UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Ariel David Jančar

Mobilna aplikacija za pomoč pri iskanju izdelkov v zaprtem prostoru

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana 2018

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Ariel David Jančar

Mobilna aplikacija za pomoč pri iskanju izdelkov v zaprtem prostoru

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Damjan Vavpotič

Ljubljana 2018

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Mobilna aplikacija za pomoč pri iskanju izdelkov v zaprtem prostoru.

Opis teme: V okviru diplomske naloge predstavite problem navigacije v zaprtih prostorih, kjer ni mogoče loviti signala globalnih navigacijskih sistemov. Kratko predstavite možne rešitve omenjenega problema, nato pa analizirajte obstoječa programska ogrodja za navigacijo v zaprtih prostorih. Na podlagi opravljene analize izberite najprimernejše ogrodje in z njegovo uporabo razvijte prototip aplikacije, ki bo omogočala kupcem lažje iskanje izdelkov v trgovini. Delovanje prototipa preizkusite in kritično ovrednotite rezultate dela.

Rad bi se zahvalil mentorju za vso pomoč tekom pisanja diplomskega dela. Prav tako, bi se zahvalil družini in punci, za vso finančno in moralno podporo tekom študija.

Kazalo

Kazalo slik

Kazalo tabel

Seznam uporabljenih kratic

kratica	angleško	slovensko
GPS	Global positioning system	globalni sistem pozicioniranja
IPS	Indoor positioning system	sistem za navigacijo v zaprtem prostoru
XML	Extensible markup language	razširljivi označevalni jezik
API	Application programming interface	aplikacijski programski vmesnik
WiFi	Wireless local area networking	brezžična tehnologija
SQL	Structured Query Language	strukturiran povpraševalni jezik
SDK	Software development kit	paket za razvoj programske opreme
OS	Operation system	operacijski sistem

Povzetek

Naslov: Mobilna aplikacija za pomoč pri iskanju izdelkov v zaprtem prostoru

V diplomskem delu smo analizirali in primerjali več ogrodij za navigacijo v zaprtem prostoru in z uporabo izbranega ogrodja razvili prototip aplikacije za iskanje izdelkov v zaprtem prostoru. Preizkusili smo ga v prostorih Fakultete za računalništvo in informatiko na različnih mobilnih napravah z operacijskim sistemom Android. V delu smo analizirali razvojna ogrodja MapsIndoors SDK, IndoorAtlas SDK, Indoo.rs SDK in jih med seboj primerjali. Pri izdelavi prototipa je bilo uporabljeno razvojno ogrodje IndoorAtlas SDK, ki vključuje tudi spletni vmesnik za nadzor aplikacij in lokacij, ter mobilna aplikacija MapCreator 2 za ustvarjanje podatkov o prostoru na podlagi geomagnetnih zapisov, akustičnih signalov in signalov oddajnikov WiFi ter Bluetooth. Poleg tega smo razvili tudi lastno programsko logiko za izračun kota med smerjo hoje uporabnika in njegove interesne točke z uporabo pospeškomera in giroskopa.

Ključne besede: navigacija v zaprtih prostorih, Android, prototip, IndoorAtlas SDK, MapCreator 2, pospeškomer, giroskop, magnetno polje, senzorji

Abstract

Title: A mobile application assisting in locating products indoors

This work presents the analysis and comparison of different development kits that provide solutions for navigation indoors. The work also provides a more in-depth overview of the development kit chosen for the development of a prototype application for locating products indoors. The prototype was tested on the premises of Faculty of Computer and Information science, on several different mobile devices with Android mobile operating system. We analysed and compared MapsIndoors SDK, IndoorAtlas SDK and indoo.rs SDK development kits. The development kit used for the prototype was IndoorAtlas SDK, which includes a web interface, to control applications and locations, and a mobile application MapCreator 2, for creating indoor data based on geomagnetic fields, acoustic signals, Bluetooth signals and WiFi signals. We have also developed our own program logic that calculates the angle between the users path and his point of interest. This was achieved with the use of accelerometer and gyroscope.

Keywords: Indoor navigation, Android, prototype, IndoorAtlas SDK, MapCreator 2, accelerometer, gyroscope, geomagnetic field, sensors

1 Uvod

Navigacijski sistemi so že leta dolgo del našega vsakdana. Pomagajo nam, da se laže orientiramo na neznanem območju oz. hitreje najdemo neko točno določeno lokacijo. Ključna omejitev tovrstnih sistemov je, da so uporabni le v zunanjem okolju, saj temeljijo na sistemu GPS.

Sistem GPS je pod imenom TRANSIT leta 1959 razvila ameriška vojna mornarica. Danes ima sprejemnik signala GPS že vsak pametni telefon, zato ga lahko uporabljajo tudi aplikacije za navigacijo v zunanjem okolju. Sistem deluje najbolje, kadar med sprejemnikom in oddajnikom signala GPS ni preprek. V zaprtih prostorih pa je signal šibkejši in popačen zaradi prehoda skozi različne materiale, kar povzroči napačne ali nepravilne izračune [23].

Vse več sprejemnikov signala GPS omogoča tudi sprejem signala GLONASS (angl. Global Navigation Satellite System), ki ga je leta 1976 razvila ruska vesoljska agencija. Ruski sistem v primerjavi s sistemom GPS ponuja boljšo natančnost, saj lahko lokacijo določi do treh metrov natančno. V praksi se uporabljata oba sistema hkrati, kar pomeni, da naprava sprejema signale satelitov obeh sistemov (skupaj 55 satelitov), s čimer izboljša natančnost lokacije in pokritost s signalom. Seveda pa tudi v tem primeru signali še vedno potujejo proti sprejemnikom signala s satelita, zato le-ti dobro delujejo le v odprtem prostoru [18].

Sistem notranjega pozicioniranja IPS (angl. Indoor Positioning System) odpravlja te težave, saj za lociranje ljudi ali objektov v zaprtem prostoru uporablja akustične ter elektromagnetne signale in geomagnetna polja. Standardni sistem notranjega pozicioniranja ne obstaja, obstajajo pa nestandardne rešitve različnih komercialnih ponudnikov. [4]

V diplomskem delu so predstavljene težave z navigacijo v zaprtih prostorih, predstavljene pa so tudi možne rešitve. Našteta in analizirana so razvojna ogrodja, ki vključujejo programsko logiko reševanja opisanih težav. Na podlagi analize razvojnih ogrodij smo izbrali najprimernejšega ter ga uporabili pri izdelavi prototipa. Prototip je izdelan v obliki mobilne aplikacije, ki z uporabo senzorja za pospešek, senzorja za magnetno polje, giroskopa, mikrofona in sprejemnikov elektromagnetnih valov določi lokacijo v zaprtem prostoru.

V 2. poglavju opredelimo težave in možne rešitve navigacije v zaprtih prostorih. V 3. poglavju predstavimo izbrano ogrodje za navigacijo v zaprtih prostorih. V 4. poglavju se osredotočimo na prototip mobilne aplikacije in podrobno opišemo njegov čelni in zaledni sistem. Delo se v 5. poglavju zaključi z opisom testiranja prototipa aplikacije.

2 Problem navigacije v zaprtih prostorih

Sprejemnik signala GPS je danes na voljo skorajda vsem, saj je vgrajen v vsak nov pametni telefon. Zahvaljujoč temu si lahko pred odhodom na pot v mobilno napravo zabeležimo cilj, do katerega nas nato vodi s pomočjo sistema GPS. Ker pa se signali sistema GPS prenašajo po zraku na takšni frekvenci in jakosti signala, ki ne omogoča optimalnega prehoda skozi trdne snovi, sistem GPS ne deluje, ko vozimo skozi tunel ali parkiramo v parkirni hiši. Po prihodu iz tunela ali parkirne hiše pa signal sistema GPS hitro vzpostavi stik s sprejemnikom signala v mobilni napravi in nas vodi naprej do cilja [1].

Toda kadar smo dlje časa v zaprtem prostoru, nam sistem GPS pri lociranju stvari ali prostorov v velikem objektu ne more več veliko pomagati. S to težavo se najpogosteje srečujejo v nakupovalnih središčih, muzejih, bolnišnicah ali drugih objektih, v katerih bi lahko bila vpeljava navigacije uporabna. V nadaljevanju se bomo osredotočili na navigacijo v trgovskih središčih. Predstavili bomo, kako lahko vpeljava navigacije pomaga pri iskanju izdelkov ali prostorov v notranjosti večjih objektov.

Orientacija na javnih prostorih

Veliki javni prostori so praviloma opremljeni s številnimi obvestilnimi tablami za obiskovalce in informacijami o njihovih interesnih točkah. Vsa ta obvestila - od označitve vhodov in izhodov, toaletnih prostorov, različnih poslovnih prostorov do zasilnih izhodov - so za obiskovalce nujna, saj jim pomagajo pri lažjem orientiranju v prostoru, še posebej ko se nekje znajdejo prvič. V primeru javnih prostorov, kot so npr. parkirne hiše, muzeji, trgovine, bolnišnice itd., so za obiskovalce ključne informacije o interesnih točkah – kje se nahaja določeno parkirno mesto, muzejski eksponat, polica z izdelki iz ekološke pridelave ali bolniška soba št. 16.

Z vidika obiskovalcev se problemi navigacije v zaprtih javnih prostorih spreminjajo glede na značilnosti prostora, v katerem se nahajajo. Na primer, v parkirnih hišah, trgovinah ter njim podobnih prostorih, pri katerih gre za večje število podobnih interesnih točk, lahko obiskovalce zmedejo številne skupine interesnih točk, zaradi česar izgubijo sled o točki, ki jo pravzaprav iščejo. V bolnišnicah, na letališčih in kar je še podobnih objektov, za katere so značilni dolgi hodniki, lahko obiskovalci obtičijo v labirintu različnih poti. Ko se to zgodi,

morajo poiskati pomoč zaposlenega ali pa se vrniti na zadnjo znano točko, ki je v primeru bolnišnic ali letališča največkrat vhod v bolnišnično oz. letališko stavbo.

Okorna navigacija v zaprtih prostorih je sicer lahko mučna za obiskovalce, vseeno pa lastnikom prostora lahko prinese določene koristi. Kljub slabim oznakam na koncu potrošnika vendarle pripeljejo tudi do težje dostopnih delov trgovskega centra.

Z izboljšanjem navigacije v zaprtem prostoru lahko ustrežemo obema ciljnima skupinama: obiskovalcem omogočimo enostavnejše in hitrejše iskanje izdelkov ali poti, lastnikom pa možnost lokacijskega oglaševanja, s katerim lahko še vedno vplivajo na odločitve potrošnikov.

2.1 Reševanje problema navigacije v zaprtih prostorih

Določanje uporabniške lokacije je lahko zaradi nenehnega premikanja uporabnika in števila razpoložljivih lokacijskih virov zapleteno. Napake, ki pri tem nastajajo, so lahko posledica [20]:

- raznovrstnosti lokacijskih virov

  Lokacijski viri Bluetooth, GSM in WiFi sistemi lahko predlagajo namige o uporabniški lokaciji. Toda kadar jih imamo na voljo več, se moramo odločiti, katerim zaupamo in katere lahko uporabimo. Izbira lokacijskih virov vpliva na natančnost in hitrost ugotavljanja lokacije, kakor tudi na porabo baterije mobilne naprave.

- premikanja uporabnika

  Zaradi spreminjanja moramo uporabniško lokacijo periodično ocenjevati.

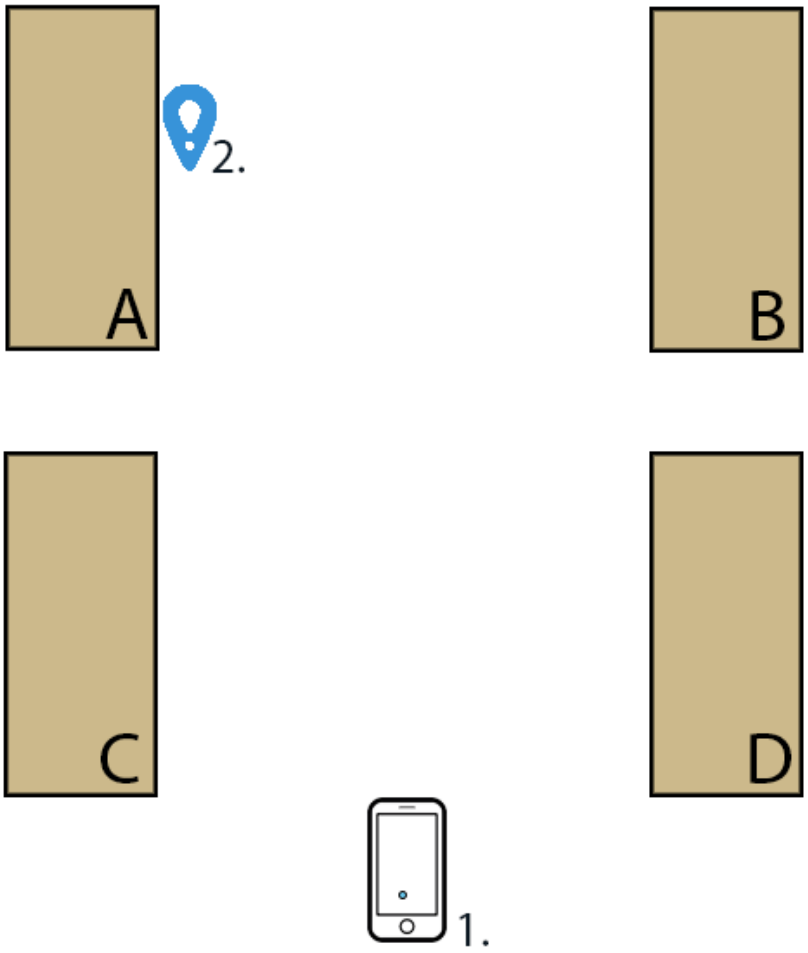
- spremenljive natančnosti

  Lokacijski viri nimajo usklajene natančnosti. Ocena lokacije, ki smo jo določili s pomočjo enega lokacijskega vira pred desetimi sekundami, je lahko bolj natančna od novejše ocene istega ali drugega vira.

Za vodenje uporabnika do interesne točke (njegovega cilja) moramo poznati okolje, v katerem se nahaja. Te podatke lahko pridobimo s pomočjo senzorjev v mobilnih napravah. Slika 1 predstavlja zaprt prostor, kjer se nahajajo:

- mobilna naprava uporabnika (točka 1),

- interesna točka (točka 2),
- razni neprehodni objekti pod znaki od A do D.



Slika 1: Mobilna naprava in interesna točka v neznanem prostoru

Slika 1 prikazuje situacijo, v kateri želimo uporabnika voditi do njegove interesne točke. O okolju nimamo dovolj podatkov, da bi lahko uporabnika pravilno usmerjali, saj so s stališča mobilne naprave objekti A-D med seboj enaki, razdalja in kot med napravo ter interesno točko pa sta neznana. Za pravilno vodenje uporabnika moramo torej spoznati različne načine in rešitve, s katerimi lahko zberemo dovolj podatkov o napravi in okolju.

Uporaba senzorjev mobilnih naprav

Pri uporabi senzorjev mobilnih naprav je pomembno, da poznamo strojno in programsko opremo mobilne naprave. Mobilne naprave z operacijskim sistemom Android nudijo uporabo naslednjih kategorij strojne opreme [3]:

- senzorji gibanja, ki merijo pospešek in rotacijske sile vzdolž treh osi (x, y in z) (pospeškomer, giroskop, senzor korakov, gravitacijski senzorji);
- okoljski senzorji, ki merijo različne okoljske parametre, kot npr. temperatura zraka, tlak, razsvetljava in vlažnost (barometer, termometer, kamera);
- položajni senzorji, ki ugotavljajo položaj naprave (orientacijski senzorji, sprejemniki signalov WiFi in Bluetooth, senzor elektromagnetnih valov in geomagnetnih polj).

Hitrost in smer gibanja uporabnika lahko določimo s pomočjo senzorjev gibanja, jakosti in smeri signalov. Pri določanju njegove lokacije pa si lahko pomagamo s položajnimi senzorji in triangulacijo. Triangulacija je način določanja položaja neznane točke v ravnini s pomočjo trikotniških pravil - pri čemer potrebujemo za izračun najmanj dve znani točki. Lokacijo uporabnika lahko precej enostavneje določimo s pomočjo geomagnetnih polj, z uporabo okoljskih senzorjev pa lahko še dodatno obogatimo nabor vrednosti podatkov o okolju.

Načini zbiranja podatkov o okolju

Podatke o napravi in okolju zbiramo s pomočjo strojne opreme mobilnih naprav. Zanimajo nas predvsem podatki o elektromagnetnih valovih, geomagnetnem polju in premikanju naprave. Načinov, kako jih zbiramo, pa je več:

- pospeškomer, giroskop in senzor korakov

Za določanje smeri in hitrosti gibanja uporabnika lahko uporabljamo pospeškomer in giroskop. S pomočjo prvega ugotovimo, ali se uporabnik premika, s pomočjo drugega pa, v katero smer se premika. Za sledenje gibanja uporabnika lahko uporabimo tudi senzor korakov. Pospeškomer in giroskop dajeta dovolj točne in natančne rezultate za potrebe lociranja v zaprtem prostoru. Pri uporabi senzorja korakov se lahko točnost zmanjša, saj je tudi počep na mestu prepoznan kot korak v določeno smer. Zato je ta način za določanje lokacije v zaprtem prostoru manj primeren.

- geomagnetno polje

Vsak prostor ima enoličen geomagnetni zapis, ki ga ustvari Zemeljsko magnetno polje. Struktura in elementi objektov, kot so zidovi, stebri in druge železne ali železobetonske konstrukcije, zemeljske geomagnetne valove sicer popačijo, toda tudi v popačeni obliki se s časom ne spreminjajo in ravno zato so zelo primerni za uporabo pri navigaciji. Preberemo ga lahko s senzorjem geomagnetnega polja. Z uporabo tega senzorja se lahko izognemo uporabi ločene strojne opreme, kot so oddajniki WiFi in Bluetooth signalov (angl. Bluetooth Beacons) [4].

Ideja za navigacijo z uporabo geomagnetnih polj izvira iz živalskega sveta, saj živali (ptice, netopirji, čebele, lisice ipd.) za navigacijo v prostoru uporabljajo geomagnetno polje našega planeta. Mobilne naprave pa lahko ‘berejo’ geomagnetne zapise notranjih prostorov.

- elektromagnetni valovi

Elektromagnetni valovi so valovi vidne svetlobe, Bluetooth signali, WiFi signali in radijski valovi. Med seboj se razlikujejo le v frekvenci in valovni dolžini. Pri navigaciji v zaprtem prostoru si zato lahko deloma pomagamo z radijskimi valovi, deloma pa tudi z vidno svetlobo.

WiFi in Bluetooth signali se oddajajo na frekvenci 2,4 GHz (novejši oddajniki WiFi signalov delujejo tudi na frekvenci 5 GHz). S sprejemniki WiFi in Bluetooth signalov ter s pomočjo triangulacije lahko določamo lokacijo uporabnika. Znane točke za izračun triangulacije so oddajniki WiFi in Bluetooth signalov, ki jim položaj določimo ob namestitvi. Neznana točka pa je uporabniška naprava, ki ji položaj izračunamo. Oddajniki signalov Bluetooth in WiFi repetitivno oddajajo lasten ID in morebitne dodatne podatke, kot npr. opozorila ali podrobnejši podatki o lokaciji naprave. Ravno na podlagi informacije o lokaciji oddajnikov in jakosti signalov pa lahko izračunamo lokacijo uporabnika.

Za zbiranje podatkov o okolju lahko uporabimo tudi ostale radijske valove (AM in FM valovi) in si tako z beleženjem jakosti radijskih valov obogatimo nabor znanih signalov na neki lokaciji. Toda samo na podlagi jakosti radijskih valov lokacije ne moremo določiti. Če pa zabeleženo jakost radijskih valov shranimo poleg že zbranih podatkov o okolju, kot so npr. geomagnetni zapisi in signali elektromagnetnih valov, lahko za določanje lokacije neke točke v prostoru precej obogatimo nabor znanih vrednosti in to uporabimo pri kasnejših izračunih.

Temu naboru lahko dodajamo tudi podatke o svetlobi, ki jo lahko spremljamo s pomočjo svetlobnih senzorjev in kamer. Svetloba se v prostoru spreminja, zato z njeno pomočjo lokacije ne moremo določati. Lahko pa na ta način - podobno kot z radijskimi valovi - obogatimo pridobljene informacije o prostoru. Temu lahko dodamo tudi zvočne zapise,

zbrane s pomočjo mikrofona. To je uporabno predvsem v prostorih, kjer preko zvočnikov oddajajo periodična obvestila, saj lahko z mikrofonom beležimo jakosti zvoka v prostoru. Tako pridobimo novo spremenljivko, ki jo kasneje uporabimo pri določanju lokacije.

2.2 Razvojna ogrodja

Za izboljšanje navigacije v zaprtih prostorih imamo na voljo več razvojnih ogrodij, ki nudijo vnaprej pripravljene rešitve. Nekatera so brezplačna, za uporabo drugih je zahtevano mesečno, letno ali enkratno plačilo. Cene so dostopne na spletnih straneh ponudnikov. V času pisanja dela se pozornost velikih podjetij, npr. Samsung, Nokia, Google itd., vse bolj usmerja ravno v razvoj rešitev za navigacijo v notranjih prostorih.

V nadaljevanju bomo opisali nekaj razvojnih ogrodij in jih med seboj primerjali. Na osnovi rezultatov bomo izbrali za naše potrebe najprimernejše razvojno ogrodje in ga uporabili pri izdelavi prototipa mobilne aplikacije za navigacijo v zaprtem prostoru.

2.2.1 MapsIndoors SDK

Razvojno ogrodje MapsIndoors SDK temelji na storitvah Google Maps, s pomočjo katerih nudi na zemljevidu enostavne prehode med zunanjim in notranjim okoljem. Ogrodje nudi navigacijo s pomočjo:

- oddajnikov signalov Bluetooth,
- WiFi pozicioniranja,
- svetlobe,
- geomagnetnih zapisov.

MapsIndoor SDK je razvilo podjetje MapsPeople, ki prodaja tudi licence Google Maps Premium Plan. To je zbirka razvojnih ogrodij, s pomočjo katerih lahko v prilagojenih zemljevidih Google Maps prikažemo lastne tlorise. Za prikaz uporabniške lokacije se uporablja razvojno ogrodje Google Maps SDK. Ker so zemljevidi MapsIndoors razviti z uporabo storitev Google Maps, omogočajo enostavne prehode med zunanjim in notranjim okoljem. Za prikaz zunanjega okolja se uporabljajo zemljevidi storitve Google Maps, za prikaz notranjega okolja pa zemljevidi storitve MapsIndoors. Razvojno ogrodje MapsIndoor SDK ponuja tudi spletne storitve, ki preko aplikacijskega programskega vmesnika (API) nudijo odgovore na različne poizvedbe o tipih prostorov in interesnih točkah. Tako na primer

ogrodje v primeru parkirišč interesne točke prikazuje kot parkirna mesta ali v primeru bolnišnic kot sobe. Z razvojnim ogrodjem MapsPeople SDK lahko znotraj storitev Google Maps prikazujemo tudi tlorise prostorov. Za razliko od ostalih razvojnih ogrodij za navigacijo v zaprtih prostorih MapsIndoor SDK ne potrebuje predhodnih sprehodov po prostoru, temveč se interesne točke definirajo preko spletne storitve MapsPeople. Ponujena je tudi analitična programska oprema Indoor Location Analytics, s katero lahko organizatorji dogodkov pregledujejo interese obiskovalcev [7]. Lastniki prostorov lahko tlorise oddajo preko spletne aplikacije, oddani tlorisi pa so nato na voljo uporabnikom v storitvah Google Maps in posledično tudi v razvojnem ogrodju MapsIndoors SDK [19].

- *Prednosti MapsIndoor SDK*
    - partnerstvo z Googlom,
    - prikaz okolja s pomočjo Google Maps zemljevidov,
    - enostavni prehodi med zunanjim okoljem in zaprtim prostorom.
- *Slabosti MapsIndoor SDK*
    - slabša natančnost brez uporabe dodatne strojne opreme,
    - manj primerno za neodvisne razvijalce, saj je za prikaz lastnih tlorisov na zemljevidih Google Maps treba kupiti licenco Google Maps Premium Plan.

2.2.2 IndoorAtlas SDK

Razvojno ogrodje IndoorAtlas SDK poleg zunanjih knjižnic in aplikacijskega programskega vmesnika nudi tudi mobilno aplikacijo MapCreator 2 in spletni vmesnik. S pomočjo mobilne aplikacije lahko opredelimo in kalibriramo opise notranjih prostorov, s spletnim vmesnikom pa upravljamo naše aplikacije in tlorise. IndoorAtlas SDK omogoča lociranje z natančnostjo od 2 do 0,2 metrov, navigacija pa je možna s pomočjo [10]:

- oddajnikov signalov Bluetooth (dodano v zadnji verziji knjižnic SDK 2.3),
- WiFi pozicioniranja,
- geomagnetnih zapisov,
- akustičnih signalov.

- *Prednosti IndoorAtlas SDK*
    - odlična natančnost samo z branjem geomagnetnih polj (od 2 do 0,2 metrov),
    - opcijska uporaba dodatne strojne opreme, ki zagotavlja boljšo natančnost,
    - mobilna aplikacija za branje podatkov o okolju,
    - primerno tudi za neodvisne razvijalce, saj je ogrodje na voljo brezplačno, dokler aplikacije ne uporablja vsaj 100 uporabnikov (poslovni model 'plačaj, ko zrasteš' (angl. 'Pay as you grow')).
- *Slabosti IndoorAtlas SDK*
    - Manj natančni (slabši) zemljevidi zunanjega okolja v primerjavi z zemljevidi Google Maps. To lahko povzroča težave pri izbiri objektov za dodajanje tlorisov v oblačne storitve IndoorAtlas. Če območje, ki ga dodajamo, nima ustrezno posodobljenih zemljevidov, lahko to vpliva tudi na prehode med zunanjim in notranjim okoljem.

2.2.3 Indoo.rs SDK

Razvojno ogrodje Indoo.rs SDK ponuja navigacijo v zaprtih prostorih s pomočjo:

- izračuna triangulacije med oddajniki signalov Bluetooth,
- tehnologije SLAM[1].

Tehnologija SLAM beleži podatke o prostoru s pomočjo sprehoda. Pred začetkom zbiranja podatkov v aplikacijo SLAM vnesemo interesne točke, med katerimi se bomo sprehajali, nato pa se med sprehodom po prostoru v njej zbirajo podatki o okolju [17].

- *Prednosti Indoo.rs SDK*
    - mobilna aplikacija za branje podatkov o okolju (tehnologija SLAM),
    - vgrajena podpora za sledenje inventarju in zaposlenih podjetja preko spletnih storitev.

[1] Hkratna lokalizacija in kartiranje (angl. simultaneous localization and mapping)

- *Slabosti Indoo.rs SDK*
    - ne podpira navigacije z branjem geomagnetnega polja,
    - nujna uporaba dodatne strojne opreme,
    - za uporabo ogrodja je treba pridobiti dovoljenje s strani razvijalca ogrodja.

2.2.4 Primerjava različnih razvojnih ogrodij

Za končno primerjavo razvojnih ogrodij smo se osredotočili na kriterije, ki so pomembni tako za njihovo delovanje kot za izdelavo prototipa. Z vidika delovanja nas je zanimala natančnost določanja lokacije, doseganje natančnosti z ali brez dodatne strojne opreme in cena naročnine na storitve. Zaželena je bila visoka natančnost, neobveznost dodatne opreme in brezplačna uporaba storitev. Za izdelavo prototipa smo tudi preverili, za katere operacijske sisteme so ponujeni API ter zahtevnost njihove implementacije[2]. Iskali smo ogrodje z najlažjo implementacijo in podporo za operacijski sistem Android (kar je ciljni OS našega prototipa). Posamezna ogrodja med seboj primerjamo tudi glede na njihove ključne prednosti in slabosti.

API	Natančnost lokacije (v metrih)	Zahtevnost implementacije[1]	Dodatna strojna oprema	Operacijski sistemi	Cena	Ključna prednost (+) / slabost (-)
Maps Indoors	2,5 – 2	srednje	ni obvezna	Android in iOS	sklenitev partnerstva z naročnino	+: temelji na Google Maps -: slabša natančnost brez dodatne strojne opreme
Indoor Atlas	2 – 0,2	enostavna	ni obvezna	Android, Cordova in iOS	brezplačno do 100 uporabnikov	+: odlična natančnost -: slabši zemljevidi

[2] Zahtevnost implementacije je bila ocenjena glede na dokumentacijo in enostavnost implementacije aplikacijskih programskih vmesnikov razvojnega ogrodja za mobilni operacijski sistem Android.

Indoo.rs	5 - 2	srednje	oddajniki signala Bluetooth	Android in iOS	naročnina in cca. 20 € za posamezen oddajnik signala Bluetooth	+: tehnologija SLAM -: nujna uporaba dodatne strojne opreme

Tabela 1: Primerjava različnih SDK za določanje lokacije v zaprtem prostoru[3]

Na podlagi primerjave različnih ogrodij lahko ugotovimo, da za izdelavo prototipa najbolj ustreza ogrodje IndoorAtlas. Implementacija ogrodja je enostavnejša kot pri ostalih ogrodjih in dobro dokumentirana. Interesne točke lahko locira do 20 centimetrov natančno, kar ob uporabi v trgovini pomeni, da lahko določimo, ne le lokacijo posameznega prodajnega regala, ampak tudi izdelkov na njem. Uporaba ogrodja za potrebe izdelave prototipa bo brezplačna, saj število uporabnikov ne bo preseglo omejitve za brezplačno uporabo, nakup dodatne strojne opreme pa ni potreben. Lokacija se bo določala na podlagi geomagnetnega polja prostora. Izboljšanje natančnosti in točnosti določanja lokacije pa lahko brez sprememb prototipa dosežemo z dodatno uporabo oddajnikov elektromagnetnih valov.

[3] Vrednosti so vzete s spletnih strani ponudnikov razvojnih ogrodij, navedenih v Seznamu literature.

3 Uporabljena orodja in tehnologije

Ugotovili smo že, da sistem GPS ni primeren za navigiranje v zaprtih prostorih. Ko signali sistema GPS potujejo skozi razne materiale v zaprte prostore postanejo namreč prešibki, da bi jih lahko sprejemnik signala GPS uporabil za pravilen izračun lokacije [1]. Orodja za navigacijo, ki smo jih uporabili za izdelavo prototipa in jih predstavljamo v tem poglavju, pa omogočajo ravno to.

Programski jezik, podatkovno bazo, ostala orodja in tehnologije, uporabljene za razvoj prototipa, opisujemo na koncu tega poglavja. Orodja, ponujena kot podporni del razvojnega ogrodja IndoorAtlas SDK, pa so:

- razvojne knjižnice in aplikacijski programski vmesniki za ustvarjanje aplikacije,
- mobilna aplikacija MapCreator 2 za kalibracijo podatkov o prostorih,
- oblačne storitve IndoorAtlas za obdelavo in hrambo podatkov o prostorih[4].

3.1 Orodja ogrodja Indoor Atlas SDK

Z razvojnim ogrodjem IndoorAtlas SDK lahko nudimo navigacijo v zaprtih prostorih s pomočjo:

- elektromagnetnih valov (WiFi, radijski valovi, Bluetooth),
- geomagnetnih zapisov,
- akustičnih signalov.

Pri razvoju prototipa smo najprej uporabili različico IndoorAtlas SDK 2.1, ki je nudila navigacijo s pomočjo WiFi, radijskih valov in geomagnetnih zapisov. Kasneje smo prešli na

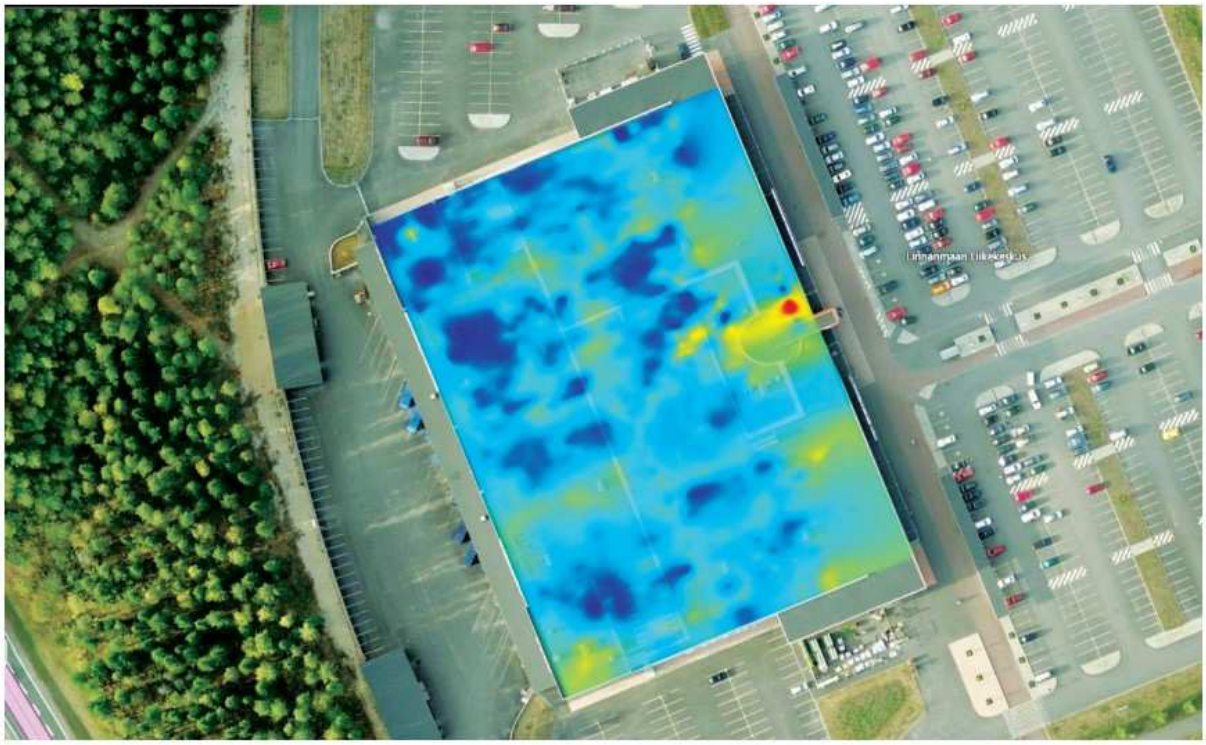
[4] Oblačne storitve IndoorAtlas omogočajo dostop do strojne in programske opreme, potrebne za izdelavo rešitve za navigacijo v zaprtih prostorih. Tovrstni model računalništva v oblaku imenujemo platforma kot storitev (PaaS, angl. Platform as a Service).

različico IndoorAtlas SDK 2.3, ki navedenemu doda še možnost navigacije s pomočjo oddajnikov signala Bluetooth in akustičnih signalov.

Kot smo ugotovili že v 2. poglavju, je pomembna prednost ogrodja IndoorAtlas SDK tudi to, da ne zahteva dodatne strojne opreme in je kompatibilno z mobilnima operacijskima sistemoma Android in iOS.

Navigiranje z IndoorAtlas SDK

Navigacija z ogrodjem Indoor Atlas SDK temelji na geomagnetnih valovih. Primer na Sliki 2 kaže, da lahko iz barvnih odtenkov, ki prikazujejo popačenja geomagnetnega zapisa, razberemo različne variacije geomagnetnega polja. Prikazan je tudi primer geomagnetnega zapisa nekega prostora, prebranega z uporabo razvojnega ogrodja IndoorAtlas SDK. S slike je razvidno prelivanje barv iz hladnih v tople, kar ponazarja popačenost zemeljskega geomagnetnega polja.



Slika 2: Geomagnetni zapis notranjega prostora (Vir: IndoorAtlas, 2014)

Z vidika navigacije je pomembno, da ima vsak zaprt prostor sebi lasten geomagnetni zapis. Poznavalci navigacije s pomočjo magnetnih polj sicer opozarjajo, da snovi, kot sta železo ali beton, magnetna polja popačijo in s tem zmanjšajo natančnost in točnost navigacije [25]. Ponudnik storitev IndoorAtlas pa ravno popačenost magnetnega polja omenja kot prednost

tega ogrodja. Zaradi dejstva, da se popačenost magnetnega polja s časom ne spreminja, je namreč navigacija s pomočjo geomagnetnega zapisa zelo natančna in točna. V 2. poglavju smo že omenili navedbo ponudnika storitev IndoorAtlas, da lahko z omenjeno storitvijo lokacije določimo od 2 do 0,2 m natančno.

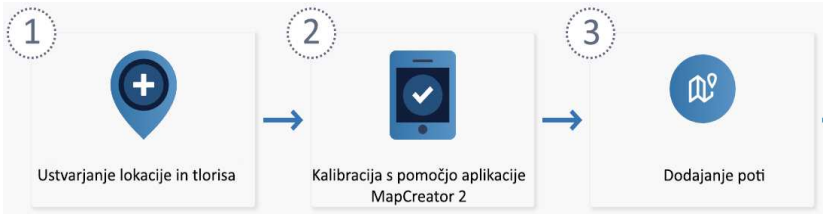
Izdelava tlorisov prostorov s pomočjo MapCreator 2

Osnova navigacije v zaprtih prostorih je tloris posameznega prostora. Da lahko torej uporabniku ponudimo navigacijo v zaprtem prostoru, moramo najprej izdelati tloris in ga posredovati v oblačne storitve IndoorAtlas.

V oblačnih storitvah IndoorAtlas izdelanemu tlorisu prostora določimo lokacijo in pregledujemo nabor informacij o tlorisu, ki so bile nabrane s pomočjo mobilne aplikacije MapCreator 2. Omenjena aplikacija zbira podatke tako, da se z napravo, na kateri je aktivirana, sprehodimo po prostoru. Ko končamo, zbrane podatke enostavno prenesemo v oblačne storitve IndoorAtlas. Na podlagi prejetih podatkov oblačne storitve IndoorAtlas posodobijo tloris prostora, ki je nato na voljo preko aplikacijskega programskega vmesnika razvojnega ogrodja IndoorAtlas SDK.

Ko so tlorisi izdelani, so takoj na voljo za nadaljnjo uporabo. Vsak tloris, ki ga ustvarimo, je dostopen preko njegove identifikacijske številke. Zavarovan je tudi z aplikacijskim ključem, ki hkrati omogoča identifikacijo med oblačnimi storitvami IndoorAtlas in mobilno aplikacijo. Priporočljivo je, da ima vsaka aplikacija lastni aplikacijski ključ, saj to zagotavlja pravilno distribucijo podatkov in dodatno varnost skrbniškega računa. Distribucija podatkov je še posebej pomembna, kadar imamo več aplikacij in želimo prikazovati pravilne tlorise.

Poudarimo naj, da uporaba aplikacije MapCreator 2 ni nujna, pripomore pa k temu, da je končni izdelek (npr. naš prototip) uporabnikom prijaznejši. S predhodno kalibracijo opisov notranjih prostorov z omenjeno aplikacijo namreč že prvemu uporabniku našega prototipa ponudimo izračune poti do njegovih interesnih točk. Brez uporabe te aplikacije bi moral uporabnik za uspešno generiranje tlorisa večkrat prehoditi prostor, da bi oblačne storitve prejele zadostno količino podatkov. Slika 3 prikazuje proces ustvarjanja tlorisa s pomočjo IndoorAtlas SDK.



Slika 3: Proces izdelave tlorisa v IndoorAtlas SDK

Tloris zaprtega prostora v IndoorAtlas izdelamo po naslednjih korakih:

1. določitev osnovnih dimenzij prostora,

2. sprehod skozi prostor z napravo, na kateri je nameščena in aktivirana aplikacija MapCreator 2, ki podatke posreduje v oblačne storitve IndoorAtlas,

3. bogatenje nabora znanih vrednosti tlorisa na izbrani lokaciji v obliki različnih poti in interesnih točk.

Z izdelanimi tlorisi in kalibriranimi podatki o prostoru začnemo ugotavljati lokacijo uporabnika, ki jo na tlorisih tudi prikazujemo. Ko to združimo z izračunano uporabnikovo lego, lahko na tlorisu predstavimo še lokacijo interesne točke in pot do nje.

Kako izdelamo tloris zaprtega prostora v IndoorAtlas SDK

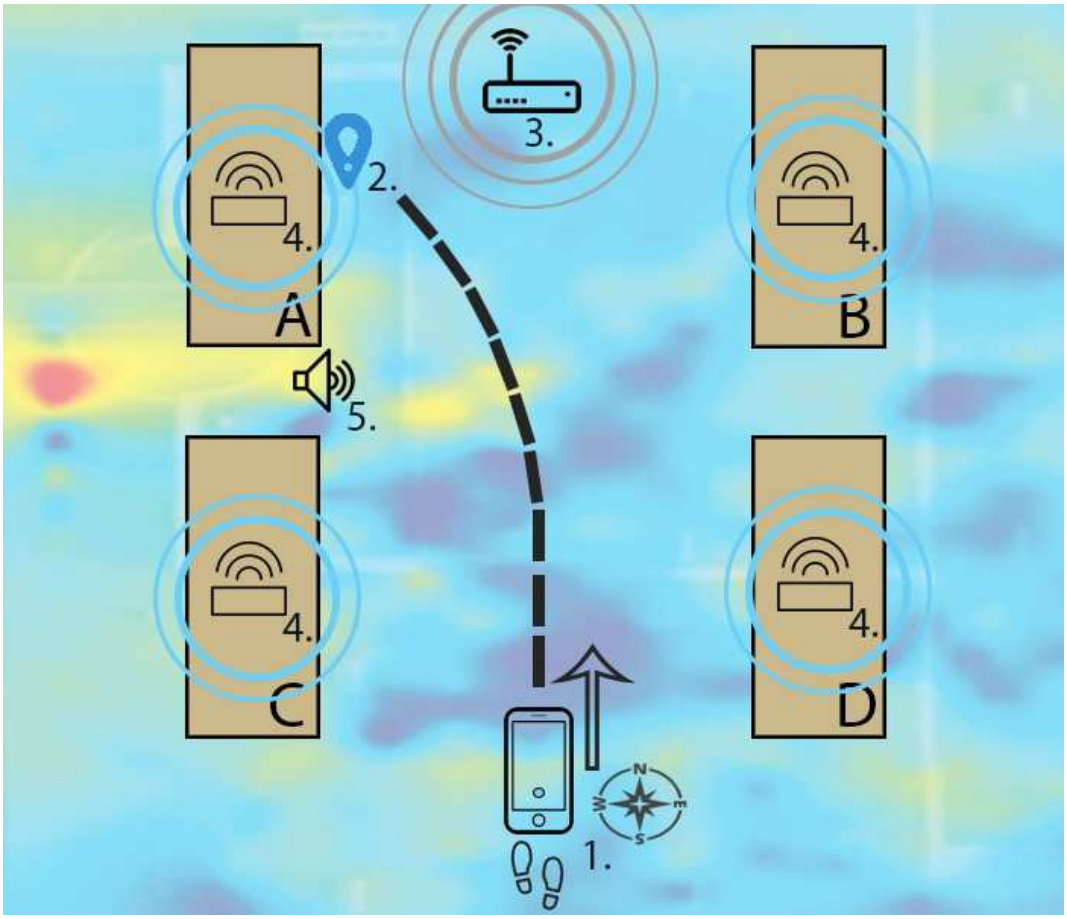
Po sprehodih z mobilno aplikacijo MapCreator 2 se v oblačnih storitvah IndoorAtlas naši tlorisi posodobijo s podatki kalibriranih prostorov. Posodobljeni tlorisi so nam znotraj oblačnih storitev IndoorAtlas na voljo v raznih prikazih, kot je npr. prikaz točno določene poti skozi prostor (slika 4) ali prikaz jakosti geomagnetnih polj (slika 5). Omenjeni prikazi omogočajo tudi določanje poti, namenjenih prehodom, s čimer preprečimo usmerjanje uporabnika ali prikazovanje njegove lokacije na neuporabnih poteh skozi prostor (npr. skozi steno).

Slika 4: Določanje poti na FRI

Slika 5: Jakost magnetnih signalov na FRI, kjer je z rumeno označena zadovoljiva, z zeleno pa odlična jakost

Na ta način oblačnim storitvam IndoorAtlas posredujemo dovolj podatkov o prostoru, da uporabniku omogočimo navigacijo v zaprtem prostoru. Slika 6 dopolnjuje Sliko 1 s prikazom vseh podatkov o okolju, ki jih lahko zberemo s pomočjo podpornih orodij razvojnega ogrodja IndoorAtlas SDK. Slika prikazuje:

- mobilno napravo uporabnika z uporabljenimi mobilnimi senzorji (točka 1),
- interesno točko (točka 2),
- oddajnik signala WiFi (točka 3),
- oddajnike signala Bluetooth (točka 4),
- zvočni oddajnik (točka 5),
- razne neprehodne objekte (znaki od A do D),
- geomagnetni zapis prostora.



Slika 6: Mobilna naprava in interesna točka v prostoru z znanimi podatki o okolju

S pomočjo tako zbranih podatkov o okolju lahko uporabnika zanesljivo usmerjamo do njegove interesne točke. V primeru na Sliki 6 poznamo o prostoru med uporabnikom (točka 1) in njegovo interesno točko (točka 2) naslednje podatke:

- odtis geomagnetnega polja oz. geomagnetni zapis prostora,
- oddajnik signala Bluetooth na oviri A,
- smer in oddaljenost do oddajnika signala WiFi (točka 3),
- zvočni zapis, ki ga oddaja zvočnik (točka 5),
- lego mobilne naprave oz. premikanja uporabnika, kar je izračunano z uporabo senzorjev pospeškomera in giroskopa (podrobneje opisano v 4. poglavju).

Za izračun lokacije uporabnika v zaprtem prostoru je nujen samo geomagnetni zapis. Vse ostalo le pripomore k boljši natančnosti in točnosti lokacije. Za usmerjanje uporabnika do njegove točke pa moramo poleg že znanih lokacij upoštevati še lego mobilne naprave.

3.2 Ostala orodja in tehnologije

Za izdelavo prototipa, predstavljenega v 4. poglavju in razvitega za mobilni operacijski sistem Android, smo uporabili tudi v nadaljevanju navedene tehnologije in razvojna okolja.

Java

Prototip mobilne aplikacije je napisan v programskem jeziku Java. Zaradi navideznega stroja Java Virtual Machine je neodvisen od sistema. Java Virtual Machine se na sistemu, na katerega je nameščen, obnaša kot samostojen sistem, na katerem se nato izvaja koda programskega jezika Java [2].

SQLite

SQLite je relacijska podatkovna baza za lokalno shranjevanje podatkov. Za razliko od večine podatkovnih baz SQL ne izvaja ločenih strežniških procesov. SQLite hrani in obdeluje podatke v navadnih datotekah na disku naprave [21].

Android Studio

Uradno razvojno okolje za razvoj android aplikacij je Android Studio. Ponuja tekstovne in grafične urejevalnike za pisanje programske kode in oblikovanje zaslonskih mask [5].

Android Emulator

Android Emulator je virtualna mobilna naprava z mobilnim operacijskim sistemom Android, ki je del osnovnega razvojnega ogrodja Android SDK. Konfiguriramo ga lahko po lastnih željah in mu na primer določimo starejšo ali novejšo različico sistema Android ali pa določamo izmišljeno lokacijo [22].

Gimp

Gimp je brezplačni in odprtokodni grafični urejevalnik [6].

4 Prototip mobilne aplikacije

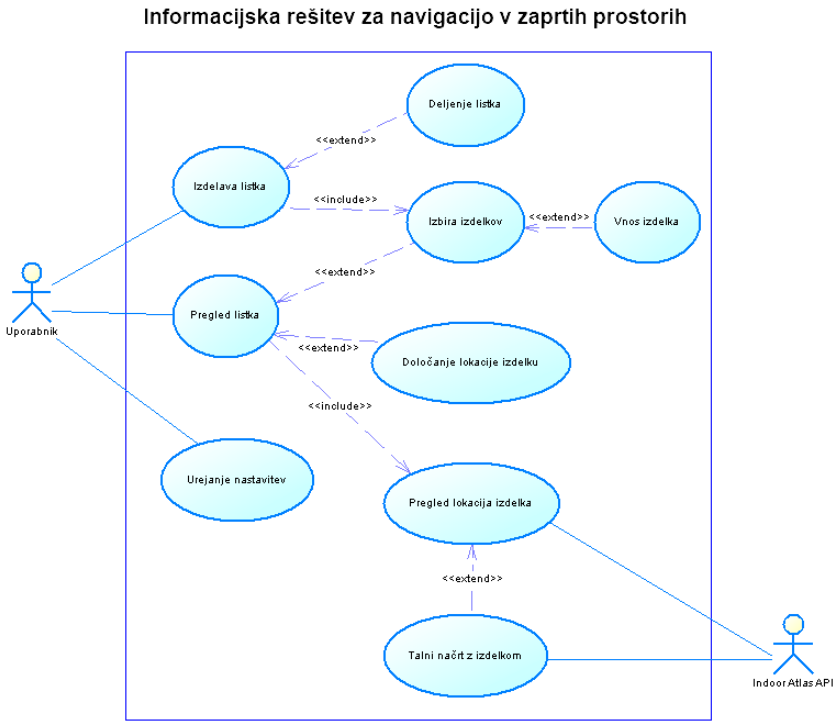
Danes imamo na voljo vrsto pripomočkov, orodij in tehnologij, ki nam pomagajo pri vsakodnevnih opravilih. Če se odpravljamo na pot, si pomagamo s sistemom GPS. Če nas zanima, v kateri trgovini ima določeni artikel najnižjo ceno, si pomagamo s spletom. Ko se odločimo, v kateri trgovini ga bomo kupili, nas lahko do nje vodi mobilna naprava s pomočjo sistema GPS. Takšna pomoč sodobne tehnologije je možna, dokler je izbrana lokacija na prostem – v zunanjem prostoru. Zaenkrat pa še ni ponudnika univerzalnega orodja, ki bi ponujal navigacijo v zaprtih prostorih, kot to za zunanje okolje ponujajo npr. Google Maps [24]. Zato smo se odločili za razvoj prototipa mobilne aplikacije, ki nudi navigacijo v različnih zaprtih prostorih. Takšno mobilno aplikacijo bi lahko, na primer, uporabili v trgovini za iskanje artiklov z našega nakupovalnega listka ali pa bi nam bila v pomoč pri iskanju točno določene sobe v bolnišnici.

4.1 Načrt aplikacije

Pred začetkom razvoja prototipa smo določili njegove funkcionalnosti. Določili smo, da mora uporabniku omogočati izbiro interesnih točk, do katerih ga tudi vodi. Za uporabniško izkušnjo je pomembno, da sprememba tipa interesnih točk ali razširitev uporabnosti aplikacije ne vpliva na izgled zaslonskih mask. Interesna točka dobi s tem abstrakten pomen, na zaslonskih maskah pa je treba narediti le manjše spremembe poimenovanj. Razviti prototip je omejen na izbiro interesnih točk (artikli na prodajnih regalih) v trgovinah, zato smo ga poimenovali 'Nakupovalni vodič' (ang. 'ShopAssistant').

Prototip omogoča ustvarjanje nakupovalnega listka s pomočjo vnosne zaslonske maske. Uporabnik lahko artikle uporabnik izbere s spustnega seznama ali pa preko prostega vnosa, če posamezni artikel še ne obstaja v podatkovni bazi prototipa. Ob zagonu aplikacije so uporabniku na voljo vsi nakupovalni listki, ki jih je že ustvaril in med katerimi lahko izbira. Ko izbere obstoječi nakupovalni listek, se uporabniku prikažejo informacije o izbranih artiklih. Prikaže se tudi kažipot do najbližjega artikla, pregled tlorisa s pozicijo uporabnika in izbranega artikla ter gumb za urejanje nakupovalnega listka. Ker se lahko en artikel prodaja v več trgovinah, je v nastavitvah prototipa možno spreminjati tudi trenutni prostor. Uporabniku se prikazani artikli razvrščajo glede na njihovo oddaljenost (od najbližjega do najbolj

oddaljenega artikla). Če informacija o lokaciji artikla ni na voljo, mu je lokacija določena, ko ga uporabnik izbere na nakupovalnem listku.



Slika 7: Diagram primerov uporabe

Slika 7 prikazuje primere uporabe prototipa, pri kateri lahko uporabnik ustvarja ali ureja nakupovalne listke preko aktivnosti *izdelava listka* in *pregled listka*. Preko aktivnosti *urejanje nastavitev* uporabnik ustvarja ali spreminja prostore in njihove tlorise. Kot že omenjeno, lahko v slednji izdelkom določa tudi lokacijo. Programska logika in zaslonske maske prototipa so podrobneje opisane v nadaljevanju.

Tipični način uporabe sistema za primer, prikazan na Sliki 7, je lahko naslednji:

1. uporabnik zažene aplikacijo,
2. uporabnik izbere akcijo izdelave listka,
3. uporabnik izbira izdelke,
4. uporabnik shrani listek pod želenim nazivom,
5. uporabnik izbere aktivnost *pregled listka*,

6. aplikacija uporabniku prikaže izdelke na listku,
7. aplikacija razvršča izdelke od najbližjega do najbolj oddaljenega,
8. aplikacija prikazuje smer do najbližjega izdelka,
9. uporabnik najde vse izdelke,
10. uporabnik ugasne aplikacijo.

Alternativni način uporabe sistema pa bi lahko bil naslednji:

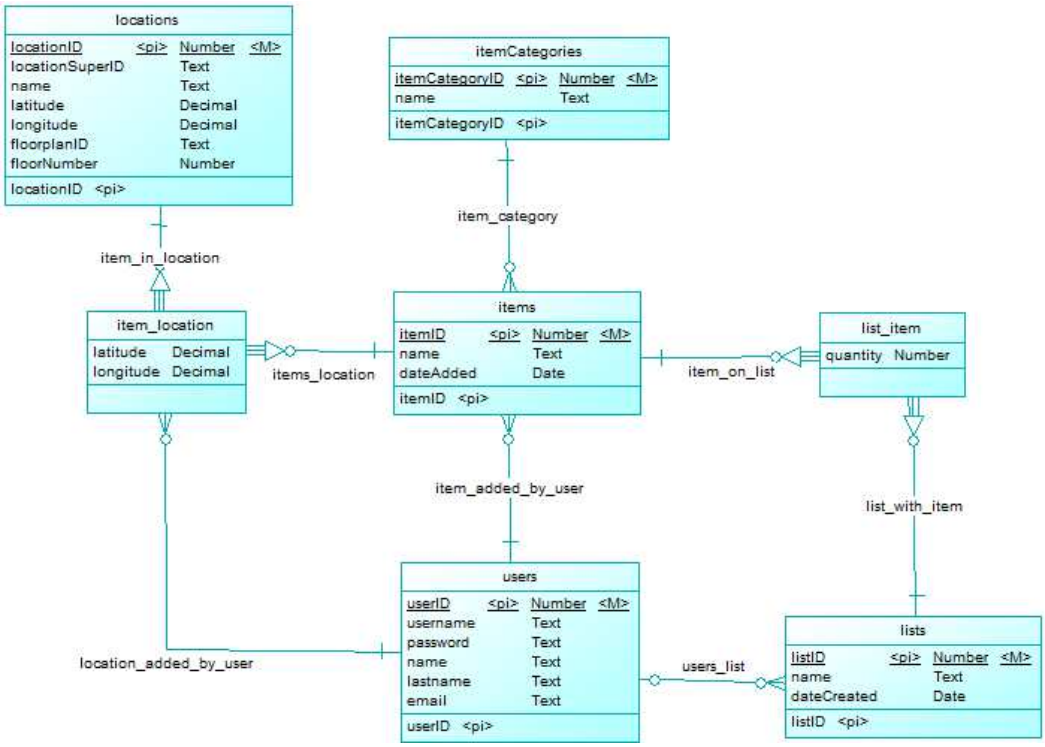
1. uporabnik zažene aplikacijo,
2. uporabnik izbere akcijo izdelave listka,
3. uporabnik doda izdelke, ki jih podatkovna baza ne vsebuje,
4. uporabnik shrani listek pod želenim nazivom,
5. uporabnik izbere aktivnost *pregled listka*,
6. aplikacija uporabniku prikaže izdelke na listku in označi tiste, za katere ne pozna lokacije,
7. aplikacija izdelke, za katere pozna lokacijo, razvršča od najbližjega do najbolj oddaljenega,
8. aplikacija prikazuje smer do najbližjega izdelka,
9. uporabnik najde izdelke, ob izbiri izdelka z neznano lokacijo pa se mu zabeleži nova lokacija,
10. uporabnik najde vse izdelke,
11. uporabnik ugasne aplikacijo.

Pri načrtovanju aplikacije je pomembno, da izberemo tudi vrsto podatkovne baze in ji ustvarimo pravilno strukturo. V nadaljevanju bomo predstavili podatkovno bazo SQLite, ki jo za hranjenje podatkov uporablja naš prototip, in njen konceptualni model.

Podatkovna baza

Podatki našega prototipa se shranjujejo v podatkovno bazo SQLite. SQLite je odprtokodna relacijska podatkovna baza, v celoti shranjena znotraj tekstovne datoteke, ki jo nudi vsaka naprava z operacijskim sistemom Android [13]. Uporaba te podatkovne baze zadošča za naš namen, saj potrebujemo podatke le za enega uporabnika, lahko so vnaprej pripravljeni in s tem na voljo že ob namestitvi aplikacije. Če bi bilo treba sinhronizirati uporabnike, artikle in lokacije med več napravami, bi morali uporabiti centralni SQL strežnik, kjer bi se shranjevali in delili podatki.

Struktura podatkovne baze je predstavljena v relacijskem modelu na Sliki 8.



Slika 8: Konceptualni model podatkovne baze aplikacije ShopAssistant

Posamezni uporabnik (entitetni tip “user”) lahko ustvari poljubno število listkov, jim doda poljubno število artiklov ter določi njihovo lokacijo, če še ni določena. Posamezni nakupovalni listek (entitetni tip “lists”) lahko pripada le enemu uporabniku, vsebuje pa lahko poljubno število artiklov, za katere hrani tudi količinsko enoto. Posamezen artikel (entitetni tip “item”) je lahko dodan le enkrat, lahko pa se mu določi poljubno število lokacij znotraj različnih prostorov. Pri dodajanju artikla je zabeležen tudi uporabnik, ki ga je dodal. Posamezni artikel lahko pripada le eni kategoriji artiklov. V posameznem prostoru (entitetni tip “location”) je lahko več artiklov. Za hranjenje relacij lokacija - artikel se uporablja šibek

entitetni tip “item_location”, kjer sta shranjena še uporabnik, ki je artiklu določil lokacijo, in zemljepisna širina in dolžina. Za hranjenje relacij artikel - list pa se uporablja šibek entitetni tip “list_item”, kjer je shranjena še količina artikla na listku.

Na podlagi takšnega načrta aplikacije so bile razvite aktivnosti prototipa, ki jih predstavljamo v nadaljevanju.

4.2 Aktivnosti prototipa

Razred aktivnosti v mobilnem operacijskem sistemu Android se imenuje 'Activity' in skrbi za ustvarjanje ter nadzor zaslonskih mask. Vsak razred, ki deduje razred 'Activity', hrani referenco na lastno zaslonsko masko, zapisano v razširljivem označevalnem jeziku XML (angl. Extensible Markup Language), in programsko logiko za uporabniške vhodne in izhodne podatke [12].

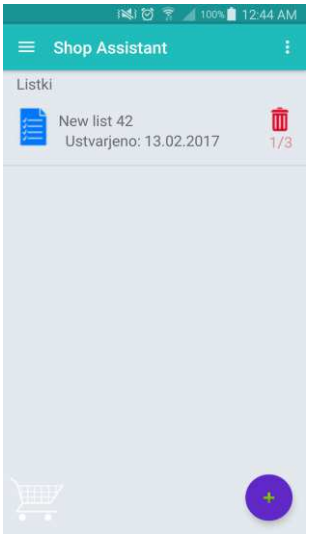
Uporabnik prototipa ima na voljo štiri zaslonske maske:

- *glavno okno aplikacije*, kjer lahko dodaja ali pregleduje obstoječe listke in dostopa do nastavitev aplikacije,
- *nastavitve*, kjer lahko določa prostor in tloris iz oblačnih storitev Indoor Atlas,
- *izdelava listka*, kjer lahko dodaja in ureja količinske ter cenovne informacije o artiklih,
- *pregled listka*, kjer lahko pregleduje podrobnosti o artiklih, smer do najbližjega artikla, lokacijo artikla na tlorisu, deli listek preko ostalih aplikacij na napravi in dostopa do zaslonske maske za ustvarjanje in spreminjanje listka.

4.2.1 Glavno okno aplikacije

Na glavnem oknu aplikacije lahko uporabnik pregleduje, dodaja ter odstranjuje listke in dostopa do nastavitev aplikacije.

Slika 9 prikazuje zaslonsko masko glavnega okna.



Slika 9: Zaslonska maska glavne aktivnosti

Glavno okno nudi dostop do nastavitev preko drsnega polja, ki je objekt razreda DrawerLayout. Drsno polje se prikaže z drsljajem od levega dela zaslonske maske proti desnemu ali s pritiskom na gumb na levi strani orodne vrstice. Glavni del zaslonske maske sestavljata seznam z listki in gumb za dodajanje listkov. Seznam je objekt razreda ListView s spremenjenim gradnikom razreda ListHolderAdapter, ki deduje razred BaseAdapter. Sprememba gradnika je potrebna zaradi prilagoditve posameznih elementov seznama. V našem primeru posamezni element sestavlja ikona listka, ime ter datum nastanka listka, gumb za izbris listka in informacije o poteku izbire izdelkov na listku. Za to smo uporabili naslednje poglede razvojnega ogrodja Android SDK (angl. View):

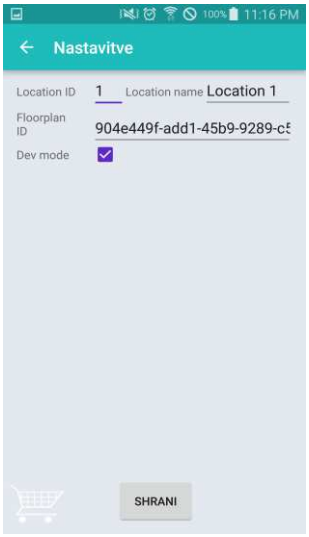
- sliko (angl. ImageView) za ikono listkov,
- tekstovno polje (angl. TextView) za razne opise listka,
- gumb (angl. Button) za izbris listka s sliko koša za smeti.

V spodnjem desnem delu zaslonske maske se nahaja plavajoči gumb (angl. FloatingActionButton), ki ob kliku sproži prehod v aktivnost ustvarjanja listka. Plavajoči gumb se imenuje zato, ker lahko “lebdi” nad ostalimi gradniki.

4.2.2 Nastavitve

V nastavitvah lahko uporabnik ureja prostor in tloris, možen pa je tudi vklop načina za razvijalce, s katerim med uporabo aplikacije prikazujemo dodatne informacije o lokaciji

uporabnika in izdelkov. Nastavitve se beležijo v statičnem razredu AppContext, ki je bil ustvarjen ravno za ta namen. Slika 10 prikazuje zaslonsko masko nastavitev.

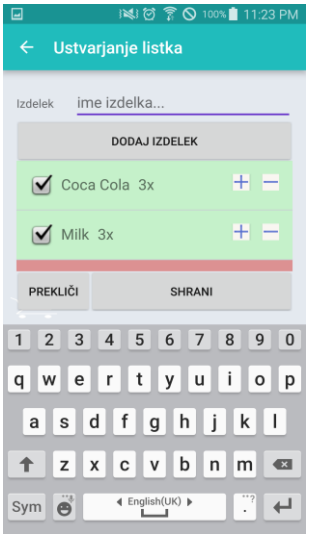


Slika 10: Zaslonska maska aktivnosti nastavitev

Zaslonska maska aktivnosti *Nastavitve* je sestavljena iz tekstovnih in izbirnih (angl. CheckBox) polj. Spremenjena vsebina na zaslonski maski nastavitev se ob pritisku na gumb 'Shrani' shrani v podatkovno bazo SQLite.

4.2.3 Izdelava listka

Pri ustvarjanju listka uporabnik vnaša izdelke preko tekstovnega polja na vrhu zaslonske maske z opisom 'Izdelek'. Ob vnosu niza se uporabniku izpiše spustni seznam (angl. Dropdown list) z že obstoječimi artikli, ki ustrezajo trenutno vnesenemu iskalnemu nizu. Artikel, ki v podatkovni bazi ne obstaja, se zapiše v podatkovno bazo po ustvarjanju listka. Če je izbran artikel s prikazanega spustnega seznama pod tekstovnim poljem "Izdelek', je samodejno dodan na seznam artiklov, sicer pa mora uporabnik dodajanje potrditi s pritiskom na gumb 'Dodaj izdelek'. V seznamu artiklov sta poleg ikone in imena artikla na voljo še gumba za upravljanje s količino (znak "+" in "-"). Artikel je izbrisan s seznama, ko je količina enaka 0. Na dnu zaslonske maske sta gumba 'Prekliči' in 'Shrani', kjer velja, da s pritiskom prvega zavržemo spremembe, s pritiskom drugega pa jih shranimo. Slika 11 prikazuje ustvarjanje listka.



Slika 11: Zaslonska maska aktivnosti ustvarjanja listka

Prehod na aktivnost ustvarjanja listka je možen preko glavnega okna in preko pregleda izdelkov. V prvem primeru uporabnik ustvari nov listek, v drugem pa popravi obstoječega. V drugem primeru se na novo dodani artikli obarvajo rdeče, s čimer je uporabniku omogočeno sledenje napredku. Tudi seznam artiklov (angl. ListView) ima - podobno kot seznam listkov - spremenjen gradnik. Gradnik je objekt razreda ItemHolderAdapter, ki deduje razred BaseAdapter. S tem pridobimo prilagodljiv prikaz za ime in količino izdelka ter gumba za povečevanje ali zmanjševanje količine. Ob pritisku na gumb ‘Shrani’ se uporabniku prikaže dialog, v katerega lahko vnese ime listka in ga shrani, ali pa zavrne in izbriše.

Zaslonsko masko za ustvarjanje listka poleg prilagodljivega seznama artiklov sestavljajo tudi:

- tekstovno vnosno polje s samodejnim izpolnjevanjem (angl. Auto Complete Text View), ki ob vnosu niza predlaga artikle, katerih ime se začne s trenutno vnesenim nizom,
- gumba ‘Prekliči’ in ‘Shrani’.

4.2.4 Pregled listka

Zaslonska maska za pregled listka se deli na dva prikaza glede na izbiro pri načinu za razvijalce v nastavitvah prototipa. Če je način za razvijalce aktiven, se uporabniku na vrhu maske dodatno izpišejo podatki o njegovi lokaciji, na dnu maske pod smerjo do najbližjega artikla pa status kalibracije podatkov o prostoru. Sicer pa ima uporabnik na vrhu zaslonske maske izpisano ime listka in pod imenom listka gumb ‘Urejanje’. Gumb ‘Urejanje’ uporabnika preusmeri na zaslonsko masko *Ustvarjanje listka*, kjer lahko ureja izbrani listek.

Pod njim se nahaja seznam izbranih artiklov, ki poleg imena artikla in količine vsebuje tudi gumb za izbiro, s katerim uporabnik označi artikel kot izbran. Slika 12 prikazuje zaslonsko masko za pregled artiklov.

Slika 12: Zaslonska maska za pregled artiklov

Ko je aktiviran način za razvijalce, so na vrhu glavnega dela zaslonske maske prikazane informacije o uporabnikovi lokaciji. Seznam, ki hrani artikle, je tipa ListView s spremenjenim gradnikom ItemHolderAdapter (natančneje je opisan v opisu aktivnosti ustvarjanja listka v poglavju 4.2.3). Za urejanje vsebine listka se zažene aktivnost ustvarjanja listkov z dodatnimi parametri zagona. V mobilnem operacijskem sistemu Android za to skrbita metodi *putExtra()* za shranjevanje vrednosti in *getExtra()* za prejem vrednosti. Metodi sta vsebovani v razredu Intent, ki skrbi za prehode med aktivnostmi.

Skupna raba podatkov

Na vrhu zaslonske maske je gumb *dodatne možnosti* (tri navpično razporejene pike), ki nudi možnost skupne rabe vsebine listka preko razreda ShareActionProvider. Razred ShareActionProvider razvojnega ogrodja Android SDK ponuja skupno rabo podatkov med nameščeni aplikacijami v napravi.

Ko imamo izdelan načrt aplikacije in aktivnosti, se lahko posvetimo delovanju aplikacije.

4.3 Delovanje aplikacije

V nadaljevanju bomo podrobneje opisali sestavne dele razvitega prototipa, s katerimi uporabniku ponudimo funkcionalnosti na podlagi zgornjih primerov in načinov. V prototipu se v ozadju izvaja programska koda, ki skrbi za klice oblačnih storitev Indoor Atlas preko njihovega API, poizvedbe in ažuriranja v podatkovni bazi SQLite in preračunavanje uporabnikove smeri gibanja.

Komunikacija z oblačnimi storitvami IndoorAtlas preko njihovega API

Razvojno ogrodje IndoorAtlas SDK nudi API, ki skrbi za programske klice za izmenjavo podatkov z oblačnimi storitvami IndoorAtlas. Prototip ga uporablja za pridobivanje informacij o tlorisu prostora in lokacijah uporabnika ter izdelkov.

Vsaka aktivnost, ki obdeluje podatke o uporabniški lokaciji, mora implementirati vmesnika IALocationListener in IARegion.Listener razvojnega ogrodja IndoorAtlas SDK. Z njima nadomestimo metode, ki vračajo uporabniško lokacijo in njegove prehode med posameznimi regijami v prostoru. Regije prostora so lahko posamezne trgovine znotraj nakupovalnega centra ali pa posamezna nadstropja – odvisno, kako jih določimo ob ustvarjanju tlorisa. Ena izmed pomembnejših metod, ki jo v programski kodi nadomestimo z implementacijo omenjenih vmesnikov, je metoda *onLocationChanged(IALocation iaLocation).* Navajamo primer njene uporabe za osveževanje uporabnikove lokacije:

```
1.  @Override
2.  public void onLocationChanged(IALocation iaLocation) {
3.      tvViewLatValue.setText(iaLocation.getLatitude().toString()); // zemljepisna širi
    na
4.      tvViewLonValue.setText(iaLocation.getLongitude().toString()); // zemljepisna dol
    žina
5.
6.      // Ostale vrednosti shranimo v statični razred
7.      AppContext.setAltitude(iaLocation.getAltitude()); // nadmorska višina
8.      AppContext.setPosAccuracy(iaLocation.getAccuracy()); // natančnost lociranja
9.      AppContext.setPosBearing(iaLocation.getBearing()); // lega
10.
11.     // Posodobimo uporabnikovo lokacijo na sliki tlorisa
12.     if (blueDotView != null && blueDotView.isReady()) {
13.         IALatLng latLng = new IALatLng(iaLocation.getLatitude(), iaLocation.getLongi
    tude()); // koordinate v našem tlorisu
14.         PointF point = iaFloorPlan.coordinateToPoint(latLng); // naša točko v tloris
    u
15.         blueDotView.setDotCenter(point); // premik uporabnikove točke
16.         blueDotView.postInvalidate(); // potrditev sprememb
17.     }
18. }
```

Rezultat klicanja te metode je osveževanje lokacije uporabnika v prostoru, ki je podan kot parameter. S pomočjo metode *IARegion.floorPlan(AppContext.getFloorPlanID())* dobimo

objekt tipa IARegion, ki hrani podatke o tlorisu prostora. Ta objekt – razreda IARegion, pa podamo tudi objektu tipa IALocationManager, ki hrani podatke o lokaciji. To se izvede s pomočjo metode *setLocation()*. Spreminjamo lahko tudi pogostost sinhronizacije oziroma klicev metode *onLocationChanged()* v obliki časovnih ali merskih enot. V prototipu se podatki o lokaciji osvežujejo ali na vsako sekundo ali ob premiku uporabnika za pol metra - odvisno, kaj se prej zgodi. Kadar želimo uporabniku prikazati tloris prostora, v katerem se nahaja, uporabimo razred IAResourceManager. Ta skrbi za pridobivanje objektov tipa IAFloorPlan, ki hranijo informacije o tlorisu prostora, iz oblačnih storitev Indoor Atlas preko njihovega API. Ti objekti vsebujejo tudi slike tlorisov, ki se potem med uporabo aplikacije prikažejo uporabniku.

Z znano lokacijo uporabnika in ostalimi zbranimi informacijami o prostoru, v katerem se uporabnik nahaja, smo že korak bližje navigaciji v zaprtem prostoru. Manjka pa nam še uporabnikova smer.

Določanje uporabnikove smeri

Za izračun uporabnikove smeri v zaprtem prostoru smo z uporabo osnovnih knjižnic Android SDK razvili lastno programsko logiko. Te nudijo vmesnike za branje vseh informacij, povezanih s senzorji mobilne naprave. V našem primeru nas zanimata senzor pospeška in giroskop, s katerima lahko izračunamo kot med severom in usmerjenostjo uporabnika. Uporabili bi lahko tudi senzor magnetnega polja (magnetometer), a smo se zaradi boljše točnosti odločili za senzor pospeška in giroskop [21]. Z dobljeno informacijo lahko uporabniku prikažemo smer do najbližje interesne točke. Preračunavanja uporabnikove smeri so smiselna le, kadar se uporabnik premika, zato smo našo logiko preračunavanja dodali v povoženo (angl. Override) metodo *onSensorChanged(SensorEvent event)*. Metoda *onSensorChanged(SensorEvent event)* je del osnovnega razvojnega ogrodja Android SDK, ki se kliče ob spremenjenih vrednostih senzorjev naprave. V prototipu smo tej metodi dodali tudi logiko zbiranja podatkov o magnetnem polju in preverjanja kvalitete kalibriranja podatkov o okolju. Zaradi lažjega pregleda predstavljamo poenostavljeno metodo *onSensorChanged(SensorEvent event)*, ki je namenjena le preračunavanju uporabnikove smeri:

```
1.  @Override
2.  public void onSensorChanged(SensorEvent event) {
3.      // preverimo tip dogodka
4.      if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
5.          AppContext.setGravityData(event.values);
6.      }
7.      if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
8.          AppContext.setGeomagneticData(event.values);
9.      }
```

```
10.     // preverimo ali imamo dovolj podatkov
11.     if (AppContext.getGravityData() != null &&
12.             AppContext.getGeomagneticData() != null) {
13.         // podatki o lokaciji
14.         double uLat = user.getLatitude(), uLon = user.getLongitude(), iLat = item.ge
    tLatitude(),
15.                 iLon = item.getLongitude(), uAltitude = user.getAltitude();
16.         float[] distances = new float[3];
17.         // razred za hranjenje raznih vrednosti o lokaciji in geomagnetnem polju
18.         GeomagneticField geomagneticField = new GeomagneticField(Double.valueOf(uLat
    ).floatValue(),
19.                 Double.valueOf(uLat).floatValue(), Double.valueOf(uAltitude).floatVa
    lue(),
20.                 System.currentTimeMillis());
21.
22.         float baseBearing = Double.valueOf(user.getBearing()).floatValue(); // smer
    proti severu
23.         float azimut = Double.valueOf(user.getBearing()).floatValue() -
     geomagneticField.getDeclination(); // kot med uporabnikovo smerjo in smerjo proti s
    everu
24.
25.         // metoda distanceBetween vrne tudi smer proti severu
26.         Location.distanceBetween(uLat, uLon, iLat, iLon, distances);
27.         float bearingToItem = distances[2];
28.
29.         // iračunamo smer proti artiklu s pomočjo azimuta
30.         float directionToItem = bearingToItem - azimut;
31.
32.         Log.d(TAG, 'Smer do artikla: ' + directionToItem);
33.         }
34.     }
35. }
```

Uporabnikova smer do interesne točke je odvisna od kota med njim in interesno točko. Enolično določanje smeri na podlagi prostorskih lokacij uporabnika in artiklov bi zadoščalo le, če bi bila naprava vedno usmerjena proti severu. Praksa je seveda drugačna: ljudje se v prostoru prosto premikajo in obračajo, zato moramo izračunati tudi njihov azimut (angl. Azimuth), tj. kot med uporabnikovo smerjo in smerjo proti severu. Uporabnikova smer do interesne točke se nato izračuna tako, da kotu med njima odštejemo njegov azimut.

Navigiranje do posameznih artiklov in prikaz tlorisa

Seznam artiklov se osveži ob vsaki spremembi (dodajanje ali brisanje artikla), artikli pa so razvrščeni glede na oddaljenost od najbolj oddaljenega do najbližjega. Do najbližjega artikla vodi puščica na dnu zaslonske maske (kažipot). Za razvrščanje artiklov glede na njihovo oddaljenost od uporabnika smo napisali metodo *getDistance(User user, Item item)*. Za razvrščanje skrbi t.i. mehurčno urejanje (angl. Bubblesort), ki se izvaja ob vsaki spremembi uporabniške lokacije. Mehurčno urejanje smo izbrali zaradi njegove enostavne implementacije in hitre izvedbe razvrščanja pri majhni količini podatkov [26].

```
1. // razvrščanje s pomočjo mehurčnega urejanja
2. public static ArrayList<Item> sortNearestToFurthest(User user, ArrayList<Item> items
    ) {
```

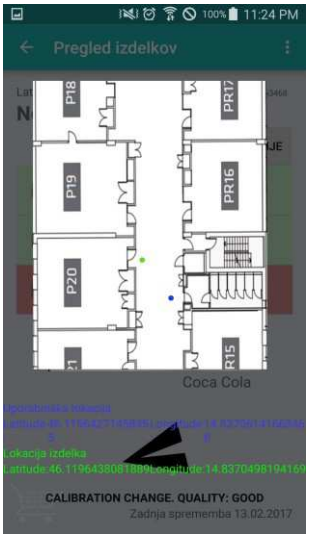
```
3.      boolean flag = true;
4.      while (flag) {
5.          for (int i=0; i<items.size() - 1; i++) {
6.              flag = false;
7.              int j = i;
8.              // primerja se glede na razdaljo med lokacijo uporabnika in artikla
9.              while (getDistance(user, items.get(j)) > getDistance(user, items.get(j+1
    ))) {
10.                 Item temp = items.get(j);
11.                 items.set(j, items.get(j+1));
12.                 items.set(j+1, temp);
13.                 flag = true;
14.             }
15.         }
16.     }
17.     return items;
18. }
```

Za izračun dolžine med točkama se uporablja metoda *distanceBetween()* razreda Location razvojnega ogrodja Android SDK:

```
1.  public static double getDistance(User user, Item item) {
2.      float[] distances = new float[3];
3.      Location.distanceBetween(user.getLatitude(), user.getLongitude(), item.getLatitu
    de(), item.getLongitude(), distances);
4.      // distances[0] = razdalja v metrih !
5.      // distances[1] = začetna lega
6.      // distances[2] = končna lega
7.      Log.d(TAG, 'Razdalja je: ' + (double) distances[0] + ' m');
8.  }
```

Metoda vrne tabelo tipa *float*, ki hrani razdaljo in kot med točkama. Kot med točkama je uporabljen pri prikazovanju smeri do najbližjega artikla. Za podatke o lokaciji in kotu uporabljamo API razvojnega ogrodja IndoorAtlas SDK (3. poglavje).

Seznam artiklov v aktivnosti pregleda artiklov nudi dodatno možnost izbire z dolgim klikom, za katerega skrbi metoda *onItemLongClick()* razreda ListView. Akcija sproži pregled tlorisa prostora z informacijo o lokaciji uporabnika in izbranega artikla, kot je to razvidno na Sliki 13.



Slika 13: Tloris z lokacijo uporabnika in artikla

Posamezna aktivnost ima lahko le en izgled, a s pomočjo razreda Inflater, ki je del osnovnega razvojnega ogrodja Android SDK, lahko nad trenutni izgled aktivnosti dodajamo druge. S tem uporabnika ne zmedemo z nepotrebnimi prehodi med aktivnosti. Slika 13 prikazuje sliko tlorisa z lokacijama uporabnika in izbranega artikla, v spodnjem delu pa prikazuje podrobnosti o lokacijah. V ozadju je še vedno razvidna aktivnost pregleda artiklov, ki ne miruje in na kateri lahko uporabnik spremlja obračanje smeri puščice do najbližjega artikla.

Tloris je hranjen v objektu tipa BlueDotView, ki deduje razred SubsamplingScaleImageView. Razred SubsamplingScaleImageView razširja razred ImageView, ki je del osnovnega razvojnega ogrodja Android SDK in ki je v aplikaciji že bil uporabljen za prikaz slik. BlueDotView je del odprtokodne zbirke za prikaz slik z možnostjo njihove manipulacije.

Poleg statičnega prikaza slike uporaba razreda BlueDotView omogoča tudi krčenje, raztezanje in obračanje slike, pa tudi risanje oblik. V osnovi razred BlueDotView podpira možnost prikaza samo ene točke na sliki, na primer, lokacije uporabnika. Z nekaj dodatnega programiranja smo razredu BlueDotView dodali možnost prikaza dodatne točke, ker smo želeli na tlorisu prikazovati tako lokacijo uporabnika kot tudi lokacijo njegove interesne točke. Tako smo tudi omogočili hranjenje obeh lokacij. Lokacija artikla je statična in se ne spreminja, medtem ko je lokacija uporabnika dinamična in se med prikazom lahko spreminja. Prikaz lokacije uporabnika na sliki se spreminja s pomočjo klicev metod *coordinateToPoint()* razreda IAFloorPlan, s katero se koordinate zemljepisne širine in dolžine preslikajo v koordinate slike tlorisa, ter *setDotCenter()* in *postInvalidate()* razreda BlueDotView, ki sprejemata osvežene lokacije in osvežita izris točke na sliki. Ta akcija osveževanja slike se

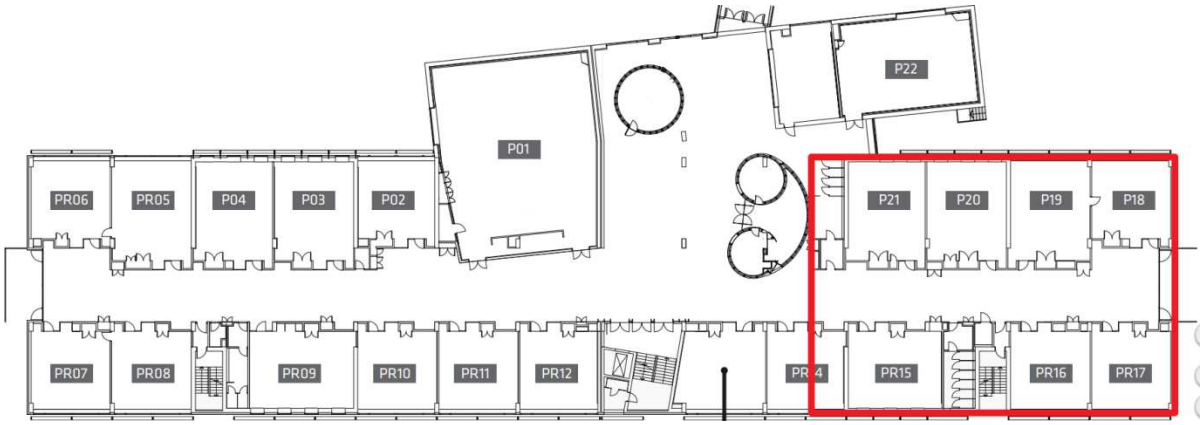
izvaja v metodi *onLocationChanged(IALocation iaLocation)*, ki je podrobneje opisana na začetku tega poglavja.

Pravilnost delovanja prototipa je bila preverjena s testiranjem v prostorih Fakultete za računalništvo in informatiko (FRI) na različnih mobilnih napravah in različnimi verzijami operacijskega sistema Android. Ugotovili smo, da ima FRI odlične pogoje za navigacijo. Razlog je v železni konstrukciji objekta (ki sicer močno popači geomagnetno polje, a posledično ustvari edinstvene geomagnetne zapise) in velikem številu oddajnikov signalov WiFi in Bluetooth, ki izboljšajo natančnost določanja lokacije.

5 Testiranje prototipa

Testiranje prototipa predstavlja zaključni del razvoja. Med testiranjem ugotavljamo točnost in natančnost lociranja ter morebitne napake. Ker smo delovanje prototipa želeli čim podrobneje preveriti, smo za testiranje uporabili več različnih mobilnih naprav z različnimi verzijami mobilnega operacijskega sistema Android. Testiranje se je izvajalo v prostorih FRI in smo ga večkrat povsem nemoteno ponovili.

Za pripravo na testiranje je bilo treba najprej ustvariti tloris FRI s pomočjo aplikacije MapCreator 2. Po uspešnem naboru informacij o tlorisu prostora so te informacije na voljo v oblačnih storitvah IndoorAtlas. Osnovni tloris je prikazan na sliki 14.



Slika 14: Tloris 1. nadstropja FRI z označenim območjem testiranja

Testiranje znotraj FRI je potekalo tri dni. Vsak dan je bilo na posamezni napravi opravljenih več meritev. V nadaljevanju predstavljamo uporabljene naprave, metode testiranja in rezultate.

Uporabljene naprave

Testiranje je bilo izvedeno na naslednjih napravah:

- Samsung Galaxy Note 4 (na tej napravi smo prototip tudi razvijali), Android 6.0.1,
- Samsung Galaxy S4 mini, Android 4.1.0,

- Samsung Galaxy J5, Android 4.0.1,
- Samsung Xcover 2, Android 4.0.1,
- HTC One, Android 5.0.0,
- LG K8, Android 4.1.0,
- LG X Power, Android 5.0.1,
- Asus ZenPad (tablični računalnik), Android 5.0.1,
- Presitigio Multipad 2 (tablični računalnik), Android 4.0.0.

Metode testiranja

Na vseh napravah je bila nameščena enaka verzija aplikacije z identično vsebino podatkovne baze SQLite. Vsaka naprava je imela tako vnaprej definiran listek z nazivom 'FRI Shopping' in vsebino:

- Coca Cola (količina 1),
- Chocolate (količina 1),
- Ice cream (količina 1),
- Notebook (količina 2).

Artikla "Coca Cola" in "Chocolate" sta imela vnaprej določeni lokaciji. Artikloma "Ice cream" in "Notebook" pa smo lokaciji določili v aplikaciji. Pri testiranju so imele naprave vklopljene sprejemnike Bluetooth, GPS in WiFi signalov.

Testirali smo:

- točnost in natančnost lociranja,
- točnost in natančnost merjenja kota med uporabnikom in artiklom.

Točnost in natančnost lociranja smo testirali z obiskovanjem lokacij zgoraj naštetih artiklov tako, da nas je do njih vodil prototip. Ob obisku točke artikla smo si zabeležili zemljepisno širino in dolžino ter se premaknili na naslednjo najbližjo točko, kot jo je izračunal prototip.

Točnost in natančnost merjenja kotov pa smo testirali tako, da smo se postavili pred artikel ‘Coca Cola’ in zabeležili, kako prototip meri kot med uporabnikom in artiklom.

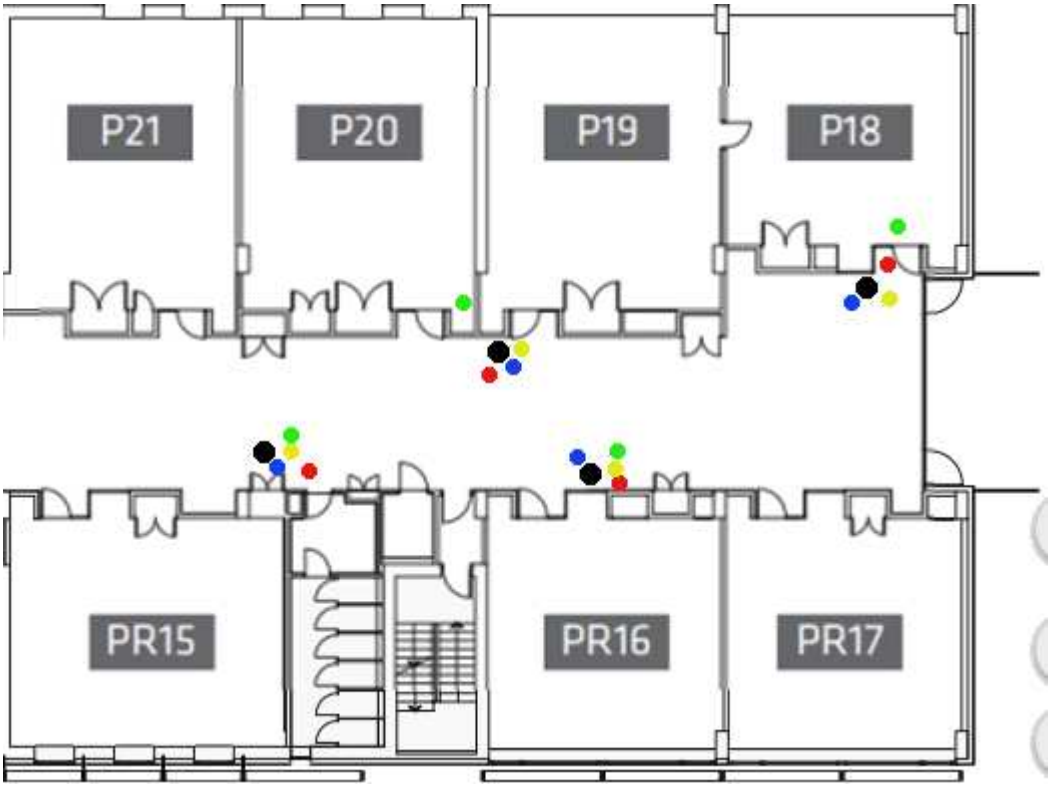
5.1 Rezultati

Na podlagi rezultatov testiranja se lahko odločamo o naslednjih korakih. Natančneje: ob uspešnem testiranju lahko razmišljamo o naslednjih stopnjah razvoja, v primeru neuspeha pa moramo razmisliti, na kakšen način lahko odpravimo napake. Kot uspešno testiranje smo opredelili vse rezultate, ki pri določanju lokacije od izmerjene razdalje odstopajo do največ 2 metrov in pri merjenju kotov do največ 45° od izmerjenih kotov. Ti vrednosti smo izbrali zato, ker bi ob tolikšnem odstopanju v praksi lahko uporabnika še vedno uspešno usmerjali do prodajnih regalov z artikli.

Točnost in natančnost lociranja

Za testiranje točnosti in natančnosti lociranja smo v prostorih FRI določili lokacije navideznih artiklov, se med njimi sprehodili ter na vsaki lokaciji navideznega artikla z uporabo posamezne naprave preverili točnost in natančnost lociranja. V praksi bi to pomenilo sprehajanje med prodajnimi regali v trgovini. Zaradi lažje preglednosti je sprehod razdeljen na dva dela. V prvem delu, prikazanem na sliki 15, smo uporabili naslednje naprave:

- Samsung Galaxy Note 4, modre točke,
- Samsung Galaxy S4 mini, rdeče točke,
- Samsung Galaxy J5, zelene točke,
- Samsung XCover 2, rumene točke.



Slika 15: Prvi del sprehoda po FRI

V drugem delu, prikazanem na sliki 16, sprehoda pa smo uporabili:

- HTC One, modre točke,
- LG K8, rdeče točke,
- LG X Power, zelene točke,
- Asus ZenPad, rumene točke,
- Prestigio Multipad 2, sive točke.

Slika 16: Drugi del sprehoda po FRI

Črne točke na slikah 15 in 16 prikazujejo lokacije artiklov. Pri opravljanju meritev smo se najprej sprehodili do točke med učilnicama P20 in P19. Od tu smo se premikali do naslednjih najbližjih točk, kot jih je izračunal prototip: točka pred učilnico PR16, PR15 in nazadnje P18. Najbližje točke so bile na vseh napravah izračunane pravilno.

Pri usmerjanju do posameznih artiklov pa niso bile vse naprave enako uspešne. Na slikah lahko vidimo, katere naprave so bile pri ugotavljanju lokacij točnejše. Opazimo lahko, na primer, da sta bili pri ugotavljanju lokacij najbolj točni napravi Samsung Galaxy Note 4 in Samsung XCover 2, saj so na slikah njune točke najbližje črnim točkam. Kot najmanj natančni sta se pri ugotavljanju lokacij izkazali napravi Samsung Galaxy J5 in Prestigio Multipad 2.

Tabela 2 prikazuje odstopanja med ugotovljenimi in dejanskimi lokacijami navidezni artiklov v metrih. Med točkami zemljepisnih dolžin in širin smo izračunali razdaljo z uporabo formule haversine, ki je primerna zlasti za izračunavanje majhnih razdalj med dvema točkama, podanima z zemljepisno širino in dolžino [14].

Telefon	PR15	PR16	P18	P19	Povprečje naprave
Samsung Galaxy Note 4	0,42	0,39	0,37	0,39	**0,39**
Samsung Galaxy S4 mini	0,71	0,70	0,93	0,51	**0,71**
Samsung Galaxy J5	1,00	0,89	1,53	1,33	**1,19**
Samsung XCover 2	0,65	0,54	0,49	0,44	**0,53**
HTC One	0,57	0,63	0,57	0,67	**0,61**
LG K8	0,78	0,67	0,84	1,39	**0,92**
LG X Power	0,99	0,76	0,71	0,94	**0,85**
Asus ZenPad	0,75	0,70	0,70	0,51	**0,67**
Prestigio Multipad 2	1,61	1,50	0,99	0,96	**1,27**
Povprečje lokacije	**0,83**	**0,75**	**0,79**	**0,79**	**0,79**

Tabela 2: Tabela odstopanj v metrih

Naprave z boljšimi rezultati, na primer Samsung Galaxy Note 4 in Samsung XCover 2, nimajo velikih nihanj, kar pomeni, da so lokacije ugotavljale enakomerno. Pri napravah s slabšimi rezultati, na primer Samsung Galaxy J5 in LG K8, pa prihaja tudi do dvakratnih razlik. Med napravama, ki imata najnižje in najvišje povprečno odstopanje (Samsung Galaxy Note 4 in Prestigio Multipad 2), znaša razlika skoraj en meter. Sklepamo torej lahko, da igra veliko vlogo pri točnosti in natančnosti lociranja sama naprava, zlasti njena strojna in specifična programska oprema, morda pa tudi zunanjost telefona. Povprečna odstopanja in razlike med njimi so na posameznih lokacijah majhna. To pomeni, da lokacija navideznega artikla ni vplivala na točnost lociranja. Da je bilo ugotavljanje lokacij enakomerno, potrjujejo ravno majhne razlike med povprečnimi odstopanji. Navkljub enakomernosti pa so nekatere naprave lokacijo ugotavljale bolj in druge manj točno.

V 2. poglavju je bila omenjena trditev ponudnika storitev Indoor Atlas, da lahko z razvojnim ogrodjem IndoorAtlas SDK lokacijo določamo do 0,2 metrov natančno. Temu smo se približali le z dvema napravama (Samsung Galaxy Note 4 in Samsung XCover 2), povprečno odstopanje je bilo 0,79 metra. Testiranje je bilo kljub temu uspešno, saj so bila vsa odstopanja manjša od vnaprej določenega dovoljenega odstopanja 2 metrov.

V praksi to pomeni, da uporabnike lahko usmerimo do določenega dela prodajnega regala, na katerem bi našli iskani artikel. Naprave z najboljšimi rezultati (na primer Samsung Galaxy Note 4) pa bi uporabnika lahko usmerile še bližje artiklu, s čimer bi mu dodatno olajšali iskanje.

Kot med uporabnikom in artiklom

Poleg navigiranja prototip uporabniku nudi tudi informacije o smeri do artikla. Točnost teh informacij v prototipu smo preverili z merjenjem odstopanja kotov. Tabela 3 prikazuje stopinjska odstopanja med izmerjenimi in dejanskimi koti, tj. med uporabnikom in artiklom. Odstopanja pri izmerjenih kotih so zaokrožena na 5°.

Telefon	**Meritev 1**	**Meritev 2**	**Meritev 3**	**Meritev 4**	**Povprečje naprave**
Samsung Galaxy Note 4	10°	10°	5°	15°	**10°**
Samsung Galaxy S4 mini	5°	10°	15°	10°	**10°**
Samsung Galaxy J5	20°	15°	20°	20°	**19°**
Samsung XCover 2	20°	15°	15°	15°	**16°**
HTC One	35°	25°	25°	25°	**28°**
LG K8	40°	30°	30°	35°	**34°**
LG X Power	40°	35°	35°	35°	**36°**
Asus ZenPad	20°	15°	15°	15°	**16°**
Prestigio Multipad 2	20°	20°	20°	20°	**20°**
Povprečje meritev	**23°**	**19°**	**20°**	**21°**	**21°**

Tabela 3: Odstopanja izmerjenih kotov

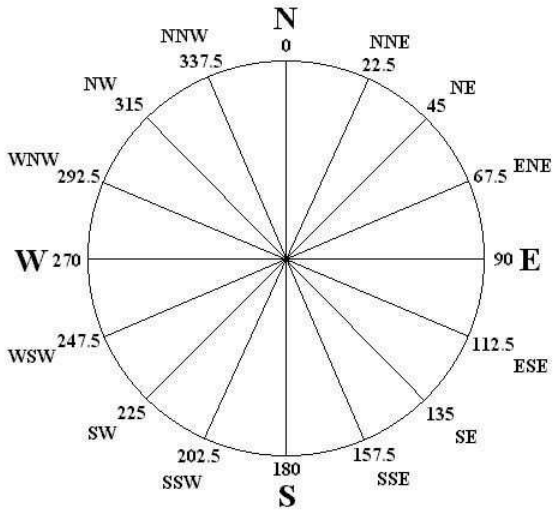
Iz zgornje tabele lahko razberemo, da nobena naprava ni izmerila povsem točnega kota med uporabnikom in artiklom. Razlog za to je lahko stanje naprave ali pa popačenost magnetnega polja, ki lahko vpliva na delovanje kompasa v stavbah tako, da magnetna igla v kompasu ne kaže vedno točno proti severu.

Sicer pa so bili rezultati podobni tistim pri računanju točnosti in natančnosti lociranja, saj so najboljše rezultate imele novejše naprave (npr. Samsung Galaxy Note 4 in Samsung Galaxy S4 mini).

Testiranje je bilo uspešno, saj je bilo skupno povprečno odstopanje enako 21°, tj. nižje od vnaprej določenega dovoljenega odstopanja 45°. Nihanja med izmerjenimi odstopanji za posamezne naprave so bila majhna, kar pomeni, da so naprave kote merile enakomerno.

Pri merjenju bi si lahko pomagali tudi z vetrnico na sliki 17, kjer velja, da smer proti severu (N, angl. North) predstavlja smer naravnost naprej, vzhod (E, angl. East) desno, jug (S, angl. South) naravnost nazaj in zahod (W, angl. West) levo. Z uporabo meritev lahko na podlagi vetrnice za vsako smer podamo interval vrednosti kotov. S pomočjo intervalov vrednosti kotov izračunani kot 21° predstavimo kot smer naravnost, izračunani kot 100° pa kot smer

desno itd. Na podlagi ustrezno razdeljenih intervalov vrednosti kotov vetrnice bi lahko uporabnika usmerjali le s pomočjo glavnih straneh neba ali - v našem primeru - v smereh desno, levo, naravnost in nazaj. Tak način meritev bi za uspešno testiranje dovoljeval večja odstopanja pri rezultatih, saj ne bi primerjali le dveh vrednosti, temveč odstopanje izmerjene vrednosti od pragov intervala vrednosti.



Slika 17: Smeri neba v stopinjah

Slika 18 primerja delovanje prototipa na napravi z najboljšimi rezultati obeh testiranj (Samsung Galaxy Note 4) in z napravo, katere rezultati so bili med najslabšimi (Samsung Galaxy J5). Tokrat smo artiklu določili lokacijo v aplikaciji in se od njega umaknili. Nato smo se z napravo (modra točka) ponovno navigirali točno do artikla. Lokacijo artikla (zelena točka) smo določili na prehodu med računalniško učilnico P20 in P19 v prostorih FRI. Opazimo lahko, da je izračunana smer pravilna (uporabnik je bil usmerjen naravnost proti artiklu), prišlo pa je do napak pri lociranju na napravi Samsung Galaxy J5.

Slika 18: Prikaz tlorisa z lokacijo na napravah Samsung Galaxy Note 4 in Samsung Galaxy J5

Rezultati merjenja oddaljenosti in kotov oz. smeri so pokazali, da lociranje v zaprtih prostorih z razvojnim ogrodjem IndoorAtlas SDK ni enako uspešno na vseh mobilnih napravah. Očitno je tudi, da so rezultati boljši in odstopanja manjša pri novejših napravah (Samsung Galaxy Note 4 in Samsung XCover2), saj so senzorji starejših naprav, na katerih smo izvajali testiranje (LG K8, LG X Power in Samsung Galaxy J5) manj zmogljivi, naprave pa imajo tudi starejšo programsko opremo. Poleg naprav je razlog za prisotnost napak lahko tudi podobna popačenost magnetnih polj različnih prostorov.

Različni prostori imajo lahko podobno popačenost magnetnih polj, zaradi česar je za natančno določanje lokacije priporočena dodatna uporaba sprejemnikov WiFi in Bluetooth signalov. S tem se točnost navigacije poveča. V nasprotnem primeru bi lahko prišlo do napačnih izračunov lokacije ali celo prikazovanja napačnih tlorisov ter posledično do vtisa, da se uporabnik nahaja v drugem objektu [27, 28]. Primer: uporabnik bi se sprehajal v eni izmed trgovin neke trgovske verige, razvojno ogrodje IndoorAtlas SDK pa bi ga zaradi podobnih geomagnetnih zapisov trgovin, poskušalo locirati v drugi trgovini iste trgovske verige. Zaradi tega bi prišlo do napak, tako pri lociranju kot tudi pri prikazovanju tlorisa. Pri testiranju znotraj Fakultete za računalništvo in informatiko to ni vplivalo na rezultate, kar je razvidno pri opisu rezultatov v tabeli 2.

6 Sklepne ugotovitve

Na podlagi rezultatov testiranja prototipa lahko zaključimo, da je izdelani prototip uspešno prestal testiranje in, da smo vanj vključili vse vnaprej določene funkcionalnosti. Ugotovimo lahko, da je bilo razvojno ogrodje IndoorAtlas SDK z enostavnostjo implementacije in številnimi orodji, ki jih ponuja, prava izbira za razvoj prototipa za navigacijo v notranjih prostorih. Z rezultati testiranja prototipa smo zadovoljni, saj so bila odstopanja izmerjenih vrednosti od dejanskih majhna. Zato bi bila implementacija te rešitve v nekem produkcijskem okolju z razvojnim ogrodjem IndoorAtlas SDK povsem sprejemljiva.

Edina omembe vredna pomanjkljivost, ki smo jo zaznali pri uporabi razvojnega ogrodja IndoorAtlas SDK, je, da je treba za branje podatkov geomagnetnih zapisov v vsakem novem prostoru najprej narediti obhod prostora z aplikacijo MapCreator 2. Pri ostalih SDK, omenjenih v 2. poglavju, tega ni treba početi. Vsekakor pa velja, da sposobnost ugotavljanja lokacije na podlagi geomagnetnih zapisov predstavlja veliko prednost pred ostalimi rešitvami, saj zmanjšuje stroške in omogoča enostavnejšo uporabo ogrodja in njegovih orodij.

Omeniti moramo, da lahko pri navigaciji v zaprtih prostorih zaradi podobnosti geomagnetnih zapisov različnih prostorov pride do napačnih izračunov lokacije, če hkrati ne uporabljamo tudi oddajnikov Bluetooth in WiFi signalov. V takšnih primerih je brez uporabe sprejemnikov WiFi in Bluetooth signalov skorajda nemogoče določiti, v katerem prostoru se nahajamo. Na tak primer, da bi zaradi podobnosti geomagnetnih zapisov razvojno ogrodje IndoorAtlas napačno določalo lokacijo, med razvojem prototipa in testiranjem sicer nismo naleteli.

Naprave z mobilnim operacijskim sistemom Android nudijo odlično strojno in programsko opremo za razvoj navigacije v zaprtih prostorih. V pomoč pri usmerjanju uporabnika do interesnih točk sta tudi senzorja za pospešek in giroskop. Dokumentacija IndoorAtlas SDK obljublja enake rezultate tudi za mobilni operacijski sistem iOS (Apple Inc).

Naslednji korak v razvoju prototipa bi bilo izboljšanje njegove uporabnosti. To bi dosegli s posodobitvijo strukture podatkovne baze s spremembo razmerja med lokacijami in artikli. V času pisanja diplomskega dela je struktura podatkovne baze takšna, da ima lahko artikel znotraj trgovine le eno lokacijo. To lahko povzroči napake pri navigiranju, če se, na primer, trgovina nahaja znotraj večjega trgovskega centra, saj se lahko en artikel pojavi na več mestih.

S posodobljeno strukturo podatkovne baze bi se temu izognili, podprli pa bi lahko tudi navigacijo v drugih zaprtih prostorih, npr. v muzejih.

Za navigacijo v muzejih bi morali ustrezno poimenovati razne aktivnosti aplikacije, ki so opisane v 4. poglavju, in določiti tipe uporabnikov. S tem bi različnim tipom uporabnikov lahko nudili različne tipe listkov. Nakupovalci bi imeli opravka z nakupovalnimi seznami, obiskovalci muzeja pa s seznami muzejskih eksponatov. Ko bi pokrili področje navigacije v muzejih, bi lahko naslednji korak predstavljala ureditev navigiranja v bolnišnicah.

Z razvojem rešitve sem podrobneje spoznal različne načine navigacije, pa tudi mobilni operacijski sistem Android. Vse to bom z veseljem nadgrajeval tudi v prihodnje. Mobilne naprave nudijo veliko več, kot od njih pričakujemo, in šele na primerih kot je naš, lahko spoznamo njihovo široko uporabnost.

Literatura

[1] Fredrick, H. (2017). *Why Doesn't GPS Work Inside a Building?* Pridobljeno na http://techin.oureverydaylife.com/doesnt-gps-work-inside-building-18659.html, (Dostopano: 01.06.2017).

[2] Venners, B. (2017). *Platform Independence.* Pridobljeno na http://www.artima.com/insidejvm/ed2/platindep.html, (Dostopano: 01.06.2017).

[3] Developers. (2017). *Sensors Overview.* Pridobljeno na https://developer.android.com/guide/topics/sensors/sensors_overview.html, (Dostopano: 10.06.2017).

[4] IndoorAtlas Ltd. (2017). *What Are Indoor Positioning Systems (IPS).* Pridobljeno na http://www.indooratlas.com/how-it-works/, (Dostopano: 11.06.2017).

[5] Android Studio. (2017). *Everything you need to build on Android.* Pridobljeno na https://developer.android.com/studio/features.html, (Dostopano: 11.06.2017).

[6] The GIMP Team. (2016). *The Free & Open Source Image Editor.* Pridobljeno na https://www.gimp.org/, (Dostopano: 11.06.2017).

[7] IndoorAtlas Ltd. (2017). *Developers.* Pridobljeno na http://docs.indooratlas.com/, (Dostopano: 12.06.2017).

[8] IndoorAtlas Ltd. (2017). *Mapping Quick Start Guide with MapCreator 2.* Pridobljeno na http://docs.indooratlas.com/app/, (Dostopano: 12.06.2017).

[9] Google. (2017). *Fused Location Provider Api.* Pridobljeno na https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi, (Dostopano: 13.06.2017).

[10] MapsPeople. (2017). *Digital Indoor Wayfinding.* Pridobljeno na https://www.mapspeople.com/, (Dostopano: 13.06.2017).

[11] Estimote, Inc. (2017). *Beacon Tech Overview.* Pridobljeno na http://developer.estimote.com/, (Dostopano: 14.06.2017).

[12] Android, Inc. (2017). *Activity.* Pridobljeno na https://developer.android.com/reference/android/app/Activity.html, (Dostopano: 14.06.2017).

[13] Tutorials Point, Inc. (2017). *Android - SQLite Database.* Pridobljeno na https://www.tutorialspoint.com/android/android_sqlite_database.htm, (Dostopano: 14.07.2017).

[14] Movable Type, Ltd. (2017). *Calculate distance, bearing and more between Latitude/Longitude points*. Pridobljeno na http://www.movable-type.co.uk/scripts/latlong.html, (Dostopano: 01.08.2017).

[15] IDG Communications, Inc.(2017). *A brief history of GPS*. Pridobljeno na http://www.pcworld.com/article/2000276/a-brief-history-of-gps.html, (Dostopano: 10.08.2017).

[16] IndoorAtlas Ltd. (2017). *Pricing.* Pridobljeno na http://www.indooratlas.com/pricing, (Dostopano: 12.06.2017).

[17] Indoo.rs GmbH. (2017*). It is SLAM time.* Pridobljeno na https://indoo.rs/the-slam-engine-is-here/, (Dostopano: 15.06.2017).

[18] Beebom Ltd. (2017). *What is GLONASS And How It Is Different From GPS*. Pridobljeno na http://beebom.com/what-is-glonass-and-how-it-is-different-from-gps/, (Dostopano: 20.06.2017).

[19] Google Inc. (2017*). Google Maps Indoors FAQs*. Pridobljeno na https://maps.google.com/help/maps/indoormaps/faqs.html, (Dostopano: 16.06.2017).

[20] Developers. (2017). *Location Strategies.* Pridobljeno na https://developer.android.com/guide/topics/location/strategies.html, (Dostopano: 10.08.2017).

[21] SQLite. (2017). *About SQLite.* Pridobljeno na http://www.sqlite.org/about.html, (Dostopano: 20.08.2017).

[22] Gargenta, Marko. *Learning android*. "O'Reilly Media, Inc.", 2011.

[23] Kaplan, Elliott, and Christopher Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.

[24] Werner, Martin. *Indoor location-based services: Prerequisites and foundations*. Springer, 2014.

[25] Pritt, Noah. *Indoor navigation with use of geomagnetic anomalies*. 2014. Pridobljeno na http://www.noahpritt.net/files/IGARSS2014-WithPubHeading.pdf, (Dostopano: 23.12.2017).

[26] Astrachan, Owen. *Bubble sort: an archaeological algorithmic analysis. 2003.* Pridobljeno na: http://cygnus-x1.cs.duke.edu/courses/cps182s/compsci342s/cps182s/fall03/latex/checkout/bubble182.pdf, (Dostopano: 24.12.2017).

[27] Li, Binghao, *et al. 'How feasible is the use of magnetic field alone for indoor positioning?.' Indoor Positioning and Indoor Navigation (IPIN)*, 2012 International

Conference on. IEEE. 2012. Pridobljeno na: https://www.researchgate.net/profile/Chris_Rizos/publication/261310654_How_feasible_is_the_use_of_magnetic_field_alone_for_indoor_positioning/links/00b7d5352eff555586000000.pdf, (Dostopano: 02.01.2018).

[28] Li, You, *et al. 'WiFi-aided magnetic matching for indoor navigation with consumer portable devices.' Micromachines*. 2015. Pridobljeno na: http://www.mdpi.com/2072-666X/6/6/747/htm, (Dostopano: 02.02.2018).