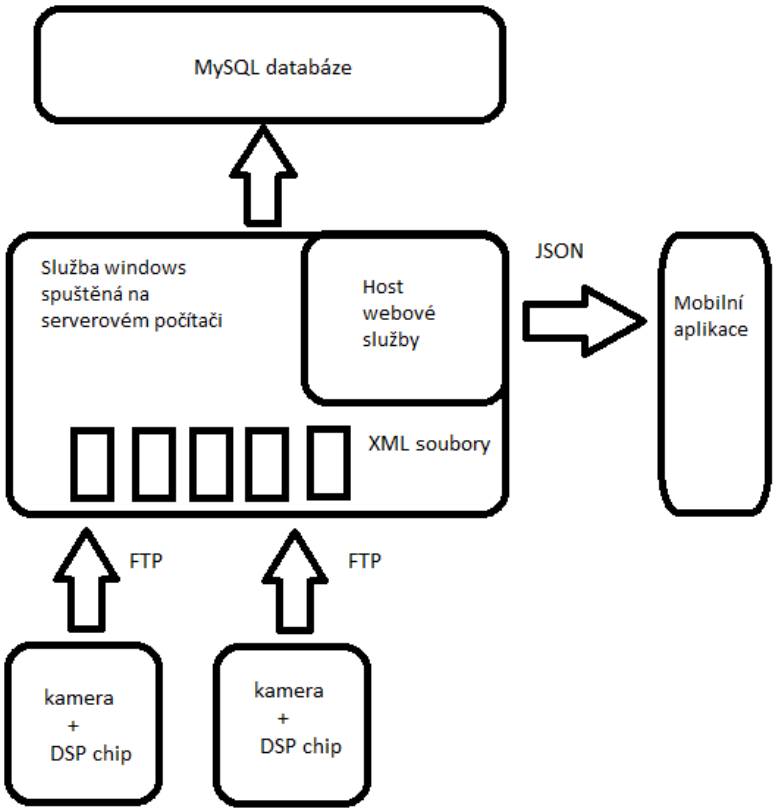


Obrázek 9: Schéma parkovacího systému (převzato z [11])

3.1 Popis principu

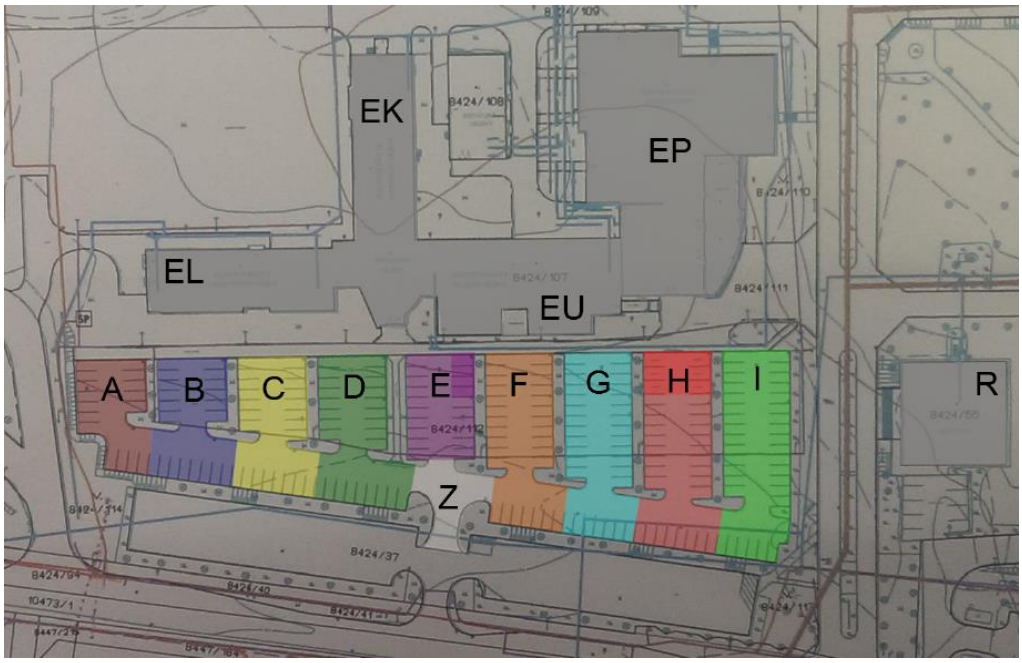
Hlavním úkolem serverové strany všech parkovacích systémů je zpracovat data ze vstupních systémů rozpoznávání a předat je klientům do mobilních zařízení. Data bývají uložena v databázi pro další využití (rezervace, poplatky). V našem případě uvažujme soustavu kamer s jednotkou pro digitální zpracování signálu, která nahrává za použití protokolu FTP na server XML soubory s výstupními hodnotami algoritmu zpracování signálu jednotlivých kamer. Každá kamera sleduje jeden záliv parkoviště a generuje specifický soubor. Soubor

obsahuje datum a čas kamerové inspekce, počet volných míst a pravděpodobnost obsazenosti jednotlivých pozic.



Obrázek 10: Schéma řešení

Na serverovém počítači s operačním systémem Windows je spuštěna služba periodicky načítající soubory z výstupů digitálního zpracování. Hodnoty obsazenosti míst pak načte do paměti v podobě pole celočíselných proměnných. V případě změny hodnoty na nějaké pozici vytvoří záznam v databázi. Jejím dalším důležitým úkolem je provozovat webovou službu, z které získávají data klienti s mobilní aplikací.



Obrázek 11: Značení zálivů před budovou FEL (převzato z [11])

3.2 Formát XML

Formát XML byl definován konsorciem W3C pro přenos obecných dokumentů a dat. Zkratka znamená eXtensible Markup Language – rozšiřitelný značkovací jazyk. Spolu s formátem HTML vychází tento název ze staršího obecnějšího standartu SGML (Standarted Generalized Markup Language). Oproti zmíněnému HTML nemá XML pevnou sadu značek a lze ji definovat rozdílně pro různé sady dokumentů. Sadu značek lze definovat přímo v dokumentu, může být dohodnuta předem, nebo ji lze specifikovat odkazem.

Značky označují části dokumentu a vyjadřují jeho syntaktickou strukturu. Tvoří je otevírací závorka(start-tag) a zavírající závorka (end-tag). XML nedefinuje význam obsahu značek, nebo jejich sémantiku. U zpráv tedy neznáme její obsah ani způsob zpracování, to je úlohou aplikací s ní pracujících. Smyslem XML je stanovit strukturu dokumentu a umožnit jeho kontrolu a zpracování obecnými nástroji [17].

Při návrhu formátu XML byly stěžejní následující požadavky:

- Použitelnost v rámci internetu
- Podpora široké škály aplikací
- Kompatibilita s formátem SGML
- Čitelnost a pochopitelnost pro člověka

- Minimální množství variant – nejlépe žádné
- Jednoduchá programová manipulace s dokumenty

3.2.1 Dokument XML

Součástí XML dokumentu jsou prvky zvané entity, každá entita může obsahovat rozpoznatelná nebo nerozpoznatelná data. Rozpoznatelná data se stávají ze znaků vybrané abecedy a představují značky(markups) nebo znaková data. Za pomocí značek je tvořena logická struktura dokumentu s tím i jeho rozložení(layout). Dokument obsahuje elementy vymezené závorkami s jeho jménem. V elementech mohou být posloupnosti znakových dat, další vnořené elementy, odkazy na jiné entity nebo komentáře. Další možností elementů je přidat jim atributy pro bližší specifikaci. Ty jsou pak vloženy do otevírací závorky elementu v podobě parametrů zadaných klíčovými slovy. Atribut je specifikován jménem a hodnotou v podobě jméno = "hodnota", kde hodnota je vždy v uvozovkách, nebo apostrofech [17].

Vstupní XML soubor pro naši aplikaci obsahuje kořenovou entitu data se třemi elementy – *datetime, count* a place. V *datetime* je uložen čas vygenerování souboru, *count* obsahuje počet volných míst. Element place obsahuje unikátní identifikační číslo místa na parkovišti a v atributu *value* je zaznamenána procentuální hodnota obsazenosti. Počet výskytu elementu place pak odpovídá počtu parkovacích míst v zálivu. V příloze A je ukázka XML souboru pro záliv A.

3.3 Windows Server 2012 R2

Důvodem vzniku Windows Server 2012 bylo umožnit optimalizaci operačního systému pro různé rozměry využití. Existují verze pro jednoduché hostující servery, síťové servery pro hostování doménových služeb, tak i robustní podnikový server pro hostování nezbytných aplikací nebo server datových center pro hostování důležitých obchodních řešení. Operační systém Windows Server lze provozovat pouze na 64bitovém hardwaru. Počítače se 64bitovými verzemi systému fungují rychleji a lépe, jsou také rozšířenější, protože umožňují zpracovat větší množství dat za hodinový takt, adresovat více paměti a rychleji provádět výpočty. 64bitová architektura je určena k provádění operací náročných na paměť a vyžadují rozsáhlé číselné výpočty. Díky 64bitovému zpracování mohou aplikace plně načítat velké soubory dat do fyzické paměti (tj. RAM), což snižuje potřebu volného místa na disku a výrazně zvyšuje výkon. Instance Windows Server lze spouštět na skutečných zařízeních nebo na virtuálních [18].

Do Windows Server 2012 R2 je přidáno 300 nových funkcí a je přizpůsobený pro cloudové technologie. Cloudová technologie je způsobem využití sítě vzdálených serverů pro ukládání, správu a zpracování dat namísto lokálního serveru. Windows Server 2012 R2 zpřístupňuje tyto technologie korporacím jejichž zaměstnanci je budou používat stejným způsobem. Všechna firemní data za pomocí virtuálních počítačů nebo jednotlivých pracovních stanic mohou být zálohována přímo do cloudu buď na serveru, nebo mimo něj. Cloudové technologie mají velký potenciál v budoucnosti [19].

Pro účely zprovoznění a vyzkoušení serverové strany projektu mobilní aplikace byl zřízen cloudový server společnosti Amazon, konkrétně produkt Amazon EC2 z nabídky Amazon Web Services. Cloudová technologie však pro účel našeho parkovacího systému není příliš vhodná. Server by bylo lepší mít v jedné síti s jednotkami zpracování signálu. Server běží na výše zmíněném operačním systému Windows Server 2012 R2. Server byl nastaven a spuštěn pomocí konzole portálu Amazon, poté se k němu lze připojit za použití vzdálené plochy. Bylo nutné nastavit veřejnou IP adresu a vytvořit pravidla pro komunikaci na portech. Stejná pravidla pak bylo potřeba nastavit i v bráně firewall systému Windows Server 2012 R2 a nainstalovat IIS – Internet Information Services.

Port	Využití
80	Protokol HTTP
21	Protokol FTP aktivní
50000-60000	Protokol FTP pasivní
8733	Webová služba
3389	Protokol vzdálené plochy
443	Protokol HTTPS

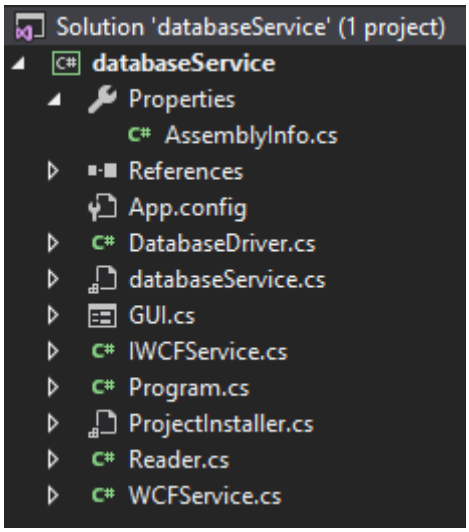
Tabulka 1: Nastavení výjimek na portech pro příchozí spojení odkudkoliv

Na server byl nainstalován a spuštěn FileZilla Server ke sdílení složky místního disku pro možnost nahrávat data za pomocí protokolu FTP. K tomu je potřeba v programu vytvořit účet a přidat k němu zvolenou složku. Poté je potřeba v záložce nastavení pasivního módu přiřadit IP adresu – veřejnou IP adresu našeho serveru. Dále byla na serveru spuštěna služba operačního systému Windows a byl nainstalován MySQL Server 5.7 probrané v dalších kapitolách.

3.4 Windows service - služby Windows

Windows service je druh aplikace, běžící na pozadí systému. Služba nevyžaduje pro svůj běh přihlášení uživatele. Proto nemají grafické rozhraní a běží bez přímé interakce s uživatelem. Například webový server, který očekává příchozí HTTP spojení a obsluhuje požadavky. Služby lze spouštět se startem systému a zastavují se jeho ukončením. Lze je spouštět až na vyžádání nebo jejich start zakázat, pokud jejich běh není aktuálně v systému nutný. Podstatné množství služeb využívá i operační systém k obsluze zcela základních funkcí. Jako třeba služba pro zpracování audio výstupu, firewall, cache DNS záznamů, kryptografické služby, plánovač úloh. S přibývajícím komponenty pak počet služeb může narůstat – například náš databázový server [20].

Za účelem zpracování XML souborů, zpracováním jejich obsahů a předání dat klientům byla vytvořena služba *databaseService*. Služba načítá soubory z disku v pětivteřinových intervalech, do paměti si nahraje pole s aktuální procentuální hodnotou obsazenosti všech míst. S dalším taktem pak kontroluje změnu oproti předchozímu stavu a v případě změny zapíše do databáze aktuální čas, místo a hodnotu. Dalším úkolem služby je hostovat webovou službu, která přebírá pole s hodnotami obsazenosti a po úpravě poskytne klientům.



Obrázek 12: Struktura projektu databaseService

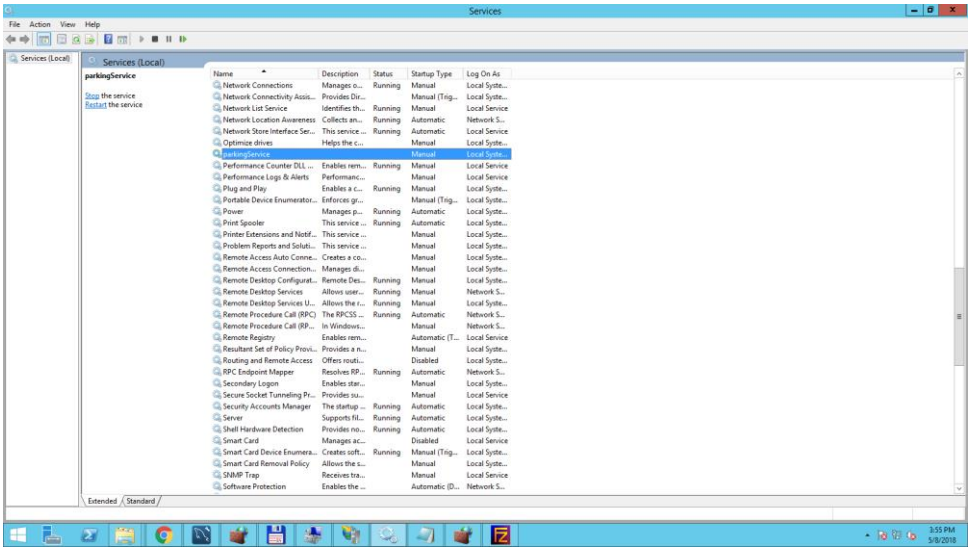
Přiložený projekt služby *databaseService* byl vytvořen ve Visual Studio Community 2017 a kódován v jazyce C# za použití .NET Frameworku 4.5. Hlavním vstupem programu je třída *Program.cs* spouštějící naši službu definovanou ve stejnojmenné třídě *databaseService.cs*. V třídě je vytvořen časovač a nastavena jeho perioda. Rovněž se zde definuje pole řetězců, kde každý řetězec je cestou k jednomu XML souboru. Dále je tu konstruktor, konstruktor instance, metody pro spuštění nebo zastavení služby a obsluhu časovače. Pro svou činnost využívá funkcí třídy *Reader.cs*, která obsahuje metody a proměnné pro zpracování XML souboru. V případě nutnosti zápisu do databáze je použita třída *DatabaseDriver.cs* obsahující vstupní parametry pro připojení k databázi, metody pro připojení/odpojení a vložení záznamu. Objekt *ProjectInstaller.cs* obsahuje objekty *serviceProcessInstaller* a *serviceInstaller*. V prvním z nich je potřeba nastavit vlastnost *Account* na *LocalSystem* k nastavení práv pro instalaci. V objektu serviceInstaller byl zvolen název *DisplayName* parkingService. Soubor *App.config* obsahuje informace o konfiguraci služby, jejím chování a koncových bodech pro webové služby. Třídě *WCFService.cs* spolu s jejím rozhraním *IWCFService.cs* je věnována další kapitola.

Windows služba je prakticky .exe soubor, bez možnosti přímého spuštění. Vnitřně je služba koncipována ke komunikaci se systémem, primárně pro předání příkazů. Hlavními jsou příkazy “Start” a “Stop”. Případně “Pause” a “Continue”, pokud služba podporuje dočasné pozastavení [20].

Po sestavení projektu můžeme službu nainstalovat do systému spuštěním příkazového řádku v režimu správce a vložením příkazu:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe c:\...\databaseService.exe
```

Nebo spuštěním souboru *install-service.bat* (potřeba upravit cestu k souboru databaseService.exe). Služba se pak zobrazí v konzoly Services systému Windows odkud ji lze spouštět nebo zastavovat.



Obrázek 13: Konzole služeb běžících na serveru

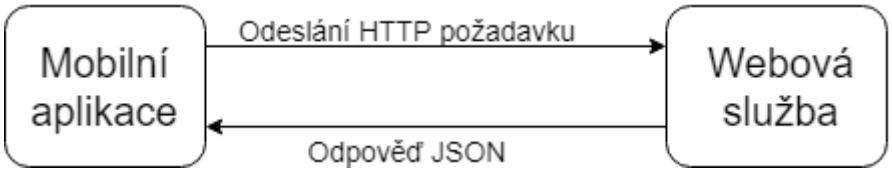
3.5 Webové služby

Webové služby jsou komponenty určené ke komunikaci pomocí protokolů jako HTTP, HTTPS nebo SMTP, to zaručuje snadný průchod přes aktivní prvky obsažených v každé počítačové síti. Univerzálnost, dostupnost na vyšších vrstvách a nezávislost na platformě zaručuje přenášení dat a dotazů(volání) pomocí standardizovaných protokolů jako XML, JSON nebo SOAP. Spojení mezi klientem a službou je realizováno velmi volně, takže jednotlivé prvky zprávy na sobě bývají nezávislé a mají vlastní pravidla. Komunikace služby s klientem staví na asynchronním zpracování zpráv. Webové služby sami popisují vlastní funkcionalitu včetně charakteristiky datových typů pro předávání parametrů a návratových hodnot jednotlivých funkcí, klient si tedy může kdykoliv zjistit komunikační rozhraní. Webové služby na platformě Microsoft jsou speciálními typy aplikací ASP.NET (rozhraní .NET Framework pro tvorbu webových aplikací) [21].

Microsoft přišel s vlastní sadou pro webové služby nazvanou Windows Communication Framework zkráceně WCF. S její pomocí můžete vytvářet jak klasické webové služby, tak i služby sofistikovanější využívajících i jiné protokoly než HTTP nebo HTTPS. V adrese služby pak bude uveden komunikační protokol, adresa cílového zařízení, port a další specifická cesta, nebo URI (Universal Resource Identifier – Univerzální identifikátor prostředků). Adresy pak mají zpravidla následující formát:

[transportní protokol]://[počítač nebo doména][:port]/[volitelné URI nebo cesta]

WCF služby rovněž nabízejí možnost takzvaného self-hostingu, jedná se o hostování služby v procesu zajištěném jejím vývojářem. V našem případě hostování ve službě Windows [22].



Obrázek 14: Princip webové služby typu REST

V projektu *databaseService* je ve třídě *WCFService.cs* vytvořena WCF služba a její rozhraní obsahuje třída *IWCFService.cs*. V souboru *App.config* jsou konfigurovány koncové body služeb (jejich adresy, chování, vazby a používaný interface). Naše WCF služba má pouze jeden koncový bod vázaný na webové metody protokolu HTTP a používá interface *IWCFService.cs*. Webové služby vázané na požadavky protokolu http se nazývají služby typu REST. V něm jsou definovány kontrakt služby a kontrakt dat. Kontrakt služby obsahuje kontrakty operací, tedy jednotlivých metod a jejich nastavení pro komunikaci. V našem případě metodu zpracovávající pole s procentuální hodnotou obsazenosti na třístavové pole o stejné délce. Metoda má nastaven koncový bod odpovídající na GET požadavek protokolu HTML a typ zprávy na JSON. Datový kontrakt obsahuje přenášená data zprávy a u nás zůstává prázdný, protože data přenášíme v odpovědi na požadavek. Třída *WCFService.cs* obsahuje definici zmíněné metody, která projede celé pole a pokud je vstupní hodnota menší než 40 přiřadí nulu, mezi 40 a 60 přiřadí jedničku a pokud je větší než 60 přiřadí dvojku. Adresa služby je v našem

http://veřejná IP adresa:8733
/Design_Time_Addresses/databaseService/WCFService/REST/getData

případě:

3.6 MySQL Databáze

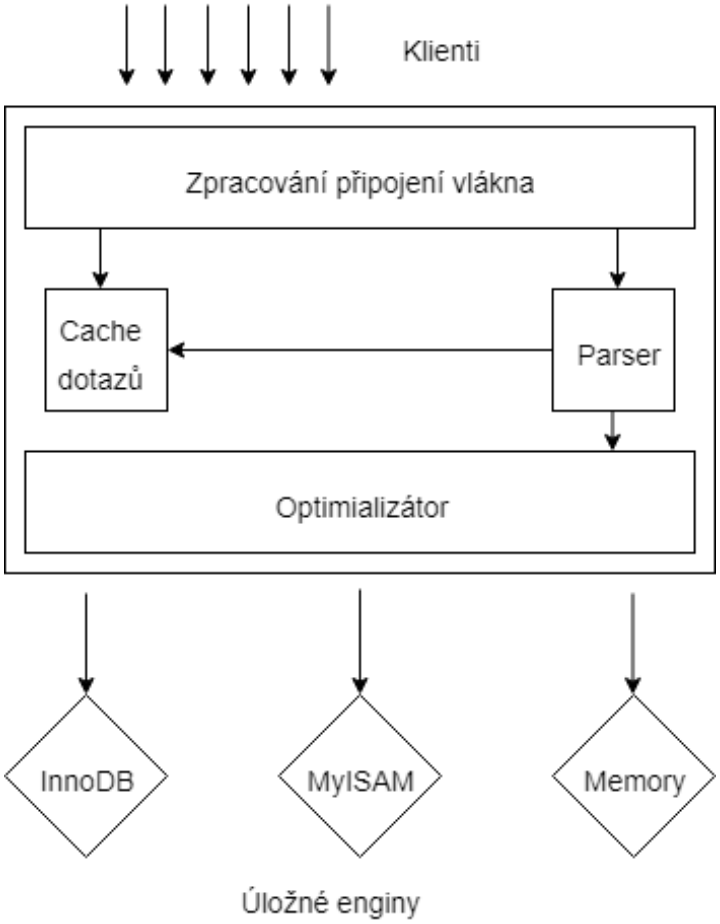
Pojem databáze označuje jakoukoliv uspořádanou kolekci dat, kromě digitální(elektronické) může mít i fyzickou podobu (například zásuvka s dokumenty zařazenými do složek). Digitální databáze se člení dle způsobu ukládání a uspořádávání dat. Nejrozšířenějšími typy jsou:

- Prosté databázové soubory – sekvenční ukládání dat většinou ve formátu prostého textu. Snadné vytváření souborů a přidávání dat, avšak pomalé vyhledávání a riziko poškození souborů. Příkladem může být textový soubor hry Solitaire obsahující výsledky hráčů.
- Hiearchické databáze – ukládají data ve stromové struktuře za pomoci vztahu rodič a dítě. Jde o velmi organizované databáze umožňující rychlé efektivní vyhledávání. Bez znalosti vztahů se však jejich procházení může stát velmi obtížné a s plynoucím časem se může komplikovat i údržba vztahů mezi daty. Příkladem takové databáze je registr systému Windows.
- Dokumentově orientované databáze – ukládají data ve volné podobě a indexují je pomocí klíčových nebo hashovacích hodnot. Navzdory problémům shodným

s prostými databázovými soubory nabízejí celkem dobrou škálovatelnost v rozsáhlých sítích. V databázích totiž nejsou přítomny žádné vztahy mezi daty a vyhledávání tak může být pomalé. Mezi nejoblíbenější databázové systémy tohoto druhu se řadí Redis a CouchDB.

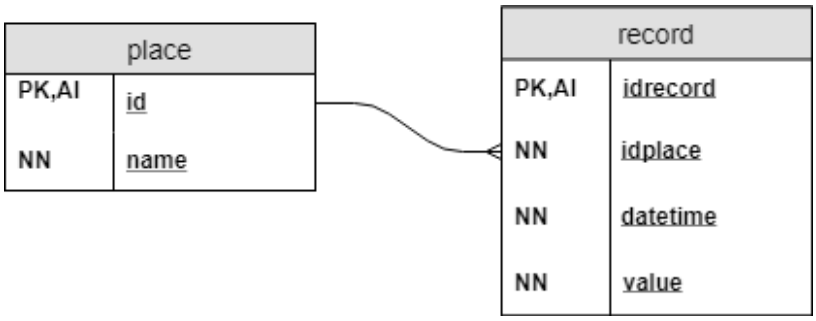
- Relační databáze – ukládají data do tabulek tvořených řádky a sloupci. Tento typ databází podporuje indexování velkých objemů dat, což umožňuje jejich rychlé načítání navzdory často velmi složitým vztahům mezi jednotlivými tabulkami.

U drtivé většiny současných digitálních databází řídí databázový server. Databázový server je aplikace vytvořená za účelem řízení databází, přístupu k datům a práce s nimi. V případě systémů relačních databází se nikdy nepracuje přímo s daty. Na server se odesílají požadavky o přidání, změnu, výmaz či načtení dat. Databázový server pak provede požadované operace a předá nám výsledky. Pro komunikaci s relačními systémy se stal standardem jazyk SQL (Structured Query Language – strukturovaný dotazovací jazyk). Jazyk SQL obsahuje příkazy pro vkládání, načítání a řízení dat, příkazy pro vytváření a údržbu tabulek i příkazy pro správu databází [23].



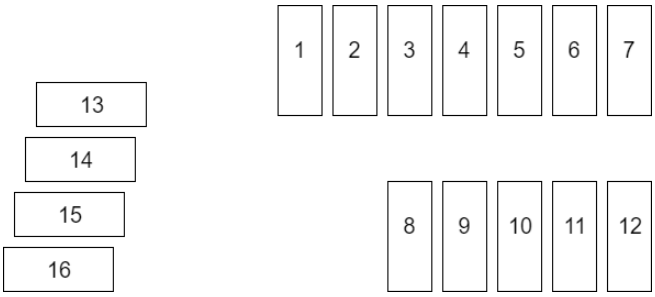
Obrázek 15: Architektura MySQL Serveru (převzato z [23])

MySQL Server se svou architekturou velmi odlišuje od architektur jiných databázových serverů. má široký záběr a je užitečná pro řešení mnoha různorodých úloh. Nejsvrchnější vrstva architektury MySQL serveru obsahuje služby pro komunikaci s klientem. Tyto služby zastřešují většinu potřebných nástrojů klient/server založených na síti. Ve druhé úrovni se nachází veškerá funkcionalita MySQL. Tato vrstva obsahuje kód pro rozbor vstupů (parsing), analýzu, nástroje pro optimalizaci a pro všechny zabudované funkce. Ve třetí vrstvě jsou obsaženy nástroje pro ukládání a zisk všech dat uložených v MySQL Serveru.



Obrázek 16: Model databázového schématu parking

V našem případě bude smyslem databáze evidovat procentuální hodnoty obsazenosti jednotlivých míst v čase. Pomocí dotazu (tzv. query) si pak můžeme vybrat hodnoty jednotlivých míst během časového úseku, nebo stav parkoviště v určitou dobu. V případě, že by zařízení pro zpracování kamerových signálu kromě XML souboru ještě nahráli snímek aktuálního stavu může být taková evidence velmi vhodnou zpětnou vazbou pro kontrolu funkce zpracování signálu.



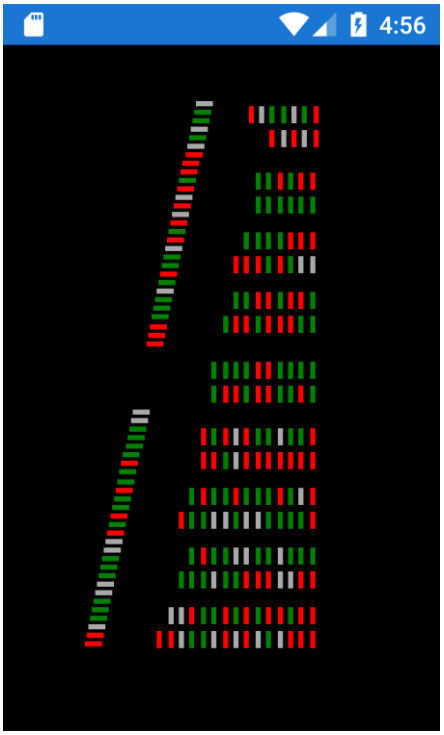
Obrázek 17:Navrhované přiřazování identifikátorů pro záliv A

Za tímto účelem byl na serveru spuštěn MySQL Server 5.7 a bylo na něm vytvořeno schéma *parking*. Windows služba databaseService při svém spuštění zapíše všechny hodnoty do tabulky a pak při každé inspekci zapíše jen změny hodnot. Schéma obsahuje tabulku s místy *place* a tabulky se záznamy *record*. Tabulka *record* má čtyři sloupce *idrecord*, *idplace*, *value* a *datetime*. První tři jsou typu integer a poslední časového typu datetime. Žádný nesmí být při vkládání prázdný. První je identifikační číslo záznamu, autoinkrementuje (s přidáním záznamu vzroste hodnota jeho obsahu) a je primárním klíčem tabulky. Celočíselná hodnota *idplace* odkazuje do tabulky s místy. *Value* je vyhodnocená pravděpodobnost obsazení daného místa a časová známka *datetime* vyjadřuje moment kamerové inspekce. Tabulka *place* se skládá pouze ze dvou sloupců *id* a *name*, přičemž první je identifikační číslo místa v rámci parkoviště a druhé krátký název – například písmeno označující záliv a číslo místa v rámci zálivu. K oběma tabulkám lze samozřejmě přidat další sloupce s doplňujícími informacemi.

4 Mobilní aplikace pod systémem Android

Aplikace obsahuj uživatelské rozhraní a může spouštět kód na pozadí(multi-tasking). Ještě donedávna většina Android mobilních aplikací byla vytvářena v jazyce Java, v poslední době však hlavně spolu se zaměřením na operačním systému nezávislý vývoj roste obliba C# nebo JavaScriptu. Důvodem je podobnost konceptu Android aplikací k Java servletům (program pro tvorbu dynamického webového rozsahu např. vrstva mezi klientem a databází). Místo metody *main* se vytvoří podtřídy některých základních tříd dodávaných pomocí systému Android, které definují čtyři komponenty aplikace a metadata informující Android o těchto podtřídách.

Základním stavebním prvkem uživatelského rozhraní jsou aktivity. Můžete si je představit jako Androidovou podobu okna PC aplikace, nebo stránku klasické webové aplikace. Představuje hlavní část uživatelského rozhraní a v některých případech je přímým vstupem do aplikace (způsob vzájemného odkazování aplikací). Obvykle zabere většinu obrazovky a nechá místo jen pro ikony jako ukazatel síly signálu, nebo hodiny.



Obrázek 18: Aktivita předložené aplikace pro parkování

Životní doba aktivit je velmi krátká a může být kdykoliv ukončena například stisknutím tlačítka Zpět. Oproti tomu služby jsou navrženy pro nepřetržitý běh po určitou dobu bez potřeby

aktivity. Typickým příkladem využití služby je stahování dat, nebo přehrávání hudby bez řídící aktivity.

Další komponentou jsou poskytovatelé obsahu. Ti zpřístupňují data uložená v zařízení pro více aplikací. Android přístup podporuje vzájemnou dostupnost dat mezi aplikacemi, u které lze nastavit vzájemné vztahy a způsoby přístupu. Pro příklad si představte PDF soubor stažený uživatelem, který mu chcete zobrazit. Lze vytvořit poskytovatele obsahu umožňující zobrazit soubor u kterého uživatel spolu s Androidem stanoví, že požadavek na zobrazení PDF se zpracuje danou aktivitou.

Systém nebo aplikace čas od času vysílají relace za různými účely od informace o nízkém stavu baterie po vypnutí obrazovky nebo změny připojení z Wi-Fi na mobilní data. Přijímače vysílání umožňují tyto informace přijímat a náležitě na ně reagovat.

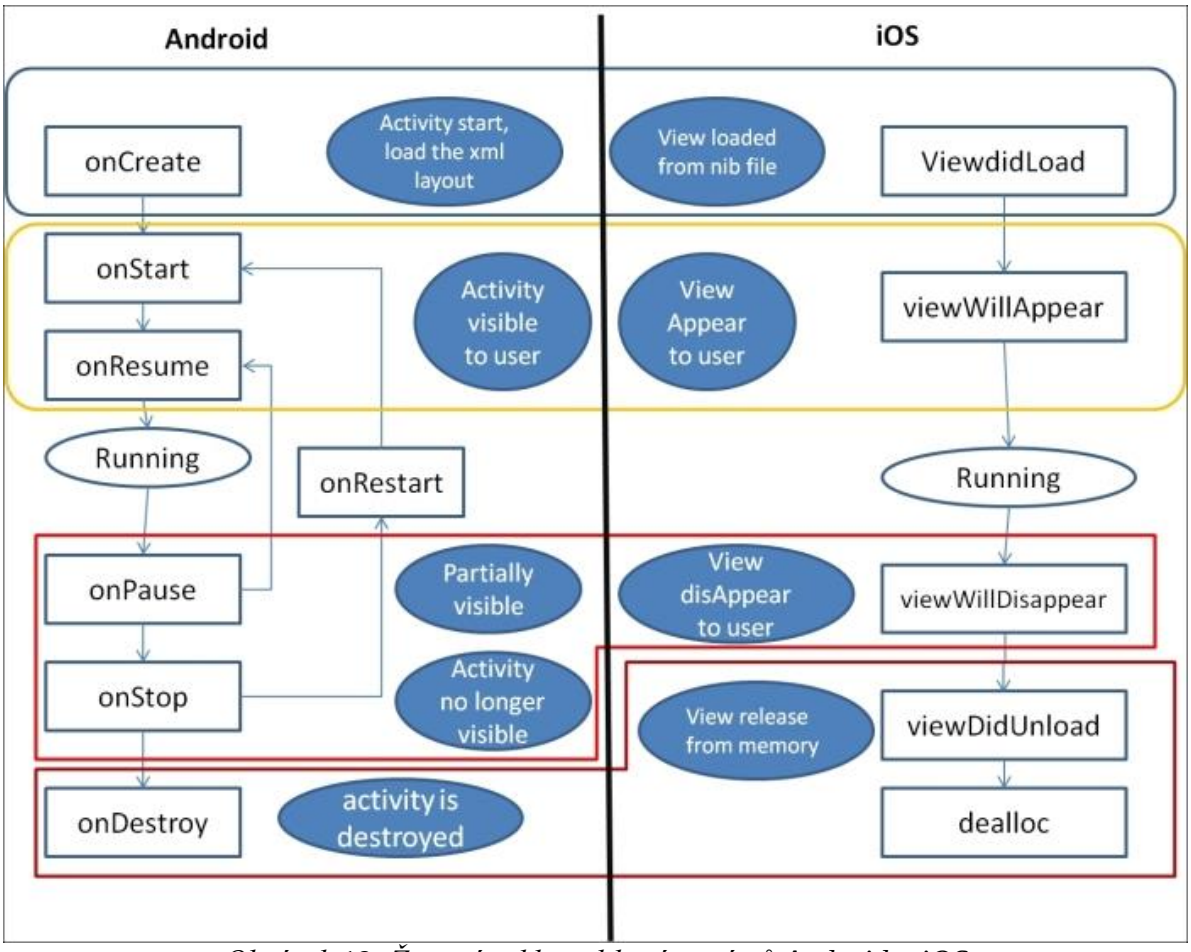
Stěžejním je pro androidové aplikace soubor *AndroidManifest.xml*, v něm se nastavuje obsah aplikace – aktivity, služby atd. Rovněž se zde určují vzájemné vztahy těchto prvků (např. určení spouštěcí aktivity). Dále jsou zde definovány cílové verze operačního systému a potřeby povolení přístupu k různým funkcím telefonu [24].

4.1 Xamarin Forms

Xamarin Forms umožňuje vytvářet aplikace na třech hlavních platformách mobilních zařízení. Je zde přítomna většina standartních prvků uživatelského rozhraní. Jedná se o abstraktní knihovnu pro uživatelská rozhraní. Abstraktní znamená způsob zakrytí detailů implementace daného souboru funkcí a rozdělení zájmů za účelem usnadnit interoperabilitu a nezávislost platformy. Pracuje jako Portable Class Library – PCL (knihovna přenosných tříd) usazená na cílových platformách a zprostředkující uživatelské rozhraní potřebné pro aplikaci. Umožňuje rychlou tvorbu uživatelského rozhraní a zároveň valná část kódu může být sdílená mezi platformami. Xamarin Forms rovněž umožňuje přístup k hardware zařízení za pomoci tzv. injekce úpravy skrz jejich renderery popsanou dále. Abstraktní knihovny umožňují používat na všech platformách pouze prvky vzájemně podobné. PCL obsahuje podsoubor standartních tříd knihoven .NET, které jsou podporovány všemi platformami. To v některých případech může vést ke komplikacím, protože není jisté, zda platforma disponuje stejnou třídou. To se zdá být jasné, pokud uvažujete souborový systém ukládání u PC nebo mobilních telefonů. Oproti tomu embedded zařízení nebo jednoúčelová zařízení mají vlastní verzi .NET frameworku bez

nepotřebných knihoven. Prvky Xamarin Forms se dělí do 4 oblastí: stránky, rozvržení, pohledy a buňky.

Stránky ve většině případů zabírají celou plochu obrazovky. V Androidu mají blíže k rozložení zobrazení *Resource.Layout* a neřadí se k aktivitám. Existuje pět typů stránek: jednoduchou s jedním pohledem, karuselovou umožňující přecházet mezi pohledy přetahováním prstu, navigační s navigační lištou, stránku s dvěma rovinami a kontejnerovou pro přiřazení jednoduchých stránek k tabulce. Pohledy jsou míněny zobrazované prvky jako například tlačítka, popisky, obrázky a podobné. Xamarin Forms obsahuje celkem 19 různých takových objektů. Buňky si lze představit jako zvláštní pohled popisující popisující vykreslování objektů v seznamu, nebo tabulce. Rozvržení je obsah, který umožňuje vkládat další rozvržení nebo pohledy. Existuje jich sedm druhů. *ContentLayout* udrží jen jeden prvek a je užíván jako hlavní třída pro uživatelem definované pohledy. Dalším druhem je rámeček *Frame* obsahující jeden člen, který zarámuje. Pak jsou tu absolutní a relativní rozvržení lišící se ve způsobu umístění objektů. Zásobníkové umisťuje prvky do řady a je asi nejpoužívanější. Jiným případem je *ScrollView* které může být mnohem větší než obrazovka telefonu. Posledním typem rozvržení je mřížka obsahující pohledy organizované do řádků a sloupců.

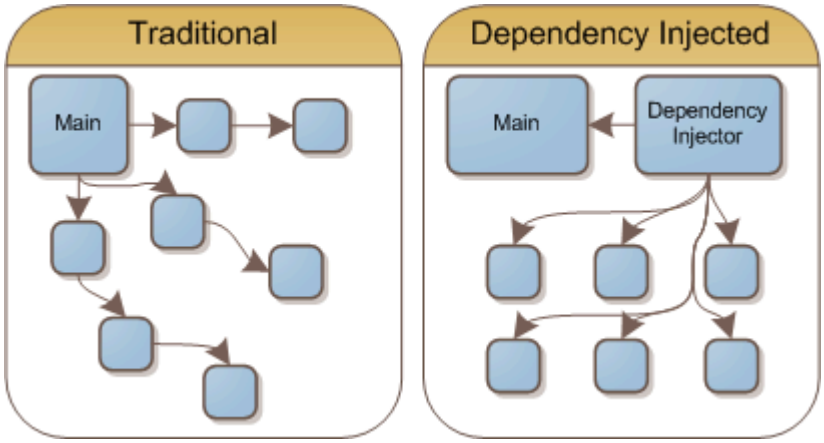


Obrázek 19: Životní cykly aplikací systémů Android a iOS

Z obrázku je patrná shodnost procesu v jednotlivých bodech. Spuštění a zobrazení uživatelského rozhraní, obsluha událostí a konec. Jelikož všechny platformy následují stejný proces událostí a používají stejný jazyk mělo by být možné vytvořit knihovny obsluhující všechny kroky. Problémem může být jejich provázání. Jsou vytvořené tři knihovny se stejnými názvy metod ale přiřazené k jiným elementům uživatelského rozhraní a jsou volány skrz PCL. Uživatelské rozhraní je sestavováno v PCL, to řekne platformě například: „Chci tlačítko ve prostředku pohledu.“. Android a iOS pak použijí prvky ze svých knihoven za tímto účelem. Poté PCL řekne: „Událost kliknutí zavolá metodu v PCL, takže ji přesměruj, když je zavolána.“ i zde je použit životní cyklus ve zjednodušené podobě. Při spuštění aplikace je volána metoda *OnStart*. Když jde aplikace do pozadí nebo se vypne je volána metoda *OnSleep* a po jejím znovu zavolání z pozadí je volána metoda *OnResume*.

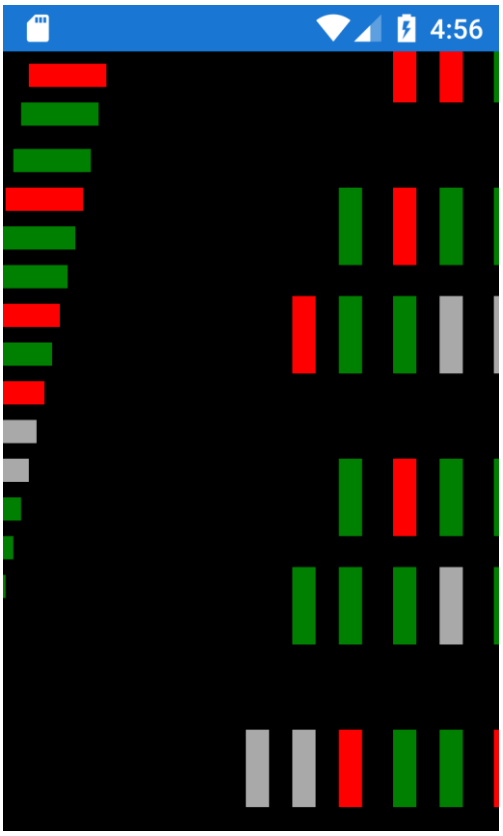
Dalším důležitým mechanizmem Xamarin Forms je inverze kontroly. Jakmile je aplikace spuštěna je vše zastřešováno PCL. Avšak některé činnosti potřebují integrované funkce zařízení nebo jeho časti, a tak je kontrola předána aplikaci (tj. byla invertována). Inverze

kontroly jde ruku v ruce společně s injekcí závislostí platformy, která vkládá informace přímo do PCL. Příkladem může být použití SQLite databáze, kde musí být zapsán soubor, a to pouze PCL nezvládne. V takových případech musí být vytvořena třída rozhraní v rámci PCL [25].



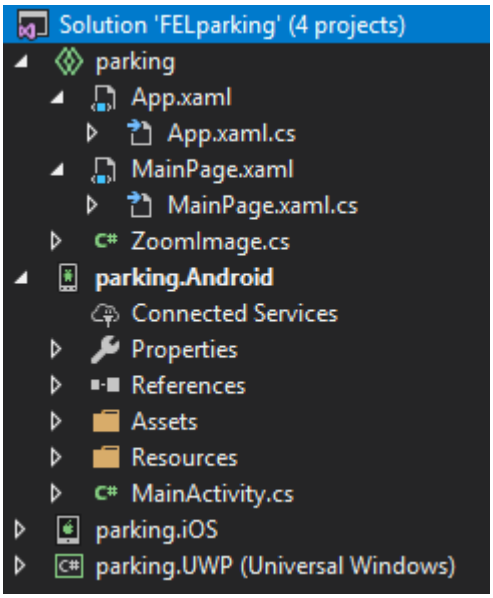
Obrázek 20: Princip inverze ovládání (převzato z [25])

4.2 Mobilní aplikace pro parkovací systém



Obrázek 21: Zobrazení po přiblížení

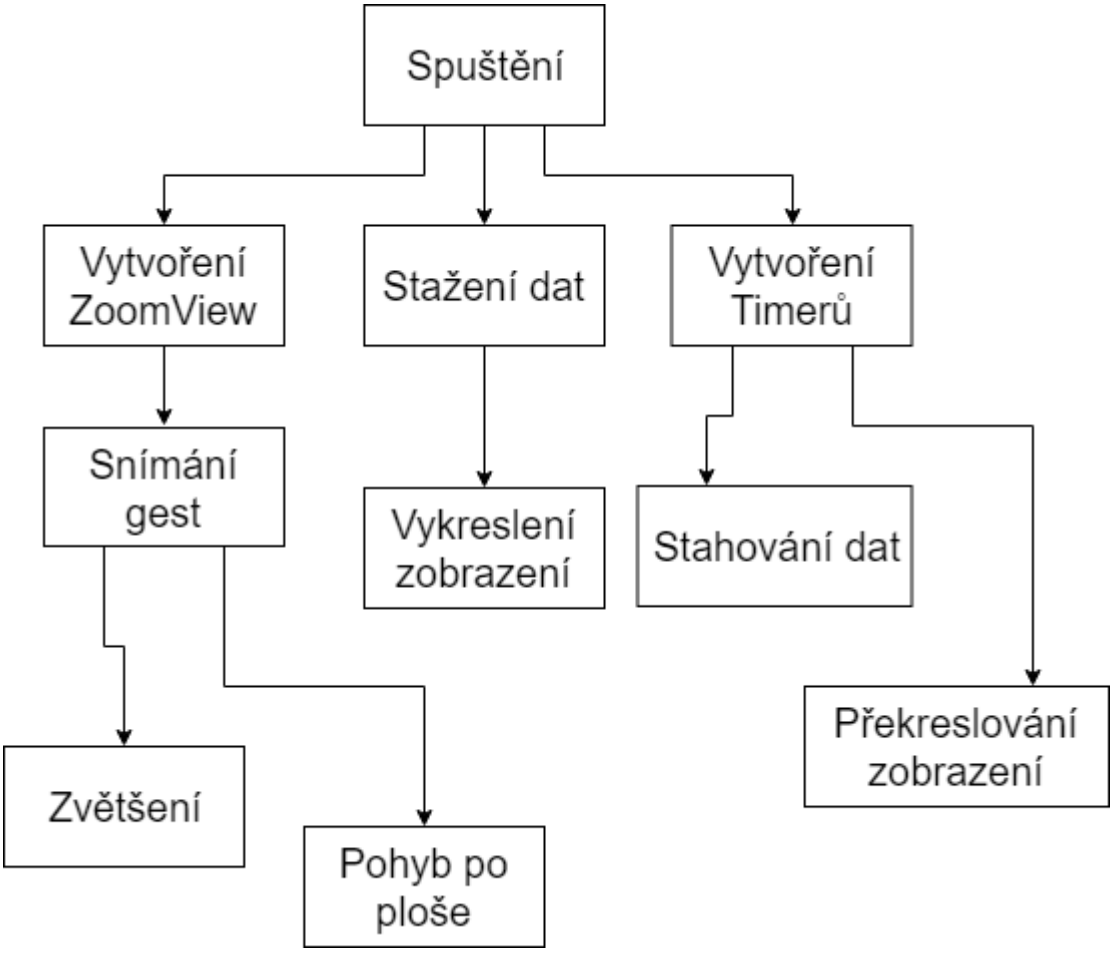
Za účelem zobrazení situace na parkovišti před budovou FEL studentům a zaměstnancům univerzity bylo vytvořeno ve Visual Studiu Community 2017 s pluginem Xamarin multiplatformní řešení *FELparking*. Vzhledem k velikosti parkoviště stačí pro zobrazení situace pouze jedna stránka s možností přiblížení.



Obrázek 22 : Struktura projektu FELparking

Řešení obsahuje čtyři projekty: sdílený *parking* a pak *parking.Android*, *parking.iOS* a *parking.UWP* pro Windows Mobile. Sdílené řešení obsahuje pouze tři objekty. V prvním *App.xaml* jsou přiřazeny externí zdroje, naše aplikace však se žádnými nepracuje, tudíž je tento soubor prázdný. Jeho potomek *App.xaml.cs* obsahuje metody *OnStart, OnSleep* a *OnResume* a odkaz na hlavní stránku v konstruktoru. Lze v podstatě říct, že veškeré údaje o konfiguraci aplikace jsou uloženy v těchto dvou objektech. Hlavní stránkou celé aplikace je stránka *MainPage* a také sestává z dvou objektů. V objektu *MainPage.xaml.cs* je definováno mřížkové rozložení prvků mřížka a absolutní zobrazení do kterého se pak vykreslují jednotlivá místa. Mřížkové rozložení je zde proto, že je zde rovněž vytvořen odkaz na objekt *ZoomImage* a jeho obsah. Ve třídě *MainPage.xaml.cs* je obsažen funkční kód aplikace. Metoda *getDataBlock* získává data z webové služby za použití protokolu HTTP. Na začátku metody je spuštěn požadavek s odkazem na URL adresu služby, dále jsou nastaveny parametry požadavku: formát zprávy JSON a použití HTML příkazu GET. Po získání odpovědi je zpráva přečtena a deserializována na pole celých čísel. Zmíněné pole je pak vstupem metody *drawLayout*, která do absolutního zobrazení vykresluje řádky a sloupce s parkovacími místy. Metoda během vykreslování prochází pole a na základě číselné hodnoty vybarví políčko červeně, šedivě nebo zeleně. Integerové pole je vytvořeno globálně v konstruktoru třídy. Obě zmíněné metody jsou spouštěné vlastními časovači, s mírným rozestupem čímž vznikne vlákno pro obnovu dat a vlákno pro obnovu zobrazení. Ve vlastním vlákně je spuštěna také obsluha zoomu a následného pohybu po zvětšeném obraze. Absolutní zobrazení obsahující vykreslená místa je totiž

v *MainPage.xaml* definováno jako obsah objektu *ZoomImage.cs*. Ve třídě *ZoomImage.cs* jsou vytvořeny potomci objektů pro rozpoznávání gest na obrazovce a jsou jim přiřazeny metody pro obsluhu.



Obrázek 23: Blokový popis hlavní stránky

Podprojekt *parking.Android* obsahuje kód spolu se zdroji(obrázky, seznam textových řetězců, rozložení ovládacích prvků) potřebnými k sestavení aplikace pro operační systém Android. Je zde pouze jedna aktivita *MainActivity.cs* nakonfigurovaná jako spouštěcí se zobrazením obsahu pouze v jedné poloze obrazovky. Dále je zde přiřazen název aplikace, odkaz na obrázek její ikony a metoda *OnCreate* pro vytvoření aktivity. V této metodě je vytvořen hlavní obsah andoidové aplikace odkazující na knihovnu Xamarin Forms spolu s objekty *Tabbar* a *Toolbar* z knihoven Android, do kterých se pak promítá obsah sdíleného projektu. Kdysi býval klíčovým prvkem pro zobrazení androidové aktivity objekt *action bar*. Tento objekt býval používán k navigaci, hledání, používán pro menu v aplikacích pro Android. Ve starších verzích býval tento prvek doporučován pro zmíněné funkce. V novějších verzích operačního systému Android je obsažena komponenta *Toolbar* za účelem nahradit objekt *action*

bar [26] *Toolbar* může být použit kdekoliv v rámci rozložení obrazovky aplikace a je mnohem více přizpůsobitelná. Objekt *Tabbar* má víceméně stejný účel, avšak je orientován na starší verze operačního systému. Oba objekty jsou definovány v souborech složky *layout* zdrojů *Resource* projektu [27]. V této složce jsou také spouštějící ikony aplikace.

Obrázek 24: Ikona Android aplikace

Běh aplikace se po delším spuštění potýká s problémy zadrhávání zobrazení při častější gestikulaci na obrazovce, které se mi nepodařilo úplně vyřešit, Získávání dat ze serverové strany probíhá v pořádku a k aktualizaci zobrazení dojde v dostatečném časovém intervalu. Problém může být způsoben nedostatečnou portací multiplatformního projektu na operační systém Android.

5 Závěr

První část diplomové práce seznamuje s inteligentními parkovacími systémy a uvádí do dané oblasti. Jsou zde uvedeny základní pojmy, principy a způsoby přístupu k problematice a používané technologie. Dále jsou zde zmíněná existující komerční i konceptuální řešení.

Další část je věnována serverovým stranám těchto systémů. V úvodu je popsán jejich účel spolu s úkoly. Také jsou zde podrobněji rozebrané jednotlivé bloky a jejich funkčnost. V další části je předložena navržená struktura XML souboru, který by měl obsahovat výstupní data předchozího bloku detekce míst na parkovišti. Úvod této podkapitoly seznamuje s formátem XML a dále je rozebrána struktura dokumentu. Byl zvolen formát XML vzhledem k jeho přehlednosti a snadnému zpracování dat. Tato struktura obsahuje elementy parkovacích míst s atributem jejich procentuálního obsazení a element počtu volných míst v zálivu které umí algoritmus zpracovat spolu s časovým otiskem.

Dále v je v této části představeno realizované řešení serverové strany a použité technologie pro jeho běh. Jsou zde zmíněny nastavené výjimky ve firewallu, nastavení cloudové testovací instance serveru, instalace a nastavení serveru FileZilla spolu s důvody proč se cloudové technologii v reálném použití vyhnout. Na serveru byla spuštěna vytvořená služba Widnows periodicky snímající vstupní data, zapisující je do databáze a předávající je klientům mobilní aplikace. Seznámení s použitými technologiemi Windows služby, webové služby a databáze MySQL a popis předkládaných řešení je ve zbytku kapitoly. Tato Windows služba může být spuštěna i na operačním systému Windows 10, takže může být jako server použit i obyčejný počítač. Smysl databáze v tomto případě je k provádění různých statistik vytíženosti a sběru dat z parkoviště. Spolu se snímky jednotlivých kamerových inspekcí pak mohou být tato data skvělou zpětnou vazbou pro sledování a korekce funkce algoritmu zpracování obrazu z kamer. Databáze může rovněž být zdrojem dat pro webovou aplikaci. Jednotlivé bloky serverové strany jsou rozšiřitelné o další parametry či další funkce.

Poslední kapitola je věnována aplikaci pro operační systém Android zobrazující situaci na parkovišti parkujícím uživatelům. V úvodu je popsán princip funkce aplikací pro Android a přístup k jejich programování. Další část je pak věnována programování aplikací pro více platforem a použité technologii Xamarin. Pomocí Xamarinu byl vytvořen projekt portovatelný na tři hlavní operační systémy mobilních zařízení Android, iOS, a Windows Phone. Realizována byla pouze verze pro Android. Aplikace získává v pětivteřinových intervalech data z webové služby serverové strany. Pozorováním parkoviště jsem zhodnotil, že nejrychlejší parkování trvá zhruba 8-10 sekund, naopak parkování minimálně 15 sekund a v práci věnující

se zpracování signálu byl interval inspekce navržen na 5 sekund, což se zdá s přihlédnutím na zmíněné časy vhodné. Následně je na obrazovce vykresleno rozložení míst na parkovišti, kde místa jsou zabarvena do tří barev v závislosti na jejich obsazenosti, nebo nerozhodnutelném stavu. Aplikace dále umožňuje si zobrazení přiblížit a v přiblíženém zobrazení se pohybovat. Její nevýhodou je příležitostné trhání zobrazování způsobené možná nesprávnou optimalizací kódu nebo nedostatečnou portací pro Android.

Tato práce je dalším krokem v projektu inteligentního parkovacího systému pro parkoviště FEL/ZČU. Je zde představeno možné řešení získání a zpracování dat z předchozí práce věnované videodetekci. Jednotlivé bloky řešení a jejich technologie jsou podrobně popsány spolu s postupy k jejich implementaci. Bloky jsou snadno rozšiřitelné k další manipulaci nebo úpravám. Dále byla předložena mobilní aplikace získávající data a zobrazující je v jednoduchém grafickém rozhraní. Představenou aplikaci lze do budoucna zužitkovat pro nejrozšířenější operační systémy chytrých telefonů. U jednotlivých bloků parkovacího systému lze v budoucnu použít jiné technologie a komunikace zbylých bloků zůstane zachována díky použitým standardům

6 Seznam literatury a informačních zdrojů

[1] „V ČR je registrováno celkem 5 368 660 aut,“ 25 1 2017. [Online]. Dostupné z: http://portal.sda-cia.cz/clanek.php?id=5794&v=m. [cit. 2018-01-9].

[2] MATHUR, Suhas, Tong JIN, Nikhil KASTURIRANGAN, Janani CHANDRASHEKHARAN, Wenzhi XUE, Marco GRUTESER a Wade TRAPPE. ParkNet: Drive-by Sensing of Road-Side Parking Statistics [online]. WINLAB, Rutgers University, North Brunswick, NJ, USA, 2010, , 14 [cit. 2018-01-9]. Dostupné z: http://www.winlab.rutgers.edu/~gruteser/papers/mathur_parknet10.pdf

[3] www.MarshProducts.com, „The Basics of Loop Vehicle Detection,“ 10 11 2000. [Online]. Dostupné z: http://www.marshproducts.com/pdf/Inductive%20Loop%20Write%20up.pdf. [cit. 2018-01-9].

[4] V. MARKEVICIUS, N. DANGIRUTIS , Z. MINDAUGAS, D. ANDRIUKAITIS, A. VALINEVICIUS a M. CEPENAS, „Dynamic Vehicle Detection via the Use of Magnetic Field Sensors: NCBI,“ National Center for Biotechnology Information, 16 1 2016. [Online]. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4732111/. [cit. 2018-01-9].

[5] A. PRASAD, U. SHARATH a B. AMIT, „Fiber Bragg Grating sensor instrumentation for parking space occupancy management,“ v *2012 International Conference on Optical Engineering (ICOE)*, Belgaum, Indie, 2012.

[6] A. KIANPISHEH, N. MUSTAFFA, P. LIMTRAIRUT a P. KEIKHOSROKIANI, „Smart Parking System (SPS) Architecture Using Ultrasonic Detector,“ *International Journal of Software Engineering and Its Applications*, pp. 52-58, 3 7 2012.

[7] B. XU, O. WOLFSON, J. YANG, L. STENNET, P. S. YU a P. C. NELSON, „Real-Time Street Parking Availability Estimation,“ v *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on Mobile Data Management*, Milán, Itálie, 2013.

[8] N. DESHPANDE, „Parking Management System for Georgia Tech,“ Georgia Institute of Technology, Atlanta, 2014.

[9] B. L. KOCHANÍČEK, „Seznámení s RFID čipy,“ 16 2 2011. [Online]. Dostupné z: https://coptkm.cz/portal/?doc=12149&docGroup=179&cmd=0&instance=1. [cit. 2018-

01-14].

[10] K. GANESAN a K. VIGNESH, „Automated Parking Slot Allocation using RFID Technology,“ School of Computing Sciences, Vellore, Indie, 2007.

[11] SAK, Tomáš. *Obrazová analýza kamerových dat parkovacích stání.* Plzeň, 2015. Diplomová práce. Západočeská Univerzita v Plzni. Vedoucí práce Ing. Vladimír Pavlíček, Ph.D.

[12] R. PECÁK, „Test parkovacích zón v Praze,“ Hospodářské Noviny, 25 8 2016. [Online]. Dostupné z: https://domaci.ihned.cz/c1-65414030-test-parkovacich-zon-v-praze-automaty-jsou-tu-daleko-od-sebe-obcas-jsou-zony-nesmyslne-znacene. [cit. 2018-01-21].

[13] GRAZIOLI, Alessandro, Marco PICONE, Francesco ZANICHELLI a Michele AMORETTI. Collaborative Mobile Application and Advanced Services for Smart Parking. In: 2013 *IEEE 14th International Conference on Mobile Data Management* [online]. IEEE, 2013, 2013, s. 39-44 [cit. 2018-01-29]. DOI: 10.1109/MDM.2013.63. ISBN 978-0-7695-4973-6. Dostupné z: http://ieeexplore.ieee.org/document/6569060

[14] N. PALDE, C. NAWALE a S. KUTE, „Car Parking System an Android Approach,“ *International Journal of Innovative Research in Computer*, sv. 4, č. 3, pp. 2953-2958, 4 2016.

[15] S. N. SHINDE, K. V. SHINDE, R. D. NAGPURE, A. S. TUPKAR a M. S. ANKOSHE, „An Android Application for Parking Management and Dissemination System,“ *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, sv. 4, č. 3, pp. 1075-1080, 2015.

[16] „Parkování v Praze může být oříšek, pomohou s ním dvě nové české aplikace Jablíčkář.cz,“ Text Factory s. r. o., 10 1 2017. [Online]. Dostupné z: https://jablickar.cz/parkovani-v-praze-muze-byt-orisek-pomohou-s-nim-dve-nove-ceske-aplikace/. [cit. 2018-01-25].

[17] HOLUBOVÁ, Irena a Jaroslav POKORNÝ. XML technologie: principy a aplikace v praxi. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2725-7.

[18] STANEK, William R. Microsoft Windows server 2012 inside out. Redmond, Wash.: Microsoft Press, c2013. ISBN 978-0-7356-6631-3.

[19] MINASI, Mark. Mastering Windows server 2012 R2. Indianapolis, Indiana: Sybex, 2014.

ISBN 978-1-118-28942-6.

[20] JECHA, Tomáš. Programování windows services. In: Dotnetportal.cz [online]. 24.01.2012 [cit.2018-04-08] . Dostupné: https://www.dotnetportal.cz/clanek/194/Programovani-Windows-Services

[21] KAČMÁŘ, Dalibor. *Programujeme .NET aplikace ve Visual Studiu .NET*. Praha: Computer Press, 2001. Všechny cesty k informacím. ISBN 80-7226-569-5.

[22] LÖWY, Juval. Programming WCF services. 3rd ed. Sebastopol: O'Reilly, 2010. ISBN 978-0-596-80548-7

[23] BORONCZYK, Tim. MySQL okamžitě. Přeložil Milan DANĚK. Brno: Computer Press, 2016. ISBN 978-80-251-4737-5.

[24] MURPHY, Mark L. The busy coder's guide to Android development. 2nd ed. United States: CommonsWare, 2009. ISBN 978-0-9816780-0-9.

[25] JOHNSON, Paul F. Cross-platform UI Development with Xamarin.Forms. 1. Birmingham: Packt Publishing, 2015. ISBN 978-1-78439-119-5.

[26] Toolbar. Xamarin.com [online]. 2018, 3.1.2018 [cit. 2018-04-29]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/android/user-interface/controls/tool-bar/

[27] Tabbed Layouts with the ActionBar: Using the ActionBar for Tabs in Xamarin.Android. Xamarin [online]. Microsoft, 2018, Květen 2017 [cit. 2018-05-16]. Dostupné z: https://developer.xamarin.com/guides/android/user_interface/tab_layout/actionbar/

7 Příloha A

Ukázka XML dokumentu pro záliv A

<data>
<datetime>23 / 4 / 2018 19:17:50</datetime>
<count>11</count>
<place value="30">1</place>
<place value="43">2</place>
<place value="62">3</place>
<place value="86">4</place>
<place value="59">5</place>
<place value="78">6</place>
<place value="12">7</place>
<place value="8">8</place>
<place value="46">9</place>
<place value="30">10</place>
<place value="44">11</place>
<place value="5">12</place>
<place value="47">13</place>
<place value="68">14</place>
<place value="85">15</place>
<place value="42">16</place>
</data>

8 Příloha B

Obsah přiloženého CD

1. **DP_Libor_Peleška_E15N0036P.pdf**

celkové znění závěrečné práce, shodné s tištěnou podobou

2. **Složka data**

Obsahuje ukázkové vstupní soubory pro další zpracování

3. **databaseService.sln**

Řešení služby Windows Service ve Visual Studio Community 2017

4. **FELparking.sln**

Více platformní projekt mobilní aplikace Visual Studio Community 2017