Univerzita Hradec Králové
Fakulta informatiky a managementu

Diplomová práce

Indoor lokalizace v systému pro podporu výuky

(Indoor localization in the system of education support)

Autor: Petr Weissar
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Tomáš Kozel, Ph. D.

Hradec Králové, duben 2018

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedených zdrojů.

.....

V Hradci Králové, dne 23.4.2018 Bc. Petr Weissar

Anotace

Cílem práce je vytvořit aplikaci pro podporu výuky využívající dostupné senzory v mobilním zařízení k zprostředkování indoor lokalizace. Uživateli je umožněno ovládání systému pro správu akademických aktivit, rozvrhových akcí a studijních výsledků prostřednictvím mobilní aplikace, která rovněž usnadňuje orientaci v budovách univerzity. Aplikace poskytuje pohled a interakci s událostmi a informacemi o učebnách v reálném čase a přibližuje tak uživateli polohu v rámci budov na akademické půdě. Přihlášení uživatelé aplikace mohou dále využít i širšího rozsahu služeb, které jim nabízí API systému STAG.

Annotation

The goal of a diploma thesis is to create an application for education support which also uses available sensors in a mobile device to mediate indoor localization. User can manage study academic activities, timetable and study results by mobile application. There is a real-time overview of classrooms and theirs schedules and also a module used for getting indoor position in any building of university campus. The signed-in users can use wider specter of services that is available on API of the STAG system.

Obsah

Seznam obrazové dokumentace

Seznam grafů

Seznam tabulek

Pojmový aparát

API ..... rozhraní pro programování aplikací – sbírka procedur, funkcí, aj.
DB ..... databáze
GPS ..... vojenský globální družicový polohový systém
GUI ..... grafické uživatelské rozhraní aplikací
Indoor lokalizace ..... určování polohy v zastřešených prostorech
Outdoor lokalizace ..... určování polohy v venkovních prostorech
REST ..... způsob práce s daty pomocí HTTP volání (podpora různých formátů)
SOAP ..... způsob práce s daty pomocí HTTP volání (založen na XML formátu)
SQL ..... structured query language – jazyk pro práci s daty v databázi

1 Úvod

1.1 Mobilní zařízení, motivace

Současná doba je dobou neustálého rozvoje mobilních technologií a jejich inovací. Zejména v průběhu posledních deseti let zaplavují trhy zařízení kategorie *přenosné elektroniky*[1], především chytré telefony, které mohou svojí výbavou a výkonem leckdy konkurovat levným nebo starším laptopům či stolním počítačům. Každou chvílí je představen nový typ zařízení, který se snaží na trhu konkurovat buď svou výbavou, designem nebo nízkou cenou. Až na některé extrémní případy jsou tato zařízení poměrně levná a pro uživatele snadno dostupná a počet uživatelů s chytrými telefony a nositelnou elektronikou neustále stoupá. [1] Díky jejich připojení k síti Internet skrze mobilní operátory se navíc zařízení stávají nejjednodušší cestou přístupu do kybernetického světa plného informací a tím se i zvyšuje jeho aktivní používání mezi uživateli.

Vzhledem k tomuto faktu není divu, že se vývojáři začali mnohem více soustřeďovat na využití potenciálu a své aplikace a weby začali tvořit a přizpůsobovat tak, aby byly použitelné i na kompaktních zařízeních. Typickým příkladem je responzivní design webových stránek, který se svým rozložením prvků, velikostí textů a ovládáním na kompaktních zařízeních často stává i lepším způsobem, jak danou webovou aplikaci ovládat než skrze prohlížeč na počítači. Mobilní zařízení má navíc vždy uživatel po ruce a ve volné chvíli jej často využívá právě pro získávání nových informací.

Pokud jsou systém nebo webová stránka příliš komplikované na to, aby mohly být transformovány efektivním způsobem za pomocí jazyka HTML a stylování, nebo pokud může být funkce rozšířena o *přidanou hodnotu ze zařízení*[2], pak jsou vyvíjeny nové nativní aplikace, spustitelné přímo v operačním systému zařízení, a o data k zobrazení žádají server přes jeho API.

---

[1] Mobilní zařízení kompaktních rozměrů – např.: smartphone, tablet, chytré hodinky

[2] Rychlejší zpracování obrazu, menší náročnost dat, využití senzorů apod.

1.2 Cíle diplomové práce

Cíle diplomové práce lze rozdělit do dvou hlavních částí. První se týká vytvoření aplikace, která je připojena na API školního systému, a tak studentovi slouží jako nástroj pro podporu výuky. Popis API a způsob připojení k němu popisuje kapitola 1.2.1.

Na základech této aplikace je v ní zřízen modul zajišťující indoor lokalizaci s využitím senzorů v mobilním zařízení, který v kombinaci s daty poskytovanými ze systému slouží jako nástroj pro navigaci. Tato funkcionalita je druhým hlavním cílem diplomové práce a její problematika je popsána v kapitole 1.2.2.

1.2.1 Aplikace pro podporu výuky, systém STAG

Univerzita Hradec Králové patří k těm vysokým školám, které jako svůj systém pro administraci studijní agendy zvolily informační systém STAG, vytvořený a spravovaný Západočeskou univerzitou. [2] Tento systém provází uchazeče od počátku studia až po vydání diplomu. Systém disponuje desktopovou aplikací na operační systém Windows, která je využívána především zaměstnanci a správci ze studijního oddělení při využívání širšího spektra služeb. Pro studenty a vyučující je přístup zřízen přes webové služby za pomocí prohlížeče. Student zde spravuje svůj studijní plán – zapisuje si předměty a jejich rozvrhové akce, přihlašuje se na termíny zápočtů a zkoušek, získává přehled o svých hodnoceních nebo zde spravuje svou kvalifikační práci. Pro velké obrazovky je webové rozhraní dobře přizpůsobené a intuitivní. Některé části systému např. „Průběh studia“ jsou vytvořeny v responzivním designu, ale pokročilejší funkce, jako je například vyhledání místností a jejich rozvrhových akcí, se na malém zařízení stává obtížné.

Pro vývojáře bylo v systému vytvořeno API s REST a SOAP službami, pomocí nichž je možné získávat aktuální dostupné informace nebo údaje upravovat. Služby jsou rozděleny podle několika základních kritérií. Služby, kde je nutné ověřování pomocí uživatelského jména a hesla, a služby dostupné bez ověření. Dalším kritériem je přidělená uživatelská role. Nejčastěji zastoupenou rolí je „student“ – ten má dostupné služby, např. zobrazování svého rozvrhu, úpravu osobních údajů, zápisy na zkouškové termíny, správu studijního plánu, údaje o mobilitách a mnoho dalších. Pro vyučující jsou služby rozšířené o možnosti vytváření zkouškových termínů,

vypisování témat *VŠKP*[3], zaslání hromadného mailu studentům aj. V současné době nad tímto API bylo již prezentováno několik aplikací, které jsou často zaměřeny na specifický modul systému. [3]

Cílem jedné z částí diplomové práce je vytvořit mobilní aplikaci, která s využitím výše uvedených funkcí usnadní studentovi orientaci v průběhu studia, správu jeho aktivit a kompletní informace o budovách univerzity, jejich učebnách a aktuálním stavu rozvrhových akcí v reálném čase.

1.2.2 Určování pozice

Problémem, kterým trpí častěji studenti z nižších ročníků, je orientace mezi budovami univerzity, a především v jejich prostorách. Najít včas učebnu, ve které začíná za pár minut další cvičení, se pro někoho může zdát složité. Aplikace má přístup k senzorům mobilního zařízení, díky nimž může zjednodušit řešení problému.

Pro určení pozice se již dlouhé roky používá systém GPS, který je pro potřeby outdoor lokalizace dostačující, a s celkem uspokojivou přesností lze s pomocí získané souřadnice zobrazit polohu uživatele na mapě.

Problém navigace však přetrvává, je-li uživatel v místech, kde není pokrytí družic systému GPS – typicky mezi vysokými budovami nebo přímo v nich. Existuje mnoho komerčních řešení a principů, jak indoor lokalizaci řešit. [4]

Možným způsobem je užití dalšího zařízení, u kterého bude předem dána pozice umístění a se kterým se mobilní zařízení může spojit a dle síly signálu při jejich komunikaci lze odhadnout vzdálenost k němu, a tím i pozici. Budou-li v každém místě měření v dosahu alespoň dvě nebo tři taková zařízení, pak lze na principu triangulace spočítat pozici s vysokou přesností. Nevýhodou tohoto principu je nelinearita a odrazivost vysílaného signálu. Budovy jsou stavěny z nejrůznějších materiálů s odlišnými vlastnostmi vůči elektromagnetické vlnění. V praxi se může stát, že měřitelná intenzita signálu zařízení A je vyšší než u zařízení B, a přitom je zařízení A až o několik metrů dále od mobilního zařízení nežli B. [5]

[3] Vysokoškolská kvalifikační práce

Další nevýhodou je cenová a časová náročnost takového řešení. K realizaci je zapotřebí dostatečné množství vysílacích stanic, jejichž poloha musí být správně zaznačena v systému a musí být napájeny buď přímo ze sítě, nebo pomocí baterií. V obou případech je řešení nákladné buď kvůli použité kabeláži a neustálému odběru nebo na pravidelnou výměnu baterií ve stanicích.

Další možností, jak určovat pozici v budovách, je pomocí směrového vektoru pohybu uživatele a chůze. Za předpokladu existence virtuální reprezentace místností v jednotlivých patrech budov je možné využít některé ze senzorů v mobilním zařízení, jejichž naměřené hodnoty lze analyzovat a dle určitých pravidel přepočítat na směrový vektor, četnost a intenzitu pohybu uživatele. Při zpětné analýze dosavadní dráhy pohybu lze pak určit přibližnou pozici a zobrazit objekty v okolí.

1.2.3 Podpůrné funkce

Mimo služby správy studijní agendy existují potíže s komunikací mezi studenty a vyučujícími ohledně průběhu výuky. STAG není vybaven informacemi o událostech změn ve výuce, nejsou-li změněny samotné rozvrhové akce v databázi. V současné době je situace řešena externím systémem oficiálních webových stránek univerzity v sekci „Změny ve výuce“. Případné náhlé změny agendy cvičení i přednášek vyučující přidávají na toto místo nebo je rozesílají mailem studentům.

Tato skutečnost komplikuje situaci na obou stranách. Studenti musí kvůli změnám ve výuce navštívit zcela jiné webové stránky nebo musí číst pravidelně mail. Aplikace tyto změny stahuje za uživatele a zobrazuje v seznamu, což přispívá k centralizaci dat a lepší orientaci. I přes funkci zobrazení jednoduše dostupných změn by se mohlo stát, že si student události nevšimne včas, a že je konání rozvrhové akce změněno zjistí až v učebně.

Těmto a podobným situacím lze zabránit užitím *push notifikací*.[4] V případě že nastane nečekaná změna rozvrhové akce nebo je vypsána nová zajímavá přednáška či studijní oznámení, pak se uživateli v horní části obrazovky zařízení zobrazí

[4] Způsob komunikace, oznámení iniciované serverem ke klientovi

notifikace, obsahující kategorii zájmu a samotné informace. Tato oznámení mohou vytvářet pouze autorizované osoby z akademické sféry a uživatel si může zvolit, které informace si přeje odebírat, a nechá se na ně upozorňovat.

1.2.4 Shrnutí

V následujících částech práce je soupis kompletního procesu vývoje popisovaného softwaru. Nejprve je uveden výčet použitých *HW i SW technologií* [5], nezbytných pro vývoj aplikace. V další kapitole je uvedena analýza problematiky a model aplikace s popisem principů možných řešení. V posledních částech práce následují kapitoly o samotné implementaci a testování vyvinuté aplikace.

[5] Hardware (senzory) i software (knihovny, operační systém) technologie

2 Technologie

V této kapitole jsou představeny softwarové technologie a jejich využití k připojení hardwaru zařízení. Tato kombinace se rovněž využívá při vývoji aplikace. Dále jsou zde popsány vývojové nástroje, využívané pro implementaci.

2.1 Operační systém Android

Android je open source operační systém založený na Linuxu. Jeho vývoj byl zahájen v roce 2003 a od té doby se stále rozvíjí. První verze Android 1.0 (2008) již prošel 25 aktualizacemi. V roce 2010 se mu podařilo překonat ostatní platformy v podílu na trhu. V současné době mu patří okolo 75 %. [6] Velkou výhodou je právě otevřenost systému, díky které mohou výrobci jeho prostředí dále upravovat. Pravděpodobně i z tohoto důvodu došlo k tak masivnímu rozšíření na trhu.

Primární programovací jazyk při vývoji aplikací pro operační systém Android je *Java*[6] v kombinaci s *XML*[7], ale zdrojový kód může být psán i v jazycích C/C++ (pomocí *NDK*[8]). (V posledních letech je také velmi prosazován vývoj pomocí jazyka Kotlin, nebo využití multiplatformních technologií, jako je např. React Native nebo Flutter.) Aplikace jsou spouštěny jako samostatný proces a běží ve virtuálním stroji zvaném Dalvik VM. O řízení těchto procesů a přidělování pamětí se stará operační systém. Běhové prostředí Androidu nad jádrem Linuxu realizuje interakci s nízkoúrovňovým hardwarem. Pro vývojáře je zdarma k dispozici Android SDK, obsahující Android API, nástroje a dokumentaci. Oficiálním IDE je v současné době Android Studio. [7]

2.1.1 Možnosti mobilních technologií, senzory

Přes Android API a jádro OS je k dispozici přístup k naměřeným hodnotám ze všech dostupných senzorů. Instanci třídy „SensorManager“ je možné zaregistrovat k odposlouchávání měření na senzorech. V takový okamžik je zaslána zpráva všem instancím, které těmto změnám naslouchají. Některé ze senzorů mají pravidelnou

[6] Objektově orientovaný jazyk vyvinutý společností Sun Microsystems
[7] eXtensible Markup Language – obecný značkovací jazyk vyvinutý konsorciem W3C
[8] Native Development Kit – pro práci s knihovnami libc a libm

odezvu měření, jiné měří hodnoty s ohledem na výdrž baterie zařízení a zprávu s hodnotami zašlou, pouze nastane-li změna nebo určitý stav předurčený ke spouštění události. Ke každému měření je přidáno i časové razítko obsahující přesný čas v nanosekundách.

2.2 Vhodné senzory pro indoor lokalizaci

Z celé řady dostupných senzorů, ke kterým API nabízí přístup, je možné využít takové, které jsou schopny vracet měřené hodnoty vyjadřující nějaký pohyb zařízení nebo umožňují orientaci zařízení v prostoru světa. Takové údaje jsou využitelné k realizaci indoor lokalizace. Některé z těchto senzorů měří aktivitu na třech osách, které jsou orientovány buď vůči zařízení (viz obrázek 1), nebo vůči Zemi (viz obrázek 2). Popis jednotlivých vybraných senzorů pro aplikaci je uveden v následujících kapitolách.

2.2.1 Akcelerometr, lineární akcelerometr

Akcelerometr snímá změny zrychlení na jedné ze tří os. Mimo pohyb vyvolaný změnami pozice zařízení vždy měří i gravitační zrychlení, které je v Androidu využíváno například pro detekci naklopení zařízení, podle čehož může reagovat zobrazením na šířku/výšku. Tříosý akcelerometr je zkonstruován třemi akcelerometry a všechny 3 hodnoty z každé osy jsou zasílány do volání události. Dalším důležitým využitím akcelerometru je detekce kroků nebo pádu. Lineární akcelerometr se hodnotami od toho klasického liší vynecháním hodnot gravitačního zrychlení. [8]

2.2.2 Gyroskop

Gyroskop získává informace o rotaci zařízení vzhledem k jednotlivým osám zařízení. V kombinaci s akcelerometrem dokáže získat kompletní informace o pohybu zařízení i s jeho natočením. Právě hodnoty gyroskopu jsou vhodnou přidanou hodnotou do výpočtu možného pohybu uživatele. [8]

2.2.3 Magnetometr

Magnetometr (kompas) slouží k určení úhlu zařízení vzhledem k severnímu magnetickému pólu. Volání obsahuje hodnoty všech tří os. Ze získaných hodnot čerpá senzor zvaný „Rotation vector“, který vyjádří natočení zařízení reprezentované *kvaternionem*[9]. [8]

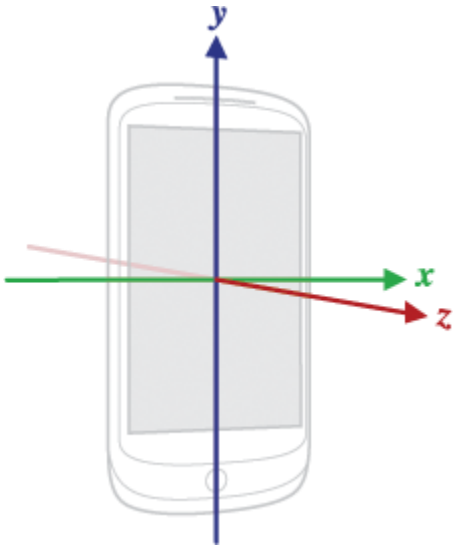
2.2.4 Barometr

Barometr je měřič hodnot okamžitého atmosférického tlaku prostředí. Patří ke skupině senzorů měřících okolní prostředí. Jeho nevýhodou je nižší rozšíření v zařízeních – často se implementuje pouze do dražších procesorů. [8]
Stejně tak jsou naměřené hodnoty ovlivněny, je-li zařízení vystaveno i jinému než atmosférickému tlaku (klepání prstem na displej apod.).

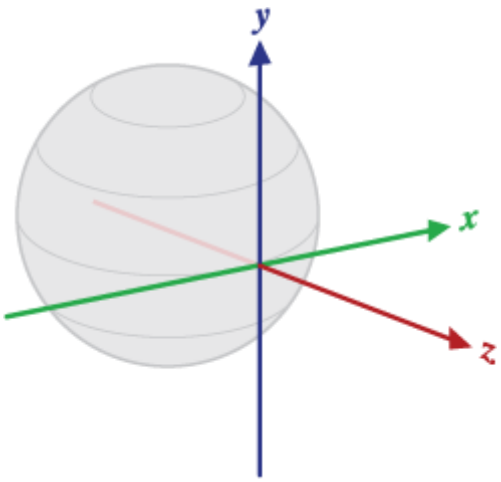
2.2.5 Krokoměr

Krokoměr vyvolává událost, jestliže uživatel učiní krok. [9] Výhodou je nižší energetická náročnost, jelikož je implementován hardwarově. Nevýhodou senzoru je nižší počáteční přesnost – pravděpodobně z důvodu eliminace nechtěných pohybů, které se svým průběhem podobají lidskému kroku. Často se stává, že senzor začne s voláním události až po cca pěti krocích, což může být v jistých případech nevhodné.

[9] Uspořádané čtveřice reálných čísel se speciálně definovanými operacemi sčítání a násobení.



Obrázek 1 osy prostoru senzoru vzhledem k zařízení [10]



Obrázek 2 osy prostoru zařízení vzhledem k Zemi [10]

2.3 Podpůrné vývojové nástroje

Vývoj softwaru je proces sestávající se z několika kroků, jejichž výsledkem je vytvořená aplikace. Zpravidla je aplikace tvořena po částech při, čemž je po každé větší části vydána do testování, aby mohly být vyhledány případné chyby, které ve zdrojovém kódu vznikly. Všechny úpravy zdrojového kódu je vhodné ukládat pomocí verzovacího systému, díky němuž lze snadno obnovit uvedené změny. Také

usnadňuje týmu vývojářů současnou práci na projektu. Mezi další z nástrojů pro vývoj stabilní aplikace patří knihovny umožňující zachycení chyb v aplikaci a jejich evidenci na místo, odkud je může vývojář ihned zjistit a řešit dříve, než je uživatel vůbec oznámí.

2.3.1 Verzování kódu aplikace

Při psaní rozsáhlé práce je vždy dobré si uchovávat jednotlivé verze před posledními úpravami. Píše-li někdo článek nebo dokument na etapy, pak si může každou verzi uložit jako kopii nejlépe na jiný disk, aby v případě poškození souboru nepřišel o veškerou práci, kterou doposud vykonal. Programování a vývoj jsou v podstatě také pouze tvorba či úpravy zdrojového kódu v textových souborech do podoby, ve které bude překladačem zkompilován, a tak bude program určitým způsobem spustitelný.

K těmto účelům byl vytvořen v roce 2005 projekt Git, na jehož vývoji má největší zásluhy Linus Torvalds. Git funguje na principu verzování změn provedených na jednom nebo více souborech. Pro každé uložení změn se používá „commit“, který změny uzavře, a mezi těmito stavy se lze přepínat. Jsou-li změny připraveny pro odeslání na server, kde budou perzistentně a bezpečně uloženy, pak se používá příkaz „push“, který ověřeného uživatele připustí k zápisu změn na server. Obdobně pro stahování ze serveru platí podobný princip, a to stahování online změn příkazem „pull“.

Výhodou tohoto systému je možnost současné práce na projektu celého týmu vývojářů, aniž by došlo ke ztrátě dat. Logicky může docházet při sjednocování kódu ke konfliktům, ale existuje mnoho nástrojů na řešení těchto tzv. „merge konfliktů“. [11]

Jedním takovým nástrojem disponuje i samotné Android Studio. Stejně tak obsahuje i grafické rozhraní pro ovládání příkazů Gitu, ale správu těchto změn lze vykonávat i v externích programech, kterých existují na různé operační systémy také desítky.

2.3.2 Nástroje řešení chyb

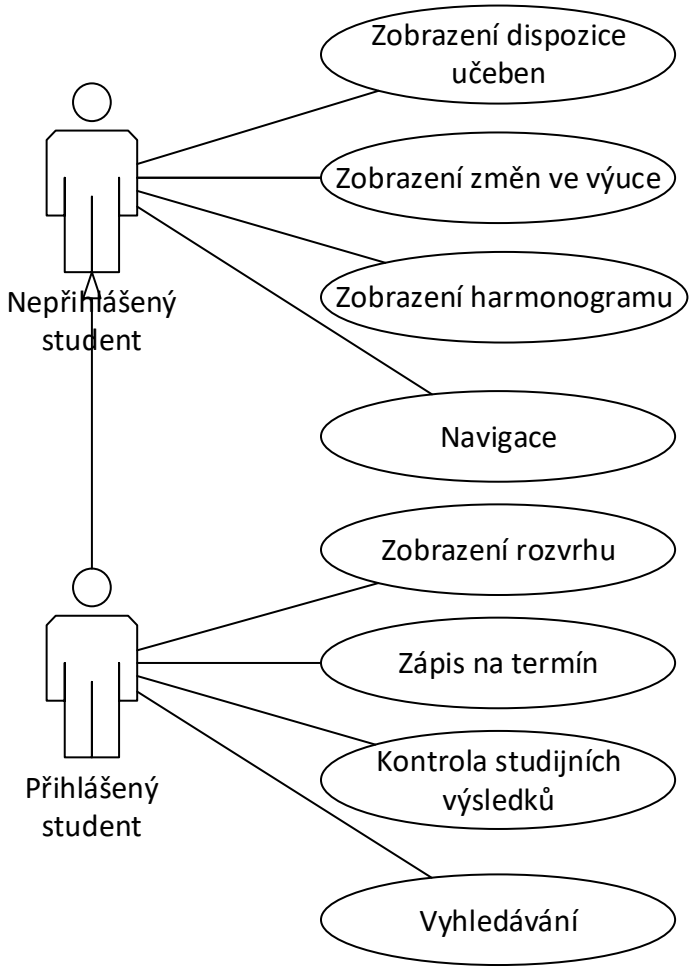
Pro odhalování chyb v aplikaci je poměrně často využíván nástroj „Crashlytics“. [12] Do projektu je jednoduše vložen a po registraci aplikace funguje tak, že všechny chyby, které se v aplikaci vyskytnou, ihned zašle na server s kompletním výpisem či kódem chyby, stejně jako s údaji o zařízení a operačním systému. Takový nástroj usnadňuje odhalování chyb mezi uživateli na jednotlivých zařízeních. Opravy mohou být řešeny téměř okamžitě a stejně tak jsou vydávány opravné verze.

3 Model aplikace

Tato kapitola pojednává o provedené analýze před začátkem vývoje, kde jsou stanoveny typové úlohy, funkční požadavky, uživatelské role, způsob komunikace se serverem, grafické rozhraní či perzistence dat.

3.1 Analýza požadavků

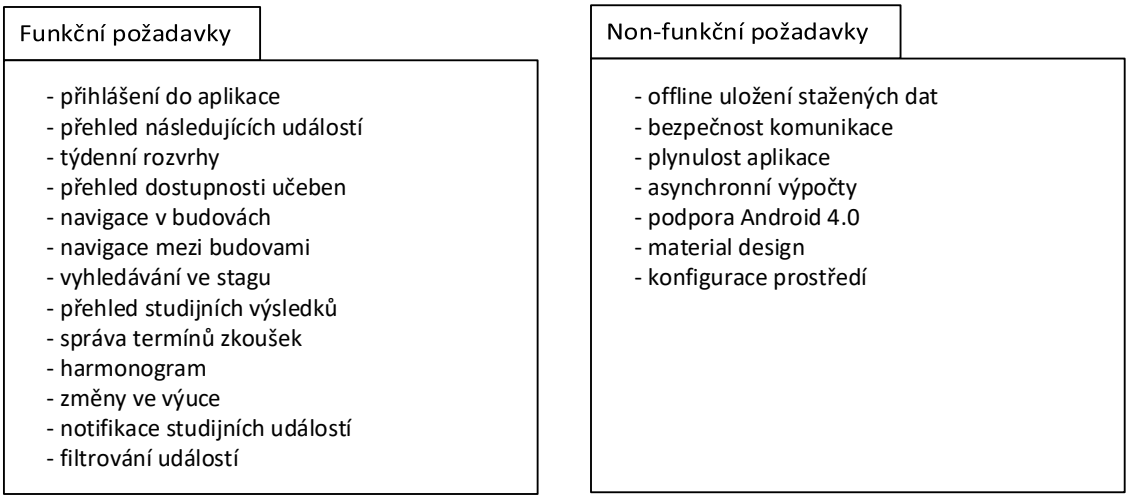
Před samotnou implementací byl zhotoven *diagram užití*[10], který shrnuje všechny dostupné nebo žádané funkce aplikace. Zohledněny jsou role přihlášeného i nepřihlášeného uživatele a jejich typové úlohy jsou uspořádány tak, aby bylo možné je realizovat za pomocí služeb STAG API. Nepřihlášený uživatel má z toho důvodu logicky menší počet možných funkcí v aplikaci.



Obrázek 3 diagram užití aplikace [autor práce]

[10] Diagram vnějšího pohledu na modelovaný systém (jak systém vidí uživatel)

Z uvedených typových úloh lze sestavit seznam funkčních požadavků a s přihlédnutím k *best-practices*[11] vývoje mobilních technologií a potřebám uživatelů jsou stanoveny i non-funkční požadavky.



Obrázek 4 funkční a non-funkční požadavky [autor práce]

Plynulost aplikace, která se řadí mezi non-funkční požadavky, během vykonávání firemních procesů aplikace znamená následující: Na hlavní procesní vlákno aplikace jsou kladeny, mimo operace vykreslování grafického uživatelského rozhraní, i zbytky výpočtů v aplikační logice. Typicky se jedná o řazení seznamů, formátování textů, *I/O operace*[12], volání webových služeb aj. Jsou-li takové operace náročné na čas nebo výkon procesoru, pak jsou prostředky rozdělovány způsobem, že může docházet k vynechání vykreslování obrazu nebo jeho zpoždění. Pro uživatele je takový stav nepříjemný a kazí celkový *UX*[13] aplikace. Jsou-li předem známy takové případy možného výskytu, pak je možné zpracovávat je na vedlejších vláknech procesoru paralelně, čímž neovlivní výpočty obrazu grafického rozhraní. Až ve vhodný okamžik jsou načtená data předána na hlavní vlákno pro dodatečné zpracování. Samotné UX je pak tvořeno s ohledem na „material design“.

[11] Osvědčené postupy, procesy či metody řízení, pomocí kterých se ve více společnostech dosáhlo dobrých výsledků a používají se proto jako doporučení pro ostatní
[12] Input/Ouput operace – zápis či čtení souborů skrze souborový systém
[13] User eXperience – „prožitek“ z používání aplikace

3.2 Material design

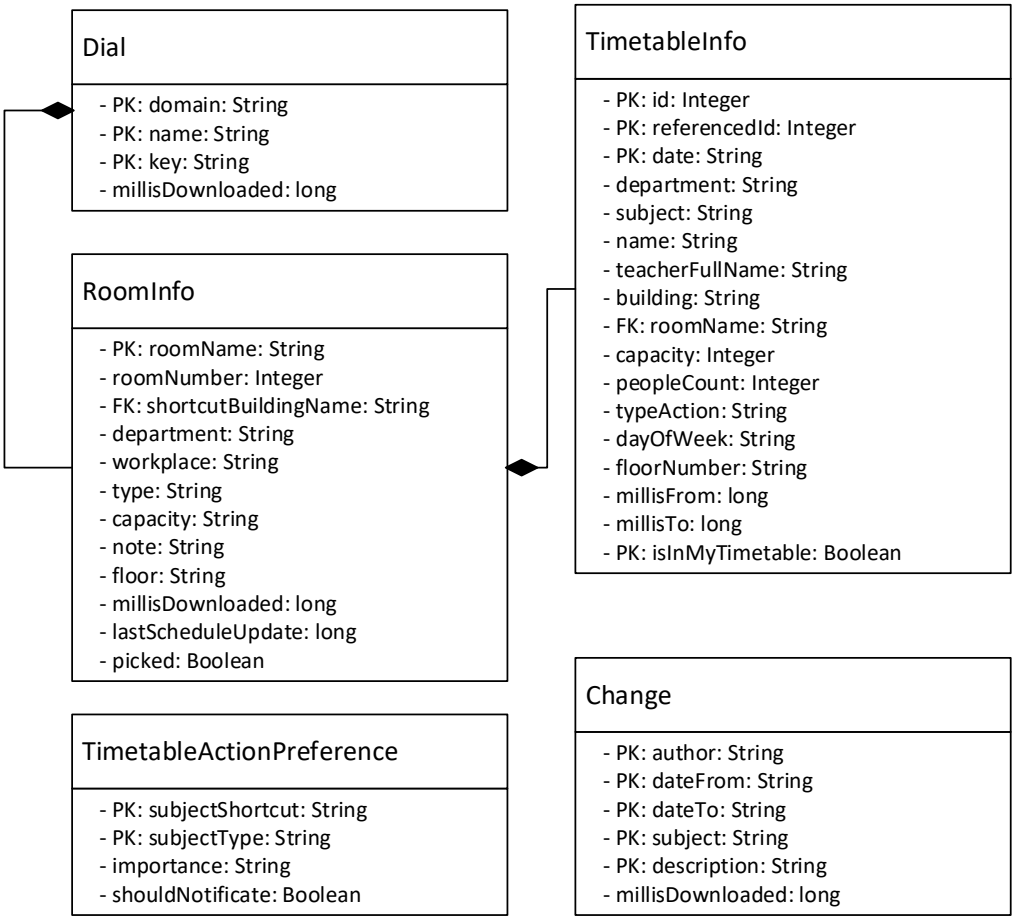
„Material design“ je označení pro grafický jazyk vyvinutý společností Google, který byl světu představen v roce 2014 na Google I/O konferenci. V dokumentaci jazyka jsou popsány všechny případy užití grafických prvků, které jsou vývojářům doporučovány při vývoji nových aplikací. Výsledkem dodržování těchto pravidel jsou jednoduché, čisté aplikace s jasnými ovládacími prvky, které jsou využívány napříč komunitou Android vývojářů a uživatelé jsou na tuto grafiku často zvyklí.
Grafický jazyk mimo pravidel rozměrů, barev či překrývání jednotlivých vrstev komponent také obsahuje popis ikon. Vývojářům jsou k dispozici stovky ikon přímo z dílen Googlu v rastrovém i vektorovém provedení, nebo si dle stanovených pravidel mohou ikony tvořit sami. Od novějších verzí Android je možné převést SVG soubory s vektorovým zápisem ikony do formátu XML. Zařízení vykresluje data přesně pro potřeby velikosti a hustoty pixelů zobrazovače. Pro starší verze je nutné užití rastrového provedení ikon, nebo implementace zpětně kompatibilních komponent, které převody z vektoru do rastru vykonají. [13]

3.3 Komunikace klient-server

Jedním z nejvíce využívaných způsobů komunikace se serverem pomocí jednoduchých HTTP volání jsou způsoby využití REST (Representational State Transfer) nebo SOAP (Simple Object Access Protocol). SOAP je protokolem pro výměnu zpráv přes síť, které jsou založeny na XML formátu. Zasílané zprávy obsahují všechny náležitosti, které při zpracování požadavku pomohou serveru při autorizaci a určení definovaného chování od odesílatele, které má server s daty vykonat. REST pracuje na podobných principech, a oproti SOAP má výhodu v možnosti jiných formátů požadavku. Z tohoto důvodu je REST kombinován s JSON (JavaScript Object Notation), který je dobře čitelný jak pro strojové zpracování, tak i pro člověka a v posledních letech se stává velmi populárním.
STAG API podporuje obě z výše uvedených metod a v případě REST zvládne pracovat i se zápisem JSON, což je pro potřeby aplikace vhodné, jelikož pro tento způsob komunikace existuje pro Android mnoho open-source knihoven, které usnadní vývoj. [14]

3.4 Databáze

Nejlepší informace jsou aktuální informace – rozvrhové akce, výsledky zkoušek nebo třeba změny ve výuce jsou kategorií dat, pro které se provádí časté změny v systému. Při zavolání služby pro získání některých z těchto údajů server vrací poslední dostupná data a jejich stav je v tu chvíli aktuální. Pokud chceme docílit nižší náročnosti na server nebo snížit průtok dat po síti, pak je vhodné vytvořit si v aplikaci zrcadlovou kopii posledně získaných dat a uživateli zobrazovat tyto údaje uložené i za cenu možného zastarání dat v případě, že by byly mezitím na serveru změněny. Jedním z případů, kde se vyplatí ukládat informace do lokální databáze jsou rozvrhové akce. Jsou-li pro každou učebnu na celý týden staženy aktuální informace o průběhu výuky v nich, pak se počet záznamů pro všechny vybrané místnosti pohybuje ve stovkách. To je pro server a přenos dat poměrně velký počet operací nutných pro odpověď. Dle těchto kritérií byla stanovena struktura lokální databáze.



Obrázek 5 struktura DB aplikace [autor práce]

„Dial“ je tabulka kopírující strukturu číselníků, které obsahují budovy univerzity. Pro každou takovou budovu jsou staženy všechny místnosti („RoomInfo“). Tyto dvě tabulky se aktualizují ručně a nejsou časově omezeny, jelikož počet budov nebo učeben se v nich nemění tak často, aby musely být po určité chvíli obnoveny.

Jiným případem je „TimetableInfo“, který obsahuje nezbytné údaje rozvrhové akce pro každou učebnu. Mimo atributy kopírující hodnoty systému STAG je zde i jeden z kompozitních primárních klíčů „isInMyTimetable“ z toho důvodu, že stejná rozvrhová akce stažená pro učebnu může být logicky navrácena i při stažení rozvrhu pro studenta. Při aplikaci pozdějších změn nebo nastavení pro konkrétní akci by mohla být ovlivněna i v sekci, kde ovlivněna být neměla.

Bez vazby na okolí jsou samostatně umístěné tabulky změn ve výuce („Change“) a nastavení chování rozvrhové akce („TimetableActionPreference“). Student si pomocí vytvoření záznamu v tabulce může například nastavit, jestli chce ve svém rozvrhu zobrazovat danou událost a dle důležitosti zobrazovat notifikace.

Tabulky se vytváří za užití technologie SQLite, která si data ukládá do souboru a zpřístupňuje užitím syntaxe SQL přístup a úpravy dat stejně jako standardní relační databáze. SQLite podporuje také omezenou sadu operací nad tabulkami, jako třeba přidání nového sloupce apod. [15]

4 Návrh a implementace aplikace

V této kapitole je popsán vývoj aplikace. Po úvodním představení životního cyklu a ovládacích prvků aplikace je popsána komunikace se serverem a řešení perzistence dat. Závěrem jsou navrženy funkce pro podporu výuky včetně jejich obrazovek.

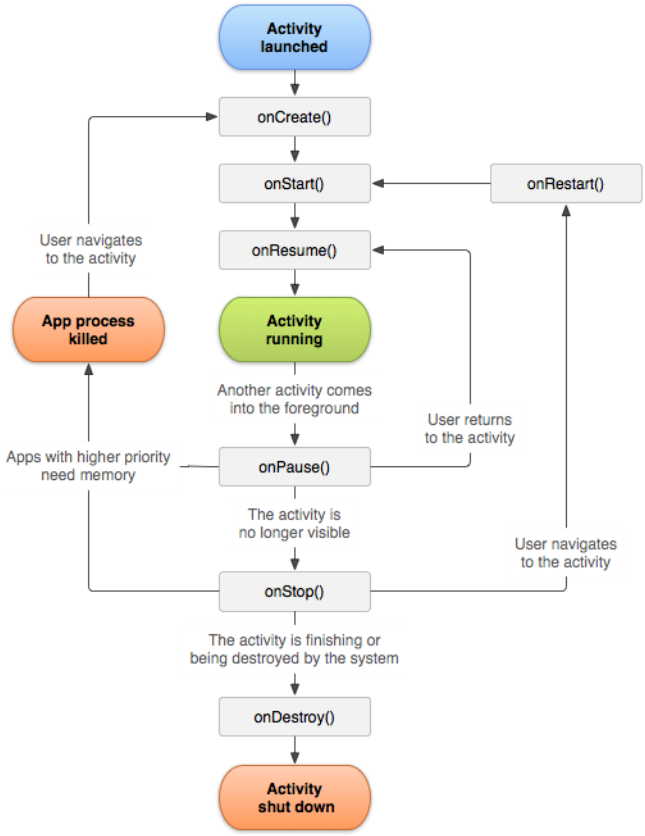
4.1 Životní cyklus aplikace, uspořádání

Základní jednotkou většiny Android aplikací je instance třídy „Activity“ (dále jen aktivita). Aktivita je základním kamenem pro aplikaci a v podstatě reprezentuje jednu obrazovku aplikace. Obrazovky se mohou navzájem překrývat a navigace mezi nimi je tvořena buď za pomoci systémových tlačítek nebo ovládacích prvků ve vyvíjených formulářích. V „manifest“ souboru aplikace je zvolena aktivita, která se spouští jako první a přichází do svého životního cyklu. Z vykonaného kódu je pak již možné pracovat s grafickým uživatelským rozhraním (dále jen GUI), daty nebo spouštět další takové aktivity. V metodě „onCreate“ v aktivitě se nastaví, z jakého XML souboru má být sestaveno GUI a jakým způsobem na něj bude aplikace reagovat a jak s ním bude manipulovat. [16]

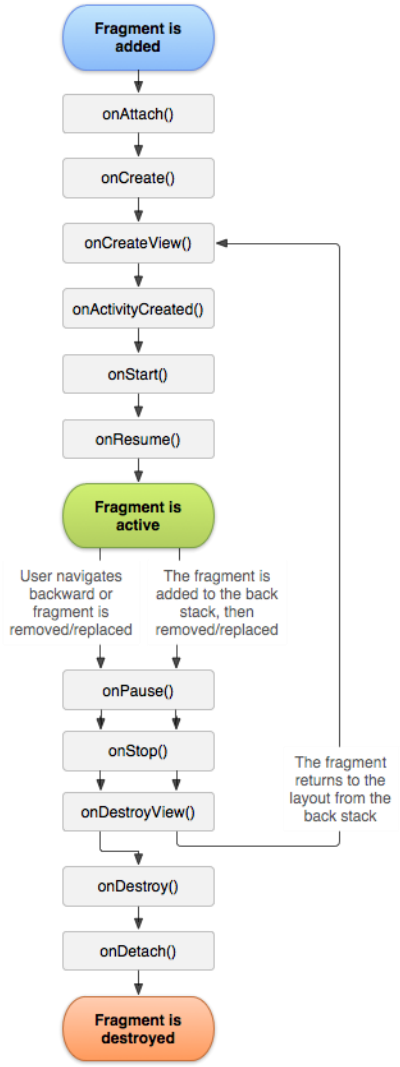
V rámci aktivit se řeší i ukládání vepsaných či zjištěných dat při otočení orientace zařízení. Do dočasné datové struktury jsou vložena data přes identifikační klíče do asociativního pole a při znovuvytvoření jsou data navrácena. V tento moment rovněž dochází k destrukci GUI a tvoří se nové – výhodou tohoto principu je možná záměna rozložení ovládacích prvků a rozdílné přizpůsobení obrazu na šířku a na výšku.

Aplikační logiku a grafické rozhraní lze mimo aktivit rozdělit do fragmentů. Fragmenty jsou struktury, které do jistých hranic vychází z životního cyklu aktivity a obsahují velmi podobná volání a možnosti. Mezi výhody patří využití fragmentů pouze pro část obrazové plochy nebo možnost jejich přepínání v rámci jedné aktivity při zachování částí grafického rozhraní užitím instance z třídy „FragmentManager“. Fragmenty jsou vývojářskou komunitou často využívány, jelikož mohou být umístěny do různých komponent a mohou běžet současně v rámci

jednoho okna. Následují obrázky porovnání životního cyklu aktivity a fragmentu, kde jsou patrné korelace jejich životních fází. [17]



Obrázek 6 životní cyklus aktivity [16]



Obrázek 7 životní cyklus fragmentu [17]

Pokud využijeme uvedených vlastností fragmentů, může být aplikace tvořena jednou aktivitou, která je zodpovědná za vytváření a orchestraci jednotlivých komponent a funkcí aplikace, jejichž funkcionalita je implementována v rámci jednoho fragmentu. Výběr funkce je realizován skrze typické menu skryté v bočním panelu a je to v podstatě jediná akce, kterou aktivita vykonává. Přístup k datům a volání webových služeb jsou realizovány v konkrétním fragmentu. Z velké části

fragmenty obsahují seznamy, jejichž adaptér je napsán jako vnořená třída v kódu fragmentů, což umožňuje přístup k atributům fragmentu při současném udržení struktur a čistoty kódu.

4.2 Komunikace se serverem

V každém fragmentu se na počátku jeho životního cyklu volá metoda „initItems“. Jejím úkolem je zjistit aktuální stav dat a zvolit způsob jejich získání. Pro uložení dočasných hodnot je v Android k dispozici práce se „shared preferences“. Jedná se o XML soubor, který pod stanovenými klíči udržuje perzistentně vložené hodnoty, a to i při vypnutí aplikace nebo dokonce restartu systému. Soubor s hodnotami se maže pouze při ručním vymazání dat aplikace nebo při odebrání aplikace ze zařízení. Tento způsob je vhodný pro uložení uživatelského jména, času posledních aktualizací apod. Není však zcela vhodný pro ukládání hesel v *plain textu*[14] z toho důvodu, že do souboru může nahlédnout kterýkoliv uživatel pomocí určitých nástrojů.

Pro potřeby záznamu o poslední aktualizaci dat je však postačující, a proto je využit při zjišťování stáří posledních dat. Při stažení nových hodnot ze STAG API se do tohoto souboru přidá údaj obsahující časové razítko v milisekundách nynějšího času. Je-li nastavena hranice pro obnovení dat na určitý čas, pak je možné s tímto údajem dopočítat uplynulou dobu, a převyšuje-li čas minimální hranici, pak jsou data obnovena z webových služeb.

K volání webových služeb je zapotřebí zaslat správný *request*[15] na server s údaji, které chceme stáhnout nebo společně s autorizačními údaji. Jelikož téměř každá aplikace pracuje s webovými službami, existuje obrovské množství open-source knihoven umožňující jednodušší práci při vytváření hlavičky nebo těla requestu a stejně tak i se zpracováním odpovědí ze serveru.

Jednou z takových knihoven je „Retrofit“, umožňující například používat anotace pro definici volání webové služby. Tím zjednodušuje plnění požadavku. Stejně tak umožňuje asynchronní volání, díky kterému není hlavní vlákno pozastaveno

[14] Obyčejný čitelný text bez zakódování
[15] Požadavek na data

a časové prodlevy vzniklé při navazování spojení a přenosu dat nijak neovlivní průběh činnosti aplikace. [18]

Součástí je i knihovna GSON, která s užitím *reflexe*[16] mapuje a dosazuje příchozí hodnoty formátu JSON na požadovanou *DTO*[17] třídu. Zpětně umí z dat v atributech takové třídy sestavit text ve formátu JSON. S těmito vlastnostmi pak stačí vytvořit třídu, jejíž atributy budou pojmenovány stejně jako klíčové názvy v JSON, nebo jim bude přidána anotace „@SerializedName“, do níž je klíčová hodnota vložena. Vývojáři STAG použili kombinaci českého a anglického jazyka pro klíčové hodnoty v požadavcích a odpovědích z *WS*[18], a proto je právě tato anotace využívána v každé DTO třídě, aby byly atributy definovány v anglickém jazyce.

Způsobů, jak ověřit autorizovaného uživatele pomocí takových requestů, existuje celá řada. Častým způsobem je užití vygenerovaného tokenu serverem z přechozího autorizačního volání, kde byly s užitím šifrování zaslány přihlašovací údaje uživatele a tento otisk byl na serveru porovnán s lokálními údaji. Když se hodnoty shodují, uživateli je poskytnut textový řetězec – token – který je do requestů zasílán buď v hlavičce, v *URL*[19] adrese nebo v těle požadavku.

Komunikace se serverem STAG funguje na principu „basic access authentication“. Funguje tak, že je kombinace přihlašovacího jména a hesla převedena na textový řetězec za užití kódování „Base64“. Řetězec je přidáván k hlavičce do každého requestu, který je na server zaslán. Vygenerovaná hodnota je zpětně převeditelná na původní text. „Base64“ tedy v podstatě neposkytuje ochranu, co se týče bezpečnosti při odposlouchávání nebo zachycení celého znění požadavku. Aby byl tento princip bezpečný, je užíván v kombinaci s protokolem HTTPS, který sám o sobě zajišťuje šifrovanou komunikaci mezi dvěma uzly v síti. [19] [20]

---

[16] Schopnost programovacího jazyka zjistit za běhu zdrojového kódu informace o určitém programovém objektu

[17] Data Transfer Object – třída, sloužící převážně pouze pro přenos získaných dat mezi částmi aplikace

[18] Web Services – webové služby

[19] Uniform Resource Locator – řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací

4.3 Přístup k perzistentním datům

Databáze systému Android je zajištěna za pomocí SQLite a i zde existuje mnoho způsobů, jak s ni pracovat. Jedním ze způsobů je napsat si pro každé volání vlastní text obsahující SQL příkaz. Těmito příkazy je nutno před prvním spuštěním aplikace vytvořit databázi a všechny tabulky, jejich sloupce, primární klíče apod. Veškeré dotazy, které jsou nad tabulkami volány, se vrací v podstatě ve formě dvourozměrného pole. Přístup k němu je zpravidla zřizován přes kurzor, kterým lze posouvat vertikálně po řádcích pole. Na každém řádku jsou k dispozici hodnoty dle čísla nebo názvu sloupce. Tento způsob není příliš přehledný, a mohou se tedy často vyskytovat chyby duplicitních názvů sloupců při spojování tabulek v dotazech. Vývoj se těmito záležitostmi zdržuje.

Moderním způsobem je využití objektově relačního mapování (ORM), což je programovací technika zajišťující automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem. V praxi to znamená, že je vytvořena třída, jejíž atributy se shodují se sloupci dané tabulky. Při volání vnitřní mechanismus zařídí převod z kurzoru rovnou na objekt naplněný správnými daty.

Jednou z knihoven, podporující tyto funkce, je knihovna DBFlow. [21] Tabulky a její sloupce jsou tvořeny za užití anotace, přičemž během vytváření buildu aplikace jsou vytvořeny generované třídy, které mapování z relační DB na objekty realizují. V rámci knihovny jsou dostupné statické metody, kterými se lze jednoduše dostat k sestavení dotazu se správnou sémantikou, což umožňuje eliminovat chyby. Další zásadní výhodou je řešení databázových migrací. Tabulky v SQLite lze např. rozšiřovat o nové sloupce. Nastane-li potřeba tabulku upravit, je pouze navýšena verze DB v obslužné třídě a skrze anotaci určující migraci dat je na zařízení při tomto přechodu zavolána definovaná změna.

Je zde rovněž řešen asynchronní přístup k datům a navrácení přes zpětné volání až v okamžiku kompletního načtení. Pokud je během firemního procesu při načítání dat zjištěno, že jsou předchozí stažená data stále aktuální, pak je místo volání webových služeb využito asynchronní načítání nedávno stažených dat a jsou uživateli prezentována jako aktuální.

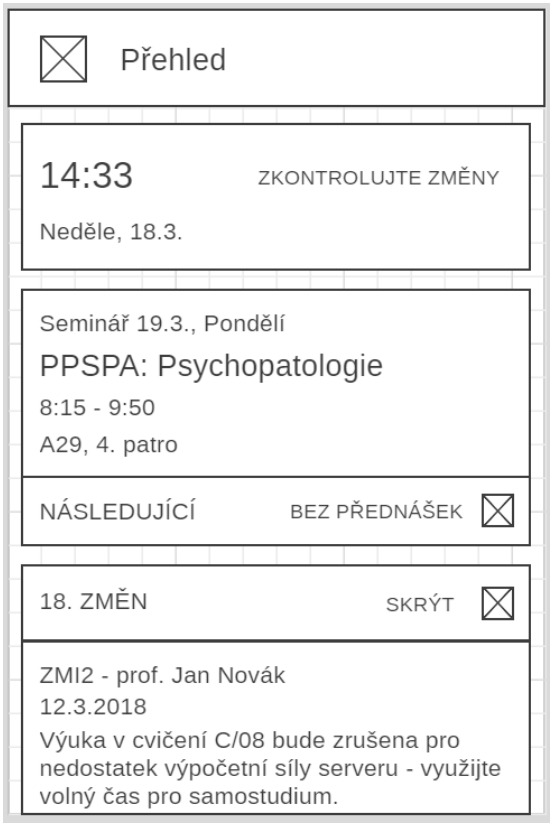
Rychlost čtení z DB je zpravidla v rámci milisekund, ale i přes tyto mimořádné vlastnosti je v GUI indikován průběh načítání pro případ, že by byly systémem I/O operace pozastavovány, nebo by byl dotaz velmi náročný. Využití databázového úložiště a asynchronního načítání opět přispívá k plynulosti celé aplikace a ke snížení datové náročnosti při využívání webových služeb.

4.4 Funkce pro podporu výuky

Fragmenty s rozdílnými daty logicky obsahují jiné grafické rozhraní položek dle vlastností dat. V některých případech mohou být informace shlukovány a v seznamu se zobrazují jako malé skupiny, jindy jsou položky řazeny po jedné a může nad nimi být logicky k dispozici interakce, vedoucí k další rozšiřující akci. Vzhledem k charakteru zobrazovaných dat přizpůsobuje každý ze seznamů svůj zobrazovaný obsah se správným rozložením ovládacích prvků. To vše samozřejmě s využitím doporučení a standardů „material designu“. Následující kapitoly popisují jednotlivě charakteristiky a vhodné rozložení dat, popř. možnosti dodatečných interakcí.

4.4.1 Přehled

Úvodním fragmentem a taktéž základním stavebním kamenem aplikace je sekce celkového přehledu. Oproti ostatním částem aplikace je tato složena z několika druhů informací, jakými jsou například rozvrhové akce přihlášeného studenta nebo změny ve výuce. Rozložení položek by mělo hned na první pohled uživatele informovat o veškerých probíhajících nebo následujících aktivitách či výjimkách v jeho denní agendě. Přepokládaná je karta s aktuálním datem a časem, která je rovněž využita pro zobrazování upozornění. Jsou-li ve změnách ve výuce detekovány shody s jedním z předmětů v uživatelském rozvrhu, potom je aktivní indikátor, který na tyto skutečnosti upozorňuje. Další z karet v sobě ukrývá rozvrh uživatele na jistý časový úsek dopředu. Standardně je zobrazována jedna rozvrhová akce, která aktuálně probíhá nebo je nejbližší v pořadí. Tato záložka obsahuje interaktivní prvky, jimiž lze zobrazit více z následujícího programu. Je zde i jednoduchý filtr sloužící k rychlému přeskupení informací. V neposlední řadě jedna z karet obsahuje výpis všech aktuálních změn ve výuce pro danou fakultu studenta. *Wireframe*[20] takového fragmentu může vypadat jako na uvedeném obrázku 8.

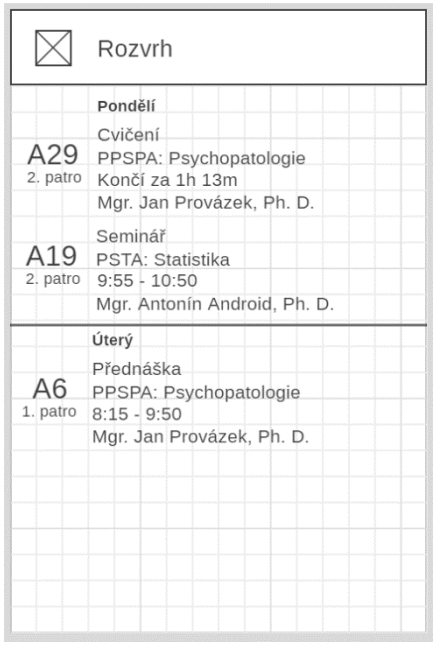


Obrázek 8 wireframe sekce přehledu [autor práce]

[20] Náčrt grafického rozložení

4.4.2 Rozvrh

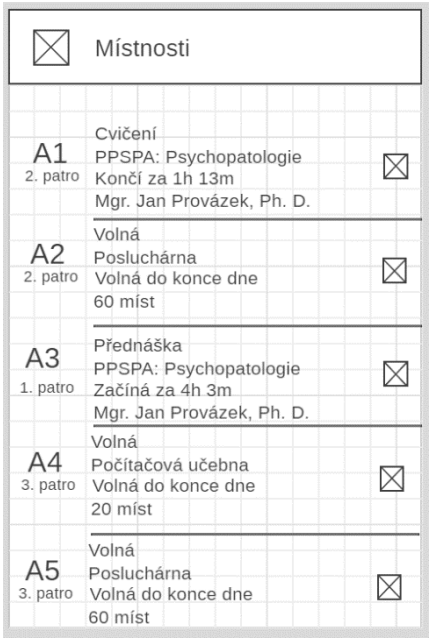
Tato sekce je určena k zobrazení týdenního rozvrhu studenta. Je tedy dostupná pouze pro přihlášené uživatele kvůli nutnosti autorizace pro získání těchto dat. Položky v seznamu jsou seskupeny dle jednotlivých dní v týdnu a obsahují informace o typu rozvrhové akce, učebně, předmětu a času konání. Možným rozšířením tohoto fragmentu je nástroj pro přepínání mezi jednotlivými týdny. V databázi je uložen pouze aktuálně probíhající týden a ostatní jsou na vyžádání staženy ze serveru. Na každém řádku s rozvrhovou akcí se zobrazuje indikátor o průběhu akce společně s výpočtem času zbývajícího do začátku / konce výuky. Uživateli je tímto způsobem umožněna jednodušší interakce a odhad při práci s rozvrhem. Návrh obrazovky je na obrázku 9.



Obrázek 9 wireframe sekce rozvrhu [autor práce]

4.4.3 Učebny

Při prvotním spuštění aplikace je seznam s vybranými učebnami logicky prázdný a prostřednictvím menu je uživatel přesměrován do sekce obstarávající načtení všech budov univerzity a jejich dostupných učeben. V těchto službách jsou stahovány všechny místnosti, jako jsou kanceláře či pracovny akademiků, a proto je obsah filtrován pouze na objekty, ke kterým existují rozvrhové akce. Typicky se jedná o seminární učebny, počítačové učebny, laboratoře, posluchárny aj. Načtená data jsou uložena do vhodných tabulek v DB a uživatel volí, kterou z místností chce zobrazit ve svém seznamu. Po výběru se uživatel vrací zpět na seznam již zvolených místností a pro každou z nich



Obrázek 10 wireframe sekce místností [autor práce]

je rozvrh buď stažen ze serveru, nebo načten z databáze (podle posledních aktualizací hodnot). Každá položka v seznamu mimo informace o učebně obsahuje tedy údaje o právě konané rozvrhové akci nebo o budoucích akcích. Stejně jako v sekci rozvrhu je zobrazen vhodný indikátor a přepočet časového rozmezí. Pro každou učebnu je stahován týdenní rozvrh. Pro jeho zobrazení uživatel klepne na řádek akce. V tu chvíli se zobrazí dialog s výpisem všech rozvrhových akcí na aktuální týden. Nákres rozložení prvků v seznamu místností je zobrazen na obrázku 10.

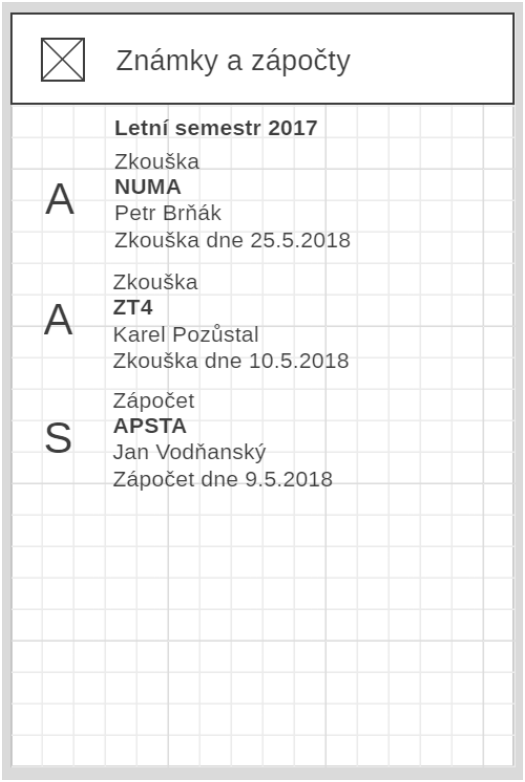
4.4.4 Zápis na termíny

V době zápočtového a zkouškového období jsou organizovány zápočtové/zkouškové termíny. Svou strukturou se podobají rozvrhové akci, ale studentovi je dostupná funkce přihlášení a odhlášení na tento termín. Voláním webové služby jsou vráceny všechny dostupné zkouškové termíny. Zobrazí se i ty, které jsou obsazeny, nebo se na ně uživatel z jiných důvodů nemůže přihlásit. V takové situaci se z API vrací doplňující popis se zdůvodněním nemožného vykonání operace. Jakmile jsou data načtena, jsou seřazena do skupin dle předmětu a typu. S přihlédnutím k možným akcím jsou přizpůsobeny ovládací prvky položek. Díky seskupení totožných informací je pro uživatele obsah přehlednější a lépe se v něm orientuje. Pokud uživatel vykoná interakci s ovládacími prvky, je zobrazen dialog pro potvrzení operace. Po potvrzení je zaslán požadavek akce na server a data jsou znovu načtena.

Obrázek 11 wireframe sekce zápisu na termíny [autor práce]

4.4.5 Známky a zápočty

V neposlední řadě je k dispozici fragment obsahující údaje o dosavadním průběhu studia. Webové služby vrací informace o plnění zapsaných předmětů za celou dobu studovaného oboru. K dispozici jsou údaje o vyučujících, kteří udělili zápočet nebo zkoušku, datum této události a hodnocení. Data jsou seskupena dle předmětů za semestr a seřazena od nejaktuálnějšího semestru do minulosti. Z důvodu ne příliš častého načítání těchto dat nemá aplikace tabulku, která by tyto hodnoty měla k dispozici lokálně, ale při vstupu jsou hodnoty načteny ze serveru. Náčrt rozhraní průběhu studia je na obrázku 12.



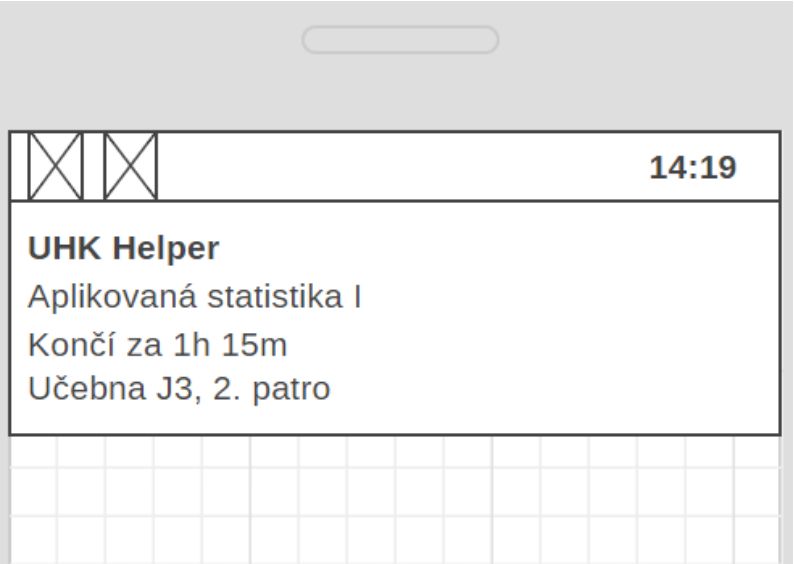
Obrázek 12 wireframe sekce známek a zápočtů [autor práce]

4.4.6 Podpůrné funkce, notifikace

Aplikace obsahuje další rozšiřující funkce, jako je například zobrazení dostupného harmonogramu současného školního roku. Harmonogram zvýrazňuje určitým způsobem nadcházející události. Aplikace rovněž obsahuje odkazy na externí aplikace pro obsluhu a správu jídelny nebo zaslání zpětné vazby pro nahlášení chyby či nápadu na možné modifikace. Mimo tyto funkce aplikace podporuje zobrazování notifikací.

Jedním z druhů notifikací je napojení na nově vznikající newsletter, který bude sloužit vyučujícím a studijnímu oddělení pro efektivnější uvědomění studentů o jejich povinnostech či o možnostech školních aktivit, jako jsou výjezdy, zvané přednášky aj. V ideálním případě by systém mohl být v budoucnu propojen s intranetem univerzity a při přidání nové změny ve výuce vyučujícím by byli všichni studenti notifikováni o této události. Druhým typem notifikace pro studenty

je zobrazení aktuální a následující výuky. Výhodou pro uživatele je to, že je notifikace k dispozici kdykoliv stáhnutím horní lišty v systému a údaje jsou dostupné bez nutného otevírání aplikace nebo umísťování widgetu[21] na domovskou obrazovku. Takový widget je v aplikaci také k dispozici a skrývá údaje pro přihlášeného studenta, jako třeba rozvrh nebo seznam změn ve výuce. Notifikace má však oproti widgetu výhodu v tom, že je zobrazena až ve chvíli, kdy výuka začíná a její obsah je dynamicky měněn při průběhu rozvrhové akce. Možné rozložení takové notifikace je vyobrazeno na obrázku 13.



Obrázek 13 rozložení notifikace [autor práce]

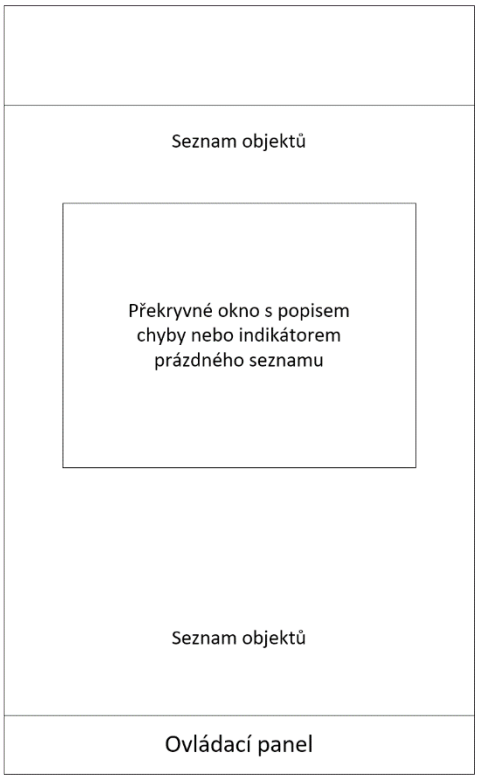
[21] Grafické rozhraní pro ovládání aplikace z domovské obrazovky

4.5 Využití objektového přístupu

Při vývoji jsou dodržena pravidla objektově orientovaného a generického programování. Všechny třídy jsou zapouzdřené, tedy své atributy si spravují samy a s okolním světem komunikuje přes rozhraní volání metod. Tam, kde je to možné, je využit princip polymorfismu a dědičnosti – předek nemusí nutně znát konkrétní implementaci a typy proměnných – stačí, když bude mít možnost volání nutných metod. Takové principy snižují závislosti a také rozsáhlý či duplicitní kód. Stejně tak je možné tvořit datové struktury s minimální závislostí na algoritmech.

V jazyce Java jsou pro takové přístupy dostupné struktury ve formě *interface*[22], *abstraktních tříd*[23] a lze využít také generika pro předem nedefinovaný typ objektu, jehož původ bude znát až samotná instance potomka, která bude mít práci s ním ve své režii.

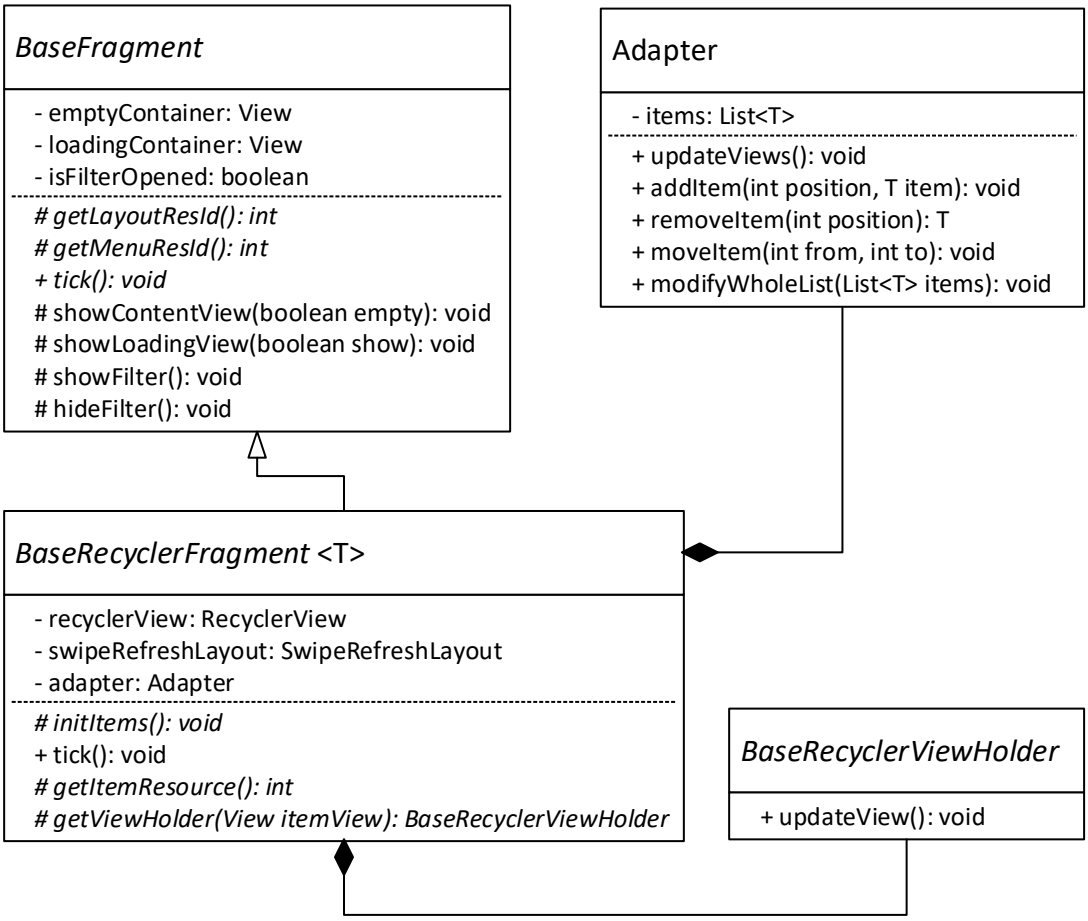
Generalizace nabízených funkcí a obsahu fragmentů aplikace vede k několika zásadním informacím. Každý fragment obsahuje grafické uspořádání prvků (dále jen view) pro zobrazení seznamu. Je-li seznam prázdný, nebo se jeho data načítají, zobrazí takový stav vhodným způsobem na místo prázdné plochy. Nad seznamy může fragment provádět operace filtrování, nebo zprostředkovává interakci s položkami seznamu. Každý fragment mění svůj obsah po jistých časových okamžicích a nabízí nabídku funkcí přes *overflow menu*[24] vyvolatelné z *toolbaru*[25] aplikace. Schéma takové obrazovky je k vidění na obrázku 14.



Obrázek 14 sestavení obrazovky aplikace [autor práce]

[22] Rozhraní – způsob zajišťující předpis chování
[23] Konstrukce sloužící k zobecnění tříd a současnému předpisu chování potomků
[24] Kontextová nabídka překrývající ostatní prvky
[25] Ovládací panel

Všechny tyto ovládací prvky, volání a zpracování mohou být přesunuty do předka. Využitím abstraktních tříd je možné donutit potomky k implementaci metod, které předek pokrýt nedokáže. Každý fragment potomek pak do generika dosazuje instance svých tříd a stará se o načítání dat z DB nebo o jejich aktualizaci z webových služeb. Následující obrázek je model tříd pro zobrazení struktury abstraktních tříd ukrývající metody a struktury, které jsou shodné napříč všemi moduly aplikace.



Obrázek 15 model tříd fragmentů se seznamy [autor práce]

„BaseFragment“ je abstraktní třídou, která zajišťuje vykreslení obrazu dle rozložení daného potomkem. Rovněž přiřazuje tyto instance do atributů. Pro potomky poskytuje metody k zobrazení načítání nebo grafického zpracování pro nedostupná data. Předepisuje metodu „tick“, s jejíž pomocí potomci reagují na změnu časové

události – potomci mohou v tomto případě reagovat změnou vypisovaných hodnot, nebo kompletně změnit zobrazovaný obsah.

Pro zobrazování listu je k dispozici „ListView“ nebo lze využít „RecyclerView“ (dále jen recycler). Recycler se odlišuje tou vlastností, že pro každou položku nevlastní jedno view. Je-li v listu posouváno uživatelem na další položky, pak se pro tyto nově zobrazené nevytváří nové view. Vezme se to, jehož položka se posunem ztratila z dohledu – vyplní jej správnými hodnotami a přesune na správnou pozici v seznamu. Díky tomuto principu je možné mnohem efektivněji a plynuleji reagovat na posun seznamu, aniž by bylo vytěžováno vlákno pro kreslení GUI. Právě tvorba nového view je časově a výkonově poměrně náročná. Na pomalejších zařízeních a při složitější struktuře může docházet k vynechání několika vykreslovaných snímků nebo k pozastavení posouvání listu, nežli je vše připraveno k publikaci.

Každý recycler potřebuje vlastní adaptér, který bude určovat pořadí zobrazovaných položek a je zodpovědný za přidávání nebo odebírání stávajících. V rámci struktury není potřeba adaptér tvořit genericky a jedna třída vystačí díky generikům pro jakékoliv potomky. V „BaseRecylerFragmentu“ je abstraktní metoda „getViewHolder“, pro kterou potomek dosadí vlastní implementaci abstraktní třídy „BaseRecyclerViewHolder“ (dále jen viewHolder). Ten je pouhou schránkou, která doplňuje texty, ikony nebo chování jednotlivých řádků pro každou položku v datech. Adaptér pak v podstatě pouze view odebere od mizející položky od její současného viewHoldera a na jeho místo dosadí viewHoldera nadcházejícího.

„Base fragment“ je kromě zmíněných funkcí dále zodpovědný za vykreslení filtrovacího panelu. Jsou-li přepsány vhodné abstraktní metody podstrčením vlastního pole filtrovacích položek, ve kterých je pole definováno, jedná-li se o zaškrtávací pole nebo posouvací panel, pak tento spodní panel předek vykreslí a přiřadí panelu definované reakce na události. Příkladem je filtr v sekci učeben, který obsahuje zaškrtávací pole pro zobrazování pouze volných učeben, nebo pouze učeben specifického typu. Po zaškrtnutí pole dojde k filtraci položek seznamu pouze na vybrané učebny. Při posunu v seznamu směrem ke konci se panel s animací schová pod displej, aby uživateli nepřekážel při průchodu seznamem. Je vysunut při posunu směrem nahoru.

5 Lokalizace

Kapitola lokalizace pojednává o principu využití naměřených hodnot ze senzorů k detekci lokálního pohybu. Před konečným řešením je nastíněna funkce akcelerometru v praxi a využití jeho měření k výpočtu přesně uražené vzdálenosti. Dále je zde uvedeno, proč je tento způsob vhodný pouze v určitých scénářích pohybu. Popsána je také účast senzoru okolního tlaku při lokalizaci a závěrem se uvádějí možnosti a metody zpřesňování celého procesu.

5.1 Úvod do užití senzorů pro lokalizaci

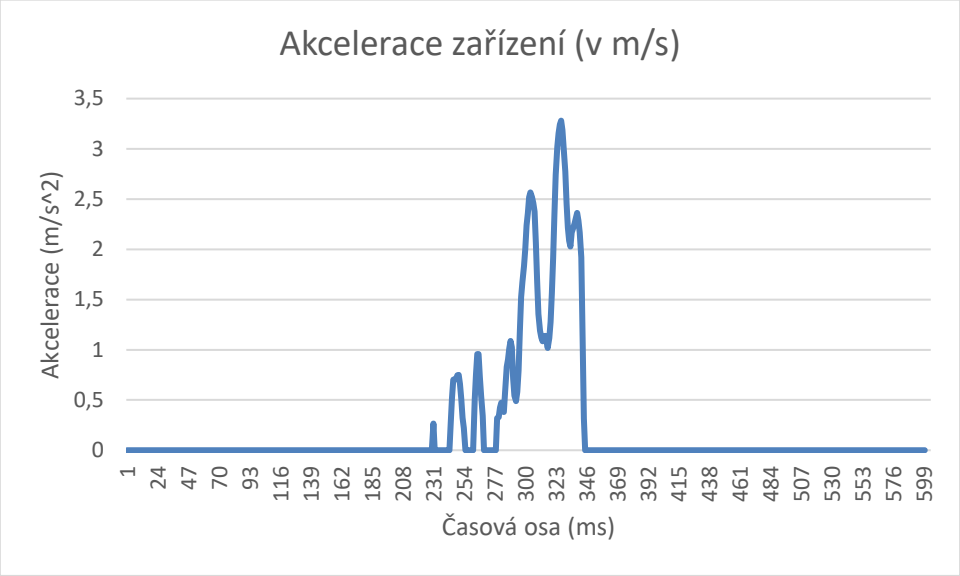
Odezva měření akcelerometru není vždy stejná. Zařízení někdy vrací hodnoty zrychlení zařízení každé 2 milisekundy, ale zpravidla je časový interval o něco delší. Spolu s informací o intenzitě akcelerace je k dispozici časové razítko v nanosekundách. Jsou-li známy hodnoty akcelerace v daný časový okamžik a údaj o aktuální rychlosti, je možné spočítat novou rychlost pro daný směr v prostoru. Uvažujme obecnou rovnici výpočtu dráhy ze znalosti rychlosti „v“ a zrychlení „a“ za čas „t“.

$$s = \mathrm{v.t} + \frac{1}{2}.a.t^2$$

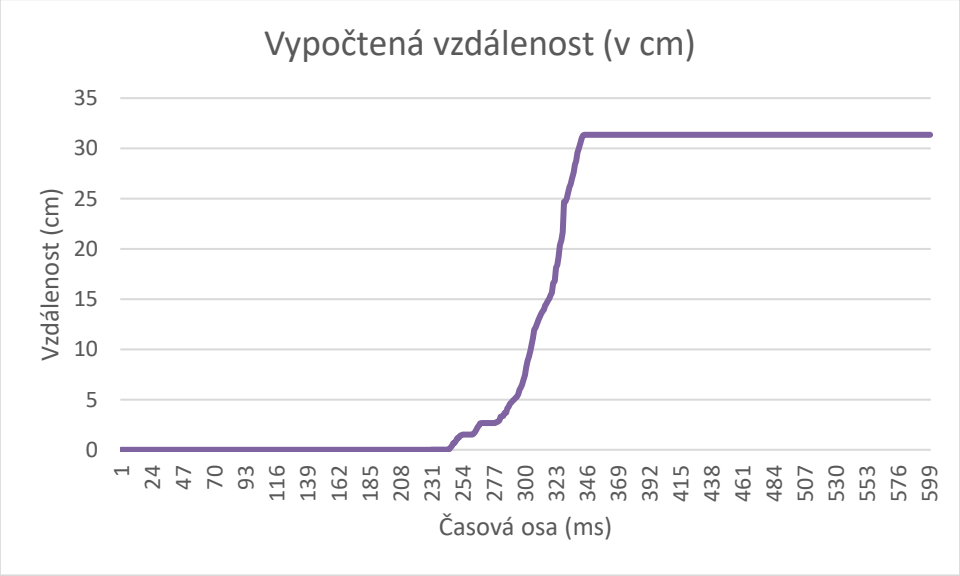
Hodnoty proměnné „a“ jsou nám známy stejně jako „t“, které lze dopočítat z časového razítka v době naměřené hodnoty. Je-li výpočet zahájen v době, kdy je počáteční rychlost nulová, jelikož uživatel stojí na místě, pak je možné vypočítat kompletní rychlost celého pohybu v průběhu etap. Do dalších úseků se do počáteční rychlosti dosazuje poslední vypočítaná hodnota a opět se výpočtem rovnice určuje rychlost následující. Násobením rychlostí těchto úseků s jejich dobou trvání je možné dopočítat celkovou uraženou vzdálenost.

Tato metoda by byla přesná, pokud by akcelerometr vracel velmi přesné hodnoty ve vysoké odezvě a vykonávaný pohyb by byl přímočarý. Když pomineme poměrně náročný výpočet a paměťovou náročnost polí s hodnotami, pak lze za těchto ideálních podmínek dojít k velmi přesným hodnotám. Při implementaci tohoto principu byl proveden pokus s měřením jedné z os zařízení a vykonání přímočarého pohybu po desce stolu s jasně definovaným časovým intervalem. Chyba, která

vznikla, byla v řádu několika centimetrů na dráze přibližně jednoho metru. Důvodem těchto chyb je zpravidla nižší časová odezva a nepřesnosti nebo šum, který se v signálu z akcelerometru objevuje. V následujících grafech 1 a 2 je zobrazeno měření akcelerace v zařízení a přepočet na vzdálenost. Vykonaný pohyb byl svou délkou okolo 30 cm.



Graf 1 akcelerace zařízení při přímočarém pohybu [autor práce]

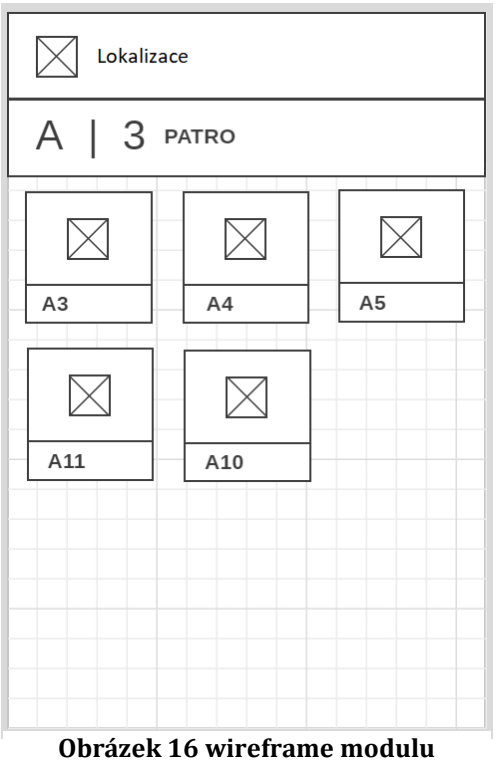


Graf 2 přepočet akcelerace přímočarého pohybu na vzdálenost [autor práce]

Pohyb uživatele však není nikdy přímočarý a na scéně se objevuje logický problém. Akcelerometr je ovlivněn odstředivou silou vzniklou z rotačních pohybů, které vyvolává držení zařízení v ruce během pohybu. Navíc je zařízení v různých pozicích a hodnoty mezi osami je nutno interpolovat dle hodnot gravitace, které jsou vykonávaným pohybem také mírně ovlivňovány. Výpočet dráhy pohybu se v těchto případech od té skutečné může lišit i o několik metrů během několika kroků. Při vykonání prudkých rotačních pohybů (vyvolání velkého množství odstředivé energie) jsou chyby v řádech i desítek metrů. Výpočet pohybu tímto způsobem je velmi nepřesný a není použitelný pro potřeby indoor navigace. O podobných závěrech je vedena zmínka i v podobných výzkumech. [22]

Akcelerometr lze pro zjištění pohybu využít i jiným způsobem. Získané hodnoty zrychlení jsou pro svou chybovost užívány ve zcela odlišných situacích – existuje mnoho softwarů zabývajících se například detekcí gest uživatele, jako jsou například zvednutí sluchátka k uchu, detekce chůze či zjištění orientace zařízení na šířku nebo na výšku. To vše jsou operace, které nepracují s přesnými hodnotami, ale spíše s intenzitami nebo opakujícím se vzorem. Vzory bývají často zpracovány mechanismy na principu neuronových sítí nebo jakýmkoliv jiným druhem strojového učení, který pomůže danou událost odhalit.

Právě detekce kroku je záležitostí, která může být v indoor lokalizaci nápomocna. Je-li se zařízením manipulováno standardně, zatímco je aplikace spuštěna, pak může být krok považován za jakousi epizodu, během níž se uživatel posunul o určitý úsek v prostoru. Je-li znám vektor takového pohybu, může být údaj přepočten na reálné souřadnice. V aplikaci se pak taková změna projeví jako zobrazení nejbližších místností, které se nacházejí v bezprostředním okolí uživatele. Jednou z možných variant je řazení seznamu místností v reálném čase společně s pohybem uživatele.



Obrázek 16 wireframe modulu lokalizace [autor práce]

5.2 Postup zjišťování pozice a pohybu

Firemní proces práce s fragmentem lokalizace je realizován následovně: Po otevření fragmentu jsou připojeny všechny dostupné senzory pro sběr dat a začíná zároveň jejich analýza a zpracování. V tutéž chvíli dochází k nalezení polohy pomocí GPS za účelem zjištění budovy, ve které se uživatel právě nachází. Je-li určena budova, následuje krok detekce aktuálního patra. Takovou informaci lze získat z nadmořské výšky, ve které se zařízení nachází. Některá ze zařízení mohou obsahovat senzor pro určení atmosférického tlaku, ke kterému zřizuje Android API přístup. Přesný postup získání nadmořské výšky z naměřených hodnot je popsán v následující kapitole 5.2.1.

Když proces přejde předchozími fázemi, pak jsou pro každé patro načteny virtuální reprezentace rozložení objektů v nich. V prvotním určení patra není známa pozice uživatele a vyčkává se na vykonaný pohyb. Klíčem k odhalení pozice uživatele je postup ve světových souřadnicích, díky němuž mohou být logicky vyřazeny ty pozice, u kterých by v důsledku posunu došlo k nesmyslnému vymknutí se z prostor budovy. Jinými slovy – čím déle bude uživatel chodit v prostorách budovy, tím přesnější budou výsledky filtrace okolních objektů na chodbách. Možnou indicií pro startovní bod je změna atmosférického tlaku a tím i aktuálního patra – taková událost může nastat pouze v místech, kde je tomu budova přizpůsobena – výtah, schodiště.

V případě, že je GPS pozice vypnuta, nebo zařízení nedisponuje senzorem tlaku, je pak grafické rozhraní fragmentu přizpůsobeno možnostem manuálního výběru budovy a jejího patra skrze ovládací prvky ve formuláři. Pokud dojde ke změně patra uživatelem, pak tato událost není zahrnuta pro potřeby zpřesnění pozice uživatele.

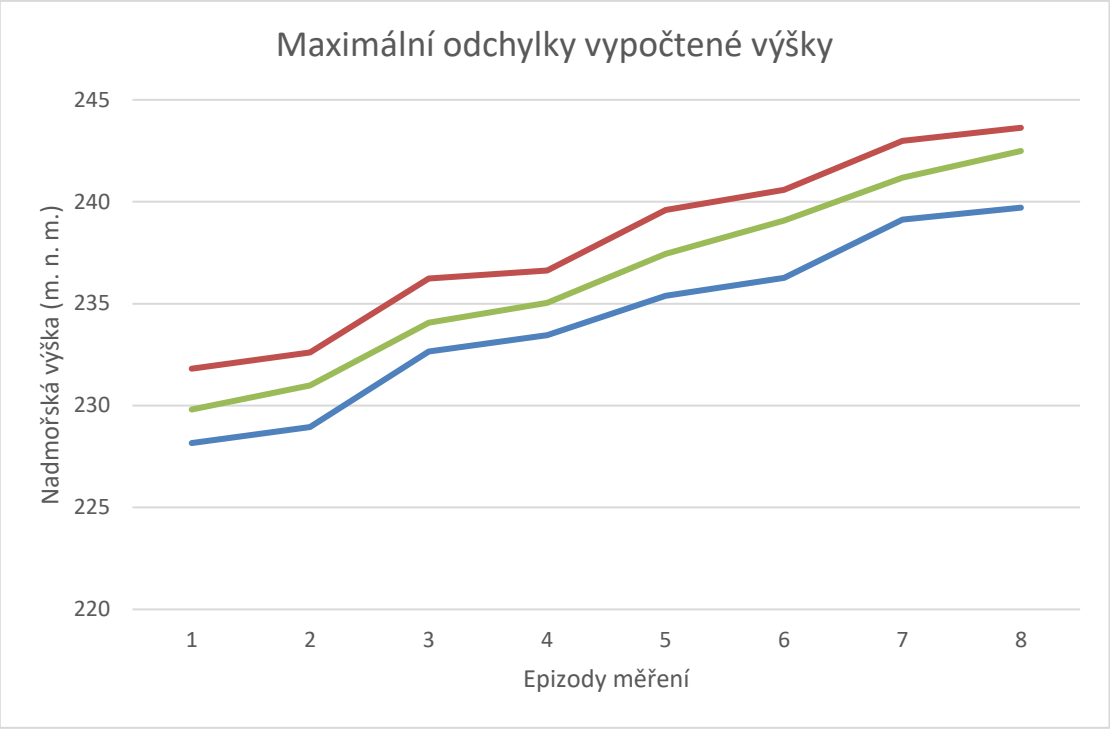
5.2.1 Detekce pater

Je několik způsobů výpočtů nadmořské výšky dle aktuálního měřeného atmosférického tlaku. Existují rovnice, které zohledňují vlastnosti prostředí, jakými jsou třeba teplota, vlhkost vzduchu či rychlost větru. Tyto atributy však nejsou k dispozici a nemusela by být ověřena jejich správnost. Minimálním kritériem, které je při výpočtu nutné, je hodnota atmosférického tlaku daného místa přepočtena na

hladinu moře. V dostupné třídě „SensorManager“ od Google je implementována funkce s následující rovnicí výpočtu. [23]

$$h = 44330 \,.\, (1 - \frac{\mathrm{p}}{\mathrm{p0}})^{\frac{1}{5{,}255}}$$

Za hodnoty „p“ je dosazena aktuální hodnota tlaku v jednotkách hPa a za hodnoty „p0“ aktuální tlak přepočtený na hladinu moře. Právě tato hodnota velmi ovlivňuje výslednou vypočtenou nadmořskou výšku zařízení, a proto musí být určena co nejpřesněji. V dokumentaci je doporučeno využít hodnoty dostupné online z důvěryhodných meteorologických stanic institucí či blízkých letišť. V Hradci Králové jsou pak dvě varianty meteorologických stanic, a to z ulice na Pražském předměstí nebo na několik kilometrů vzdáleném letišti severovýchodně od města.
Po dosazení hodnot je s vysokou přesností určena předpokládaná výška, ve které se zařízení nachází, a dle aktuálně navštívené budovy je určeno poschodí, k jehož průměrné nadmořské výšce je číslo nejbližší. Hodnoty výšky poschodí jsou naměřeny s přihlédnutím k dostupným informacím o nadmořské výšce základů budov. Hodnoty jsou zaznamenány při používání zařízení během chůze po chodbách. Z těchto hodnot je aritmetickým průměrem určena prahová hodnota.
Meteorologické stanice svůj aktuální tlak mohou zobrazit s nesprávným přepočtem na tlak na hladině moře, nebo nemusí být jejich hodnoty zcela aktuální, nebo je v okolí odlišný tlak než v okolí budovy a zařízení. V důsledku těchto malých odchylek může dojít k driftu ve výpočtu až o 2-3 metry. V následujícím grafu 3 je zaznamenána vypočtená výška pro 3 měření provedená v odlišný den při různé teplotě a tlaku. Byly vybrány nejextrémnější vypočtené hodnoty (červená, modrá) a hodnoty odpovídající realitě (zelená). Měření obsahuje 8 epizod, kde první je naměřená výška v prvním patře na podlaze, druhá značí první patro se zařízením v úrovni pasu, třetí druhé patro u podlahy atd. až do 4. patra z budovy J.



Graf 3 maximální odchylky vypočítaných výšek na budově J [autor práce]

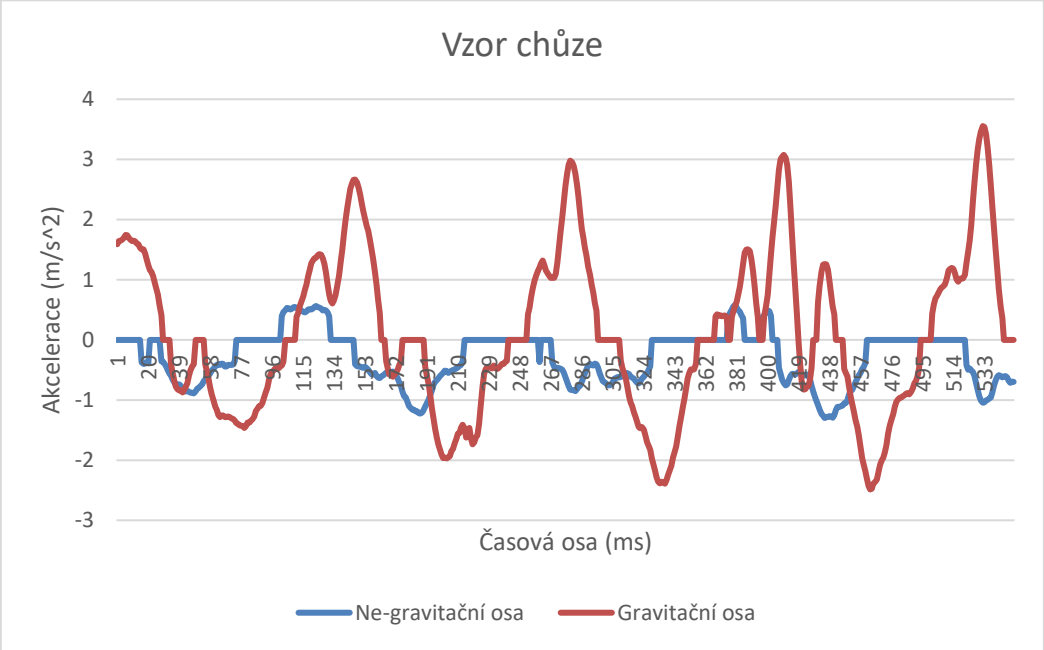
V grafu je možné pozorovat, že samotný pohyb mezi patry je určen pro potřeby detekce změny poschodí dostatečně správně, ale kvůli inicializační chybě může být patro zjištěno o jedno nesprávně v obou směrech. Pokud se tak stane, pak je uživateli umožněno přepnutí na aktuální podlaží. Aplikace se tím o driftu dozví a bere jej v daný časový interval v potaz při příštích výpočtech podlaží.

5.2.2 Detekce kroku

Android API poskytuje senzor pod označením „STEP_DETECTOR“, který dle dokumentace vrací vyvolanou událost do okamžiku, kdy je detekován krok. Pro potřeby aplikace se však toto řešení ukázalo jako nevhodné z toho důvodu, že je kvůli implementaci brána v potaz eliminace nechtěných pohybů, a proto jsou události vyvolány přibližně až po 5 ušlých krocích se zařízením. Po zastavení je událost ještě pětkrát vyvolána. To znamená, že pokud uživatel udělá 3 kroky na jednu stranu, zastaví se a po chvíli znovu rozejde, pak by se takový pohyb ztratil, navíc by nebyl synchronizován s pomocnými hodnotami akcelerometru, gyroskopu a především magnetometru, které jsou ve výpočtu také využívány.

Pro efektivní výpočet pohybu je tedy implementován vlastní jednoduchý detektor kroku.

Graf 4 obsahuje zobrazení hodnot *gravitační osy*[26] akcelerometru (červená) a hodnoty ne-gravitační osy[27], která měří pohyb vpřed a vzad (modrá). Zde je možné pozorovat opakující se vzor vyvolaný chůzí lidského těla při držení zařízení v ruce.



Graf 4 vzor chůze z hodnot akcelerometru [autor práce]

[26] Osa, na kterou, oproti ostatním, působí nejvyšší míra gravitačního zrychlení
[27] Jedna z os, jejichž vektorový součin se limitně blíží k směrovému vektoru osy gravitační

Pro získání epizody jednoho kroku jsou výpočty přizpůsobeny tak, aby tento vzor odhalily a na jeho konci tuto skutečnost oznámily metodám, které naměřené hodnoty dále zpracovávají. Hlavním indikátorem jsou výkyvy do vysokých kladných a následně záporných hodnot na ose měřící hodnoty směrem ke gravitačnímu poli. Způsob získávání přesného směrového vektoru i při různém používání zařízení uživatelem je popsáno v následující kapitole 5.2.3.

Při detekci odpovídajícího pohybu, který se podobá kroku jsou zohledněny věci, které slouží k eliminaci nechtěného chování. Jedním z faktorů pro odstranění chyby je časové razítko. Lidský krok může být vykonáván standardně pouze v jistých časových intervalech, které přesahují spodní prahovou hodnotu. Při určení této hodnoty pak bylo provedeno několik měření rychlé chůze. Nejnižší evidované hodnoty byly použity jako práh, při jehož nedosažení jsou výpočty zastaveny, jelikož vykonaný pohyb pravděpodobně nebyl krokem. Druhým faktorem je sledování chování na ne-gravitačních osách. Je-li na ose, která směřuje před uživatele, zatímco je vykonávána chůze, znatelná jistá intenzita, pak může být evidována nová epizoda pro výpočet. Pokud se zařízením pouze pohybuje na místě (pokud uživatel pouze přešlapuje), pak tento pohyb není brán v potaz. Samozřejmě existují druhy nestandardních pohybů, kterými lze mechanismy ošálit. Systém tyto chyby správně eliminuje při standardním pohybu uživatele po chodbách budovy.

Kvůli šumu, který vzniká na senzorech, jsou hodnoty ořezávány o prahové hodnoty, bez čehož by algoritmus častěji vyhodnocoval možné epizody pohybu. Prahové hodnoty byly stanoveny tak, že bylo zařízení měřeno na všech osách, jestliže bylo zařízení drženo v ruce bez větších pohybů. Maximální hodnoty, které v podstatě neměly být nijak brány v potaz, jsou ořezávány. Některé senzory by však i přes tyto korekce mohly vracet špatné výsledky – v takovém případě lze implementovat jiné principy, viz kapitola 6.3. Další vlastností je pozastavení výpočtu v situaci, kdy je zařízení na jednom nebo na druhém boku. Právě to, že aplikace může být vykreslena na výšku nebo na obě strany, by nám znemožnilo určit, kterým směrem se zařízení pohybuje, jelikož nelze z důvodu akce a reakce spoléhat na určení tohoto počátečního pohybu z ne-gravitačních os senzoru.

5.2.3 Směrový vektor a epizody kroku

Při schválení epizody nového kroku se tedy předpokládá standardní chování uživatele při chůzi, přičemž je jeho zařízení s otevřenou aplikací a seznamem okolních objektů používáno. Avšak i v této situaci se nelze vyhnout komplikacím, které toto užívání zařízení uživateli způsobuje. Zařízení s vysokou pravděpodobností nikdo nepoužívá tak, aby bylo jednou ze stran stoprocentně namířeno k zemi, ale pravděpodobně ho drží volně – to znamená, že gravitačních zrychlení a hodnoty akcelerace jsou v jistém poměru rozloženy na osách y a z. Během kroku může dokonce docházet i ke změnám poměru třeba podle toho, jestli uživatel zařízení drží a dívá se před sebe, nebo hledí přímo na obrazovku – v dané chvíli se bude lišit úhel k zemi.

Pro tento případ je společně naslouchán hodnotám ze senzoru „LINEAR_ACCELEROMETER“ (zaznamenává pohyb vyvolaný pohybem zařízení – bez gravitačního zrychlení) a „GRAVITY“ (reaguje pouze na změny v gravitačním zrychlení). Oproti senzoru „ACCELEROMETER“ je výhoda v tom, že měřené hodnoty jsou odlišeny dle původu vykonání a v podstatě je jejich součtem docíleno hodnoty, vracející se z tohoto jednoho senzoru.

Když je k dispozici hodnota všech os a jejich gravitační zrychlení, pak je možné tyto hodnoty *normalizovat*[28] a sestavit z nich vektor, který bude dále využit pro interpolaci během výpočtu. Pro všechny hodnoty naměřené od počátku epizody kroku až do jejího konce se tímto způsobem sestaví jednotlivé gravitační vektory, prozrazující míru naklopení zařízení během pohybu. Díky předpokladu užívání zařízení v režimu na výšku je znám i vektor, kterým se řídí směr uživatele.

Pro každý takto sestavený vektor jsou z měření uloženy hodnoty ze senzoru „TYPE_ROTATION_VECTOR“, který obsahuje hodnoty nastavení zařízení vůči severnímu pólu z kvaternionu získaného pomocí magnetometru v daný moment. Vynásobením směru pohybu akcelerometru zařízení kvaternionem dostáváme vektor přepočtený do prostoru reálného světa. *(Osy zařízení a světa jsou na obrázkách 1 a 2.)*

[28] Podíl každé hodnoty z dimenze vektoru součtem všech těchto hodnot

Pro všechny intervaly mezi naměřenými hodnotami je tímto způsobem sestaven řetězec spojených mikropohybů, na jehož konci je jeho průběh využit opět pro interpolaci při mapování na určitou jednotku délky. Hodnotou, která určuje míru intenzity každého mikropohybu, je časové razítko současného a následujícího měření. Rozdíl těchto dvou hodnot tvoří určité procento z celkové délky intervalu epizody a vektorům jsou dávány váhy ve správném poměru. Tím je průběh výpočtu uzavřen a do fragmentu jsou zaslány údaje o uražené vzdálenosti ve světových souřadnicích, který na tyto změny reaguje. Určení délky kroku a eliminace chyb je popsána detailně v kapitole 5.3.

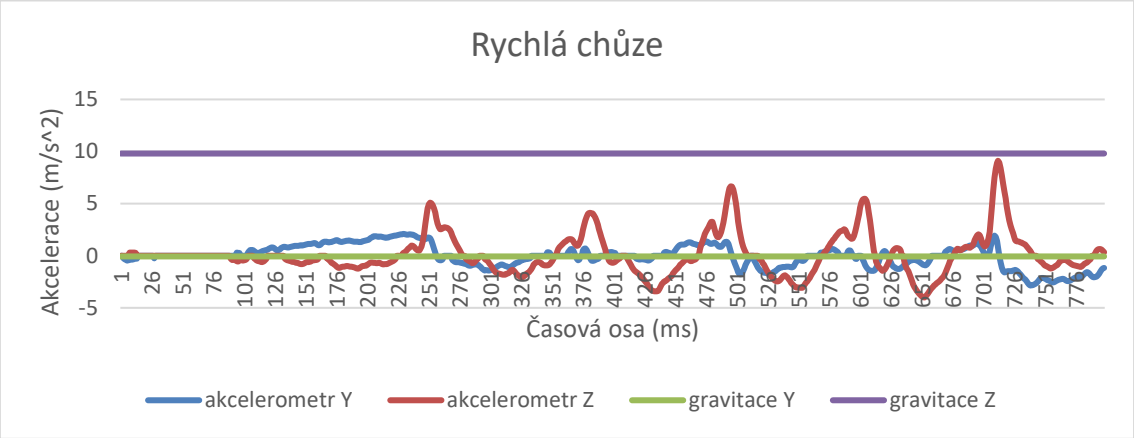
Opakováním těchto výpočtů pro určování pozice je rovněž uchovávána celková trasa vykonávaná uživatelem a je v podstatě tvořena mapa pohybu, která může být zohledňována při pozdějším rozhodování.

5.3 Zpřesňování pozice a eliminace chyb

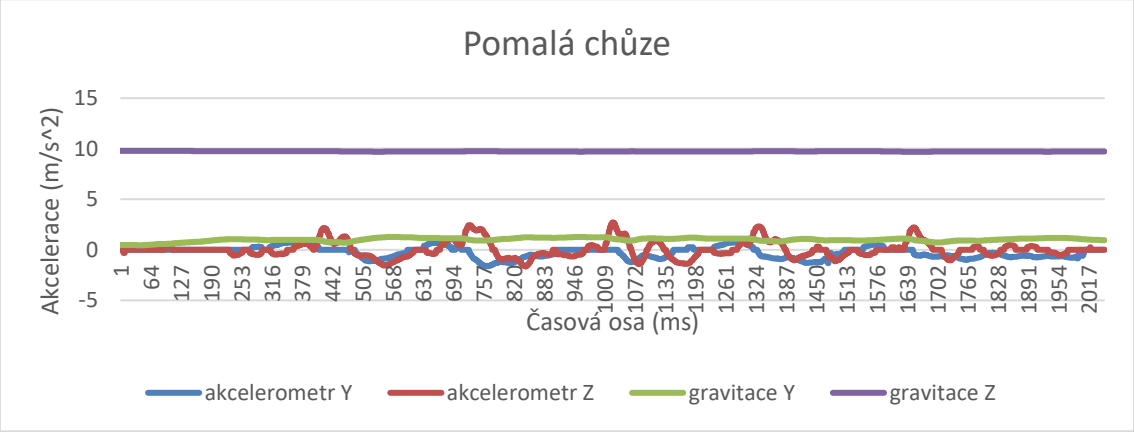
Určit správnou pozici pouze za pomocí detekce kroku nemůže být ani za ideálních podmínek nikde dostatečně přesné. Častým způsobem řešení je zadávání výšky osoby, která zařízení používá, a dle stanovených tabulek průměrných kroků na výšku uživatele je taková hodnota brána v potaz. To není špatný způsob z hlediska např. určování dodatečných vlastností provedených aktivit nebo orientační uražené vzdálenosti. V případech, kdy je nutné co nejpřesněji určit vzdálenost, je zapotřebí provést další analýzu a využít ji k zpřesňování výsledků.

Základním nezbytně nutným zpřesněním je úprava délky kroku na základě vykonávaného pohybu. Krok dospělého člověka se natahuje a zkracuje o desítky centimetrů stylem chůze a jeho rychlostí. Je-li porovnána délka kroku z pomalé vycházkové chůze (~2-3 km/h) s délkou kroku z chůze rychlé (~6-7 km/h), pak je rozdíl natolik znatelný, že by mohl během několika málo kroků vytvořit chybu v řádu metrů.

Při bližším zkoumání vlastností pomalé a dynamické chůze jsou evidentní intenzita výkyvů na jednotlivých osách akcelerometru. Rychlejší chůze zpravidla značí dlouhé kroky, při nichž logicky dochází k větším propadům středu těla, než když jsou délky mezi chodidly kratší. Na grafech 5 a 6 je zobrazeno chování za rozdílných podmínek.



Graf 5 hodnoty akcelerometru za pomalé chůze [autor práce]



Graf 6 hodnoty akcelerometru za rychlé chůze [autor práce]

Z hodnot v grafu je patrná možnost užití vlastností intenzity zrychlení pro zpřesnění délky kroku, která bude pro směrový vektor přiřazena ve výpočtu. Hodnoty, které bývají délkou kroku při rychlosti chůze vychýleny, jsou ty z akcelerometru na ose gravitace, tedy akcelerometr Z (červená).

Testování vlivu hodnot akcelerometru bylo provedeno na měření nezávislých osob s rozdílnými výškami od 165 do 187 cm. Předmětem měření byly délky kroku za pomalé, standardní a rychlé chůze. Hodnoty byly dále přepracovány a pro každou situaci byly určeny průměry intenzit a délek.

Na základě těchto informací je možné pokusit se o sestavení aproximačního polynomu pro měřené intenzity. Díky užití takového polynomu při výpočtu délky uražené vzdálenosti mohou být hodnoty různých intenzit po dosazení aproximovány na hodnoty délky kroků blížící se realitě. Tím se zpřesňuje uražená vzdálenost, a tedy i celkové určování pozice uživatele. Jedním z možných způsobů je vytvoření polynomu s principy užití metody nejmenších čtverců.

5.3.1 Aproximační polynom

Metoda nejmenších čtverců označuje postup pro přibližné řešení neurčených nebo nepřesně zadaných soustav rovnic, založený na minimalizaci kvadrátů jejich reziduí. Mezi významnou skupinu úloh, které lze metodou nejmenších čtverců řešit, je prokládání dat křivkami – nechť *t* je nezávisle proměnná, například čas, a *y(t)* je neznámá funkce proměnné *t*, kterou chceme aproximovat. Předpokladem pro tento výpočet je provedené pozorování (měření), které nám hodnoty proměnné i její funkce zajistí. Záměrem je modelovat *y(t)* lineárních kombinací tak, aby řešení minimalizovalo součet čtverců odchylek vůči rovnici. [24]
Aproximovat hodnoty lze pomocí polynomu *n-tého* stupně s předpisem:

$$f(x) = a_0 + a_1.x^1 + a_2.x^2 + \cdots + a_n.x^n$$

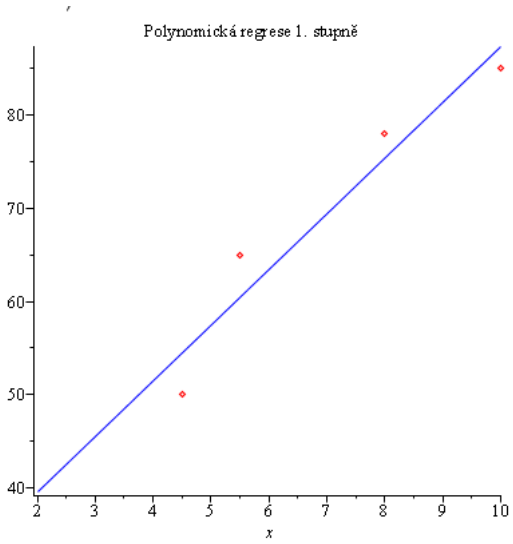
Stupeň polynomu lze volit jakýkoliv – vhodným způsobem je vytvořit si aproximaci pomocí polynomu několika stupňů a výsledky porovnat mezi sebou za účelem zjištění nejvhodnější varianty. Pro zachování úspory při výpočtech je proveden test s aproximací pomocí přímky a aproximací parabolou. Naměřené hodnoty rozptylů akcelerometru (X v m/s) a délky kroků jednotlivých osob (Y v cm) byly seřazeny a zprůměrovány do následujících hodnot obsažené v tabulce 1.

X	4.5	5.5	8	10
Y	50	65	78	85

Tabulka 1 hodnoty pro aproximační polynom [autor práce]

Užitím této numerické metody při dosazení naměřených průměrných intenzit do výpočtu polynomu bylo dosaženo následujícího výsledku. Výpočet polynomu 1. i 2. stupně byl proveden užitím systému Maple[29]. Následující rovnice vystihuje polynom 1. stupně a dále graf 7 tuto funkci zobrazuje společně s body z tabulky 1. Vodorovná osa grafu vyjadřuje míru intenzity rozptylu akcelerometru, horizontální osa délku v centimetrech.

$$y = 5{,}97.x + 27.68$$



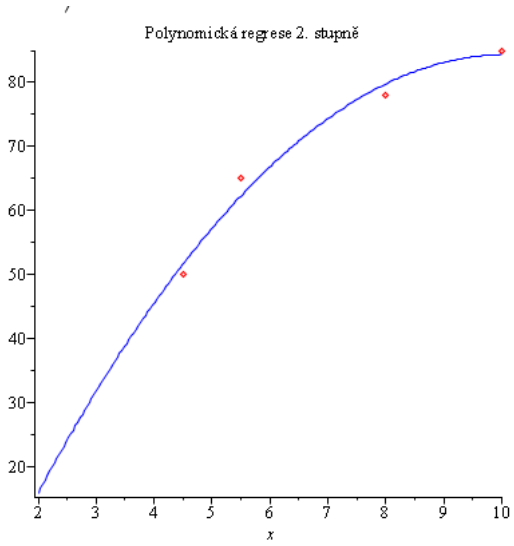
Graf 7 funkce aproximačního polynomu prvního stupně [autor práce]

Užití polynomu prvního stupně se do jisté míry přibližuje měřeným bodům, ale z jejich rozložení je možné pozorovat jejich konkávní zakřivení v průběhu pomyslné funkce. Proto je testován také polynom druhého stupně, který by měl být pro určení vzdáleností dostatečně přesný při současném šetření výpočetního výkonu zařízení. Následující rovnice vyjadřuje vypočítaný polynom 2. stupně a níže uvedený graf 8 zobrazuje průběh této funkce společně s body z tabulky 1. Vodorovná osa grafu

[29] Matematický software pro oblast školství a výzkumu od společnosti Czech Software First, s. r. o.

vyjadřuje míru intenzity rozptylu akcelerometru a horizontální osa délku v centimetrech.

$$y = -1{,}04.x^2 + 21{,}08.x - 22$$

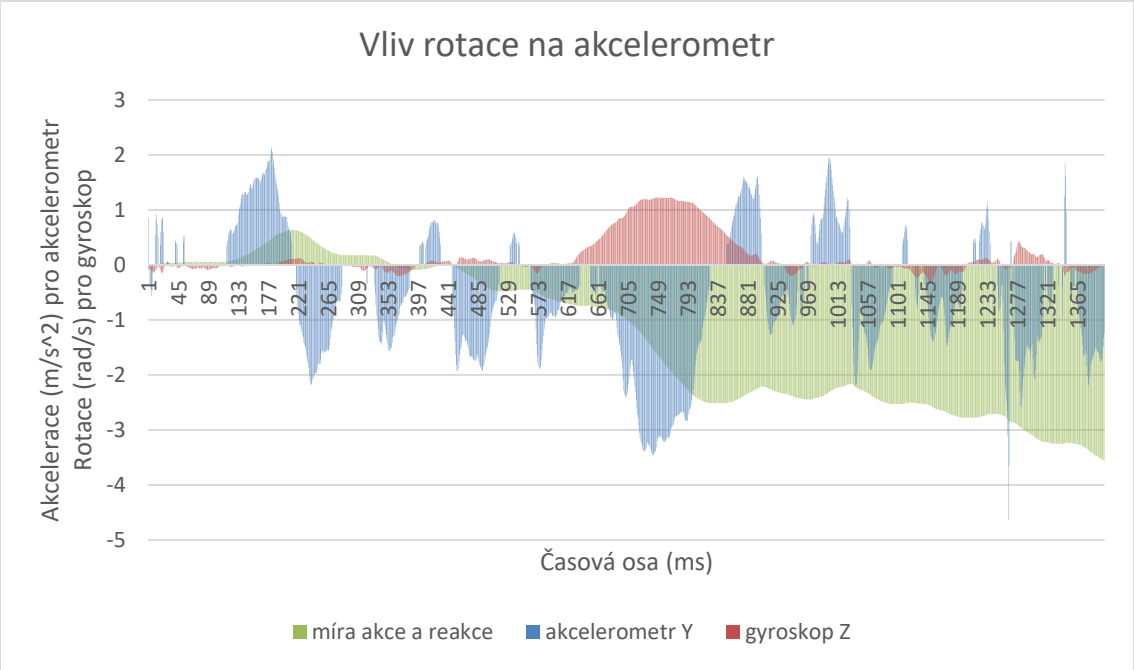


Graf 8 funkce aproximačního polynomu druhého stupně [autor práce]

V kapitole 6.2 jsou popsány výsledky využití tohoto polynomu při výpočtu v praxi.

5.3.2 Odstředivá síla

V předchozí kapitole již byl zmíněn vliv odstředivé síly na hodnoty akcelerometru, která negativně ovlivňuje výpočet uražených vzdáleností nebo zapříčiňují nechtěné volání výpočtu epizody kroku. Ve většině případů je tato síla vyvolána při rotaci uživatele, při zatáčení na chodbách apod. Díky tomu, že se jedná o rotace, lze do výpočtu zahrnout gyroskop, který míru rotace velmi přesně měří. Na následujícím grafu 9 je zobrazen vliv rotace na akcelerometr (modrá) a gyroskop (červená). Graf je doplněn o míru akce a reakce (zelená), která určuje, jak jsou síly akce a reakce při chůzi vyrovnané.



Graf 9 vliv pohybu rotace na hodnoty senzorů [autor práce]

Modré hodnoty představují změny sil vpřed a vzad během chůze. Na začátku chůze, když uživatel zahájí pohyb vpřed, je míra vyvolané akce vpřed a následné reakce při zpomalení vyšší než v následujících třech krocích. Nicméně i tato ne-gravitační osa tvoří opakující se vzor pohybu, který by byl pro detekci kroku použitelný.

Zlom přichází v polovině grafu, kde červené hodnoty, vyjadřující rotaci zařízení na gravitační ose, rostou do vysokých hodnot. Je to v důsledku zatočení o 90° na chodbě budovy během současně vykonávané chůze. Vzor modrých hodnot akcelerometru, který byl do současné chvíle pravidelný, je v důsledku této události ovlivněn odstředivou silou do takové míry, že znemožňuje detekci počtu kroků v průběhu rotace. Po ukončení pohybu se hodnoty opět vrací do stejného vzoru.

Míra akce a reakce (zelená linie v grafu) zobrazuje poměr intenzity akce a reakce během měření. Ideálně by se hodnoty této linie měly limitně blížit nule, což by znamenalo kompletní informace o akceleraci a deceleraci. Tím by se přispělo ke správnému výpočtu rychlostí a délek uražené vzdálenosti. V důsledku rotace je však patrný vysoký nárůst nepoměru měřených hodnot, a dochází tak k chybám ve výpočtech.

Vysoké hodnoty snižují podíl ovlivněné osy na výpočtu pohybu. Tento princip zamezuje chybné volání kroků například v případě, kdy se uživatel otáčí se zařízením na jednom místě.

6 Výsledek práce

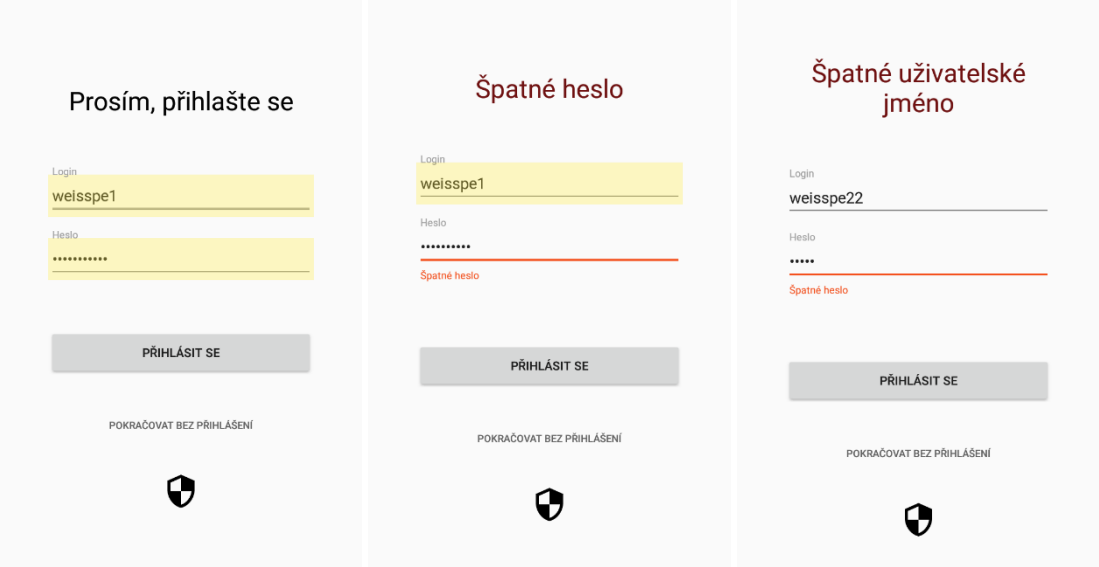
Aplikace byla vyvíjena v IDE Android Studio s použitím dříve uvedených technologií. Verzování projektu s užitím Gitu bylo provedeno na serveru GitHub od společnosti GitHub Inc. Aplikace byla vyvíjena a testována primárně na zařízeních Sony Xperia X Compact, Sony Xperia XZ1 Compact a Samsung Galaxy Tab A (2016). Aplikace obsahuje 3 jazyky – češtinu, slovenštinu a angličtinu, která je volena dle systémového jazyka zařízení, jinak je primární angličtina.

Přístup k funkcím se děje prvotním zadáním uživatelského jména a hesla, které jsou po ověření platnosti uloženy do paměti aplikace a jsou využívány pro všechna webová volání. Služby, které nepotřebují autorizaci uživatele, jsou zpřístupněny v omezené verzi bez rozvrhu nebo možnosti vyhledávání. Tento výběr probíhá na úvodní obrazovce s přihlašovacími údaji. Pokud není uživatel přihlášen, pak je aplikace zabarvena do šedé barvy, jinak je výběr barvy založen na nejbližší hodnotě z barev „material designu“ ke grafickému manuálu jednotlivých fakult univerzity.

V následující kapitole 6.1 jsou výsledky zobrazení aplikace zobrazeny na obrázcích s komentáři o funkcích jednotlivých grafických prvků.

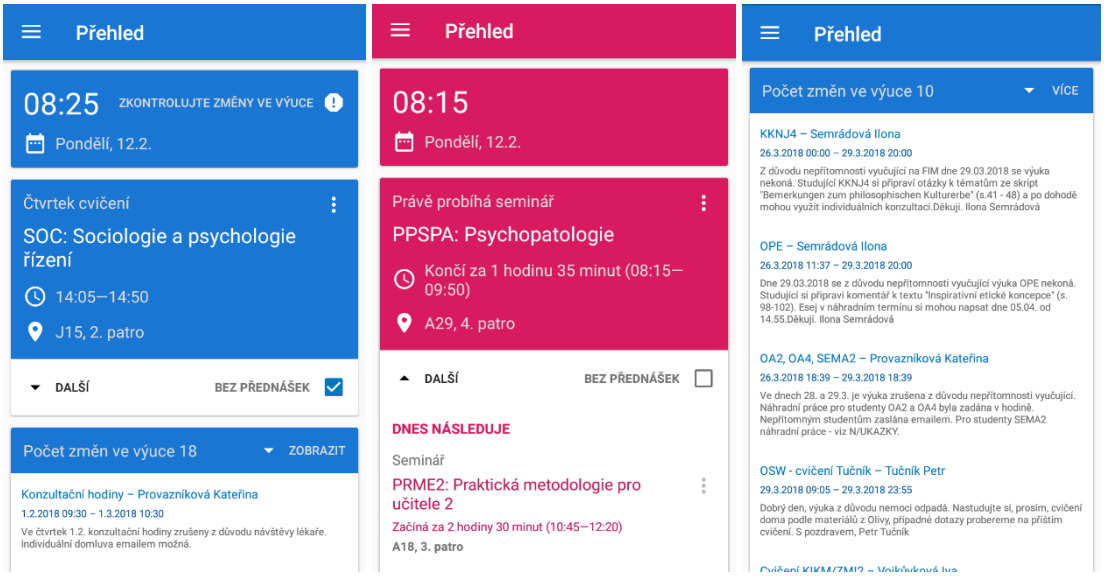
6.1 Výsledky funkce pro podporu výuky

V kontextové nabídce nebo při vypršení platnosti hesla je uživatel převeden na úvodní obrazovku s prvky pro zadání přihlašovacích údajů. V rámci této obrazovky je po potvrzení údajů uživatelem stahováno identifikační číslo studenta (které je potřeba jako vstupní parametr v několika webových službách) a také se zjišťuje, zdali je kombinace údajů platná. Pokud pro uvedené uživatelské jméno nelze stáhnout číslo, pak je zobrazena informace o nesprávném uživatelském jménu, jinak je ověřeno heslo. Vrátí-li se chybový kód, je údaj opět zobrazen uživateli, jinak je propuštěn do aplikace s plnými funkcemi a barvou schématu reprezentující jeho fakultu.



Obrázek 17 sekce přihlašování a kontextové nabídky [autor práce]

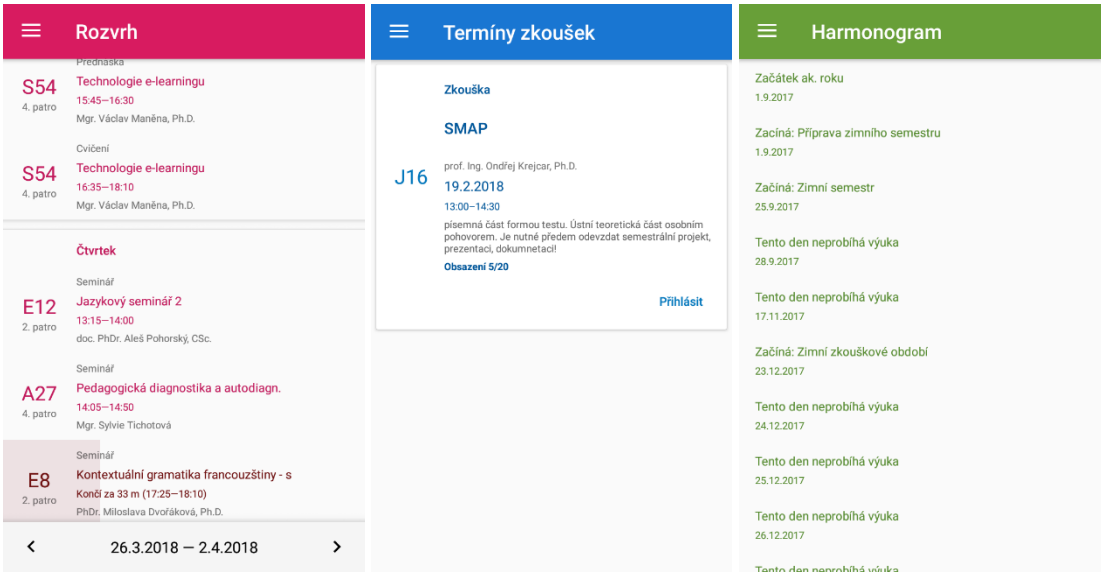
V sekci přehledu jsou uživateli zobrazovány údaje o aktuálně probíhajících rozvrhových akcích a změnách ve výuce. Společně s notifikačním panelem jsou tyto funkce rozděleny do kartiček vertikálně pod sebe. Na některých z nich jsou k dispozici ovládací prvky pro zobrazení rozšiřujících informací nebo kompletních seznamů. Řádky reprezentující rozvrhovou akci jsou vybaveny možností vyvolání menu s funkcí skrývání obdobných akcí do budoucna. Tyto preference jsou obnovitelné z hlavního nastavení aplikace.



Obrázek 18 sekce přehledu [autor práce]

Mezi fragmenty zobrazující čistě online informace patří částečně týdenní rozvrh, známky a zápočty, termíny zkoušek a harmonogram. Všechny tyto seznamy jsou po otevření inicializovány s probíhajícím načítáním z webových služeb – jsou tedy zpravidla o něco pomalejší než načítání z databáze.

V dolní části fragmentu rozvrhu je umístěn panel s popisem aktuálně zobrazovaného týdnu, na jehož stranách se nachází šipky pro přesun na předchozí nebo následující týden. Stejně tak je možné tento posun provést gestem potažení. Z těchto sekcí je rozvrh jediným, který takovým panelem disponuje – ostatní sekce mají ovládací prvky pouze v řádcích seznamu. Zbylé sekce reagují tak, jak je popsáno v kapitole s náčrty fragmentů.

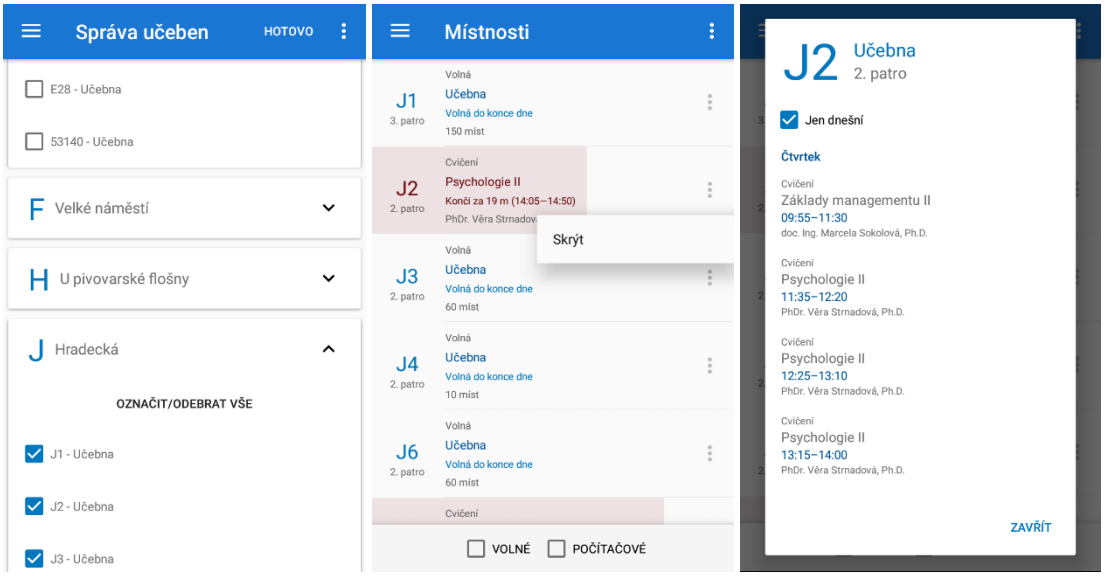


Obrázek 19 sekce rozvrhu, termínů zkoušek, harmonogramu [autor práce]

Nejkomplexnější sekcí je fragment s rozvrhem místností. Je nutné, aby obsahoval možnost výběru místností pro další stahování jejich rozvrhových akcí. Do této oblasti se lze dostat skrze vrchní menu jako správa učeben. Po otevření je obraz přesunut na seznam budov, jejichž položky jsou interaktivní a tyto učebny lze vybírat do svého přehledu.

Po výběru učeben jsou abecedně řazeny v seznamu a řádky nesou jejich informace. Pokud však v danou chvíli nebo v blízké budoucnosti probíhá nějaká rozvrhová akce, pak jsou v řádku zobrazovány právě informace o ni a časový identifikátor.

V dolní části fragmentu se nachází lišta s možnou filtrací učeben dle jejich vlastností nebo aktuální vytíženosti. Realizace výběru učeben spočívá v přeskládání či vynechání položek učeben, aby splňovaly kritéria zvolená uživatelem. V případě možnosti „volné“ jsou procházeny seznamy rozvrhových akcí pro každou učebnu, a pokud žádná akce z nich aktuálně neprobíhá, pak je učebna zahrnuta do výsledku. Po klepnutí na řádek učebny je zobrazen dialog skrývající rozvrhové akce. V horní části tohoto dialogu je k dispozici přepínání mezi dnešním nebo týdenním rozvrhem pro učebnu.

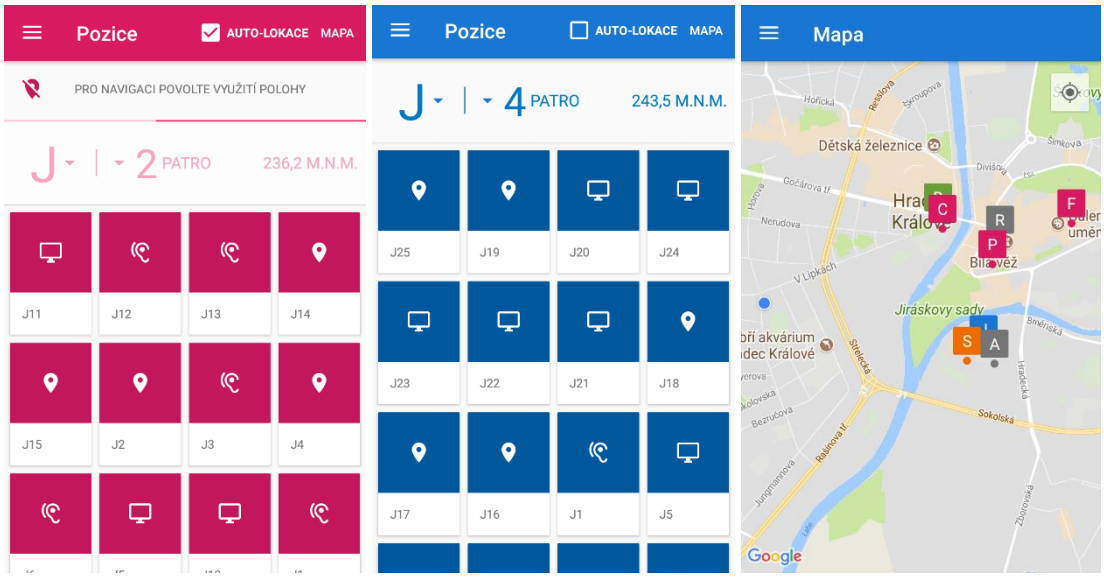


Obrázek 20 sekce učeben [autor práce]

6.2 Testování modulu indoor lokalizace

Fragment zobrazování blízkých objektů neuvádí na první pohled nijak obsáhlé informace. Smyslem seznamu je zachytit co nejvíce položek v rámci prostoru obrazovky a uživateli zobrazovat seznam seřazený vždy od nejbližších objektů k těm vzdálenějším. Proto je v tomto případě použit sloupcového layoutu, obsahujícího obdélníkové dlaždice s názvem učebny a s ikonou řadící se k jejímu typu. Dojde-li ke změně směru, patra či pozice, pak systém přeřadí učebny v aktuální budově. Tato změna je uživateli prezentována se současně vykonávanou animací změn pozic jednotlivých dlaždic.

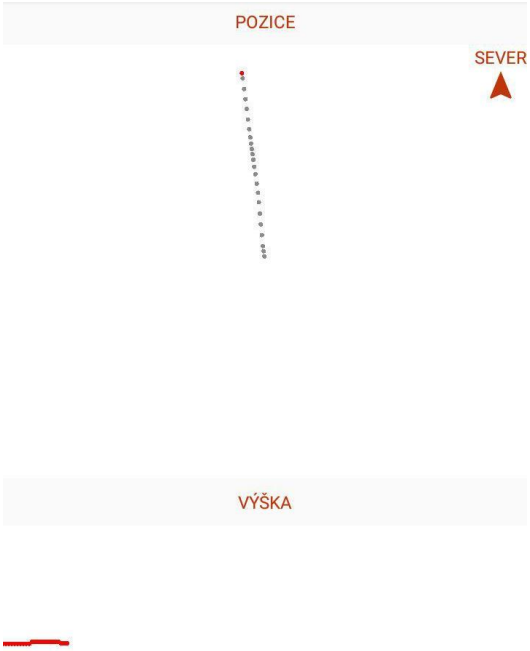
Nad seznamem jsou umístěny ovládací prvky pro změnu budovy či patra společně s informací o aktuální nadmořské výšce. V horním panelu je pak možnost výběru automatické lokalizace nebo vlastního určování a k dispozici je i odkaz na mapu, usnadňující orientaci mezi budovami ve městě.



Obrázek 21 sekce indoor lokalizace [autor práce]

Pro účely testování byla aplikace dočasně upravena pro zobrazování tvořené virtuální mapy pohybu uživatele. Na plátno byl každý krok vykreslen jako šedá tečka, která svou vzdáleností od předchozí byla v takovém poměru, aby se na plátno vešel delší pohyb vykonávaný do světových stran. Pod plátnem pohybu se nacházela část věnovaná přepočtu hodnot tlaku na nadmořskou výšku, která byla zobrazována ve formě postupující křivky zleva do pravé strany.

Užitím tohoto vykreslování bylo možné otestovat v praxi funkce uskutečnění detekce vlastností pohybu. Jedním z provedených testů byl na délku kroku v závislosti na rychlosti chůze. V uzavřené dlouhé místnosti, orientované na severní až mírně na severozápadní světovou stranu, byla vykonávána chůze s aktivní lokalizací a pohyb byl vykonáván takto: 3 kroky pomalou chůzí, dále přechod 4-5 kroků rychlé chůze a opět pokles v rychlosti, přičemž byl tento pohyb monitorován a zakreslen na plátno. Výsledek takového pohybu je na následujícím obrázku 22.

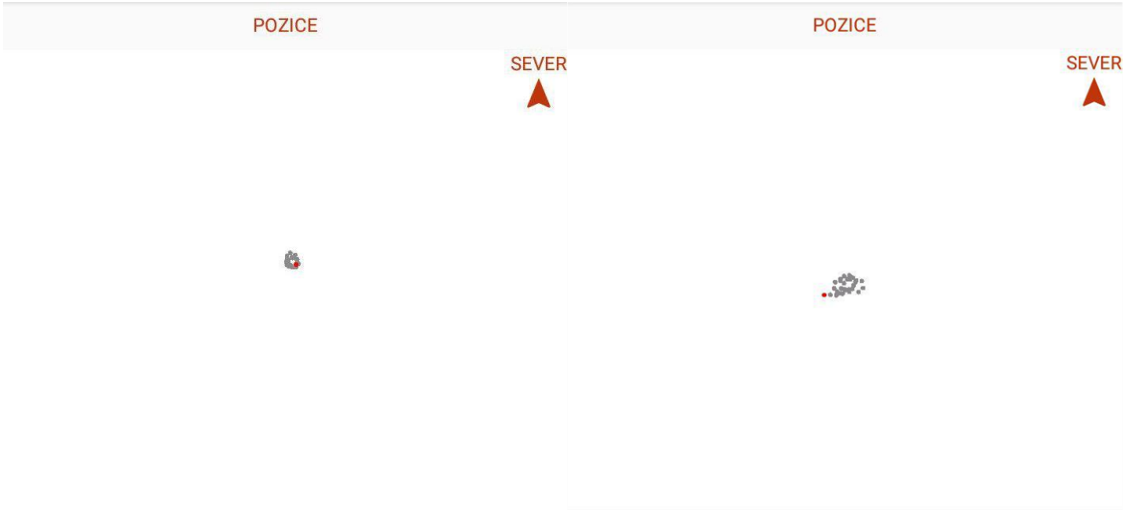


Obrázek 22 kresba virtuální mapy postupu uživatele při změně tempa během chůze [autor práce]

Na plátně je vidět několik počátečních kratších kroků ze středu plátna, které se záhy protáhnou v delší délkové intervaly a po několika dalších krocích se znovu zkracují. Z tohoto testu je patrné, že systém do jisté míry reaguje na změny ve stylu chůze i během vykonávaného pohybu a tím se zpřesňuje cílový výsledek.

Dalším z provedených testů je test detekce odstředivé síly. Největší chybovost tohoto jevu se projevuje v místě, kdy uživatel stojí na místě a otáčí se sem a tam – například rozhlíží-li se uživatel, na kterou stranu chodby se má vydat. Při vykreslení této situace na plátno je v případě vypnuté detekce odstředivé energie znatelný veliký rozptyl a nepřesnosti při výpočtu těchto hodnot, zatímco s aktivním systémem se

díky využití gyroskopu daří dostatečně potlačovat vypočítávanou délku kroku. Výsledky testu otáčení se o 720° je možné pozorovat na obrázku 23.



Obrázek 23 otáčení uživatele na místě s aktivním systémem potlačení odstředivé síly (vlevo) a bez zapnutého systému (vpravo) [autor práce]

Následné testy byly provedeny na několika stanovených trasách, které byly naměřeny a obsahují různé zatáčky či výškové rozdíly, aby bylo možné výsledné vzdálenosti při různých vlastnostech chůze porovnat s výpočty, které provádí systém, a určit tak relevanci výstupních dat.

Jedna z tras (trasa D v tabulce 2) se sestává z průchodu po točitém schodišti o dvě patra níže a zabočením k východu z budovy. Na tomto příkladu je poměrně hezky znázorněn pohyb dle směrového vektoru a současný sestup ve vypočtené nadmořské výšce. V jednotlivých zatáčkách je pak zřejmé zpomalení pohybu. Společně se všemi opatřeními systém svými výpočty v jednotlivých patrech s určitou chybovostí koreluje. Záznam trasy je na obrázku 24.

Obrázek 24 sestup točitým schodištěm [autor práce]

Vlastnosti všech tras a výsledky měření z jejich průchodu jsou v následující tabulce 2. Hlavními faktory, které výpočty ovlivňují, jsou například schodiště, kde jsou hodnoty akcelerometru logicky složeny z vyšších rozptylů. Tím negativně prodlužují délku kroku. Dalším z faktorů jsou např. drobná vychýlení při natočení zařízení v situacích, kdy se uživatel rozhlíží do stran (zařízení v ruce je mírně otáčeno a mění se směrový vektor).

Trasa	Reálná délka trasy (m)	Počet zatáček v trase (90°)	Obsahuje schodiště	Naměřená délka (m)		
				chůze (~2 km/h)	chůze (~4 km/h)	chůze (~6 km/h)
A	16	2	Ano	~16	~18	~18
B	22	1	Ne	~20	~22	~21
C	44	6	Ne	~41	~49	~48
D	32	7	Ano	~29	~34	~33
E	26	3	Ano	~23	~28	~29

Tabulka 2 chybovost výpočtů systému na jednotlivých trasách [autor práce]

6.3 Možnosti využití modulu lokalizace

Jedním z možných využití modulu indoor lokalizace na bázi využití senzorů je podpora jiných lokalizačních systémů ve smyslu dead reckoningu. Dead reckoning v navigaci je proces výpočtu nové pozice užitím předešlé nebo s pomocí jakéhosi šetření, což může být vlastnost nebo přidaná informace o průběhu mezi měřeními, která mohou při vhodném použití eliminovat chyby, a tak zpřesnit výpočet výsledné pozice.

Uvažujme příkladnou situaci již realizovaného systému využívající bluetooth signál a vysílací stanice, jejichž intenzitu zařízení změří a na principu trilaterace (obdobně jako GPS) určuje pozici zařízení vzhledem k pevné pozici stanic – tedy pozici zařízení v prostoru budovy. Dojde-li k průchodu signálu materiálem nebo k odrazu signálu od mnohem vzdálenější vysílací stanice, pak se pozice může lišit třeba o deset metrů. V době měřených intervalů signálu bluetooth je současně aktivní systém detekce lokální pozice v zařízení. Od posledního měření uvažuje takový systém možný posun o 3 metry. V době, kdy server z naměřených hodnot určuje novou pozici, se jako přidaná hodnota k požadavku přikládá i možná pozice, kterou vypočítalo naše zařízení. V tu chvíli aplikační logika na serveru může zavrhnout variantu výpočtu posunu o desítku metrů, a dokonce se z těchto chyb i poučit do příštích určování pozice v obdobných situacích.

7 Závěr

Přes všechny metody použité pro zpřesnění jsou časté výpočetní chyby na delších nebo náročnějších trasách. Seznamy učeben, které jsou uživateli zobrazovány, jsou proto uloženy dle svého pořadí v jednotlivých chodbách budovy, protože základ určení směru pohybu a přibližný posun je přesnější, a uživateli tak vrací relativnější výsledky, než kdyby se učebny zobrazovaly z virtuální mapy, na níž by mohlo dojít k vysokému zkreslení.

Způsob, jakým zpřesnit výpočet vzdáleností na bázi kroků, je brát v potaz výšky uživatelů, a ne zprůměrované délky kroků různě vysokých uživatelů. I přestože se délky kroků nelišily o nejvyšší hodnoty, každý centimetr zpřesnění na delších vzdálenostech jistě povede k přesnějším výsledkům. V takovém případě by bylo provedeno nové měření pro více výškových kategorií a dle volby uživatele by byla použita jiná výpočetní rovnice.

Dalším možným vylepšením je implementace spolehlivějšího krokoměru, který by eliminoval všechny pohyby špatně určené jako kroky. V tom ještě lepším případě by krokoměr fungoval i při zamčeném zařízení umístěném například v kapse uživatele. V tu chvíli by však nastal opětovný problém s určením směru pohybu – pro získání takové informace by bylo vhodné využít strojová učení, jejichž principy by v takových případech dokázaly dle vzoru tento vektor odhadnout.

Do výpočtu směrového vektoru by mohla být brána v potaz informace o změně tlaku, a tedy naklopení směrového vektoru i v rámci gravitačního pole. V tu chvíli by musela být virtuální mapa rozšířena o třetí dimenzi, což by zvýšilo nároky na paměti a výpočetní výkon.

Možný šum, který vzniká na akcelerometru, komplikuje situaci s detekcí kroků. Řešením by bylo provést měření na desítkách zařízení (s odlišnými procesory) a zjistit hranice chyb. Pokud by se šum pro ořez hodnot ukázal jako signifikantní, bylo by nutné rozlišit hranice pro různá zařízení, nebo užít jiné principy ořezu.

Jako nástroj pro dead reckoning a podobné případy by však popsaný algoritmus vedl i k širšímu využití. Při vykonání výše uvedených typů zpřesnění může být implementován a přiřazen k jiným systémům zjišťování pozice.

8 Bibliografie

[1] Statista, "Number of smartphone users worldwide from 2014 to 2020 (in billions)," 2018. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide. [Accessed 4 Březen 2018].

[2] Západočeská univerzita v Plzni, „IS/STAG," 20 Červenec 2017. [Online]. Available: https://is-stag.zcu.cz. [Přístup získán 04 Březen 2018].

[3] Západočeská univerzita v Plzni, „Mobilní aplikace," 245 Duben 2017. [Online]. Available: https://is-stag.zcu.cz/zakaznici/mobilni_aplikace. [Přístup získán 4 Březen 2018].

[4] M. Kus, *GPS,* České Budějovice: Vysoká škola ekonomická v Praze, 2007.

[5] K. Hanušová, *Gamifikace sběru rádiových fingerprintů,* Hradec Králové: Univerzita Hradec Králové, 2017.

[6] StatCounter, „Mobile Operating System Market Share Worldwide," Únor 2018. [Online]. Available: http://gs.statcounter.com/os-market-share/mobile/worldwide. [Přístup získán 11 Březen 2018].

[7] B. J. Vacula, *Vývoj aplikací pro Android a iOS,* Rudná: Vysoká škola ekonomická v Praze, 2014.

[8] O. Oravčok, *Získavanie náhodných dát zo senzorov v OS Android,* Brno: Masarykova univerzita, Fakulta informatiky, 2016.

[9] O. Essam, „Delay in android step counter," 10 Květen 2016. [Online]. Available: https://stackoverflow.com/questions/37136080/delay-in-android-step-counter. [Přístup získán 11 Březen 2018].

[10] Google Inc., „SensorEvent," [Online]. Available: https://developer.android.com/reference/android/hardware/SensorEvent.html. [Přístup získán 15 Březen 2018].

[11] Atlassian, „What is Git,“ [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-git. [Přístup získán 25 Březen 2018].

[12] Google Inc., „Fabric,“ [Online]. Available: https://get.fabric.io/. [Přístup získán 25 Březen 2018].

[13] Google Inc., „Material Design,“ [Online]. Available: https://material.io/guidelines/. [Přístup získán 15 Březen 2018].

[14] B. M. Z. Bajúszová, *Návrh a vytvoření REST / SOAP služby pro přístup do databáze,* Zlín: Univerzita Tomáše Bati ve Zlíně, 2017.

[15] Google Inc., „Save Data using SQLite,“ [Online]. Available: https://developer.android.com/training/data-storage/sqlite.html. [Přístup získán 17 Březen 2018].

[16] Google Inc., „The Activity Lifecycle,“ [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle.html. [Přístup získán 17 Březen 2018].

[17] Google Inc., „Fragments,“ [Online]. Available: https://developer.android.com/guide/components/fragments.html. [Přístup získán 17 Březen 2018].

[18] Square, Inc., „Retrofit,“ 2013. [Online]. Available: http://square.github.io/retrofit. [Přístup získán 17 Březen 2018].

[19] J. Reschke, „The 'Basic' HTTP Authetication Scheme,“ Září 2015. [Online]. Available: https://tools.ietf.org/html/rfc7617. [Přístup získán 18 Březen 2018].

[20] R. I. A. S. T. S. I. E. Rescorla, „The Secure HyperText Transfer Protocol,“ Srpen 1999. [Online]. Available: https://tools.ietf.org/html/rfc2660. [Přístup získán 18 Březen 2018].

[21] A. Grosner, „GitHub: DBFlow,“ [Online]. Available: https://github.com/Raizlabs/DBFlow. [Přístup získán 18 Březen 2018].

[22] M. Veškrna, *Positioning system for small devices using principles of inertial navigation system,* Brno: Masaryk University, Faculty of Informatics, 2013.

[23] Google Inc., „Google Git: SensorManager,“ [Online]. Available: https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/hardware/SensorManager.java. [Přístup získán 22 Březen 2018].

[24] L. Č. a. P. Hlavička, Numerické metody, Brno: Akademické nakladatelství CERM, 2005.