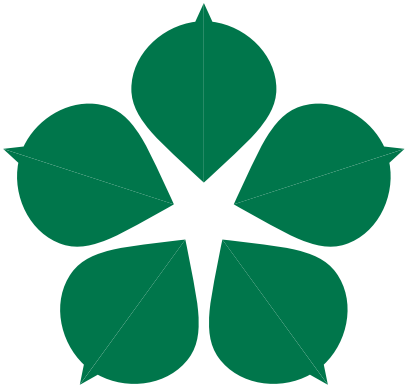
Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta



Vizualizace dat měřidel energií

Bakalářská práce

Libor Máca

Vedoucí práce: Ing. Václav Novák, CSc.

České Budějovice 2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: **Libor Máca**
(jméno, příjmení, tituly)

Obor – zaměření studia: 1801R001 / Aplikovaná informatika

Katedra/ústav, kde bude práce vypracovávána: Ústav aplikované informatiky

Školitel: Ing. Václav Novák, CSc., vacnovak@prf.jcu.cz, M: 606 666 694
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
.....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Vizualizace dat měřidel energii
Cíle práce:

Ztráty času při vyhledávání měřidel energii v plenéru se dají velmi efektivně eliminovat znalostí kmenových informací o umístění přístroje a dalších podrobnějších informací. V současnosti k tomu slouží mapové podklady, a to je nedostatečné. Student má za úkol doplnit vyhledávání využitím Augmented Reality (AR). Provede rešerši vhodných knihoven a zvolí vhodný algoritmus s ohledem na operační systém a pro použití fotoaparátu nebo kamery tak, aby byla možnost doplnit obraz o kmenová data. Je možné použít souřadnice GPS i kompas mobilního telefonu. Kmenová data budou k dispozici v mobilním telefonu (nebude řešit jejich přenos). Naprogramuje příslušnou aplikaci. Provede testy a doporučení pro nasazení do produkce.

Literatura:

- Schéma realit [cit. 26. 11. 2015] Dostupné z: tttps://www.itscj.ipsj.or.jp/hasshin_joho/hj_newsletter/hj_nl97/files/nl97-tpz1.PNG>
- Reality [cit. 26. 11. 2015] Dostupné z: <http://www.tcworld.info/uploads/RTEmagicC_313_Lumera_4.jpg.jpg >
- Rozšířená virtualita [cit. 26. 11. 2015] Dostupné z: http://www.markmcgill.co.uk/images/projects/augmented_virtuality/blending.png
- Encyklopedie v AR [cit. 29. 11. 2015] Dostupné z: <http://www.credencys.com/datafiles/2014/05/tablet.png>
- Památka v AR [cit. 29. 11. 2015] Dostupné z: <http://www.moremobilemarketingx.com/wp-content/uploads/2012/03/reichstag-augmented-reality-more-mobile-marketing-x.jpg>
- Dvojrozměrné objekty [cit. 29. 11. 2015] Dostupné z: http://www.thedigitalmarketingbureau.com/wp-content/uploads/2013/11/AR2.jpg

Vedoucí práce: Ing. Václav Novák, CSc..... podpis: Václav Novák
Garant oboru bak. studia, pokud je obor zajišťován jinou katedrou/ústavem, než ze které je školitel (nepožaduje se u oboru biologie): ..... podpis: .....
Vedoucí katedry: RNDr. Libor Dostálek podpis: .....
Případný souhlas vedoucího ústavu AV: ..... podpis: .....

V Českých Budějovicích dne ..... Podpis studenta: .....

Máca, L., 2020: Vizualizace dat měřidel energií. [Data visualization of energy meters. Bc. Thesis, in Czech.] – 43 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Cílem této bakalářské práce je seznámit čtenáře s technologiemi rozšířené reality. Vytvoření mobilní aplikace pro systém Android, implementující tuto technologii. Výsledná aplikace bude zobrazovat data měřidel energií a jejich lokalitu.

Klíčová slova

Rozšířená realita, Akcelerometr, Azimuth

Abstract

The goal of thesis is to describe technologies of augmented reality. Development of mobile application for Android that implements this technology. Developed application will display energy meter data with their location.

Keywords

Augmented reality, Accelerometer, Azimuth

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 20. 5. 2020 Libor Máca

Poděkování

Děkuji panu Ing. Václavu Novákovi, CSc. za vstřícné jednání a rady poskytnuté při psaní této bakalářské práce.

Obsah

1 Úvod

Technologie jsou nedílnou součástí našich životů a málokdo z nás by si dovedl představit život bez nich. Jedná se například o elektronická zařízení, kterými jsou drobné spotřebiče, mobilní zařízení nebo něco většího jako je například lednice či pračka. Tato zařízení ovšem nelze používat bez energií, které je udržují v provozu. Energie, kterými jsou například elektřina a plyn, nejsou volnými zdroji, a proto musí být evidováno, kolik se daných energií spotřebuje, ať už v rámci malé domácnosti či velké společnosti. Zaznamenávání těchto údajů mají na starosti měřidla energií, ze kterých jsou tato data pravidelně čerpána. Tento proces je znám jako odečet energií.

Odečty energií jsou prováděny zaměstnanci distribučních společností. Tito pracovníci mají určenou oblast, ve které odečty energií provádějí. V této oblasti musí provést odečet každého měřidla energie, které je pro ně určeno. V některých částech měst či obcí, může být velice obtížné se orientovat a určit, kterým směrem se vydat. Nyní s orientací pomáhá aplikace na mapové bázi, ale to může být občas matoucí a nedostačující.

Z toho důvodu byla vypracována tato bakalářská práce, jejímž cílem je zpřehlednění a zlepšení vizualizace dat měřidel energií. Bude toho dosaženo implementací rozšířené reality v aplikaci pro chytré telefony s operačním systémem Android. Technologie rozšířené a virtuální reality za posledních pár let velmi pokročily a jsou nyní snadno dostupné a implementovatelné. Pro tento projekt například postačí mobilní zařízení s objektivem a dvěma senzory, akcelerometrem a kompasem. Díky technologickému pokroku již není problém takováto zařízení získat a vlastní je již velká část populace. Rozšířená realita v tomto projektu bude využita tak, že na obraze, získaném pomocí objektivu mobilního zařízení, bude překrývat měřidla energií daty o nich. Díky tomu bude snadné určit, kterým směrem jsou jednotlivá měřidla situována a zároveň bude snadné získat přesné informace o daných měřidlech.

2 Analýza problematiky

2.1 Současný stav a existující řešení

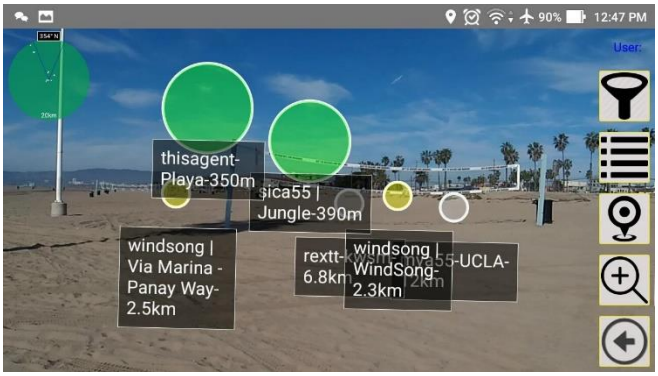
Odečty energií lze dělit do dvou kategorií. Odečty manuální a odečty na dálku. Současným nejčastějším řešením v českém prostředí je takzvaný odečet pochůzkou, kdy pracovník provádí za pomoci modemu a terminálu odečet buď ve společných prostorách domu nebo vně objektu. [1] Zůstává však zhruba 10 % elektroměrů, které je nutné odečíst manuálně z jejich číselníku. Ať už je to z důvodu technické závady dálkového odečtu, nebo je zaviněno lidským faktorem (poškození elektroměru, poničení sítě apod.)

Existujícím řešením při hledání těchto neodečtených měřidel je aplikace běžící na zařízení určeném pro provádění odečtů energií. Tato aplikace zobrazuje přesné umístění měřidla na mapě. Aplikace získává data z databáze po řádcích, kde jedna řádka odpovídá jednomu měřidlu energií. Tato řádka je poté rozdělena dle specifikovaného počtu znaků na jednotlivé atributy s důležitými informacemi (zeměpisná poloha, telefonní číslo odběratele, adresa apod.)

2.2 Podobná řešení

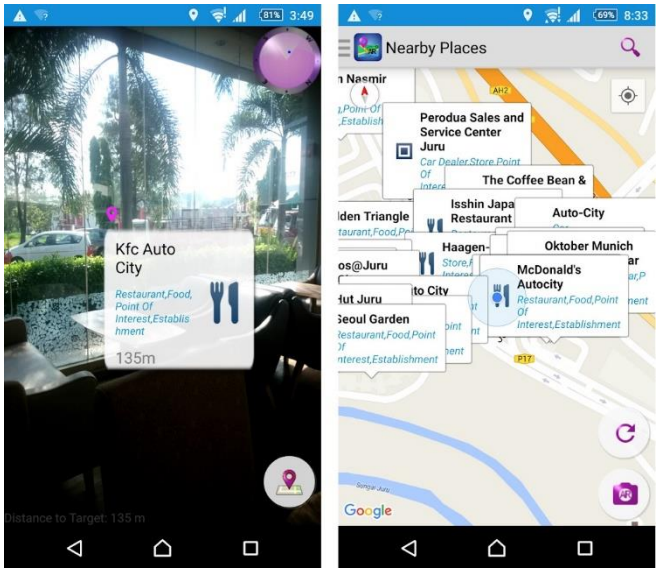
Jelikož se jedná o velmi konkrétní problém, projekty, které by řešily přesnou problematiku zadání této bakalářské práce, nelze nalézt. Je ovšem možné nalézt mobilní aplikace, které řeší části této problematiky a tato řešení analyzovat. Ať už se jedná o uživatelské rozhraní, či způsob implementace rozšířené reality a mapového podkladu.

Jednou z těchto aplikací je například **WAR - Widespread Augmented Reality II**. Tato aplikace umožňuje vytvoření virtuální značky pro zeměpisné souřadnice, na kterých se uživatel právě nachází. Tuto značku si poté uživatel může zobrazit za použití rozšířené reality spolu se značkami svých přátel. [2] Aplikace nedisponuje pouze rozšířenou realitou, ale také mapovým rozhraním.



Obrázek 1 Aplikace WAR II

Další aplikace, která bude analyzována je **Augmented Reality Map**. Tato aplikace vypovídá více o výstupní aplikaci této bakalářské práce a jejích funkcích. Aplikace používá rozšířenou realitu pro zlepšení mapové navigace. Základní funkcí aplikace je zobrazení Google Maps lokací, pro usnadnění orientace. [3][1] Stejně jako první aplikace i tato disponuje mapovým rozhraním, ve kterém jsou lokace zobrazeny.



Obrázek 2 Aplikace Augmented Reality Map

2.3 Shrnutí poznatků

Je pochopitelné, že autoři těchto aplikací nezveřejnili zdrojové kódy projektů, a proto je analýza zaměřena především na grafické rozhraní aplikací a jejich funkcionalit. Poznatky

získané analýzou těchto aplikací jsou použity jako odrazový můstek při plánovaní postupu vývoje daného projektu.

První stěžejní poznatek, jenž analýza přinesla, pojednává o určování zeměpisné polohy. V aplikacích založených na zeměpisné poloze, je nesmírně důležité, získání co nejpřesnějších souřadnic. Je tedy akceptovatelné, aby určování co nejpřesnější polohy bylo na úkor času, který tato operace zabere. Proto bude kladen důraz na metody pro získání polohy uživatele.

Druhý poznatek je spíše estetického rázu, ale částečně plní i důležitou funkcionalitu. Jedná se o orientaci aplikace. Tím je myšleno, zda obraz aplikace bude zobrazován v takzvaném portrait mode[1], nebo v landscape mode[2].

Pro zobrazení rozšířené reality je vhodnější variantou landscape mode. Pokud je zařízení v této poloze, obraz získaný objektivem zařízení má širší úhel zorného pole. Širší úhel zorného pole znamená, že bude možné zobrazit data s většími rozestupy a celkově bude uživateli zobrazeno širší okolí. Není ovšem zapotřebí, aby aplikace toto rozložení vynucovala, a proto bude aplikace v tzv. full sensor mode, kdy aplikace reaguje na převrácení zařízení, čímž dojde ke změně způsobu vykreslování.

Další poznatek je zaměřen na hodnotu pitch. Obě tyto aplikace pracovaly s tímto údajem tak, že zařízení muselo být v kolmé poloze, aby byla data vykreslena na obrazovce. Stejné chování bude možné nalézt i v tomto projektu.

Posledním poznatkem je způsob, kterým jsou zobrazeny všechny lokace v rozhraní rozšířené reality. Obě aplikace toho docílily použitím radaru, na kterém jsou lokace vykresleny jakožto jednotlivé body. Tento způsob zobrazení bude implementován i v tomto projektu. Na radaru bude také možné rozlišit, které měřidlo energií je právě zobrazeno, a to díky odlišné barvě jež daný bod na radaru bude mít.

[1] Portrait mode – Mobilní zařízení se nachází v horizontální poloze
[2] Landscape mode – Mobilní zařízení se nachází ve vertikální poloze

2.4 Definice pojmů

2.4.1 Rozšířená (augmentovaná) realita:

Pojem rozšířená realita bývá mnohdy špatně interpretován. Člověk, který se v těchto technologiích příliš neorientuje, jej často zaměňuje s pojmem virtuální realita. Kniha autorů Jonathana Linowese a Kristiana Babilinskeho o virtuální realitě říká, že vás kouzelně, přesto ale přesvědčivě, přenese do jiného (počítačem generovaného) světa. Na rozdíl od rozšířené reality, kterou autoři popisují jako realitu, která rozšíří stávající vnímání skutečného světa přidáním digitálních dat. [4] Tato definice pojmu rozšířené reality je snadno pochopitelná i pro laika, ovšem kniha „Practical augmented reality“ autora Steva Aukstakalnise, poskytuje odbornější definici tohoto pojmu. Popisuje jej jako obecný pojem, aplikovatelný pro různé, obraz zobrazující technologie, jež jsou schopny překreslit či zkombinovat alfanumerické, symbolické či grafické informace s pohledem uživatele na skutečný svět. [5] Tato definice není v rozporu s definicí Linowese a Babilinskeho, jedná se pouze o odborněji popsanou definici, s tentýmž významem.

Historie

Roku 1992 se zrodil pojem rozšířená realita. Tento pojem se poprvé objevil na dvacáté páté havajské mezinárodní konferenci systémových věd a jeho autory jsou Thomas P. Caudell a David Mizell. Ve stejném roce byl vytvořen funkční systém rozšiřující realitu v laboratořích amerického letectva. [6]

V roce 1997 byl vyvinut první venkovní systém podporující tuto technologii. Jedná se o takzvaný cestovatelský stroj (The Touring Machine). [7] Tento stroj pracuje na stejném principu, jako výsledná aplikace této práce.

Obrázek 3 Touring Machine (levý). Záznam generovaného obrazu pořízeného na verzí stroje z roku 1999 (pravý).

Rozdělení

1. Rozšířená realita založena na značkách (marker-based AR):

   Také známá jako rozšířená realita založená na rozpoznání značek. Detekuje značku zachycenou objektivem zařízení a na této značce vykreslí požadovaný objekt.

2. Rozšířená realita bez použití značek (marker-less AR):

   Také známá jako rozšířená realita založená na lokaci. Z názvu vyplívá, že pro vykreslování údajů nejsou použity značky, ale zeměpisné souřadnice. Jinými slovy lze vytvořit aplikace, které vykreslí data na daných souřadnicích (adresy domů, památky apod.) Tato bakalářská práce implementuje tento typ rozšířené reality.

3. Projekční rozšířená realita (projection AR):

   Jedná se o jednu z nejjednodušších rozšířených realit. Data jsou promítána na povrch objektu a překrývají skutečnou realitu. Uživatel může s touto projekcí interagovat.

4. Rozšířená realita založená na překreslení (Superimposition Based AR)

   Jedná se o jev, kdy po naskenování objektu (rentgen), je rozšířená realita schopná tento objekt překreslit. Například po naskenování části těla je možné překreslit tuto část těla tak, aby zobrazovala nervový systém, jenž se v ní nachází.

2.4.2 Akcelerometr:

Akcelerometr je senzor, kterým je vybavena většina chytrých telefonů. Tento senzor měří zrychlení zařízení. Akcelerometr využívá mikroskopické krystaly, na kterých se při působení vibrací generuje napětí, odpovídající určitému zrychlení. [8] Díky tomuto jevu je možné detekovat změny orientace zařízení a určit směr gravitace.

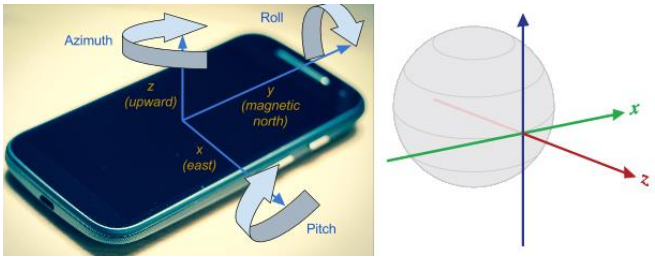
2.4.3 Pitch, roll, yaw, azimuth:

Údaje pitch, roll a yaw popisují orientaci zařízení. Lze si je představit jako tři osy v prostoru (X, Y, Z).

Osa X je kolmá k boční (delší) straně zařízení. Otáčení zařízení po této ose se nazývá **pitch**.

Osa Y je kolmá k horní (kratší) straně zařízení. Otáčení zařízení po této ose se nazývá **roll**.

Osa Z je kolmá ke středu planety Země. Otáčení zařízení po této ose se nazývá **yaw**. Tato osa také umožňuje získat hodnotu **Azimuth**, která určuje rozdíl osy Y k severnímu pólu. Dalo by se tedy říci, že se jedná o údaj, který určí světovou stranu, k níž je toto zařízení v danou chvíli situováno.



Obrázek 4 Osy vůči zařízení. (levý) Osy vůči středu země. (pravý)

2.5 Funkční požadavky

Zadání této bakalářské práce je velmi obecné a je obtížné si pod ním představit něco konkrétního, proto je tato část zaměřena na vymezení funkčních a nefunkčních požadavků výsledné aplikace.

1. Rozšířená realita

a. Zobrazení dat měřidel energií pouze v případě, že je zařízení orientováno směrem k měřidlu

b. Otevření informačního okna měřidla energií

c. Zahájení telefonního hovoru na číslo, uvedené v informačním okně konkrétního měřidla energií

d. Označení zařízení, na kterém byl odečet již proveden

e. Zobrazení měřidel energií formou bodů vykreslených na radaru

2. Mapový podklad

a. Zobrazení měřidel energií na mapě

b. Otevření informačního okna měřidel energií

c. Zahájení telefonního hovoru na číslo, uvedené v informačním okně konkrétního měřidla energií

d. Označení zařízení, na kterém byl odečet již proveden

3. Seznam měřidel energií

a. Zobrazení seznamu měřidel energií

b. Zahájení telefonního hovoru na kterékoliv číslo, uvedené v seznamu měřidel energií

c. Označení měřidel, na kterých byl odečet již proveden

d. Označení všech měřidel energií současně

4. Nastavení

a. Změna maximální vzdálenosti pro zobrazení měřidel energií

b. Změna vzhledu informačních oken a mapového podkladu

c. Skrytí radaru v rozhraní rozšířené reality

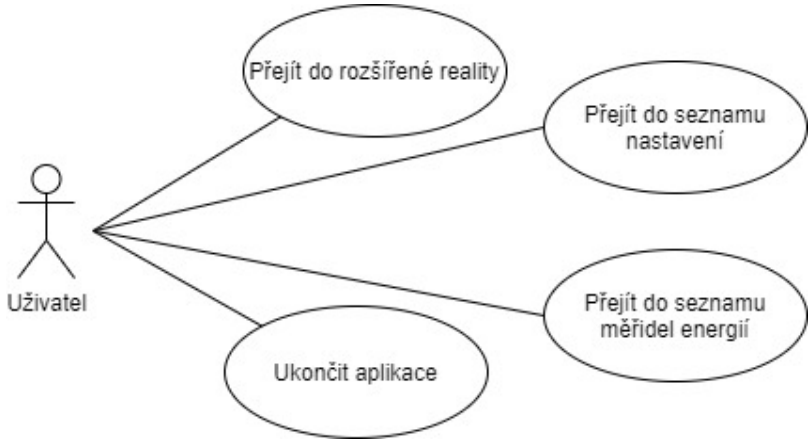
2.6 Nefunkční požadavky

1. Aplikace pro operační systém Android
2. Minimální SDK verze 21
3. Programovací jazyk Kotlin
4. Kmenová data uložená v mobilním zařízení
5. Získání žádoucích dat ze souboru a vložení do lokální databáze
6. Vykreslení obrazu pořízeného kamerou mobilního zařízení
7. Získání dat pomocí senzorů mobilního zařízení
8. Použití MVVM architektury

2.7 Případy užití

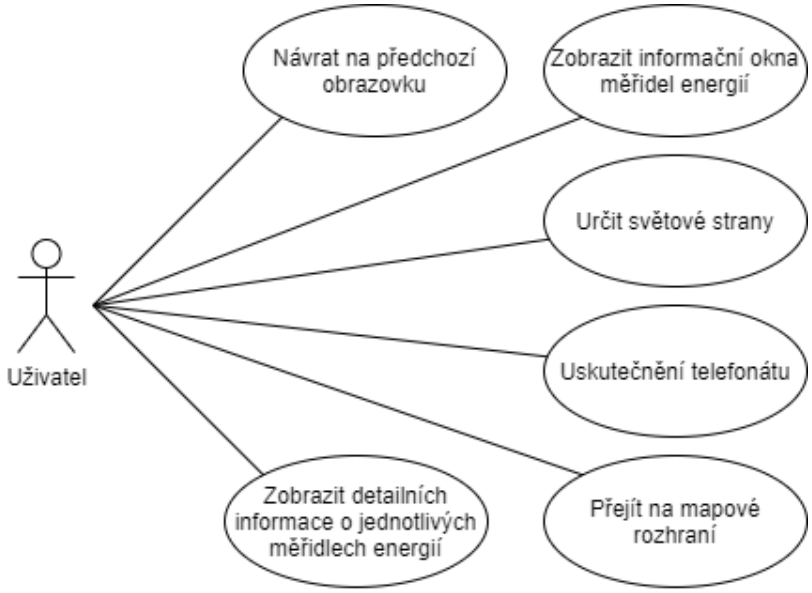
Aplikace bude umožňovat pouze uživatelský způsob použití. To například znamená, že nebude existovat administrátorské rozhraní a všem uživatelům bude umožněno používat aplikaci stejným způsobem.

První způsob použití ukazuje možnosti uživatele po spuštění aplikace. Jedná se pouze o možnost navigace aplikací a její ukončení.



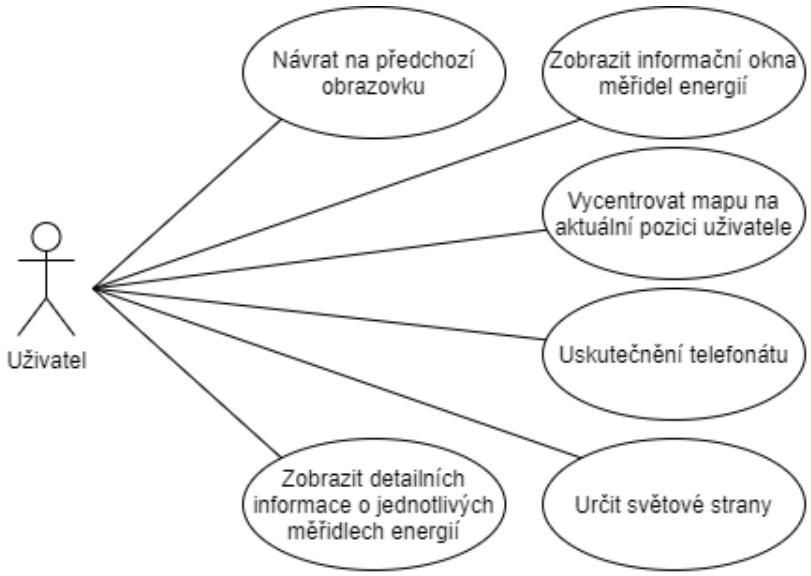
Obrázek 5 Diagram užití – Hlavní obrazovka

Druhý způsob užití se zabývá částí aplikace implementující rozšířenou realitu. Na tuto část je kladen nejvyšší důraz. Zde budou vykreslována data měřidel energií formou malého informačního okna, obsahujícího adresu a vzdálenost k danému měřidlu energií. Jednotlivá okna budou interaktivní a při jejich otevření se zobrazí podrobnější informace o daném měřidle. V podrobnějším zobrazení měřidel, je uživateli umožněno zahájit telefonní hovor na číslo, jež je zde uvedeno. V rozhraní rozšířené reality bude zobrazen radar, díky kterému bude uživatel schopen určit světové strany a přibližnou lokaci všech měřidel energií, nacházejících se v blízkosti aktuální polohy uživatele. Dále bude umožněn návrat do hlavní nabídky, či přepnutí do mapového rozhraní.



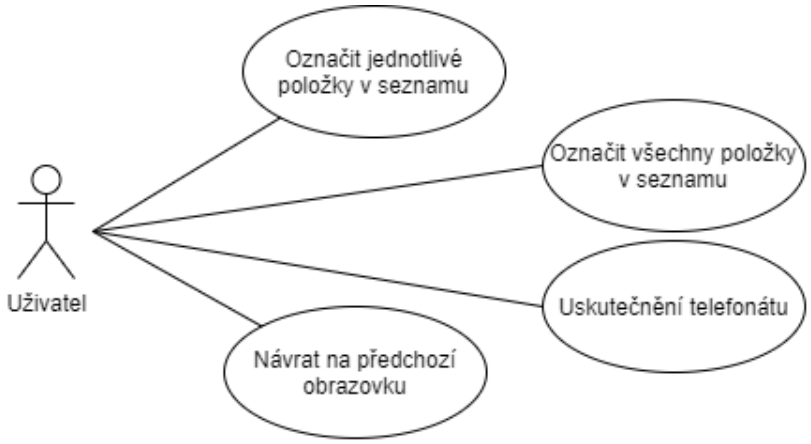
Obrázek 6 Diagram užití – Rozšířená realita

Rozhraní s mapovým podkladem bude nabízet podobné případy použití jako rozhraní rozšířené reality. Budou zde vykreslena měřidla energií na mapovém podkladu a jednotlivá měřidla energií bude možné otevřít pro zobrazení podrobnějších informací o daném měřidle. V podrobnějším zobrazení měřidel, je uživateli umožněno zahájit telefonní hovor na číslo, jež je zde uvedeno. Dále bude možné vycentrovat mapový podklad na aktuální polohu uživatele. Z této obrazovky bude možný návrat do části aplikace implementující rozšířenou realitu.



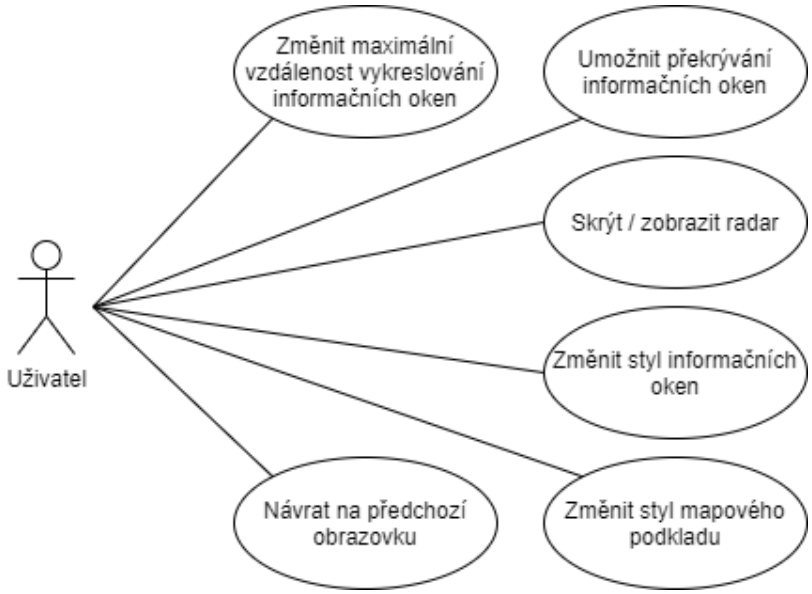
Obrázek 7 Diagram užití – Mapový podklad

Seznam měřidel energií bude možné využít pro přepínání mezi stavy, určující, zda jednotlivá, či všechna zařízení budou vykreslována, nebo skryta. V seznamu budou zobrazeny stručné informace o jednotlivých měřidlech energií a pro konkrétní měřidla energií bude uživateli umožněno zahájit telefonní hovor na číslo, které je zde uvedeno.



Obrázek 8 Diagram užití – Seznam měřidel energií

V nastavení bude možné změnit maximální vzdálenost pro zobrazení informačních oken, vzhled mapového podkladu a informačních oken, přepínaní mezi skrytím a zobrazením radaru, a jiné.



Obrázek 9 Diagram užití – Nastavení

3 Návrh řešení a technologie

Tato kapitola je zaměřena na použité technologie a návrh řešení. Popis návrhu řešení bude zahájen návrhem grafického rozhraní, postupně se propracuje k objektovému návrhu aplikace, a na závěr i k její architektuře. Dle zadání tento projekt neimplementuje přístup ke vzdálené databázi, pouze pracuje s lokální databází pro uchování stavu dat.

3.1 Použité technologie

3.1.1 Programovací jazyk

Výběr programovacího jazyka byl zúžen na tři možné varianty. Aplikace mohla být vyvíjena v jazycích C#, Kotlin nebo Java. Pro jazyk C# existuje mnoho frameworků, které usnadňují práci s rozšířenou realitou, tento jazyk ovšem není oficiálním jazykem pro vývoj nativních Android aplikací a je vnímán spíše jako jazyk určený pro vývoj mobilních her. Z těchto důvodů nebyl zvolen pro vývoj tohoto projektu.

Jazyky Kotlin a Java jsou podporovány vývojáři společnosti Android jakožto primární jazyky pro vývoj nativních Android aplikací. Jazyky mají ekvivalentní úroveň oficiální dokumentace a příkladů zdrojového kódu. Kotlin je však z těchto jazyků novější a má oproti jazyku Java mnoho výhod, které usnadňují vývoj mobilních aplikací. Dále se jazyk Kotlin frekventovaněji vyskytuje v příkladech zdrojového kódu vývojářské komunity. Z těchto důvodů byl pro vývoj tohoto projektu zvolen jazyk Kotlin.

3.1.2 Vývojové prostředí

Výběr vývojového prostředí se odráží od výběru programovacího jazyka. Pokud by byl zvolen programovacím jazykem C#, bylo by voleno mezi vývojovými prostředími Unity a Microsoft Visual Studio. Pro jazyky Kotlin a Java je více než dostačujícím prostředím Android Studio, které je zaměřeno na tyto jazyky a vývoj Android aplikací. Mimo jiné je toto vývojové prostředí od vývojářů operačního systému Android, je zdarma dostupné a poskytuje veškeré prostředky, potřebné pro vývoj Android aplikací.

3.1.3 Frameworky

I přes velké množství frameworků, vyvinutých pro usnadnění implementace rozšířené reality, projekt nevyužívá žádný z nich. Frameworků, které podporují rozšířenou realitu založenou na lokalitě, není mnoho a u většiny z nich se jedná o komerční produkty, jejichž licence je pro použití v tomto projektu příliš nákladná. Jedním z nejlepších frameworků, jenž toto umožňuje, je Wikitude. Náklady na licenci pro použití v jednom projektu činí téměř sedmdesát tisíc korun českých. Navíc se jedná o licenci, která nepodporuje například uskutečnění telefonních hovorů z aplikace, jež je v tomto projektu žádoucí. Po hlubší analýze bylo také zjištěno, že se ani nejedná o nativní řešení a framework využívá javascript spolu s html a css styly pro implementaci rozšířené reality.

Dalším frameworkem, jež je často využíván pro implementaci rozšířené reality, je ARCore od společnosti Google. Přestože se jedná o bezplatný framework, nebyl zvolen z důvodů malé kompatibility se zařízeními. Tento framework podporuje pouze specifická zařízení a spousta mobilních telefonů, jež disponují kompasem i akcelerometrem (předpoklady pro implementaci rozšířené reality), nemůže spustit aplikace jež byly vyvinuty za použití ARCore.

Existují i open source knihovny a další frameworky, které by bylo možné využít. U těchto knihoven a frameworků ovšem bývá často problém s omezeným počtem funkcí. Například jedna z knihoven, jež byla analyzována pro potřeby vývoje, neumožňovala interakci s informačními okny. Tato funkce je důležitá pro projekt a její implementace za použití této knihovny by znamenala značný zásah do kódu knihovny, což by ve finále bylo komplikovanější než vývoj vlastní logiky, jež implementuje rozšířenou realitu.

Z důvodů ceny, omezenosti funkcí, použití nenativního přístupu a jednoduchosti vlastní implementace nebyl zvolen žádný framework. Vlastní implementace také poskytuje absolutní kontrolu nad kódem.

3.1.4 Komponenty

V projektu jsou použity komponenty, které byly oficiálně představeny teprve v roce 2019 a zatím nejsou hromadně využívány. Právě z tohoto důvodu se tato část zabývá stručným

popisem daných komponent, aby si čtenář lépe představil, k čemu slouží a jak budou využity v daném projektu.

Data Binding

Jedná se o knihovnu, která umožňuje propojit prvky uživatelského rozhraní spolu s datovými zdroji. [9] Vezměme si například třídu, která provádí generování náhodných čísel, jež jsou poté zobrazena na obrazovce. Pokud by nebyl použit Data Binding, musela by třída provádějící generování čísel při každé změně hodnoty informovat uživatelské rozhraní o dané změně. Naopak při použití Data Bindingu se v uživatelském rozhraní určí, která hodnota bude pozorována a uživatelské rozhraní tak bude mít neustálý přehled o aktuálním stavu dané hodnoty.

Live Data

Live Data, je pozorovatelný držitel dat. Na rozdíl od běžných pozorovatelných dat, prvek Live Data je obeznámen o životním cyklu aplikace. [10] Životní cyklus aplikace říká, kdy je například aplikace na pozadí, kdy ve stavu spouštění, ukončení apod. Z toho tedy vyplívá, že Live Data bývají používána spolu s Data Bindingem a že se pozorování těchto dat ukončí nebo pozastaví právě ve chvíli, přejde-li aplikace do stavu, kdy není vyžadováno, aby bylo uživatelské rozhraní aktualizováno novými daty.

ViewModel

ViewModel je navržen pro uchovávání a správu dat, souvisejících s uživatelským rozhraním. Dále umožní datům, aby byla zachována při konfiguračních změnách, kterou je například změna orientace zařízení. [11] V praxi tedy ViewModel odpovídá stejnojmenné vrstvě v architektuře MVVM[3]. Při konfiguračních změnách je obraz zničen a znovu vykreslen. Tento jev vede k nežádoucí ztrátě dat, pokud není vytvořena metoda pro jejich uchování. ViewModel tyto operace vykonává automaticky.

[3] MVVM je zkratkou pro: Model, View, ViewModel

Navigation

Navigování skrze aplikaci je stěžejní součástí jejího designu. Pomocí komponenty Navigation lze navrhnout interakce, které umožní uživateli snadnou navigaci napříč aplikací. [12] Použitím této komponenty bylo zaznamenáno zlepšení výkonu aplikace a usnadnění práce při implementaci přechodů mezi fragmenty.

Room

Knihovna Room poskytuje abstraktní vrstvu nad SQLite[4], která umožňuje robustnější přístup k databázi, zatímco využívá plného potenciálu SQLite. [13] Knihovna Room je tedy použita pro zapisování a čtení dat z lokální databáze zařízení.

3.1.5 Logování

Logování je proces, při němž jsou vytvářeny takzvané logy. Logy zaznamenávají data, určená pro analýzu a informace o chybách, které se vyskytly při běhu aplikace. [14] Mobilní aplikace, stejně jako backendové[5], je důležité logovat. Zásadní rozdíl je ovšem ten, že u mobilních aplikací nemá administrátor ve většině případů přístup k danému zařízení, na kterém je aplikace nainstalována, jako je tomu u backendových aplikací. Z tohoto důvodu musí mobilní aplikace využívat služby pro logování, které pomocí internetu zasílají logy na místo, ke kterému má administrátor přístup. Tyto služby si lze vytvořit samostatně, ale existuje mnoho knihoven a poskytovatelů těchto služeb, a právě z těchto důvodů bylo pro tento projekt zvoleno již existující řešení, kterým je Bugfender. Bugfender je snadno implementovatelná služba, která disponuje webovým rozhraním, kde je možné si zobrazit dané logy. Bugfender nabízí bezplatnou licenci, která je omezena na sto tisíc řádků logů denně. Pro aktuální stav aplikace je tato licence více než dostačující. Další silnou stránkou této služby je uchovávání logů a informací o pádech aplikace v mobilním zařízení do doby, než je zařízení připojeno k internetu, díky čemuž nedojde ke ztrátě logů, v případě, že zařízení není momentálně připojeno k internetu.

[4] SQL lite je nízko zátěžová softwarová knihovna poskytující systém pro správu relační databáze
[5] Backendová aplikace zůstává na pozadí, nejčastěji umístěná na serveru

3.1.6 Profily

Profily jsou důležitou součástí, kterou nelze opomenout při vývoji aplikací. Název se může lišit dle vývojových prostředí, jazyků či frameworků. Například framework spring boot nazývá tyto profily environmenty. U Android aplikací se jedná o takzvané build typy.

Profily slouží pro definování chování aplikace a proměnných, které se mohou lišit dle účelu dané verze aplikace. Příkladem takovéto proměnné může být například api klíč, sloužící pro použití služby logování. Pro profil sloužící k vývoji aplikace není zapotřebí používat placenou licenci, proto postačí pouze api klíč bezplatné verze. Pro verzi, která je již poskytnuta uživatelům ovšem může stát za zvážení, zda není zapotřebí jiné licence, která by přinesla výhody nepotřebné během vývoje aplikace. Tento projekt bude definovat tři profily dle použití aplikace:

Develop

Profil develop je určen pouze pro vývoj aplikace a případné manuální testování vývojářem. Pod tímto profilem jsou také spouštěny automatické testy.

Staging

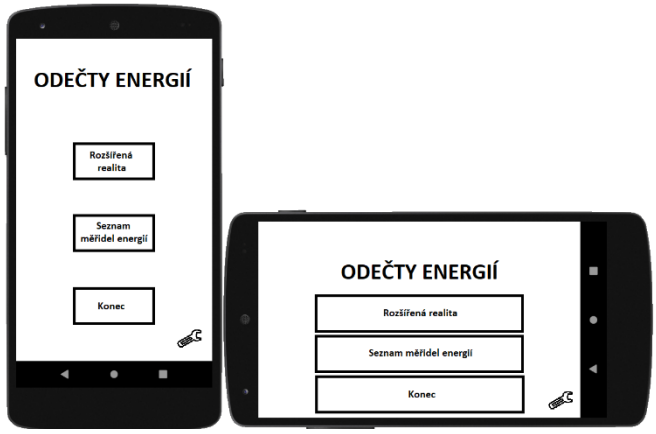
Profil staging je určen pro širší skupinu uživatelů, jež se podílí na testování dané aplikace.

Release

Profil release slouží k poskytnutí již otestované verze zákazníkovi.

3.2 Návrh grafického rozhraní

Při spuštění aplikace se uživatel ocitne na úvodní obrazovce. Jak již vyplívá z diagramu použití, uživatel bude mít z této obrazovky přístup k nastavení, seznamu měřidel energií, ukončení aplikace a samozřejmě i k rozhraní, implementující rozšířenou realitu. Úvodní obrazovka tedy bude obsahovat pouze 4 tlačítka pro navigaci aplikací.



Obrázek 10 Návrh grafického rozhraní hlavní obrazovky

Nastavení na rozdíl od ostatních fragmentů nepoužívá klasický layout, ale soubor s příponou xml spolu s třídou PreferenceScreen. Grafické prvky jsou vykreslovány v pořadí, ve kterém jsou napsány v tomto souboru. Nastavení bude tedy disponovat obdobným designem jako klasické android nastavení. Výhodou této implementace je snadné přidání případných nových položek.

Převážná část obrazovky seznamu energií bude překryta informačními okny reprezentujícími jednotlivá měřidla energií. Dále zde budou popisné informace v horní části obrazovky a přepínače pro zobrazení a skrytí měřidel energií.

Návrh obrazovky s rozšířenou realitou není zapotřebí příliš popisovat. Přes celou obrazovku bude vykreslen obraz, získaný objektivem zařízení, spolu s tlačítkem pro přechod do mapového rozhraní. V pravém dolním rohu bude také zobrazen dynamický radar pro usnadnění orientace a získání představy o poloze měřidel energií.

Obrazovka s mapovým podkladem neobsahuje téměř žádné prvky uživatelského rozhraní. Bude zde pouze přes celou obrazovku zobrazen mapový podklad, na kterém budou vykresleny interaktivní ikonky měřidel energií a v pravém horním rohu bude umístěno tlačítko pro návrat do rozhraní rozšířené reality.

3.3 Objektový návrh aplikace a architektury

Objekty v tomto projektu budou dodržovat princip jedné odpovědnosti (single responsibility principle). V praxi to tedy znamená, že bude například vytvořena třída, která se bude zabývat pouze správu lokace. Tato třída bude definovat jednotlivé metody, které budou mít také pouze jednu odpovědnost, například metoda pro získání lokace, metoda pro ukončení procesu získávání lokace apod. Díky objektovému návrhu aplikace, bude snadné tuto aplikaci spravovat a zároveň bude přehledná, snadno testovatelná a do budoucna i snadno rozšiřitelná.

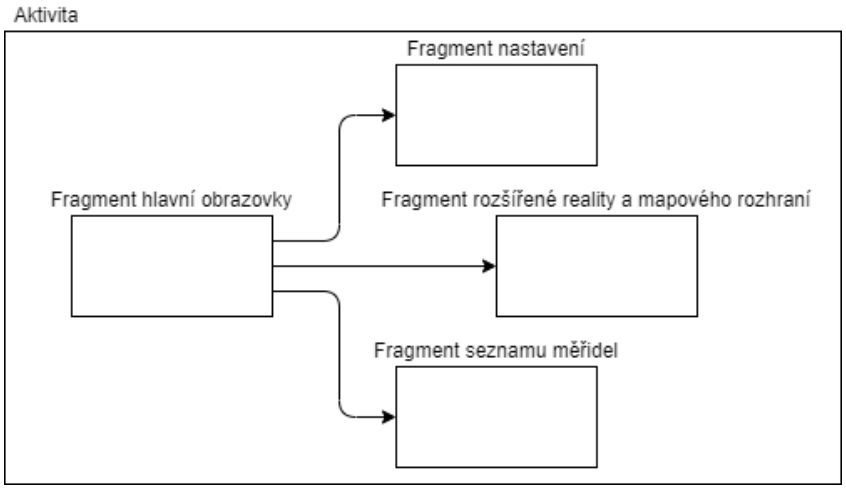
Při navrhování aplikace bylo voleno z několika typů architektur, které jsou použity při vývoji aplikací a softwarů. Jsou jimi například architektury MVC[6], MVP[7], již zmíněna MVVM a jiné. Pro tento projekt byla zvolena architektura MVVM. Jedná se o nejnovější architekturu z uvedené trojice. Vydáním již zmíněných komponent společnost Android usnadnila vývoj aplikací s touto architekturou.

3.3.1 Aktivita a fragmenty

Projekt bude vyvíjen dle vzoru jedné aktivity (single activity pattern). Z názvu vyplívá, že aplikace disponuje pouze jednou aktivitou, která slouží jako vstupní bod aplikace. Na aktivitu se neváže grafické rozhraní. Grafická rozhraní jsou poté propojena s jednotlivými fragmenty.

[6] MVC je zkratkou pro: Model, View, Controller

[7] MVP je zkratkou pro: Model, View, Presenter



Obrázek 11 Návrh fragmentů

V návrhu je možné vidět, že jedna aktivita zastřešuje všechny fragmenty. Šipky v tomto návrhu reprezentují akce, které jsou poté volány ve zdrojovém kódu jednotlivých fragmentů, z nichž šipky vychází. Díky těmto akcím je možné přepínat jednotlivé obrazovky a vracet se na předchozí. Pokud by existoval fragment, který by byl přístupný odkudkoliv, byl by reprezentován šipkou, která by neměla počátek v žádném z fragmentů. Takováto akce se nazývá globální.

3.3.2 Třídy reprezentující view:

Jedná se o třídy, které slouží pro interakci s grafickým rozhraním. Celkem bude tyto třídy čtyři. Pro každou obrazovku jedna (úvodní obrazovka, nastavení, seznam měřidel, obrazovka rozšířené reality a mapového podkladu).

3.3.3 Třídy reprezentující viewmodel:

Tyto třídy slouží pro uchování dat, důležitých pro vykreslení obrazovky. Třídy rozšiřují již zmíněnou komponentu ViewModel z balíčku Jetpack. Dále třídy slouží jako prostředník mezi view a presentery, které provádí různé operace aplikace.

3.3.4 Třídy reprezentující presenter:

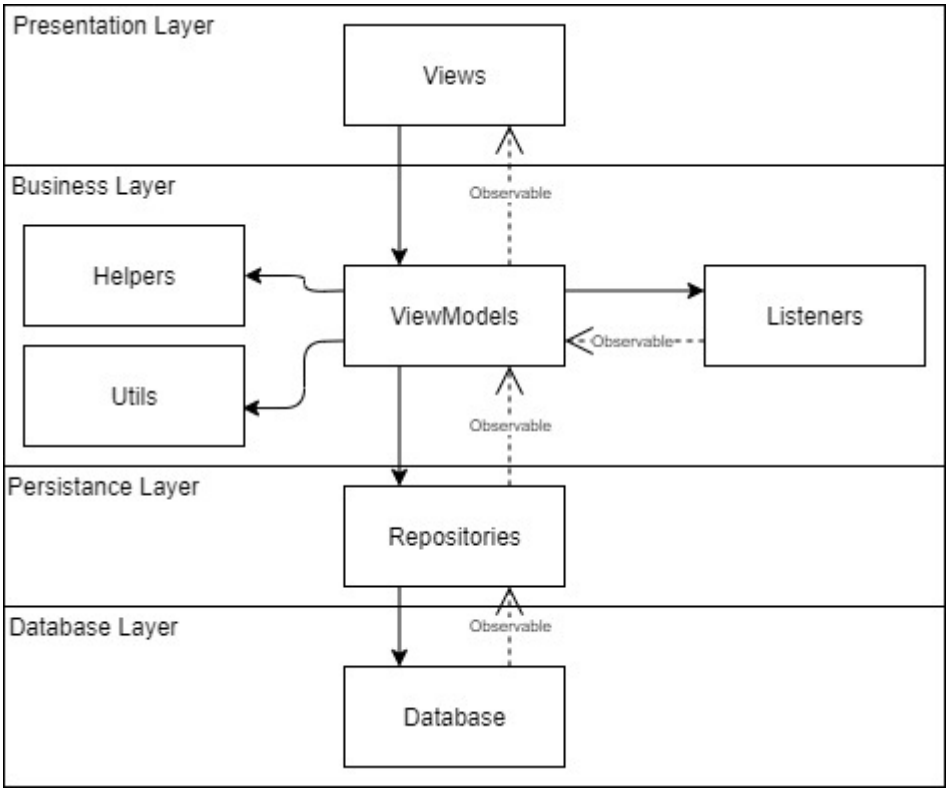
Třídy viewmodel by neměly obsahovat příliš mnoho kódu. Pouze kód pro zpracování požadavků z view a udržování jej v žádoucím stavu. Pro provádění výpočetních operací slouží takzvané presentery. Presentery provádí různé operace od modifikací proměnných po ty komplexnější, jako je například vytváření bitmapových objektů, zpracování údajů z databáze a jiné. Viewmodel za použití těchto presenterů pak vyhodnocuje data, která jsou následně předána view.

3.3.5 Třídy reprezentující model:

Model je reprezentací již konkrétních dat, získaných například z databází nebo souborů. Nejpoužívanějším příkladem bývá Uživatel. Uživatel má své jméno, příjmení a funkci. Třída, která bude reprezentovat tohoto uživatele, bude tedy obsahovat tyto proměnné: „jmeno“, „prijmeni“, „funkce“ a patřičné metody pro jejich získání a modifikaci. V tomto projektu jsou to například konkrétní body na mapě, či měřidla energií, jež jsou uchovávána v databázi s veškerými informacemi o nich.

3.3.6 Návrh vrstvení tříd

Využitím architektury MVVM je snadné dosáhnout rozdělení aplikace do několika vrstev. Každá vrstva má v dané aplikaci specifickou roli. [15] Třídy z jednotlivých vrstev mohou komunikovat pouze s třídami v sousedících vrstvách. V tomto projektu je tato restrikce ještě pozměněna tak, že třídy mohou komunikovat pouze s třídami, které se dle diagramu nachází v nižší vrstvě. Data z nižších vrstev jsou získávána pomocí návrhového vzoru observable. Návrhový vzor je implementován již zmíněnými Live Daty, která jsou pozorovatelná třídami vyšších vrstev. Díky tomu jsou třídy hned obeznámeny o změně dat, které se vyskytnou.



Obrázek 12 Návrh vrstvení tříd

Návrhový vzor observable je možné použít i v rámci jedné vrstvy. Tento jev je v diagramu možné vidět mezi třídami **ViewModels** a **Listeners**. Třídy Listeners, například SensorLiveDataListener, neustále získávají nová data, která jsou důležitá pro třídy ViewModels. Z toho důvodu je výhodné použít návrhový vzor observable, který bude tato data pozorovat, a díky tomu budou mít třídy ViewModels aktuální data. Dalším způsobem, jak je toho možné docílit, je implementací interface či callbacků. Použití těchto způsobů může ovšem vést k takzvanému jevu memory leak. Tento jev by mohl nastat v případě, kdy viewmodel, komunikuje s třídou SensorLiveDataListener pomocí rozhraní (interface). Pokud bude životní cyklus třídy view ukončen a viewmodel si bude stále držet vazbu na SensorLiveDataListener, tak nezanikne. SensorLiveDataListener bude neustále získávat již nepotřebná data o orientaci zařízení a tím bude udržovat viewmodel v paměti, což je nežádoucí. [16]

4 Implementace

Tato kapitola je zaměřena již na praktickou stránku práce a bude zde možné nalézt popis fungování některých tříd, ukázka zajímavého či důležitého zdrojového kódu a UML diagram tříd, popisující část aplikace. Jelikož se jedná o rozsáhlý projekt, není možné zde popsat všechny třídy.

4.1 Stěžejní třídy

4.1.1 Třída LocationLiveDataListener

V aplikaci založené na práci s lokací je nesmírně důležité, jakým způsobem bude tato lokace získávána. Android nabízí několik řešení. Lokaci je možné získat hned z několika různých zdrojů, kterými jsou GPS, Wi-Fi a Cell-ID. Od volby zdroje lokality se poté odráží přesnost, rychlost a energetická náročnost. [17] Dalším rozdělením by mohlo být získání lokality pomocí senzorů v zařízení, či použitím poslední známé lokace, kterou nabízí služby Google Play. Získání poslední známé lokace pomocí služeb Google Play je rychlé, ale v mnoha případech může být nepřesné a zároveň vyžaduje připojení k internetu. Z těchto důvodů bylo využito obou možností pro urychlení prvotního získání lokality a poté pro nejlepší přesnost.

4.1.2 Třída SensorLiveDataListener

Tato třída slouží pro získání hodnoty azimuth a pitch. Získávání údajů ze senzorů zařízení je žádané pouze v případě, je-li uživatel na obrazovce rozšířené reality, a proto tato třída také obsahuje metody pro spuštění a ukončení naslouchání senzorů. Tyto metody jsou automaticky vykonávány dle životního cyklu view. Díky tomu je aplikace méně náročná na baterii. Tento jev je stejný i pro třídu LocationLiveDataListener.

4.1.3 Třída RadarRenderer

Po neúspěšném hledání knihovny, která by poskytla metody pro snadné vytváření a úpravu bitmapových objektů, byla vytvořena tato třída. Třída slouží k úpravě bitmapového objektu, který reprezentuje radar zobrazená v rozhraní rozšířené reality. Pomocí třídy jsou na radar vykresleny body, reprezentující lokaci měřidel energií.

4.1.4 Třída InfoBoxRenderer

Tato třída rozšiřuje třídu View. InfoBoxRenderer je používán jako „kanvas“, na kterém jsou vykreslována požadovaná data. InfoBoxRendereru jsou posílány informace z třídy AugmentedRealityVM o měřidlech energií, které je možné vykreslit a rozdíl odchylek, který vypovídá o tom, kde se daná měřidla nachází vůči mobilnímu zařízení. Tato třída poté vypočítá polohu na obrazovce, kde mají být data vykreslena.

Tato poloha na obrazovce je počítána takto :

1. Spočítá, kolika procentům odpovídá rozdíl odchylky v porovnání s kružnicí

$$\text{rozdíl} = \frac{100 * \text{rozdíl odchylky}}{360}$$

2. Rozdíl je poté vynásoben s procentem vertikálního úhlu záběru kamery vůči kružnici

$$\text{umístění v úhlu záběru } = \frac{100 * \text{vertikální úhel záběru}}{360} * \text{rozdíl}$$

3. Když je známo umístění v úhlu záběru v procentech stačí již tuto hodnotu převést na procenta obrazovky zařízení a tím je získána hodnota v pixelech pro umístění středu okna měřidla energie.

$$\text{pozice x} = \frac{\text{šířka obrazovky}}{2} + \text{ umístění v úhlu záběru} * 4$$

Nakonec třída vytvoří bitmapová informační okna, která obsahují určitý text a styl. Tato informační okna jsou poté vykreslena na „kanvas“.

4.1.5 Třída AugmentedRealityVM

Třída slouží jako spojka mezi výše uvedenými třídami. Z údajů získaných třídami SensorLiveDataListener a LocationLiveDataListener vypočítá, zda jsou měřidla energií v dosahu určeném maximální vzdáleností pro zobrazení a směr, kterým jsou situována. Proces je rozdělen do těchto kroků:

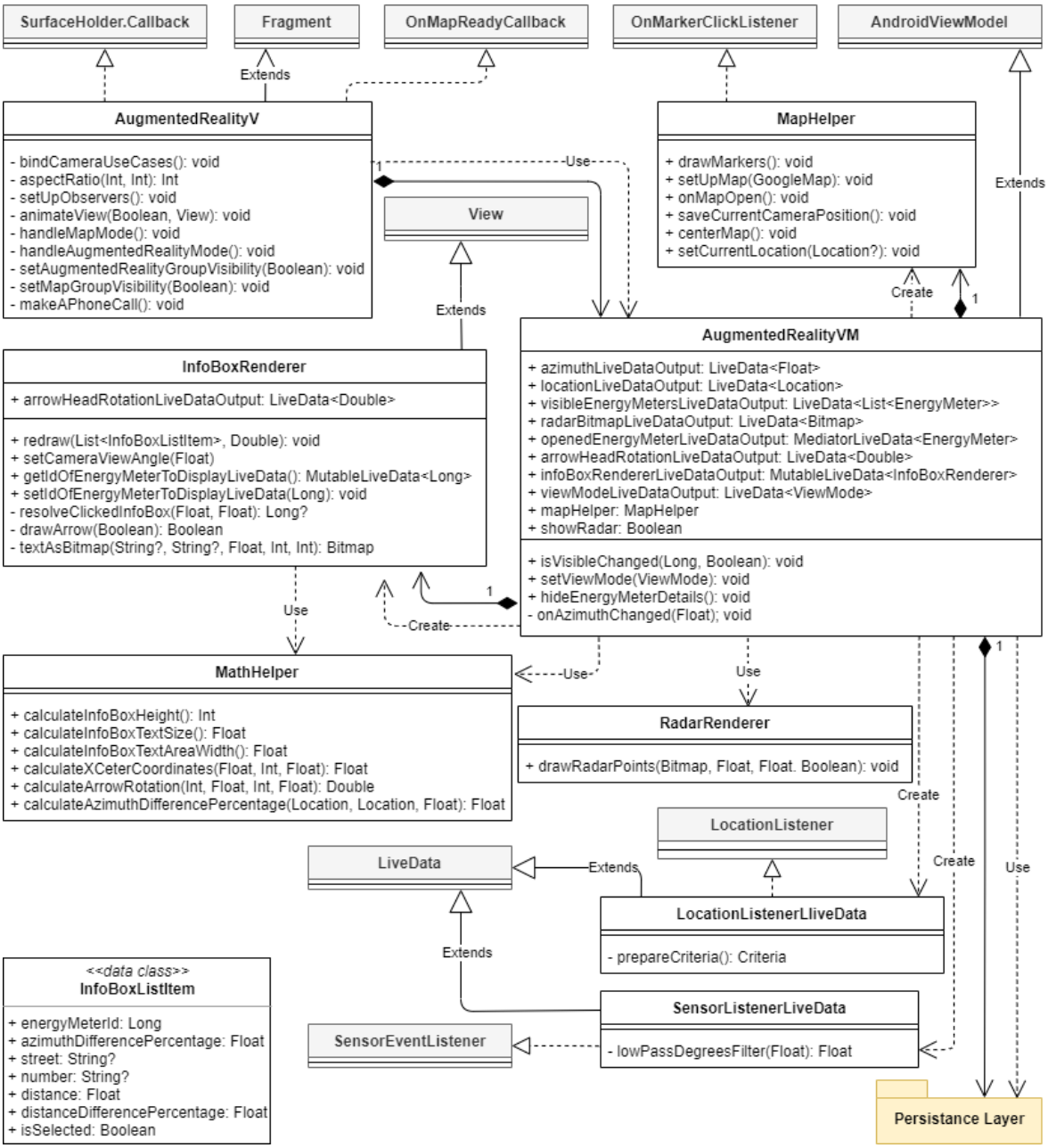
1. Získání aktuální zeměpisné polohy uživatele pomocí třídy LocationLiveDataListener.
2. Iterování seznamem měřidel energií a výběr těch, která jsou v zadaném dosahu.
3. Získání informací o tom, kterým směrem je zařízení orientováno pomocí třídy SensorLiveDataListener.
4. Vypočítání stupně odchylky měřidel energií v dosahu od zařízení pomocí již existující metody **bearingTo** z knihovny **android.location**.
5. Odečtení odchylky jednotlivých měřidel od hodnoty azimuth.

   Pokud je rozdíl odchylek více než 180 odečte 360, naopak pokud je rozdíl odchylek menší než -180, přičte 360.

4.2 UML diagramy

UML diagram tříd této aplikace je příliš velký na to, aby byl v této práci vyobrazen celý. Z tohoto důvodu zde budou popsány UML diagramem pouze dvě části aplikace.

4.2.1 UML diagram rozšířené reality



Obrázek 13 UML Diagram rozšířené reality

Tento diagram reprezentuje část aplikace implementující rozšířenou realitu a mapový podklad. Diagram slouží k zobrazení metod a proměnných jednotlivých tříd a také k vyjádření vztahu vůči jiným třídám. Diagram bude krátce popsán pro jeho lepší pochopení. Prvky, jež jsou šedě

podbarveny, reprezentují třídy a rozhraní z knihoven, které nebyly vyvinuty v rámci tohoto projektu. Pro zlepšení přehlednosti nejsou metody ani proměnné těchto prvků zobrazeny v diagramu tříd. Dále nejsou zobrazeny privátní proměnné jednotlivých tříd a takzvané „override“ metody. Třída AugmentedRealityV rozšiřuje třídu Fragment a váže se na ní grafické rozhraní. Tato třída využívá jako svůj viewmodel instanci třídy AugmentedRealityVM, jež rozšiřuje třídu AndroidViewModel z balíčku Jetpack. Třída AugmentedRealityV definuje mapové rozhraní a rozhraní rozšířené reality zároveň. Rozhraní pracují s téměř totožnými údaji a v mnoha případech využívají stejných metod. Z tohoto důvodu obě rozhraní používají stejné view. Pokud by třídy nepoužívaly stejné view a instanci viewmodelu, musela by být data, jako je například aktuální lokace uživatele, při přechodu mezi danými rozhraními, znovu získána, a to by vedlo k nežádoucím časovým prodlevám.

Třída AugmentedRealityVM vytváří a používá třídy InfoBoxRenderer, RadarRenderer, MapHelper, LocationLiveDataListener, SensorLiveDataListener, MathHelper a třídu EnergyMeterRepository jež se nachází v prezenční vrstvě. Po získání dat z tříd LocationLiveDataListener, SensorLiveDataListener a databáze, pomocí návrhového vzoru observable, AugmentedRealityVM vyfiltruje měřidla energií, která se nacházejí pouze v žádané vzdálenosti a předá údaje třídě InfoBoxRenderer a RadarRenderer. InfoBoxRenderer zjistí pomocí třídy MathHelper umístění informačních oken na obrazovce a poté vykreslí data na „kanvas“, který je přes viewmodel předán třídě AugmentedRealityV a následně vykreslen na obrazovce mobilního zařízení. Třída RadarRenderer tato data vykreslí na radar, jenž je poté také na obrazovce zobrazen.

Pokud uživatel otevře některé z informačních oken, AugmentedRealityV zobrazí panel s informacemi o měřidle energií, které jsou získány pomocí databindingu z viewmodelu.

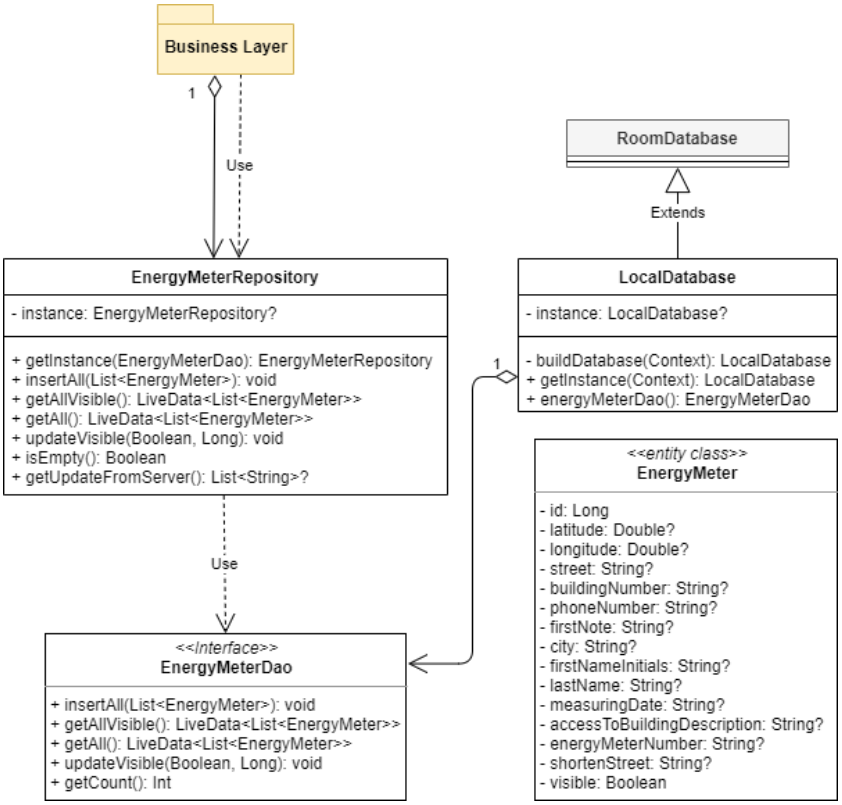
Třída MapHelper implementuje rozhraní **OnMarkerClickListener**. Tato třída je použita pro správu a komunikaci s mapovým rozhraním. Třída obsahuje například metody pro vykreslení elektroměrů na mapovém podkladu, vycentrování mapového podkladu na aktuální lokaci uživatele a jiné.

Dále stojí za povšimnutí, že třídy SensorLiveDataListener a LocationLiveDataListener rozšiřují třídu **LiveData** z balíčku **androidx.lifecycle**. Díky tomu jsou třídy pozorovatelné a je v nich možné reagovat na životní cyklus aplikace. Například je možné pozastavit činnost těchto tříd kdykoliv, kdy je aplikace spuštěna na pozadí mobilního zařízení.

4.2.2 UML diagram databázového přístupu

Dalším UML diagramem, který bude krátce popsán, je diagram zohledňující přístup k lokální databázi. Tento diagram neobsahuje tolik tříd jako předešlý, ale jedná se o důležitou část aplikace. Z tohoto důvodu je dobré mít představu o tom, jak daná část aplikace funguje. Základem je třída RoomDatabase z balíčku **androidx.room**. Tato třída je extendována třídou LocalDatabase. Třída LocalDatabase slouží k vytvoření databáze a popsání její struktury, poskytnutím tříd, jež popisují jednotlivé tabulky v databázi. Takovouto třídou je EnergyMeter, kterou lze vidět v diagramu tříd. Tato třída se na první pohled jeví jako datová třída s proměnnými a metodami, pro jejich získání a modifikaci. Metody pro přehlednost diagramu nejsou zobrazeny. Rozdíl oproti datovým třídám by byl ovšem viditelný při zobrazení zdrojového kódu této třídy. Třída je anotována anotací **@Entity** z knihovny **androix.room**. Díky této anotaci může třída LocalDatabase vytvořit tabulku odpovídající této entitní třídě. Databázové dotazy jsou vytvářeny pomocí rozhraní EnergyMeterDao. Toto rozhraní je anotováno anotací **@Dao**, což je zkratkou pro **data access object** (objekt pro přístup k datům). Metody v této třídě reprezentují jednotlivé databázové dotazy. Třída EnergyMeterRepository používá třídu implementující EnergyMeterDao pro získání dat z databáze, které poté předává aplikační vrstvě. Třída EnergyMeterRepository je singleton[8] a slouží jako jediný přístupový bod aplikační logiky k databázi. Tato třída také slouží k získávání dat z externích zdrojů, jako jsou například webové služby. V tomto projektu je toto získávání dat pouze simulováno a data jsou ve skutečnosti získávána ze souboru.

[8] Singleton je návrhový vzor pro vytvoření jedné globálně přístupné instance třídy



Obrázek 14 UML Diagram databázového přístupu.

4.3 Ukázka kódu

Tato část bakalářské práce se zabývá ukázkou implementace v podobě zdrojového kódu. Budou zde ukázky částí zdrojového kódu, u kterých bude popsáno, čím jsou zajímavé a jakou funkci v aplikaci plní.

4.3.1 Výpočet rozdílu hodnoty azimuth měřidel energií

```
private fun onAzimuthChanged(azimuth: Float, pitch: Double) {
    val infoBoxList = ArrayList<InfoBoxListItem>()
    if (currentLocation != null
        && visibleEnergyMetersLiveDataOutput.value != null) {
        val currentRadarBitmap =
            radarBitmap.copy(Bitmap.Config.ARGB_8888, true)
        for (energyMeter in visibleEnergyMetersLiveDataOutput.value!!) {
            val energyMeterLocation =
                LocationUtils.initializeLocation(
                    energyMeter.latitude!!,
                    energyMeter.longitude!!
                )
            val distanceToEnergyMeter =
                currentLocation!!.distanceTo(energyMeterLocation)
            if (distanceToEnergyMeter < maxViewDistance) {
                val distanceDifferencePercentage =
                    100 * distanceToEnergyMeter / maxViewDistance
                val isSelected = energyMeter.id ==
                        openedEnergyMeterLiveDataOutput.value?.id
                RadarRenderer.drawRadarPoints(
                    currentRadarBitmap,
                    distanceDifferencePercentage,
                    (currentLocation!!.
                    bearingTo(energyMeterLocation) + 360) % 360,
                    isSelected)
                infoBoxList.add(
                    InfoBoxListItem(
                        energyMeter.id,
                        MathHelper.calculateAzimuthDifferencePercentage(
                            currentLocation!!,
                            energyMeterLocation,
                            azimuth),
                        energyMeter.shortenStreet,
                        energyMeter.buildingNumber,
                        distanceToEnergyMeter,
                        distanceDifferencePercentage,
                        isSelected
                    )
                )
            }
            radarBitmapLiveData.postValue(currentRadarBitmap)
        }
    }
    infoBoxRenderer.redraw(infoBoxList, pitch)
}
```

Metoda je volána při každé změně hodnoty azimuth a její chování je krátce popsáno také v kapitole 4.1.5. Jedná se tedy o ukázku implementace již známého konceptu. Metoda si vytvoří list pro ukládání instancí třídy InfoBoxListItem. Tato datová třída nese důležité informace pro vykreslení informačních oken na obrazovce, jimiž je například hodnota identifikátoru měřidla, odchylka měřidla od aktuálního směru, kterým je objektiv mobilního zařízení směrován a jiné. Pokud aktuální lokace uživatele není nulová a zároveň jsou z databáze získána data, metoda iteruje všemi daty. V tomto momentu je vytvořeno kopie radaru, na kterou budou následně nanášeny jednotlivé body, reprezentující měřidla energií. Pro každé měřidlo energií je vypočítána vzdálenost a je-li výsledná vzdálenost menší než maximální vzdálenost pro zobrazení měřidel, je dané měřidlo přidáno do již připraveného listu a následně je zavolána metoda, jež dané měřidlo vykreslí na radar. Po skončení iterace je v třídě InfoBoxRenderer zavolána metoda redraw, které je předán naplněný list a která vykreslí informační okna na „kanvas“.

4.3.2 Výpočet souřadnic pro zobrazení informačního okna

```
private fun calculateXCenterCoordinates(azimuthDifference: Float): Float {
    val difference = 100 * azimuthDifference / 360
    val percentagePlacementInViewAngle =
        (100 * cameraViewAngle / 360) * difference

    return canvasWidth / 2 + percentagePlacementInViewAngle * 4
}
```

Tato metoda provádí operace popsané v kapitole 4.1.4. Metoda je volána pokaždé, když senzor zaznamená změnu. Metoda je volána pro každé z měřidel energií, jež je v listu, poskytnutém předešlou částí kódu, jež byla popsána.

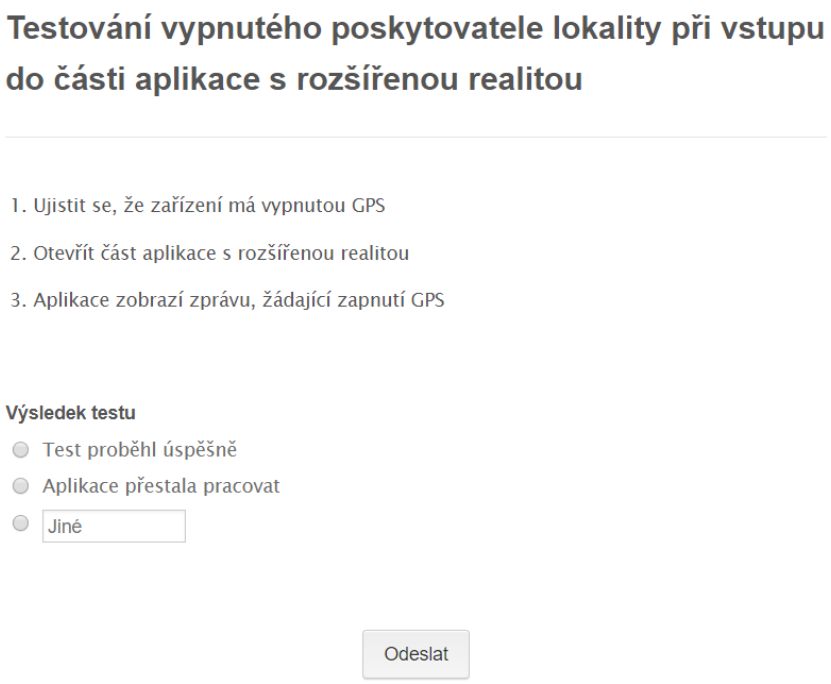
4.3.3 Dao interface

```
@Dao
interface EnergyMeterDao {

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insertAll(energyMeters: List<EnergyMeter>)

    @Query("SELECT * FROM energy_meter WHERE visible = 1")
    fun getAllVisible(): LiveData<List<EnergyMeter>>

    @Query("SELECT * FROM energy_meter")
    fun getAll(): LiveData<List<EnergyMeter>>

    @Query("UPDATE energy_meter SET visible=:visibleUpdate WHERE id=:id")
    suspend fun updateVisible(visibleUpdate: Boolean, id: Long)

    @Query("UPDATE energy_meter SET visible = :visibleUpdate")
    suspend fun updateAllVisible(visibleUpdate: Boolean)

    @Query("SELECT COUNT(id) FROM energy_meter")
    suspend fun getCount(): Int
}
```

V ukázce je možné vidět rozhraní, jež je anotováno anotací @Dao z knihovny Romm, a jednotlivé metody sloužící pro dotazování databáze jsou anotovány @Insert (vkládání nových dat do databáze), či @Query (získávání a modifikace již existujících dat), také z knihovny Room. Tyto dotazy jsou specifikovány v anotacích, například: „SELECT * FROM energy_meter“. Tento dotaz vybere všechny záznamy z tabulky energy_meter. Dále stojí za povšimnutí označení suspend, které je použito u některých metod. Toto označení říká, že metoda nemůže být volána z hlavního vlákna.

5 Testování

Testování bylo rozděleno na několik částí dle způsobu jejich provedení. První způsob lze nazvat „testování dle scénářů“. Tyto testy jsou prováděny na fyzických zařízeních. Testům předchází sepsání různých scénářů použití. Tyto scénáře jsou poté ve formě formulářů předány testerům, kteří postupují dle kroků, uvedených ve formuláři a zaznamenávají chování aplikace. Pro usnadnění předávání formulářů a zpětné vazby, byla použita služba JotForm. JotForm umožňuje vytváření online formulářů a jejich publikaci. Po každém vyplnění formuláře, je zaslán email autorovi tohoto formuláře. [18] Příklad jednoho z formulářů:



Obrázek 15 Příklad scénáře použití

Aplikace byla uživatelům distribuována za použití vývojářské platformy Firebase. Firebase je vlastněn společností Google a nabízí mnoho prostředků pro vývojáře mobilních aplikací. Jedním z těchto prostředků je například rozhraní App Distribution. Zde může vývojář snadno distribuovat aplikaci mezi své testery. [19]

Druhým typem jsou testy, které jsou prováděny ve vývojovém prostředí. Zde jsou použity unit testy, které jsou zaměřeny na jednotlivé metody a třídy provádějící logické operace. Například

pro rozhraní, jež slouží k získávání dat z databáze, mohou být napsány testy pro jednotlivé metody, které reprezentují jednotlivé databázové dotazy. Níže je uveden příklad Unit testu pro jednu z těchto metod.

```
@Before
fun createDatabase() {
    context = ApplicationProvider
                .getApplicationContext<Context>()
    localDatabase = Room
                .inMemoryDatabaseBuilder(context,
                        LocalDatabase::class.java)
                .build()
    energyMeterDao = localDatabase.energyMeterDao()
    // Data insertion code
}

@After
@Throws(IOException::class)
fun closeDb() {
    localDatabase.close()
}

@Test
fun testGetCount() {
    runBlocking {
        // When
        val count = energyMeterDao.getCount()
        // Then
        assertEquals(3, count)
    }
}
```

Jedná se o jednoduchý test, a proto jej není nutné příliš popisovat. Pro tento test je vytvořeno nové vlákno, v němž je zavolána metoda pro získání celkového počtu měřidel energií, jež jsou uloženy v lokální databázi. V této ukázce stojí za povšimnutí anotace **@Before** a **@After**. Pokud je metoda anotovaná anotací @Before, bude provedena před každou metodou s anotací **@Test** v dané třídě. V tomto případě je v takto anotované metodě prováděno vytvoření lokální databáze a její následné naplnění daty. Naopak v metodě anotované anotací @After, jež je prováděna po skončení každé metody s anotací @Test, je databáze smazána. Díky tomu se databáze, na začátku každého testu, nachází ve stejném stavu.

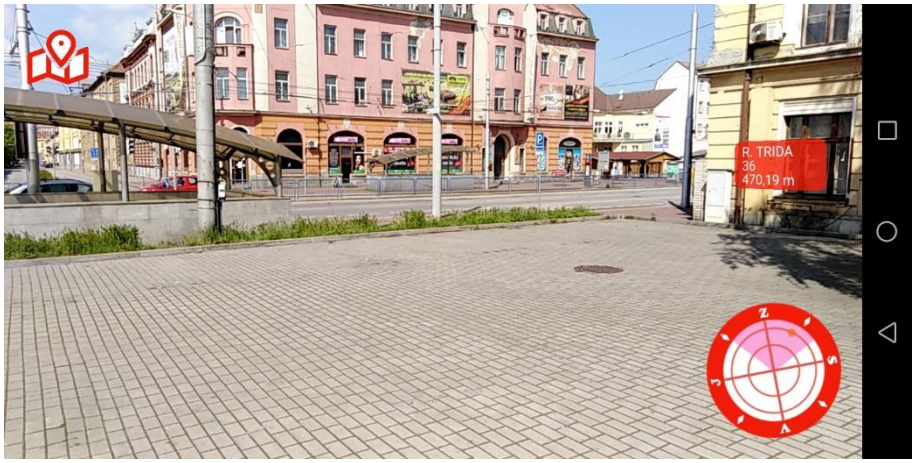
Pro testování uživatelského rozhraní byl použit framework „Espresso“, který byl vyvinut právě pro tyto případy.

```
@get:Rule
var activityRule: ActivityTestRule<MainActivity>
        = ActivityTestRule(MainActivity::class.java)


@Test
fun testNavigateToList() {
    val scenario = launchFragmentInContainer<HomeV>()
    scenario.onFragment {
        Navigation.setViewNavController(
            it.requireView(),
            navController)
    }
    validateElementVisibleAndPerformClick(R.id.toList)
    assertEquals(
        R.id.energyMeterListV,
        navController!!.currentDestination?.id)
}
```

Anotací **@Rule** je nastaveno pravidlo, určující, se kterou aktivitou je právě pracováno. V metodě „testNavigateToList“, po fragmentu, je vyhledáno tlačítko podle jeho identifikátoru, proběhne kontrola, zda je zobrazeno, a je simulováno jeho stisknutí. Po stisknutí se očekává zobrazení nového fragmentu.

Další způsob testování využívá služby Test Lab. Firebase Test Lab je cloudová infrastruktura pro testování aplikací. [20] Zde je vývojáři umožněno nahrát svou aplikaci, která je poté spuštěna na mnoha zařízeních. Tato zařízení pak simulují různá použití dané aplikace. Výsledkem je detailní zpráva o běhu aplikace na jednotlivých zařízeních, ve které jsou popsány všechny akce, jež byly provedeny, spolu s grafem, jenž je znázorňuje. Zpráva dále obsahuje video průběhu testování spolu se záznamy obrazu.

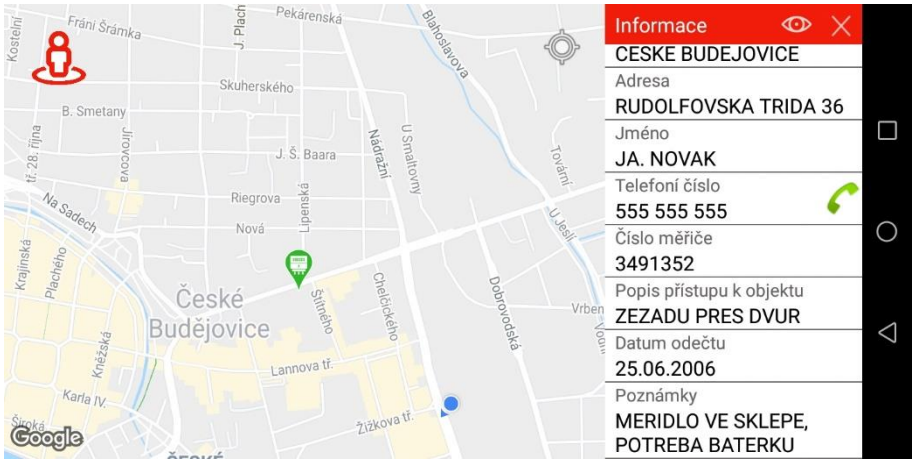
Nakonec byla výsledná aplikace otestována jako celek. Tento test byl proveden z pohledu pracovníka distribuční společnosti, kterému byla přidělena oblast, v níž se nachází měřidla energií, jež nebyla odečtena dálkově. Po příjezdu na dané místo byla aplikace spuštěna. Díky radaru, kterým aplikace disponuje, bylo zjevné, kterým směrem je měřidlo energií situováno.



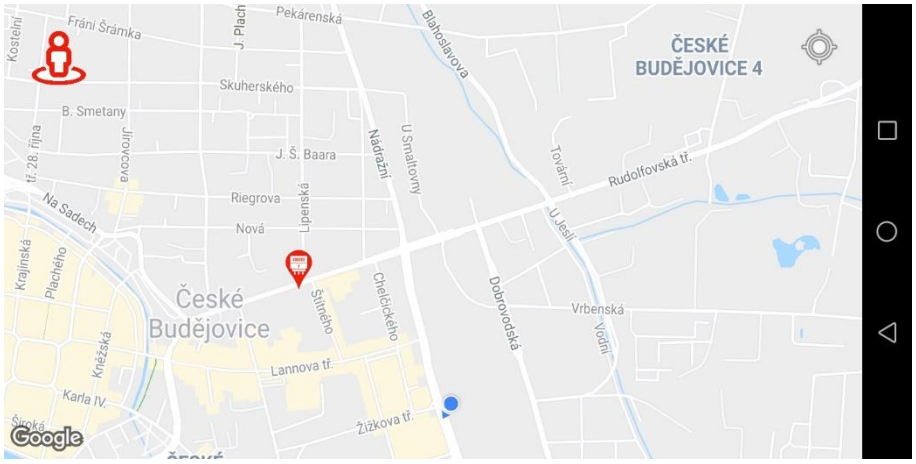
Obrázek 16 Rozhraní rozšířené reality

Obrázek 17 Rozhraní rozšířené reality s detailem měřidla energií

Aplikace dále umožnila zobrazení detailních informací o měřidle energií, či zobrazení mapového rozhraní, na kterém bylo měřidlo energií vykresleno.



Obrázek 18 Mapové rozhraní s detailem měřidla energií



Obrázek 19 Mapové rozhraní

Po příchodu k objektu bylo nutné ujistit se, zda se skutečně jedná o správnou budovu. Toho bylo dosaženo získáním čísla popisného z informačního okna, jež rozšířená realita zobrazuje. Také bylo zjištěno, zda není zapotřebí speciálních nástrojů (svítilna pro temná prostředí, žebřík apod.), pro přístup k danému měřidlu energií. Tato informace byla získána z dodatečných údajů, zobrazených po interakci s informačním oknem.

Obrázek 20 Rozhraní rozšířené reality před objektem

Poté již stačilo požádat o přístup do objektu, čehož mohlo být docíleno například pomocí telefonního hovoru, zahájeného stisknutím tlačítka v části obrazovky zobrazující detailní údaje o měřidle. Posledním krokem bylo lokalizování měřidla energií, jež se v daném objektu nachází.

Obrázek 21 Rozhraní rozšířené reality u měřidla energií

6 Diskuze

Tato kapitola pojednává o silných a slabých stránkách tohoto projektu.

Za silné stránky projektu považuji to, že nebyly použity knihovny a frameworky třetích stran pro implementaci rozšířené reality. Použití těchto frameworků by bylo jednak finančně nákladné, v budoucnu by mohlo dojít k nekompatibilitě při vydání nové verze některé z komponent a také tím vývojář ztrácí jistou kontrolu nad zdrojovým kódem aplikace. Za další klad této práce považuji využití komponent z balíčku Jatpack, o kterých si myslím, že do budoucna usnadní vývojářům práci a povedou k psaní efektivnějšího kódu.

Jedním z nedostatků této práce je designe, na který nebyl kladen příliš velký důraz. Dalším nedostatkem je, že aplikace byla vyvíjena pouze pro operační systém Android a pokud by do budoucna měla být přístupná i pro zařízení s jiným operačním systémem, musela by být znovu vyvíjena pro daný systém. Nakonec, za částečný nedostatek této práce považuji použití služeb třetí strany pro logování událostí. Vývoj vlastních logovacích služeb by byl možný, ale domnívám se, že by výrazně přesáhl rámec zadání této bakalářské práce.

7 Závěr

Výsledkem této bakalářské práce je aplikace pro mobilní telefony Android, která usnadní zaměstnancům distribučních společností práci při provádění odečtů energií. Aplikace implementuje rozšířenou realitu dle zadání a kritérií, definovaných ve funkčních požadavcích aplikace. Dále umožňuje interakci s prvky rozšířené reality, zobrazení měřidel energií na mapovém podkladu, zobrazení seznamu všech měřidel energií a zahájení telefonního hovoru na číslo uvedené u měřidel.

Byla provedena analýza různých technologií a postupů, které mohly být alternativou pro vývoj daného projektu. Na základě této analýzy byly poté zvoleny, již výše zmíněné prostředky.

Projekt byl vyvíjen s ohledem na objektově orientovaný návrh a architekturu MVVM, díky čemuž bude do budoucna snadno udržovatelný a rozšiřitelný o další funkcionality.

8 Seznam použité literatury

[1] Dálkový odečet elektřiny se postupně stane standardem, od roku 2027 by mohl být povinný. *Hybrid.cz* [online]. Chamanne s.r.o. 2018 [cit. 2020-08-03]. ISSN 1802-5323. Dostupné z: http://www.hybrid.cz/dalkovy-odecet-elektriny-se-postupne-stane-standardem-od-roku-2027-mohl-byt-povinny

[2] Google Play: WAR – Widespread Augmented Reality II. *Play.google.com* [online]. Los Angeles: Mansfield Ave, 2020 [cit. 2020-03-08]. Dostupné z: https://play.google.com/store/apps/details?id=org.warmixare2

[3] Augmented Reality Map. *Play.google.com* [online]. Google Commerce Ltd, 2016 [cit. 2020-03-08]. Dostupné z: https://play.google.com/store/apps/details?id=gklapp.armap&rdid=gklapp.armap

[4] LINOWES, Jonathan a Krystian BABILINSKI. *Augmented Reality for Developers: Build practical augmented reality applications with Unity, ARCore, ARKit, and Vuforia*. Brimingham: Packet publishing, 2017. ISBN 978-1-78728-643-6.

[5] AUKSTAKALNIS, Steve. *Practical augmented reality: a guide to the technologies, applications and human factors for ar and vr*. Boston: Addison-Wesley, 2017. ISBN 978-013-4094-236.

[6] Augmented reality. *Wikipedia.org* [online]. 2020 [cit. 2020-03-08]. Dostupné z: https://en.wikipedia.org/wiki/Augmented_reality

[7] SCHMALSTIEG, Dieter a Tobias HÖLLERER. *Augmented reality: principles and practice*. Boston: Addison-Wesley, 2016. ISBN 978-0-321-88357-5.

[8] Senzory v mobilních telefonech od A do Z. *Beryko.cz* [online]. 2020 [cit. 2020-03-08]. Dostupné z: https://www.beryko.cz/blog/recenze/senzory-v-mobilnich-telefonech-od-a-do-z.html

[9] Data Binding Library. *Developer.android.com* [online]. 2019 [cit. 2020-03-08]. Dostupné z: https://developer.android.com/topic/libraries/data-binding/

[10] LiveData Overview. *Developer.android.com* [online]. 2020 [cit. 2020-03-08] Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

[11] ViewModel Overview. *Developer.android.com* [online]. 2020 [cit. 2020-03-08] Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel

[12] Navigation. *Developer.android.com* [online]. 2020 [cit. 2020-03-08] Dostupné z: https://developer.android.com/topic/libraries/architecture/navigation/

[13] Room Persistence Library. *Developer.android.com* [online]. 2020 [cit. 2020-03-08] Dostupné z: https://developer.android.com/topic/libraries/architecture/room

[14] *It-slovnik.cz* [online]. [cit. 2020-08-03] Dostupné z: https://it-slovnik.cz/pojem/logovani

[15] VERRECKT, Jeffrey. Layered Architecture. *Thinktocode.com* [online]. 2018 [cit. 2020-03-08]. Dostupné z: https://www.thinktocode.com/2018/07/05/layered-architecture/

[16] ViewModels and LiveData: Patterns + AntiPatterns. *Medium.com* [online]. 2017 [cit. 2020-08-03] Dostupné z: https://medium.com/androiddevelopers/viewmodels-and-livedata-patterns-antipatterns-21efaef74a54

[17] Location. *Developer.android.com* [online]. 2020 [cit. 2020-03-08] Dostupné z: https://developer.android.com/guide/topics/location/strategies

[18] *Eu.JotForm.com* [online]. 2020 [cit. 2020-08-03] Dostupné z: https://eu.jotform.com/

[19] Firebase App Distribution *Firebase.google.com* [online]. 2019 [cit. 2020-08-03] Dostupné z: https://firebase.google.com/products/app-distribution

[20] Firebase Test Lab. *Firebase.google.com* [online]. 2019 [cit. 2020-08-03] Dostupné z: https://firebase.google.com/docs/test-lab

9 Seznam obrázků

10 Seznam příloh

1. Elektronická verze textu bakalářské práce
2. Komprimovaný soubor obsahující zdrojové kódy projektu
3. Uživatelská dokumentace