Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka (INF)**
Specjalność: **-**

PRACA DYPLOMOWA INŻYNIERSKA

Rozszerzona rzeczywistość dla aplikacji mobilnej „Multimedialny przewodnik po Dolnym Śląsku"

Michał Lubowicz

Opiekun pracy
dr. inż. Krzysztof Waśko

Słowa kluczowe:
Rozszerzona rzeczywistość
Aplikacja mobilna
Rozpoznawanie obrazu

WROCŁAW 2021

Streszczenie

Temat: Rozszerzona rzeczywistość dla aplikacji mobilnej - Multimedialny przewodnik po Dolnym Śląsku

Praca przedstawia projekt modułu rozszerzonej rzeczywistości do aplikacji mobilnej będącej przewodnikiem turystycznym. Przedstawiony moduł służy do pozyskiwania informacji o eksponatach muzealnych, ułatwieniu oraz uatrakcyjnieniu interakcji ze sztuką poprzez przekazanie szerszych informacji o każdym rodzaju sztuki. Początek pracy poświęcony jest analizie zagadnienia rozszerzonej rzeczywistości oraz istniejących rozwiązań. W dalszej części opisywany jest proces projektowania oraz implementacji. W ramach rozszerzonej rzeczywistości aplikacja rozpoznaje eksponat dzięki wykorzystaniu algorytmu rozpoznawania obrazu, a następnie przedstawia podstawowe informacje o nim. Pozwala to użytkownikowi na uzyskanie większej ilości informacji o dziele niż dzięki standardowym tabliczkom muzealnym. Uzupełnieniem dla informacji w rozszerzonej rzeczywistości jest widok szczegółów eksponatu, w którym użytkownik może przeczytać pełniejsze informacje o dziele. Omówiona w pracy aplikacja została ukończona i jest możliwa do użytku dla wybranych dzieł z Muzeum Narodowego we Wrocławiu.

Abstract

Title: Augmented reality for mobile application „Multimedia guide to Lower Silesia”

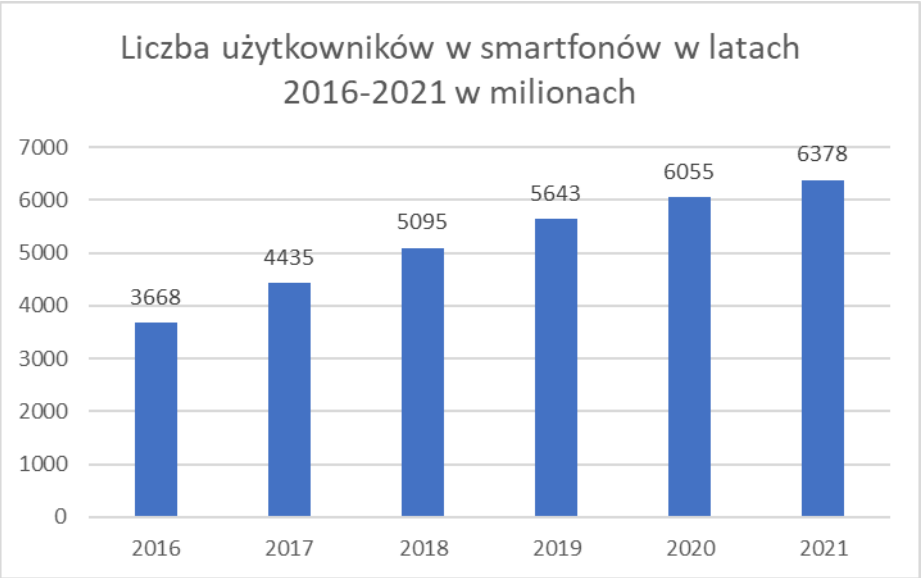
The work presents project of an augmented reality module for a mobile application, which is a tourist guide. Presented module is used for sourcing information about museum exhibits, and for facilitating and making interaction with art through providing more information about each type of art more attractive. The beginning of this project is devoted to the analysis of the issue of augmented reality and existing solutions. The next part describes the process of design and implementation. As part of augmented reality, the application recognizes the exhibit thanks to the use of a recognition algorithm and then presents basic information about it. This allows the user to obtain more information about the work than through standard museum plaques. Complementing the information in augmented reality is the exhibit detail view, where the user can read more complete information about the work. The application discussed in the work has been completed and is available for use for selected exhibits in National Museum in Wrocław.

Spis treści

Wstęp

Wpływ smartfonów na życie ludzi

Telefony komórkowe, a następnie smartfony, zmieniły sposób funkcjonowania ludzkości, z roku na rok stanowią coraz istotniejszą część życia większości ludzi. Już lata temu zastąpiły wiele innych urządzeń takich jak kalkulatory, nawigacje samochodowe, odtwarzacze muzyki czy aparaty fotograficzne. Akcesoria te bardzo rzadko spotyka się dzisiaj jako osobne urządzenia. Ludzie powszechnie używają telefonów, żeby ułatwiać sobie życie lub je uprzyjemnić, czy to grając w gry, czy oglądając filmy. Na rysunku 1. przedstawiono jak na przestrzeni lat 2016-2021 zmieniała się liczba smartfonów na świecie. W 2016 roku na świecie było 3,664 mld smartfonów - zestawiając to z liczbą ludności na świecie, na jednego mieszkańca Ziemi przypadało średnio 0,49 smartfonu, w 2021 wskaźnik wzrósł już do 0,81 [1][2].



Rysunek 1: Wykres liczby użytkowników telefonów komurkowych w latach 2016-2021 w milionach – opracowanie na podstawie [1]

Aplikacje mobilne

Rynek aplikacji mobilnych w 2020 roku w skali świata był wart ponad 170 miliardów dolarów [3]. Z roku na rok coraz więcej firm inwestuje w tego typu oprogramowania w celu czerpania zysków - czasem sama aplikacja jest produktem, a czasem tylko platformą reklamową lub pośredniczącą w dostępie do innego produktu.

Wprowadzenie do tematu rozszerzonej rzeczywistości

Alan B. Craig w swojej książce *Understanding Augmented Reality: Concepts and Applications:* „Rozszerzona rzeczywistość to medium, w którym informacje są dodawane do świata fizycznego w interakcji z nim” [4] – definicja ta dobrze obrazuje proporcje w rozszerzonej rzeczywistości, podstawom jest świat fizyczny, a inne rzeczy są do niego dodawane.

Jeszcze kilkanaście lat temu mały odsetek ludzi miał pojęcie, czym jest rozszerzona rzeczywistość. Dziś jest ona powszechna, a w oparciu o nią powstają liczne rozwiązania, poczynając od specjalistycznych, skierowanych do konkretnych specjalistów, np. systemy wyświetlające informacje ułatwiające wykonanie zadań w procesie produkcji, poprzez aplikacje pomagające użytkownikom w codziennym życiu, jak nawigacje, aż po rozrywkę i gry, w których kontekście należy od razu zaznaczyć różnice pomiędzy rozszerzoną rzeczywistością, a powszechną w dzisiejszych czasach w grach komputerowych, wirtualną rzeczywistością. Główną różnicą pomiędzy AR a VR jest wykorzystywany sprzęt. W przypadku rozszerzonej rzeczywistości niezbędne jest urządzenie wyposażone w kamerę oraz ekran, który wyświetla obraz będący podglądem z kamery w czasie rzeczywistym, z naniesionymi dodatkowymi informacjami. Wirtualna rzeczywistość wymaga natomiast zastosowania specjalnych gogli, które imitują wrażenie przestrzennej i fizycznej obecności w komputerowo generowanym świecie wirtualnym. Umożliwia to zupełne odcięcie zmysłu wzroku od świata rzeczywistego. „W przeciwieństwie do tradycyjnych interfejsów, VR umieszcza użytkownika wewnątrz doświadczenia. Zamiast oglądać ekran przed sobą, użytkownicy są zanurzeni i mogą wchodzić w interakcję z trójwymiarowymi światami.” [5].

Rozszerzona rzeczywistość ma bardzo wiele zastosowań praktycznych, zarówno naukowych, jak i biznesowych, często łączona jest z rozpoznawaniem obrazu. Takie połączenie umożliwia precyzyjne wyświetlanie informacji o obiektach znajdujących się w zasięgu kamery urządzenia.

Ostatnimi laty AR bardzo mocno się spopularyzowała, zarówno dzięki rozwojowi oprogramowania i bibliotek umożliwiających sprawne tworzenie aplikacji, jak i dzięki wzrostowi mocy obliczeniowej urządzeń przenośnych – głównie telefonów komórkowych. Warto wspomnieć dwie aplikacje, które bardzo mocno się do tego przyczyniły.

Pierwsza z nich to fenomen, jeśli chodzi o gry mobilne, czyli „Pokemon Go”. Została pobrana łącznie ponad miliard razy, co oczywiście nie przekłada się na miliard aktywnych użytkowników, jednak według szacunków Sensor Tower, gra przynosi wydawcy miliardowe zyski [6]. Jedną z głównych mechanik gry jest łapanie tytułowych pokemonów właśnie w rozszerzonej rzeczywistości.

Drugą aplikacją, która jest odpowiedzialna za popularyzacje tej technologii, jest Snapchat, czyli aktualnie ósmy najpopularniejszy portal społecznościowy w Polsce [7], który swoją popularność zawdzięcza filtrom działającym w rozszerzonej rzeczywistości, pozwalającym na zabawną interakcję ze zdjęciami i filmami użytkownika. Mechanika ta została skopiowana przez wiele innych portali, takich jak Instagram [8] czy Facebook [9].

Opracowana w ramach tej pracy inżynierskiej aplikacja może zostać udostępniona do użytku publicznego, jednak wymagać będzie to współpracy z obiektami, w których mogłaby być używana. Powstały projekt ma umożliwić odwiedzającym wszelkiego rodzaju muzea i wystawy sztuki łatwiejszy dostęp do informacji o zgromadzonych tam dziełach. Dzięki algorytmom rozpoznawania obrazu użytkownik nie będzie miał trudności ze zidentyfikowaniem dzieła, na które patrzy i bez trudności odczyta wszelkie interesujące go informacje z ekranu telefonu.

Geneza pracy

Niniejsza praca jest następstwem powstania aplikacji – Multimedialny przewodnik po Dolnym Śląsku, który powstał w ramach przedmiotu Zespołowe Przedsięwzięcie Inżynierskie, nakładem pracy trzyosobowego zespołu w składzie: Sebastian Ciźla, Michał Lubowicz oraz Piotr Walczak. Aplikacja w fazie projektowania została podzielona na trzy główne części: interfejs aplikacji, rozszerzona rzeczywistość oraz baza danych. W niniejszej

pracy inżynierskiej omówiona zostanie tematyka rozszerzonej rzeczywistości, zaprezentowany zostanie również proces implementacji tej części aplikacji.

Powstanie multimedialnego przewodnika po Dolnym Śląsku ma na celu zachęcenie jego użytkowników do aktywności turystycznej, poprzez odwiedzanie szeregu dostępnych atrakcji. Aplikacja realizuje ten cel na dwa sposoby. Pierwszym jest sugerowanie użytkownikowi ciekawych miejsc, które może odwiedzić, drugim natomiast jest uatrakcyjnienie zwiedzania takich miejsc jak muzea czy galerie sztuki, w których osoba nie mająca zbyt dużej wiedzy w tej dziedzinie, może podziwiać piękno dzieł sztuki oraz wzbogacać swoją wiedzę, aby w pełni je zrozumieć. W wielu miejscach eksponaty opisane są bardzo lakonicznie, czasem opisy ograniczają się do tytułu i nazwiska autora. Dzięki wykorzystaniu rozszerzonej rzeczywistości, użytkownik będzie mógł w prosty sposób uzyskać informacje o interesujących go obiektach. Wystarczy, że skieruje swój telefon w stronę eksponatu i natychmiast uzyska dostęp do szerszego zakresu informacji.

Na rynku jest wiele rozwiązań aplikacji mobilnych będących lokalnymi przewodnikami, jednak dla Dolnego Śląska jeszcze nie funkcjonuje taka aplikacja, dlatego postanowiono, że to atrakcje tego regionu będą opracowane w aplikacji. Wiele muzeów i innych atrakcji turystycznych również posiada swoje aplikacje, natomiast rzadkością jest wykorzystanie rozszerzonej rzeczywistości. Dużo powszechniejszym rozwiązaniem jest identyfikowanie eksponatów na podstawie kodów QR, których główną wadą jest to, że raczej nie pasują one do artystycznego wyrazu tych miejsc, przez co zwykle są za małe i w przypadku większego nagromadzenia ludzi może dojść do powstania kolejki do kodu. Eksponaty natomiast są zazwyczaj dobrze widoczne i jeśli zwiedzający jest w stanie dostrzec obiekt, to ma również możliwość skierowania na niego telefonu, a ten dzięki algorytmowi rozpoznawania obrazu, rozpozna go i pozwoli użytkownikowi dowiedzieć się o eksponacie więcej ciekawych informacji.

Cel pracy

Celem pracy jest wykonanie modułu aplikacji mobilnej, umożliwiającej użytkownikowi identyfikowanie eksponatów muzealnych lub innych dzieł sztuki, oraz dostarczanie mu informacji o nich za pomocą rozszerzonej rzeczywistości. Moduł ten będzie częścią projektu multimedialnego przewodnika po Dolnym Śląsku, wykonanego z myślą o telefonach firmy Apple, działających na systemie operacyjnym iOS.

Zakres pracy

Praca składa się z pięciu rozdziałów, wstępu, zakończenia i wykazu cytowanej literatury. W ramach wstępu umówiono genezę i cel pracy oraz pokrótce wprowadzono do tematyki pracy. W rozdziale pierwszym przedstawiono wymagania projektowe. Rozdział drugi to omówienie obecnego stanu wiedzy i techniki dotyczącej tematyki pracy, w ramach tego rozdziału dokonano również przeglądu istniejących/funkcjonujących rozwiązań. Trzeci rozdział zawiera sformułowane założenia projektowe, jego wymagania funkcjonalne i niefunkcjonalne. Następnie w rozdziale czwartym zaprezentowano projekt modułu rozszerzonej rzeczywistości, w szczególności przypadki użycia oraz projekt interfejsu. Rozdział piąty to opis procesu implementacji, w którym omówiono wykorzystane technologie oraz opisano poszczególne etapy prac nad projektem. W zakończeniu podsumowane zostały wyniki uzyskane w pracy, a także wskazano możliwe kierunki dalszych prac. Praca kończy się wykazem cytowanej literatury.

1. Wymagania projektowe

1.1. Wprowadzenie

W tym rozdziale zdefiniowano wymagania stawiane aplikacji – Mobilnego przewodnika po Dolnym Śląsku. Przenalizowano również sposób i jakość dostarczania informacji zwiedzającym przez muzea wraz z możliwościami wprowadzenia popraw w tym zakresie.

1.2. Specyfikacja wymagań

Projekt modułu rozszerzonej rzeczywistości dla aplikacji mobilnej, realizowany w ramach niniejszej pracy, jest częścią projektu multimedialnego przewodnika po Dolnym Śląsku. Przewodnik w założeniu ma zawierać zbiór atrakcji turystycznych, które użytkownik będzie mógł poznawać wraz z aplikacją. Głównymi modułami będą:

- Moduł mapy – mapa jako podstawowy widok aplikacji będzie zwierała pineski reprezentujące atrakcje, które można odwiedzić z pomocą aplikacji. Po naciśnięciu na pineskę wyświetlone zostaną szczegóły danej atrakcji;

- Moduł atrakcji – atrakcje będę skategoryzowane według miast, w których się znajdują, oraz według kategorii do jakiej należą. Po wybraniu, którejś z nich ponownie wyświetlone zostaną jej szczegóły, tak jak w przypadku modułu mapy;

- Moduł wycieczek – wewnątrz aplikacji zostaną przygotowane wycieczki tematyczne pozwalające zwiedzać atrakcje w zaplanowany sposób, mają to być głównie trasy piesze, dlatego atrakcje powinny znajdować się blisko siebie, Każda z wycieczek będzie zawierać szacowany czas jej przejścia oraz zwiedzenia każdej z atrakcji;

- Moduł rozszerzonej rzeczywistości – będzie dostępny dla wybranych atrakcji, będą to przede wszystkim takie obiekty, jak muzea i galerie sztuki, ponieważ będzie się on opierał na rozpoznawaniu eksponatów oraz wyświetlaniu w rozszerzonej rzeczywistości informacji o nich.

Do prawidłowego działania wszystkich swoich funkcji, aplikacja wymagała będzie szeregu zgód od użytkownika takich jak:

- zgoda na dostęp do internetu, aby umożliwić połączenie z bazą danych w celu pobrania zawartości aplikacji;
- zgoda na dostęp do lokalizacji użytkownika, która odpowiadać będzie za prawidłowe wyświetlanie atrakcji w jego okolicy. W przypadku, gdy użytkownik nie udzieli tej zgody, sugerowane będą losowe atrakcje;
- zgoda na dostęp do aparatu użytkownika w celu działania modułu rozszerzonej rzeczywistości, w przypadku nieudzielania tej zgody moduł ten nie będzie działał;
- zgoda na wysyłanie powiadomień, dzięki którym użytkownik będzie zachęcany do odwiedzania kolejnych atrakcji.

Wszystkie powyższe zgody, z wyjątkiem dostępu do internetu, będą dobrowolne. Użytkownik będzie mógł korzystać z aplikacji bez ich udzielania, ale wiązało się to będzie z brakiem dostępu do części funkcjonalności wyszczególnionych powyżej.

Szczególne wymagania modułu rozszerzonej rzeczywistości:

- wyświetlanie obiektów wirtualnych na obrazie z kamery w czasie rzeczywistym,
- rozpoznawanie obrazu, które pozwoli na prawidłowe wyświetlanie obiektów rozszerzonej rzeczywistości,
- obiektami wyświetlanymi w rozszerzonej rzeczywistości będą tytuły, opisy lub inne elementy opisujące rozpoznane eksponaty, takie jak obrazy, rzeźby, płaskorzeźby i tym podobne,
- rozszerzona rzeczywistość będzie działała tylko dla wybranych obiektów, których charakter będzie zgodny z sensem jej działania. W szczególności będą to muzea i galerie sztuki, posiadające w swoich zbiorach eksponaty, które będą mogły być skanowane przez aplikację.

Aplikacja będzie skierowana do turystów oraz mieszkańców Dolnego Śląska, którzy chcą lepiej poznać ten malowniczy region Polski, i będzie przeznaczona na telefony komórkowe pracujące na systemie iOS. Będzie mogła zostać udostępniona w sklepie z aplikacjami do darmowego pobrania, najlepszym sposobem jej monetyzacji będzie odpłatne dodawanie nowych atrakcji turystycznych, których administratorzy wyrażą taką chęć. Aplikacja będzie stanowić dla nich platformę reklamową, dzięki której za pomocą odpowiedniego zaprezentowania swojego obiektu, będą mogli skłaniać użytkowników do ich odwiedzenia.

1.3. Dostęp do informacji w muzeach i galeriach sztuki

Przed procesem projektowania aplikacji przeanalizowano, jak muzea dostarczają odwiedzającym informacje o dziełach, które można zobaczyć w trakcie ich zwiedzania.

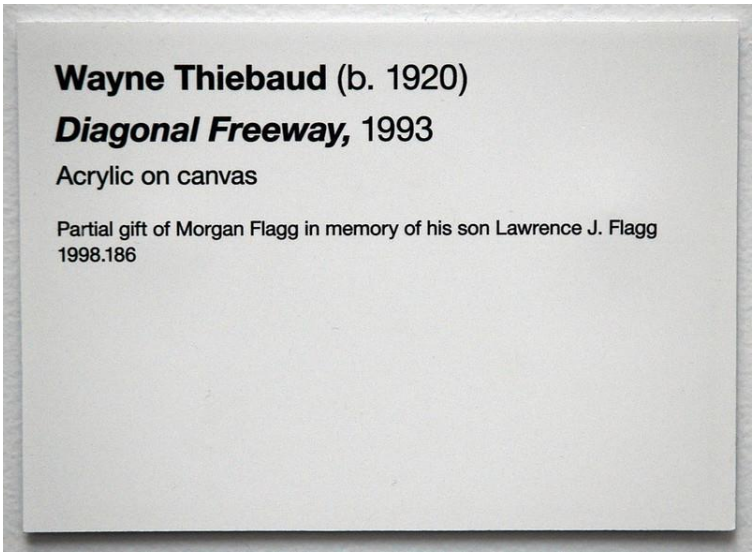
W Muzeum Narodowym w Krakowie tabliczka opisująca eksponat widoczna na rysunku 2. zawiera informację o imionach i nazwiskach autorów oraz latach ich życia, nazwę dzieła, rok bądź lata w jakich dzieło powstało oraz technikę jaką zostało wykonane. Wszystkie informacje są w języku polskim i angielskim.

Rysunek 2: Tabliczka opisująca eksponat w Muzeum narodowym w Krakowie – źródło [10]

Tabliczka opisująca obraz z Muzeum Narodowego we Wrocławiu na rysunku 3. zawiera te same podstawowe informacje, które znajdują się również w Muzeum Narodowym w Krakowie, dodatkowo można z niej odczytać z jakiego zbioru pochodzi eksponat.

Rysunek 3: Tabliczka opisująca eksponat w Muzeum Narodowym we Wrocławiu – źródło własne

Tak krótkie opisy eksponatów to nie tylko domena polskich muzeów. Na rysunku 4. znajduje się opis dzieła sztuki z muzeum Fine Arts Museums of San Francisco, który podobnie jak wcześniej wspomniane, nie zawiera informacji, które pozwoliłby odwiedzającemu muzeum lepiej poznać przedstawiane dzieło.



Rysunek 4: Tabliczka opisująca eksponat w Fine Arts Museums of San Francisco – źródło [11]

Opisy dzieł sztuki udostępnionych zwiedzającym są skąpe i nie są pomocne dla osób, które dopiero zaczęły interesować się sztuką. Warto również zauważyć, że znaczna część obrazów jest uboga w ciekawe informacje nawet na różnych stronach internetowych. Odnalezienie niektórych dzieł i ich interpretacja lub opis tła historycznego, często zajmuje zbyt dużo czasu.

Jednym z możliwych rozwiązań braku informacji na tabliczkach, jest umieszczenie przy eksponatach kodów QR, które pozwalają wejść na dedykowaną stronę internetową zawierającą więcej informacji. Kody są zazwyczaj bardzo małe, aby nie naruszać wyrazu artystycznego miejsca, w którym się znajdują, wymusza to na odwiedzającym podejście bardzo blisko w celu zeskanowania, co ogranicza liczbę osób, które jednocześnie mogą dany

kod skanować. W skrajnych sytuacjach może to prowadzić do tworzenia się kolejek, co na pewno zniechęci część osób do zeskanowania kodu i pozyskania informacji o eksponacie, przez co nie zdołają oni dowiedzieć się nic więcej o dziełach i przez to zmniejszy się szansa na ich trwałe zainteresowanie sztuką.

Rysunek 5: Tabliczka opisująca eksponat w Muzeum Miasta Łodzi – źródło 12

Podstawową wadą klasycznych tabliczek przy eksponatach jest bardzo wąski zakres informacji, które dostarczają. Jednym z rozwiązań tego problemu może być umieszczania na nich kodów QR, jak na rysunku 5., te jednak mają swoje wady, dlatego optymalnym rozwiązaniem wydaje się być zastosowanie rozszerzonej rzeczywistości, która mogłaby po rozpoznaniu eksponatu, wyświetlić informacje na ekranie urządzenia mobilnego – telefonu lub tabletu. Zastosowanie tej technologii daje również możliwość przejścia do dedykowanego ekranu opisującego eksponat w klasycznej formie, tak jak w przypadku identyfikowania obrazów po kodach QR, jednak bez konieczności ich umieszczania w muzeach

2. Stan wiedzy i techniki w zakresie programowania aplikacji mobilnych ze szczególnym uwzględnieniem rozszerzonej rzeczywistości

2.1. Wprowadzenie

W rozdziale tym zostaną porównane różne technologie pozwalające na implementacje aplikacji mobilnych, w szczególności tych wykorzystujących rozszerzoną rzeczywistość. Następnie omówiona szczerzej zostanie sama rozszerzona rzeczywistość oraz zostaną przeanalizowane dostępne aplikacje, które ją wykorzystują.

2.2. Aplikacje mobilne

Implementacje aplikacji mobilnych można podzielić na podstawie platform, na które powstają. Aplikacje pisane na tylko jeden z systemów to aplikacje natywne. Alternatywą dla nich są aplikacje wieloplatformowe, które raz napisane mogą zostać skompilowane na wiele systemów.

2.2.1. Podejście natywne

Aplikacje natywne są pisane z myślą tylko o jednym systemie operacyjnym. Ponad 99% [13] sprzedawanych w 2020 roku smartfonów posiadało system iOS [14] lub Android [15], dlatego analizie poddane zostaną tylko te dwa systemy.

Do programowania natywnego na platformę iOS służy środowisko programistyczne Xcode [16], a językami do tego wykorzystywanymi są starszy Objective-C [17] oraz nowszy, wydany w 2014 roku, Swift [18]. Powstał on na potrzeby firmy Apple, co sprawia, że jest bardzo dobrze zoptymalizowany pod kątem tworzenia aplikacji mobilnych. Dużą zaletą pracy jednej firmy zarówno nad systemem jak i językiem, w którym powstają aplikacje na ten system, jest spójność tych dwóch bytów, nowe funkcjonalności telefonów od razu są wspierane przez Swifta. Jest on również prostszy w nauce, popularniejszy oraz zauważalnie szybszy od Objective-C [18].

Najpopularniejszymi językami do programowania aplikacji mobilnych są Kotlin [19] oraz Java [20], środowiskiem pracy jest Android Studio [21]. Java jest bardzo uniwersalnym językiem oraz jednym z najpopularniejszych na świecie [22], co sprawia, że dla wielu programistów rozpoczęcie pracy z systemem Android jest stosunkowo proste. Jednakże od powstania Kotlina w 2011 roku ten nieustannie się rozwija, aplikacjom w nim napisanym rzadziej zdarzają się błędy powstałe nie z winy programisty. Ponad 80% z top 1000 aplikacji w Sklepie Play [23] została napisana w Kotlinie, co świadczy o tym, że według najpopularniejszych firm wytwarzających aplikacje na system operacyjny firmy Google, język ten jest lepszym rozwiązaniem.

2.2.2. Podejście wieloplatformowe

Aplikacje wieloplatformowe pisane są z myślą zarówno o systemie iOS jak i Android, co skraca czas implementacji, ponieważ nie wymaga dwukrotnego pisania aplikacji na różne systemy. Najpopularniejszymi technologiami opartymi na podejściu wieloplatformowym są Flutter [24] oraz React Native [25]. Te dwa frameworki są używane przez 80% programistów implementujących wieloplatformowe aplikacje mobilne. Pierwszy z nich został wskazany przez 42% ankietowanych, drugi zaś przez 38% [26].

Flutter jest rozwiązaniem firmy Google, które wykorzystuje język Dart [27]. Dla wielu może to być minusem, ponieważ język ten nie jest powszechnie używany poza tą technologią. Jego zaletami są wysoka wydajność oraz możliwość podglądu zmian w czasie rzeczywistym, wadami są duża waga aplikacji oraz fakt, że sam jego zamysł nie pozwala na bezproblemowe udostępnianie aplikacji w sklepie z aplikacjami firmy Apple. Dzieje się tak, ponieważ domyślnie powstające w nim projekty nie spełniają w pełni wymagań Human Interface Guidelines [28], czyli instrukcji firmy Apple jak powinny wyglądać aplikacje wydawane na ich system. Niespełnienie ich w pełni, może skutkować odmową wprowadzenia aplikacji do sklepu.

React Native to technologia firmy Meta (do niedawna Facebook). Jego główną zaletą jest fakt, że aplikacje w niej implementowane wyglądem niczym się nie różnią w stosunku do aplikacji natywnych, zarówno, iOS jak i Andorid. Technologia ta jest przystępna, jeśli chodzi o rozpoczęcie w niej pracy, ponieważ wykorzystuje język JavaScript [29], który zajmuje pierwsze miejsce w rankingu najpowszechniej używanych języków programowania [22].

2.3. Przegląd przydatnych technik i technologii pozwalających na implementację rozszerzonej rzeczywistości

2.3.1. Technologie firmy Apple

ARKit [30] to framework, umożliwiająca implementacje aplikacji w oparciu o rozszerzoną rzeczywistość - „ARKit łączy śledzenie ruchu urządzenia, przechwytywanie scen z kamery, zaawansowane przetwarzanie scen i udogodnienia wyświetlania, aby uprościć tworzenie doświadczeń AR. Dzięki tym technologiom można tworzyć wiele rodzajów doświadczeń AR za pomocą przedniej lub tylnej kamery urządzenia z systemem iOS.” [30]. Powstał z myślą o wyświetlaniu obiektów w rozszerzonej rzeczywistości, z naciskiem na integracje ich ze światem rzeczywistym. Posiada mocno rozbudowane funkcje rozpoznawania obrazu oraz skanowania trójwymiarowego, jednakże jego główne zadanie to wyświetlanie obrazów płaskich, dwuwymiarowych. W jego najnowszej, piątej wersji wydanej w 2021 roku, postawiono nacisk na rozpoznawanie twarzy. ARKit jest podstawą rozszerzonej rzeczywistości w rozwiązaniach na platformę iOS, odpowiada za wszelką integrację pomiędzy światem rzeczywistym, a tym co jest na niego nanoszone na ekranie telefonu.

RealityKit [31] powstał z myślą o rozszerzeniu funkcjonalności wspomnianego wcześniej ARKit. RealityKit służy do implementacji wydajniejszych rozwiązań symulujących oraz renderujących obiekty trójwymiarowe, umożliwia również dokładne skanowanie przedmiotów i późniejsze ich wykorzystanie wewnątrz aplikacji. Ciekawym rozwiązaniem, które pozwala on zastosować, jest synchronizacja wielu urządzeń, która umożliwia grupowe doświadczanie rozszerzonej rzeczywistości.

SceneKit [32] podobnie jak RealityKit, umożliwia prace na bazie danych z ARKit, jednakże jest prostszy i mniej zaawansowany, pozwala na wyświetlanie trójwymiarowych obiektów w ramach sceny, na której zawartość może być poruszana oraz animowana.

Reality Composer [33] to aplikacja umożliwiająca opracowywanie modeli trójwymiarowych oraz animowanie ich w celu późniejszego wykorzystania tych prefabrykatów w ARKit lub RealityKit.

2.3.2. Technologie innych firm

ARCore [34] - w przypadku swojej technologii do rozszerzonej rzeczywistości, Google postanowiło nie rozbijać swoich rozwiązań na poszczególne frameworki, tak jak zrobiło to Apple, tylko postawiło na jedno kompleksowe rozwiązanie. Według Google: „ARCore ma dwie podstawowe funkcjonalności: śledzi pozycję urządzenia mobilnego podczas jego ruchu i analizuje świat rzeczywisty. Technologia ARCore wykorzystuje aparat telefonu do identyfikowania interesujących punktów i monitoruje, jak te punkty poruszają się w czasie. Dzięki połączeniu ruchu tych punktów i odczytów z czujników telefonu, ARCore określa zarówno położenie, jak i orientację telefonu podczas poruszania się." [34] Główną cechą ARCore jest to, czym w systemie firmy Apple jest ARKit, czyli rejestrowanie świata rzeczywistego i umożliwianie wyświetlania na ekranie telefonu obiektów, które będą ten świat rozszerzać. Technologia ta może być używana zarówno na telefonach z systemem Android jak i iOS.

AR Foundation [35] to technologia rozszerzonej rzeczywistości od firmy Unity, która jest właścicielem silnika do tworzenia gier pod tą samą nazwą. Jej domyślną funkcjonalnością są właśnie gry, co jest jego wadą przy tworzeniu innych aplikacji. Jednak największą wadą jest fakt, że do prawidłowego działania wymaga pracy z wcześniej wspomnianymi ARKit lub ARCore, nie jest więc w stanie pracować samodzielnie. Zastosowanie tej technologii wiązałoby się potencjalnie z większym nakładem pracy.

2.4. Różnice pomiędzy rozszerzoną a wirtualną rzeczywistością

Są to dwie zbliżone do siebie technologie, obie u podstawy mają generowanie obrazu, jednak mają wiele różnic takich jak potrzebny do ich działania sprzęt czy zastosowania.

„Rozszerzona rzeczywistość łączy w sobie trójwymiarowe obiekty i tekst generowane komputerowo, z realnym obrazem w czasie rzeczywistym." [36] Jej podstawową cechą jest nierozerwalne połączenie z realnym światem, to on jest sceną, na której pojawia się obraz, którym może być dowolny obiekt. Przedmioty widoczne jako trójwymiarowe, mogą zostać w takim trybie bez większych trudności dokładnie zbadane, najczęściej zostają umieszczane nieruchomo w przestrzeni, a wokół nich porusza się użytkownik obserwujący je przez odpowiednie urządzenie. W przypadku tekstów, obrazów lub innych dwuwymiarowych obiektów, najczęściej służą one do opisania, zaznaczenia lub uatrakcyjnienia czegoś co znajduje się w świecie rzeczywistym.

Termin wirtualna rzeczywistość został wymyślony przez Jarona Laniera, który zdefiniował go jako „generowane komputerowo, interaktywne, trójwymiarowe środowisko, w którym zanurzona jest osoba" [37]. Nie ma związku z przestrzenią w jakiej znajduje się użytkownik, zazwyczaj jest od niej całkowicie odcięty. Podstawowym sprzętem do rozszerzonej rzeczywistości są gogle VR, które nie pozwalają zobaczyć niczego poza światem wirtualnym.

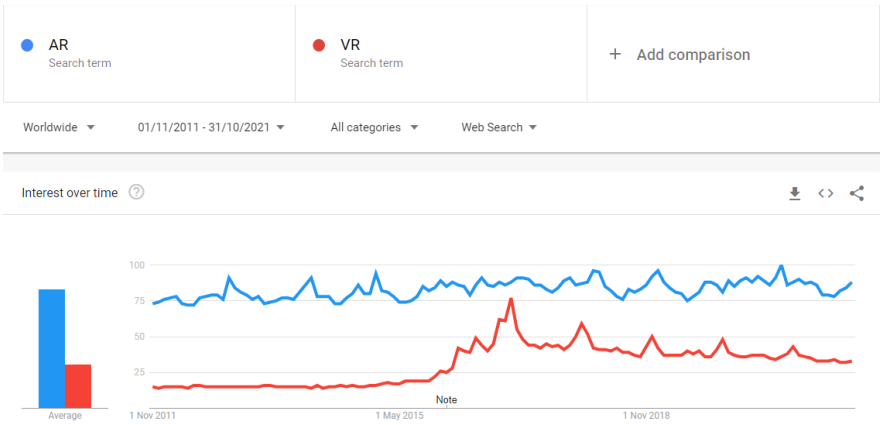
„Główną różnicą między wirtualną, a rozszerzoną rzeczywistością jest to, że ta druga wykorzystuje prawdziwe obrazy, klatki wideo i grafikę 3D." [36], podczas gdy wirtualna rzeczywistość nie wykorzystuje prawdziwych obrazów, wszystko jest generowane

komputerowo. W tabeli 1. przedstawiono porównanie rozszerzonej i wirtualnej rzeczywistości.

Tabela 1: Porównanie rozszerzonej i wirtualnej rzeczywistości – opracowaie na podstawie [38],[39]

	Rozszerzona rzeczywistość	**Wirtualna rzeczywistość**
Co wchodzi w skład obrazu	Obraz ze świata rzeczywistego z naniesionymi na niego obiektami wirtualnymi	Całość jest wirtualna
Minimalny potrzebny sprzęt	Urządzenie z kamerą i wyświetlaczem	Gogle wirtualnej rzeczywistości
Sprzęt opcjonalny	Słuchawki	Słuchawki, specjalne kontrolery
Perspektywa użytkownika	Użytkownik widzi świat rzeczywisty oraz wyświetlany obiekt	Użytkownik widzi jedynie wygenerowany obraz świata wirtualnego
Zastosowanie	Aplikacje mobilne, gry, edukacja,	Gry, przemysł, szkolenia, edukacja

Na rysunku 6. obrazującym częstotliwość wyszukiwania w serwisie google.com na całym świecie w okresie październik 2011 – listopad 2021, można zauważyć zdecydowaną przewagę, jeśli chodzi o zainteresowanie rozszerzoną rzeczywistością w stosunku do wirtualnej rzeczywistości. Technologia wirtualnej rzeczywistości drastycznie zyskała na popularności na początku 2016 roku i zapewne ma to związek z wypuszczeniem na rynek dużej liczby gogli VR, takich jak Oculus Rift w marcu [40], HTC Vive w kwietniu, PlayStation VR w październiku [41]. VR piki swojej popularności osiągał w grudniu co najprawdopodobniej jest związane z okresem świątecznym i może być spowodowane częstymi zakupami gogli VR jako prezent dla graczy. Na podstawie tych domysłów można wysnuć tezę, iż głównym rynkiem rozszerzonej rzeczywistości jest rynek gier mobilnych.

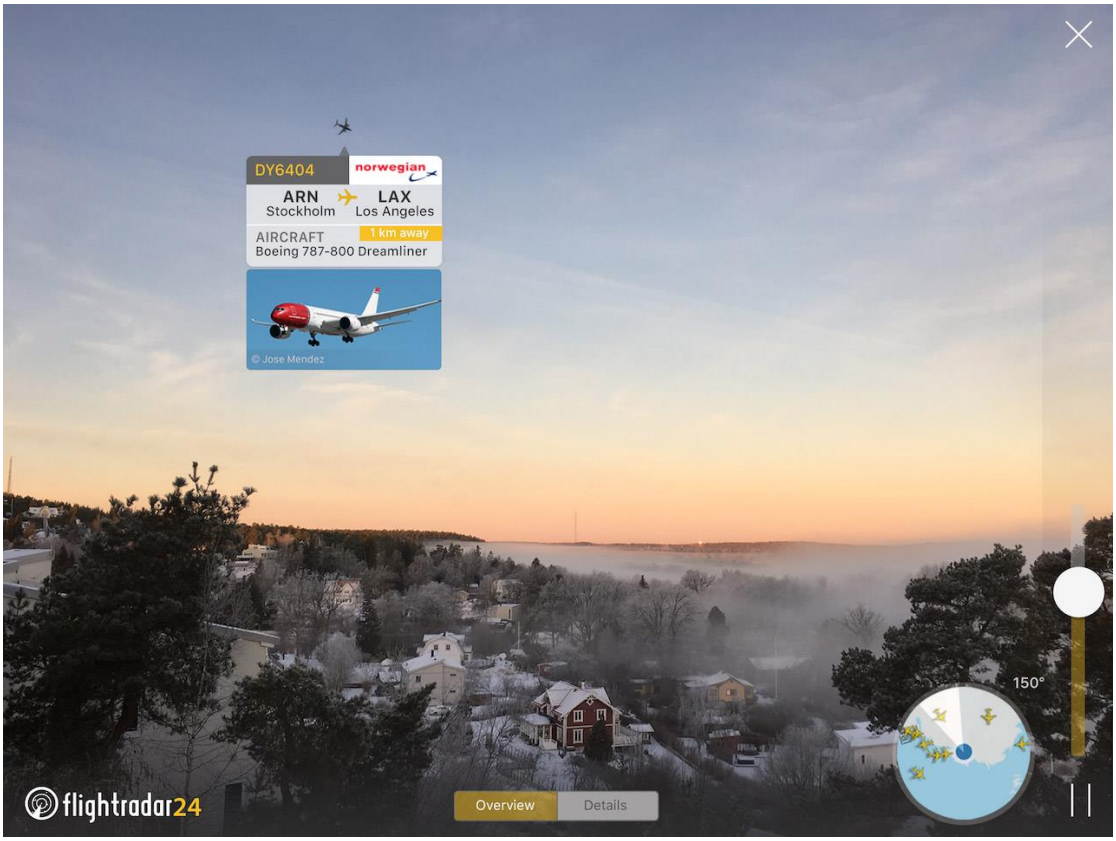


Rysunek 6: Porównanie częstotliwości wyszukiwania haseł AR i VR w wyszukiwarce Google w okresie 11.2011-10.2021 - źródło [42]

2.5. Przegląd rozwiązań wykorzystujących rozszerzoną rzeczywistość

Przed rozpoczęciem związanych z projektowaniem aplikacji, przeanalizowano szereg aplikacji mobilnych, które wykorzystują rozszerzoną rzeczywistość. Starano się dostrzec mocne i słabe strony wszystkich aplikacji by zaprojektować jak najlepszy produkt. Podsumowanie niektórych z przeanalizowanych aplikacji poniżej.

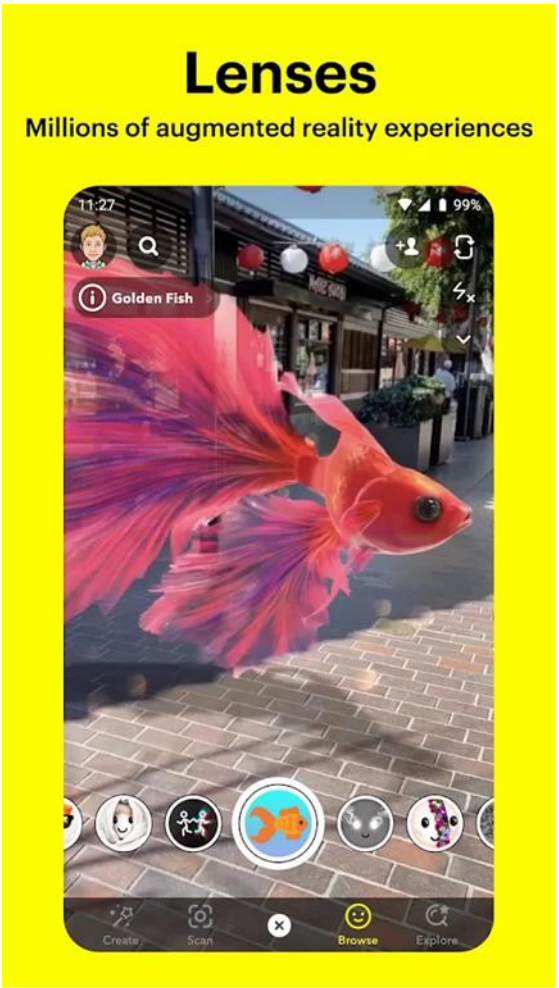
Flightradar24 to popularna aplikacja do śledzenia samolotów w czasie rzeczywistym, została przedstawiona na rysunku 7. Posiada opcję rozszerzonej rzeczywistości, która na bazie lokalizacji użytkownika oraz danych z żyroskopu telefonu, wskazuje w rozszerzonej rzeczywistości samoloty, helikoptery oraz inne statki powietrzne. Na ekranie telefonu widoczny jest podgląd z kamery oraz naniesione na niego podstawowe dane o obiektach będących w polu jej widzenia. Po naciśnięciu na baner odnoszący się do danego lotu, użytkownik przekierowywany jest do widoku aplikacji, który wyświetla wszystkie możliwe informacje. To rozwiązanie wydaje się bardzo dobrym, ponieważ w rozszerzonej rzeczywistości widać tylko małe banery, a te są czytelne nawet przy wielu widocznych na ekranie samolotach. Gdyby próbowano wyświetlić wszelkie możliwe informacje, widok ten byłby bardzo nieprzejrzysty.



Rysunek 7: Zrzut ekranu z aplikacji Flightradar24 – źródło 43

Snapchat to jeden z najpopularniejszych mediów społecznościowych posiadający ponad 250 milionów aktywnych użytkowników [4], opiera się na przesyłaniu między użytkownikami zdjęć lub filmów. Od 2015 roku aplikacja wykorzystuje rozszerzoną rzeczywistość do modyfikowania przesyłanych przez nią multimediów. Dzięki wykorzystaniu AR użytkownicy mają możliwość w prosty sposób modyfikować swoje zdjęcia i filmy. Obraz tych zmian widziany jest w czasie rzeczywistym na podglądzie z kamery, użytkownik może zmieniać filtry i wybrać ten który sprawi, że będzie on zadowolony z efektów. Aplikacja została przedstawiona na rysynkach 8. oraz 9.

Rysunek 8: Zrzut ekranu z aplikacji Snapchat - źródło [44]



Rysunek 9: Zrzut ekranu z aplikacji Snapchat - źródło [44]

Pokemon GO to gra, która na telefonach komórkowych zadebiutowała w 2016 roku. Wykorzystuje rozszerzoną rzeczywistość do mechaniki łapania tytułowych pokemonów, co zostało przedstawione na rysunku 10.

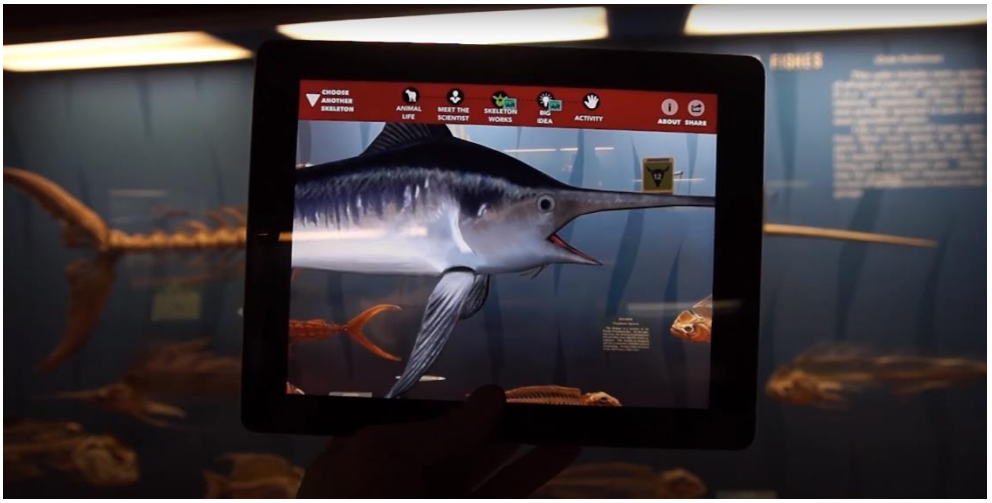
Pierwszą czynnością, jaką użytkownik robi w grze, to stworzenie swojego wirtualnego awatara. Po dokonaniu tego kroku może cieszyć się nią w pełni. Za pomocą GPS'a aplikacja lokalizuje gracza i umieszcza go na wirtualnej mapie, będącej otworzeniem rzeczywistego otoczenia. Następnie pokazuje wygenerowane wcześniej miejsca, do których należy się udać, aby złapać pokemony. Po dotarciu do danej lokalizacji gracz kieruje kamerę i szuka stworzenia. Gdy tego dokona, za pomocą ruchu palcem po ekranie, imituje rzucanie tzw. Pokè Balla. Zebrane Pokemony można wystawiać na arenach (również zlokalizowanych w realnym świecie) i konkurować z innymi graczami.

Gra została bardzo dobrze odebrana przez użytkowników, ponieważ zachęcała do aktywnego trybu życia. W świecie rzeczywistym w miejscach, gdzie umieszczone były areny, spotykało się wielu ludzi dzielących wspólne zainteresowanie światem Pokemonów.

Rysunek 10: Zrzuty ekranu z aplikacji Pokemon Go – źródło [45]

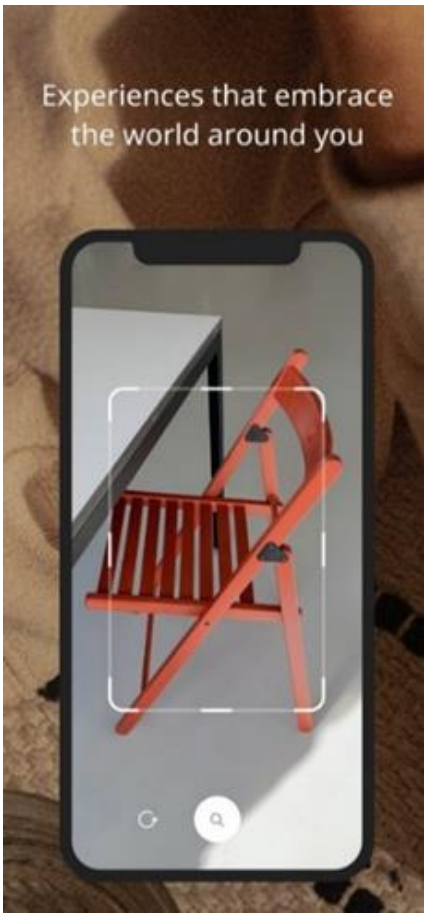
Skin & Bones to aplikacja mobilna, która powstała, aby uatrakcyjnić zwiedzanie sali kości w Narodowym Muzeum Historii Naturalnej w instytucie Smithsonian w Waszyngtonie. Umożliwia nałożenie na zgromadzone w muzeum szkielety zwierząt ich ciał jako trójwymiarowych modeli, mechanika ta widoczna jest na rysunkach 11. oraz 12. Aplikacja ta jest w swoim zamyśle podobna do modułu rozszerzonej rzeczywistości Multimedialnego przewodnika po Dolnym Śląsku, ponieważ jej zadaniem jest uatrakcyjnienie procesu zwiedzania muzeum poprzez dostarczanie dodatkowych informacji o eksponatach, robiąc to za pośrednictwem rozszerzanej rzeczywistości.

Rysunek 11: Zdjęcie eksponatu z Narodowego Muzeum Historii Naturalnej w instytucie Smithsonian - źródło [46]



Rysunek 12: Zdjęcie tabletu, na którym wyświetlana jest aplikacja Skin & Bones - źródło [46]

IKEA Place umożliwia klientom tej szwedzkiej sieci sklepów sprawdzenie, jak wybrane przez nich meble będą wyglądały w ich domach, zostało to przedstawione na rysunkach 13. oraz 14. Aplikacja w ramach rozszerzonej rzeczywistości umieszcza we wskazanym miejscu trójwymiarowy model wybranego mebla, który automatycznie skalowany jest do rozmiarów rzeczywistych. Na ekranie można jednocześnie umieścić wiele przedmiotów, co pozwala zobaczyć w pełni umeblowane pomieszczenie. Wszystkie przedmioty posiadają indywidualny link do strony sklepu, gdzie można dokonać zakupu.



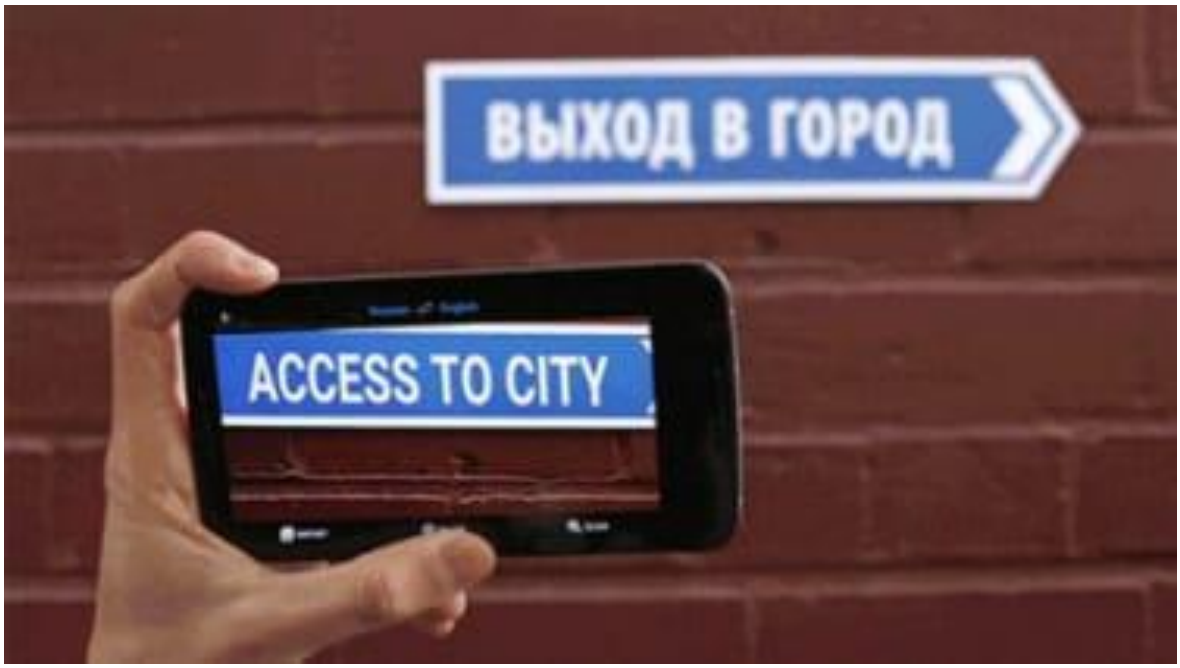
Rysunek 13: Zrzut ekranu aplikacji IKEA Place – źródło [47]

Rysunek 14: Zrzut ekranu aplikacji IKEA Place – źródło [47]

Aplikacja Roomle jest bliźniaczo podobna do aplikacji Ikei, oferuje jednak również funkcje planowania pomieszczeń bez rozszerzonej rzeczywistości. Dużą zaletą jest fakt niepowiązania aplikacji z żadnym producentem lub sklepem, dzięki czemu posiada znacznie większą bazę produktów, niestety bez żadnych odnośników umożliwiających ich zakupienie. Jednak większość z nich łatwo odnaleźć w internecie. Działanie aplikacji zaprezentowano na rysunku 15.

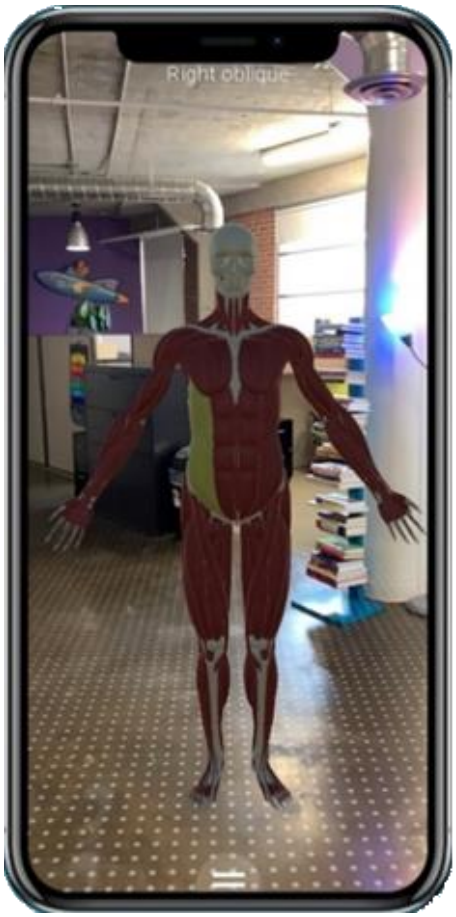
Rysunek 15: Zdjęcie tabletu z uruchomioną aplikacją Roomle – źródło [48]

Tłumacz Google posiada funkcje tłumaczenia tekstów znajdujących się w świecie rzeczywistym za pomocą rozszerzonej rzeczywistości. Skanuje tekst znajdujący się w podglądzie z kamery i na bieżąco tłumaczy go, wyświetlając w jego miejsce tekst w języku wybranym przez użytkownika. Wyświetlane tłumaczenie zachowuje styl oryginalnego napisu poprzez dobranie podobnej wielkości i koloru czcionki, co ułatwia poprawne interpretowanie tłumaczenia. Działanie aplikacji przedstawia rysunek 16. Opcja ta najlepiej sprawdza się do tłumaczenia krótkich zwrotów, raczej nie nadaje się do czytania długich teksów, ponieważ gdy telefon nie jest trzymany w bezruchu, tłumaczenie może znikać i pojawiać się w nieco innym formacie, co znacząco zmniejsza komfort użytkowania.

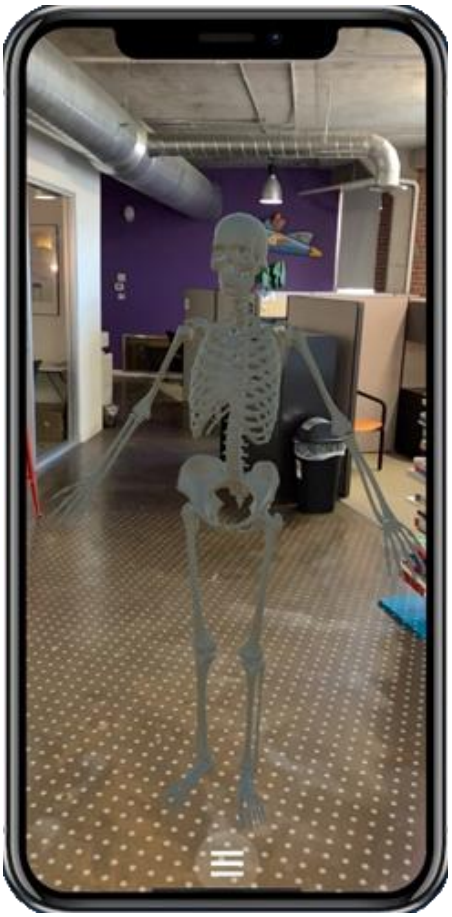


Rysunek 16: Zdjęcie telefonu z uruchomioną aplikacją Tłumacz Google – źródło [49]

Anatomy AR+ to aplikacja skierowana do osób chcących dobrze opanować anatomię człowieka. Pozwala wybrać spośród trzech trybów: mięśnie, szkielet lub mięśnie i szkielet. Dotykając model w wybrane miejsce, można w łatwy sposób sprawdzić nazwę wybranego elementu anatomicznego. Na ekranie wyświetlony zostaje realistyczny model człowieka co widać na rysunkach 17. oraz 18. Po kliknięciu w wybrany element anatomiczny ukazuje się nazwa danego elementu. Jest to idealna aplikacja dla studentów medycyny oraz pasjonatów biologii.



Rysunek 17: Zrzut ekranu aplikacji Anatomy AR+ – źródło [50]



Rysunek 18: Zrzut ekranu aplikacji Anatomy AR+ – źródło [50]

3. Założenia projektowe

3.1. Wprowadzenie

W ramach tego rozdziału przedstawiono wymagania funkcjonalne oraz niefunkcjonalne aplikacji, ze szczególnym uwzględnieniem modułu rozszerzonej rzeczywistości. Opisano również sposób realizacji oraz wzorzec projektowy użyty w procesie implementacji.

3.2. Wymagania funkcjonalne

Sformułowane wymagania funkcjonalne:

1. Aplikacja będzie automatycznie rozpoznawała eksponat i wyświetlała podstawowe informacje o nim w rozszerzonej rzeczywistości.
2. Z ekranu rozszerzonej rzeczywistości użytkownik będzie miał możliwość przejścia do pełnych informacji o eksponacie, który jest aktualnie rozpoznany.
3. W ramach rozszerzonej rzeczywistości będzie możliwość wyświetlania informacji o wielu eksponatach w danym momencie.
4. Po rozpoznaniu eksponatu i dotknięciu jego wizualizacji na ekranie lub dowolnego elementu stanowiącego jego opis, wyświetlony zostanie widok eksponatu, przedstawiający jego zdjęcie oraz pełen opis.
5. Po naciśnięciu na zdjęcie eksponatu z poziomu widoku eksponatu, zdjęcie te zostanie wyświetlone w trybie pełnoekranowym, z możliwością przybliżania przez użytkownika.

3.3. Wymagania niefunkcjonalne

Sformułowane wymagania niefunkcjonalne:

1. Aplikacja będzie wymagała dostępu do aparatu, w przeciwnym razie moduł rozszerzonej rzeczywistości nie będzie działał.
2. Aplikacja nie będzie wykorzystywała lampy błyskowej aparatu telefonu, ponieważ w przypadku niektórych dzieł sztuki jej działanie może doprowadzić do uszkodzenia eksponatu.
3. Aplikacja będzie dostępna w języku polskim, jednak zostanie przygotowana w taki sposób by w przyszłości można było ją rozbudować o wsparcie dla innych języków.
4. Aplikacja będzie działać na telefonach z systemem iOS w wersji 14 lub wyższej.
5. Aplikacja zostanie napisana w języku Swift i będzie aplikacją natywną na platformę iOS.
6. Aplikacja będzie spełniała wymagania firmy Apple stawiane aplikacjom publikowanym w ich sklepie z aplikacjami App Store.

3.4. Sposób realizacji

Aplikacja będzie aplikacją mobilną na platformę iOS. Zostanie zaimplementowana w języku Swift i będzie aplikacją natywną. Wykonana zostanie za pomocą środowiska programistycznego Xcode. Do rozpoznawania obrazu oraz realizacji funkcji rozszerzenie rzeczywistości zostanie użyta biblioteka ARKit.

3.5. Wzorzec architektoniczny

Moduł rozszerzonej rzeczywistości, tak jak cała aplikacja multimedialnego przewodnika po Dolnym Śląsku, został zaimplementowany w oparciu o wzorzec MVVM [51], który w swoim założeniu dzieli aplikację na trzy podstawowe warstwy,

- pierwszą z nich jest model, który reprezentuje wszystkie obiekty,
- drugą widok (ang. view), który zawiera interfejs użytkownika,
- trzecią jest model widoku (ang. view-model), który jest połączeniem pomiędzy modelem oraz widokiem, jego zadaniem jest wymiana danych pomiędzy nimi.

Głównymi zaletami wzorca MVVM, są:

- Możliwość ponownego wykorzystania kodu – przez fakt rozdzielenia kodu na warstwy istnieje większe prawdopodobieństwo, że część kodu będzie można użyć ponownie;
- Łatwość testowania testami jednostkowymi – warstwy modelu i modelu widoku mogą zostać przetestowane w oderwaniu od interfejsu;
- Modyfikowalność interesu – warstwa wizualna może zostać przeprojektowana, a w jej nowej wersji można zastosować te same funkcje z warstwy modelu widoku;
- Możliwość niezależnej pracy nad warstwą wizualną oraz logiką biznesową – oddzielni deweloperzy mogą pracować nad różnymi warstwami aplikacji, dzięki ustandaryzowanemu podziałowi pomiędzy ich obowiązkami;

4. Projekt modułu rozszerzonej rzeczywistości

4.1. Wprowadzenie

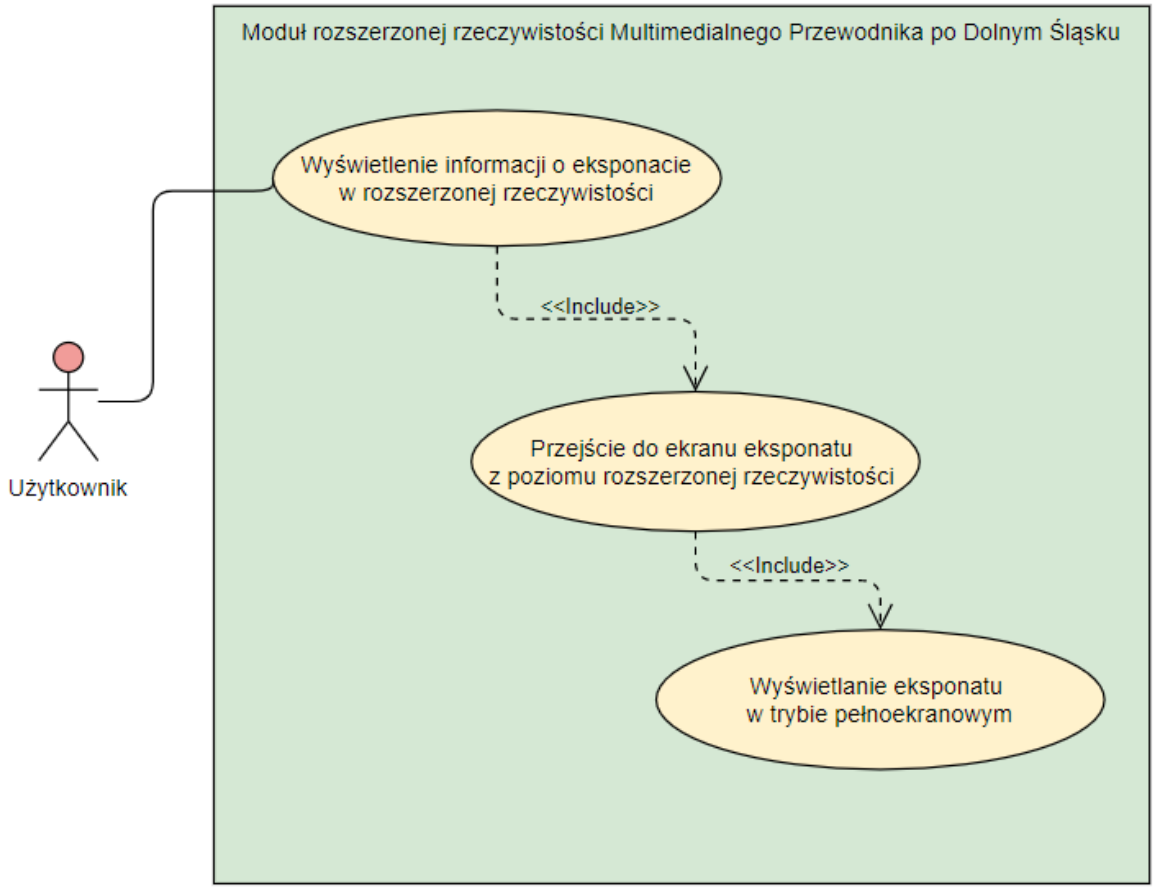
Rozdział zawiera opis projektowania interfejsu modułu rozszerzonej rzeczywistości. Zdefiniowano również jego funkcjonalności, które zostały przedstawione w postaci przypadków użycia.

4.2. Przypadki użycia

Zdefiniowano trzy przypadki użycia

- Wyświetlenie informacji o eksponacie w rozszerzonej rzeczywistości,
- Przejście do ekranu eksponatu z poziomu rozszerzonej rzeczywistości,
- Wyświetlanie eksponatu w trybie pełnoekranowym.

Ich zależności przedstawiono na diagramie przypadków użycia – rysunek 19.



Rysunek 19: Diagram przypadków użycia - opracowanie własne

Przypadki użycia zostały szczegółowo opisane w tabelach 2. – 4. Dla każdego z nich zdefiniowano aktora, warunki początkowe, krótki opis, ścieżkę główną, w dwóch przypadkach ścieżki alternatywne oraz warunki końcowe.

Tabela 2: PU-1 - Wyświetlenie informacji o eksponacie w rozszerzonej rzeczywistości - opracowanie własne

Lp.	PU-1
Nazwa	Wyświetlenie informacji o eksponacie w rozszerzonej rzeczywistości
Aktor	Użytkownik
Warunki początkowe	Użytkownik fizycznie znajduje się w atrakcji turystycznej posiadającej eksponaty, których dane znajdują się w bazie danych aplikacji
Opis	Użytkownik chce zobaczyć informacje o interesującym go eksponacie
Ścieżka główna	1. Użytkownik z poziomu aplikacji przechodzi do atrakcji turystycznej, w której się znajduje. 2. Użytkownik naciska przycisk odkryj. 3. Aplikacja uruchamia tryb rozszerzonej rzeczywistości. 4. Użytkownik ustawia urządzenie tak, aby w podglądzie kamery znajdował się eksponat. 5. Aplikacja rozpoznaje eksponat i wyświetla informacje o nim w trybie rozszerzonej rzeczywistości.
Ścieżka alternatywna	2a. Aplikacja nie wyświetla przycisku odkryj, ponieważ dana atrakcje turystyczna nie wspiera trybu rozszerzonej rzeczywistości.
Stan końcowy	Wyświetlanie informacji o eksponacie w trybie rozszerzonej rzeczywistości.

Tabela 3: PU-2 – Przejście do ekranu eksponatu z poziomu rozszerzonej rzeczywistości - opracowanie własne

Lp.	PU-2
Nazwa	Przejście do ekranu eksponatu z poziomu rozszerzonej rzeczywistości
Aktor	Użytkownik
Warunki początkowe	Aplikacja jest w trybie rozszerzonej rzeczywistości
Opis	Użytkownik chce przejść do widoku zawierającego szczegóły eksponatu, który został rozpoznany
Ścieżka główna	1. Użytkownik kieruje urządzenie w stronę eksponatu. 2. Aplikacja rozpoznaje eksponat i wyświetla informacje o nim w trybie rozszerzonej rzeczywistości. 3. Użytkownik dotyka na ekranie telefonu eksponatu lub dowolnego elementu rozszerzonej rzeczywistości z nim powiązanego. 4. Aplikacja wyświetla ekran eksponatu z dostępnymi informacjami.
Ścieżka alternatywna	-
Stan końcowy	Wyświetlanie ekranu eksponatu

Tabela 4: PU-3 – Wyświetlanie zdjęcia eksponatu w trybie pełnoekranowym - opracowanie własne

Lp.	PU-3
Nazwa	Wyświetlanie eksponatu w trybie pełnoekranowym
Aktor	Użytkownik
Warunki początkowe	Aplikacja jest otwarta na ekranie eksponatu
Opis	Użytkownik
Ścieżka główna	1. Użytkownik naciska na zdjęcie eksponatu. 2. Aplikacja otwiera zdjęcie w trybie pełnoekranowym
Ścieżka alternatywna	3a. Użytkownik wykonuje dotykając ekranu gest oddalenia od siebie dwóch palców. 4a. Aplikacja przybliża zdjęcie zgodnie z gestem użytkownika. 3b. Użytkownik wykonuje dotykając ekranu gest przybliżenia do siebie dwóch palców. 4b. Aplikacja oddala zdjęcie zgodnie z gestem użytkownika.
Stan końcowy	Wyświetlenie zdjęcia eksponatu w trybie pełnoekranowym z możliwością zmiany przybliżenia

4.3. Projekt interfejsu

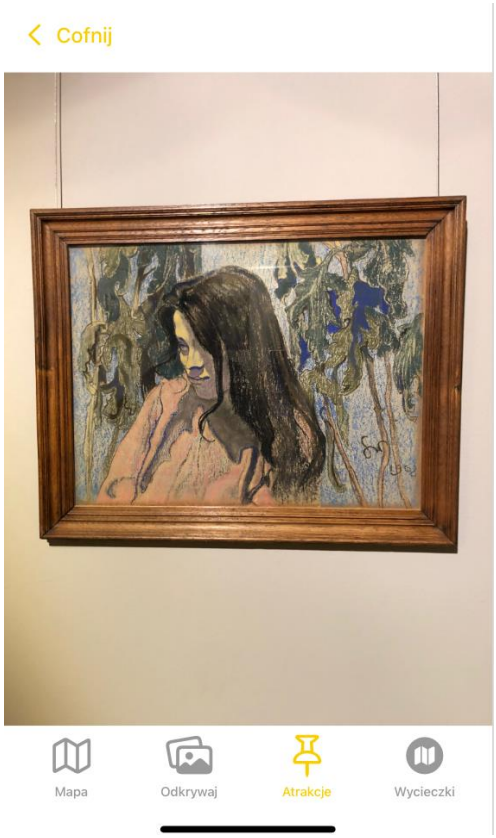
Przed rozpoczęciem procesu implementacji w ramach prac przygotowawczych wykonano projekt interfejsu aplikacji przy użyciu programu Adobe XD. Przy opracowywaniu warstwy wizualnej dużą uwagę poświęcono ogólnej czytelności przedstawianych informacji, ponieważ produkt finalny powinien być jak najbardziej przystępny dla użytkownika.

Na rysunku 20. zaprezentowano planowany widok rozszerzonej rzeczywistości – większość ekranu będzie stanowił podgląd z kamery urządzenia.

Po wykryciu jednego z szukanych obrazów zostaną pod nim wyświetlone informacje o nazwie dzieła, jego autorze oraz krótki opis. Zostało to przedstawione na rysunku 21.

Rysynek 22. przedstawia projekt interfejsu widoku szczegółów eksponatu, widok ten zawiera dodatkowe informacje w stosunku do widoku rozszerzonej rzeczywistości, w trakcje jego projektowania skupiono się na prostocie i przejrzystości.

Ostatnim zaprojetkowanym widokiem jest widok pełnoekranowego podglądu zdjęcia eksponatu, przedstawiony na rysunkach 23. oraz 24., jest on zdecydowanie najprostszym z widoków aplikacji. Założeniem było, aby użytkownik miał możliwość dowolnie przybliżyć zdjęcie i dostrzec w ten sposób wszystkie szczegóły obrazu, które mogłyby być niewidoczne gołym okiem.

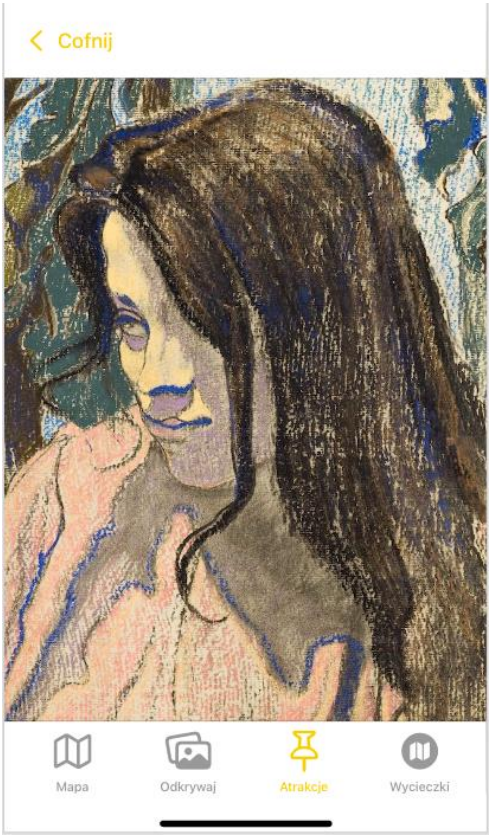


Rysunek 20: Projekt interfejsu – widok rozszerzonej rzeczywistości przed rozpoznaniem obrazu – źródło własne

Rysunek 21: Projekt interfejsu – widok rozszerzonej rzeczywistości po rozpoznaniu obrazu – źródło własne

Rysunek 22: Projekt interfejsu – widok szczegółów eksponatu – źródło grafiki [52]

Rysunek 23: Projekt interfejsu – widok pełnoekranowego podglądu zdjęcia bez przybliżenia – źródło grafiki [52]



Rysunek 24: Projekt interfejsu – widok pełnoekranowego podglądu zdjęcia po przybliżeniu – źródło grafiki [52]

4.4. Model danych

Dane przedstawiające eksponaty, które będą rozpoznawane, zostały określone w fazie planowania jako obiekty o nazwie „ARData” posiadające następujące pola:

- id – string,
- name – string,
- author – string,
- year – int,
- shortDescription – string,
- longDescription – string.

Pole id jest unikalne, odpowiada również nazwie zdjęcia, które przedstawia dany eksponat, pozwoli to na powiązanie rozpoznanego obiektu z danymi. Zgodnie z przyjętymi normami pola zostały nazwane w języku angielskim.

5. Implementacja

5.1. Wprowadzenia

W tym rozdziale omówiono implementację modułu rozszerzonej rzeczywistości, przedstawiono wykorzystane technologie z naciskiem na opis działania algorytmu rozpoznawania obrazu. Następnie opisano krok po kroku proces implementacji wraz z fragmentami kodu.

5.2. Wykorzystane technologie

W tym podrozdziale pokrótce opisano wykorzystane podczas pracy nad aplikacją technologie, środowiska oraz biblioteki. Największa uwaga została poświęcona opisowi działania algorytmu rozpoznawania obrazu w bibliotece ARKit.

5.2.1. XCode

Do implementacji użyto programu Xcode, który jest zintegrowanym środowiskiem programistycznym (IDE) od firmy Apple, służącego do wytwarzania aplikacji na systemy macOS, iOS, iPadOS, tvOS oraz watchOS, czyli wszystkie aktualnie wspierane systemy od Apple. Wykorzystano środowisko w wersji 12.4.

5.2.2. Swift

Cała aplikacja została napisana w języku Swift w najnowszej wersji 5.5, wydanej we wrześniu 2021 roku, Język ten jest podstawą, jeśli chodzi o programowanie natywne na platformy, jest bardzo przyjazny i ma niski próg wejścia, posiada wiele nowoczesnych opcji, jak m.in. możliwość zwrócenia wielu wartości różnych typów, co jest niemożliwe w wielu językach. Podczas programowania nie trzeba pamiętać o zarządzaniu pamięcią, ponieważ język ten posiada ARC (Automatic Reference Counting), czyli system zarządzania pamięcią, który zwalnia pamięć używaną przez instancje klas, gdy nie są one już potrzebne.

5.2.3. SwiftUI

Framework SwiftUI służy do wykonywania widoków w aplikacjach pisanych w języku Swift. Jego użycie to najbardziej wydajna opcja implementacji widoków, opiera się w pełni na kodzie, nie ma potrzeby ręcznego konfigurowania niczego w dodatkowych opcjach, wszystko jest oparte o programowanie.

5.2.4. ARKit

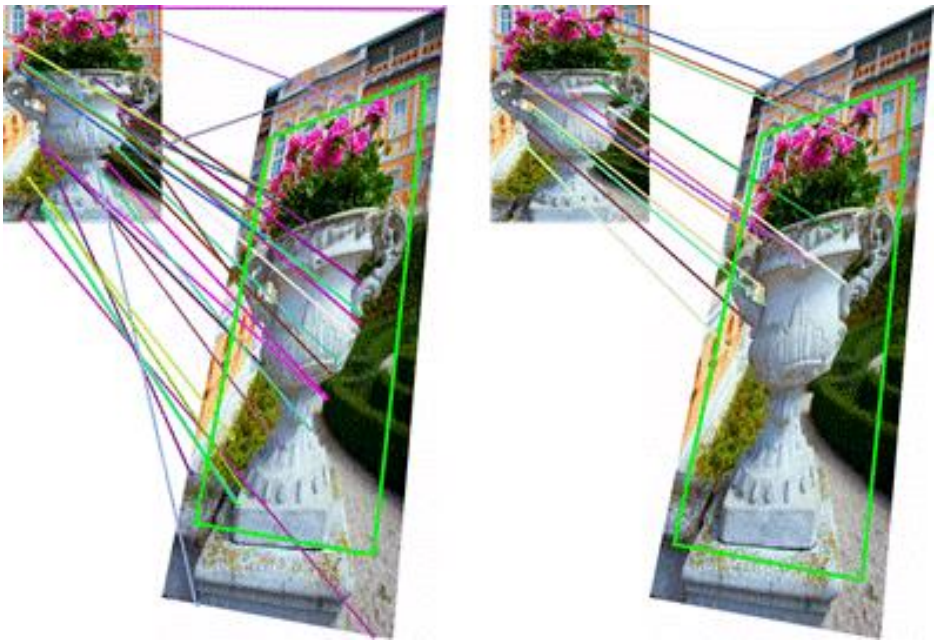
ARKit to najpopularniejsza biblioteka w języku Swift służąca do implementacji rozszerzonej rzeczywistości, posiada zaawansowane, lecz proste w użyciu algorytmy rozpoznawania obrazu. Umożliwia również wyświetlanie dowolnych obiektów dwu- oraz

trójwymiarowych. W przed procesem realizacji projektu biblioteka została porównana z dostępnymi konkurencyjnymi rozwiązaniami i wybrana ze względu na wysoką jakość algorytmu rozpoznawania obrazu, jego wysoką skuteczność oraz prostotę implantacji.

5.2.5. Algorytm rozpoznawania obrazu w bibliotece ARKit

Rozpoznawanie obrazów odbywa się poprzez oznaczenie cech szczególnych na grafikach źródłowych, które zostały podane na etapie implementacji aplikacji. W trakcie próby rozpoznania obrazu algorytm szuka w rzeczywistości właśnie tych cech szczególnych, które uprzednio wybrał. Gdy wykryje ich wystarczającą liczbę, uznaje, że wykrył jeden z szukanych obrazów. Wszystkie jednocześnie szukane obrazy muszą znajdować się w jednym folderze plików.

Oznaczanie punktów szczególnych zostało zaprezentowane na rysunku 25. Jako punkty szczególne oznaczone są przede wszystkim krawędzie oraz miejsca, gdzie następuje drastyczna zmiana koloru. Takie punkty styku różnych kolorów są na tyle charakterystyczne, że algorytm na podstawie wielu takich punktów jest w stanie skutecznie stwierdzić, że na aktualnym podglądzie kamery znajduje się jeden z szukanych obrazów.



Rysunek 25: Reprezentacja wyboru punktów szczególnych obrazu - źródło [53]

Obraz z kamery rejestrowany jest w 60 klatkach na sekundę, a każda klatka jest przetwarzana oddzielnie, zapamiętywane są jednak obiekty wykryte we wcześniejszych klatkach i aktualizowane są tylko ich współrzędne, jeśli się zmieniły.

Każdy wykryty obraz będzie interpretowany jako oddzielna powierzchnia, w przypadku wykrycia więcej niż jednego, potencjalne informacje, jak na przykład napisy bądź grafiki odnoszące się do obrazów, będą osadzone na jednej powierzchni, która będzie musiała być zdefiniowana. Istnieje jednak możliwość jednoczesnego umieszczania różnych obiektów, nawet wizualnie takich samych, jednak zdefiniowanych jako osobne byty na różnych powierzchniach.

Do prawidłowego działania algorytmu obrazy, które będą miały być wykrywane, muszą spełniać dwa podstawowe wymogi:

- rozdzielczość co najmniej 480 pikselu w każdym z obu wymiarów,
- wystarczająca ilość punktów szczególnych.

Pierwszy z wymogów jest łatwy do spełnienia, ponieważ rozdzielczość 480x480 jest o wiele niższa niż standard zdjęć robionych nawet przez telefony komórkowe niższej klasy. Drugi wymóg jest niemierzalny zgodnie z dostępnymi na stronach Apple informacjami, o tym, że nie został on spełniony informacja pojawia się po dodaniu do projektu takiego zdjęcia. W przypadku niespełniania któregoś z wymagań, nadal będzie możliwa próba wykrycia takiego przedmiotu, jednak nie ma gwarancji, że algorytm zadziała poprawnie.

5.3. Proces implementacji

Wstępna konfiguracja modułu rozszerzonej rzeczywistości polegała na zaimportowaniu biblioteki ARKit oraz napisaniu krótkiego kodu, będącego podstawą dalszych prac.

```
class ARViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!
    var items = [String: ARData]()

    override func viewDidLoad() {
        super.viewDidLoad()
        sceneView.delegate = self
        sceneView.showsStatistics = false
        sceneView.autoenablesDefaultLighting = true
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        let configuration = ARWorldTrackingConfiguration()
        let referenceImages = ARReferenceImage.referenceImages(inGroupNamed: "AR Resources", bundle: nil)!

        configuration.detectionImages = referenceImages

        sceneView.session.run(configuration)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        sceneView.session.pause()
    }
}
```

Rysunek 26: Kod aplikacji – wstępna konfiguracja – źródło własne

W kodzie zaprezentowanym na rysunku 26. użyto konfiguracji ARWorldTrackingConfiguration zamiast alternatywnej ARImageTrackingConfiguration, ponieważ w tej konfiguracji przechowywane są dane o wykrytych obiektach również, gdy znikną one z pola widzenia kamery. Zapamiętywana jest ich lokalizacja i wykorzystując odczyty żyroskopu oraz akcelerometru, urządzenie jest w stanie określić, że gdy wróci do danej lokalizacji powinien znajdować się tam wykryty wcześniej eksponat. Konfiguracja ta gwarantuje też, że gdy początkowo zostanie wykryty cały obiekt, a następnie wskutek zbliżania się do niego lub obrócenia kamery widoczny będzie tylko jego fragment, to dalej będzie możliwość interakcji z nim. Wymaga to założenia, że obiekty nie będą się przemieszczały, ponieważ gdy któryś z nich nie będzie znajdował się w początkowej lokalizacji, algorytm nie będzie w stanie tego stwierdzić.

Dane obrazów do wykrycia przez algorytm zostały przekazane jako zawartość folderu „AR Resources”. Dane te poza obrazami zawierają też informacje o fizycznych wymiarach tych obiektów.

Po skonfigurowaniu podstawowych parametrów, proces implementacji modułu rozszerzonej rzeczywistości został rozpoczęty od poprawnego rozpoznawania obrazów widzianych w podglądzie kamery. W bibliotece ARKit używana jest do tego funkcja renderer, która „w zależności od konfiguracji, może automatycznie dodawać kotwice (ang. anchor) do sesji. Widok wywołuje tę metodę raz dla każdej nowej kotwicy.” [54]. W konfiguracji przestawionej powyżej tj. ARWorldTrackingConfiguration(), funkcja ta będzie wywoływana za każdym razem, gdy rozpoznany zostanie obiekt, którego zdjęcie znajduje się w katalogu „AR Resources”.

```
func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
    guard let imageAnchor = anchor as? ARImageAnchor else { return nil }
    let size = imageAnchor.referenceImage.physicalSize

    let plane = SCNPlane(width: size.width, height: size.height)
    plane.firstMaterial?.diffuse.contents = UIColor.blue.withAlphaComponent(0.9)

    let planeNode = SCNNode(geometry: plane)
    planeNode.eulerAngles.x = -.pi / 2

    let node = SCNNode()
    node.addChildNode(planeNode)
    return node
}
```

Rysunek 27: Kod aplikacji – funkcja renderer na początkowym etapie implementacji – źródło własne

W kodzie znajdującym się na rysunku 27. rozpoznane obiekty zostaną pokryte niebieskim kolorem o intensywności 90%, aby w dalszym ciągu widoczne było co znajduje się pod nim. Efekt został przedstawiony na rysunkach 28, oraz 29.

Rysunek 28: Zrzut ekranu z aplikacji w trakcie implementacji – rozpoznanie oraz oznaczenie obrazu – źródło własne

Rysunek 29: Zrzut ekranu z aplikacji w trakcie implementacji – rozpoznanie oraz oznaczenie obrazu – źródło własne

Do identyfikacji obrazów użyto ich nazw, które można otrzymać z rozpoznanych obiektów dzięki dodaniu do wcześniejszego kod dodatkowej linii, która znajduje się na zdjęciu 30.

```
guard let name = imageAnchor.referenceImage.name else { return nil }
```

Rysunek 30: Kod aplikacji – funkcja zapisanie nazwy wykrytego zdjęcia – źródło własne

Z racji zastosowania identycznych wartości zarówno dla pliku przedstawiającego zdjęcie eksponatu, po którym jest rozpoznawany oraz dla identyfikatora danych tego eksponatu, można je jednoznacznie powiązać ze sobą.

Dane te zostały zapisane w formacie JSON [55] i są ładowane z pliku po uruchomieniu modułu rozszerzonej rzeczywistości. Dane są w formacie zgodnym ze strukturą „ARData" przedstawioną na rysunku 31.

```
struct ARData: Decodable {
    let id: String
    let name: String
    let author: String
    let description: String
    let year: Int
    let longdescription: String
}
```

Rysunek 31: Kod aplikacji – struktura ARData – źródło własne

```
"portretMarii": {
  "id": "portretMarii",
  "name": "Portret Marii Pareńskiej",
  "author": "Stanisław Wyspiański",
  "description": "Stanisław Wyspiański w 1902 roku namalował 18 letnią wtedy Marynę Pareńską. Dziewczyna pojawiła się w
      najsłynniejszym dramacie Wyspiańskiego, Modelka została przedstawiona na wyrazistym, roślinnym tle, w którym dostrzec
      można łodygi i liście chryzantem.",
  "year": 1902,
  "longdescription": "Portret Maryny Pareńskiej – w rzeczywistości nazywała się Maria jednak pod imieniem Maryna wystąpiła w
      Weselu Wyspiańskiego, namalowany w 1902 roku znajduje się w zbiorach Muzeum Narodowego we Wrocławiu. Modelka została
      przedstawiona na wyrazistym, roślinnym tle, w którym dostrzec można łodygi i liście chryzantem. W 1900 roku Maria
      uczestniczyła w weselu Lucjana Rydla z Jadwigą Mikołajczykówną, co pozwoliło uczynić z niej jedną z postaci Wesela
      Stanisława Wyspiańskiego. Dorastała w kręgach artystycznych Krakowa."
},
```

Rysunek 32: Kod aplikacji – przykładowe dane eksponatu w formacje JSON – źródło własne

Dane z pliku „ARData.json", którego przykładowy fragment zaprezentowano na rysunku 32. są wczytywane za pomocą funkcji loadData, przedstawionej na rysunku 33.

```
func loadData() -> [String: ARData]{
    guard let url = Bundle.main.url(forResource: "ARdata", withExtension: "json") else {
        fatalError("Unable to find JSON in bundle")
    }

    guard let data = try? Data(contentsOf: url) else {
        fatalError("Unable to load JSON")
    }

    let decoder = JSONDecoder()

    guard let loadedItems = try? decoder.decode([String: ARData].self, from: data) else {
        fatalError("Unable to parse JSON.")
    }
    return loadedItems
}
```

Rysunek 33: Kod aplikacji – funkcja loadData – źródło własne

Po wczytaniu dane mogą zostać wyświetlone jako tekst w rozszerzonej rzeczywistości, aby nie zaburzać czytelności wyświetlane będą tylko podstawowe dane eksponatu: nazwa, autor oraz krótki opis.

Do generowania napisów jako obiektów 3D, które mogą zostać później wykorzystane do wyświetlania ich w rozszerzonej rzeczywistości, została zaimplementowana funkcja textNode – na rysunku 34. Służy ona do odpowiedniego ustawienia obiektu SCNText, w zależności czy jest to tekst zawierający nazwę eksponatu, jego autora czy opis. W celu zwiększenia przejrzystości i czytelności do tekstu dodane zostało tło. Niestety taka możliwość nie została dodana domyślnie dla klasy SCNText, dlatego zaimplementowana została funkcja addBackground – na rysunku 35., która dla podanego tekstu dodaje tło w kolorze czarnym. Zapewnia to dobrą czytelność bez względu ta otoczenie i oświetlenie.

Rozmiary czcionki są początkowo ustawiane, jednak później sprawdzane jest czy w przypadku tytułu nie jest on dłuższy od obrazu, pod którym ma się wyświetlać. Jeśli tak jest, to jest on skalowany by był takiej samej długości jak eksponat, który opisuje. W przypadku autora zastosowano podobne ograniczenie, jednak w tym przypadku napis nie może być dłuższy niż 70% długości dzieła. Dla opisu natomiast tekst jest zawijany by był nie dłuższy niż obraz.

```
func textNode(_ str: String, maxWidth: Float, type: TextType ) -> SCNNode {
    var font: UIFont
    let metersToCentymeters = Float(0.01)
    switch type {
        case TextType.Title:
            font = UIFont.boldSystemFont(ofSize: 8)
        case TextType.Author:
            font = UIFont.boldSystemFont(ofSize: 6)
        default:
            font = UIFont.boldSystemFont(ofSize: 4)
    }
    var scale = metersToCentymeters*maxWidth

    let text = SCNText(string: str, extrusionDepth: 0.1)
    text.font = font

    var textNode = SCNNode(geometry: text)
    textNode.scale = SCNVector3(scale, scale, 0.0001)

    let minVec = textNode.boundingBox.min
    let maxVec = textNode.boundingBox.max
    let w = Float(maxVec.x - minVec.x)

    switch type {
        case TextType.Title :
            if(Float(maxWidth) < w*metersToCentymeters){
                scale = Float(maxWidth/(w))
                textNode = SCNNode(geometry: text)
                textNode.scale = SCNVector3(scale, scale, 0.0001)
            }
        case TextType.Author :
            if(Float(maxWidth*0.7) < (w*metersToCentymeters)){
                scale = Float(maxWidth*0.7/(w))
                textNode = SCNNode(geometry: text)
                textNode.scale = SCNVector3(scale, scale, 0.0001)
            }
        default:
            text.containerFrame = CGRect(origin: .zero, size: CGSize(width: Int(100), height: 500))
            text.isWrapped = true
    }

    let background = addBackground(textNode: textNode)
    textNode.addChildNode(background)
    return textNode
}
```

Rysunek 34: Kod aplikacji – funkcja textNode – źródło własne

```
func addBackground(textNode: SCNNode)-> SCNNode{
    let minVec = textNode.boundingBox.min
    let maxVec = textNode.boundingBox.max
    let w = CGFloat(maxVec.x - minVec.x) + CGFloat(2)
    let h = CGFloat(maxVec.y - minVec.y) + CGFloat(2)
    let d = CGFloat(maxVec.z - minVec.z)

    let background = SCNPlane(width: w, height: h)
    background.cornerRadius = CGFloat(2)
    background.firstMaterial?.diffuse.contents = UIColor.black
    background.firstMaterial?.isDoubleSided = true
    let backgroundNode = SCNNode(geometry: background)
    backgroundNode.position = SCNVector3Make((maxVec.x - minVec.x) / 2 + minVec.x, (maxVec.y - minVec.y) / 2 + minVec.y,
        -Float(d))
    return backgroundNode
}
```

Rysunek 35: Kod aplikacji - funkcja addBackground – źródło własne

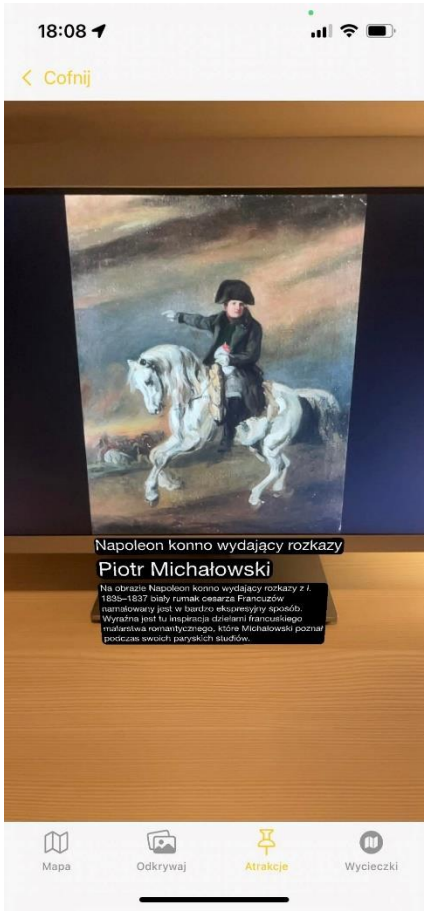
W funkcji renderer opisywanej już wcześniej, funkcja textNode wykorzystywana jest trzykrotnie – co zostało przedstawione na rysunku 36. – każdorazowo dla tekstu, który ma zostać wyświetlony. Najpierw dla tytułu dzieła, ponieważ ten znajduje się bezpośrednio pod obrazem, następnie dla danych autora i opisu. Wszystkie teksty umieszczane są pod obrazem i wyrównywane są do lewej jego krawędzi.

```
func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
    guard let imageAnchor = anchor as? ARImageAnchor else { return nil }
    guard let name = imageAnchor.referenceImage.name else { return nil }
    guard let item = items[name] else { return nil }
    let size = imageAnchor.referenceImage.physicalSize
    let plane = SCNPlane(width: size.width, height: size.height)
    plane.firstMaterial?.diffuse.contents = UIColor.clear

    let planeNode = SCNNode(geometry: plane)
    planeNode.eulerAngles.x = -.pi / 2
    planeNode.name = name

    let spacing = Float(0.015*size.height)

    let titleNode = textNode(item.name, maxWidth: Float(size.width), type: TextType.Title)
    titleNode.pivotOnTopRight()
    titleNode.position.x -= Float(plane.width / 2)
    titleNode.position.y -= (Float(plane.height / 2) + spacing)
    titleNode.name = name
    titleNode.childNodes[0].name = name
    planeNode.addChildNode(titleNode)

    let authorNode = textNode(item.author, maxWidth: Float(size.width), type: TextType.Author)
    authorNode.pivotOnTopRight()
    authorNode.position.x -= Float(plane.width / 2)
    authorNode.position.y = titleNode.position.y - (titleNode.height + spacing)
    authorNode.name = name
    authorNode.childNodes[0].name = name
    planeNode.addChildNode(authorNode)

    let descriptionNode = textNode(item.description, maxWidth: Float(size.width), type: TextType.Description)
    descriptionNode.pivotOnTopRight()
    descriptionNode.position.x = titleNode.position.x
    descriptionNode.position.y = authorNode.position.y - (authorNode.height + spacing)
    descriptionNode.name = name
    descriptionNode.childNodes[0].name = name
    planeNode.addChildNode(descriptionNode)

    let node = SCNNode()
    node.addChildNode(planeNode)
    return node
}
```

Rysunek 36: Kod aplikacji – funkcja renderer – źródło własne

Na tym etapie informacje są poprawnie wyświetlane co zostało zaprezentowane na rysunkach 37. oraz 38.

Rysunek 37: Zrzut ekranu z aplikacji w trakcie implementacji – wyświetlenie informacji dotyczących obrazu w rozszerzonej rzeczywistości – źródło własne



Rysunek 38: Zrzut ekranu z aplikacji w trakcie implementacji – wyświetlenie informacji dotyczących obrazu w rozszerzonej rzeczywistości – źródło własne

Kolejna dodana funkcjonalność to możliwość przechodzenia z ekranu rozszerzonej rzeczywistości do widoku eksponatu i została ona zaimplementowana poprzez obsługę naciśnięcia na dowolny element dotyczący danego obrazu lub jego samego. Na wykrytym obrazie generowana jest przezroczysta powierzchnia, dzięki której możliwe jest wirtualne dotknięcie eksponatu. Wszystkie wyświetlane teksty oraz wspomniany przezroczysty element, posiadają parametr name równy identyfikatorowi rozpoznanego obrazu, pozwala to przekazać do widoku eksponatu konkretny obiekt, którego dane mają zostać tam wyświetlone. Funkcjonalność polega na obsłudze gestu z klasy UITapGestureRecognizer, funkcja handleTap zaprezentowana na rysunku 39., została przypisania w konfiguracji jako domyślnie uruchamiana po wykryciu dowolnego gestu ze wspomnianej klasy, który jest zwykłym dotknięciem ekranu, zostało to przedstawione na rysunku 40. Dzięki temu wyświetlane elementy stały się niejako przyciskami.

```
@objc func handleTap(sender: UITapGestureRecognizer){
    let tappedView = sender.view as! SCNView
    let touchLocation = sender.location(in: tappedView)
    let hitTest = tappedView.hitTest(touchLocation, options: nil)
    if !hitTest.isEmpty {
        let result = hitTest.first!
        let name = result.node.name
        let hostingController = UIHostingController(rootView: ExhibitDetailView(object: items[name!]))
        addChild(hostingController)
        view.addSubview(hostingController.view)
        hostingController.didMove(toParent: self)
        self.navigationController?.pushViewController(hostingController, animated: true)

    }
}
```

Rysunek 39: Kod aplikacji – funkcja handleTap – źródło własne

```
let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(handleTap))
sceneView.addGestureRecognizer(tapGestureRecognizer)
```

Rysunek 40: Kod aplikacji – dodanie funkcji handleTap do konfiguracji – źródło własne

Sam widok eksponatu jest standardowym widokiem zbudowanym we frameworku SwiftUI, jego kod został zaprezentowany na rysunku 41. Sam widok składa się z grafiki przedstawiającej eksponat oraz tekstów wypisanych pod nią. Przekazują one użytkownikowi zarówno podstawowe informacje jakie widział on w rozszerzonej rzeczywistości, jak i dodatkowe, które dostępne są tylko w tym widoku - rok powstania oraz dodatkowe informacje w postaci dłuższej wersji opisu.

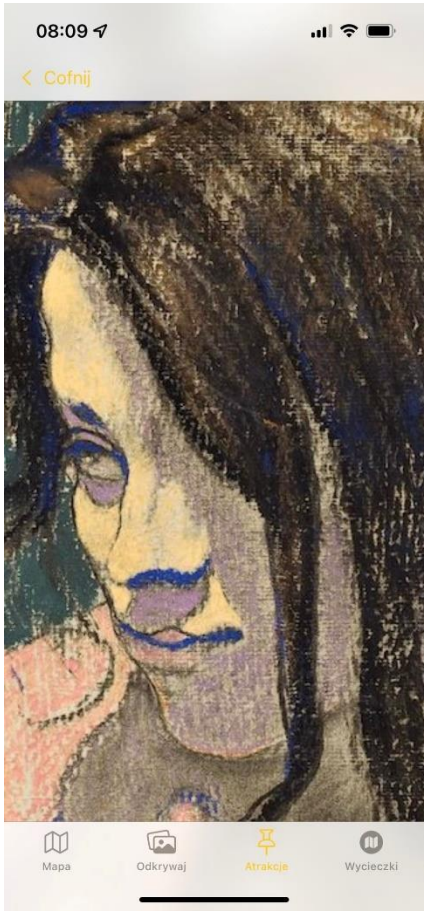
Po naciśnięciu na obrazek następuje przejście do tryby pełnoekranowego wyświetlania grafiki, w którym jest możliwość przybliżania grafiki co umożliwia dokładne obejrzenie szczegółów obrazu.

```
struct ExhibitDetailView: View {

    var object: ARData!
    @Environment(\.presentationMode) var presentationMode: Binding<PresentationMode>
    var btnBack : some View { Button(action: {
            self.presentationMode.wrappedValue.dismiss()
            }) {
                HStack {
                    Image(systemName: "chevron.backward")
                        .foregroundColor(.yellow)
                    Text("Cofnij")
                }
            }
        }

    var body: some View {
        ScrollView {
            VStack {
                NavigationLink(destination: FullScreenImage(image: UIImage(named: "\(object.id)Col")!)) {
                    Image("\(object.id)Col")
                        .resizable()
                        .scaledToFit()
                }
                .padding(5)
                HStack{
                    Text(object.name)
                        .font(.title)
                    Spacer()
                }
                .padding(5)
                HStack{
                    Text("Autor " + object.author)
                    Spacer()
                }
                .padding(5)
                HStack{
                    Text("Rok powstania " + String(object.year))
                    Spacer()
                }
                .padding(5)
                Text(object.longdescription)
                    .padding(5)
                Spacer()
            }
        }
        .navigationBarItems(leading: btnBack)
        .navigationBarBackButtonHidden(true)
        .navigationBarTitleDisplayMode(.inline)
        .navigationViewStyle(StackNavigationViewStyle())
    }
}
```

Rysunek 41: Kod aplikacji - widok eksponatu – źródło własne

Na rysunku 42. zaprezentowano finalny wygląd zaimplementowanego widoku eksponatu, jest on w pełni zgodny z widokiem zaprojektowanym w fazie projektowej. Widok pełnoekranowego podglądu zdjęcia eksponatu znajdujący się na rysunku 43. również spełnia założenia postawione w fazie projektowej.

Rysunek 42: Zrzut ekranu z aplikacji w trakcie implementacji – widok szczegółów eksponatu – źródło własne



Rysunek 43: Zrzut ekranu z aplikacji w trakcie implementacji – widok pełnoekranowego podglądu zdjęcia eksponatu – źródło własne

5.4. Efekt końcowy

W ramach prac nad aplikacją zostały zaimplementowane dane dla 18 obrazów pochodzących ze zbiorów Muzeum Narodowego we Wrocławiu, z dwóch wystaw stałych: „Sztuka polska XVII–XIX w.” oraz „Sztuka europejska XV–XX w.”. Aplikacja rozpoznaje m.in:

- Portret Marii Pareńskiej – Stanisława Wyspiańskiego,
- Targ koński na Pradze – Juliusza Kossaka,
- Napoleon konno wydający rozkazy – Piotra Michałowskiego,
- Madonna z Dzieciątkiem i św. Janem – Agnolo Bronzinego,
- Wieczór - Wassilego Kandinsky'ego.

Działanie aplikacji wewnątrz muzeum przedstawiono na rysunkach 44., 45,. 56. oraz 47.

Rysunek 44: Zrzut ekranu skończonej aplikacji – rozpoznanie obrazu „Portret Marii Pareńskiej” w Muzeum Narodowym we Wrocławiu – źródło własne

Rysunek 45: Zrzut ekranu skończonej aplikacji – rozpoznanie obrazu „Napoleon konno wydający rozkazy” w Muzeum Narodowym we Wrocławiu – źródło własne

Rysunek 46: Zrzut ekranu skończonej aplikacji – rozpoznanie obrazu „Targ koński na Pradze” w Muzeum Narodowym we Wrocławiu – źródło własne

Rysunek 47: Zrzut ekranu skończonej aplikacji – rozpoznanie obrazu „Chłopiec niosący snop” w Muzeum Narodowym we Wrocławiu – źródło własne

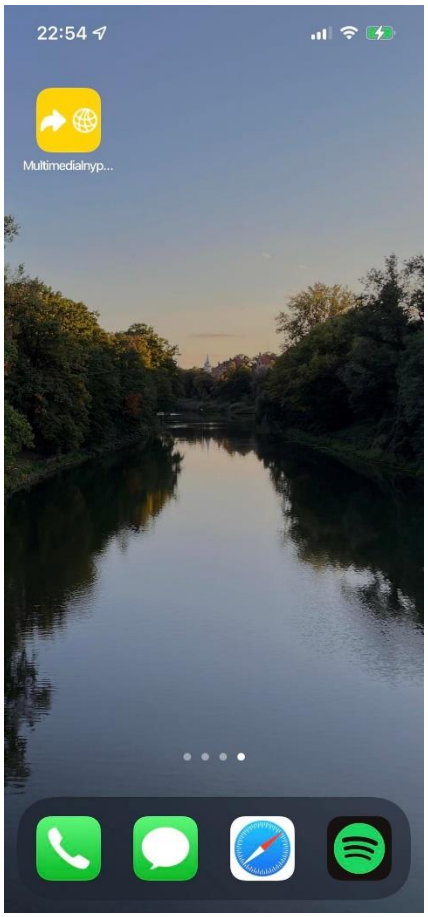
5.5. Instrukcja użytkowania

Instrukcja obsługi ma za zadanie wyjaśnić w jaki sposób użytkownik powinien korzystać z aplikacji, aby ta mogła spełnić swoje zadanie, czyli przekazać mu dodatkowe informacje o eksponatach muzealnych.

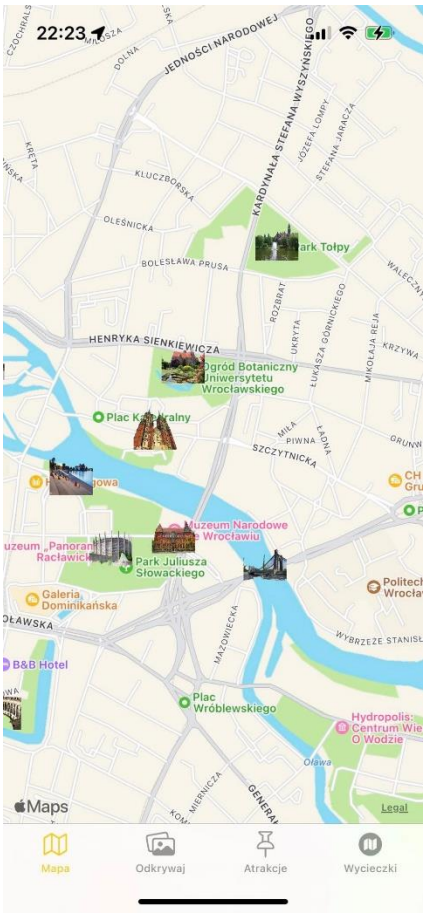
1. Uruchomienie aplikacji

Aplikacja Multimedialnego przewodnika po Dolnym Śląsku posiada charakterystyczną żółtą ikonę widoczną na rysunku 48., którą z poziomu smartfonu należy nacisnąć w celu jej uruchomienia.

Po uruchomieniu pierwszym widokiem jest widok mapy – rysunek 49.



Rysunek 48: Zrzut ekranu menu telefonu - ikona aplikacji widoczna w lewym górnym rogu – źródło własne



Rysunek 49: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok mapy – źródło własne

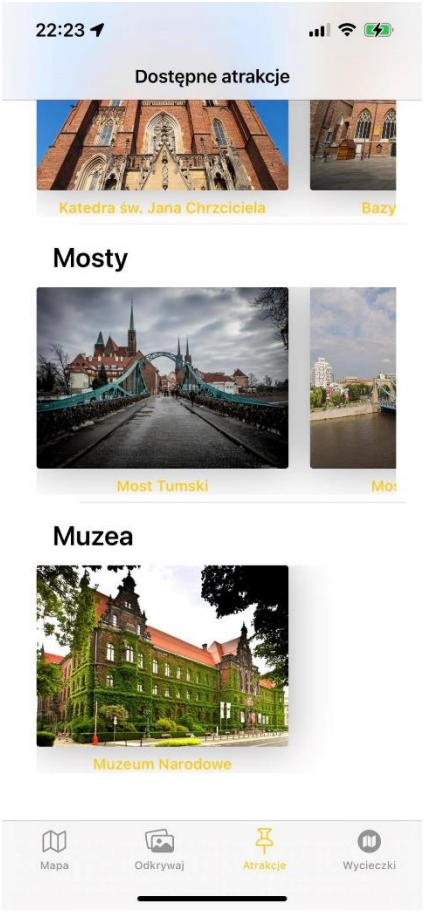
2. Przejście do widoku atrakcji

Przejście do widoku atrakcji jest możliwe, z każdego ekranu aplikacji, należy w tym celu nacisnąć na ikonę znajdującą się w dolnym menu aplikacji. Ikona jest opatrzona podpisem „Atrakcje” oraz symbolem pinezki.

Po przejściu do widoku atrakcji na ekranie wyświetli się lista ze skategoryzowanymi atrakcjami – rysunek 50. Użytkownik musi wybrać rodzaj atrakcji, w której się znajduje, aby móc używać w niej rozszerzonej rzeczywistości.

Jeśli dla wybranej atrakcji turystycznej dostępny jest widok rozszerzonej rzeczywistości, to pod zdjęciem, a nad nazwą, będą dostępne dwa przyciski „Pokaż trasę” oraz „Odkryj” – rysunek 51. Jeśli dana atrakcja nie posiada eksponatów, które można odkrywać w rozszerzonej rzeczywistości, wyświetli się tylko pierwszy z przycisków.

Chcąc przejść do trybu rozszerzonej rzeczywistości należy naciskać przycisk „Odkryj”.



Rysunek 50: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok listy atrakcji – źródło własne

Rysunek 51: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok atrakcji – Muzeum Narodowe we Wrocławiu – źródło własne

3. Tryb rozszerzonej rzeczywistości

Po przejściu w tryb rozszerzonej rzeczywistości, po uprzednim zezwoleniu aplikacji na dostęp do aparatu, użytkownik zobaczy podgląd z tylnej kamery telefonu – co zostało przedstawione na rysunku 52. Należy skierować telefon tak, aby w podglądzie znajdował się cały eksponat, o którym użytkownik chce pozyskać dodatkowe informacje.

Gdy obraz znajdzie się w polu widzenia kamery, aplikacja automatycznie go wykryje oraz wyświetli informacje o nim – rysunek 53. Dane będą się wyświetlały na ekranie telefonu poniżej rozpoznanego dzieła i od chwili, w której informacje pojawią się na ekranie nie jest już wymagane, aby całość obrazu była w kadrze.

Rysunek 52: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok rozszerzonej rzeczywistości – źródło własne

Rysunek 53: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok rozszerzonej rzeczywistości z rozpoznanym obrazem – źródło własne

4. Przejście do ekranu eksponatu

W rozszerzonej rzeczywistości wyświetlana jest tylko część dostępnych danych, aby uzyskać dostęp do pełnych informacji należy dotknąć na ekranie telefonu rozpoznanego wcześniej obrazu lub dowolnego tekstu odnoszącego się do niego.

5. Widok eksponatu

Widok eksponatu prezentuje pełne dostępne informacje o danym dziele w postaci:

- nazwy,
- imienia oraz nazwiska autora,
- roku powstania,
- pełnego opisu.

Nazwa oraz dane autora są takie jak w trybie rozszerzonej rzeczywistości, rok powstania jest dodatkową informacją, opis wyświetlany jest w dłuższej wersji niż ten wyświetlany w rozszerzonej rzeczywistości.

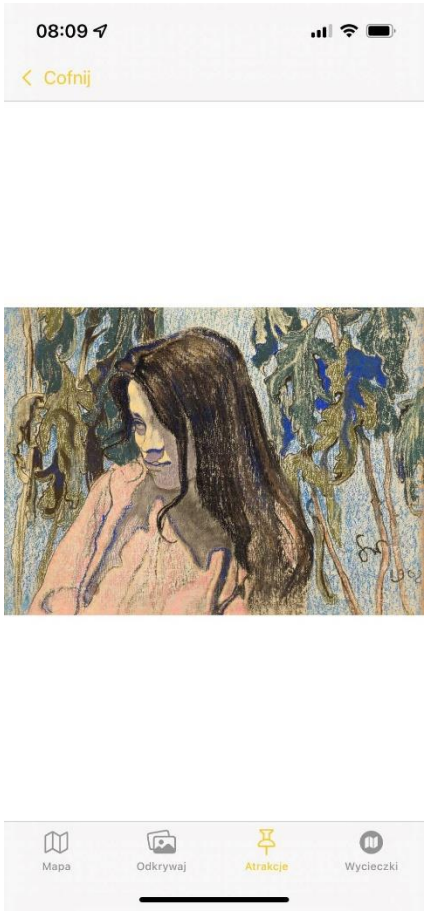
Widok ten podobnie jak cała aplikacja dostępny jest w dwóch motywach kolorystycznych: jasnym – rysunek 54. oraz ciemnym – rysunek 55. Wybór trybu nie jest możliwy w aplikacji, dostosowuje się on do ustawień, które użytkownik wybrał w swoim telefonie.

Rysunek 54: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok szczegółów eksponatu w trybie jasnym – źródło własne

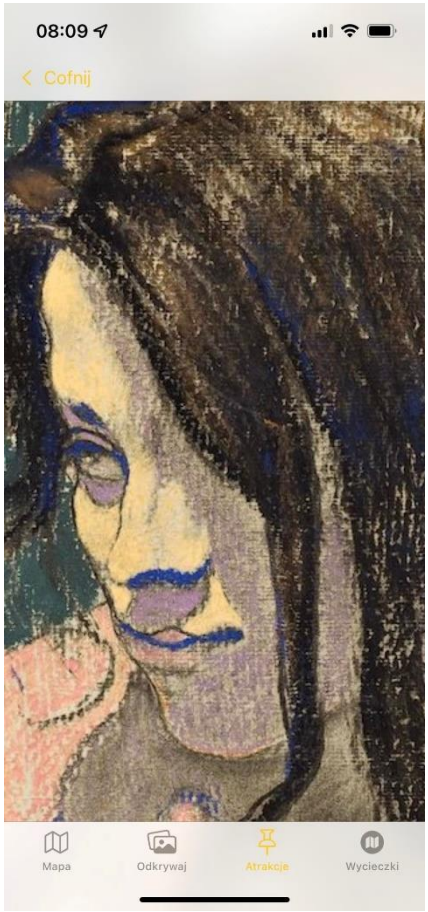
Rysunek 55: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok szczegółów eksponatu w trybie ciemnym – źródło własne

6. Pełnoekranowe wyświetlanie zdjęcia eksponatu

Użytkownik ma możliwość dokładnego przyjrzenia się dziełu sztuki dzięki przejściu w tryb pełnoekranowego wyświetlania zdjęcia – rysunek 56. W trybie tym można przybliżać oraz przesuwać zdjęcie – rysunek 57., co umożliwia dostrzeżenie pewnych szczegółów, być może początkowo niewidocznych. W celu przejścia do tego trybu należy w ekranie eksponatu nacisnąć na zdjęcie. Przybliżania można dokonywać gestem oddalenia od siebie dwóch palców na ekranie telefonu, zbliżając je do siebie zdjęcie będzie pomniejszane.



Rysunek 56: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok podglądu zdjęcia w trybie pełnoekranowym – źródło własne



Rysunek 57: Zrzut ekranu Multimedialnego przewodnika po Dolnym Śląsku – widok podglądu zdjęcia w trybie pełnoekranowym w przybliżeniu – źródło własne

5.6. Instalacja oprogramowania

Proces instalacji niczym nie różni się od instalacji innych aplikacji na platformie iOS, w obecnym stadium projektu przebiega jak poniżej:

- kiedy aplikacja nie została wydana do sklepu z aplikacjami App Store:
  - należy zainstalować aplikacje podłączając telefon kablem do komputera, na którym znajduje się oprogramowanie Xcode wraz z projektem aplikacji,
  - z poziomu telefonu wyrazić zgodę na przesyłanie danych z komputera,
  - z poziomu Xcode wybrać telefon jako urządzenie docelowe do zainstalowania aplikacji oraz w ustawieniach telefonu przejść do:
    - → ogólne,
    - → VPN i zarządzane urządzeniem,
    - → w sekcji aplikacja deweloper wybrać dewelopera, którego do którego należy instalowana aplikacja, a następnie nacisnąć przycisk zaufaj.
  - Po wykonaniu tych kroków, aplikacja znajduje się na telefonie i można ją uruchomić.
- Gdy aplikacja zostanie wydana do sklepu z aplikacjami, będzie dostępna dla każdego kto posiada na telefonie system operacyjny iOS 14.0 lub nowszy, a proces instalacyjny będzie polegał na:
  - wyszukaniu aplikacji w App Storze i naciśnięciu przycisku pobierz,
  - zaakceptowaniu przez użytkownika pobrania aplikacji przez uwierzytelnienie konta.

5.7. Testy poprawności działania

Ze względu na fakt, iż aplikacja została wykonana z myślą o telefonach pracujących na systemie operacyjnym iOS, została przetestowana na czterech modelach telefonów firmy Apple. Były to modele iPhone 13, 11, 8 oraz 7 Plus, których specyfikacje przedstawiono w tabeli 5. Najlepiej sprawdziły się modele 13 oraz 11 ze względu na największe ekrany telefonów, dzięki nim aplikacja była najbardziej czytelna. Jednak nawet na najmniejszym iPhonie 8 wyświetlana czcionka była możliwa do odczytania bez najmniejszych trudności.

Na żadnym z testowanych modeli nie zaobserwowano żadnych błędów czy spowolnień aplikacji. Nigdy nie było problemów z rozpoznaniem obrazu ani wyświetleniem elementów rozszerzonej rzeczywistości.

Tabela 5: Porównanie parametrów urządzeń na których testowano aplikacje opracowanie na podstawie [56], [57], [58]

Model	iPhone 13	iPhone 11	iPhone 8	iPhone 7 Plus
Procesor	Apple A15 Bionic	Apple A13 Bionic	Apple A11 Bionic	Apple A10 Fusion
Data premiery	III kwartał 2021	III kwartał 2019	III kwartał 2017	III kwartał 2016
Pamięć RAM	4 GB	4 GB	2 GB	3 GB
Ekran	1170 x 2532 px, 6,10 cali	828 x 1792 px, 6,10 cali	750 x 1334 px, 4.70 cali	1080 x 1920 px, 5.50 cali
Aparat	12 Mpix, f/1,5	12 Mpix, f/1,8	12 Mpix, f/1,8	12 Mpix, f/1,8
System operacyjny	iOS 15.1	iOS 15.1	iOS 15.1	iOS 15.1

5.7.1. Wpływ wielkości obrazu na działanie modułu rozszerzonej rzeczywistości

Aplikacja działa poprawnie bez względu na rozmiar obrazu jaki ma za zadanie rozpoznać. Obraz „Śluby Jana Kazimierza” autorstwa Jana Matejki ma wymiary 315 x 500 cm – rysunek 58, natomiast „Pejzaż zimowy z łyżwiarzami i pułapką na ptaki” Pietrera Bruegla Młodszego ma wymiary 39 x 57 cm – rysunek 59. Różnica w wymiarach obrazów jest znacząca, jednak nie wpływa to w żadnym stopniu na działanie aplikacji oba dzieła rozpoznawane są prawidłowo. Dla każdego z nich wymagana jest tylko odpowiednia przestrzeń, by użytkownik mógł stanąć w odpowiedniej odległości, wystarczającej do tego, aby cały obraz zmieścił się w podglądzie kamery.

Nie zaobserwowano różnic w działaniu aplikacji wynikających z różnic w wielkościach obrazów.

Rysunek 58: Obraz Śluby Jana Kazimierza, rozpoznany w trybie rozszerzonej rzeczywistości – źródło własne

Rysunek 59: Obraz Pejzaż zimowy z łyżwiarzami i pułapką na ptaki rozpoznany w trybie rozszerzonej rzeczywistości – źródło własne

Zakończenie

Wykonane prace

W ramach pracy zaimplementowano moduł rozszerzonej rzeczywistości do aplikacji mobilnej, będącej przewodnikiem po Dolnym Śląsku.

Realizacja celu pracy

Cel pracy został zrealizowany poprzez ukończenie aplikacji i przygotowanie działającego modułu rozszerzonej rzeczywistości dla wybranych obrazów z dwóch wystaw stałych w Muzeum Narodowym we Wrocławiu: „Sztuka polska XVII-XIX w.” oraz „Sztuka europejska XV-XX w.”. W ramach aplikacji możliwa jest interakcja z osiemnastoma dziełami, które są poprawnie rozpoznawane przez aplikację i wyświetlane są dla nich informacje, dostępne w ramach rozszerzonej rzeczywistości.

Aplikacja została w pełni wykonana i jest gotowa do wprowadzenia jej na rynek.

Z punktu widzenia muzeów, galerii sztuki lub innych potencjalnych zainteresowanych, aplikacja jest prosta we wprowadzeniu do użytku, działa samodzielnie bez konieczności instalacji dodatkowych urządzeń wewnątrz budynku. Jedynym wymaganiem, jest uzupełnienie bazy aplikacji o dane kolejnych wybranych dzieł.

Dla użytkownika aplikacja jest łatwa w obsłudze dzięki swojej intuicyjności. Wystarczy skierować telefon w stronę dzieła, a algorytm rozpoznawania obrazu automatycznie rozpozna eksponat i przedstawi jego dane na ekranie telefonu.

Wnioski

Rozszerzona rzeczywistość pozwala uatrakcyjnić kontakt ze sztuką poprzez proste dostarczenie podstawowych informacji o dziele bez potrzeby wyszukiwania ich przykładowo w internecie. Informacje mogą być tak jak w przypadku tej pracy przedstawiane natychmiastowo po skierowaniu telefonu w stronę jednego z eksponatów.

Możliwe dalsze ścieżki rozwoju

Możliwymi ścieżkami rozwoju aplikacji są przede wszystkim zwiększenie skali jej działania i rozszerzenie o dodatkowe obiekty będące atrakcjami turystycznymi, w których można zastosować moduł rozszerzonej rzeczywistości. W dalszej kolejności przeniesienie aplikacji na platformę Android i opublikowanie jej w sklepach z aplikacjami dla obu platform.

Inną możliwością jest wyodrębnienie modułu rozszerzonej rzeczywistości i użycie go w mniejszej aplikacji zorientowanej na konkretnym obiekcie turystycznym takim jak muzeum czy galeria sztuki i przygotowanie aplikacji skrojonej pod specyficzne potrzeby.

W każdym przypadku aplikacja wymagać będzie przetłumaczenia jej na inne języki niż polski, aby umożliwić korzystanie z niej turystom z zagranicy.

Bibliografia

1. Statista, *Number of smartphone users from 2016 to 2021* [Dostęp w dn. 18.12.2021], https://www.statista.com/statistics/330695/number-of-smartphone-usersworldwide/
2. Worldmeter, *Current World Population* [Dostęp w dn. 18.12.2021], https://www.worldometers.info/world-population/
3. Grand View Research, *Mobile Application Market Size, Share & Trends Analysis Report By Store Type (Google Store, Apple Store), By Application (Gaming, Music & Entertainment, Health & Fitness), By Region, And Segment Forecasts, 2020 - 2027* [Dostęp w dn. 18.12.2021], https://www.grandviewresearch.com/industry-analysis/mobile-application-market
4. Alan B. Craig, *Understanding Augmented Reality: Concepts and Application*, Newnes, 2013 [Dostęp w dn. 18.12.2021], https://books.google.pl/books?id=7_O5LaIC0SwC&lpg=PP1&ots=LHHBs1uUsb&dq=augmented%20reality&lr&hl=pl&pg=PA1#v=onepage&q&f=false
5. Bardi J., *What is Virtual Reality? [Definition and Examples]* [Dostęp w dn. 18.12.2021], https://www.marxentlabs.com/what-is-virtual-reality/
6. Chapple C., *Pokémon GO Catches $5 Billion in Lifetime Revenue in Five Years* [Dostęp w dn. 18.12.2021], https://sensortower.com/blog/pokemon-go-five-billion-revenue
7. Empemedia, *Social Media w Polsce 2021 nowy raport* [Dostęp w dn. 18.12.2021], https://empemedia.pl/social-media-w-polsce-2021-nowy-raport/
8. Olafson K., *How to Make Your Own Instagram AR Filters: A Step-by-Step Guide* [Dostęp w dn. 18.12.2021], https://blog.hootsuite.com/instagram-ar-filters/
9. Buxton M., *Facebook Just Added Face Filters - Here's What They Look Like* [Dostęp w dn. 18.12.2021], https://www.refinery29.com/en-us/2016/08/119089/facebook-photo-effects-face-filters
10. Muzeum Narodowe w Krakowie, *XX + XXI. Galeria Sztuki Polskiej* [Dostęp w dn. 18.12.2021], https://mnk.pl/wystawy/xx-xxi-galeria-sztuki-polskiej
11. Astor M., *Label for Wayne Thiebaund's Diagonal Freeway. From the collection of De Young Museum in San Francisco, CA* [Dostęp w dn. 18.12.2021], https://www.flickr.com/photos/15965815@N00/247470991
12. Niezła Sztuka, *Kody QR w Muzeum Miasta Łodzi* [Dostęp w dn. 18.12.2021], https://niezlasztuka.net/projekty/kody-qr-w-muzeum/
13. Statista, *Mobile operating systems' market share worldwide from January 2012 to June 2021* [Dostęp w dn. 18.12.2021], https://www.statista.com/statistics/272698/global-market-share-held-by-mobileoperating-systems-since-2009/
14. Apple, *iOS 15* [Dostęp w dn. 18.12.2021], https://developer.apple.com/ios/
15. Android, *Android 12* [Dostęp w dn. 18.12.2021], https://developer.android.com/about
16. Apple, *Xcode 13* [Dostęp w dn. 18.12.2021], https://developer.apple.com/xcode/

17. Apple, *Documentation Objective-C*
[Dostęp w dn. 18.12.2021],
https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html
18. Apple, *Swift*
[Dostęp w dn. 18.12.2021], https://www.apple.com/in/swift/
19. Android, *Kotlin*
[Dostęp w dn. 18.12.2021], https://developer.android.com/kotlin
20. Oracle, *Java*
[Dostęp w dn. 18.12.2021], https://docs.oracle.com/en/java/
21. Android, *Studio*
[Dostęp w dn. 18.12.2021], https://developer.android.com/studio
22. ZDNet, *Top programming languages: Most popular and fastest growing choices fo developers*
[Dostęp w dn. 18.12..2021], https://www.zdnet.com/article/top-programming-languages-most-popular-and-fastest-growing-choices-for-developers/
23. Android, *Developer stories*
[Dostęp w dn. 18.12.2021], https://developer.android.com/kotlin/stories
24. Flutter, *Flutter documentation*
[Dostęp w dn. 18.12.2021], https://docs.flutter.dev/
25. React, *React documentation*
[Dostęp w dn. 18.12.2021], https://reactjs.org/
26. Statisa, *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021*
[Dostęp w dn. 18.12.2021], https://www.statista.com/statistics/869224/worldwide-software-developer-workinghours/
27. Dart, *Dart documentation*
[Dostęp w dn. 18.12.2021], https://dart.dev/guides
28. Apple, *Human Interface Guidelines*
[Dostęp w dn. 18.12.2021], https://developer.apple.com/design/human-interface-guidelines/
29. Mozilla, *JacaScript documentation*
[Dostęp w dn. 18.12.2021], https://developer.mozilla.org/en-US/docs/Web/JavaScript
30. Apple, *ARKit*
[Dostęp w dn. 18.12.2021], https://developer.apple.com/documentation/arkit
31. Apple. *RealityKit*
[Dostęp w dn. 18.12.2021], https://developer.apple.com/documentation/realitykit/
32. Apple, *SceneKit*
[Dostęp w dn. 18.12.2021], https://developer.apple.com/documentation/scenekit/
33. Apple, *AR Creation Tools*
[Dostęp w dn. 18.12.2021], https://developer.apple.com/augmented-reality/tools/
34. Google, *ARCore documentation*
[Dostęp w dn. 18.12.2021], https://developers.google.com/ar/develop
35. Unity, *AR Foundation*
[Dostęp w dn. 18.12.2021], https://unity.com/unity/features/arfoundation
36. Jeffrey H. Shuhaiber, MD, *Augmented Reality in Surgery*, Arch Surg. 2004;139:170-17
[Dostęp w dn. 18.12.2021],
https://jamanetwork.com/journals/jamasurgery/articlepdf/396410/srv3004.pdf

37. Gobetti E., Scateni R., *Virtual Reality: Past, Present, amd Future*
[Dostęp w dn. 18.12.2021], http://www.p-arch.it/bitstream/handle/11050/682/vrreport98.pdf?sequence=1&isAllowed=y
38. Greenwald W., *Augmented Reality (AR) vs. Virtual Reality (VR): What's the Difference?*
[Dostęp w dn. 18.12.2021], https://www.pcmag.com/news/augmented-reality-ar-vs-virtual-reality-vr-whats-the-difference
39. Tulane University, *What's the DIfference Between AR and VR?*
[Dostęp w dn. 18.12.2021], https://sopa.tulane.edu/blog/whats-difference-between-ar-and-vr
40. Oculus, *Oculus Rift Pre-Orders*
[Dostęp w dn. 18.12.2021], https://www.oculus.com/blog/oculus-rift-pre-orders-now-open-first-shipments-march-28/
41. House A., Playstation.Blog, *Playstation VR: Launching October for $399*
[Dostęp w dn. 18.12.2021], https://blog.playstation.com/2016/03/15/playstation-vr-launching-october-for-399/
42. Google Trend, *AR, VR*
[Dostęp w dn. 18.12.2021], https://trends.google.com/trends/explore?date=2011-11-01%202021-10-31&q=AR,VR
43. Petchenik I., *Introducing the All-New Flightradar244 Mobile App*
[Dostęp w dn. 18.12.2021], https://www.flightradar24.com/blog/introducing-the-all-new-flightradar24-mobile-app/
44. Google Play, *Snapchat*
[Dostęp w dn. 18.12.2021],
https://play.google.com/store/apps/details?id=com.snapchat.android&gl=PL
45. Thier D., *Pokémon GO's New AR+ On iOS Is Actually Cool, But Needs Work*
[Dostęp w dn. 18.12.2021],
https://www.forbes.com/sites/davidthier/2017/12/21/pokemon-gos-new-ar-on-ios-is-actually-cool-but-needs-work/?sh=7960f8347330
46. Smithsonina's National Museum of Natural History, *Skin & Bones promotional video*
[Dostęp w dn. 18.12.2021], https://www.youtube.com/watch?v=7agVb4IG16M
47. App Store, *Ikea Place*
[Dostęp w dn. 18.12.2021], https://apps.apple.com/us/app/ikea-place/id1279244498
48. Netural, *Rommle - 3D & AR Planning Tool*
[Dostęp w dn. 18.12.2021], https://www.netural.com/en/projects/roomle-3d-ar-room-planner
49. The Week, *Google Translate review: how well does the new app work?*
[Dostęp w dn. 18.12.2021], https://www.theweek.co.uk/google/62130/google-translate-review-how-well-doesthe-new-app-work
50. App Store, *Anatomy AR+*
[Dostęp w dn. 18.12.2021], https://apps.apple.com/dk/app/anatomy-ar/id1496828001
51. Microsoft, *Wzorzec Model-View-ViewModel*
[Dostęp w dn. 18.12.2021], https://docs.microsoft.com/pl-pl/xamarin/xamarin-forms/enterprise-application-patterns/mvvm
52. Francikowska A., *Muzy Stanisława Wyspiańskiego, 2018*
[Dostęp w dn. 18.12.2021], https://sztukipiekne.pl/muzy-stanislawa-wyspianskiego/

53. de la Hera O., *ARKit Theory: The Point Cloud, Image Recognition, AR Ready Images, True Scale, The Renderer and Nodes*
[Dostęp w dn. 18.12.2021], https://medium.com/ar-tips-and-tricks/arkit-theory-the-point-cloud-imagerecognition-ar-ready-images-true-scale-the-renderer-and-e1508398dd4
54. Apple, *Documentation - Instant Method renderer*
[Dostęp w dn. 18.12.2021],
https://developer.apple.com/documentation/arkit/arscnviewdelegate/2865794-renderer
55. JSON, *Introducing JSON*
[Dostęp w dn. 18.12.2021], https://www.json.org/json-en.html
56. mGSM, *Porównanie telefonów: Apple 13 vs Apple iPhone 11 vs Apple iPhone 8 vs Apple iPhone 7 Plus*
[Dostęp w dn. 18.12.2021], https://www.mgsm.pl/pl/porownanie/apple-iphone13-vs-apple-iphone11-vs-appleiphone8-vs-apple-iphone7plus/
57. Apple, *Porównaj modele iPhone 13, iPhone11*
[Dostęp w dn. 18.12.2021],
https://www.apple.com/pl/iphone/compare/?modelList=iphone13,iphone11
58. Apple, *Porównaj modele iPhone 8, iPhone7 Plus*
[Dostęp w dn. 18.12.2021],
https://www.apple.com/pl/iphone/compare/?modelList=iphone8,iphone7plus

Spis rysunków

Spis tabel