FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Vizualizace rozvodů elektřiny a vody pomocí rozšířené reality
Student:	Daniel Ridzoň
Vedoucí:	Ing. Lukáš Hromadník
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat mobilní aplikaci pro operační systém iOS, která bude zobrazovat umístění rozvodů elektřiny a vody ve zdech bytu pomocí rozšířené reality. Aplikace bude podporovat kromě samotné vizualizace jednotlivých rozvodů i jejich vytváření v rozšířené realitě a bude podporovat jejich ukládání.

1. Analyzujte existující aplikace pro vizualizaci rozvodů v bytě.
2. Na základě analýzy specifikujte funkční a nefunkční požadavky budoucí aplikace.
3. Vytvořte návrh uživatelského rozhraní a návrh softwarového řešení.
4. Na základě analýzy a návrhů implementujte funkční prototyp.
5. Proveďte uživatelské testování prototypu a analyzujte výsledky.
6. Shrňte výsledek práce a popište její přínosy.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 15. září 2022 v Praze.

Bakalářská práce

VIZUALIZACE ROZVODŮ ELEKTŘINY A VODY POMOCÍ ROZŠÍŘENÉ REALITY

Daniel Ridzoň

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Lukáš Hromadník
11. května 2023

Odkaz na tuto práci: Ridzoň Daniel. *Vizualizace rozvodů elektřiny a vody pomocí rozšířené reality.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Poděkování x

Prohlášení xi

Abstrakt xii

Seznam zkratek xiii

1 Úvod 1

2 Analýza 3

2.1 Rozšířená realita . . . 3

2.1.1 Rozšířená realita v iOS . . . 3

2.1.2 Proč vizualizovat rozvody v AR? . . . 4

2.1.3 Vizuálně-inerciální odometrie . . . 4

2.2 Analýza konvenčních metod zakreslení rozvodů . . . 4

2.2.1 AutoCAD . . . 5

2.2.2 CADKON+ MEP . . . 5

2.2.3 ProfiCAD . . . 5

2.2.4 PlumbingCAD . . . 5

2.2.5 PlumbersCAD . . . 6

2.2.6 Shrnutí konvenčních metod . . . 6

2.3 Analýza existujících aplikací pro vizualizaci rozvodů v AR . . . 6

2.3.1 Measure . . . 7

2.3.2 RoomScan Classic . . . 8

2.3.3 magicplan . . . 9

2.3.4 vGIS . . . 11

2.3.5 spectar . . . 11

2.3.6 Shrnutí analýzy dostupných AR aplikací . . . 12

2.4 Požadavky . . . 13

2.4.1 Funkční požadavky . . . 13

2.4.2 Nefunkční požadavky . . . 14

3 Technologie 15

3.1 Figma . . . 15

3.2 iOS . . . 16

3.3 Swift . . . 16
3.4 UI framework . . . 17
3.4.1 SwiftUI . . . 17
3.4.2 UIKit . . . 17
3.4.3 Zvolený framework . . . 18
3.5 ARKit . . . 19
3.5.1 Systém souřadnic . . . 19
3.5.2 Transformační matice . . . 19
3.5.3 ARAnchor . . . 20
3.5.4 Detekce rovin . . . 20
3.5.5 Detekce obrázků . . . 20
3.5.6 Ideální referenční obrázek . . . 21
3.5.7 AR konfigurace . . . 24
3.5.8 Zvolené nastavení frameworku ARKit . . . 27
3.6 Framework pro tvorbu 3D obsahu pro ARKit . . . 27
3.6.1 RealityKit . . . 27
3.6.2 SceneKit . . . 28
3.6.3 SpriteKit . . . 28
3.6.4 Metal . . . 28
3.6.5 Zvolený framework . . . 28

4 Návrh **29**
4.1 UIKit navigace . . . 29
4.2 Návrhový vzor . . . 30
4.2.1 Problémy návrhového vzoru MVC v UIKit aplikacích . . . 30
4.2.2 Návrhový vzor MVVM . . . 31
4.3 Flow koordinátory . . . 32
4.4 Název a ikona aplikace . . . 33
4.4.1 Ikona . . . 33
4.4.2 Název . . . 34
4.5 Případy užití . . . 35
4.5.1 UC1 – Nastavení referenčního obrázku . . . 35
4.5.2 UC1.1 – Nastavení vlastní referenčního obrázku . . . 35
4.5.3 UC1.2 – Nastavení výchozího referenčního obrázku . . . 35
4.5.4 UC2 – Seznam místností . . . 35
4.5.5 UC3 – Detail místnosti . . . 35
4.5.6 UC4 – Detail zdi . . . 36
4.5.7 UC5 – Detail obrázku . . . 36
4.5.8 UC6 – AR editor rozvodů . . . 36
4.5.9 UC7 – Nastavení . . . 37
4.6 Uživatelské rozhraní . . . 37
4.6.1 Hlavní struktura aplikace . . . 37

4.6.2 První nastavení . . . 37
4.6.3 První nastavení – Vlastní referenční obrázek . . . 38
4.6.4 První nastavení – Výchozí referenční obrázek . . . 38
4.6.5 Seznam místností . . . 39
4.6.6 Nastavení . . . 39
4.6.7 Detail místnosti . . . 40
4.6.8 Detail zdi . . . 40
4.6.9 AR editor rozvodů . . . 42

5 Implementace 45
5.1 Struktura projektu . . . 45
5.2 Perzistence dat . . . 46
5.2.1 UserDefaults . . . 46
5.2.2 CoreData . . . 47
5.3 Repository . . . 49
5.4 Lokalizace . . . 50
5.4.1 Registrace textových řetězců . . . 51
5.5 Obrazovky . . . 51
5.5.1 FlowCoordinator . . . 52
5.5.2 View . . . 52
5.5.3 ViewController . . . 52
5.5.4 ViewModel . . . 53
5.6 AR editor rozvodů . . . 53
5.7 SwiftUI ovládací prvky editoru . . . 54
5.8 Komunikace SwiftUI s ARView pomocí Combine . . . 54
5.8.1 Publikace a odběr zpráv v Combine . . . 54
5.8.2 ARAction enum . . . 55
5.9 Implementace ARView . . . 56
5.9.1 Úvod do funkčnosti editoru . . . 56
5.9.2 Struktura souborů editoru . . . 56
5.9.3 IT_ARView . . . 56
5.9.4 IT_ARView+TapHandler . . . 59
5.9.5 IT_ARViewController . . . 59
5.9.6 IT_ARViewController+ARSessionDelegate . . . 62
5.9.7 IT_ARViewControllerRepresentable . . . 65
5.9.8 IT_ViewModel . . . 65

6 Testování 67
6.1 Zásady uživatelského testování použitelnosti . . . 67
6.2 Scénáře uživatelského testování . . . 68
6.3 Výběr účastníků testu . . . 69
6.3.1 Popis jednotlivých účastníků . . . 70

6.4 Průběh uživatelského testování 70
6.5 Výsledky uživatelského testování 71
6.5.1 1. skupina účastníků 71
6.5.2 2. skupina účastníků 72
6.5.3 3. skupina účastníků 73
6.5.4 4. skupina účastníků 73
6.5.5 Shrnutí výsledků uživatelského testování 74

7 Závěr **75**

A Ukázky diagramů rozvodů elektřiny a vody v CAD programech **77**

B Naměřená data vzdálenosti detekce referenčních obrázků **83**
B.1 Test 1 – ideální referenční obrázek – naměřená data 83
B.2 Test 2 – vliv velikosti obrázku na maximální vzdálenost detekce – naměřená data 84

C Scénáře uživatelského testování – instrukce **85**

D Formulář pro účastníky uživatelského testování použitelnosti **91**

E Snímky obrazovky výsledné aplikace **93**

Obsah přiloženého datového archivu **103**

Seznam obrázků

2.1 Ukázka aplikace *Measure* . . . 8
2.2 Ukázka aplikace *RoomScan Classic* . . . 9
2.3 Ukázka aplikace *magicplan* . . . 10
2.4 Ukázka aplikace *vGIS* . . . 11
2.5 Ukázka aplikace *spectar* . . . 12

3.1 Výchozí systém souřadnic ve frameworku *ARKit* . . . 19
3.2 Referenční obrázek A – šachovnice . . . 22
3.3 Referenční obrázek B – *Brosvision marker* . . . 22
3.4 Referenční obrázek C – *ArUco marker* . . . 22
3.5 Referenční obrázek D – QR kód . . . 22
3.6 Referenční obrázek E – fotografie . . . 22
3.7 Referenční obrázek F – fotografie . . . 22
3.8 Graf maximální vzdálenosti začátku detekce referenčního obrázku . . . 23
3.9 Graf vzdálenosti konce detekce referenčního obrázku . . . 23
3.10 Graf závislosti začátku detekce referenčního obrázku na velikosti obrázku . . . 24
3.11 Graf závislosti konce detekce referenčního obrázku na velikosti obrázku . . . 25

4.1 Diagram návrhového vzoru MVC . . . 30
4.2 Typické vazby v MVC uvnitř *UIKit* aplikací . . . 30
4.3 Diagram návrhového vzoru MVVM v *UIKit* aplikaci . . . 31
4.4 Návrh ikony aplikace . . . 34
4.5 Ikona a název aplikace nainstalované v systému *iOS* . . . 34
4.6 Návrh obrazovky – nastavení referenčního obrázku . . . 38
4.7 Návrh obrazovky – seznam místností . . . 39
4.8 Návrh obrazovky – nastavení . . . 40
4.9 Návrh obrazovky – detail místnosti . . . 41
4.10 Návrh obrazovky – detail zdi . . . 41
4.11 Návrh obrazovky – AR editor rozvodů (vliv nastavení) . . . 43
4.12 Návrh obrazovky – AR editor rozvodů (editace diagramu rozvodů) . . . 44

5.1 ERD schéma *CoreData* modelů . . . 49
5.2 Snímky obrazovky vykresleného 3D obsahu . . . 58

A.1 Ukázka návrhu rozvodů elektřiny v programu *AutoCAD* . . . 78

A.2 Ukázka návrhu odpadního potrubí v programu *AutoCAD* 78
A.3 Ukázka návrhu rozvodů elektřiny v programu *CADKON+ MEP* 79
A.4 Ukázka návrhu rozvodů pro teplou a studenou vodu v programu *CADKON+ MEP* 79
A.5 Ukázka návrhu rozvodů elektřiny v programu *ProfiCAD* 80
A.6 Ukázka návrhu rozvodů vody v programu *PlumbingCAD* 80
A.7 Ukázka návrhu potrubí v programu *PlumbersCAD* 81

E.1 Snímky obrazovky nastavení kotvícího obrázku 93
E.2 Snímky obrazovky seznamu místností a detailu místnosti 94
E.3 Snímky obrazovky detailu zdi 94
E.4 Snímky obrazovky AR editoru rozvodů 95
E.5 Snímky obrazovky nastavení 95

Seznam tabulek

B.1 Vzdálenost prvotní detekce u odlišných referenčních obrázků 83
B.2 Vzdálenost konce detekce odlišných referenčních obrázků 83
B.3 Vzdálenost začátku detekce referenčního obrázku v závislosti na velikosti obrázku 84
B.4 Vzdálenost konce detekce referenčního obrázku v závislosti na velikosti obrázku . 84

Seznam výpisů kódu

1 *Hello world!* ve frameworku *SwiftUI* 17
2 *Hello world!* ve frameworku *UIKit* 18
3 Inicializace výchozích hodnot nastavení aplikace v *UserDefaults* 46
4 Ukázka použití *@AppStorage* wrapperu 47
5 Ukázka načtení a uložení dat do *UserDefaults* 48
6 Ukázka překladů ve slovníku *Localizable.strings* 50
7 Protokol definující rozhraní *flow* koordinátora 52
8 Protokol definující rozhraní *ViewModelu* obrazovky detailu místnosti 53
9 Implementace třídy *ARActionManager* 54
10 Implementace publikace zpráv v *Combine* frameworku 55
11 Implementace odběru zpráv v *Combine* frameworku 55
12 Implementace *ARAction enum* 55

13 Výběr kotvy roviny zdi za detekovaným referenčním obrázkem 63

Rád bych upřímně poděkoval Ing. Lukáši Hromadníkovi za vedení mé práce, věnovaný čas a cenné rady ve zpětných vazbách. Mé díky patří i rodině, kamarádům a přítelkyni, kteří mě během studia a psaní práce podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č.121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 Daniel Ridzoň

Abstrakt

Bakalářská práce se zabývá návrhem a implementací iOS aplikace, která pomocí rozšířené reality vizualizuje polohu rozvodů elektřiny a vody ve zdech bytu. Práce analyzuje existující řešení tohoto problému. Následně jsou definovány funkční a nefunkční požadavky aplikace, je proveden výběr vhodných technologií a je navrženo uživatelské rozhraní a architektura aplikace. Na základě analýzy a návrhu je vytvořena funkční iOS aplikace, jejíž uživatelské rozhraní je implementováno ve frameworku SwiftUI a implementace rozšířené reality je provedena kombinací frameworků ARKit a RealityKit. Výsledkem práce je funkční aplikace, ve které lze pomocí rozšířené reality zaznamenat a později znovu zobrazit informace o rozvodech v jednotlivých zdech bytu.

Klíčová slova vizualizace bytových rozvodů, mobilní aplikace, iOS, rozšířená realita, ARKit, SwiftUI

Abstract

This bachelor thesis deals with the design and implementation of an iOS application that uses augmented reality to visualize the location of electricity and water distribution in the walls of an apartment. The thesis analyzes existing solutions to this problem. Subsequently, the functional and non-functional requirements of the application are defined, the selection of appropriate technologies is made and the user interface and architecture of the application is designed. Based on the analysis and design, a functional iOS application is created, whose user interface is implemented in the SwiftUI framework and augmented reality implementation is done by combining the ARKit and RealityKit frameworks. The result of the work is a functional application in which augmented reality can be used to record and later re-display information about the wiring in each wall of the apartment.

Keywords plumbing and wiring visualization, mobile app, iOS, augumented reality, ARKit, SwiftUI

Seznam zkratek

AR	Augumented Reality
BIM	Building Information Modeling
CAD	Computer Aided Design
ERD	Entity Relationship Diagram
GPS	Global Positioning System
GUI	Graphical User Interface
ID	Identification
IMU	Inertial Measurement Unit
JPEG	Joint Photographic Experts Group
LiDAR	Light Detection And Ranging
MR	Mixed Reality
MVC	Model View Controller
MVVM	Model View View-Model
OS	Operating System
PDF	Portable Document Format
PEX	Cross-linked polyethylene
QR	Quick Response
SDK	Software Development Kit
UC	Use Case
UI	User Interface
VIO	Visual-Inertial Odometry
VR	Virtual Reality

Kapitola 1

Úvod

Znalost umístění rozvodů elektřiny a vody ve zdech bytu je častým nedostatkem vlastníků a obyvatel bytů. Přestože někteří vlastníci mohou mít k dispozici projektovou dokumentaci daného objektu, nemusí to nutně znamenat, že pomocí informací obsažených v dokumentaci jsou schopni lokalizovat jednotlivé rozvody uvnitř zdí, jelikož projektová dokumentace slouží primárně pro řemeslníky, u kterých se počítá se znalostmi platných stavebních norem, které uvádějí např. kudy vedou zóny pro položení elektrických rozvodů. Umístění rozvodů je však nutné znát pro prevenci nehod při úpravách bytu, jež zahrnují operaci vrtání do zdi (např. věšení poličky, montáž nástěnného držáku na televizi či ukotvení závěsného nábytku).

Tato problematika je v práci vyřešena vytvořením mobilní aplikace, pomocí které lze zobrazit polohu jednotlivých rozvodů v prostředí rozšířené reality. Uživatelé této aplikace budou mít možnost jednoduše uložit schéma rozvodů ve zdi bez jakýchkoliv odborných znalostí, které jsou potřeba při použití konvenčních metod zakreslení. Jedinou nutnou podmínkou pro použití této aplikace bude znalost, kudy jednotlivé rozvody vedou. Bakalářská práce se metodikou získávání reálné polohy rozvodů zabývat nebude. Funkčnost aplikace však nebude ovlivněna změnou textury samotné zdi, tak se jako jedno řešení nabízí zachycení schématu v okamžik, kdy jsou rozvody položené, ale zatím nezakryté.

Motivací pro výběr tohoto tématu byla probíhající rekonstrukce kuchyně, při které si autor práce uvědomil, že je nutné nové změny v rozvodech elektřiny a vody zaznamenat, aby bylo možné provedené změny zpětně za několik let dohledat, například až se budou provádět další stavební úpravy.

První část práce se zabývá analýzou existujících způsobů pro zakreslení plánu rozvodů a analýzou aplikací pro vizualizaci rozvodů v bytě. Na základě této analýzy jsou specifikovány funkční a nefunkční požadavky aplikace. Druhá část práce se věnuje popisu technologií, návrhu uživatelského rozhraní aplikace a návrhu softwarového řešení. Poté následuje kapitola, která popisuje implementaci a jednotlivé dílčí části výsledné aplikace. Závěr práce se věnuje uživatelskému testování aplikace a shrnutí jejích přínosů.

Hlavním cílem práce je navržení a implementace mobilní aplikace pro operační systém iOS, která zobrazuje umístění rozvodů elektřiny a vody ve zdech bytu pomocí rozšířené reality. Aplikace umožňuje kromě samotné vizualizace jednotlivých rozvodů i jejich vytváření v rozšířené realitě a podporuje jejich ukládání.

Kapitola 2

Analýza

V této kapitole je provedena analýza technologie AR, analýza stavebních norem, analýza konvenčních metod zakreslení rozvodů a analýza existujících AR aplikací pro vizualizaci rozvodů v bytě. V závěru kapitoly jsou specifikovány funkční a nefunkční požadavky aplikace.

2.1 Rozšířená realita

Rozšířená realita (dále jen AR[1]) je důležitou částí práce, proto je zde uvedena její definice z Encyklopedie Multimédií: AR umožňuje uživateli vidět skutečný svět, který je překrytý a doplněný počítačově generovanými informacemi. […] Kombinace 3D objektů s reálným světem je užitečná v tom, že zlepšuje uživatelské vnímání reálného světa a interakci s ním.[1][2] AR se v dnešní době využívá jak v pracovním prostředí (např. v lékařství, maloobchodním prodeji či produktovém designu), kde zvyšuje efektivitu a bezpečnost práce, tak i v zábavním průmyslu, kde AR hry vkládají virtuální herní postavy do reálného světa v okolí hráče [2].

Pro realistické vnímání virtuálních objektů v reálném prostředí je nutné, aby virtuální objekty byly pevně ukotvené do skutečného světa, aby se při pohybu zařízení zdálo, že jsou napevno usazené v reálném světě. Toho je docíleno tím, že zařízení pomocí různých senzorů měří svůj pohyb a rotaci vzhledem k okolnímu světu a následně tyto veličiny aplikuje na kameru ve virtuálním světě. Obraz z reálné kamery je následně překryt obrazem z kamery virtuálního světa a pokud byl změřen posun a rotace kamery dostatečně přesně, virtuální objekty se na výsledném obrazu zdají, že jsou součástí skutečného světa.

2.1.1 Rozšířená realita v iOS

Na telefonech s operačním systémem iOS se pro sledování okolí využívá *vizuálně-inerciální odometrie* (viz 2.1.3) [3]. AR funkce jsou implementované v iOS pomocí frameworku ARKit, který

[1] AR z anglického výrazu *augumented reality*, česky rozšířená realita.

[2] „*AR allows the user to see the real world, but superimposes computer-generated information upon or composed with the real world. […] Combining 3D graphics with the real world in a 3D space is useful in that it enhances a user's perception of and interaction with the real world.*“ [1]

umožňuje nejen snímání pohybu zařízení, ale i detekci dalších objektů v místnosti (např. podlahy, zdí, obrázků a lidí), které pak můžou sloužit, jako kotvy pro umístění virtuálního obsahu do scény [4].

Telefony *iPhone* v *Pro* konfiguraci (od modelu *iPhone 12 Pro* a novější) mají ve výbavě navíc (oproti ostatním *iPhone* modelům) senzor *LiDAR*, který pomocí měření doby odrazu laserového paprsku vytváří přesný model okolí uživatele. Důsledkem integrace senzoru *LiDAR* je rychlejší detekce vertikálních a horizontálních ploch, přesnější snímání pohybu a umožnění překrytí virtuálních objektů reálnými, které jsou blíže ke kameře [5]. *LiDAR* senzor nejen vylepšuje framework ARKit, ale nabízí i přístup k naskenovaným 3D modelům scény – pomocí frameworku RoomPlan [6] lze získat 3D plán bytu včetně naměřených rozměrů a jednotlivých kusů nábytku.

2.1.2 Proč vizualizovat rozvody v AR?

Vizualizace rozvodů – cíl této práce – je praktický případ použití AR v praxi. Mobilní aplikace umožní uživatelům skrze AR zobrazit rozvody, které jsou jinak schované uvnitř zdí. Člověk, který si bude chtít pověsit poličku na zeď, již nebude muset obcházet místnost s metrem v ruce s cílem lokalizovat rozvody dle dříve provedených záznamů. Namísto toho otevře aplikaci v telefonu, ve které se dříve zaznamenané rozvody zobrazí pouhým namířením kamery na zeď.

2.1.3 Vizuálně-inerciální odometrie

Dle článku *Visual-Inertial Odometry of Aerial Robots* [7] je vizuálně-inerciální odometrie (dále jen VIO) metoda pro odhad stavu agenta (polohy a rychlosti) použitím kombinace dat z kamer a inerciálních měřicích jednotek (dále jen IMU). Kamery poskytují přesné informace během pomalého pohybu, ale mají omezenou rychlost výstupu a nejsou robustní v situacích s nízkou texturou, vysokou rychlostí a vysokým dynamickým rozsahem. Na druhou stranu, IMU jsou nezávislé na scéně, mají vysokou rychlost výstupu a jsou ideální pro doplnění kamer v náročných podmínkách. Avšak trpí špatným poměrem signálu a šumu a rychle akumulují odchylku kvůli zkreslení senzorů, proto nejsou vhodné pro sledování pomalého pohybu agenta (zařízení).

VIO může pracovat s více kamerami a IMU, ale minimální požadavek je jedna kamera a jedna IMU. Přístupy VIO lze rozdělit na volně spojené a těsně spojené metody. Volně spojené metody zpracovávají vizuální a inerciální měření odděleně před jejich sloučením, zatímco těsně spojené metody přímo vypočítávají konečný výstup z nezpracovaných měření kamer a IMU. Dle [7] je těsně spojená metoda VIO přesnější.

2.2 Analýza konvenčních metod zakreslení rozvodů

V této sekci jsou představeny tradiční metody zakreslení plánu rozvodů v bytě. Primárně jsou tyto plány určené pro elektrikáře a instalatéry pro použití během pokládání rozvodů. Dříve byly tyto plány kresleny ručně na papíře, nyní jsou však plány nejčastěji tvořeny v CAD programech [8].

U některých staveb není plán rozvodů součástí dokumentace, umístění rozvodů je pak založené na kvalifikovaném úsudku řemeslníka s cílem propojit požadované místa (v mezích stanovených platnými normami[3]).

2.2.1 AutoCAD

AutoCAD je software od firmy *Autodesk* pro univerzální 2D a 3D projektování. Program lze využít pro celý proces tvorby plánů domu [9]. Jednotlivé části procesu návrhu lze navíc urychlit pomocí použití dodatečných sad nástrojů (*Architecture*, *Electrical*, *MEP*) [10].

Program využívá formát souborů *.dwg*, který je jeden z nejvíce používaných typů souborů pro sdílení CAD projektů napříč všemi možnými obory [11]. Díky své popularitě jej podporuje mnoho dalších CAD programů nejen od firmy *Autodesk*.

2.2.2 CADKON+ MEP

CADKON+ MEP je CAD program společnosti *Graitec s.r.o.*, který je dle svých stránek [12] plnohodnotný CAD nástroj vhodný pro všechny fáze projektové dokumentace. Součástí programu jsou i specializované nástroje pro návrh potrubí, elektroinstalace a další rozvody.

2.2.3 ProfiCAD

Dalším programem pro tvorbu návrhu elektrických rozvodů je *ProfiCAD*, který je dle popisku na své webové stránce [13] nejjednodušší elektro CAD. Výhodou oproti předchozím programům je bezplatná licence pro domácí použití. Nevýhodou programu je, že slouží primárně pro návrh elektro dokumentace, tudíž neobsahuje rozsáhlé nástroje pro tvorbu plánů bytu. Program však podporuje import výkresů ze softwaru *AutoCad*, který lze využít, pokud uživatel vyžaduje propracovanější podklady, na které následně navrhne elektrické rozvody.

2.2.4 PlumbingCAD

Program *PlumbingCAD* je software umožňující pouze návrh vodovodních potrubí. Půdorys budovy je proto nutné do programu nahrát – lze vyžít grafiku ve formátu PDF či JPEG nebo import existujícího návrhu z programu *AutoCAD*. Dle informací na webových stránkách výrobce [14] je *PlumbingCAD* program pro rychlý návrh vodoinstalace, umožňující zobrazit seznam potřebných materiálů a následný cenový odhad (za předpokladu využití *PEX* potrubí). Výhodou tohoto programu je jeho snadné ovládání a možnost importu již vytvořených plánů z více typů souborů. Nevýhodou je omezený výběr potrubí (pouze *PEX* trubky), funkce umožňující pouze návrh potrubí a roční předplatné 200 $. Pokud chce uživatel použít program jen párkrát, nabízí se možnost využít zdarma 30denní trial verzi bez omezení funkčnosti.

[3] Například umístění rozvodů elektřiny je určeno normou *ČSN 33 2130 ed. 3*, ve které jsou stanovené specifické instalační zóny pro vedení elektrických rozvodů.

2.2.5 PlumbersCAD

PlumbersCAD [15] je dalším CAD programem určeným primárně pro instalatéry. Oproti programu *PlumbingCAD* – software umožňuje uživatelovi nejen zakreslit plán potrubí na existující půdorys budovy, ale taktéž zakreslit samotný plán budovy, jelikož se jedná o plnohodnotný CAD nástroj neobsahující pouze funkce pro návrh potrubí (program *PlumbersCAD* je rozšířenou verzí univerzálního CAD programu *RealCAD*). Výhodou tohoto programu je možnost využít univerzální CAD funkce (nejen funkce pro návrh vodoinstalace) a snadno ovladatelné rozhraní (dle informací na stránkách výrobce [15]). Nevýhodou je optimalizace programu pro australský trh – uživatelé z jiných zemí musí změnit nastavení tak, aby návrhy byly v souladu s lokálními standardy. Druhou nevýhodou je vysoká cena programu s cenou okolo 100 $ na měsíc. (Pomocí kontaktního formuláře lze zažádat o trial verzi.)

2.2.6 Shrnutí konvenčních metod

Z diagramů rozvodů elektřiny a vody (v příloze A) – z výše popsaných CAD programů – lze pozorovat, že plány rozvodů jsou často zakreslovány pouze na diagram půdorysu patra budovy. Na diagramech není zapsána informace o výšce vedení rozvodů, jelikož se předpokládá, že řemeslník provádějící práci bude znát patřičné stavební normy, ve kterých jsou uvedené instalační zóny, do kterých se rozvody mají položit. To však vytváří další komplikaci pro obyvatele bytů – přestože mohou mít přístup k projektové dokumentaci svého bytu, z plánu rozvodů nezjistí přesné umístění, kudy rozvody ve zdech vedou, bez složitého studia ČSN norem nebo konzultace s odborníkem.

Pokud plán rozvodů není součástí projektové dokumentace a obyvatel bytu by si ho sám chtěl vytvořit (např. při rekonstrukci pro pozdější použití), použití CAD programů není ideální řešení. Jelikož se pro ovládání těchto programů očekávají profesní znalosti stavebních inženýrů, kteří jsou cílová skupina těchto nástrojů. Druhým důvodem je vysoká cena těchto programů, která je těžko odůvodnitelná pro uživatele, kteří program nevyužívají jako součást svého podnikání.

2.3 Analýza existujících aplikací pro vizualizaci rozvodů v AR

Přestože je technologie AR dostupná na telefonech s operačním systémem iOS již od roku 2017 – vydaní iOS 11 [16, 17], v oficiálním obchodě s aplikacemi pro iOS – *App Store* – mnoho nástrojů pro praktické využití AR v domácnosti není.

AR nástroje pro domácnost, dostupné v obchodě App Store, lze rozdělit do těchto tří kategorií:

- Vzhled interiéru

  Jedná se především o aplikace sloužící k testování velikosti nábytku před zakoupením (např. aplikace *IKEA* [18]) nebo aplikace pro vizualizaci dekorace zdí.

- Měření

  Tyto aplikace slouží pro měření délky a plochy reálných objektů (např. aplikace *Measure* [19]).

- Skenování místností

  Aplikace v této kategorii slouží ke tvorbě půdorysů nemovitostí pomocí skenování místností v AR (např. aplikace *RoomScan Classic* [20] a aplikace *magicplan* [21]).

V navazujících podkapitolách jsou podrobněji analyzovány aplikace *Measure*, *RoomScan Classic* a *magicplan*, jakožto zástupci iOS aplikací využívající funkce AR pro měření reálných dat místností. Ačkoli tyto aplikace neslouží pro vizualizaci rozvodů v bytě, jsou zde uvedeny pro účely analýzy implementace AR funkcí a analýzy GUI AR části aplikací. Následně jsou analyzovány aplikace *vGIS* [22] a *spectar* [23] – aplikace určené pro použití v profesionálním prostředí, sloužící pro vizualizaci inženýrských sítí a BIM[4] pomocí AR/MR[5].

2.3.1 Measure

Measure je aplikace vyvinutá společností *Apple Inc.* V současnosti je předinstalovaná na všech nově prodaných telefonech iPhone. Aplikace slouží k měření délky a plochy objektů skrze funkce AR. Dle svých webových stránek v obchodě *App Store* [19] má aplikace na telefonech se senzorem *LiDAR*[6] funkce navíc – a to detekci vertikálních a horizontálních hran objektů, měření výšky lidí a ukládání zaznamenaných měření.

Aplikace je uživatelsky přívětivá, GUI aplikace obsahuje pouze 4 tlačítka, u kterých je na první pohled jasná jejich funkce (přidat nový bod, vytvořit snímek scény, vrátit se o krok zpět a vymazat scénu). Pokud má aplikace problém provést měření, v horní části obrazovky se uživateli zobrazí pomocná hláška navádějící k vyřešení problému (např. málo světla nebo telefon je příliš blízko).

Měření probíhá tak, že se prvním stiskem tlačítka „+“ přidá do scény nový bod, který je umístěn na detekovanou rovinu v bodě indikátoru ve středu obrazovky. Následně se zobrazí náhled úsečky s informací o její délce. Druhým stiskem tlačítka „+“ se potvrdí druhý bod úsečky. Na již

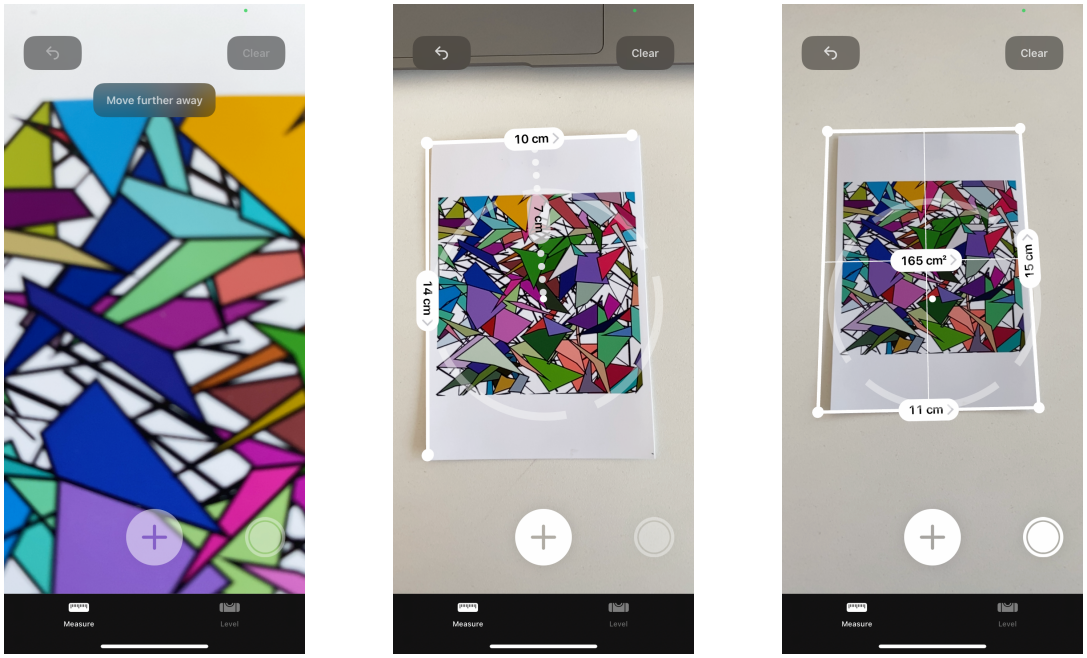
[4]BIM z anglického výrazu *Building Information Modeling*, česky informační model budovy, je proces pro vytváření a správu informací o stavebním projektu v průběhu celého jeho životního cyklu. Tyto procasy často obsahují kvalitní 3D modely jednotlivých součástí projektu. [24]

[5]MR z anglického výrazu *Mixed reality*, česky mixovaná realita, je kombinace technologií AR a VR. Zatímco ve virtuální realitě je uživatel obklopen pouze virtuálním světem, MR umožňuje uživateli interagovat plynule, jak s virtuálním světem, tak i s reálným (např. levá ruka ovládá rozhraní virtuálního světa, zatímco v pravé ruce uživatel drží nástroj v reálném světě). (Názory expertů se na přesnou definici MR liší, toto je jeden z používaných výkladů.) [25]

[6]V současnosti mají senzor *LiDAR* telefony *iPhone* pouze ve verzi *iPhone Pro* od 12. generace a novější.

naměřené vzdálenosti lze navázat dalším měřením, které je možné započít a ukončit v již zaznamenaných bodech, nebo ve středech úseček. Již zaznamenané body na obrazovce lze posunout pomocí tahu prstem po obrazovce.

Pokud by uživatel chtěl využít tuto aplikaci k zaznamenání rozvodů elektřiny a vody, mohl by jednotlivé rozvody vizualizovat pomocí virtuálních úseček a následně vytvořit fotografii měřené scény. Nevýhodou tohoto postupu by byla obtížnost zakreslení složitějších rozvodů (např. situace, kdy se rozvody větví a v jedné zdi jsou rozvody více typů) a jejich následná interpretace o několik let později, kdy by je uživatel chtěl využít.



Obrázek 2.1 Ukázka aplikace *Measure*

2.3.2 RoomScan Classic

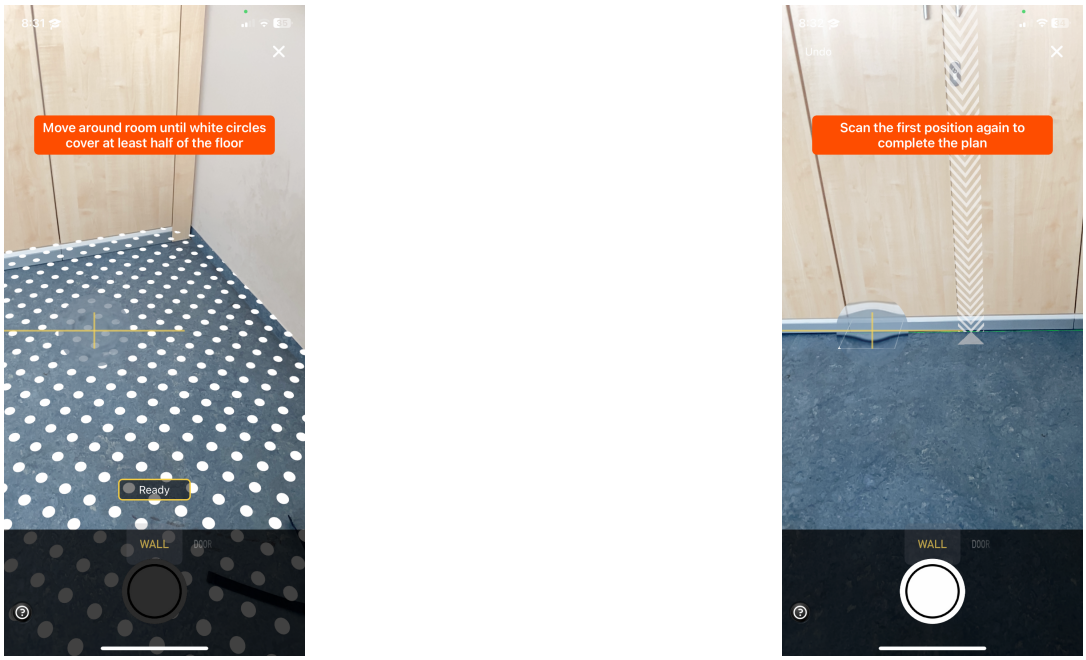
Další aplikací je *RoomScan Classic* [20] od společnosti *Locometric Ltd.* Aplikace slouží k tvorbě půdorysů nemovitostí a jejich následnému exportu do velkého množství formátů. Proces skenování probíhá postupně po jednotlivých místnostech, jejichž půdorysy lze následně spojit do jednoho plánu. Do půdorysů lze zaznačit dveře a okna, případně i přidat ručně psané poznámky a fotky místnosti. Aplikace nenabízí možnost pro zakreslení vedení rozvodů.

Tvorba plánu místnosti lze provézt třemi různými způsoby:

- První možnost, kterou aplikace nabízí, je skenování tvaru místnosti pomocí postupného přikládání telefonu na všechny zdi v místnosti. Dle popisku na stránce aplikace[20] je tato možnost vhodná, když je místnost špatně osvětlená nebo není vidět na podlahu. Tento proces nepoužívá data z kamery, jelikož je postavený pouze na měřeních senzorů akcelerometru a gyroskopu – tyto data jsou následně zpracována pomocí patentovaného algoritmu[26], který minimalizuje chyby měření a vygeneruje plán místnosti.

- Druhým způsobem je AR sken – uživatel se nejdříve musí projít po místnosti, čímž zkalibruje detekovanou rovinu podlahy. Následně pomocí vizuálního indikátoru na displeji telefonu označí jednotlivé spodní hrany zdí. Po označení všech zdí telefon vygeneruje půdorys místnosti.
- Posledním způsobem je tvorba půdorysu pomocí editoru, ve kterém uživatel tvoří plán pomocí manuálně naměřených údajů.

Pro telefony se senzorem *LiDAR* výrobce nabízí vylepšenou verzi aplikace [27], která zjednodušuje měření pomocí využití dat z hloubkové kamery *LiDAR*.



Obrázek 2.2 Ukázka aplikace *RoomScan Classic*

2.3.3 magicplan

Aplikace *magicplan* [21] od společnosti *Sensopia Inc.* slouží, podobně jako již zmíněná aplikace *RoomScan Classic*, k tvorbě půdorysů nemovitostí. Kromě tvorby a exportu půdorysů aplikace umožňuje vytvářet cenové odhady plánovaných prací (např. nátěru zdí či výměny podlahy). Do půdorysů lze zaznačit dveře, okna a nábytek, případně je možné i k jednotlivým zdem přidat poznámky a fotografie. Aplikace nenabízí možnost pro zakreslení vedení rozvodů.

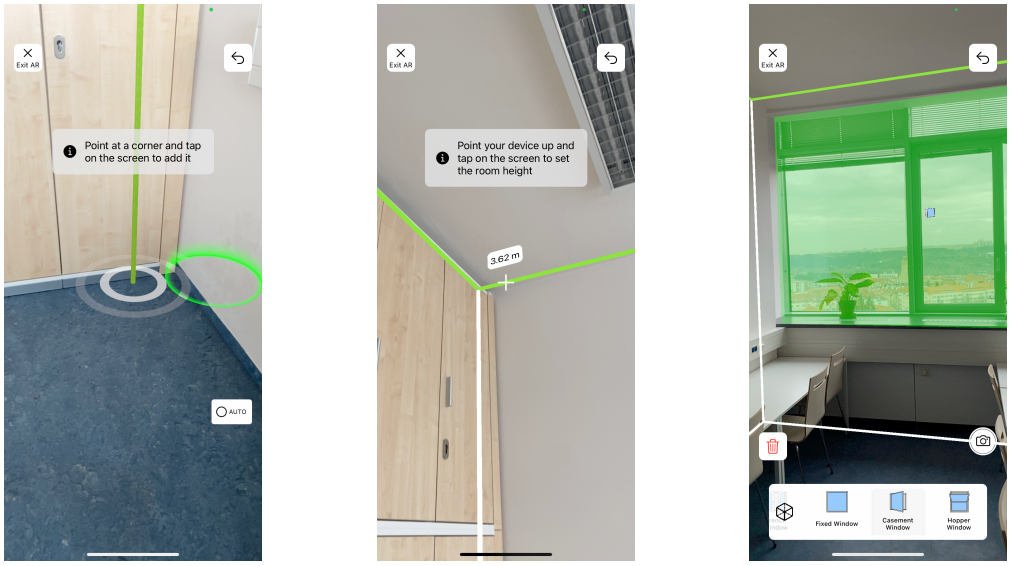
Pro tvorbu půdorysu místností jsou v aplikaci k dispozici následující dva způsoby zakreslení:

- První možností je využití grafického editoru, ve kterém uživatel aplikace postupně nakliká jednotlivé rohy místnosti a následně zadá délky jednotlivých zdí.

- Druhou možností je sken s použitím AR[7]. Při použití této metody uživatel aplikace nejdříve provede kalibraci AR prostředí namířením kamery na podlahu a její okolí. Ve chvíli, když je detekována rovina podlahy, začne proces označování rohů místnosti. Ve výchozím nastavení aplikace detekuje rohy místnosti automaticky (tuto funkci může uživatel vypnout přepínačem na obrazovce), pokud je však roh automaticky nedetekovatelný (např. schovaný za nábytkem) uživatel může označit roh místnosti kliknutím na obrazovku.

  Poté, co uživatel obejde celou místnost a naskenuje podruhé prvně naskenovaný roh, zbývá jen nastavit výšku stropu a označit polohu dveří a oken. Výška stropu se v aplikaci nastaví namířením telefonu na hranu stropu a kliknutím na obrazovku. Dveře a okna aplikace detekuje automaticky, na uživateli je pouze nastavení jejich typu pomocí posuvníku, co se objeví ve spodní části obrazovky.

  Závěrem lze říci, že skenování místnosti v aplikaci *magicplan* skrze AR prostředí je velmi jednoduché. Jednotlivé kroky skenování jsou od sebe jasně oddělené pomocí informačních textů zobrazených v horní části obrazovky. Většina akcí od uživatele probíhá pouhým kliknutím na obrazovku, GUI AR části aplikace tak obsahuje minimální počet tlačítek, uživatel se proto neztrácí ve velkém množství funkcí.



Obrázek 2.3 Ukázka aplikace *magicplan*

[7]Testováno na telefonu *iPhone 11*, který nemá senzor *LiDAR*. Dle webových stránek aplikace [28] u telefonů *iPhone* se senzorem *LiDAR* není nutné při skenování označovat rohy místnosti – tento krok lze nahradit automatickou detekcí zdí.

2.3.4 vGIS

vGIS je aplikace od společnosti *vGIS Inc.* určená pro monitorování průběhu stavebních prací v AR skrze obrazovky telefonu nebo pomocí *headsetu Microsoft HoloLens*. Dle informací na stránkách produktu [29] aplikace kombinuje reálný svět s propracovanými BIM inženýrskými návrhy. Uživatel může aplikaci využít, jak pro vizualizaci plánované stavby, tak pro vizualizaci skryté infrastruktury (např. inženýrských sítí pod zemí), aby se předešlo možným problémům při probíhající stavbě.

Dle ukázek na stránce *vGIS* lze aplikaci využít pro vizualizaci rozvodů uvnitř budov. Aplikace však není vhodná pro zákazníky z nekomerčního prostředí, jelikož její využívání stojí 1 250 $ na rok. Aplikaci se nepodařilo otestovat z důvodu již zmíněného platebního modelu. Lze však předpokládat, že aplikace je určená primárně pro vizualizaci dat stavebních plánů BIM a pro nekomerční zákazníky bude její použití pro zaznamenání rozvodů uvnitř bytu a jejich následnou vizualizaci nepoužitelné.

Obrázek 2.4 Ukázka aplikace *vGIS* [30]

2.3.5 spectar

Aplikace *spectar* od stejnojmenné společnosti je MR nástroj pro *headsety Microsoft HoloLens* pro použití během stavebních prací. Dle informací na webových stránkách [23] *spectar* slouží pro zobrazení BIM modelů v průběhu procesu stavby. Holografické promítání plánů skrze *headsety* umožňuje pracovníkům na stavbě zefektivnit práci, identifikovat problémy dříve než nastanou a vylepšuje týmovou koordinaci.

Obrázek 2.5 Ukázka aplikace *spectar* [23]

2.3.6 Shrnutí analýzy dostupných AR aplikací

Z analýzy praktických AR nástrojů pro domácnost lze pozorovat trend, který měly všechny 3 popisované aplikace[8] společný – a to jednoduché GUI v části aplikace, která využívá rozšířenou realitu.

Jednoduché a intuitivní GUI je z hlediska návrhu aplikace důležitý prvek, který je o to důležitější, když je na obrazovce společně s dalším obsahem – v případě AR to je stream videa ze zadní kamery a dodatečné objekty přidané do scény. Pokud by na obrazovce bylo příliš mnoho GUI prvků (např. tlačítka, přepínače ...), uživatel by se snadno mohl ztratit v dostupných funkcích a AR prostředí by již nebylo hlavní funkcí obrazovky, jelikož by bylo z velké části překryté statickým obsahem GUI.

vGIS a *spectar* jsou aplikace schopné vizualizovat rozvody skrze AR. Nejsou však vhodné pro uživatele mimo obor stavebnictví, kteří by je chtěli využít pouze za tímto účelem. Jedním z důvodů je vysoká provozní cena těchto aplikací (případně i nutnost zakoupit MR *headset* pro využívání aplikace *spectar*). Druhým důvodem je potřebná znalost ovládání CAD programů, které se využívají pro tvorbu BIM plánů, jež tyto aplikace vizualizují.

[8] aplikace *Measure*, *RoomScan Classic* a *magicplan*

2.4 Požadavky

Před vytvořením návrhu vznikající aplikace je vhodné stanovit funkční a nefunkční požadavky, aby při návrhu a implementaci bylo jednoznačně definované, jaké funkce má vznikající program mít a jak se má chovat. Dle [31] funkční požadavky určují, jaké funkce má aplikace implementovat. Nefunkční požadavky – taktéž obecné požadavky – specifikují omezení kladená na systém. Stanovené požadavky by měly být jednoznačné, splnitelné a ověřitelné.

2.4.1 Funkční požadavky

F1 Využití AR pro vizualizaci rozvodů

Aplikace bude využívat funkce rozšířené reality pro vizualizaci umístění rozvodů elektřiny a vody ve zdech bytu.

F2 Tvorba diagramu rozvodů

Aplikace bude obsahovat nástroj pro tvorbu diagramů rozvodů (v prostředí AR). Vytvořené diagramy budou následně uložené do úložiště aplikace pro pozdější použití.

F3 Zobrazení a úprava již vytvořených diagramů rozvodů

Uložené diagramy rozvodů bude možné znovu zobrazit, a to tak že budou v AR prostředí stále na stejném místě, aby je bylo možné použít pro přesnou lokalizaci rozvodů. Již vytvořené diagramy rozvodů bude možné nadále editovat, aby bylo možné zaznamenat případné změny.

F4 Struktura dat v aplikaci

Aby byly diagramy rozvodů v aplikaci jednoduše dohledatelné, uživatel před vytvořením každého diagramu vyplní o jakou místnost v bytě se jedná a jakou zeď diagram zachycuje. Ke každé zdi bude možné přidat upřesňující poznámku a fotografie z galerie, či kamery telefonu.

2.4.2 Nefunkční požadavky

N1 Podporovaná zařízení

- Aplikace bude funkční na všech zařízeních *iPhone*, které stále mají softwarovou podporu výrobce – tedy na telefonech s operačním systémem *iOS 16*[9] a novějším.
- Aplikace nebude vyžadovat použití senzoru LiDAR, aby se cílová množina uživatelů neomezila pouze na vlastníky telefonů v nejdražší konfiguraci.
- Obsah aplikace bude na všech podporovaných zařízeních přizpůsoben velikosti obrazovky tak, aby aplikace byla plně funkční. (Například na malé obrazovce se nestane, že bude GUI prvek překryt jiným, nebo bude mimo obrazovku.)

N2 Stabilita

- Aplikace bude stabilní a nebude se náhodně vypínat kvůli chybám v kódu, či únikům dat v paměti.

N3 Jednoduché rozhraní

- Rozhraní AR části aplikace bude mít minimální počet GUI prvků, aby uživatel nebyl přehlcen složitým ovládáním v kombinaci se zobrazením *feedu* z kamery s AR prvky. (V návaznosti na pozorování vyvozené z analýzy konkurenčních aplikací 2.3.6.)
- Rozhraní aplikace by mělo být co nejvíce intuitivní, aby uživatel mohl používat aplikaci instinktivně bez nutnosti čtení návodu.
- Aplikace bude využívat ovládací a navigační prvky běžně využívané v iOS aplikacích tak, aby ovládání aplikace bylo přirozené a uživatele nemátlo.

N4 Jazyková lokalizace

- Aplikace v závislosti na nastavení systému telefonu zvolí jazyk prostředí aplikace. Uživatelům s českým nastavením bude prostředí zobrazeno v češtině, pro ostatní uživatele bude rozhraní přeložené do angličtiny.

N5 Noční režim

- Aplikace bude podporovat zobrazení obsahu v nočním režimu. Noční režim bude aktivován automaticky dle systémového nastavení telefonu.

[9] Seznam zařízení podporující *iOS 16*:
`https://support.apple.com/guide/iphone/supported-models-iphe3fa5df43/16.0/ios/16.0`

Kapitola 3

Technologie

V této kapitole se rozebírají technologie, které byly vybrány pro návrh a implementaci aplikace. Kapitola obsahuje analýzu a zdůvodnění výběru technologií, jako jsou Figma, iOS, Swift, SwiftUI, ARKit a RealityKit.

3.1 Figma

Pro návrh uživatelského rozhraní byl zvolen online nástroj *Figma*[1]. Tento nástroj byl zvolen z následujících důvodů:

- Bezplatný plán postačující pro potřeby tohoto projektu.
- Široká nabídka komunitou vytvořených šablon (např. UI prvky systému *iOS*, *mockupy* telefonů, …) dostupných pod otevřenou licencí *CC BY 4.0*[2].
- Systém podporuje tvorbu komponent z prvků, které se v návrhu opakují. Změny v komponentě se projeví ve všech jejích instancích v návrhu – odpadá tím nutnost provádět úpravy ve všech částech návrhu.
- Integrovaný systém pro správu verzí návrhu.
- Aplikace podporuje sdílení přístupu do projektu dalším osobám. Toho lze využít například pro revizi návrhu.
- Snadný export návrhu jednotlivých obrazovek do velkého množství grafických formátů.

[1] `https://www.figma.com/`
[2] `https://creativecommons.org/licenses/by/4.0/deed.cs`

3.2 iOS

iOS je operační systém od společnosti *Apple* poprvé představený v roce 2007[3] společně s první generací telefonu iPhone. Nejnovější verze tohoto sytému nese označení *iOS 16.4*. [32]

Při vývoji aplikací pro *iOS* je nutné specifikovat minimální verzi systému, na kterém aplikace poběží. Při specifikování minimální verze je nutné zvážit výhody a nevýhody, které s sebou jednotlivé verze nesou – čím novější verzi vývojář zvolí, tím získá přístup k více nástrojům a funkcím, ale zároveň omezí cílovou množinu telefonů, na kterých půjde aplikaci nainstalovat. Zde se však ukazuje velká výhoda systému *iOS* oproti telefonům s operačním systémem *Android* – nejnovější *major* verzi *iOS 16* mělo 7 měsíců po jejím oficiálním spuštění 74,1 % telefonů *iPhone*, zatímco nejnovější verzi OS *Android* mělo 7 měsíců po jejím oficiálním spuštění pouze 14,5 % *Android* telefonů. Pokud se vezme v potaz i předchozí *major* verze OS, tak 19 měsíců od spuštění *iOS 15* mělo tuto verzi nebo novější 91,4 % zařízení *iPhone*, kdežto u OS *Android* to bylo pouze 38,3 %. [33, 34]

Pro vývoj aplikace byla zvolena verze *iOS 16.0* jako minimální verze cílového OS – tedy poslední *major* verze systému – protože obsahuje podporu mnoha nových funkcí a ke dni 11. 4. 2023 byla nainstalována již na 74 % zařízení s výhledem, že během následujících 12 měsíců bude alespoň tato verze na přibližně 90 % aktivních zařízeních *iPhone*.

3.3 Swift

Swift je programovací jazyk od společnosti *Apple* vydaný v roce 2014, jako moderní náhrada za dosud využívaný jazyk *Objective-C* pro vývoj nativních aplikací pro operační systémy *macOS* a *iOS*[4]. Zde je přehled vlastností, které tento jazyk nabízí (informace čerpány z [35]):

- Typová inference – není nutné deklarovat typ proměnných, pokud jej lze jednoznačně odvodit z kontextu.
- Automatická správa paměti pomocí deterministického počítání referencí.
- Funkce vyššího řádu a dílčí funkce[5] umožňující používat vzory funkcionálního programování.
- Proměnné jsou vždy inicializovány. Pokud chce programátor používat *nil* hodnoty, je nutné označit proměnné jako *optional*.
- Kompatibilita s *Objective-C* kódem.

Jelikož požadavky na aplikaci vyžadují verzi pouze pro OS *iOS*, byl pro vývoj aplikace zvolen jazyk *Swift*, jakožto primární jazyk pro vývoj nativních *iOS* aplikací. Použití ne-nativních řešení[6], které umožňují vydání aplikace i na OS *Android* nebylo potřeba. Oproti ne-nativním technologiím je *Swift* optimalizovaný na míru pro OS *iOS* a *hardware* telofonů *iPhone* a díky tomu nabízí vyšší výkon, robustnost a spolehlivost.

[3] Označení *iOS* systém získal až o několik let později – v roce 2010, kdy byla uvedena verze *iOS 4.0*.
[4] Nyní jazyk slouží i pro vývoj aplikací pro *iPadOS*, *tvOS* a *watchOS*.
[5] *High-order functions*, *partial functions*.
[6] Například *Flutter*, *React Native* a *Unity*.

```swift
import SwiftUI

struct ContentView: View {
    var body: some View {
        Text("Hello, world!")
    }
}
```

Výpis kódu 1 *Hello world!* ve frameworku *SwiftUI*

3.4 UI framework

Uživatelské rozhraní *iOS* aplikací lze v jazyce *Swift* vyvíjet pomocí 2 frameworků. První možností je použít starší framework *UIKit* využívající imperativní způsob programování, druhou možností je použít relativně nový framework *SwiftUI*, který je naopak deklarativní.

3.4.1 SwiftUI

SwiftUI je moderní framework pro tvorbu UI v jazyce Swift, který byl poprvé představen společností Apple v roce 2019 [36]. Jedná se o deklarativní framework, což znamená, že programátor popisuje to, co by mělo být na obrazovce zobrazeno, namísto toho, aby popisoval, jaké kroky se mají postupně provést pro vykreslení obsahu, jak je tomu zvykem u imperativních programovacích nástrojů. Tento zjednodušený způsob tvorby UI usnadňuje vývojářům práci a umožňuje jim vytvářet a upravovat UI v reálném čase pomocí živého náhledu obrazovky uvnitř vývojového prostředí *Xcode*. Mezi další výhody *SwiftUI* patří automatické aktualizace obrazovky (*View*) dle změn v napojeném *view-modelu*, vestavěné animace a snadná přenositelnost kódu mezi různými platformami[7].

Nicméně, nevýhodou je omezená funkcionalita v porovnání s frameworkem *UIKit*, což může být omezující pro pokročilé aplikace. Protože *SwiftUI* je relativně nový framework, může být obtížné najít dostupné informace a návody pro jeho použití, zvláště v porovnání s etablovaným frameworkem *UIKit*, který má mnoho dostupných zdrojů a dokumentace.

V ukázce kódu 1 je implementován *Hello world!*[8] ve frameworku *SwiftUI*.

3.4.2 UIKit

Framework *UIKit* byl vydán v roce 2008 [36] jako součást sady nástrojů pro vývoj aplikací (dále jen SDK[9]) pro OS *iOS*. Ačkoliv byl v roce 2019 vydán nový UI framework – *SwiftUI* – *UIKit* je stále aktualizován o nové funkce a je vývojáři nadále využívaný.

[7] *SwiftUI* lze použít k vývoji UI aplikací pro OS *macOS*, *iOS*, *iPadOS*, *tvOS* a *watchOS*.

[8] *Hello world!* ukázky jsou velmi jednoduché programy, které pouze vypisují na obrazovku textový řetězec "Hello world!". Často se používají jako první krok při učení se nového programovacího jazyka. Lze je také využít pro ukázku rozdílů mezi různými jazyky a knihovnami.

[9] SDK z anglického výrazu *software development kit*, česky sada pro vývoj softwaru.

```swift
import UIKit

class ViewController: UIViewController {

    override func loadView() {

        super.loadView()

        view.backgroundColor = .white

        let frame = CGRect(x: view.center.x - 100, y: view.center.y, width: 200, height: 50)
        let label = UILabel(frame: frame)
        label.text = "Hello world!"
        label.textAlignment = .center

        view.addSubview(label)
    }
}
```

Výpis kódu 2 *Hello world!* ve frameworku *UIKit*

Jedná se o imperativní framework, což znamená, že programátor popisuje kroky, jak by měla být akce vykonána. Tento přístup je vhodný pro aplikace, které vyžadují používání nestandardních UI prvků. Mezi výhody frameworku patří robustní funkčnost, vysoký výkon a velká komunita vývojářů. Nicméně, nevýhodou je složitost a velké množství kódu potřebné pro tvorbu jednotlivých komponent UI, což může být náročné pro začínající vývojáře. Dále, kvůli omezené *cross-platform* přenositelnosti, lze framework použít pouze pro tvorbu UI na OS *iOS*, *iPadOS* a *tvOS*.

V ukázce kódu 2 je implementován *Hello world!* ve frameworku *UIKit*. Při porovnání s *Hello world!* ve *SwiftUI* 1 lze poukázat na rozdíly imperativního přístupu *UIKit* od deklarativního přístupu *SwiftUI* – ve *SwiftUI* stačí zadefinovat, že *View ContentView* bude obsahovat *View Text* s definovaným obsahem, zatímco v *UIKit* je nutné nejprve vytvořit objekt *UILabel*, následně změnit jeho parametry a nakonec je nastaven jako *subview* nadřazeného *view*.

3.4.3 Zvolený framework

Pro tvorbu UI aplikace byl zvolen framework *SwiftUI*, protože jeho deklarativní přístup a automatické vykreslování změn obsahu umožňuje efektivní programování jednotlivých obrazovek. Jelikož je *SwiftUI* relativně nový framework a nejsou v něm implementovány všechny funkcionality, které jsou v *UIKit*, byl v projektu použit i framework *UIKit*, tam kde neexistovala možnost implementace za pomoci *SwiftUI* a v případech, kde bylo řešení v *UIKit* objektivně výhodnější.

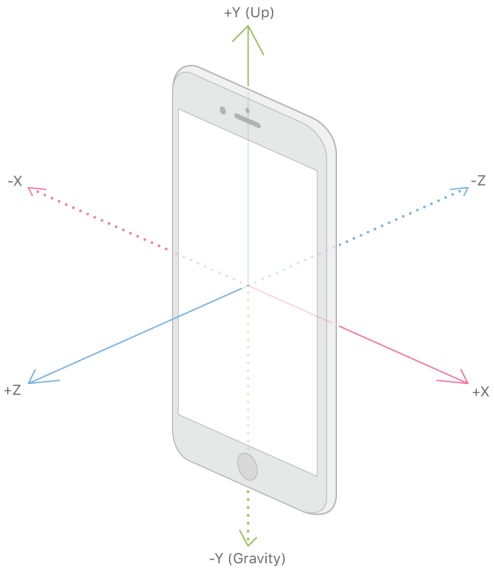
3.5 ARKit

ARKit je framework, který pomocí strojového zpracování obrazových dat z kamery a dat pohybových senzorů – akcelerometru a gyroskopu – měří pohyb zařízení relativně vůči svému okolí (viz sekce *vizuálně-inerciální odometrie* 2.1.3). Kromě měření pohybu zařízení framework umožňuje detekovat v okolním prostoru vertikální a horizontální plochy (např. podlahy, zdi a rovné plochy nábytku) a předem definované objekty (2D obrázky a 3D modely objektů). [4]

3.5.1 Systém souřadnic

Ve frameworku *ARKit* je ve výchozím nastavení implementována pravotočivá soustava souřadnic. Poloha zařízení je při spuštění AR označena jako počátek soustavy souřadnic, přičemž osa Y je nastavena kolmo nahoru (proti směru gravitace), osa X míří doprava a osa Z míří směrem k obrazovce zařízení (viz ilustrační diagram 3.1).

ARKit využívá metrický systém souřadnic – tedy objekt na souřadnicích *[0, 1, 0]* bude 1 metr nad počátkem systému souřadnic.



Obrázek 3.1 Výchozí systém souřadnic ve frameworku *ARKit* [37]

3.5.2 Transformační matice

Pro umístění objektů v AR scéně se využívá transformační matice, která popisuje, jak se má objekt transformovat, aby byl přesunut z počátku systému souřadnic na požadovanou pozici. Transformační matice slouží k matematické reprezentaci polohy, orientace a měřítka objektů v 3D prostoru vůči počátku souřadnicového systému. Jedná se o 4x4 matici, která obsahuje informace o translaci, rotaci a škálování objektu. Ve frameworku *ARKit* se pro tento účel používá datový typ `simd_float4x4`, který je poskytován knihovnou *simd*.

3.5.3 ARAnchor

Pro ukotvení virtuálního obsahu do reálného prostředí se ve frameworku *ARKit* používají kotvící objekty – *ARAnchor* – které reprezentují pozici v reálném světě. Buď je lze vytvořit manuálně poskytnutím transformační matice, nebo pomocí automatické detekce scény, kdy se kotvící objekty vytvoří poté, co se jsou ve scéně detekovány předem specifikované objekty.

Kotvy detekovaných objektů ve scéně lze získat skrze funkci
`func session(ARSession, didAdd: [ARAnchor]) {...}` protokolu `ARSessionDelegate`, která je zavolána pokaždé, když je ve scéně detekován nový sledovaný objekt (např. rovina zdi nebo referenční obrázek).

Kotvy detekovaného obsahu mohou v průběhu času měnit své parametry. V moment, kdy se změní parametry některých kotev ve scéně, je zavolána funkce
`func session(ARSession, didUpdate: [ARAnchor]) {...}`.

3.5.4 Detekce rovin

ARKit umožňuje automatickou detekci rovin v okolním prostředí[10]. V konfiguraci AR prostředí lze nastavit, které roviny budou během běhu aplikace detekovány – lze zvolit detekci pouze horizontálních rovin, jako jsou podlahy a stoly, nebo také vertikálních rovin, jako jsou stěny a dveře, či obě možnosti zároveň.

V moment, kdy je rovina detekována je zpřístupněn objekt typu `ARPlaneAnchor`, který v sobě nese informace o velikosti, tvaru, orientaci a klasifikaci roviny. Jelikož probíhající analýza scény neustále zpřesňuje virtuální model okolí zařízení, jsou informace o detekované rovině nadále aktualizovány o přesnější data (např. velikost roviny se může v čase měnit a s tím i informace o jejím středu).

Testování funkcí pro detekci rovin ukázalo, že *ARKit* spolehlivě detekuje horizontální roviny, ale má problémy s vertikálními rovinami, zejména pokud nemají dostatečně výraznou texturu pro snadnou analýzu obrazu (např. bílé stěny)[11]. Ve většině případů byla rovina zdi detekována, poté co bylo se zařízením postupně mířeno na jednotlivé detaily na zdi a zařízení bylo nošeno na různé pozice v místnosti, tak aby zeď viděl z více odlišných úhlů.

Na zařízeních se senzorem *LiDAR* by se popisovaný problém neměl vyskytovat, jelikož se pro detekci zdí využívají 3D hloubková data zmiňovaného senzoru, u kterého nezáleží na uniformitě textury povrchu.

3.5.5 Detekce obrázků

Framework lze využít pro detekci přednastavených referenčních obrázků v okolí zařízení. Při specifikaci těchto obrázků je nutné definovat jejich skutečnou šířku, jelikož framework počítá z údajů o velikosti detekovaných obrázků jejich vzdálenost od kamery zařízení. Důsledkem nepřesných dat velikosti referenčního obrázku by byla špatná pozice kotvy detekovaného obrázku – `ARImageAnchor` – uvnitř virtuální reprezentace reálného světa.

[10] Pouze v *ARWorldTrackingConfiguration* konfiguraci (3.5.7.1).
[11] Testováno na telefonu *iPhone 11*.

3.5.6 Ideální referenční obrázek

Během testování frameworku *ARKit* bylo testováno, jaký obrázek je ideální nastavit tak, aby byl aplikací detekován z co největší vzdálenosti, a jak závisí velikost obrázku na maximální vzdálenosti detekce. Pro testování byly zvoleny obrázky několika různých druhů tak, aby byly otestovány, jak obrázky, které by teoretický uživatel AR aplikace mohl mít doma, tak *ArUco* značky [38] využívané v aplikacích strojového vidění.

Pro testování byly použity následující obrázky:

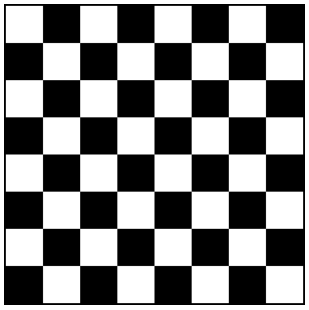
- A (3.2) – První obrázek je tradiční 8x8 šachovnice. Tento obrázek byl do testu zařazen, aby se zjistilo, jaký vliv na detekci má opakující se vzor a symetrie obrázku.
- B (3.3) – Druhým obrázkem je vygenerovaný AR *marker* z webové stránky `https://www.brosvision.com/ar-marker-generator/`. Obrázek je složen z velkého množství jednoduchých geometrických tvarů různých barev. Jednotlivé tvary mají mezi sebou výrazné kontrastní přechody.
- C (3.4) – Třetím obrázkem je *ArUco marker* o velikosti 7x7. *ArUco* značky se využívají v aplikacích strojového vidění (např. pro lokalizaci autonomních robotů). Značky jsou z dálky lehce detekovatelné a každá značka v sobě nese zakódovanou informaci vlastního ID. [38]
- D (3.5) – Čtvrtým obrázkem je QR kód, který se běžně využívá v aplikacích strojového vidění pro zakódování dat do 2D obrazové matice [39]. Oproti 7x7 *ArUco* značce má zvolený QR kód[12] větší datovou kapacitu, která však vyžaduje větší datové pole pro zakódování obsahu. Důsledkem jsou menší *datové pixely* u kterých lze předpokládat, že budou detekovatelné z menší vzdálenosti, jak větší *pixely ArUco* značky.
- E (3.6) – Pátý obrázek představuje fotografii, kterou by teoretický uživatel aplikace mohl mít doma vytištěnou. Jedná se o fotografii siluet lidí stojících na ostrově při západu slunce. Pro testování byla upravena na čtvercový formát, aby test nebyl ovlivněn jiným poměrem stran jednotlivých obrázků.
- F (3.7) – Poslední obrázek je stejně jako pátý obrázek fotografie s čtvercovým poměrem stran. Na fotografii je zachycen západ slunce prosvítající skrze stromy.

3.5.6.1 Test 1 – ideální referenční obrázek

První test testuje, zda má obsah referenčního obrázku vliv na maximální vzdálenost, ze které je referenční obrázek aplikací detekován. Během testu byly u každého obrázku měřeny 2 hodnoty:

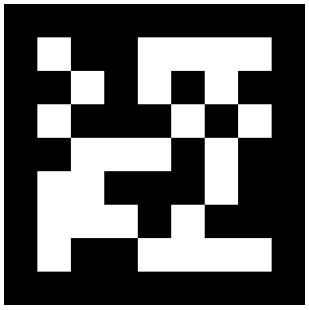
- První hodnota je vzdálenost, kdy zařízení poprvé referenční obrázek detekovalo.
- Poté, co je referenční obrázek detekován, lze zařízení oddálit, aniž by sledování bylo přerušeno. Druhou hodnotou je maximální vzdálenost zařízení, ve které je obrázek stále detekován.

[12] Zvolený QR kód má velikost 25x25 *datových pixelů*, existují však kódy v rozsahu 21x21 až 177x177 *datových pixelů* (QR kód verze 40). [39]



Obrázek 3.2 A

Obrázek 3.3 B



Obrázek 3.4 C

Obrázek 3.5 D

Obrázek 3.6 E

Obrázek 3.7 F

Pro tento test byly všechny obrázky vytištěny ve stejné velikosti – 197x197 mm. Následně proběhlo měření, kdy byly postupně pro jednotlivé obrázky měřeny výše zmíněné vzdálenosti. Pro každý obrázek byly hodnoty měřeny třikrát a následně bylo měření opakováno, aby se omezil vliv změny světla v průběhu dne[13]. Celkem bylo pro každý obrázek provedeno 6 měření.

Během testu bylo zařízení nejprve posouváno směrem k referenčnímu obrázku – v moment, kdy byl obrázek detekován, byla zaznamenána první vzdálenost. Následně bylo zařízení posouváno zpět, dokud nepřestal být obrázek detekován – pak byla zaznamenána druhá vzdálenost.

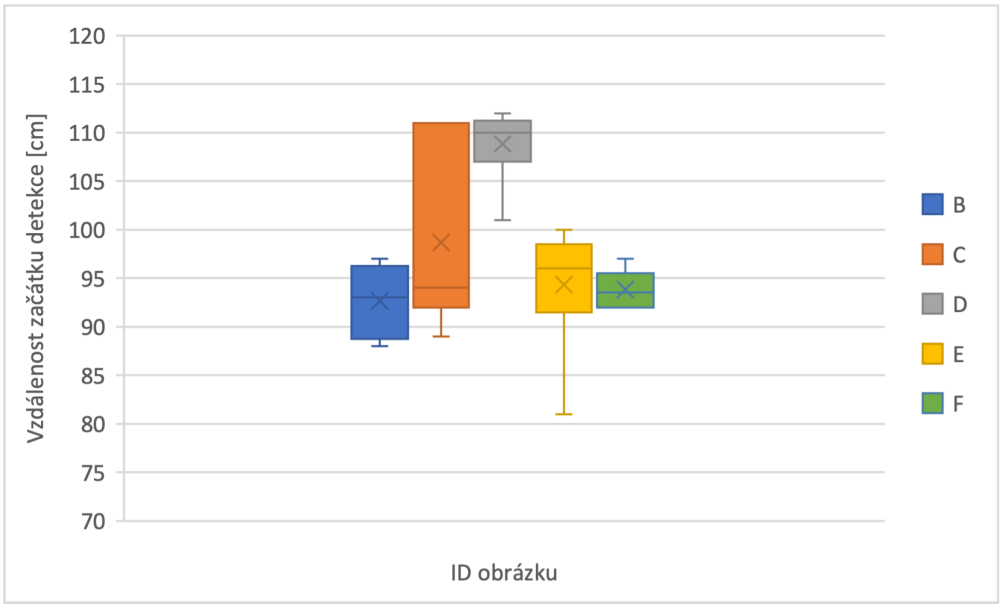
Obrázek A (3.2) – šachovnice – se nepodařilo ani jednou detekovat. Nejspíše to je důsledkem opakujícího se vzoru a několika os symetrie.

V testu vzdálenosti detekce obrázku zvítězil vzorek D (3.5), který byl detekován průměrně již v 109 cm. Na druhém místě skončil obrázek C (3.4) s průměrnou vzdáleností detekce 99 cm. V testu vzdálenosti konce detekce obrázku vedl vzorek C (3.4), který přestal být detekován v průměrné vzdálenosti 454 cm. Příčinou takto dobrého výsledku (oproti ostatním obrázkům) jsou nejspíše velké jednoduché tvary, které jsou snadno viditelné i z větší vzdálenosti. Naměřená data jsou znázorněná v grafech 3.8 a 3.9[14]. Tabulka se všemi naměřenými hodnotami je v příloze B.1.

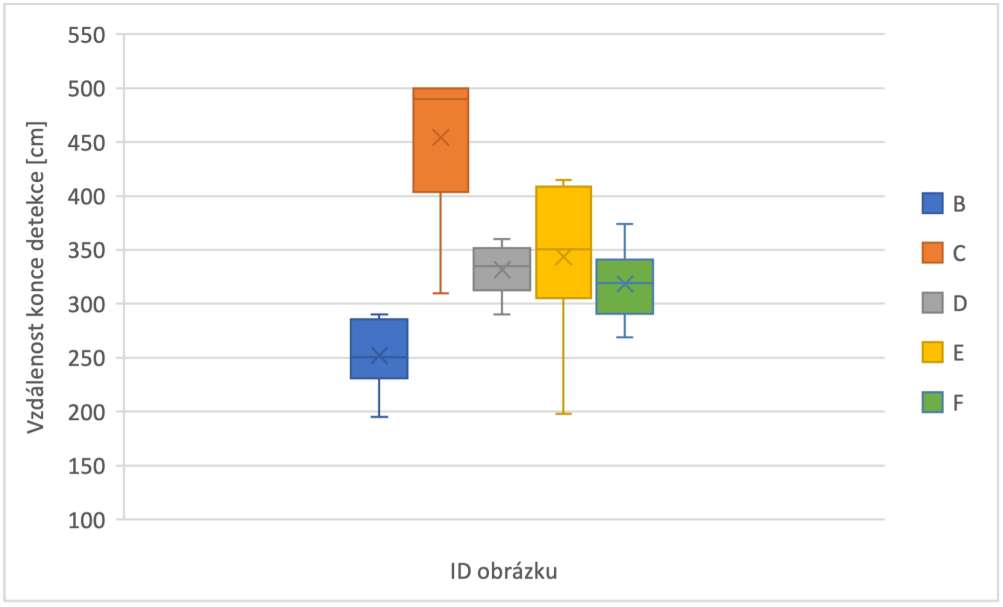
Ideálním referenčním obrázkem byl zvolen vzorek C (3.4) – *ArUco marker* – jelikož byl detekován z poměrně velké vzdálenosti a po detekci byl spolehlivě sledován do velmi velké vzdálenosti. Kromě vzorku A (3.2), který se nepodařilo detekovat, byly všechny ostatní vzorky detekovány

[13] Testované vzorky byly osvětlené i umělým světlem, aby byl vliv přirozeného světla minimalizován.
[14] Obrázek A není v grafech znázorněn, jelikož nebyl ani jednou detekován.

z relativně velké vzdálenosti[15]. Jako referenční obrázek tedy lze použít téměř jakýkoliv obrázek, pouze je nutné, aby v sobě neměl osy symetrie a opakující se vzory. Je vhodné, když obrázek obsahuje výrazné tvary, které jsou lehce viditelné i z větší vzdálenosti.



Obrázek 3.8 Graf maximální vzdálenosti začátku detekce referenčního obrázku



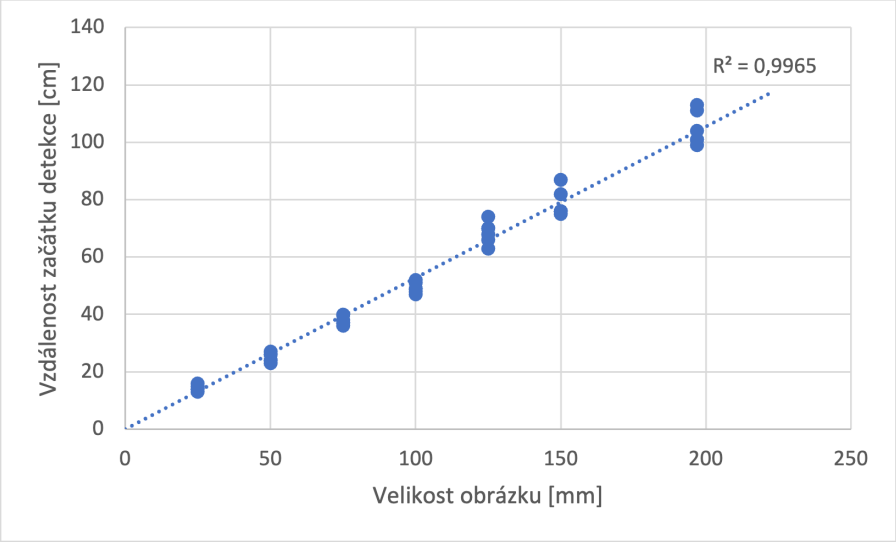
Obrázek 3.9 Graf vzdálenosti konce detekce referenčního obrázku

[15]Při měření počátku detekce byly všechny úspěšné vzorky v 15% rozmezí od nejlepšího vzorku. Při měření konce detekce byly všechny úspěšné vzorky v 45% rozmezí od nejlepšího vzorku.

3.5.6.2 Test 2 – vliv velikosti obrázku na maximální vzdálenost detekce

Cílem druhého testu bylo zjistit, jaká je závislost mezi velikostí referenčního obrázku a vzdáleností jeho detekce. Pro tento test byl zvolen obrázek C (3.4), který byl vytisknut v následujících sedmi velikostech: 197 mm, 150 mm, 125 mm, 100 mm, 75 mm, 50 mm a 25 mm. Test 2 měl stejný průběh jako test 1 – pro každou velikost obrázku bylo provedeno 6 měření 2 vzdáleností – vzdálenosti zařízení od obrázku při prvotní detekci a vzdálenosti zařízení od obrázku při přerušení detekce.

Při analýze naměřených dat bylo pozorováno, že vzdálenost začátku a konce detekce referenčního obrázku je přímo úměrná délce hrany obrázku (viz grafy 3.10 a 3.11). U grafu konce detekce (3.11) je hodnota R^2 nižší jak u grafu počátku detekce[16], jelikož detekce referenčního obrázku byla měřena pouze do vzdálenosti 500 cm kvůli prostorovým omezením testovací místnosti. Referenční obrázky s hranami o délce 197 a 150 mm by nejspíše přestaly být detekovány ve větší vzdálenosti. Tabulka se všemi naměřenými hodnotami je v příloze B.2.

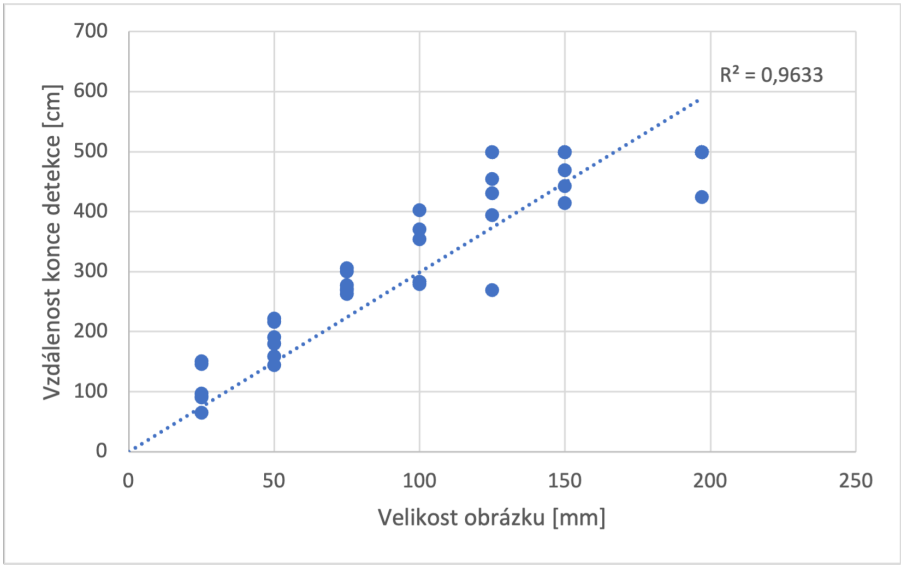


Obrázek 3.10 Graf závislosti začátku detekce referenčního obrázku na velikosti obrázku

3.5.7 AR konfigurace

ARKit lze použít pro mnoho odlišných projektů. Každý projekt však vyžaduje mírně odlišný způsob analýzy scény. Tento problém je ve frameworku *ARKit* řešen pomocí několika rozdílných konfigurací. Před začátkem implementace je nutné zvolit vhodnou konfiguraci frameworku, tak aby co nejvíce vyhovovala účelům projektu. V následujících podkapitolách jsou uvedeny jednotlivé konfigurace, které lze použít (informace o jednotlivých konfiguracích jsou čerpány z odpovídajících kapitol oficiální dokumentace frameworku *ARKit* [4]).

[16]Hodnota R^2 – též koeficient determinace – udává přesnost regresního modelu vůči naměřeným datům. Čím je hodnota blíže k 1, tím je model přesnější. [40]



Obrázek 3.11 Graf závislosti konce detekce referenčního obrázku na velikosti obrázku

3.5.7.1 ARWorldTrackingConfiguration

ARWorldTrackingConfiguration sleduje pohyb zařízení[17] vůči okolnímu prostoru v šesti osách volnosti – tedy posun zařízení po osách *[x, y, z]* a rotace okolo os *[x, y, z]*. V této konfiguraci lze používat nástroje pro další detekci scény, a to:

- Detekce horizontálních a vertikálních rovin (viz 3.5.4).
- Detekce předem zvolených 2D obrázků (viz 3.5.5).
- Detekce 3D objektů.
- *Ray-cast*[18] souřadnice z obrazovky pro získání 3D souřadnic odpovídajícího bodu uvnitř 3D scény.

3.5.7.2 ARImageTrackingConfiguration

Konfigurace *ARImageTrackingConfiguration* sleduje pohyb zařízení podobně jako *ARWorldTrackingConfiguration* v šesti osách volnosti. Nevyužívá k tomu *vizuálně-inerciální odometrii*, ale pouze *feed* videa z kamery, jelikož je pohyb zařízení sledován pomocí detekce obrázku ve scéně, z jehož perspektivní transformace lze následně dopočítat pozici a rotaci zařízení (relativně k obrázku).

Při využití této konfigurace je nutné myslet na to, že sledování pohybu zařízení probíhá pouze, pokud je obrázek alespoň částečně kamerou zařízení viditelný – v moment, kdy se obrázek dostane mimo záběr kamery, framework přestane pohyb zařízení detekovat.

[17] *ARWorldTrackingConfiguration* využívá *vizuálně-inerciální odometrii*.

[18] *Raycasting* je metoda, která se používá pro výběr objektů ve 3D scéně. Z výchozího bodu se vyšle určitým směrem paprsek a následně je vypočítáno, s jakými objekty se protne. Pokud se paprsek protne s více objekty, typicky je výsledkem první bod protnutí, tedy objekt, který je nejblíže výchozímu bodu. [41]

Oproti *ARWorldTrackingConfiguration* má tato konfigurace následující výhody:

- Úspora výkonu a energie – jelikož se v této konfiguraci nevyužívají pohybové senzory a výpočet transformace obrázku je méně náročný než pokročilá analýza obrazu, má tato konfigurace menší dopad na spotřebu energie baterie zařízení.
- Využití v pohyblivém prostředí (např. dopravních prostředcích) – protože je tato konfigurace závislá pouze na datech z kamery, lze tuto konfiguraci využívat i v nestabilním prostředí, které by rozhodilo pohybové senzory v druhé konfiguraci.

Sledování pohybu pomocí obrázku s sebou nese i své nevýhody:

- Pohyb zařízení je detekován pouze, když je obrázek v záběru kamery.
- Pro detekci pohybu je nutné přidat do okolí zařízení obrázek, zatímco v druhé konfiguraci lze detekovat pohyb zařízení kdekoliv bez nutnosti přípravy prostředí.

3.5.7.3 Alternativní konfigurace

Kromě již zmíněných konfigurací jsou ve frameworku *ARKit* dostupné i následující konfigurace. Nelze je však použít pro účely této práce, jsou zde uvedeny pouze jako ukázka dalších funkcí, které framework nabízí.

- *ARGeoTrackingConfiguration*
  - Konfigurace je vhodná pro použití ve venkovním prostředí.
  - Virtuální obsah je možné ukotvit na přesné místo v reálném světě pomocí GPS souřadnic.
  - V současné době je dostupná pouze ve vybraných městech (převážně větší města v Americe; v Evropě pouze v Londýně).
- *AROrientationTrackingConfiguration*
  - Konfigurace sleduje pouze rotaci zařízení.
- *ARPositionalTrackingConfiguration*
  - Sleduje pohyb zařízení v 6 osách volnosti.
  - Vhodné pro použití, kdy není potřeba *feed* videa z kamery (např. pro VR).
- *ARBodyTrackingConfiguration*
  - Detekuje osoby a pohyb jejich končetin.
- *ARFaceTrackingConfiguration*
  - Detekuje obličej a jeho výrazy pomocí přední kamery zařízení.

3.5.8 Zvolené nastavení frameworku ARKit

Pro sledování pohybu zařízení byla zvolena konfigurace *ARWorldTrackingConfiguration*, jelikož jako jediná dokáže plně zajistit sledování pohybu zařízení po celé místnosti. V případě využití konfigurace *ARImageTrackingConfiguration* by bylo nutné mít referenční obrázek v záběru kamery po celou dobu sledování.

Konfigurace *ARWorldTrackingConfiguration* bude v aplikaci kombinována s detekcí referenčního obrázku, podle kterého se AR diagram rozvodů na zdi zarovná. Přesná lokalizace rozvodů při opětovném načtení aplikace bude podmíněna položením referenčního obrázku na stejné místo, aby se podle něj mohl zarovnat souřadnicový systém virtuální scény. Tento přístup umožní dosáhnout přesné a stabilní vizualizace AR diagramu rozvodů v reálném prostoru, což zajistí konzistentní vizualizaci během nezávislých měření.

3.6 Framework pro tvorbu 3D obsahu pro ARKit

ARKit poskytuje pouze nástroje pro měření pohybu zařízení a pro analýzu objektů v okolním prostředí. Pro implementaci plnohodnotné AR aplikace je proto nutné zkombinovat *ARKit* s dalším frameworkem, jež vytvoří virtuální obsah, který je následně zobrazen uživateli zkombinovaný s *feedem* videa z kamery. V následujících podkapitolách jsou uvedeny frameworky, které k tomu lze použít.

3.6.1 RealityKit

RealityKit je framework zaměřený na tvorbu 3D obsahu pro použití v AR aplikacích. Vývojářům umožňuje kombinovat informace o reálném světe získané skrze framework *ARKit* s virtuálními objekty. *RealityKit* nabízí následující funkce pro vylepšení AR zážitku uživatele aplikace: [42]

- Uchycení 3D objektů na AR kotvy (viz 3.5.3) frameworku *ARKit*.
- Umístění zvukových zdrojů v 3D prostoru.
- Animace objektů pomocí manuálních i fyzických simulací.
- Možnost tvorby *multiplayer* zážitku pomocí sdílení AR scény mezi několik zařízení.

3D obsah vytvořený ve frameworku *RealityKit* lze zobrazit v AR prostředí pomocí implementace *ARView*, které je potomkem *UIKit UIView*.

S frameworkem *RealityKit* úzce souvisí aplikace *Reality Composer*, která umožňuje uživatelům vytvářet rozšířenou realitu bez nutnosti rozsáhlých znalostí programování. Aplikace je primárně určena pro tvorbu 3D modelů, animací a interakcí pro AR zážitky pomocí jednoduchého rozhraní *drag & drop*. Po dokončení AR scény v *Reality Composer* lze vytvořený obsah snadno exportovat pro použití uvnitř plnohodnotné aplikace. [43]

3.6.1.1 Systém entit

Základním prvkem frameworku *RealityKit* je systém entit. Entita reprezentuje objekty, světla, kamery nebo jiné prvky ve 3D scéně a skládá se z různých komponent, jako jsou například *ModelComponent*, *CollisionComponent* a *TransformComponent*.

Systém entit umožňuje vytvářet hierarchii entit, kde jedna entita může být potomkem jiné entity. Potomek zdědí transformační vlastnosti svého rodiče, což zahrnuje pozici, rotaci a měřítko. Tímto způsobem lze snadno vytvořit složité struktury s entitami, které mají relativní pozice vůči svým rodičům, což zjednodušuje manipulaci s celým skupinami objektů.

3.6.2 SceneKit

SceneKit je framework umožňující tvorbu 3D obsahu pro *Apple* zařízení. Framework poskytuje nástroje pro načtení, manipulaci a vykreslení 3D modelů. Kromě těchto nástrojů *SceneKit* poskytuje funkce pro tvorbu animací, fyzických simulací, *particle* efektů a funkce pro realistické *renderování* obsahu. [44]

3D obsah vytvořený ve frameworku *SceneKit* lze zobrazit v AR prostředí pomocí implementace *ARSCNView*.

3.6.3 SpriteKit

SpriteKit je framework pro tvorbu 2D grafiky. Framework lze použít pro implementaci her a dalších aplikací, které nevyžadují 3D grafické prvky. *SpriteKit* umožňuje kreslení tvarů, přidání textu, obrázků, videa a *particle* efektů do scény. Součástí frameworku jsou nástroje pro tvorbu animací a fyzikálního chování objektů. [45]

SpriteKit grafiku lze zobrazit v AR scéně pomocí implementace *ARSKView*.

3.6.4 Metal

Metal je nízkoúrovňový framework pro tvorbu 3D grafiky pro *Apple* zařízení. Framework poskytuje přímý přístup ke grafickým procesorům zařízení, které vývojář může použít jak pro tvorbu grafiky, tak pro paralelní výpočty. Výše zmíněné frameworky (*RealityKit*, *SceneKit* a *SpriteKit*) *Metal* využívají. *Metal* je vhodné použít, pokud zmíněné frameworky neposkytují dostatečné funkce nebo nejsou dostatečně efektivní pro danou úlohu. [46]

3.6.5 Zvolený framework

Ačkoliv jsou frameworky *SceneKit* a *SpriteKit* již zavedené a tudíž i nabízejí pokročilé grafické nástroje pro tvorbu 3D obsahu (respektive 2D u frameworku *SpriteKit*), byl pro implementaci vyvíjené aplikace zvolen framework *RealityKit*, jakožto framework vyvinutý přímo na míru pro tvorbu AR obsahu.

Kapitola 4

Návrh

Tato kapitola se zabývá návrhem jednotlivých částí vyvíjené aplikace. Popisuje se v ní UIKit navigace, Flow koordinátory, problémy s návrhovým vzorem MVC a řešení v podobě vzoru MVVM. Dále se v kapitole diskutuje název a ikona aplikace, případy užití a návrh uživatelského rozhraní.

4.1 UIKit navigace

Ačkoliv byl v analýze technologií zvolen *SwiftUI* framework, jako primární framework pro implementaci UI vyvíjené aplikace, byl ve výsledné aplikaci použit také framework *UIKit*, a to pro implementaci navigace mezi jednotlivými *SwiftUI* obrazovkami. Přestože lze implementovat navigaci mezi obrazovkami čistě skrze framework *SwiftUI*, byla zvolena implementace v *UIKit* z následujících důvodů:

- Stabilita a spolehlivost – Navigace v *UIKit* je osvědčené řešení, které bylo testováno a vylepšováno po mnoho let, což přináší větší stabilitu a spolehlivost oproti *SwiftUI* navigaci.
- Řešení problémů s vnořenou navigací – *UIKit* navigace snáze zvládá vnořenou navigaci a složitější navigační hierarchie.
- Delegace správy navigace mimo *View* – Oddělení navigace od *View* usnadňuje správu kódu a dodržuje zásady jedné zodpovědnosti. Tímto způsobem je dosaženo čistější a modulárnější architektury aplikace.

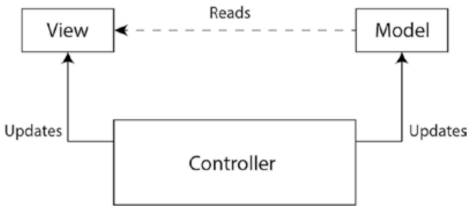
4.2 Návrhový vzor

Pro vývoj aplikace byl zvolen návrhový vzor MVVM (*Model-View-View Model*), který v *iOS* vývoji[1] nahrazuje dříve využívaný vzor MVC (*Model-View-Controller*). V následujících sekcích jsou popsány nevýhody návrhového vzoru MVC a jak jsou tyto nevýhody potlačeny použitím návrhového vzoru MVVM (informace čerpány z knihy *iOS Code Testing* od Abhishek Mishra [47][2]).

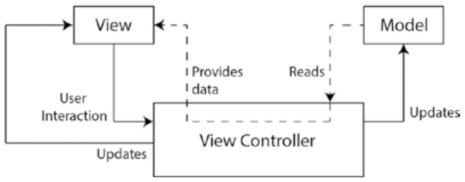
4.2.1 Problémy návrhového vzoru MVC v UIKit aplikacích

Návrhový vzor MVC slouží pro rozdělení kódu aplikace do více vrstev – *View*, *Controller* a *Model* (obr. 4.1 znázorňuje vazby mezi MVC vrstvami). Cílem tohoto oddělení vrstev je dosáhnout větší modularity, efektivity a snadnější údržby a testování kódu aplikace. Nicméně v *UIKit* aplikacích použití MVC *Controlleru* často vede k *masivním Controllerům*, kde se do jedné třídy kombinuje veškerá logika aplikace.

Příčinou popisovaného problému je vlastní koncept *View Controlleru* od společnosti *Apple*, který kombinuje *View* s *Controllerem* do jedné třídy, tím se odpovídající vrstvy návrhového vzoru MVC velmi úzce propojí a struktura návrhového vzoru MVC je porušena (obr. 4.1 znázorňuje typické vazby mezi vrstvami MVC v *UIKit* aplikacích). Jednotlivé *View* se v praxi často pojí pouze s jedním *View Controllerem*, ačkoli ideálně by dle MVC měly být nezávislé a snadno použitelné na více místech.



Obrázek 4.1 Diagram návrhového vzoru MVC (převzato z knihy *iOS Code Testing* [47])



Obrázek 4.2 Typické vazby v MVC uvnitř *UIKit* aplikací (převzato z knihy *iOS Code Testing* [47])

[1] Myšleno vývoji *iOS* aplikací v kombinaci s frameworkem *UIKit*

[2] Autor knihy zmiňuje MVC a MVVM ve spojení s vývojem „obecné“ *iOS* aplikace, jelikož kniha byla vydána v roce 2017 – tedy 2 roky před uvedením frameworku *SwiftUI* – v době, kdy byl dostupný pouze framework *UIKit*.

4.2.2 Návrhový vzor MVVM

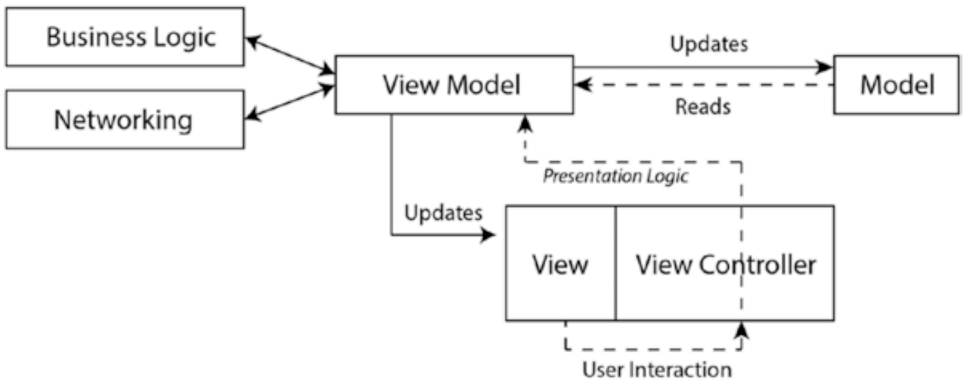
Návrhový vzor MVVM řeší problematiku úzkého propojení mezi *View* a *Controllerem* přidáním nové vrstvy – *View Modelu* – která od *View Controlleru* přebírá prezentační logiku aplikace. Jednotlivé vrstvy MVVM zastávají následující funkce:

- *Model* (M) – Slouží ke správě dat aplikace.
- *View* (V) – Slouží k vizualizaci dat uživatelovi skrze obrazovku. (V *UIKit* aplikacích *View* vrstva označuje *View* společně s *View Controllerem* dohromady.)
- *View Model* (VM) – Spravuje prezentační logiku. Poskytuje obousměrné spojení s *Modelem* a *View*. Komunikuje s dodatečnými *controllery*, které zajišťují specifické funkcionality (např. síťovou komunikaci).

Při správné implementaci návrhového vzoru MVVM nevznikají příliš velké *View Controllery*, jelikož veškerá prezentační logika je přesunutá do *View Modelu* a případné specifické funkcionality zajišťuje *View Model* komunikací s dedikovanými *controllery* (struktura vazeb MVVM v *UIKit* je vyobrazena na obr. 4.3). Pro stanovení jednoznačného komunikačního rozhraní mezi jednotlivými vrstvami MVVM se ve frameworku *UIKit* využívají protokoly, které deklarují, jaké vstupy a výstupy bude komunikační rozhraní poskytovat.

Pevné oddělení jednotlivých MVVM vrstev umožňuje vývojářům používat jednotlivé komponenty na více místech v aplikaci, jelikož nejsou silně vázány na vedlejší vrstvy. Druhou výhodou je i možnost snadnějšího testování aplikace:

- *View Model* neobsahuje žádné reference na *UIKit* prvky, proto jej lze snadno testovat pomocí *unit* testů.
- Při použití protokolů je jednoznačně definované komunikační rozhraní mezi jednotlivými vrstvami. Vrstvy lze testovat odděleně pomocí *mockingu* dat zbylých vrstev.



Obrázek 4.3 Diagram návrhového vzoru MVVM v *UIKit* aplikaci (převzato z knihy *iOS Code Testing* [47])

4.2.2.1 MVVM ve SwitUI

SwiftUI pracuje v souladu s MVVM implicitně, díky konceptům, které framework od základu používá. *SwiftUI Views*[3] jsou struktury, jejichž obsah nelze měnit přímo. Oproti frameworku *UIKit*, kde je nutné při každé změně dat upravit samotné *View*, jsou *SwiftUI Views immutable* – tedy nelze je po jejich vytvoření upravovat. Pokud vývojář vyžaduje, aby *SwiftUI View* reagovalo na změny v datech, je nutné zadefinovat stavové proměnné typu *@State*, na jejichž změny *View* reaguje svým překreslením. V nejjednodušších případech tyto *@State* proměnné odpovídají *VM* vrstvě MVVM. Nejčastěji se však *SwiftUI* využívá v kombinaci s plnohodnotnými *View Modely* typu *ObservableObject*, na jejichž změny *View* reaguje stejně – tedy svým překreslením. [48]

4.3 Flow koordinátory

Flow koordinátory[4] jsou návrhový vzor, který pomáhá řídit navigační tok v aplikaci, zejména při použití *UIKit* frameworku. Umožňují oddělení zodpovědnosti tím, že řídí navigační logiku aplikace nezávisle na *View Controllerech*, což zjednodušuje kód a činí jej více modulárním, udržitelným a testovatelným. V následujících bodech je popsáno, jak koordinátory fungují v kombinaci s *UIKit* navigací:

- Definice protokolu koordinátoru – Koordinátory jsou obvykle definovány protokolem, který popisuje základní metody a vlastnosti pro správu navigačního toku. Tento protokol může zahrnovat metody pro navigaci do různých *View Controllerů* nebo pro manipulaci s konkrétními uživatelskými akcemi, které vyvolávají navigaci.
- Implementace tříd koordinátorů – Následně jsou vytvořeny konkrétní třídy koordinátorů, které implementují zadefinovaný protokol koordinátora. Tyto třídy jsou zodpovědné za vytváření instancí *View Controllerů*, správu navigačního zásobníku a manipulaci s potřebnou navigační logikou.
- Delegování navigačních zodpovědností – Místo zabudování navigační logiky do *View Controllerů* je navigační zodpovědnost delegována koordinátorovi. Když akce ve *View Controlleru* spustí navigaci, *View Controller* informuje koordinátora, který poté rozhodne, jak postupovat v navigaci.
- Hierarchické koordinátory – Koordinátory mohou být hierarchické, kde nadřazení koordinátoři řídí navigační tok pro specifické sekce nebo moduly aplikace, zatímco podřízení koordinátoři řídí navigaci v rámci těchto sekcí. Tento přístup umožňuje čistou a organizovanou navigační strukturu, která je snadnější k pochopení a údržbě.

Pro implementaci tohoto návrhového vzoru byla zvolena knihovna *ACKategories*[5], která obsahuje funkce a nástroje usnadňující používání koordinátorů – například automatickou správu paměti při zobrazení a ukončení jednotlivých obrazovek. Tato knihovna poskytuje užitečná rozšíření pro

[3] Zde je myšlen *body* atribut *SwiftUI View*.
[4] Z anglického výrazu *Flow Coordinators*.
[5] `https://github.com/AckeeCZ/ACKategories`

UIKit a základní kostru pro koordinátory, která umožňuje efektivní a udržitelnou implementaci navigační logiky v aplikaci.

4.4 Název a ikona aplikace

Jelikož je v seznamu aplikací v obchodě *AppStore* vidět pouze ikona, název a kategorie aplikace[6], je kvalitní návrh názvu a ikony aplikace velmi důležitý aspekt při rozhodování, zda potenciální uživatel na aplikaci klikne a stáhne si ji, nebo kolem ní „*proscrolluje*“ dál bez povšimnutí.

4.4.1 Ikona

Návrh ikony aplikace byl vytvářen v souladu s grafickými doporučeními [49], která *Apple* poskytuje. Podle těchto zásad by ikona aplikace měla splňovat následující body:

- Jednoduchý design
  - Jednoduché ikony jsou pro uživatele snadno identifikovatelné a zapamatovatelné.
- Vyvarovat se použití textových prvků
  - Název aplikace je často zobrazován společně s ikonou – použití názvu v ikoně by duplikovalo tuto informaci.
  - Ikony jsou často malé a text by v nich proto snadno zanikl.
  - Text v ikonách nelze lokalizovat do různých jazyků.
- Nepoužívat fotografie
  - Fotografie obsahují příliš mnoho detailů, které by na malé ikoně zanikly.

Ikona aplikace byla vytvářena ve vektorovém nástroji *Inkscape*[7]. Jako hlavní prvek ikony byly zvoleny dva rozvody – tedy věci, které jsou primárním objektem zájmu vyvíjené aplikace. Rozvody byly vyznačeny pomocí jednoduchých symbolů – kruhů, které značí místa, kde se rozvod ohýbá a končí, a obdélníků, které propojují jednotlivé kruhy. Pro každý rozvod byla zvolena odlišná barva – bledě modrá pro rozvod vody a šedá pro rozvod elektřiny. Rozvody byly doplněny o symbol kapky a blesku, aby bylo více zřetelné, že tvary značí rozvody. Ikona lze však použít i ve verzi bez symbolů (např. v situacích, kde je ikona příliš malá na to, aby symboly byly zřetelně vidět).

Výsledná ikona aplikace je zobrazena[8] na obr. 4.4. Dále je na obr. 4.5, kde je vyobrazena v kombinaci s názvem aplikace.

[6] Tyto informace jsou zobrazené v sekcích obchodu *Hry* a *Aplikace*. Pokud uživatel využívá funkci hledání konkrétního klíčového výrazu, obchod kromě ikony, názvu a kategorie zobrazí i počet stažení a snímky obrazovky aplikace. (Testováno dne 26. 4. 2023)

[7] `https://inkscape.org`

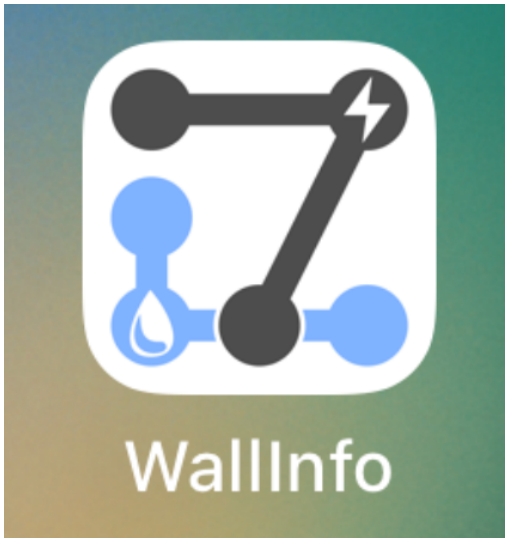
[8] Na obr. 4.4 je kolem návrhu ikony dodatečně přidán černý okraj, aby šlo její pozadí odlišit od bílého podkladu stránky.

Obrázek 4.4 Návrh ikony aplikace

4.4.2 Název

Aby se zvýšila šance instalace aplikace potenciálním uživatelem, byl pro aplikaci zvolen název, který jednak popisuje, k čemu je aplikace přibližně určená, a také název typograficky zajímavý s cílem, aby se potenciální uživatel v seznamu aplikací nad názvem pozastavil a následně si zobrazil detailnější informace o aplikaci.

Jako název aplikace bylo zvoleno anglické slovní spojení *WallInfo*, jehož význam v češtině je *Informace o zdi*. Při použití fontu *SF Pro*[9] využívaného v systému *iOS* je vzhled malého písmena „l“ a velkého písmena „I“ téměř stejný a vzniká tím zajímavý efekt tří stejných znaků za sebou (viz obrázek 4.5).



Obrázek 4.5 Ikona a název aplikace nainstalované v systému *iOS*

[9] `https://developer.apple.com/fonts/`

4.5 Případy užití

Na základě funkčních požadavků a provedené analýzy technologií byly navrhnuty následující případy užití.

4.5.1 UC1 – Nastavení referenčního obrázku

Umožňuje uživateli aplikace nastavit referenční obrázek při prvním spuštění aplikace. Nastavení referenčního obrázku je uživateli dostupné během následného používání aplikace skrze nastavení. Uživatel může nahrát buď vlastní referenční obrázek (UC1.1) nebo využít výchozí referenční obrázek (UC1.2).

4.5.2 UC1.1 – Nastavení vlastní referenčního obrázku

Umožňuje uživateli nastavit vlastní referenční obrázek. Spolu s nahráním obrázku je nutné zadat skutečnou šířku vytištěného obrázku.

4.5.3 UC1.2 – Nastavení výchozího referenčního obrázku

Umožňuje uživateli zvolit přednastavený referenční obrázek. U této volby se předpokládá, že si uživatel poskytnutý obrázek vytiskne a následně do aplikace zadá jeho skutečnou šířku.

4.5.4 UC2 – Seznam místností

Umožňuje uživateli:

- zobrazit seznam všech uložených místností
- přidat novou místnost
- ze seznamu smazat jakoukoli místnost
- přejít na detail místnosti (UC3)

4.5.5 UC3 – Detail místnosti

Umožňuje uživateli:

- zobrazit seznam všech zdí dané místnosti
- přidat novou zeď pro danou místnost
- ze seznamu smazat jakoukoli zeď
- přejmenovat danou místnost
- přejít do seznamu místností (UC2)
- přejít na detail zdi (UC4)

4.5.6 UC4 – Detail zdi

Umožňuje uživateli:

- zobrazit a editovat poznámku o zdi
- přejmenovat danou zeď
- zobrazit seznam obrázků
- přidat novou fotografii do seznamu obrázků
- přejít na detail místnosti (UC3)
- přejít na detail obrázku (UC5)
- přejít do AR editoru rozvodů dané zdi (UC6)

4.5.7 UC5 – Detail obrázku

Umožňuje uživatelovi:

- zobrazit a editovat poznámku o obrázku
- sdílet obrázek pomocí systémového dialogu
- smazat obrázek
- přejít na detail dalších obrázků uložených u stejné zdi
- přejít na detail zdi (UC4)

4.5.8 UC6 – AR editor rozvodů

Umožňuje uživatelovi:

- zaznamenat rozvody různých druhů ve zdech místnosti
- zaznamenat větvení rozvodů
- zobrazit dříve zaznamenaný diagram rozvodů
- posunout jednotlivé body diagramu rozvodů
- smazat jednotlivé body diagramu rozvodů
- vytvořit snímek obrazovky AR vizualizace rozvodů
- přejít na detail zdi (UC4)

4.5.9 UC7 – Nastavení

Umožňuje uživatelovi:

- změnit nastavení AR editoru rozvodů
- přejít do nastavení referenčního obrázku (UC1)

4.6 Uživatelské rozhraní

Na základě sepsaných případů užití (4.5) bylo navrhnuto uživatelské rozhraní aplikace v online nástroji *Figma* (3.1). Při tvorbě návrhu byly využity následující šablony dostupné pod otevřenou licencí *CC BY 4.0*:

- *mockupy* telefonů *iPhone*[10]
- *SwiftUI* UI prvky[11]
- *iOS 16 UIKit* UI prvky[12]

4.6.1 Hlavní struktura aplikace

Aplikace bude obsahovat 2 hlavní sekce, mezi kterými bude možné přecházet pomocí přepínačů umístěných v dolní části obrazovky. První sekce bude umožňovat uživateli přistupovat ke seznamu místností (4.6.5). Druhá sekce umožní uživateli přistupovat k obrazovce s nastavením aplikace (4.6.6). Při prvním spuštění aplikace se jako úvodní obrazovka zobrazí obrazovka s nastavením referenčního obrázku (4.6.2). Při dalších spuštění aplikace bude úvodní obrazovka seznam místností (4.6.5).

4.6.2 První nastavení

Při prvním spuštění aplikace je nutné nastavit referenční obrázek, který se následně použije v AR editoru rozvodů. Uživatel dostane na úvodní obrazovce možnost si vybrat, zda do aplikace nahraje svůj vlastní referenční obrázek, nebo využije obrázek poskytnutý aplikací, který si následně vytiskne. Tato obrazovka bude dostupná i při dalším použití aplikace, a to skrze obrazovku nastavení (4.6.6). Kliknutím na tlačítko „Použít vlastní obrázek“ se zobrazí obrazovka 4.6.3, respektive po kliknutí na „Použít výchozí obrázek“ bude zobrazena obrazovka 4.6.4. Návrh úvodní obrazovky je znázorněn vlevo na obrázku 4.6.

[10] Hosam Elshazly – *iPhone Mockups*: `https://www.figma.com/community/file/1204924066786787184`
[11] Joey Banks – *SwiftUI Input Kit*: `https://www.figma.com/community/file/864234074226183072`
[12] Joey Banks – *iOS 16 UI Kit for Figma*: `https://www.figma.com/community/file/1121065701252736567`

4.6.3 První nastavení – Vlastní referenční obrázek

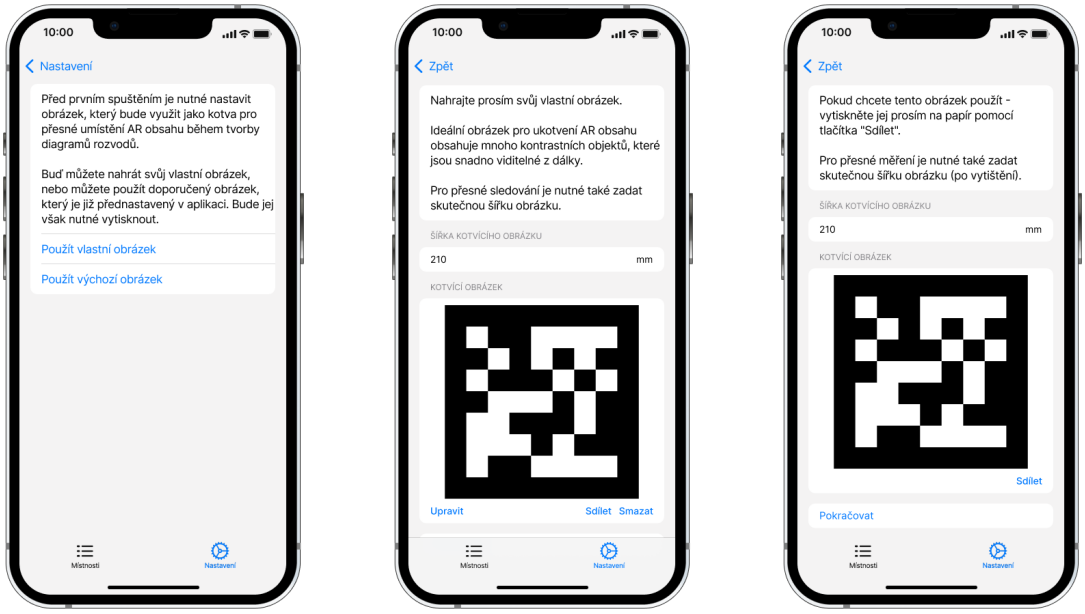
Uživatel bude moci na této obrazovce nahrát svůj vlastní referenční obrázek. K výběru správného obrázku ho bude instruovat návod v horní části obrazovky. Obrazovka bude obsahovat i pole pro vyplnění velikosti zvoleného obrázku. Návrh obrazovky s nastavením vlastního referenčního obrázku je znázorněn uprostřed na obrázku 4.6.

Poté, co uživatel nahraje obrázek, se aktivuje tlačítko „Pokračovat“, které uživatele přesměruje na obrazovku seznamu místností (4.6.5). Pokud se uživatel dostal na obrazovku pomocí navigační *flow* skrze nastavení aplikace, tlačítko „Pokračovat“ uživatele přesměruje na úvodní obrazovku nastavení (4.6.6). Uživatelovi bude umožněno jít v navigaci zpět na výběr způsobu nahrání obrázku (4.6.2).

4.6.4 První nastavení – Výchozí referenční obrázek

Tato obrazovka slouží ke zvolení výchozího referenčního obrázku. Jako výchozí referenční obrázek bude nastaven *ArUco marker*, který měl v provedených testech (viz kapitola 3.5.6) nejlepší výsledky. Poskytnutý obrázek bude moci uživatel vytisknout pomocí tlačítka „Sdílet“, které otevře systémový dialog pro sdílení souborů – pokud bude mít uživatel dostupnou tiskárnu s technologií *AirPrint*, zobrazí se tiskárna v nabídce, případně lze obrázek odeslat na jiné zařízení, které má přístup k tiskárně. Kromě obrázku obrazovka obsahuje vstupní textové pole, kam uživatel zadá šířku obrázku poté, co ho vytiskne. Návrh této obrazovky je znázorněn vpravo na obrázku 4.6.

Kliknutím na tlačítko „Pokračovat“ se nastavení uloží a uživatel bude přesměrován na obrazovku se seznamem místností (4.6.5), respektive na úvodní obrazovku nastavení (4.6.6). Uživatelovi bude umožněno jít v navigaci zpět na výběr způsobu nahrání obrázku (4.6.2).



Obrázek 4.6 Nastavení referenčního obrázku

4.6.5 Seznam místností

Tato obrazovka slouží pro zobrazení seznamu všech uložených místností v aplikaci. Na konci seznamu je vstupní textové pole, jehož vyplněním je vytvořen záznam nové místnosti. Záznam místnosti lze smazat tahem prstu z pravé části záznamu směrem doleva. Návrh obrazovky je vyobrazen na obrázku 4.7.

Kliknutím na záznam místnosti v seznamu bude uživatel přesměrován na obrazovku s detailem dané místnosti (4.6.7).

Obrázek 4.7 Seznam místností

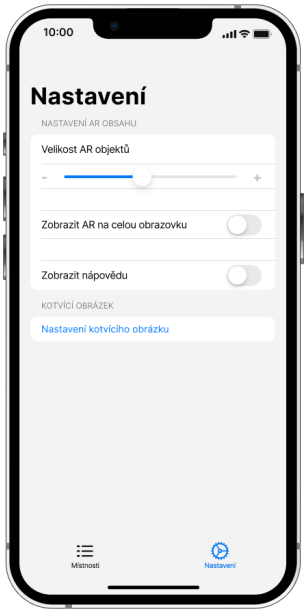
4.6.6 Nastavení

Tato obrazovka umožňuje uživatelovi změnit nastavení aplikace – primárně nastavení AR editoru rozvodů. Návrh obrazovky je vyobrazen na obrázku 4.8.

Prvním nastavením je posuvník pro škálování velikosti AR obsahu. Efekt tohoto nastavení je ukázán porovnáním obrázku vlevo a uprostřed 4.11.

Dalším nastavením je přepínač, pomocí kterého lze nastavit, zda se AR editor zobrazí na celou obrazovku (s tím, že část záběru kamery zůstane nevyužita), nebo se AR editor zobrazí tak, aby byl využit plný potenciál zorného pole kamery. Efekt tohoto přepínače lze vidět při porovnání levého a pravého obrázku 4.11.

Posledním nastavením je přepínač, pomocí kterého lze nastavit, zda se při spuštění AR editoru zobrazí nápověda ohledně vhodného umístění referenčního obrázku na zeď. Kromě těchto nastavení je na obrazovce odkaz na obrazovku s nastavením referenčního obrázku (4.6.2).



Obrázek 4.8 Nastavení

4.6.7 Detail místnosti

Na této obrazovce je zobrazen seznam zdí dané místnosti. Na konci seznamu je vstupní textové pole, jehož vyplněním je vytvořen záznam nové zdi. Záznam zdi lze smazat tahem prstu z pravé části záznamu směrem doleva. Název místnosti lze upravit pomocí ikony tužky v pravém horním rohu obrazovky. Návrh obrazovky je vyobrazen na obrázku 4.9.

Kliknutím na záznam zdi v seznamu bude uživatel přesměrován na obrazovku s detailem dané zdi (4.6.8).

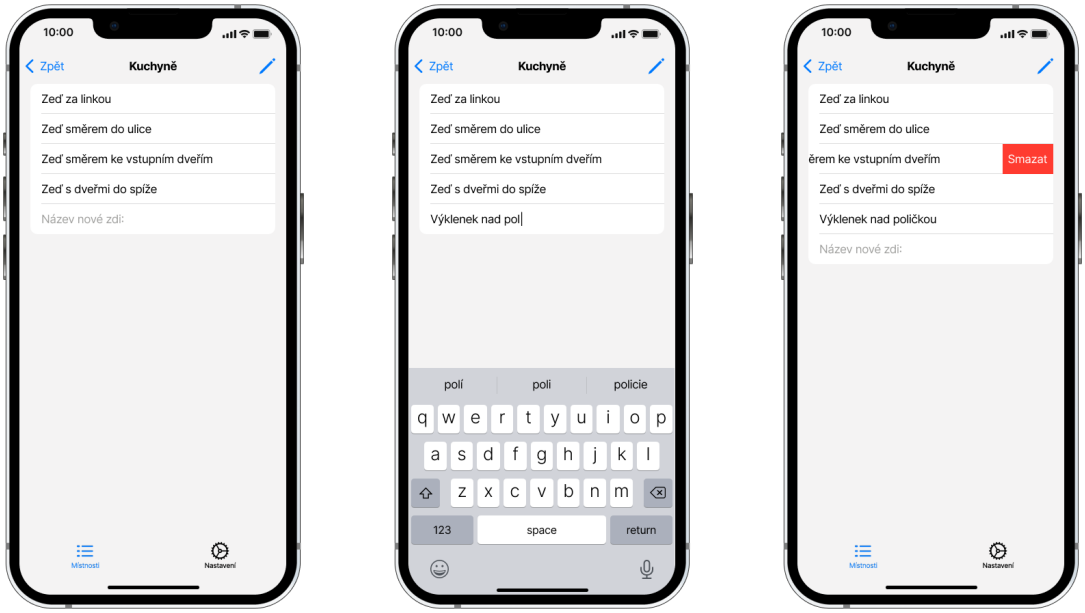
4.6.8 Detail zdi

Na této obrazovce jsou zobrazeny informace o dané zdi. Návrh obrazovky je znázorněn vlevo na obrázku 4.10.

Součástí obrazovky je textové pole pro poznámku, kam si uživatel může například zapsat dodatečné informace o zaznamenaných rozvodech.

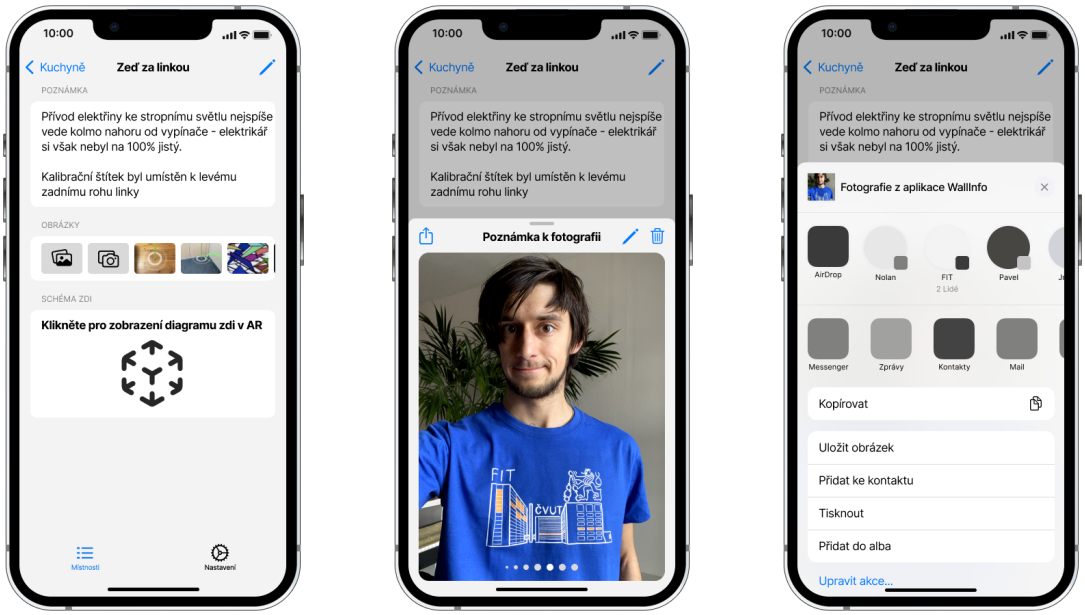
Pod textovým polem je galerie s obrázky. Obrázky lze do galerie přidat buď pomocí výběru z galerie telefonu, pomocí nové fotografie nebo pomocí vytvoření snímku obrazovky v AR editoru rozvodů dané zdi. Galerií s obrázky lze horizontálně posouvat, pokud je v ní uloženo více obrázků, než se vejde na obrazovku.

Kliknutím na obrázek v galerii se zobrazí velký náhled zvoleného obrázku (viz obrázek 4.10 uprostřed). Náhledem obrázků lze horizontálně posouvat pro zobrazení jednotlivých uložených fotografií. Náhled obrázku obsahuje 3 tlačítka. Prvním je tlačítko pro sdílení, které otevře systémový dialog pro sdílení (viz obrázek 4.10 napravo). Druhé tlačítko umožňuje uživatelovi upravit popisek obrázku. Poslední tlačítko vymaže daný obrázek z galerie zdi.



Obrázek 4.9 Detail místnosti

Posledním prvkem obrazovky detailu zdi je tlačítko pro spuštění AR editoru rozvodů (4.6.9).



Obrázek 4.10 Detail zdi

4.6.9 AR editor rozvodů

Tato obrazovka umožňuje uživateli aplikace zaznamenat polohu rozvodů v AR prostředí. Jednotlivé rozvody jsou v editoru zaznamenány pomocí bodů a spojnic. Body značí místa, kde rozvod končí, mění směr, nebo se větví. Spojnice pak propojují jednotlivé body. AR editor rozvodů je znázorněn na obrázcích 4.11 a 4.12.

4.6.9.1 Výběr typu rozvodu

AR editor rozvodů umožňuje uživatelovi zaznamenat odlišné typy rozvodů. Před zaznamenáním nového rozvodu je proto nutné vybrat typ rozvodu. Typ rozvodu lze vybrat pomocí tlačítka v levé spodní části obrazovky, které zobrazí nabídku dostupných rozvodů. Vybraný typ rozvodu je následně znázorněn ikonkou na samotném tlačítku výběru. Typy jednotlivých rozvodů v AR editoru jsou následně odlišeny svým barevným provedením. Nabídka s výběrem typů rozvodů je vyobrazena vlevo na obrázku 4.12.

4.6.9.2 Tvorba záznamu rozvodu

Kliknutím na bod rozvodu se daný bod označí jako aktivní. V editoru je aktivní bod vyznačený oranžovým čtvercem. Kliknutím mimo zvolený bod se aktivní označení zruší.

Body rozvodů se do diagramu přidávají pomocí tlačítka „+“. Nový bod je vytvořen na rovině detekované zdi v místě, kam ukazuje modrá značka uprostřed obrazovky. Pokud je v editoru vybraný aktivní bod, vytvoří se při tvorbě nového bodu propojení mezi novým bodem a aktivním bodem. Pokud před stisknutím „+“ není vybraný aktivní bod, vytvoří se v diagramu samostatný bod bez žádných napojení. Po vytvoření nového bodu je vzniklý bod označený jako aktivní. Proces tvorby nového záznamu rozvodu je vyobrazen na obrázku 4.12.

4.6.9.3 Smazání rozvodu

Další funkcí AR editoru rozvodů je tlačítko s ikonou koše, jehož stisknutím je smazán aktivní bod (včetně všech navázaných propojení). Pokud je s aktivním bodem propojený právě jeden další bod, je po smazání aktivního bodu nastavený jako aktivní tento bod.

4.6.9.4 Napojení na existující body

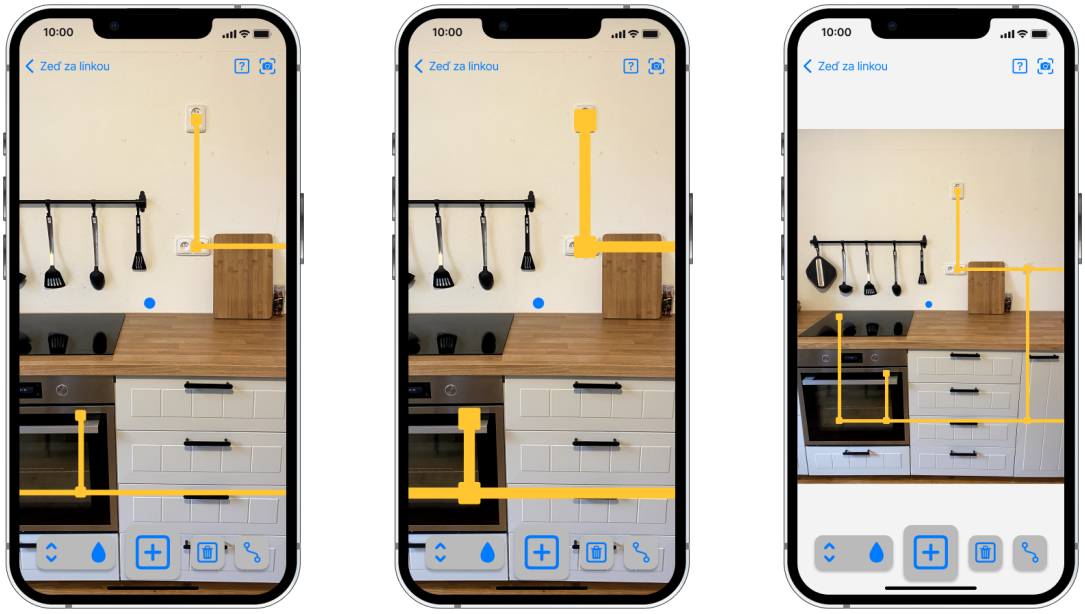
Pravé spodní tlačítko na obrazovce AR editoru zapíná a vypíná funkci, která umožňuje uživateli vytvořit propojení na již existující body diagramu. Když je funkce zapnutá a při stisknutí tlačítka „+“ je pod modrým ukazatelem již zaznamenaný bod, nevytvoří se na daném místě další bod, místo toho se použije existující bod. Když je funkce vypnutá, nový bod se vytvoří pokaždé, i v případě, kdy pod modrým indikátorem bod existuje.

4.6.9.5 Další funkce editoru

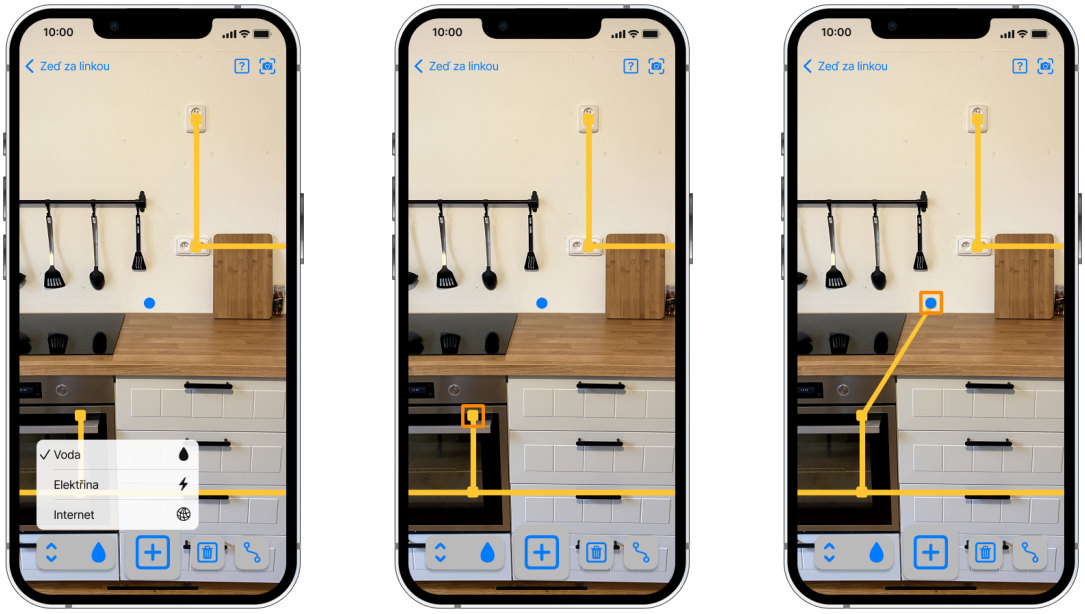
V pravém horním rohu obrazovky je tlačítko se symbolem fotoaparátu. Stisknutím tohoto tlačítka se vytvoří snímek obrazovky, který je následně uložen do galerie obrázků v detailu zdi (4.6.8). Na snímku obrazovky se zachytí pouze obsah AR scény bez UI prvků.

Posledním tlačítkem na obrazovce editoru je tlačítko „?“, které uživatelovi zobrazí nápovědu, ve které jsou stručně popsané jednotlivé funkce editoru.

Kromě výše popsaných funkcí, které se aktivují stiskem tlačítek, obsahuje editor funkci pro posun bodů. Již vytvořené body lze tahem prstu posunout na novou polohu, i pokud na sobě mají spojení na další body.



Obrázek 4.11 AR editor rozvodů (vliv nastavení)



Obrázek 4.12 AR editor rozvodů (editace diagramu rozvodů)

Kapitola 5

Implementace

V této kapitole se podrobně rozebírá implementace aplikace, od struktury projektu až po implementaci jednotlivých funkcí. Kapitola obsahuje popis implementace perzistence dat, repository modelů, lokalizace, obrazovek a AR editoru rozvodů.

5.1 Struktura projektu

Implementace prototypu aplikace má následující strukturu souborů:

- `WallInfo`
  - `AppDelegate.swift` ..... řídí životní cyklus a inicializaci aplikace
  - `AppFlowCoordinator.swift` ..... řídí inicializaci navigační *flow* aplikace
  - `Assets.xcassets` ..... katalog digitálních datových zdrojů
  - `CoreData` ..... adresář perzistentní datové vrstvy
  - `Extensions` ..... adresář s rozšířeními použitých knihoven
  - `Features`
    - `Repository` ..... adresář *repository* vrstvy
    - `Views` ..... adresář s implementacemi jednotlivých obrazovek
      - `ARViews` ..... adresář obrazovky AR editoru rozvodů
      - `RoomDetail` ..... adresář obrazovky detailu místnosti
      - `RoomsList` ..... adresář obrazovky seznamu místností
      - `Settings` ..... adresář obrazovek nastavení aplikace
      - `StartAnimation` ..... adresář obrazovky úvodní animace
      - `Utilities` ..... adresář s univerzálními komponentami
      - `WallDetail` ..... adresář obrazovky detailu zdi
  - `Info.plist` ..... obsahuje metadata konfigurace aplikace
  - `Localization` ..... adresář s překlady
  - `RealityFiles` ..... adresář s 3D modely

```swift
private func registerUserDefaults() {
    /**
     Registers user defaults, which will be used and sets their default value.
     */

    let defaults = UserDefaults.standard

    defaults.register(
        defaults: [
            "Display AR Info text": true,
            "ARView Crop": true,
            "AR Scale": 2.5,
            "Tracked Image Width": 100,
            "Automatically save tracker position": true,
            "Save screenshots to Photos": true
        ]
    )
}
```

Výpis kódu 3 Inicializace výchozích hodnot nastavení aplikace v *UserDefaults*

5.2 Perzistence dat

Pro perzistentní uložení dat byly v aplikaci použity následující dvě technologie: *UserDefaults* a *CoreData*. *CoreData* mohou být využita pro ukládání velkého množství datových typů – od primitivů až po celé soubory – k uloženým datům poskytují přístup skrze objektově orientované *Objective-C* rozhraní. *UserDefaults* na druhou stranu nabízí jednoduchý klíč-hodnota mechanismus pro ukládání dat, který může být využit například pro ukládání uživatelských preferencí jako je jazyk, země nebo nastavení aplikace.

5.2.1 UserDefaults

V aplikaci byly použity *UserDefaults* pro uložení hodnot nastavení aplikace. Aby se zabránilo problémům s neinicializovanými hodnotami během běhu aplikace, jsou při prvním spuštění aplikace registrovány jednotlivé klíče dohromady s výchozími hodnotami (viz ukázka kódu 3)

5.2.1.1 Popis použitých UserDefaults hodnot

Zde je popis jednotlivých *UserDefaults* hodnot použitých v aplikaci:

- *Display AR Info text*
  - určuje, zda se při spuštění AR editoru rozvodů zobrazí upozornění, aby si uživatel poznamenal pozici referenčního obrázku
- *ARView Crop*
  - určuje zda se AR editor ořízne, tak aby zaplnil celou obrazovku

- *AR Scale*
  - hodnota škálování velikosti generovaného obsahu v AR
- *Tracked Image*
  - nastavený referenční obrázek
- *Tracked Image Width*
  - šířka referenčního obrázku v mm
- *Automatically save tracker position*
  - určuje, zda se při detekci referenčního obrázku automaticky vytvoří snímek obrazovky
- *Save screenshots to Photos*
  - určuje, zda se snímky obrazovky zachycené v AR editoru uloží také do galerie fotek telefonu

5.2.1.2 Přístup k UserDefaults hodnotám

Pro přístup k uloženým datům byly v aplikaci použity 2 přístupy.

Ve *SwiftUI ViewModelech* byl použit wrapper *@AppStorage* z frameworku *SwiftUI*, který umožňuje přímé propojení mezi hodnotou v *UserDefaults* a *SwiftUI View*. Použitím tohoto wrapperu se *View* automaticky překreslí, když se hodnota v *UserDefaults* změní. Ukázka použití *@AppStorage* wrapperu je v následujícím výpisu kódu 4.

```
1 @AppStorage("AR Scale", store: .standard) var arScale: Float = 2.5
```

Výpis kódu 4 Ukázka použití *@AppStorage* wrapperu

Mimo *SwiftUI Views* byl v aplikaci použit druhý způsob, který k hodnotám v *UserDefaults* přistupuje pomocí funkcí *get* a *set*. Tento způsob je ukázán ve výpisu kódu 5. V ukázce se ve funkci *getArScale()* nejdříve kontroluje, zda je daný klíč již inicializovaný, aby se předešlo čtení neexistující hodnoty, které by mohlo způsobit chyby v aplikaci.

5.2.2 CoreData

Pro perzistentní uložení dat o zaznamenaných místnostech byl v aplikaci použit framework *CoreData*. V tomto frameworku byly vytvořeny datové modely, které zadefinovaly entity a vztahy mezi nimi. Každá entita odpovídá jedné tabulce v relační databázi a má definované atributy, které odpovídají sloupcům v této tabulce. V aplikaci byly vytvořeny následující *CoreData* entity:

- *RoomDataModel*
  - drží základní informace o záznamu místnosti

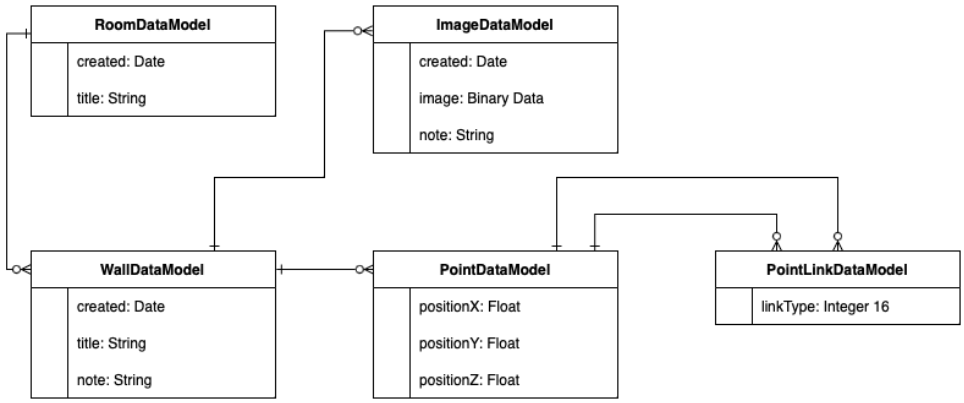
```swift
func getArScale() -> Float? {
    if UserDefaults.standard.object(forKey: "AR Scale") != nil {
        return UserDefaults.standard.float(forKey: "AR Scale")
    }
    return nil
}

func setArScale(scale: Float) {
    UserDefaults.standard.set(scale, forKey: "AR Scale")
}
```

Výpis kódu 5 Ukázka načtení a uložení dat do *UserDefaults*

- může mít vazby na 0–n zdí

- *WallDataModel*
  - drží základní informace o záznamu zdi
  - má právě jednu vazbu na záznam místnosti
  - může mít vazby na 0–n obrázků
  - může mít vazby na 0–n bodů diagramu rozvodů zdi
- *ImageDataModel*
  - drží základní informace o záznamu obrázku
  - má právě jednu vazbu na záznam zdi
- *PointDataModel*
  - drží základní informace o záznamu bodu diagramu rozvodů
  - má právě jednu vazbu na záznam zdi
  - může mít vazby na 0–n propojení bodů diagramu rozvodů
- *PointLinkDataModel*
  - drží základní informace o záznamu propojení dvou bodů diagramu rozvodů
  - drží právě 2 vazby na záznamy bodů diagramu rozvodů
  - jedná se o implementaci M-N vazby pomocí spojovací tabulky

Atributy a vazby těchto entit jsou vyobrazeny na obrázku 5.1, na kterém je znázorněno ERD schéma těchto modelů.



Obrázek 5.1 ERD schéma *CoreData* modelů

5.2.2.1 PersistenceController

V aplikaci byla implementována struktura *PersistenceController* pro správu *CoreData* entit. *PersistenceController* slouží jako abstrakční vrstva pro manipulaci s *CoreData* stackem např. pro provádění operací uložení a načítání dat z trvalého úložiště.

PersistenceController byl implementován jako *singleton*, což zajišťuje, že v apliakci se využívá pouze jedna jeho instance, což umožňuje konzistentní a centralizovanou správu dat v celé aplikaci.

Níže jsou uvedeny funkce controlleru:

- *Inicializace CoreData stacku* – *PersistenceController* inicializuje *CoreData* stack vytvořením instance kontejneru *NSPersistentContainer*, přes který je následně přistupováno ke *CoreData* modelům
- *Uložení kontextu* – Metoda *saveContext()* je zodpovědná za uložení změn v *CoreData* modelech.
- *Načítání dat* – *PersistenceController* poskytuje metody pro načítání dat z trvalého úložiště – *getAllRooms()*, *loadWalls(room: Room)*, *loadImages(wall: Wall)* a *loadPoints(wall: Wall)*. Tyto metody usnadňují získávání dat z *CoreData* kontejneru strukturovaným způsobem
- *Mapování dat* – *PersistenceController* mapuje data mezi *CoreData* modely a *Repository* modely, což usnadňuje práci s daty v aplikaci. (Repository modely jsou podrobněji popsány v následující kapitole 5.3.)

5.3 Repository

Perzistentní vrstva *CoreData* modelů a prezenční vrstva *ViewModelů* jednotlivých obrazovek je v aplikaci propojena pomocí *Repository* vrstvy, která slouží pro implementaci univerzálních funkcí modelů, jež jsou využívané ve více *ViewModelech*, ale zároveň se nehodí do *CoreData* modelů, aby byla oddělena zodpovědnost správy perzistentního úložiště od implementace dodatečných funkcí nad modely.

Zde je přehled některých funkcí, které mají *Repository* modely na starost:

- správa vztahů mezi instancemi modelů
- implementace protokolů (např. *Identifiable* a *Equatable*)
- správa *RealityKit* entit zobrazených v AR scéně

Pro následující *CoreData* modely byly v aplikaci implementovány odpovídající *Repository* modely:

- *RoomDataModel* – *Room*
- *WallDataModel* – *Wall*
- *ImageDataModel* – *ImageModel*[1]
- *PointDataModel* – *Point*
- *PointLinkDataModel* – *PointLink*

5.4 Lokalizace

Jedním z nefunkčních požadavků bylo, aby byl obsah aplikace lokalizovaný do českého a anglického jazyka. K implementaci tohoto požadavku byl použit vestavěný nástroj ve vývojovém prostředí *Xcode*, pomocí kterého lze vygenerovat seznam všech textových řetězců, které se v aplikaci vykreslují na obrazovku. Překlady jednotlivých hodnot se ukládají do souborů s příponou *.strings*. Tyto soubory pak tvoří slovník, kde na jedné straně je klíč reprezentující původní textovou hodnotu v kódu aplikace a na druhé straně je překlad hodnoty v cílovém jazyce. Tyto soubory jsou následně zaregistrovány v konfiguraci aplikace. Ve výpisu kódu 6 je ukázka tohoto souboru. Přepnutí jazyka aplikace probíhá automaticky při jejím spuštění – pokud je jazyk systému zařízení registrovaný v konfiguraci aplikace, zobrazí se aplikace v daném jazyce, pokud neexistuje pro systémový jazyk překlad aplikace, zobrazí se aplikace ve výchozím jazyce.

```
1 /* No comment provided by engineer. */
2 "Display help" = "Zobrazit nápovědu";
3
4 /* No comment provided by engineer. */
5 "Don't show again" = "Nezobrazovat znovu";
6
7 /* No comment provided by engineer. */
8 "Done" = "Hotovo";
```

Výpis kódu 6 Ukázka překladů ve slovníku *Localizable.strings*

[1] Jelikož je v aplikaci často používáno *SwiftUI View* – *Image*, byl pro *Repository* model zvolen název *ImageModel*, aby se nemusely řešit kolizní jména tříd (např. použitím *typealias*).

5.4.1 Registrace textových řetězců

Při lokalizaci aplikace je klíčové zajistit, aby byly všechny textové řetězce, které mají být přeloženy, řádně zaregistrovány.

Textové řetězce zadefinované přímo v jednotlivých *SwiftUI Views* není nutné pro lokalizaci dodatečně registrovat, jelikož ve *SwiftUI* probíhá registrace zobrazovaných řetězců automaticky. Například následující *SwiftUI* tlačítko – `Button("Done"){}` – automaticky zobrazí českou hodnotu „Hotovo“ (za předpokladu, že je systémový jazyk čeština a v aplikaci je použit výše zmíněný slovník 6).

Textové řetězce inicializované mimo prostředí *SwiftUI* je nutné dodatečně zaregistrovat, aby se při běhu aplikace použila přeložená hodnota. Například místo takto inicializované *String* proměnné – `let text: String = "Done"` – je nutné použít speciální *String* inicializátor – `let text: String = String(localized: "Done")`.

5.5 Obrazovky

Implementace jednotlivých obrazovek aplikace obsahují následující strukturu souborů[2]:

- *FlowCoordinator*
  - stará se o navigační *flow* dané obrazovky
- *View*
  - implementace obrazovky ve frameworku *SwiftUI*
- *ViewController*
  - řídí životní cyklus *View*
- *ViewModel*
  - prezenční vrstva poskytující data pro *View*

Popis jednotlivých souborů je podrobněji rozepsán v následujících sekcích. Součástí výše zmíněné struktury souborů obrazovek jsou pouze V (*View*) a VM (*ViewModel*) části návrhového vzoru MVVM, jelikož datová část M (*Model*) není nijak závislá na dané obrazovce a je v aplikaci samostatně.Pro ukázku je zde zobrazena struktura souborů obrazovky detailu místnosti:

```
RoomDetailFlowCoordinator.swift
RoomDetailView.swift
RoomDetailViewController.swift
RoomDetailViewModel.swift
```

[2]Kromě implementace obrazovky AR editoru rozvodů, který obsahuje další soubory použité pro implementaci AR funkcí.

5.5.1 FlowCoordinator

Flow koordinátory propojují implementace jednotlivých obrazovek aplikace do jednoho celku tím, že určují navigační tok mezi jednotlivými obrazovkami. Tímto způsobem je zajištěna konzistentní navigace a přehlednost celé aplikace, což usnadňuje udržitelnost kódu a případné rozšíření funkcí v budoucnu. *Flow* koordinátory zohledňují také možné závislosti mezi obrazovkami a poskytují flexibilní řešení pro správu navigace a sdílení dat mezi jednotlivými částmi aplikace.

Pro jednu obrazovku může být implementováno více *flow* koordinátorů. Tento přístup je praktický například v případě, když je implementace obrazovky použita na odlišných místech aplikace. K usnadnění implementace více *flow* koordinátorů pro jednu obrazovku se využívají protokoly, jimiž je zadefinováno rozhraní, které musí třída koordinátora implementovat.

Níže je ve výpisu kódu 7 zobrazen příklad protokolu *flow* koordinátora, který řídí navigaci obrazovky, kde uživatel nastavuje referenční obrázek. Na této obrazovce jsou dvě tlačítka, která jsou určena pro přesměrování uživatele na další obrazovky navigační *flow* – pro každé tlačítko je pak stanovena jedna z následujících metod v protokolu:

```
protocol FirstSetupFlowDelegate: AnyObject {
    func showCustomImage()
    func showPreloadedImage()
}
```

Výpis kódu 7 Protokol definující rozhraní *flow* koordinátora

5.5.2 View

V souborech *View* je implementováno uživatelské rozhraní jednotlivých obrazovek pomocí frameworku *SwiftUI*. Každá implementace *View* si udržuje vazbu jednak na *flow* koordinátor, který řeší navigaci obrazovky, a následně na *ViewModel*, který slouží pro přípravu prezentovaných dat pro *View*.

5.5.3 ViewController

S využitím *UIHostingController* je *SwiftUI View* hostováno v kontextu *UIKit*, což umožňuje použití *UIKit* navigace namísto navigace *SwiftUI*. *ViewController* tak zajišťuje implementaci navigačních prvků, jako jsou navigační titulky a tlačítka v navigační liště.

V *ViewControlleru* jsou také volány akce, které se odehrají před tím, než se obrazovka zobrazí. Tyto akce zahrnují například načítání potřebných prezentovaných dat z perzistentního úložiště. Díky tomuto přístupu je možné zajistit, že uživatelské rozhraní je připraveno a zobrazuje aktuální data ihned po zobrazení obrazovky.

Použití *ViewControlleru* pro integraci *SwiftUI View* s *UIKit* navigací umožňuje využít nejlepší vlastnosti obou frameworků. Navíc, díky oddělení navigace a prezentace dat od samotného

uživatelského rozhraní, je možné snadněji rozšiřovat a upravovat jednotlivé části aplikace bez negativního dopadu na celkovou strukturu a funkcionalitu.

5.5.4 ViewModel

ViewModel je klíčovou součástí architektury aplikace a slouží k přípravě a správě dat, která jsou následně prezentována ve *View*. Jeho rozhraní je definováno protokolem *ViewModeling*, což například usnadňuje mockování *ViewModel* dat při testování *View*.

Důležitou vlastností *ViewModelu* je absence vazeb na *View*. Díky tomu může být *ViewModel* nezávisle testován a upravován bez ohledu na změny v uživatelském rozhraní. Tato vlastnost zvyšuje modularitu aplikace a usnadňuje její údržbu a vývoj.

ViewModel je potomkem třídy *ObservableObject*, což umožňuje automatické sledování změn v jeho *@Published* atributech. *View* tak reaguje na změny v těchto atributech a překreslí se automaticky, což zajišťuje konzistentní zobrazení dat v uživatelském rozhraní.

ViewModel slouží jako spojovací článek mezi *View* a zbytkem aplikace, jako jsou například úložiště dat nebo logika pro získávání dat z externích zdrojů. Tímto způsobem je zajištěna dekompozice a oddělení zodpovědností mezi jednotlivými vrstvami aplikace, což přispívá k lepší udržitelnosti a snazšímu rozšiřování funkcí aplikace.

V následujícím výpisu kódu 8 je zobrazena implementace *ViewModeling* protokolu obrazovky detailu místnosti. Protokol definuje přístup k proměnným *room* a *newWallTitle* pro čtení a zápis, zatímco vlastnost *title* je pouze pro čtení. Dále protokol specifikuje několik funkcí, které umožňují manipulaci s daty, jako je přidání nové stěny nebo mazání stěn na základě zadaných indexů.

```
protocol RoomDetailViewModeling: ObservableObject {
    var room: Room { get set }
    var newWallTitle: String { get set }
    var title: String { get }

    func addNewWall() -> Wall?
    func delete(at offsets: IndexSet)
}
```

Výpis kódu 8 Protokol definující rozhraní *ViewModelu* obrazovky detailu místnosti

5.6 AR editor rozvodů

Hlavní funkcí aplikace je AR editor rozvodů (dále jen editor). Implementace editoru má dvě oddělené části. První částí je samotné *ARView* (5.9), druhou částí je *SwiftUI View* (5.7), v němž jsou implementované UI prvky, které ovládají funkce editoru. Tyto části jsou propojené ve *SwiftUI View* – *WallInfoARView* – které obě části kombinuje jako vrstvy v *ZStack View*. Ve spodní vrstvě je *ARView* a v horní vrstvě jsou umístěny *SwiftUI* ovládací prvky editoru.

Jelikož je *ARView UIKit View*, je přidáno do *SwiftUI ZStacku* prostřednictvím *ImageTrackingARViewControllerRepresentable* komponenty, která umožnuje zobrazit *UIKit* prvky ve *SwiftUI View*. Komunikace mezi popsanými částmi editoru je implementována pomocí frameworku *Combine* (viz 5.8).

5.7 SwiftUI ovládací prvky editoru

Ovládání editoru je řešené pomocí tlačítek ve spodní části obrazovky[3]. Tato tlačítka jsou implementovaná – stejně jako zbytek UI aplikace – ve frameworku *SwiftUI*. V následující sekci je popsáno, jak je uživatelova interakce s těmito tlačítky přeposlána do druhé části editoru.

5.8 Komunikace SwiftUI s ARView pomocí Combine

Pro odeslání informací o stisku *SwiftUI* tlačítek byl využit framework *Combine*, který umožňuje vývojářům deklarativně definovat manipulaci s událostmi a datovými toky v aplikaci. Informace jsou odesílány pomocí *singleton* třídy *ARActionManager*, která umožňuje publikovat hodnoty typu *ARAction*, které si následně odběratelé toku zpracují. V následujícím výpisu kódu (9) je ukázána implementace třídy *ARActionManager*.

```
1 class ARActionManager {
2     static let shared = ARActionManager()
3     private init() {}
4
5     var actionStream = PassthroughSubject<ARAction, Never>()
6 }
```

Výpis kódu 9 Implementace třídy *ARActionManager*

5.8.1 Publikace a odběr zpráv v Combine

V následujícím výpisu kódu (10) je ukázáno, jak *SwiftUI* tlačítka využívají *ARActionManager* pro publikaci informací o stisku tlačítka.
Příjem těchto zpráv probíhá v controlleru *ImageTrackingARViewController*, který řídí *ARView*[4]. V následujícím výpisu kódu 11 je ukázáno, jak je implementován odběr publikovaných zpráv. Aby byl odběr zpráv inicializován, je nutné funkci *subscribeToActionStream()* spustit při inicializaci třídy *ImageTrackingARViewController*. Po příjmu akce stisku tlačítka se následně spustí odpovídající funkce controlleru. Kromě zmíněné funkce je v ukázce kódu proměnná *cancellables*, která zajišťuje, že odběr zpráv bude probíhat po celou dobu délky života instance třídy controlleru.

[3] Kromě těchto tlačítek je v editoru podpora gest, která jsou implementována přímo v AR části implementace editoru (5.9).
[4] *ImageTrackingARViewController* je podrobněji popsán v sekci 5.9.5.

```
// SwiftUI implementace tlačítka pro mazání aktivního bodu, které publikuje odpovídající akci.
// Tento kód se nachází v ARViewActionButtons SwiftUI View.
ARActionButton(
    action: {ARActionManager.shared.actionStream.send(.deleteLastPoint)},
    icon: "trash.square"
)
```

Výpis kódu 10 Implementace publikace zpráv v *Combine* frameworku

```
// Implementace odběru publikovaných zpráv ve třídě ImageTrackingARViewController.
private var cancellables: Set<AnyCancellable> = []

private func subscribeToActionStream() {
    ARActionManager.shared
        .actionStream
        .sink { [weak self] action in
            switch action {
            case .placePoint(let linkType):
                self?.addPoint(linkType: linkType)
            case .deleteLastPoint:
                self?.deleteSelectedPoint()
            case .createSnapshot:
                self?.createScreenSnapshot(
                    saveToPhotos: self?.viewModel.saveScreenshotToPhotos ?? false
                )
            case .setSnapping(let snappingOn):
                self?.setPointSnapping(snappingOn: snappingOn)
            }
        }
        .store(in: &cancellables)
}
```

Výpis kódu 11 Implementace odběru zpráv v *Combine* frameworku

5.8.2 ARAction enum

ARAction je *enum* (výčtový typ), jehož hodnoty reprezentují stisknutí jednotlivých tlačítek. V následujícím výpisu kódu (12) je zobrazena jeho implementace. Za zmínku stojí poukázat na hodnoty *placePoint* a *setSnapping*, které na sobě nesou další hodnotu. U možnosti *placePoint* se jedná o typ rozvodu, který uživatel vytváří a u možnosti *setSnapping* se jedná o informaci, zda je funkce pro napojení na existující body (viz 4.6.9.4) zapnutá.

```
enum ARAction {
    case placePoint(linkType: LinkType)
    case deleteLastPoint
    case createSnapshot
    case setSnapping(snappingOn: Bool)
}
```

Výpis kódu 12 Implementace *ARAction enum*

5.9 Implementace ARView

5.9.1 Úvod do funkčnosti editoru

Než bude popsána implementace editoru rozvodů, je nezbytné pochopit základní principy jeho fungování.

AR editor rozvodů využívá *ARKit* k detekci referenčního obrázku a k detekci roviny zdi za obrázkem. Referenční obrázek slouží jako počátek lokálního souřadnicového systému pro vykreslování bodů diagramu a zároveň umožňuje vykreslit dříve zaznamenaný rozvod na stejném místě, pokud je obrázek položen na stejnou pozici. Kromě detekce obrázku je nutné detekovat i rovinu zdi za ním, protože v praxi se ukázalo, že rovina detekovaného obrázku není stoprocentně rovnoběžná s rovinou zdi. To by mělo za následek, že diagram rozvodů by odstával od zdi – čím dále od referenčního obrázku, tím více.

Proto finální implementace editoru kombinuje detekci referenčního obrázku s detekcí roviny zdi. Rotace lokálního počátku (vůči globálnímu počátku) je použita z roviny zdi, protože je přesnější, a zarovnání os X a Z je získáno promítnutím lokace referenčního obrázku na rovinu zdi. Jelikož diagram rozvodů je pouze ve 2D a leží souběžně s rovinou zdi, je hodnota Y všech bodů na diagramu rovna nule.

Výsledná globální poloha a rotace lokálního počátku je uložena v transformační matici entity *referenceAnchorEntity*. Tato informace je klíčová pro správné vykreslování a manipulaci s diagramem rozvodů v rozšířené realitě.

5.9.2 Struktura souborů editoru

Implementace AR části editoru má následující strukturu souborů (jednotlivé soubory jsou podrobně rozepsány v následujících podkapitolách):

- `IT_ARView.swift` ..... vykresluje AR scénu na obrazovku
- `IT_ARView+TapHandler.swift` ..... implementuje detekci gest
- `IT_ARViewController.swift` ..... řídí životní cyklus ARView
- `IT_ARViewController+ARSessionDelegate.swift` ..... řídí správu detekovaných kotev
- `IT_ARViewControllerRepresentable.swift` ..... SwiftUI wrapper obalující UIKit ARView
- `IT_ViewModel.swift` ..... prezentační vrstva ARView
- `IT_ViewModeling.swift` ..... protokol definující rozhraní ViewModelu

5.9.3 IT_ARView

Třída *ImageTrackingARView* je potomek třídy *ARView*, která slouží pro zobrazení AR obsahu v aplikacích. *ARView* využívá *ARKit* ve spojení s frameworkem *RealityKit*. *ImageTrackingARView* rozšiřuje základní funkcionalitu třídy *ARView* o dodatečné metody pro vykreslení vlastního obsahu a interakci s ním. Tyto metody, které zahrnují konfiguraci nastavení pro sledování okolí, vykreslování bodů, spojení mezi body a ikon spojení, jsou popsány v následujícím textu.

init() – nastavení konfigurace

Ve funkci *init* jsou nastaveny parametry konfigurace *ARWorldTrackingConfiguration*, která je v aplikaci použita pro analýzu okolí (viz kapitola 3.5.7). Níže je uveden přehled věcí, které jsou při inicializaci *ImageTrackingARView* nastaveny:

- nastavení *ARView*, aby použilo konfiguraci *ARWorldTrackingConfiguration*
- nastavení konfigurace, aby detekovala vertikální a horizontální roviny
- nastavení konfigurace, aby ve scéně detekovala referenční obrázek
- inicializace pomocné info-grafiky *ARCoachingOverlayView*, která uživatele instruuje ohledně toho, jak mají pohybovat zařízením tak, aby bylo zařízení schopno detekovat okolní zdi

renderPoint()

Tato funkce vytvoří *RealityKit* entitu bodu rozvodu, která je následně přidána do virtuální 3D scény. Jelikož jednotlivé body rozvodů mají uloženou svou polohu relativně k pozici referenčnímu obrázku, je entita bodu přidána jako potomek entity *referenceAnchorEntity*. Vytvořená entita tak využívá lokální souřadnicový systém, který má počátek v centru detekovaného referenčního obrázku (viz popis v kapitole 5.9.1).

Funkce na vstupu přijímá instanci *Repository* modelu *Point*, ze kterého získá souřadnice bodu. Do instance *Point* je následně uložena vytvořená *RealityKit* entita, aby ji následně šlo upravit nebo smazat v průběhu běhu editoru.

Vykreslené body v diagramu rozvodů jsou znázorněny na snímcích obrazovky aplikace 5.2.

renderLink()

Tato funkce vytvoří *RealityKit* entitu propojení *PointLink* mezi dvěma body rozvodu. Aby byly v editoru od sebe odlišitelné jednotlivé typy rozvodů, je pro každý typ rozvodu vytvořené propojení s odlišným materiálem. Informace o krajních bodech propojení a o typu propojení jsou do funkce předány pomocí instance *Repository* modelu *PointLink*. Vytvořená entita je následně do instance modelu *PointLink* uložena pro další použití.

renderLinkIcon()

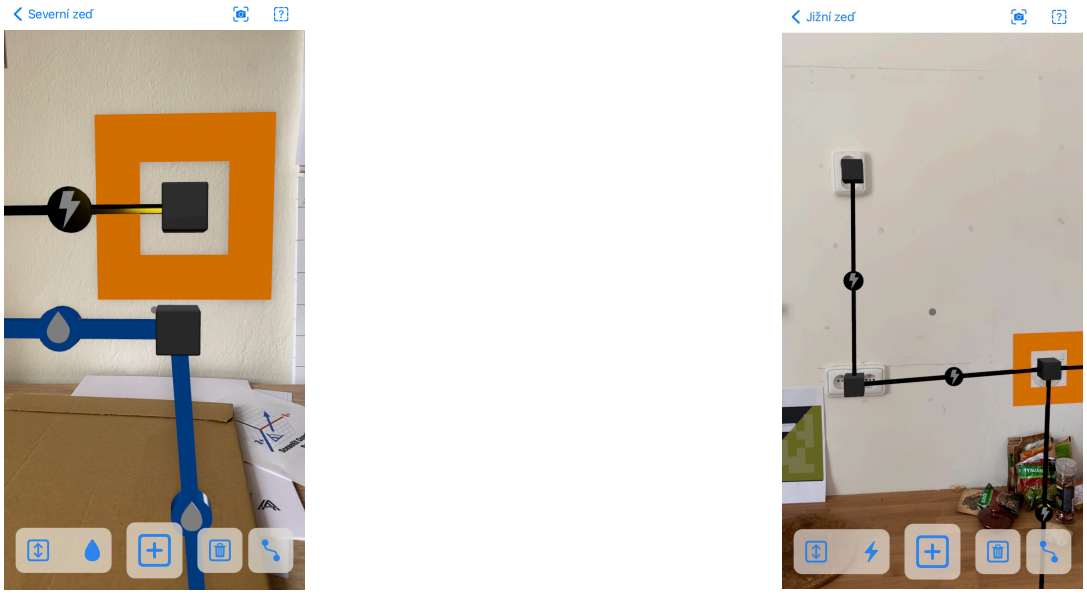
Tato funkce vytvoří uprostřed propojení – *PointLink* – *RealityKit* entitu, která tvoří dodatečný způsob identifikace typu rozvodu. Tato entita je složena z kruhu a 3D modelu, který reprezentuje typ média vedeného uvnitř rozvodu. Vytvořená *RealityKit* entita je následně uložena do *Repository* modelu *PointLink*.

Propojení dvou bodů rozvodu včetně znázornění rozdílů jednotlivých typů je zobrazené na snímku obrazovky 5.2.

renderActivePointMarker()

Tato funkce vytvoří *RealityKit* entitu, jež v AR editoru následně funguje jako indikátor právě aktivního bodu. Když se v editoru změní aktivní bod, je indikátor přesunut na polohu daného bodu. Ve chvíli, kdy není žádný bod aktivní, je indikátor přesunut mimo scénu, tak aby nebyl vidět.

Indikátor aktivního bodu je znázorněn na snímcích obrazovky 5.2.



Obrázek 5.2 Snímky obrazovky vykresleného 3D obsahu

5.9.4 IT_ARView+TapHandler

Toto rozšíření třídy *ImageTrackingARView* implementuje detekci gest, pomocí kterých může uživatel interagovat s AR obsahem. AR obrazovka podporuje následující dvě gesta:

- gesto kliknutí
  - ve chvíli, kdy uživatel klikne na obrazovku, je zavolána funkce *handleTap*
  - v této funkci je získána poloha bodu na obrazovce, kam uživatel klikl
  - získaná poloha je následně použita v *ImageTrackingARViewController*, kde se určí, zda uživatel vybral bod, nebo klikl mimo bod
- gesto táhnutí
  - ve chvíli, kdy začne uživatel táhnout prstem po obrazovce, je zavolána funkce *handlePan*
  - v *ImageTrackingARViewController* jsou následně data o tahu prstu zpracována a použita pro posun zaznamenaných bodů

5.9.5 IT_ARViewController

Třída *ImageTrackingARViewController* (dále jen *controller*) řídí životní cyklus *ImageTrackingARView*. Hlavním úkolem *controlleru* je reagovat na akce uživatele přijaté skrze *ARActionManager stream* (viz 5.8), načítat uložený obsah, implementovat uživatelská gesta a zobrazovat pomocné hlášky pro uživatele. V následujících sekcích jsou popsané důležité funkce *controlleru*.

subscribeToActionStream()

Tato funkce inicializuje odběr zpráv, které jsou odesílány skrze *Combine stream* ze *SwiftUI* UI (viz 5.8). Funkce zajišťuje, že při doručení *ARAction* akce je zavolána správná funkce v *controlleru*, která zpracuje uživatelův požadavek.

getTapRealPosition()

Tato funkce slouží pro získání polohy v lokálním souřadnicovém systému ze zaslané souřadnice na obrazovce zařízení. Toho je dosaženo pomocí *raycast* metody, která vytvoří polopřímku směrem z čočky kamery pod úhlem, který je stanoven polohou bodu na obrazovce. Výsledná hodnota funkce *getTapRealPosition* je souřadnice, kde *raycast* přímka protne rovinu zdi (pokud přímka rovinu neprotne je vrácena hodnota *nil*).

addPoint()

Funkce *addPoint* je zavolána, když uživatel klikne na tlačítko „+“. V daný okamžik je zavolána funkce *getTapRealPosition*, pomocí které je získána souřadnice bodu, na který ukazuje střed obrazovky (tam, kde je pomocný indikátor).

Na získané souřadnici je vytvořen nový bod diagramu (*Repository* model *Point*). Tento bod je následně vykreslen na obrazovku pomocí funkce *renderPoint* (viz 5.9.3). Pokud je vsak aktivní funkce pro znovu-použití existujících bodů a pod středem obrazovky již leží existující bod, nový bod se nevytvoří.

Pokud je při zavolání funkce *addPoint* nějaký bod označený jako aktivní, vytvoří se mezi aktivním bodem a nově přidaným bodem propojení. Pro toto propojení se vytvoří instance *Repository* modelu *PointLink* a propojení je následně vykresleno na obrazovku pomocí funkce *renderLink* (viz 5.9.3).

Nakonec je nový bod označen jako aktivní.

deleteSelectedPoint()

Tato funkce smaže aktivní bod. Při této akci je smazána instance *Repository* modelu *Point*. Při dealokaci bodu je i smazána vizualizace bodu z obrazovky (k tomu je využita reference na *RealityKit* entitu, kterou si *Point* v sobě drží).

Pokud je bod propojen s dalšími body diagramu, jsou všechna tato propojení – *PointLink* – taktéž smazána.

Pokud byl aktivní bod propojený právě s jedním bodem, je tento bod nastaven jako aktivní.

createScreenSnapshot()

Funkce *createScreenSnapshot* je zavolána, když uživatel klikne na tlačítko s fotoaparátem. Funkce vytvoří snímek AR scény pomocí vestavěné metody *snapshot* na instanci *ARView*. Jelikož je použita vestavěná metoda frameworku, je na výsledném snímku pouze AR scéna (na snímku není vidět *SwiftUI* UI překrytí).

Zachycený snímek AR scény je následně uložen do galerie v detailu zdi. Na základě zpětné vazby uživatelského testování bylo do aplikace přidáno nastavení, kterým lze nastavit, zda se snímek uloží kromě galerie zdi i do systémové aplikace *Fotky*.

renderSavedData()

Funkce *renderSavedData* je spuštěna, když je ve scéně detekován referenční obrázek a zeď za ním. Funkce následně vykreslí na obrazovku všechny dříve uložené body diagramu a propojení mezi nimi.

raycastActivePoint()

Funkce *raycastActivePoint* přijímá na vstupu souřadnici bodu, kde uživatel klikl na obrazovku. Pomocí *raycast* funkce je otestováno, zda byla pod místem dotyku entita nějakého bodu – pokud ano, je z vrácena instance *Repository* modelu *Point*, v opačném případě je navrácena hodnota *nil*.

selectPoint()

Funkce *selectPoint* je zavolána funkcí *handleTap*, poté co je detekováno, že uživatel klikl na obrazovku. Pomocí funkce *raycastActivePoint* je následně zjištěno, pokud byl pod bodem dotyku nějaký bod. Pokud ano, je tento bod označený jako aktivní, pokud ne, je zrušeno označení aktivního bodu.

movePoint()

Funkce *movePoint* je volána funkcí *handlePan*, když probíhá táhnutí prstem přes obrazovku – pokud tah prstu započal v nějakém bodě diagramu rozvodů, je tento bod schématu posunut. Jelikož však táhnutí prstu není jednorázová akce, je funkce *movePoint* volána opakovaně po celou dobu tahu prstu.

Kromě informace o aktuální poloze prstu na obrazovce je také funkcí *handlePan* poskytována hodnota, v jaké fázi je tah prstu po obrazovce. Přijít mohou následující tři hodnoty – *.began*, *.changed* a *.ended*. V závislosti na dané fázi jsou provedeny ve funkci *movePoint* následující akce:

- *.began*
  - Při započetí detekce tahu je pomocí funkce *raycastActivePoint* zjištěno, zda tah prstem začal z nějakého bodu diagramu rozvodů.
  - Pokud ano – *Point* je uložen pro použití v další fázi.
  - Pokud ne – další fáze se neprovedou (nebyl detekován bod, co by mohl být posunut).
- *.changed*
  - Vykreslené entity bodu a navázaných propojení jsou překresleny na novou pozici.
- *.ended*
  - Při zvednutí prstu je uložena nová pozice bodu.

updatePointEntities()

Funkce *updatePointEntities* je volána v průběhu posunu bodu. Funkce překreslí *RealityKit* entity posunutého bodu *Point* – včetně entit navázaných propojení *PointLink*. Překreslení entit je optimalizováno – většina entit bodu a navázaných spojení mění pouze svou lokaci v prostoru, pouze entitu samotného propojení je nutné inicializovat znovu, jelikož posunem bodu se mění délka jednotlivých spojení a nestačí tak pouze změnit souřadnice této entity.

displayMessage()

Funkce *displayMessage* slouží pro zobrazení textových informačních zpráv uživatelovi. Tato funkce byla implementována dodatečně na základě analýzy průběhu uživatelských testů – během testů uživatelům chyběla zpětná vazba ohledně chování editoru rozvodů (např. uživatelé neměli informaci o tom, že zeď za obrázkem zatím nebyla detekována)

Pomocí této funkce, tak lze uživatelovi předat zprávu, která se mu následně zobrazí formou *pop-up* zprávy v horní části obrazovky. Jelikož jsou zprávy zobrazovány ve *SwiftUI* části editoru rozvodů, jsou zprávy odesílány pomocí *PassthroughSubject* frameworku *Combine*. Princip těchto zpráv je stejný, jako výše popsaný způsob 5.8, který předává informace v opačném směru. Neposílá se však hodnota typu *ARAction*, ale pouze hodnota typu *String*, která obsahuje text *pop-up* zprávy.

Funkci *displayMessage* lze předat dodatečný argument *displayTime*, kterým lze určit, po jaké době bude *pop-up* zpráva schována. Tato možnost je vhodná pro zprávy, které informují o tom, že proběhla nějaká událost (např. byl detekován referenční obrázek). Pokud hodnota zůstane nevyplněna, *pop-up* zpráva zmizí pouze v případě, kdy na ni uživatel klikne. Zprávy, které samy nezmizí, jsou například pomocné zprávy s návodem, co má uživatel udělat, aby byla detekována rovina zdi.

Smazání zprávy po časové prodlevě je implementované pomocí asynchronního *tasku*. Na tento *task* je, stále držena reference ve *ViewModelu*, aby jej bylo možné zrušit v případě, když je funkce *displayMessage* zavolána před jeho dokončením.

5.9.6 IT_ARViewController+ARSessionDelegate

V souboru *IT_ARViewController+ARSessionDelegate* je *controller* rozšířen o implementaci protokolu *ARSessionDelegate*. Tento protokol obsahuje specifikaci funkcí, které reagují na detekci a aktualizaci *ARKit* kotev ve scéně. V následujících sekcích jsou podrobně popsány jednotlivé implementace těchto funkcí.

5.9.6.1 getDetectedPlaneAnchor(...)

Funkce *getDetectedPlaneAnchor* slouží pro výběr vhodné roviny zdi, za referenčním obrázkem. Jelikož funkce 5.9.6.3 vrací všechny detekované kotvy ve scéně – nezávisle na tom, zda se jedná o kotvu rovinu zdi za obrázkem, kotvu roviny vedlejší zdi, či kotvu detekovaného obrázku – je nutné z těchto kotev vybrat pouze ty, které reprezentují detekované roviny a následně z nich vybrat kotvu, která reprezentuje rovinu zdi, na které leží referenční obrázek.

Implementace výběru vhodné roviny je ukázána ve výpisu kódu 13. Při výběru ideální kotvy roviny, která leží za detekovaným obrázkem se postupuje následujícími kroky:

1. Ze seznamu všech detekovaných kotev se vyfiltrují kotvy, které reprezentují rovinu – tedy jsou typu *ARPlaneAnchor* (řádek 5).

2. Je spočítán normálový vektor roviny detekovaného referenčního obrázku (řádek 10).

3. Ze seznamu kotev rovin jsou vyfiltrovány pouze ty, jejichž normálový vektor svírá s normálovým vektorem detekovaného obrázku úhel menší než je limit stanovený v konstantě *ANGLE_TRESHOLD* (řádek 16–21).

4. Ze zbývajících kotev rovin je vybrána rovina, ke které leží referenční obrázek nejblíže (řádek 22–26).

Bod 4 byl implementován na základě výsledků uživatelského testování. Během testů prováděných v kuchyni byla kromě roviny zdi detekována také rovina přední stěny kuchyňské linky – editor rozvodů tak zobrazil diagram rozvodů půl metru před zdí. Řazení zdí dle minimální vzdálenosti od referenčního obrázku tento problém vyřešilo.

```
// Maximum angle which is accepted as similiar to the image tracker.
let ANGLE_TRESHOLD: Float = 0.5

// Filter just anchors, which are ARPlaneAnchor type.
let planeAnchors = detectedAnchors.filter({ $0 is ARPlaneAnchor }) as! [ARPlaneAnchor]

// Filter anchors, which normal angle to reference image normal is in suitable error range.
if let imageTransform = imageTransform {

    let imageNormalVector = getTransformNormalVector(imageTransform)

    // First planes are filtered by their angle
    // (Only planes facing the direction as the ref. image are accepted.)
    // After that the plane, which is the closest to the point, is selected.
    return planeAnchors
        .filter({
            angleBetweenVectors(
                imageNormalVector,
                getTransformNormalVector(Transform(matrix: $0.transform))
            ) < ANGLE_TRESHOLD
        })
        .sorted(by: { first, second in
            let firstDistance = first.distanceFromPoint(point: imageTransform.translation)
            let secondDistance = second.distanceFromPoint(point: imageTransform.translation)
            return firstDistance < secondDistance
        })
```

Výpis kódu 13 Výběr kotvy roviny zdi za detekovaným referenčním obrázkem

5.9.6.2 session(_ session: ARSession, didAdd anchors: [ARAnchor])

Tato funkce je zavolána pokaždé, když je ve scéně detekována nová kotva. V implementaci aplikace *WallInfo* se ve funkci kontroluje, zda byl detekován referenční obrázek. V moment, kdy je referenční obrázek detekován, se odehrají následující akce:

1. Obrázek je překryt obdélníkem stejného tvaru, uživatel je tak informován, že detekce obrázku probíhá.
2. Je vytvořena referenční entita *referenceAnchorEntity* pomocí transformační matice detekované kotvy obrázku, jejíž lokální souřadnicový systém je následně použitý pro vizualizaci diagramu.

3. Je vytvořena neviditelná rovina o velikosti 20 krát 20 metrů, která je následně použita při tvorbě nových bodů pomocí *raycast* metody. (Tato rovina je umístěna jako potomek entity *referenceAnchorEntity* – poté, co je upřesněna rotace *referenceAnchorEntity* pomocí dat detekované roviny zdi, je upřesněna i tato rovina.)

4. Uživatelovi je zobrazen *pop-up* s informací, že obrázek byl detekován.

5. Je vytvořen automaticky snímek obrazovky (tato funkčnost byla přidána na základě uživatelského testování, kde si uživatelé neuložili místo, kam dali referenční obrázek, ačkoliv k tomu byli vyzváni skrze textové upozornění). Tato funkčnost lze vypnout v nastavení aplikace.

6. Je spuštěn asynchronní odpočet 15 sekund. Následně je otestováno, zda již byla detekována rovina zdi za obrázkem – pokud ne, je zobrazena uživateli *pop-up* zpráva napovídající, jak má postupovat.

5.9.6.3 session(_ session: ARSession, didUpdate anchors: [ARAnchor])

Tato funkce je zavolána pokaždé, když jsou aktualizovány již detekované kotvy. Implementace funkce v této aplikaci provádí následující kroky:

1. Nejdříve je z detekovaných kotev vybrána kotva zdi za detekovaným obrázkem pomocí funkce *getDetectedPlaneAnchor*. Pokud referenční obrázek nebyl detekován nebo nebyla detekována rovina zdi za obrázkem – následující kroky se neprovedou.

2. Pokud se jedná o první detekci roviny za obrázkem, jsou provedeny následující kroky:

   a. Je zobrazena *pop-up* zpráva, že byla detekována rovina zdi a nyní je možné začít používat editor rozvodů.

   b. Pokud zeď obsahuje již vytvořený záznam diagramu, zavolá se funkce pro vykreslení dříve uložených dat – *renderSavedData*.

3. Transformační matice detekovaného obrázku je zkombinována s transformační maticí detekované roviny zdi. Výsledkem je transformační matice, která má stejnou rotaci jako rovina zdi, a jejíž poloha je nastavena na bod ležící na rovině zdi, který je nejblíže detekovanému obrázku. Tato matice je následně nastavena referenční entitě – *referenceAnchorEntity*.

5.9.6.4 session(_ session: ARSession, didUpdate frame: ARFrame)

Tato funkce je volána během každého vykreslení obrazovky. V aplikaci je tato funkce použita pro detekci zaznamenaných bodů, na které uživatel míří. Toho je využito v moment, kdy uživatel chce vytvořit propojení na již existující bod diagramu rozvodů – pokud je funkce pro použití existujících bodů aktivní a uživatel při stisknutí tlačítka „+“ míří na již vytvořený bod, je tento bod použit namísto vytvoření nového.

Editor rozvodů si s pomocí této funkce udržuje v paměti, zda právě míří středem obrazovky na existující bod, či nikoliv.

Detekce toho, zda uživatel míří na bod, je implementována pomocí funkce *raycast*, která v tomto případě testuje, zda paprsek, který pomyslně míří ze středu obrazovky směrem do prostoru před zařízením, protíná existující bod diagramu.

5.9.7 IT_ARViewControllerRepresentable

Jelikož je *ARView* typu *UIKit UIView*, je nutné ho obalit pomocí *SwiftUI wrapperu UIViewRepresentable* tak, aby šlo následně použít ve *SwiftUI* kontextu. V kontextu aplikace *WallInfo* je *ARView* zastupováno svým *controllerem*, proto je zde použit *UIViewControllerRepresentable*, který umožňuje vložit *UIViewController* do *SwiftUI* kontextu.

5.9.8 IT_ViewModel

Třída *ImageTrackingViewModel* slouží jako prezentační vrstva, která zpracovává data a zajišťuje komunikaci s dalšími rozhraními, která jsou potřeba pro funkčnost samotného *ARView* a jeho *controlleru*. Atributy a funkce *ImageTrackingViewModelu* jsou specifikovány v protokolu *ImageTrackingViewModeling*, který tento *ViewModel* implementuje.

Kapitola 6

Testování

V této kapitole je popsáno, proč byl prototyp aplikace testován pomocí uživatelského testování použitelnosti, jak testování probíhalo a jaké přineslo výsledky. Na konci kapitoly jsou uvedeny postřehy z testování, které byly aplikovány do finální verze aplikace.

Hlavní funkcí aplikace je tvorba a vizualizace diagramů rozvodů v prostředí rozšířené reality. Ačkoli použitý framework pro rozšířenou realitu – *ARKit* – umožňuje načíst předem nahraný pohyb zařízení a *stream* obrazu z kamery (*AR Session*) [50], jedná se spíše o nástroj pro usnadnění vývoje aplikace v prostředí, které neumožňuje zkoušet funkčnost aplikace (např. vývoj hry s AR prvky, pro jejíž hraní je nutný velký venkovní prostor). Jelikož je funkčnost aplikace při integraci AR vysoce ovlivněná okolními jevy (např. světelné podmínky, rychlost pohybu uživatele, zašpinění kamery, povrch stěny místnosti, …), není fyzicky možné vytvořit sadu dat pro automatické testy tak, aby byly otestovány všechny možné situace, co mohou nastat.

Pro otestování vytvořeného prototypu aplikace byly zvoleny uživatelské testy použitelnosti, které primárně testují použitelnost aplikace. Zde však uživatelské testy použitelnosti zastávají i sekundární roli, a to test AR funkcí aplikace za odlišných okolních jevů, které nelze důkladně otestovat pomocí automatizovaných testů.

6.1 Zásady uživatelského testování použitelnosti

Všechny testy použitelnosti by dle [51] měly splňovat následujících pět bodů:

1. Primární cíl je vylepšení použitelnosti produktu. U každého testu máte také konkrétnější cíle a obavy, které formulujete při plánování testu.
2. Účastníci testu reprezentují reálné uživatele.
3. Účastníci testu dělají reálné úlohy.
4. Během testu je pozorováno a nahráváno, co účastníci dělají a říkají.
5. Následně jsou data analyzována, diagnostikují se reálné problémy a jsou navrženy změny pro opravu identifikovaných problémů.

6.2 Scénáře uživatelského testování

V následujících sekcích jsou popsány jednotlivé scénáře uživatelského testování použitelnosti, které byly navrženy tak, aby pokryly navržené případy užití (viz kapitola 4.5) a simulovaly reálné procesy a situace, jež mohou nastat při používání aplikace. Cílem těchto scénářů je poskytnout uživatelům dostatečný prostor pro interakci s aplikací podle vlastního uvážení a zároveň zohlednit možné situace, které by mohly ovlivnit jejich zkušenosti.

Při vytváření scénářů byl kladen důraz na jejich volnost a flexibilitu, aby účastníci testu mohli postupovat v souladu s pokyny a nápovědou, kterou jim poskytuje samotná aplikace. Tímto způsobem bylo možné sledovat, jak si uživatelé poradí s interakcí s aplikací v reálném kontextu a jak intuitivně je aplikace schopna uživatele vést při provádění úkolů.

Instrukce pro moderátora a účastníka jednotlivých testovacích scénářů jsou v příloze C.

Scénář 1 – Nastavení vlastního kotvícího obrázku

Tento testovací scénář sleduje chování uživatele při prvním spuštění aplikace a následném nahrávání vlastního obrázku sloužícího k ukotvení scény v rozšířené realitě. Cílem tohoto testu bude ověřit, zda jsou pomocné popisky v aplikaci dostatečně navádějící. V ideálním průběhu scénáře si uživatel zvolí nejvíce kontrastní obrázek z poskytnuté nabídky, nahraje jej do aplikace a nastaví jeho přesnou šířku.

Scénář 2 – Nastavení výchozího kotvícího obrázku

Účelem tohoto scénáře je sledovat chování uživatele při prvním spuštění aplikace a volbě kotvícího obrázku, který je v aplikaci přednastavený.

V ideálním průběhu scénáře uživatel zvládne vytisknout poskytnutý obrázek pomocí systémového dialogu, sloužícího ke sdílení fotek, následně do aplikace zadat jeho šířku a potvrdit volbu tohoto nastavení přesunem na následující obrazovku.

Součástí testu je tisk obrázku skrze prostředí aplikace – pokud však v místnosti, kde probíhá test, není přístup k tiskárně, která podporuje technologii *AirPrint*, je účastníkovi testu předán již vytištěný obrázek ve chvíli, kdy se v aplikaci dostane do fáze vyhledávání dostupných tiskáren v síti.

Scénář 3 – Záznam polohy rozvodů

Tento test sleduje chování uživatele při vyplňování dat o místnosti a zdi a následně při tvorbě diagramu rozvodů v prostředí AR.

V ideálním průběhu scénáře uživatel v aplikaci vytvoří místnost, ve které zaznamená novou zeď. Následně spustí AR editor rozvodů a dle zobrazených instrukcí upevní kotvící obrázek na zeď. Polohu kotvícího obrázku uživatel uloží do detailu zdi pomocí poznámky nebo formou fotografie. Při tvorbě diagramu rozvodů uživatel správně zvolí typ rozvodu a zaznamená polohu rozvodů, tak aby odpovídala reálné situaci.

Z praktických důvodů jsou rozvody předem na zdi vyznačené pomocí malířské pásky (či jiným způsobem, který zeď neponičí).

Scénář 4 – Lokalizace rozvodů z již existujícího záznamu v aplikaci

Tento test sleduje chování uživatele při lokalizaci rozvodů ve zdi z již vytvořeného diagramu v aplikaci. Test simuluje situaci, kdy uživatel aplikace plánuje provádět stavební úpravy se zdí a potřebuje nalézt místa, kudy vedou rozvody, které dříve v aplikaci zaznamenal.

Při ideálním průběhu testu uživatel po zapnutí aplikace nalezne údaje o zadané zdi, s jejichž pomocí upevní kotvící obrázek na správné místo. Následně zapne AR editor rozvodů a dle zobrazené vizualizace malířskou páskou na zeď vyznačí umístění rozvodů.

Scénář 5 – Editace existujícího diagramu

V tomto testu jsou testovány pokročilejší funkce AR editoru rozvodů – a to mazání existujících rozvodů, posun špatně zaznamenaných rozvodů a tvorba fotografií vizualizace rozvodů.

Ideální průběh testu vypadá následovně: Účastník testu se v aplikaci dostane na obrazovku s informacemi o zadané zdi a umístí kotvící obrázek na místo určené v popisku. Následně zapne AR editor rozvodů a po načtení AR obsahu vytvoří snímek scény. Poté správně určí, jaký rozvod vede vodu a ten smaže. Po smazání rozvodu vody upraví lokaci rozvodu elektřiny tak, aby vedl po hraně podlahy a zdi. V poslední části testu účastník vyfotí upravenou scénu, vrátí se na předchozí obrazovku, otevře náhled zachycených snímků a přes kontextové menu *Sdílet* uloží obrázky do galerie telefonu.

6.3 Výběr účastníků testu

Účastníci testu by dle výše zmíněných zásad testování použitelnosti 6.1 měli představovat cílovou skupinu aplikace. Jelikož cílová skupina aplikace nelze specifikovat na úzkou množinu uživatelů, byli pro testování aplikace vybráni lidé různého věku, pohlaví a s rozdílnými zkušenostmi s novými technologiemi tak, aby byla zastoupena, co největší část cílové skupiny aplikace.

6.3.1 Popis jednotlivých účastníků

Testování se zúčastnili následující účastníci:

- Účastník A
  - Věk 18
  - Telefon *iPhone 12 mini*
- Účastník B
  - Věk 58
  - Uživatel *OS Android*
- Účastník C
  - Věk 53
  - Uživatel *OS Android*
- Účastník D
  - Věk 24
  - Bývalý uživatel *iOS*
- Účastník E
  - Věk 23
  - Telefon *iPhone 8 plus*
- Účastník F
  - Věk 24
  - Telefon *iPhone 12*

6.4 Průběh uživatelského testování

Před zahájením uživatelského testování byl každému účastníkovi předán formulář (viz příloha D), kterým byl potvrzen souhlas s natáčením videa a obrazovky aplikace pro účely zpětné analýzy průběhu testování. Testování probíhalo v místnosti, kde na úvod moderátor stručně představil aplikaci, vysvětlil její funkce a účel podobně, jak by to bylo uvedeno v popisku aplikace v obchodě s aplikacemi – *AppStore*. Účastníkům testu, kteří nevlastnili *iOS* zařízení, a účastníkům, kteří nechtěli na své zařízení nahrát prototyp aplikace, byl moderátorem poskytnut telefon *iPhone 11* s nainstalovanou aplikací.

Po úvodním seznámení s aplikací následovalo testování jednotlivých scénářů. Během testování moderátor minimalizoval zásahy a intervenoval pouze v případě, že účastník zůstával delší dobu zaseknutý na jednom bodě. Po ukončení posledního scénáře proběhl rozhovor mezi moderátorem

a účastníkem, kde byly diskutovány dojmy z používání aplikace, včetně aspektů, které by účastník považoval za nutné změnit.

V průběhu testování, mezi jednotlivými seancemi s účastníky, byly identifikované chyby opravovány. Tento postup umožnil zabránit opakovanému identifikování stejných problémů a současně dal účastníkům příležitost odhalit další chyby. Takto získaná zpětná vazba přispěla k efektivnějšímu zlepšení použitelnosti aplikace a k identifikaci potenciálních problémů, které by mohly uživatele ovlivnit.

Pro získání co největšího množství poznatků byly jednotlivé skupiny testovány v odlišných prostorech tak, aby byla v testech zahrnuta i variabilita prostředí.

6.5 Výsledky uživatelského testování

V následujících sekcích jsou popsány poznatky jednotlivých skupin uživatelských testů, mezi kterými byly průběžně opravovány identifikované chyby. V závěrečné sekci je sepsáno celkové shrnutí uživatelských testů.

Jednotlivé skupiny byly složené z následujících účastníků:

- Skupina 1 – účastník A
- Skupina 2 – účastníci B a C
- Skupina 3 – účastník D
- Skupina 4 – účastníci E a F

6.5.1 1. skupina účastníků

Během testování 1. skupiny účastníků byly odhaleny následující problémy:

1. V aplikaci nefunguje haptická odezva, když je nahráván záznam obrazovky současně se zvukem z mikrofonu.
2. Uživatel nemá informaci, že referenční obrázek a zeď za obrázkem nebyla ještě detekována – uživatel aplikace má takto dojem, že AR editor rozvodů nefunguje, jelikož nefungují tlačítka na přidání nových bodů.
3. Ikony uprostřed jednotlivých vizualizací rozvodů jsou příliš malé, aby je šlo použít k identifikaci typu rozvodu.
4. Při testování aplikace v kuchyni byla detekována rovina přední stěny linky místo roviny samotné zdi za linkou – AR diagram rozvodů tak byl umístěný přibližně půl metru od předpokládané pozice.

Tyto problémy byly opraveny pomocí následujících změn:

1. Po zbytek testování byl vypnut záznam zvuku při nahrávání obrazovky zařízení. Pro potřeby analýzy záznamu testů byl použit záznam zvuku z externí kamery, která zabírala celou místnost s probíhajícím testem.

2. Do AR editoru byly přidány informační *pop-up* zprávy, které uživatelovi zobrazí upozornění, že referenční obrázek a zeď nebyla zatím detekována. (*Pop-up* s informací o detekci obrázku je zobrazen 15 vteřin po zapnutí AR editoru rozvodů, pokud obrázek stále nebyl detekován. *Pop-up* s informací o detekci zdi je zobrazen 15 vteřin po detekci referenčního obrázku, pokud nebyla zeď detekována.)

3. Informační ikonky na jednotlivých propojeních rozvodů byly zvětšeny, tak aby byly lépe viditelné.

4. Funkce pro výběr vhodné roviny zdi 5.9.6.1 byla upravena tak, aby vybrala rovinu, která je natočená stejným směrem, jako rovina referenčního obrázku, a rovinu, od které je vzdálenost středu referenčního obrázku minimální.

6.5.2 2. skupina účastníků

Během testování 2. skupiny účastníků byly odhaleny následující problémy:

1. Uživatelé měli problém ovládat AR editor, přestože si přečetli nápovědu uvnitř aplikace.

2. Ve světlém režimu aplikace jsou *pop-up* informační texty málo kontrastní, nelze je tak proto snadno přečíst.

3. Uživatelé se snaží přidávat nové body rozvodů pomocí klikání na obrazovku místo použití tlačítka „+“.

4. Uživatelům chyběla informace, že obrázek a zeď již byly detekovány a že je možné začít používat AR editor.

5. Rozvody jsou v AR vizualizaci příliš malé, lze na ně těžce kliknout.

V aplikaci byly v reakci na výsledky těchto testů provedeny následující změny:

1. Na základě podnětů od účastníka po konci testu bylo natočeno informační video, ve kterém je stručně vysvětleno, jak se aplikace ovládá.

2. Podbarvení *pop-up* zpráv bylo upraveno tak, aby byly zprávy čitelné, jak ve světlém, tak ve tmavém režimu aplikace.

3. O možnosti přidávat nové body do diagramu pomocí *tapnutí* na dané místo na obrazovce bylo uvažováno již v průběhu návrhu aplikace. Tato funkce by však kolidovala s funkcí pro výběr a zrušení výběru bodu, proto ve finální verzi aplikace zůstala původní verze, ve které se nové body přidávají pomocí tlačítka „+“.

4. Do aplikace byly přidány nové *pop-up* zprávy s informací, že obrázek a zeď byly detekovány.

5. V nastavení aplikace lze změnit škálování AR obsahu. Na základě výše popsaného podnětu bylo zvoleno větší výchozí škálování AR obsahu.

6.5.3 3. skupina účastníků

Během testování 3. skupiny účastníků byly identifikovány následující problémy:

1. U nastavení referenčního obrázku by mělo být spíše tlačítko *Tisknout* namísto tlačítka *Sdílet*.
2. Uživatelé aplikace si neukládají informaci, kam byl položen referenční obrázek.
3. Haptická odezva při vytvoření snímku obrazovky AR editoru není dostatečná (Uživatelé si nebyli jisti, zda byl snímek zachycen).

Na základě těchto podnětů byly provedeny následující změny:

1. Přestože je v nastavení výchozího referenčního obrázku požadováno, aby si uživatel obrázek vytiskl, bylo tam ponecháno tlačítko *Sdílet*, jelikož uživatel nemusí nutně obrázek pouze tisknout, ale může ho například poslat emailem na počítač, ze kterého ho následně vytiskne.
2. Do aplikace bylo přidáno nové nastavení, v jehož výchozím stavu je po detekci referenčního obrázku automaticky zachycen snímek obrazovky, který zajistí, že poloha referenčního obrázku bude uložená pro příští použití aplikace.
3. Problém s nedostatečnou odezvou při zachycení snímku obrazovky byl vyřešen přidáním textové *pop-up* informace, která se po zachycení snímku zobrazí na 2 vteřiny v horní části obrazovky.

6.5.4 4. skupina účastníků

Během testování 4. skupiny účastníků byly identifikovány následující problémy:

1. V seznamu místností a seznamu zdí není jasné, že se má kliknout na položku v seznamu pro zobrazení jejího detailu.
2. Při zadávání vlastního referenčního obrázku nelze nahrát obrázek prostřednictvím kamery.
3. Uživatelé aplikace v nastavení nenastavují velikost obrázku.
4. Galerie fotografií v detailu zdi není příliš přehledná.
5. Snímky obrazovky nebyly uloženy do galerie telefonu.
6. Uživatelé měli problémy s detekcí čistě bílé zdi. Poté, co zeď byla detekována, diagram rozvodů se občas mírně posouval.

Zmíněné problémy byly opraveny následujícími změnami:

1. Uživatelé jsou nyní po vytvoření záznamu místnosti / zdi rovnou přesměrováni na obrazovku jejího detailu. Takto by neměla nastat situace, že nebudou vědět, jak postupovat po zadání názvu místnosti do pole v seznamu.

2. Nastavení vlastního referenčního obrázku neumožňuje volbu nahrání referenčního obrázku pomocí fotografie z kamery, aby uživatelé primárně volili originální digitální zdroje tak, aby byla kvalita referenčních obrázků, co nejvyšší.

3. Ve formulářích nastavení vlastního i výchozího referenčního obrázku je nyní nutné kliknout na vstupní pole šířky obrázku, před tím než je uživatelovi umožněno potvrdit formulář.

4. Galerie fotografií v detailu zdi byla upravena do mřížky a velikost jednotlivých fotografií byla zvětšena.

5. Do aplikace bylo přidáno nové nastavení, které ve výchozí možnosti automaticky uloží snímky obrazovky současně s uložením do galerie v aplikaci i do úložiště fotek v zařízení.

6. Problém s detekcí zdi, která na sobě nemá vizuálně výrazné body, nelze vyřešit změnami v implementaci aplikace, jelikož chyba vychází z funkčnosti frameworku *ARKit*. Tento framework detekuje roviny zdí na základě sledování vizuálně zajímavých bodů. V případě čistě bílé zdi, kde chybí tyto vizuální reference, dochází k omezené přesnosti detekce a následnému mírnému posunutí diagramu rozvodů.

6.5.5 Shrnutí výsledků uživatelského testování

Během procesu uživatelského testování byly identifikovány různé problémy, z nichž mnohé souvisely s neintuitivními prvky aplikace. Po každém testování se provedly potřebné úpravy na základě získaných zpětných vazeb, které zahrnovaly například zlepšení uživatelských interakcí v AR editoru prostřednictvím *pop-up* zpráv, či vytvoření krátkého instruktážního videa pro lepší pochopení funkcí aplikace. Tyto změny byly provedeny s cílem vylepšit uživatelský zážitek a usnadnit ovládání aplikace. Při následných testováních s dalšími skupinami uživatelů se pak již neobjevovaly problémy, které byly identifikovány a opraveny v předchozích fázích testování.

Kromě problémů s neintuitivním UI měli někteří účastníci problémy s funkčností detekce zdí bez vizuálně výrazných prvků. Tento problém je však spíše technickým omezením než nedostatkem implementace aplikace. Nicméně, tento problém by se neměl vyskytovat na telefonech, které disponují LiDAR hloubkovou kamerou, jelikož tyto kamery dokáží lépe detekovat a rozpoznat roviny zdí bez ohledu na vizuální výraznost.

Přestože diagram rozvodů v prostředí rozšířené reality občas malinko poposkočil kvůli problémům s detekcí, uživatelé byli stále schopni využít editor diagramu k zaznamenání a detekci polohy rozvodů. Často není důležité znát polohu rozvodů „na milimetr přesně“, ale spíše mít orientační představu o tom, kudy vlastně vedou, což aplikace úspěšně zajišťuje.

Navzdory těmto problémům fungovala aplikace během testování stabilně, což ukazuje na její spolehlivost a připravenost pro další nasazení.

Kapitola 7

Závěr

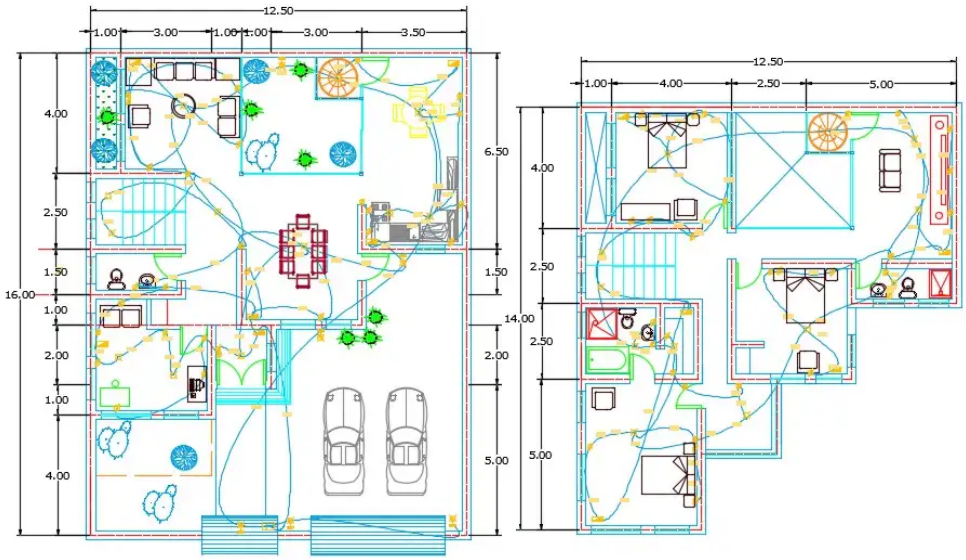
Tato práce úspěšně navrhuje a implementuje mobilní aplikaci pro operační systém *iOS*, která řeší problematiku zaznamenání a následné vizualizace umístění rozvodů elektřiny a vody ve zdech bytu pomocí rozšířené reality. Aplikace splňuje všechny zadefinované cíle, včetně analýzy existujících aplikací pro vizualizaci rozvodů v bytě, specifikace funkčních a nefunkčních požadavků, návrhu uživatelského rozhraní a softwarového řešení, implementace funkčního prototypu a provedení uživatelského testování s následnou analýzou výsledků.

Aplikace nabízí uživatelům snadno ovladatelný nástroj pro tvorbu a ukládání schémat rozvodů, který následně poskytuje orientační představu o jejich poloze. Umožňuje tak uživatelům bez odborných znalostí zaznamenat a vizualizovat rozvody ve zdech svého bytu, čímž zvyšuje bezpečnost a usnadňuje budoucí úpravy bytu. Aplikace zajišťuje funkční řešení pro zobrazení a zaznamenávání rozvodů elektřiny a vody ve zdech bytu pomocí rozšířené reality, čímž naplňuje svůj hlavní cíl.

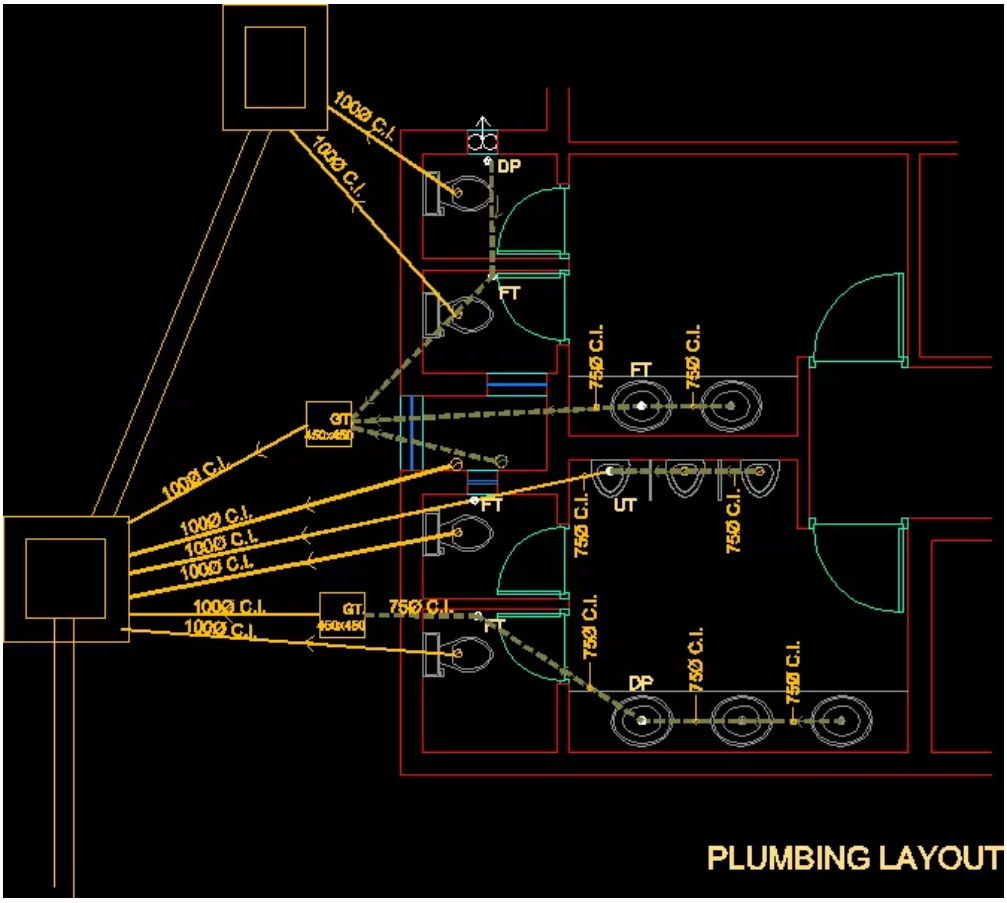
Díky splnění všech dílčích cílů představuje tato práce úspěšný přínos k řešení problematiky lokalizace rozvodů v bytových jednotkách. Navíc aplikace nabízí prostor pro další rozšíření a zlepšení, jako je například integrace frameworku *CloudKit* pro sdílení dat mezi zařízeními a jejich zálohování, což by zvýšilo uživatelský komfort a spolehlivost aplikace.

Příloha A

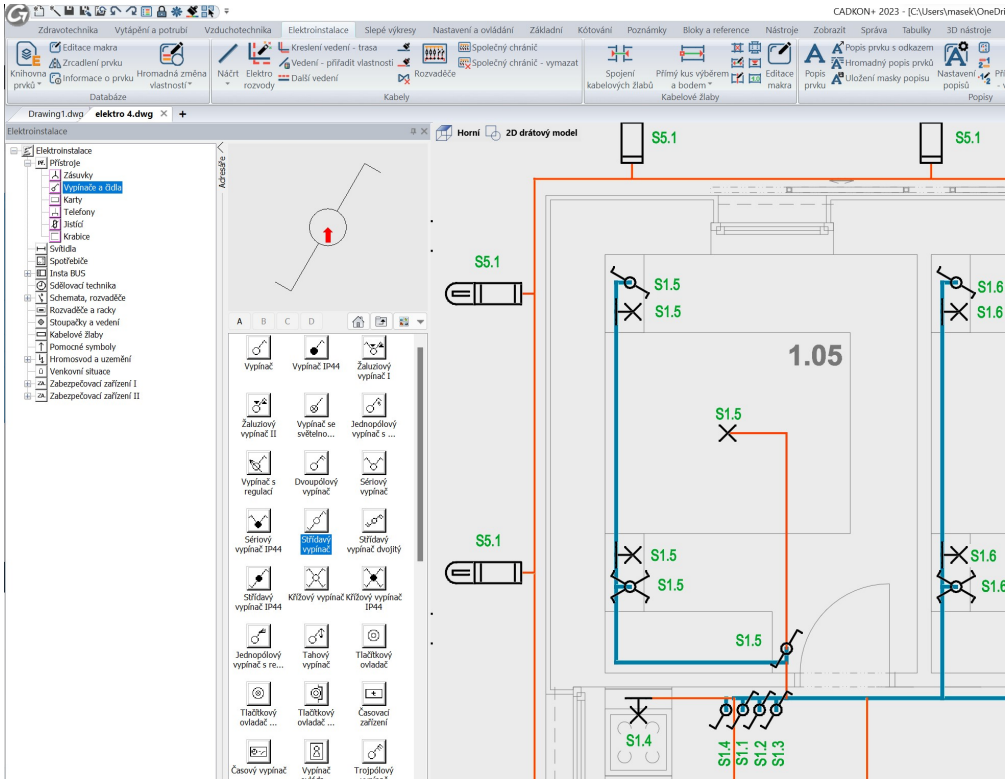
Ukázky diagramů rozvodů elektřiny a vody v CAD programech



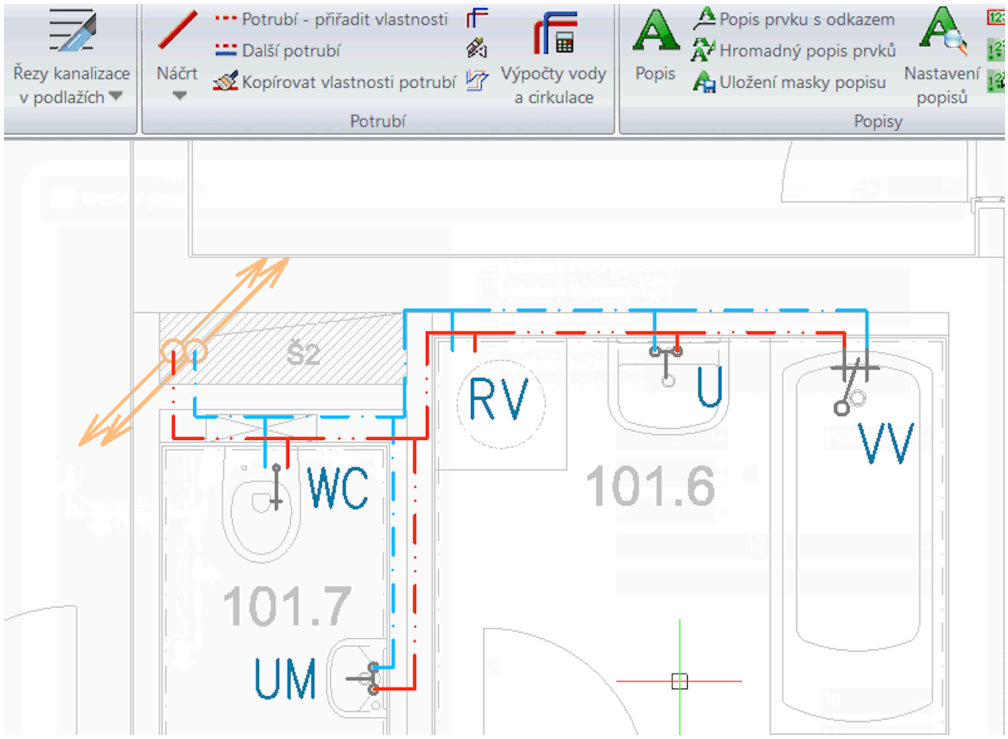
Obrázek A.1 Ukázka návrhu rozvodů elektřiny v programu *AutoCAD* [52]



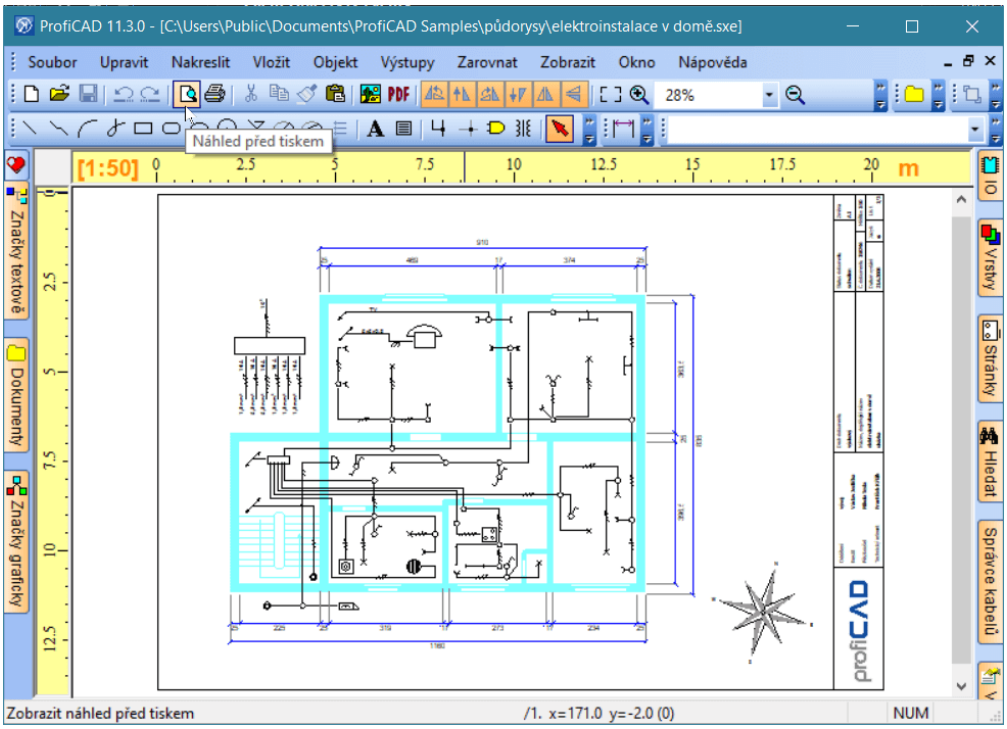
Obrázek A.2 Ukázka návrhu odpadního potrubí v programu *AutoCAD* [53]



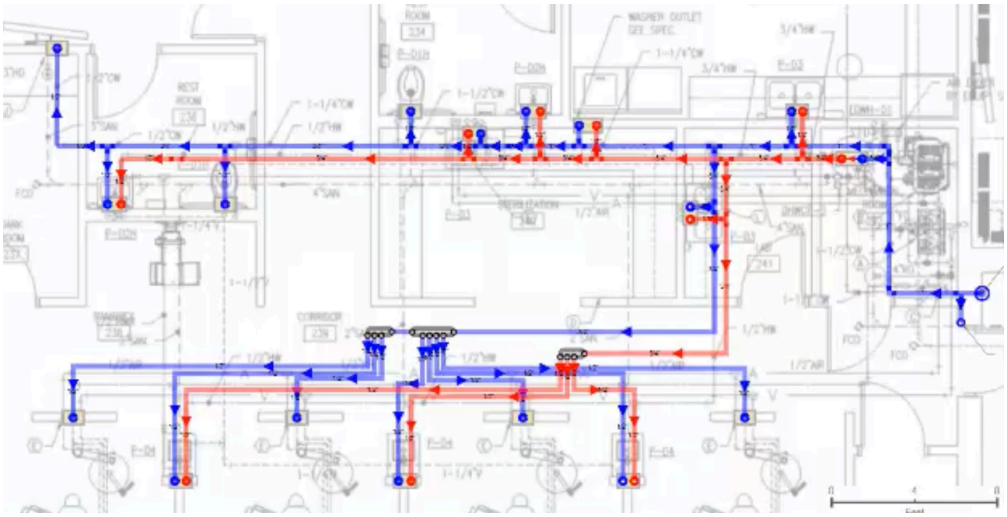
Obrázek A.3 Ukázka návrhu rozvodů elektřiny v programu *CADKON+ MEP* [54]



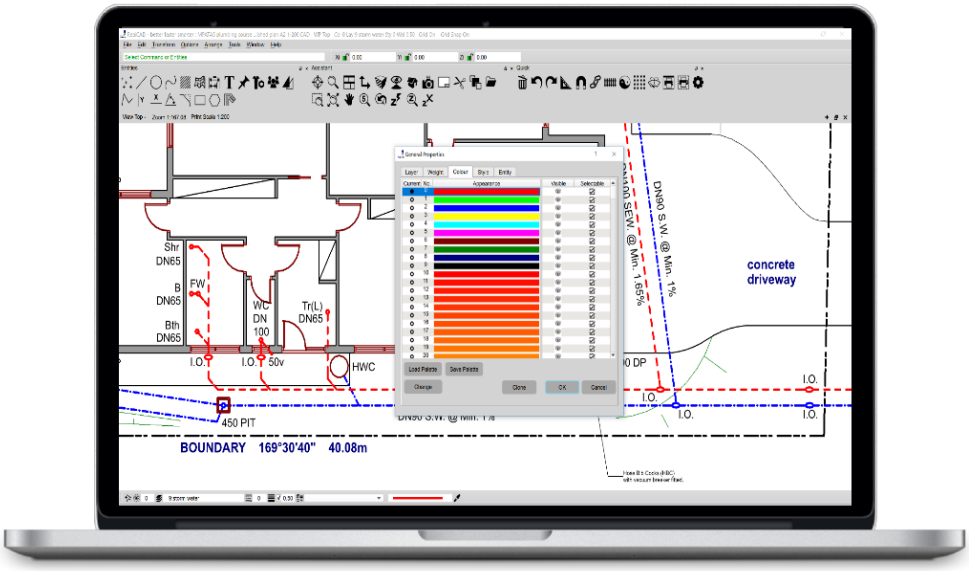
Obrázek A.4 Ukázka návrhu rozvodů pro teplou a studenou vodu v programu *CADKON+ MEP* [55]



Obrázek A.5 Ukázka návrhu rozvodů elektřiny v programu *ProfiCAD* [56]



Obrázek A.6 Ukázka návrhu rozvodů vody v programu *PlumbingCAD* [57]



Obrázek A.7 Ukázka návrhu potrubí v programu *PlumbersCAD* [15]

Příloha B

Naměřená data vzdálenosti detekce referenčních obrázků

B.1 Test 1 – ideální referenční obrázek – naměřená data

Tabulka B.1 Vzdálenost prvotní detekce u odlišných referenčních obrázků

	A	B	C	D	E	F
1. měření	-	88 cm	94 cm	109 cm	95 cm	92 cm
2. měření	-	89 cm	94 cm	111 cm	81 cm	93 cm
3. měření	-	92 cm	93 cm	101 cm	95 cm	92 cm
4. měření	-	97 cm	89 cm	111 cm	97 cm	94 cm
5. měření	-	96 cm	111 cm	112 cm	98 cm	97 cm
6. měření	-	94 cm	111 cm	109 cm	100 cm	95 cm

Tabulka B.2 Vzdálenost konce detekce odlišných referenčních obrázků

	A	B	C	D	E	F
1. měření	-	290 cm	310 cm	331 cm	357 cm	298 cm
2. měření	-	256 cm	435 cm	339 cm	198 cm	269 cm
3. měření	-	245 cm	480 cm	349 cm	344 cm	330 cm
4. měření	-	195 cm	500 cm	290 cm	341 cm	330 cm
5. měření	-	243 cm	500 cm	320 cm	415 cm	374 cm
6. měření	-	284 cm	500 cm	360 cm	407 cm	309 cm

B.2 Test 2 – vliv velikosti obrázku na maximální vzdálenost detekce – naměřená data

Tabulka B.3 Vzdálenost začátku detekce referenčního obrázku v závislosti na velikosti obrázku

	25 mm	50 mm	75 mm	100 mm	125 mm	150 mm	197 mm
1. měření	14 cm	24 cm	40 cm	49 cm	68 cm	76 cm	111 cm
2. měření	13 cm	26 cm	37 cm	49 cm	66 cm	76 cm	101 cm
3. měření	15 cm	24 cm	38 cm	51 cm	63 cm	87 cm	99 cm
4. měření	14 cm	27 cm	36 cm	47 cm	74 cm	75 cm	113 cm
5. měření	14 cm	26 cm	40 cm	52 cm	70 cm	82 cm	101 cm
6. měření	16 cm	23 cm	37 cm	48 cm	70 cm	76 cm	104 cm

Tabulka B.4 Vzdálenost konce detekce referenčního obrázku v závislosti na velikosti obrázku

	25 mm	50 mm	75 mm	100 mm	125 mm	150 mm	197 mm
1. měření	151 cm	145 cm	263 cm	280 cm	500 cm	500 cm	500 cm
2. měření	147 cm	222 cm	306 cm	355 cm	431 cm	500 cm	425 cm
3. měření	97 cm	217 cm	272 cm	283 cm	270 cm	443 cm	500 cm
4. měření	65 cm	191 cm	270 cm	371 cm	500 cm	500 cm	500 cm
5. měření	93 cm	159 cm	301 cm	403 cm	395 cm	470 cm	500 cm
6. měření	91 cm	180 cm	278 cm	283 cm	455 cm	415 cm	500 cm

Příloha C

Scénáře uživatelského testování – instrukce

Scénář 1 – Nastavení vlastního kotvícího obrázku

Instrukce pro moderátora testu

- Před začátkem tohoto testu účastník dostane spolu s telefonem několik vytištěných obrázků. Tyto obrázky jsou zároveň nahrány do galerie v telefonu.

Zde ustřihněte

Instrukce pro účastníka testu (scénář 1)

1. Zapněte aplikaci.
2. Přečtěte si instrukce a zvolte možnost vlastní obrázek.
3. Z poskytnutých obrázků vyberte ten, který co nejvíce splňuje zobrazené instrukce.
4. Vyplňte informace o poskytnutém obrázku tak, aby co nejvíce odpovídaly realitě.
5. Potvrďte svou volbu.

Scénář 2 – Nastavení výchozího kotvícího obrázku

Instrukce pro moderátora testu

- Na začátku testu uživatel dostane pouze telefon s nainstalovanou aplikací.
- Pokud je scénář prováděn na místě bez možnosti tisku, ve chvíli kdy účastník testu klikne na sdílet a následně na tisknout, je mu předán předem vytištěný kotvící obrázek.

Zde ustřihněte

Instrukce pro účastníka testu (scénář 2)

1. Spusťte aplikaci.
2. Přečtěte si instrukce a zvolte možnost výchozí obrázek.
3. Postupujte dle zobrazených instrukcí.
4. Potvrďte svou volbu.

Scénář 3 – Záznam polohy rozvodů

Instrukce pro moderátora testu

- Před začátkem testu je pomocí malířské pásky na zdi vyznačena poloha rozvodů. Na pásku jsou tužkou vyznačeny údaje, o jaký typ rozvodu se jedná.
- Než je telefon předán účastníkovi testu, je do telefonu nahrána verze aplikace s nastavenými údaji o kotvícím obrázku.
- Na začátku testu uživatel dostane telefon a vytištěný kotvící obrázek.
- Účastníkovi testu je sděleno, na jaké zdi bude test probíhat.

Zde ustřihněte

Instrukce pro účastníka testu (scénář 3)

1. Vaším úkolem je v aplikaci zaznamenat polohu rozvodů na moderátorem určené zdi.
2. Postupujte dle instrukcí v aplikaci a pomocí intuice.

Scénář 4 – Lokalizace rozvodů z již existujícího záznamu v aplikaci

Instrukce pro moderátora testu

- Před předáním telefonu účastníkovi testu, je do telefonu nahrána verze aplikace s nastavenými údaji o kotvícím obrázku a daty o poloze rozvodů.
- Na začátku testu uživatel dostane telefon, vytištěný kotvící obrázek a malířskou pásku.
- Účastníkovi testu je sděleno, na jaké zdi bude test probíhat.

Zde ustřihněte

Instrukce pro účastníka testu (scénář 4)

1. V aplikaci nalezněte informace o moderátorem zadané zdi.
2. Zapněte AR editor rozvodů a s pomocí malířské pásky na reálné zdi vyznačte, kudy v ní vedou rozvody, tak jak to ukazuje vizualizace v aplikaci.

Scénář 5 – Editace existujícího diagramu

Instrukce pro moderátora testu

- Před předáním telefonu účastníkovi testu, je do telefonu nahrána verze aplikace s nastavenými údaji o kotvícím obrázku a daty o poloze rozvodů.
- Na začátku testu uživatel dostane telefon a vytištěný kotvící obrázek.
- Účastníkovi testu je sděleno, na jaké zdi bude test probíhat.

Zde ustřihněte

Instrukce pro účastníka testu (scénář 5)

1. V aplikaci nalezněte informace o moderátorem zadané zdi.
2. Zapněte AR editor rozvodů a vytvořte fotografii místnosti poté, co se zobrazí vizualizace rozvodů.
3. Od prvotního měření proběhla rekonstrukce, proto ze scény odstraňte rozvod vody.
4. Součástí rekonstrukce byl i drobný posun vedení elektřiny. Proto v aplikaci posuňte rozvod elektřiny tak, aby lícoval s hranou podlahy.
5. Vytvořte fotografii změněné vizualizace.
6. Vraťte se v aplikaci na obrazovku s informacemi o zdi a upravte popisek zaznamenaných fotografií, tak aby bylo jasné, jaká fotografie zachycuje stav před a jaká po rekonstrukci.
7. Obě zaznamenané fotografie uložte do galerie telefonu.

Příloha D

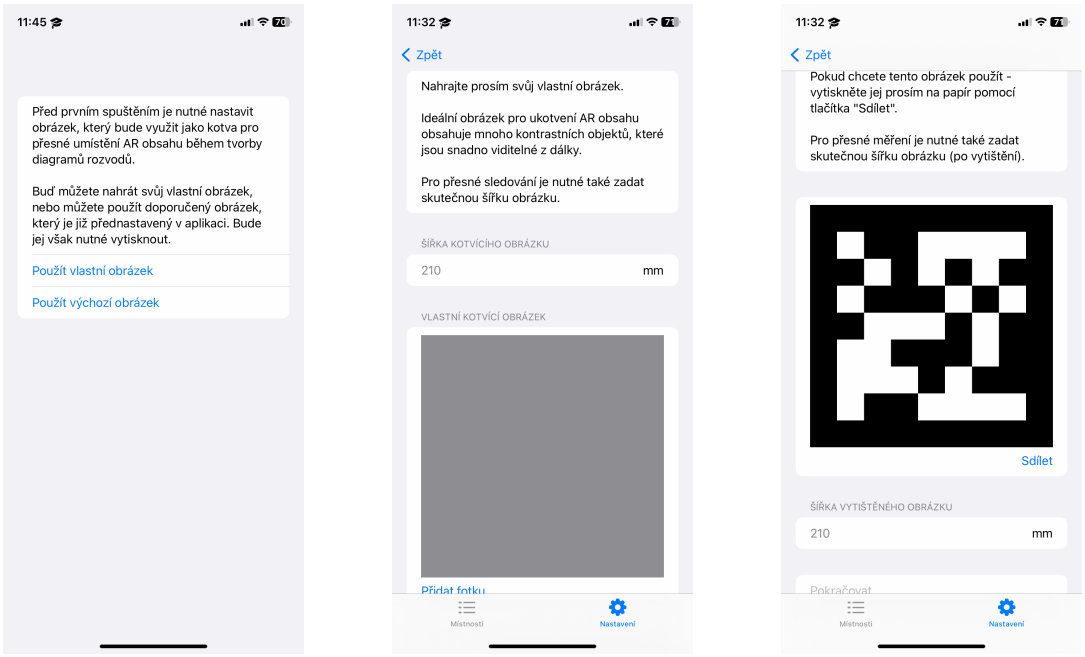
Formulář pro účastníky uživatelského testování použitelnosti

Podpisem níže souhlasíte, že budete během testu nahráváni. Tento záznam bude sloužit pouze pro analýzu průběhu testu a následně bude smazán. Nahrávat se bude záznam obrazovky na testovacím zařízení a dále bude místnost zabírat kamera, která bude nahrávat video i zvukový záznam.

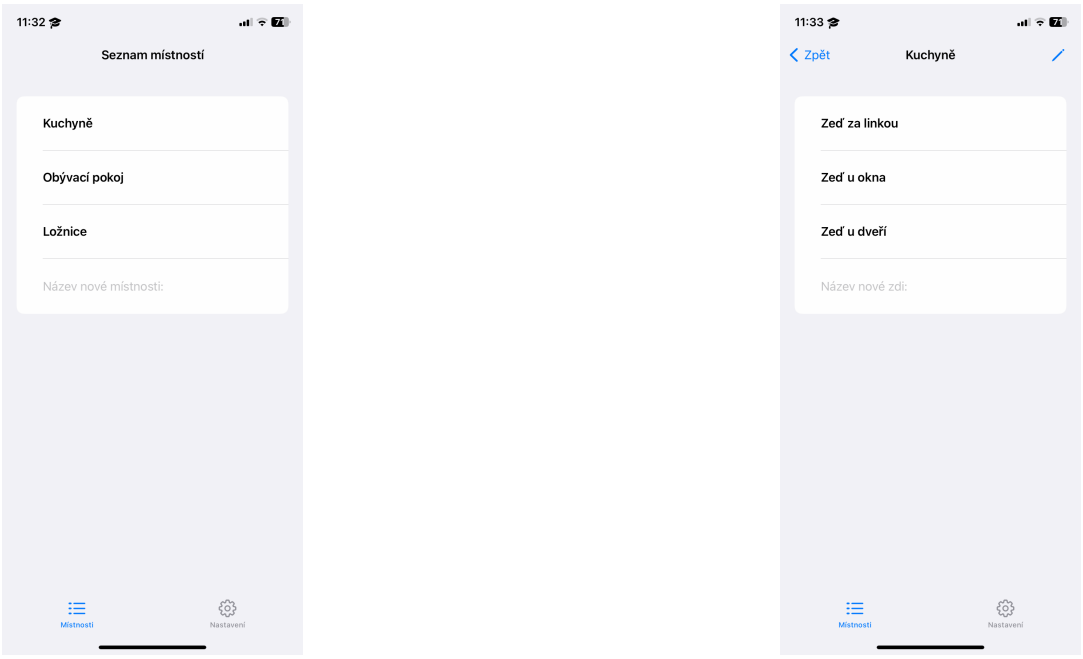
V dne Podpis

Příloha E

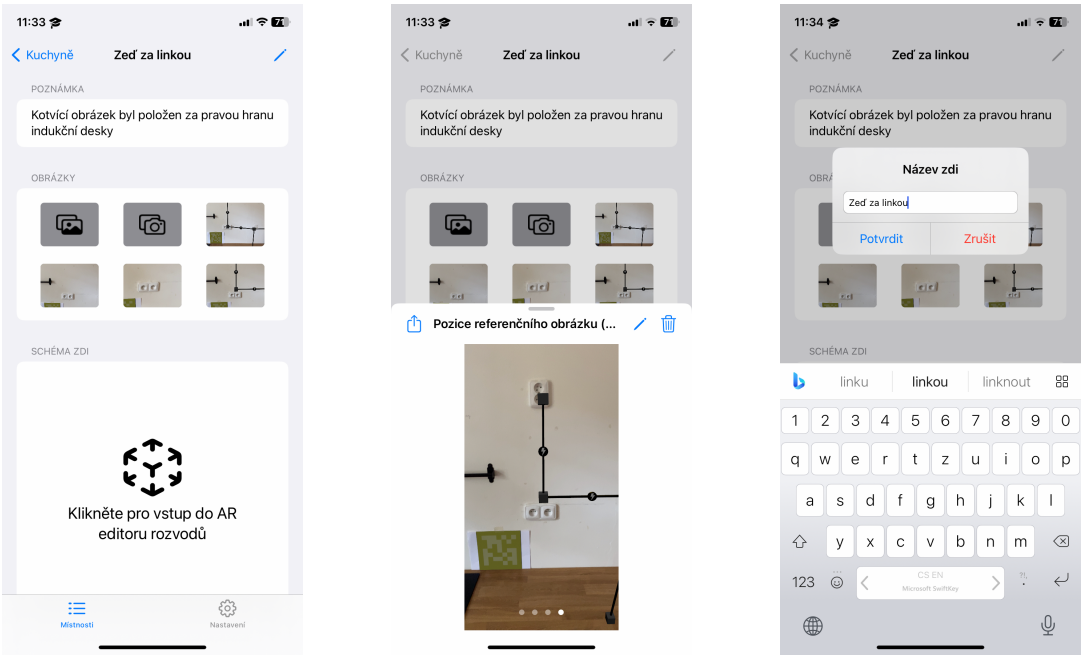
Snímky obrazovky výsledné aplikace



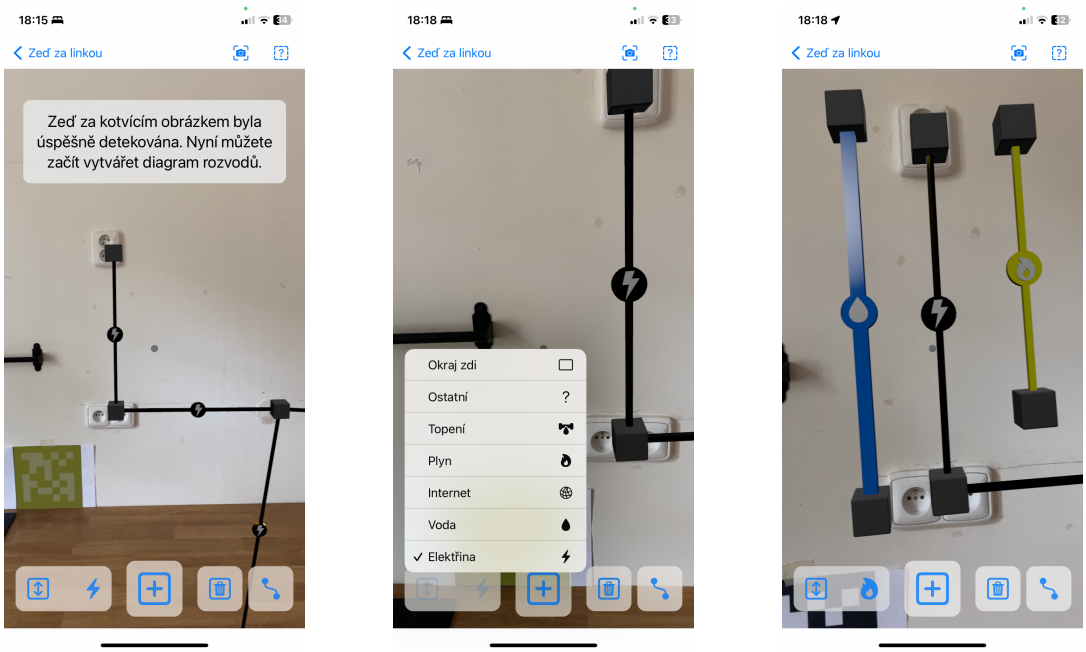
Obrázek E.1 Snímky obrazovky nastavení kotvícího obrázku



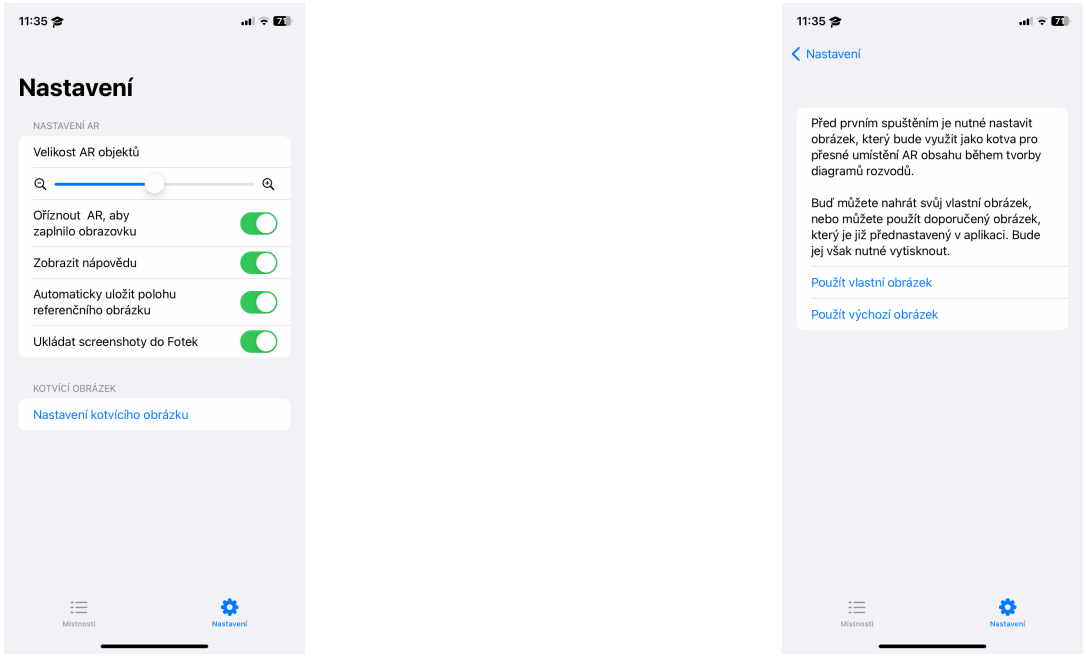
Obrázek E.2 Snímky obrazovky seznamu místností a detailu místnosti



Obrázek E.3 Snímky obrazovky detailu zdi



Obrázek E.4 Snímky obrazovky AR editoru rozvodů



Obrázek E.5 Snímky obrazovky nastavení

Bibliografie

1. FURHT, Borko (ed.). Augmented Reality. In: *Encyclopedia of Multimedia* [online]. Přel. RIDZOŇ, Daniel. Boston, MA: Springer US, 2006, s. 29–31 [cit. 2023-01-31]. ISBN 978-0-387-30038-2. Dostupné z DOI: 10.1007/0-387-30038-4_10.
2. PAINE, James. 10 Real Use Cases for Augmented Reality. *Inc.* [online]. 2018 [cit. 2023-01-31]. Dostupné z: https://www.inc.com/james-paine/10-real-use-cases-for-augmented-reality.html.
3. APPLE INC. Understanding World Tracking. *Apple Developer Documentation* [online]. 2023 [cit. 2023-02-01]. Dostupné z: https://developer.apple.com/documentation/arkit/configuration_objects/understanding_world_tracking.
4. APPLE INC. Framework ARKit. *Apple Developer Documentation* [online]. 2023 [cit. 2023-02-04]. Dostupné z: https://developer.apple.com/documentation/arkit.
5. APPLE INC. Apple unveils new iPad Pro with breakthrough LiDAR Scanner and brings trackpad support to iPadOS. *Apple Newsroom* [online]. 2020 [cit. 2023-02-04]. Dostupné z: https://www.apple.com/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados/.
6. APPLE INC. Framework RoomPlan. *Apple Developer Documentation* [online]. 2023 [cit. 2023-02-04]. Dostupné z: https://developer.apple.com/documentation/roomplan.
7. SCARAMUZZA, Davide; ZHANG, Zichao. Aerial Robots, Visual-Inertial Odometry of. In: *Encyclopedia of Robotics* [online]. Ed. ANG, Marcelo H; KHATIB, Oussama; SICILIANO, Bruno. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, s. 1–9 [cit. 2023-05-01]. ISBN 978-3-642-41610-1. Dostupné z DOI: 10.1007/978-3-642-41610-1_71-1.
8. KLAUZ, Milan. Jak vybrat vhodný elektro-CAD. *ElektroPrůmysl.cz* [online]. 2014 [cit. 2023-01-25]. ISSN 2571-0761. Dostupné z: https://www.elektroprumysl.cz/software/jak-vybrat-vhodny-elektro-cad.
9. HOME DESIGN INSTITUTE SASU. *Making your house a home with AutoCAD* [online]. 2022. [cit. 2023-01-25]. Dostupné z: https://homedesigninstitute.com/read_news/687/making_your_house_a_home_with_AutoCAD/.

10. AUTODESK INC. *AutoCAD: 2D a 3D CAD aplikace, na kterou spoléhají miliony lidí při navrhování a automatizaci návrhů odkudkoli a kdykoli* [online]. 2023. [cit. 2023-01-25]. Dostupné z: https://www.autodesk.cz/products/autocad/overview.

11. VICTORIA. *How Different Industries Use DWG Files — From architects to game designers* [online]. 2023. [cit. 2023-02-17]. Dostupné z: https://www.scan2cad.com/blog/dwg/how-different-industries-use-dwg/.

12. GRAITEC S.R.O. *CADKON+ MEP* [online]. 2023. [cit. 2023-01-25]. Dostupné z: https://www.cadkon.eu/cz/cadkon-mep.html.

13. PROFICAD. *CAD Software pro elektro dokumentaci* [online]. [cit. 2023-01-25]. Dostupné z: https://www.proficad.cz.

14. AVENIR SOFTWARE INC. *PlumbingCAD® 2021* [online]. 2023. [cit. 2023-02-21]. Dostupné z: https://www.avenir-online.com/AvenirWeb/PlumbingCAD/PCADHome.aspx.

15. CAD INTERNATIONAL. *PlumbersCAD* [online]. 2022. [cit. 2023-02-22]. Dostupné z: https://cad.com.au/software/plumbers-cad/.

16. APPLE INC. *About iOS 11 Updates* [online]. 2020. [cit. 2023-01-20]. Dostupné z: https://support.apple.com/en-us/HT208067.

17. APPLE INC. *iOS 11 is available tomorrow* [online]. 2017. [cit. 2023-01-24]. Dostupné z: https://www.apple.com/newsroom/2017/09/ios-11-available-tomorrow/.

18. INTER IKEA SYSTEMS B.V. *IKEA* [online]. 2019. [cit. 2023-02-22]. Dostupné z: https://apps.apple.com/cz/app/ikea/id1452164827.

19. APPLE INC. *Měření* [online]. 2022. [cit. 2023-02-22]. Dostupné z: https://apps.apple.com/cz/app/measure/id1383426740.

20. LOCOMETRIC LTD. *RoomScan Classic* [online]. 2023. [cit. 2023-02-22]. Dostupné z: https://apps.apple.com/cz/app/roomscan-classic/id673673795.

21. SENSOPIA INC. *magicplan* [online]. 2022. [cit. 2023-02-22]. Dostupné z: https://apps.apple.com/cz/app/magicplan/id427424432.

22. VGIS INC. *vGIS – High-accuracy Augmented Reality for BIM, GIS, and 3D Scans* [online]. 2023. [cit. 2023-03-01]. Dostupné z: https://www.vgis.io.

23. SPECTAR. *Spectar – Mixed Reality With In Construction Starts Here* [online]. 2023. [cit. 2023-03-08]. Dostupné z: https://spectar.io/home/.

24. HAMIL, Stephen. *What is Building Information Modelling (BIM)?* [online]. 2021. [cit. 2023-03-01]. Dostupné z: https://www.thenbs.com/knowledge/what-is-building-information-modelling-bim.

25. SPEICHER, Maximilian; HALL, Brian D.; NEBELING, Michael. What is Mixed Reality? [online]. 2019, s. 1–15 [cit. 2023-03-09]. ISBN 9781450359702. Dostupné z DOI: 10.1145/3290605.3300767.

26. CHRISTIAN, Max. *U.S. Patent US8868375B1 – Generation of a floor plan* [online]. 2014. [cit. 2023-03-02]. Dostupné z: https://patents.google.com/patent/US8868375B1/en.

27. LOCOMETRIC LTD. *RoomScan Pro LiDAR floor plans* [online]. 2022. [cit. 2023-03-02]. Dostupné z: https://apps.apple.com/cz/app/roomscan-pro-lidar-floor-plans/id1504050801.

28. GEIB, Zuzanna. *Why Apple's LiDAR Scanner Opens Up a Brave New World of Indoor Mapping* [online]. 2023. [cit. 2023-03-04]. Dostupné z: https://blog.magicplan.app/why-apples-lidar-scanner-opens-up-a-brave-new-world-of-indoor-mapping.

29. VGIS INC. *Explore vGIS use cases. Real-life applications of high-accuracy augmented reality.* [online]. 2022. [cit. 2023-03-08]. Dostupné z: https://www.vgis.io/vgis-utilities-use-cases-high-accuracy-survey-grade-augmented-reality-ar-bim-gis/.

30. VGIS INC. *AR Jobsite Management System* [online]. 2022. [cit. 2023-03-08]. Dostupné z: https://www.vgis.io/ar-augmented-reality-construction-building-jobsite-management-bim-architecture-design-engineering-mep-hvac-structural/.

31. MLEJNEK, Jiří. *BI-SI1.2 3. přednáška* [online]. CTU, Faculty of Information Technology, 2021 [cit. 2023-03-16]. Dostupné z: https://youtu.be/7SWBrr4HWns.

32. CASSERLY, Martyn. *Phone OS and iOS: Every version released so far* [online]. 2023. [cit. 2023-04-11]. Dostupné z: https://www.macworld.com/article/1659017/ios-versions-list.html.

33. STATCOUNTER. *Mobile iOS Version Market Share Worldwide (Mar 2022 - Mar 2023)* [online]. 2023. [cit. 2023-04-11]. Dostupné z: https://gs.statcounter.com/ios-version-market-share/mobile/worldwide/#monthly-202203-202303.

34. STATCOUNTER. *Mobile Android Version Market Share Worldwide (Mar 2022 - Mar 2023)* [online]. 2023. [cit. 2023-04-11]. Dostupné z: https://gs.statcounter.com/android-version-market-share/mobile/worldwide/#monthly-202203-202303.

35. APPLE INC. *Swift – The powerful programming language that is also easy to learn.* [online]. 2023. [cit. 2023-04-13]. Dostupné z: https://developer.apple.com/swift/.

36. JEROEN, L. *UIKit vs. SwiftUI: How to Choose the Right Framework for Your App* [online]. 2023. [cit. 2023-04-15]. Dostupné z: https://getstream.io/blog/uikit-vs-swiftui/.

37. APPLE INC. *ARKit – ARConfiguration.WorldAlignment.gravity* [online]. 2023. [cit. 2023-04-20]. Dostupné z: https://developer.apple.com/documentation/arkit/arconfiguration/worldalignment/gravity.

38. OPENCV. *Detection of ArUco Markers* [online]. 2023. [cit. 2023-04-21]. Dostupné z: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html.

39. TIWARI, Sumit. An Introduction to QR Code Technology. In: *2016 International Conference on Information Technology (ICIT)*. 2016, s. 39–44. Dostupné z DOI: 10.1109/ICIT.2016.021.

40. NOVÁK, Petr. *BI-PST Přednáška 12 – Lineární regrese* [online]. CTU, Faculty of Information Technology, 2022 [cit. 2023-04-24]. Dostupné z: https://courses.fit.cvut.cz/BI-PST/media/lectures/BI-PST-Lec12-Slides.pdf.

41. ÖZACAR, Kasım et. al. 3D Selection Techniques for Mobile Augmented Reality Head-Mounted Displays. *Interacting with Computers*. 2016, roč. 29. Dostupné z DOI: 10.1093/iwc/iww035.
42. APPLE INC. *RealityKit* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://developer.apple.com/documentation/realitykit.
43. APPLE INC. *Creating 3D Content with Reality Composer* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://developer.apple.com/documentation/realitykit/creating-3d-content-with-reality-composer.
44. APPLE INC. *SceneKit* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://developer.apple.com/documentation/scenekit.
45. APPLE INC. *SpriteKit* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://developer.apple.com/documentation/spritekit/.
46. APPLE INC. *Metal* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://developer.apple.com/documentation/metal/.
47. MISHRA, Abhishek. The MVVM Architectural Pattern. In: *iOS Code Testing: Test-Driven Development and Behavior-Driven Development with Swift*. Berkeley, CA: Apress, 2017, s. 43–60. ISBN 978-1-4842-2689-6. Dostupné z DOI: 10.1007/978-1-4842-2689-6_3.
48. NAUMOV, Alexey. *Clean Architecture for SwiftUI* [online]. 2019. [cit. 2023-04-27]. Dostupné z: https://nalexn.github.io/clean-architecture-swiftui/.
49. APPLE INC. *App icons* [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/foundations/app-icons#ios-ipados/.
50. APPLE INC. *Recording and Replaying AR Session Data* [online]. 2023. [cit. 2023-04-04]. Dostupné z: https://developer.apple.com/documentation/arkit/arsession/recording_and_replaying_ar_session_data.
51. DUMAS, J.S.; REDISH, J.C. *A Practical Guide to Usability Testing*. Intellect Books, 1999. Human/computer interaction. ISBN 9781841500201. Dostupné také z: https://books.google.cz/books?id=4lge5k%5C_F9EwC.
52. GHATGE, Akansha. *House Wiring Plan Drawing AutoCAD File Download* [online]. 2019. [cit. 2023-01-25]. Dostupné z: https://cadbull.com/detail/146973/House-Wiring-Plan-Drawing-AutoCAD-File-Download.
53. SAMIM, Sekh. *Toilet plumbing* [online]. 2023. [cit. 2023-01-25]. Dostupné z: https://www.bibliocad.com/en/library/toilet-plumbing_45150/.
54. GRAITEC S.R.O. *Projekty silnoproudu* [online]. 2023. [cit. 2023-01-25]. Dostupné z: https://www.cadkon.eu/cz/elektro-rozvody.html.
55. GRAITEC S.R.O. *Potrubí ZTI včetně výpočtů* [online]. 2023. [cit. 2023-01-25]. Dostupné z: https://www.cadkon.eu/cz/potrubi-pro-zti.html.
56. PROFICAD. *Pracovní plocha - ukázky výkresů* [online]. [cit. 2023-01-25]. Dostupné z: https://www.proficad.cz/ukazky.aspx.

57. AVENIR SOFTWARE INC. *Video Tutorial - Defining the Plumbing System* [online]. 2018. [cit. 2023-02-21]. Dostupné z: https://www.avenir-online.com/AvenirWeb/PlumbingCAD/PCADViewTutorial.aspx?Page=PCAD%5C%20Defining%5C%20Plumbing_2018.mp4.

Obsah přiloženého datového archivu

- `README.md` ..... instrukce pro instalaci a spuštění aplikace
- `src`
  - `WallInfo` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF
- `video`
  - `video.mp4` ..... video s ukázkou aplikace