МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка мобільного застосунку з елементами доповненої реальності для проведення екскурсій в музеї

Виконала студентка групи К-41
спеціальності 122 «Комп'ютерні науки»
Молчанова Анастасія Юріївна

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант _____

Рецензент к.техн.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2021

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ADT – Android Development Tools – Інструменти розробки в ОС Android

API – Application Programming Interface – програмний інтерфейс програми

AR – Augmented Reality – доповнена реальність

IDE – Integrated Development Environment – інтегроване середовище розробки

SDK – Software Development Kit – набір засобів розробки

ВСТУП

У світі, який дуже змінився за останній рік, стають все більш популярними і актуальними види діяльності, що дозволяють вчитися або дізнаватися щось нове при мінімальному особистому контакті з іншими людьми або через Інтернет за допомогою електронних пристроїв. Не варто забувати і про мистецтво. Під час карантину багато музеїв працюють в безпечному режимі. Все більше з них організовують віртуальні екскурсії, онлайн-тури, покращують свої сайти і створюють самоосвітні портали про мистецтво.

Як швидко і безпечно отримати більш детальну інформацію про картину, яка знаходиться перед очима? Можна розпитати працівника музею. Можна спробувати пошукати інформацію в Інтернеті. Але чи не буде зручніше просто навести камеру свого пристрою на картину і відразу дізнатися про неї все необхідне? Саме тому буде актуальною розробка програми для розпізнавання об'єктів мистецтва з використанням технології доповненої реальності.

Практично у кожної людини є смартфон, яким він може користуватися в будь-який час протягом дня. Тому розробка саме мобільного застосунку є актуальним завданням.

Метою кваліфікаційної роботи є розробка мобільного застосунку «Find ARt» для розпізнавання об’єктів мистецтва з використанням технології доповненої реальності. Для досягнення поставленої мети були сформульовані наступні завдання:

- провести аналіз предметної області щодо розпізнавання об’єктів мистецтва;
- провести порівняльний аналіз існуючих програм-аналогів;
- обґрунтувати вибір програмних засобів розробки та технологій;
- провести проектування системи з використанням мови UML;
- виконати реалізацію застосунку;
- підготувати інструкцію користувача;

– виконати тестування застосунку.

Структура кваліфікаційної роботи бакалавра складається з вступу, 5 розділів, висновків, переліку посилань на 18 найменувань, 1 додатку. Повний обсяг проекту становить 90 сторінок, містить 31 рисунок і 1 таблицю.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Мистецтво – це спосіб розуміння і відображення дійсності шляхом створення творів, здатних викликати емоційний відгук у людей. Поряд з наукою, мистецтво використовується людством для правильного сприйняття і осмислення навколишнього світу. Найважливіша функція мистецтва полягає в задоволенні двох духовних потреб людини: любові до прекрасного і бажанні отримувати естетичне задоволення. Мистецтво також допомагає формувати суспільну свідомість, стимулює появу нових думок і уявлень за допомогою отриманих людиною відчуттів [1].

Арістотель виділяв три функції мистецтва: пізнавальну, виховну та емоційного впливу. Сучасна естетична наука виділяє значно більшу кількість функцій [2].

– Суспільно-перетворююча та компенсаторна функція. Мистецтво залучає людей до цілеспрямованої творчої діяльності, в ході якої відбувається трансформація людиною фактів дійсності, матеріалу та самої людини. В мистецтві втілюються ідеали, відбувається рух у напрямку досягнення бажаного, компенсація певної неповноцінності життя людини.

– Пізнавально-евристична функція. Мистецтво пізнає дійсність співвідносно з людиною, в усьому багатстві форм, що сприймаються людською чуттєвістю. Завдяки цьому мистецтво здатне відкривати нове у вже відомому. Мистецтво виступає засобом просвіти, передачі досвіду, фактів життя, а також засобом навчання, передачі навичок мислення, узагальнення системи поглядів.

– Художньо-концептуальна функція. Мистецтво дозволяє побачити у художньому творі уявлення митця про світ у цілому. Через твір певна цілісна

концепція подається у чуттєво сприйманих формах, де емоційно втілені уявлення про світ, людину та її місце у світі.

– Інформативна та комунікативна функція. Кожен твір мистецтва є знаковою системою, тож має свій код, розшифровка якого полягає в особливостях культури, що породила твір. Художнє спілкування являє собою обмін культурними змістами, що зумовлює залучення до певної культури. Мистецтво здатне об'єднувати людей, особливо через катарсис — процес «очищення» за допомогою виникнення схожих почуттів. Мистецтво дозволяє пережити досвід інших людей, тим самим розширюючи історично обмежений досвід життя особистості.

– Сугестивна функція. Мистецтво здатне навіювати певний склад думок та почуттів, оскільки діє безпосередньо на почуття сприймачів, минаючи бар'єри раціонального мислення.

– Естетична функція. Мистецтво ціннісно орієнтує людину в світі, сприйняття творів мистецтва спонукає до подальшої творчості.

– Гедоністична функція. Мистецтво приносить задоволення, всі явища мистецтва співвідносні з естетичними цінностями. Джерелом естетичної насолоди є художня форма, яка перебуває в гармонійній єдності зі змістом. Залучення до творчості та ігровий аспект, присутній при сприйнятті художнього твору, здатні викликати почуття радості.

Історично мистецтво розвивається як система конкретних видів діяльності, серед яких особливо прийнято виділяти літературу, музику, архітектуру, живопис, скульптуру, декоративно-прикладне мистецтво. Поряд з естетикою велику роль в розумінні суті мистецтва і його соціальної ролі грають такі дисципліни, як психологія мистецтва, теорія та історія мистецтва, соціологія мистецтва і культурологія. Сучасні культурологічні дослідження розглядають мистецтво в системі інших феноменів культури і глобальних змін різних типів культур.

У наш вік технологій потреби і інтереси людей поступово змінюються. На перший план виходить прагматизм. Пріоритет творчої діяльності і моральних потреб втрачає важливість. Багато людей ніколи не відвідують музеї та виставки за власною ініціативою. Погляди молоді на образотворче мистецтво не відрізняються широтою. Всі бачили картину «Крик», але навряд чи назвуть її автора. Що намалював Вінсент Ван Гог крім «Зоряної ночі»? У якому столітті була створена «Мона Ліза»? Які картини представлені в Одеському художньому музеї? Чи є в його експозиції картини Івана Айвазовського?

При цьому в суспільстві цінується освіченість і ерудиція. Тому важливо не забувати про мистецтво. Твори істинного мистецтва є естетичними цінностями, незалежно від того, прекрасні чи потворні явища життя в них відображені. Естетична цінність творів мистецтва визначається мірою внутрішньої правди, тобто якістю відображення. Почуття, викликані мистецтвом, — особливі почуття, не зовсім такі, як у повсякденному житті. Вони великі, масштабні, вони знімають кордони між окремою людиною і цілим людством.

Під час карантину багато музеїв працюють в безпечному режимі, покращують свої сайти і створюють самоосвітні портали про мистецтво. Все більше в них організовують віртуальні екскурсії та онлайн-тури. Зараз особливо цінується можливість швидко і безпечно отримати необхідну інформацію, наприклад, детальні відомості про картину, яка знаходиться перед очима. Головною метою мобільного застосунку «Find ARt» є зручне та швидке отримання необхідної інформації про об’єкти мистецтва. Цей застосунок допоможе користувачам розширювати свій творчий кругозір та ставати більш ерудованими у питаннях образотворчого мистецтва.

1.2 Аналіз існуючих програмних систем

На даний час створюються різноманітні програми і сайти, що надають можливість досліджувати об’єкти мистецтва та переглядати експозиції музеїв, не виходячи з дому. Розглянемо деякі з них.

Застосунок Artivive (рис.1.1) дозволяє розпізнавати картини при наведенні на них камери та переглядати процес їх малювання. Застосунок працює у ландшафтній орієнтації, має мінімалістичний, привабливий дизайн.

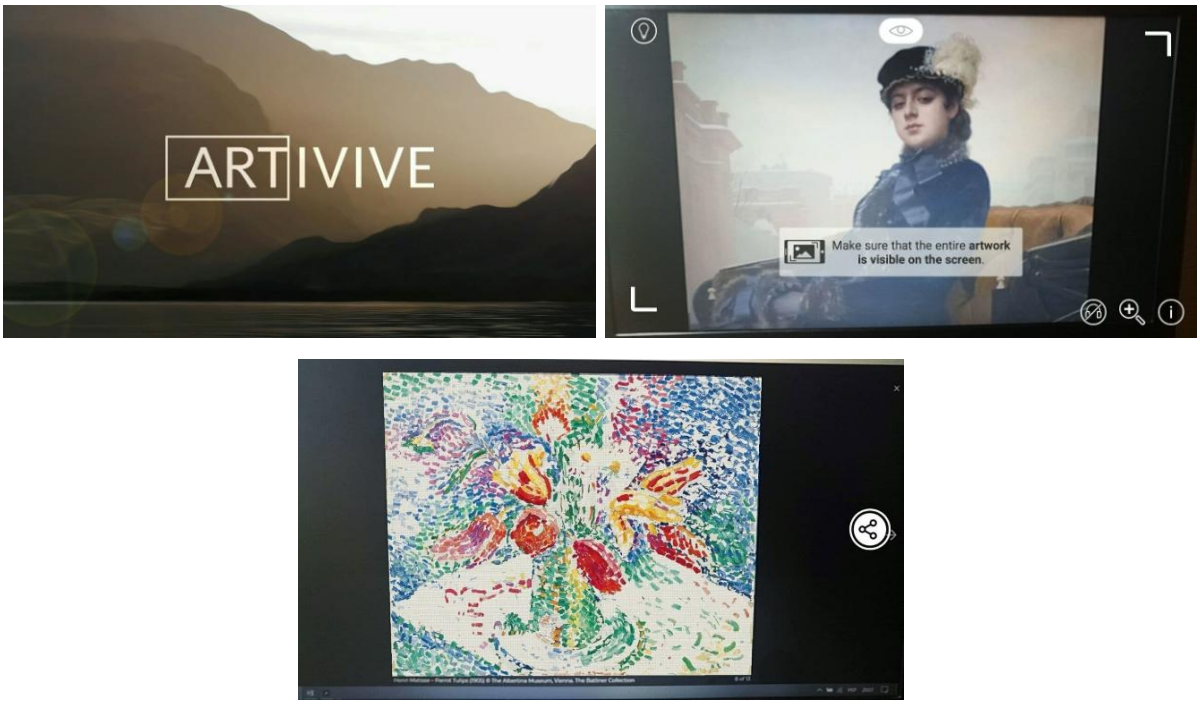


Рисунок 1.1 – Інтерфейс застосунку Artivive

Основний недолік застосунку – він працює лише за наявності спеціальних маркерів, які пов’язані з роботами, що виставлені в Австрійській галереї «Альбертіна». Таким чином, кількість картин, що можна розпізнавати, дуже обмежена.

Застосунок Find Art – Art Dealer & Auctioneer Tool (рис.1.2) використовується для розпізнавання картин з камери або з галереї. Однак застосунок не працює з технологією доповненої реальності, а використовує стандартний пошук схожих зображень в Інтернеті. Має темний дизайн і багато реклами.

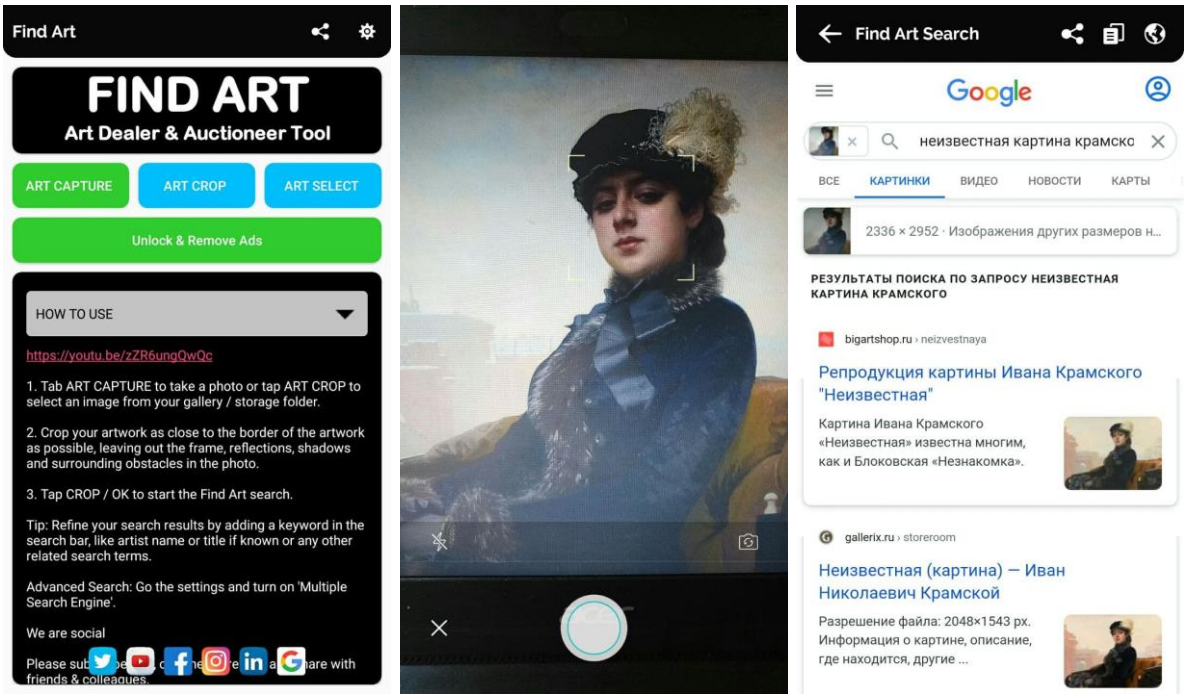


Рисунок 1.2 – Інтерфейс застосунку Find Art

Застосунок Smartify надає велику кількість можливостей: дозволяє переглядати експозиції відомих музеїв, кожного дня на головній сторінці програми пропонує досліджувати популярні роботи, пропонує найближчі тури, базуючись на даних геолокації. Застосунок має дуже велику базу даних об’єктів мистецтва, художників та музеїв, яка регулярно поповнюється. Розпізнані картини можна додавати в «Улюблене» та ділитися ними через соціальні мережі та месенджери. Застосунок має сучасний дизайн у темному стилі. Інтерфейс доступний на багатьох мовах, в тому числі, російською.

Недоліками Smartify (рис.1.3) є те, що інформація про картини доступна лише англійською і в базі відсутні будь-які музеї на території України та картини з їхніх експозицій.

Застосунок Google Arts & Culture (рис.1.4) представляє цікаві статті про музейні експонати, віртуальні тури відомими музеями, дозволяє за фотографією обличчя знаходити картини зі схожими людьми, роздивлятися картини у справжньому розмірі. Застосунок має функцію розпізнавання картин за допомогою технологій доповненої реальності, але вона працює лише в деяких

музеях і лише на деяких пристроях. Інтерфейс представлений англійською мовою, але може бути переведений за допомогою Google Translate.

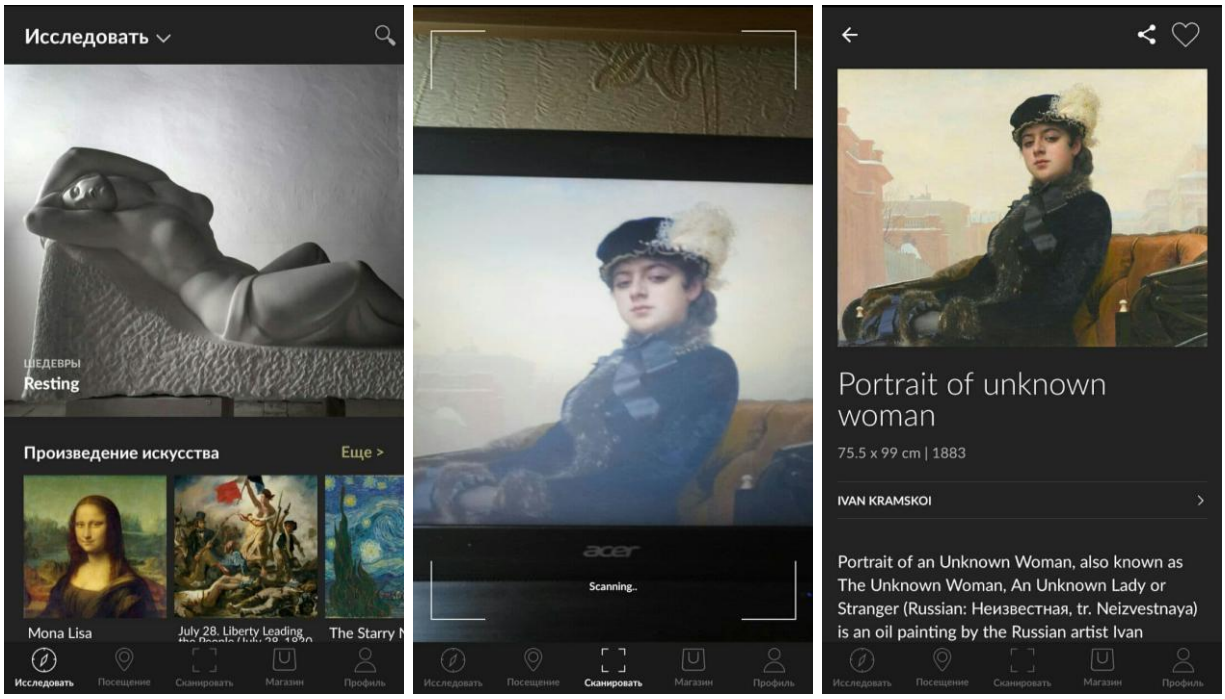


Рисунок 1.3 – Інтерфейс застосунку Smartify

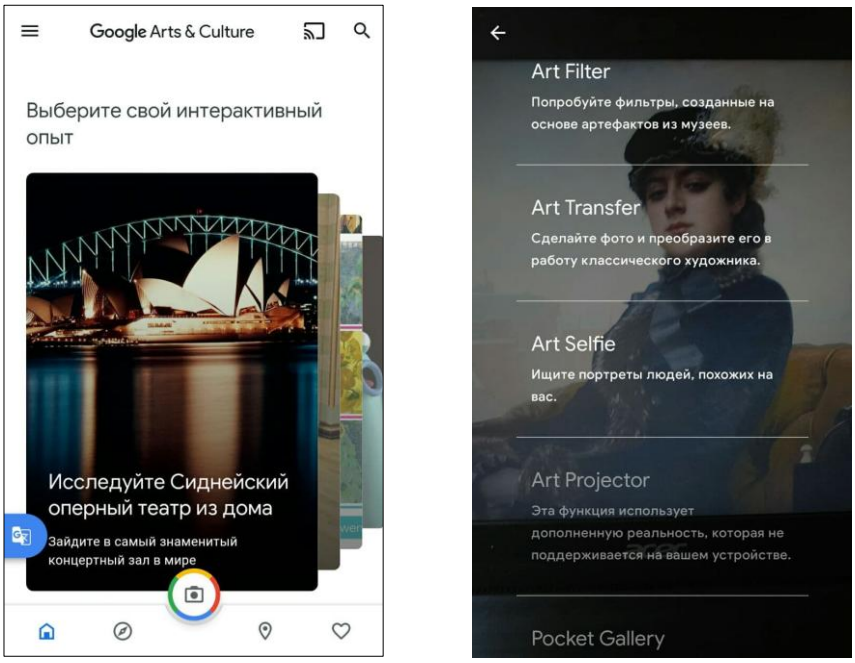


Рисунок 1.4 – Інтерфейс застосунку Google Arts & Culture

Порівняння перерахованих вище застосунків наведено у табл. 1.1.

Таблиця 1.1 – Порівняння мобільних застосунків для розпізнавання об’єктів мистецтва

Критерій порівняння	Artivive	Find Art	Smartify	Google Arts & Culture
Використання технології доповненої реальності	Так	Ні	Так/Ні	Так
Велика база об’єктів, які можна розпізнавати	Ні	База відсутня	Так	Так
Мова інтерфейсу та даних про розпізнані об’єкти	Англійська	Англійська	Російська лише для інтерфейсу, дані англійською	Інформацію можна перекласти Google Translate
Можливість створення аккаунту, зберігання даних про розпізнавання	Ні	Ні	Так	Так/Ні

Таким чином, під час виконання першого розділу бакалаврської роботи була обрана предметна область, в рамках якої розглянуті програмні аналоги і їх особливості. Переваги та недоліки розглянутих мобільних застосунків-аналогів були взяті до уваги при розробці програмного продукту.

2 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Функціональні та нефункціональні вимоги

Перед початком розробки застосунку були визначені функціональні та нефункціональні вимоги до нього.

До функціональних вимог належать:

- розпізнавання картин за допомогою камери смартфону;
- відображення основної інформації про розпізнаний об’єкт;
- можливість зберігати об’єкт в «улюбленому» для подальшого перегляду.

До нефункціональних вимог належать:

- смартфон з ОС Android версією 4.4+;
- наявність камери на смартфоні;
- підключення до Інтернету.

2.2 Обґрунтування вибору операційної системи

На рис. 2.1 представлена діаграма, що показує частки, які за даними на січень 2020 займають популярні мобільні ОС на ринку [3].

Згідно до статистики від компанії Statcounter [4], особистому досвіді та діаграмі на рис. 2.1 були виділені дві найбільш популярні ОС, які необхідно розглянути: iOS та Android.

iOS використовується тільки на пристроях Apple, таких, як iPhone. iOS не вимагає величезних обсягів оперативної пам'яті, оскільки вона може підтримувати загрузку і готовність більше десятка застосунків всього лише з 2 ГБ. І хоча роздільна здатність дисплея деяких моделей може здатися низькою, щільність пікселів залишається більш ніж достатньою. У той же час, більш низька роздільна здатність означає менше роботи для графічного процесора і, отже, меншу витрату батареї. Якість камери смартфона повністю

залежить від виробника. У випадку з Apple, у iPhone завжди були відмінні камери.

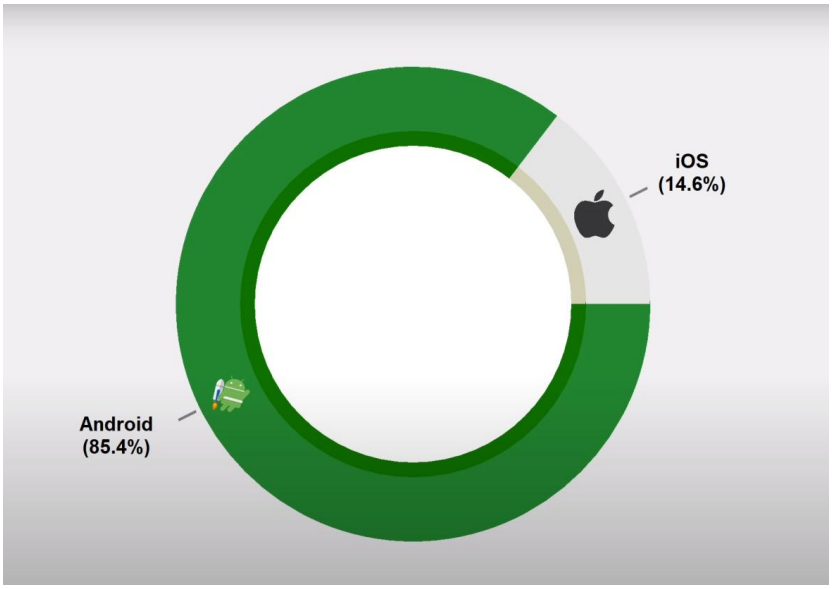


Рисунок 2.1 – Діаграма співвідношення мобільних ОС на ринку смартфонів від Pie Chart Pirate

В App Store багато платних застосунків, iOS не передбачає завантаження та встановлення застосунків з будь-яких інших джерел. З іншого боку, строгий контроль Apple над застосунками, доступними для їх платформи, гарантує, що вони функціонують належним чином, і що вони повністю безпечні і вільні від будь-якого шкідливого коду.

Пристрої Apple дорогі. Це стосується не тільки iPhone, але і всього іншого: MacBook, iMac, Apple Watch і практично всіх їхніх аксесуарів. Єдиний спосіб отримати доступний iPhone - це отримати модель більш старого покоління, потенційно навіть потриманий.

Таким чином, не дивлячись на високу продуктивність, високоякісні застосунки та довгострокові оновлення ОС, iOS не дуже приваблива для розробників. Для того, щоб опублікувати свій застосунок в App Store, необхідно мати аккаунт розробника Apple ($99 в рік) та фізичний комп'ютер з macOS

для створення проектів в Xcode. До того, кількість користувачів iOS набагато менше за кількість користувачів Android, тож постає питання окупності.

В даний час Android є найпопулярнішою платформою для смартфонів у світі і використовується багатьма виробниками телефонів. Особливість Android – відкритий вихідний код. Крім того, телефони Android дозволяють користувачам отримувати доступ до сторонніх магазинів застосунків. Телефони Android на відміну від iOS також дозволяють користувачеві завантажувати застосунки вручну, завантажуючи файли apk [5].

На 2020 рік кількість застосунків в магазині Google Play перевищує 2,8 млн. Згідно з даними, зібраними SafeBettingSites, в третьому кварталі 2020 року кількість завантажень з Google Play досягло 28,3 млрд, що в три рази перевищило показник App Store (8,2 млрд) [6].

Для того, щоб опублікувати свій застосунок в Google Play, необхідно одноразово сплатити збір $25, що значно дешевше у зрівнянні з Apple Store. До того, доступна безліч інструментів для розробки застосунків для Android.

З огляду на перераховані особливості операційних систем iOS та Android, перевага надається системі Android. Тож розробка застосунку буде виконуватися для пристроїв з ОС Android.

2.3 Вибір середовища розробки

Популярними середовищами розробки під ОС Android є Eclipse, Android Studio, Unity.

Eclipse – вільне інтегроване середовище розробки модульних кросплатформних застосунків. Розвивається і підтримується Eclipse Foundation. Для розробки Android- застосунків в Eclipse необхідно підключити плагін Android Development Tools. Однак плагін Eclipse ADT більше не підтримується, згідно з оголошенням у червні 2015 року. Цей плагін має багато відомих і потенційних помилок безпеки, які не будуть виправлені. Саме тому Google рекомендує переключитися на використання Android Studio.

Android Studio – офіційне інтегроване середовище розробки (IDE) для Android. Дане середовище орієнтоване на розробку застосунків саме під ОС Android, що виділяє його серед інших IDE.

Особливості Android Studio [7]:

- середовище розробки підтримує роботу з декількома мовами програмування, до яких відносяться найпопулярніші - C / C ++, Java.
- редактор коду, з яким зручно працювати;
- дозволяє розробляти програми не тільки для смартфонів / планшетів, а й для портативних ПК, приставок для телевізорів Android TV, пристроїв Android Wear, мобільних пристроїв з незвичайним співвідношенням сторін екрану;
- тестування коректності роботи нових ігор, утиліт, їх продуктивності на тій чи іншій системі відбувається безпосередньо в емуляторі;
- рефакторинг вже готового коду;
- досить велика бібліотека з готовими шаблонами і компонентами для розробки ПЗ;
- розробка програм для Android N - останньої версії операційної системи;
- попередня перевірка вже створеного застосунку на предмет помилок в ньому;
- великий набір засобів інструментів для тестування кожного елемента програми, ігри;
- для недосвідчених / початківців розробників спеціально створено посібник з використання Android Studio, розміщений на офіційному сайті.

Однак Android Studio не орієнтоване на розробку застосунків з використанням технології доповненої реальності. Багато платформ для роботи з AR (такі як Hololens, Oculus, Vuforia тощо) призначені для Unity.

Unity – крос-платформне середовище розробки застосунків, розроблена американською компанією Unity Technologies. Unity дозволяє створювати за-

стосунки, що працюють на більш ніж 25 різних платформах, що включають Windows, Mac, iOS, Android, PlayStation, Xbox, Nintendo Switch, а також провідні AR- і VR-платформи.

Unity не є повністю безкоштовним продуктом. Існує кілька видів підписок [8]:

- Personal. Безкоштовна версія, яка містить всі основні функції движка. Має таке обмеження: дохід в рік або обсяг залучених коштів не повинен перевищувати 100 000 $.
- Plus. За 35 доларів на місяць надаються різні звіти і аналітика, а також можливість змінювати заставку, 20%-а знижка на покупки в Asset Store і різні дрібні переваги. Має таке обмеження: дохід за рік або обсяг залучених коштів не повинен перевищувати 200 000 $.
- Pro. За 125 доларів на місяць включає в себе всі переваги версії Plus і додатково професійний сервіс і преміум-підтримку. Без обмежень по обороту або обсягу коштів.
- Окремі версії для бізнесу (використовуються великими компаніями).

Для багатьох проектів буде достатнім використання безкоштовної підписки Personal.

Unity підтримує дві системи збірки Android: Gradle та внутрішню.

Кроки, пов’язані зі збіркою для Android:

- підготовка та побудова активів Unity Assets;
- компіляція скріптів C#;
- обробка плагінів;
- розділення ресурсів на частини, які переходять до файлів .apk та OBB, якщо вибрано Split Application Binary;
- створення ресурсів Android за допомогою утиліти AAPT (лише для внутрішньої збірки);
- створення маніфесту Android;

- об’єднання маніфестів бібліотеки та маніфесту Android (лише для внутрішньої збірки);
- компіляція коду Java у формат Dalvik Executable (DEX) (лише для внутрішньої збірки);
- збірка бібліотеки IL2CPP, якщо вибрано IL2CPP Scripting Backend;
- побудова та оптимізація пакетів APK та OBB.

Таким чином, за підсумком аналізу існуючих середовищ розробки, було обрано середовище Unity, тому що воно є крос-платформним, найбільше підходить під потреби застосунку, що розроблюється, та має багато навчальної інформації і форум для підтримки.

3 ТЕХНОЛОГІЯ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

3.1 Поняття технології доповненої реальності

Доповнена реальність (англ. Augmented reality, AR) – це середовище, яке в реальному часі доповнює фізичний світ цифровими даними за допомогою відповідних пристроїв і програмної частини [9].

Специфіка технології доповненої реальності полягає в тому, що вона програмним чином візуально поєднує два спочатку незалежних простору: світ реальних об’єктів і віртуальний світ, відтворений на комп’ютері. Нове віртуальне середовище утворюється шляхом накладання запрограмованих віртуальних об’єктів поверх відеосигналу з камери і стає інтерактивним шляхом використання спеціальних маркерів. AR також можна визначити як систему, яка виконує три основні функції: поєднання реального та віртуального світів, взаємодію в реальному часі та точну 3D-реєстрацію віртуальних та реальних об’єктів. Накладена сенсорна інформація може бути конструктивною (тобто добавкою до природного середовища) або деструктивною (тобто маскувати природне середовище).

Основа технології доповненої реальності – це система оптичного трекінгу. Камера розпізнає маркери в реальному світі, «переносить» їх у віртуальне середовище, накладає один шар реальності на інший і таким чином створює світ доповненої реальності [10].

Апаратна частина пристрою для роботи з доповненою реальністю складається з процесорів, дисплеїв, різноманітних датчиків та пристроїв вводу інформації. Сучасні мобільні пристрої, такі як смартфони або планшети, мають в собі всі зазначені елементи, а також камеру та МЕМС (акселерометр, GPS, цифровий компас), що роблять їх придатними до використання в якості платформи для AR. Це в свою чергу сприяє поширенню доповненої реальності і зростаючій популярності технології серед користувачів.

3.2 Сфери використання технології доповненої реальності

Так як доповнена реальність насправді легко доступна, вона використовується безліччю способів: фільтри Snapchat, у застосунках, які допомагають знайти машину на переповненій автостоянці, в різноманітних торгових застосунках, які дозволяють приміряти одяг не виходячи з дому. Мабуть, найвідомішим прикладом технології AR є мобільний застосунок Pokemon Go, який вийшов у 2016 році і швидко став сенсацією. У грі гравці знаходять і захоплюють персонажів покемонів, які з'являються в реальному світі.

Звичайно, окрім ігор, існує дуже багато варіантів застосування технологій доповненої реальності:

- покращені навігаційні системи використовують доповнену реальність для накладання маршруту на прямий огляд дороги;
- під час футбольних ігор використовуються технології AR, щоб намалювати лінії на полі для ілюстрації та аналізу гри;
- гігант меблів та предметів домашнього вжитку IKEA пропонує програму AR (IKEA Place), яка дозволяє побачити, як якийсь предмет меблів буде виглядати та вміщуватися у вашому просторі;
- пілоти військових винищувачів бачать AR-проекцію своєї висоти, швидкості та інших даних на своєму козирку шолома;
- нейрохірурги іноді використовують AR-проекцію тривимірного мозку, які допомагають їм в операціях;
- на таких історичних місцях, як Помпеї в Італії, AR може проектувати подання давніх цивілізацій на сучасні руїни, оживляючи минуле;
- наземний екіпаж в аеропорту Сінгапуру носить окуляри AR, щоб побачити інформацію про вантажні контейнери, що прискорює час завантаження.

3.3 Основні принципи роботи технології доповненої реальності

Загальна схема створення доповненої реальності у всіх випадках така: камера пристрою доповненої реальності знімає зображення реального об'єкта; програмне забезпечення (ПЗ) пристрою проводить ідентифікацію отриманого зображення, вибирає або обчислює відповідне візуальне доповнення, об'єднує реальне зображення з його доповненням і виводить підсумкове зображення на пристрій візуалізації [11].

Для роботи з AR використовується смартфон, планшет або смарт-окуляри з відеокамерою і відповідним ПЗ. Якщо об'єктив відеокамери спрямований на об'єкт, ПЗ розпізнає його або по заздалегідь встановленому маркера, або після аналізу форми об'єкта. Після розпізнавання об'єкта, ПЗ може підключитися до тривимірного цифрового двійника об'єкта, який розміщений на сервері або в хмарі. Потім пристрій AR завантажує необхідну інформацію і накладає її на зображення об'єкта. В результаті на екрані (або через окуляри) відображається частково фізична реальність, частково цифрова. При цьому різні спостерігачі, дивлячись на один об'єкт можуть бачити різну доповнену реальність, відповідно до виконуваних функцій. Ремонтник може бачити дані про напрацювання або, припустимо, робочій температурі того чи іншого вузла, яке обслуговує. Оператору пристрій AR може допомагати управляти об'єктом за допомогою сенсорного екрану, голосом або жестами. При русі спостерігача розмір і орієнтація дисплея AR автоматично коригуються, непотрібна інформація зникає, а нова з'являється.

Цифрова модель об’єкта зазвичай створюється ще на етапі розробки об'єкта або за допомогою САПР, або шляхом оцифровки. Цей цифровий двійник збирає інформацію про стан об'єкта, що отримується від нього самого, з інформаційних систем і з зовнішніх джерел. З його допомогою програмне забезпечення доповненої реальності масштабує і точно розміщує на зображенні об'єкта або навколо нього актуальні дані.

3.4 Інструменти для розробки програмного забезпечення з використанням доповненої реальності

Основні інструменти для розробки застосунків доповненої реальності (SDK для роботи з AR) – це EasyAR, Wikitude, Google ARCore, Kudan, Vuforia [12].

EasyAR – це інструмент для розробки AR, з підтримкою основних мобільних платформ: Android, iOS, UWP, Windows, OS X и Unity. Його SDK дозволяє компаніям і розробникам розширювати можливості занурення в AR за допомогою мобільних застосунків. Розробники можуть використовувати функції EasyAR в залежності від придбаних пакетів. Набір функціональних можливостей безкоштовної ліцензії дуже обмежений.

Wikitude використовується для розробки мобільних застосунків і AR-прототипів. Новий Wikitude SDK дозволяє розробникам реалізовувати можливості геолокації, а також відстежувати зображення та розпізнавати об'єкти. Деякі з його особливостей включають в себе: 3D розпізнавання і відстеження; розпізнавання і відстеження зображень; розпізнавання в хмарі, AR на основі розташування, накладення відео, інтеграція смарт-окулярів, інтеграція з зовнішніми плагінами тощо. Wikitude не надає безкоштовної ліцензії, вартість ліцензій – від $3500 [13].

ARCore від Google заснований на двох принципах: відстеження позиції і розпізнавання об'єктів. Деякі з його особливостей: оцінка освітленості в реальному часі, точне розміщення віртуальних об'єктів, легке відстеження для створення реалістичних об'єктів, визначення розміру та місця розташування вертикальних, горизонтальних і похилих поверхонь, відстеження руху відповідно до положення телефону.

Kudan – ще один універсальний AR SDK, його основні можливості: вимоги до розташування та відстеження на основі маркерів і без них; розпізнавання зображень; Visual-SLAM; Fusion Sensor – сенсор для визначення

джерел білого світла; гнучка інтеграція; універсальна настройка; підтримка Unity SDK.

Vuforia є одним з найпопулярніших SDK для розробки застосунків доповненої реальності. Vuforia – це комплексна, масштабована корпоративна платформа доповненої реальності і інструментарій розробника програмного забезпечення доповненої реальності для мобільних пристроїв, розроблені компанією Qualcomm. Платформа була придбана компанією PTC Inc. у листопаді 2015 року. Vuforia використовує технології комп'ютерного зору, а також відстеження плоских зображень і простих об'ємних реальних об'єктів в реальному часі, розпізнає текст та має можливість розпізнавати циліндричні маркери [14].

Завдяки доступності API через Unity, Vuforia можна використовувати для розробки власних застосунків під iOS і Android. Вона також вважається повним SDK з великим набором функцій для застосунків AR: ідентифікація та відстеження цільових зображень, текстів англійською мовою і 3D-об'єктів в режимі реального часу; розміщення віртуальних об'єктів, таких як 3D-моделі, в реальному середовищі; багатоцільові 3D-конфігурації; Vuforia Engine Area Targets разом з Area Target Generator; відскановані Model Targets; розширені Model Targets; виявлення декількох моделей; продовження роботи при припиненні застосунків; режим симуляції; Vuforia Engine Tracking Scale тощо.

Основні переваги PTC Vuforia Engine [15].

Точність. Передові технології комп'ютерного зору в Vuforia Engine забезпечують неймовірну точність позиціонування AR в різних середовищах.

Можливості для творчості. Надійна технологія, пропонована в Vuforia Engine, дозволяє розробникам вільно створювати фірмові AR- застосунки для нових або вже існуючих застосунків.

Мультиплатформеність. Vuforia підтримує AR-пристрої, такі як смартфони, планшети і гарнітури на провідних платформах, щоб охопити найбільшу аудиторію.

Концепція розширеного зору. Vuforia Engine пропонує динамічне розпізнавання об'єктів і може розпізнавати зображення, 3D-моделі і середовища, забезпечуючи гнучкість розробки.

Широкі можливості для створення AR об'єктів:

- цілі у вигляді моделей. Дозволяють розпізнавати об'єкти за формою, використовуючи вже існуючі 3D-моделі;
- цілі у вигляді об'єктів. Використовують тривимірне сканування розташування і призначені для роботи в великих приміщеннях;
- цілі у вигляді зображень - найпростіший спосіб розмістити контент AR на плоских об'єктах, таких як сторінки журналів, торговельні картки і фотографії;
- мульти-цілі. Призначені для об'єктів з плоскими поверхнями і декількома сторонами або які містять кілька зображень;
- циліндричні цілі. Дозволяють розміщувати вміст AR на об'єктах циліндричної і конічної форми;
- об'єктні цілі. Створюються шляхом сканування об'єкта з багатими деталями поверхні і постійною формою;
- мітки VuMarks. Дозволяють ідентифікувати і додавати контент в серії об'єктів;
- фонові ефекти, програвання відео та віртуальні кнопки на цільових поверхнях, управління оклюзією тощо.

За допомогою інструментів PTC для розробників можливо використовувати платформу Vuforia для створення власних інтегрованих застосунків, застосунків, які працюють в режимі реального часу, або розробки графічних інтерфейсів користувача. Vuforia надає інтерфейси прикладного програмування (API) на мовах C++, Java, Objective-C++ та .NET через розширення для ігрового механізму Unity. Таким чином, SDK підтримує як власну розробку для iOS, Android, так і UWP, а також дозволяє розробляти AR- застосунки в Unity, які легко переносити на різні платформи.

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Після проведення аналізу предметної області, програмних продуктів-аналогів та вибору засобів розробки, необхідним етапом є проектування інформаційної системи. На цьому етапі необхідно створити UML-діаграми варіантів використання (прецедентів), активності і послідовності та розробити макети інтерфейсу застосунку.

4.1 Створення UML-діаграми варіантів використання системи

Діаграма варіантів використання (прецедентів) відображає функціональне призначення проектованої програмної системи. Суть діаграми прецедентів полягає в тому, що систему представляють як групу акторів, які за допомогою варіантів використання взаємодіють із нею.

Актор – це сутність, що взаємодіє з системою для вирішення деяких завдань. Актором може бути людина, інша система, пристрій або програмний засіб. В даному випадку акторами визначено користувача застосунку, адміністратора та Vuforia SDK, яка розпізнає цільові зображення. На діаграмі також зображена хмарна база даних.

Після визначення акторів системи, необхідно сформувати перелік усіх варіантів використання, з якими будуть взаємодіяти визначені актори. Серед основних варіантів використання системи: з боку користувача сканування картини за допомогою камери, перегляд інформації про картину, експозиції музею, робіт митця, авторизація в системі, додавання картини до Обраного, з боку адміністратора – додавання нових картин та інформації до них, зміна або видалення їх з бази.

За результатами сформованих варіантів використання та акторів системи було розроблено діаграму варіантів використання, яку наведено на рис. 4.1.

Рисунок 4.1 – Діаграма варіантів використання (прецедентів) системи

4.2 Створення діаграми діяльності (активностей)

Для моделювання процесу виконання операцій в мові UML використовуються діаграми діяльності. На діаграмі діяльності відображається логіка або послідовність переходу від однієї діяльності до іншої, при цьому увагу фіксується на результаті діяльності. Сам же результат може привести до зміни стану системи або повернення деякого значення. Для системи була створена діаграма діяльності, яка відображає основний процес проектованої системи – сканування зображення і його розпізнавання (рис. 4.2).

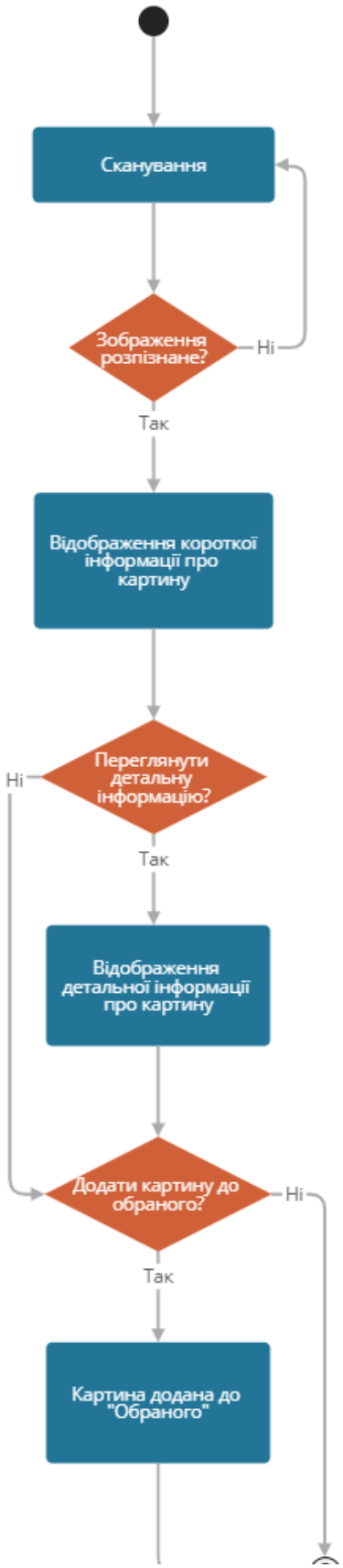


Рисунок 4.2 – Діаграма діяльності (активностей)

4.3 Створення діаграми послідовності

Також була створена діаграма послідовностей. Такі діаграми використовуються для уточнення діаграм прецедентів і більш детального опису логіки сценаріїв використання.

Головна послідовність сценарію розпізнавання зображення складається з взаємодій, які відображені на діаграмі на рис. 4.3: користувач сканує зображення за допомогою камери, отримане зображення обробляється Vuforia Engine, яка знаходить відповідну ціль у хмарній базі даних та повертає дані у застосунок, де ця інформація відображається для користувача.

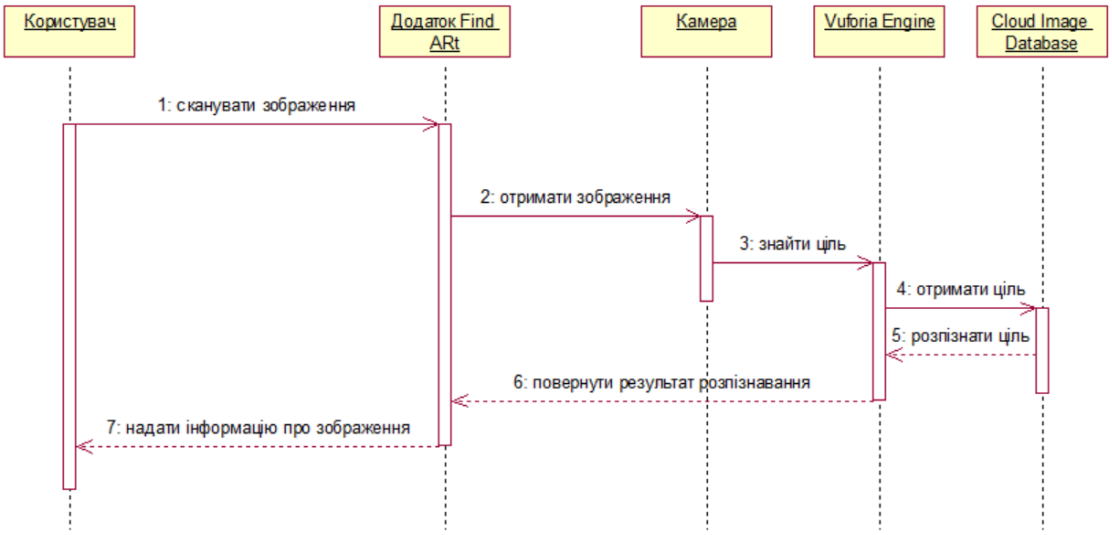


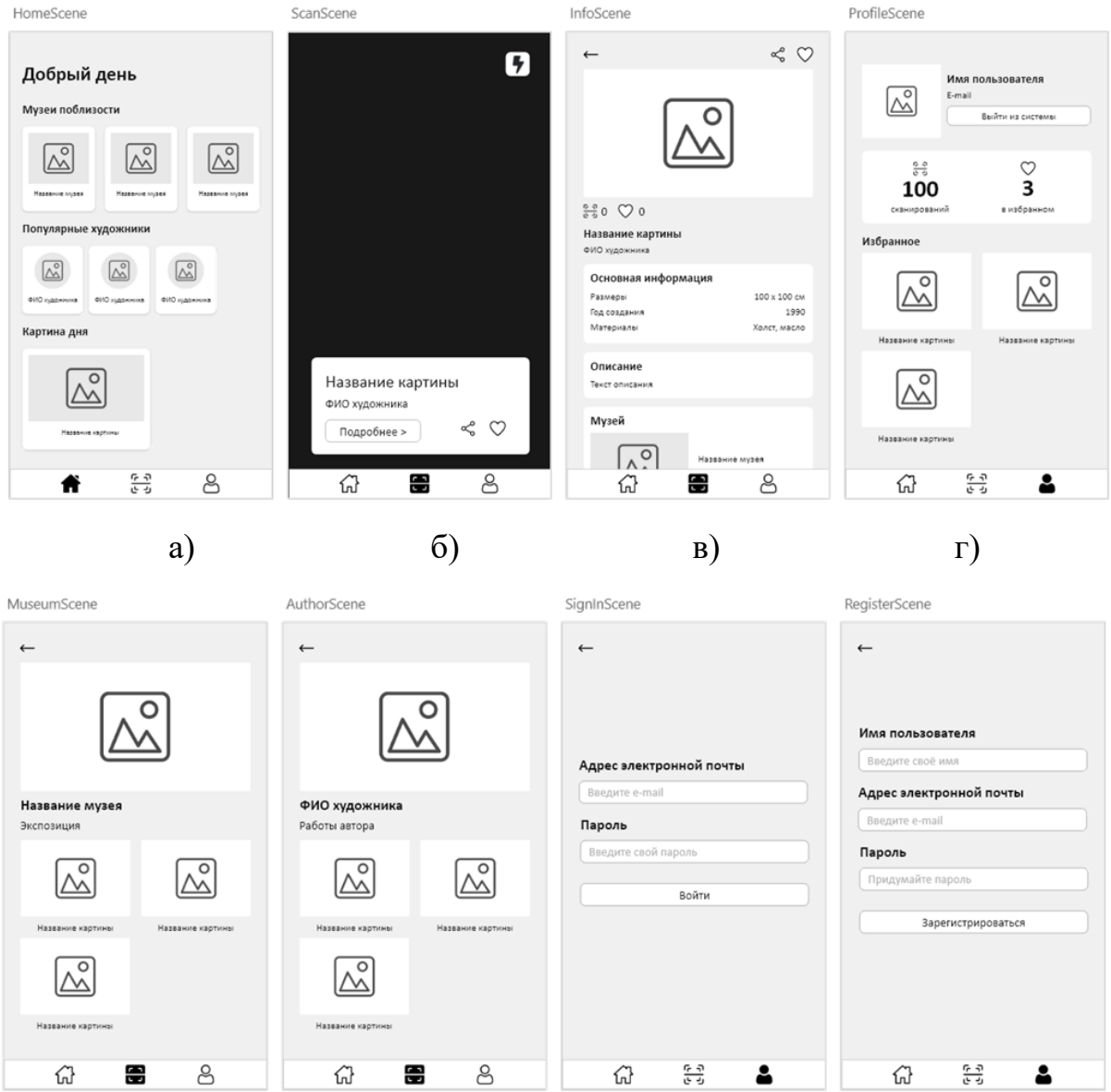
Рисунок 4.3 – Діаграма послідовностей

4.4 Проектування макетів інтерфейсу застосунку

Після цього були створені макети інтерфейсу за допомогою Adobe XD. Було вирішено розробити 8 сторінок у застосунку та зробити дизайн легким та простим. Прототипи інтерфейсу наведені на рис. 4.4.

На головній сторінці було вирішено розмістити вітання користувача та запропонувати можливість відразу перейти до перегляду експозицій музеїв

Одеси, робіт популярних митців або інформації про одну обрану з бази картину.



а) б) в) г)

г) д) е) є)

Рисунок 4.4 – Макети інтерфейсу застосунку – а) головна сторінка – б) сторінка сканування – в) сторінка з повною інформацією – г) сторінка профіля – г) сторінка експозиції музею – д) сторінка робіт автора – е) сторінка входу до системи – є) сторінка реєстрації

При скануванні на відповідній сторінці відображається коротка інформація про розпізнану картину – назва та митець. У повідомлені, що з’являється, можна натиснути «Подробнее» та перейти до сторінки з повною інформацією про картину, де вказані назва, автор, розміри, рік створення, використані матеріали, короткий опис та музей, в якому вона розташована. Можна також перейти на сторінку експозиції музею. На сторінці профіля можна увійти до системи або зареєструватись і потім переглядати своє «Обране», тобто картини, які сподобались.

Таким чином, на етапі проектування системи були створені UML-діаграми варіантів використання (прецедентів), активності і послідовності та розроблені макети інтерфейсу застосунку. Це допомогло зрозуміти основні варіанти використання системи та логіку роботи застосунку.

5 РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ «FIND ART»

5.1 Робота з Vuforia Engine в середовищі Unity

Для використання Vuforia Engine у проекті насамперед необхідно зареєструватися на порталі розробників Vuforia Engine (Vuforia Engine Developer Portal [16]) та отримати ліцензійний ключ. Створення нового ліцензійного ключа та перегляд активних можливе через менеджер ліцензій License Manager (рис. 5.1).

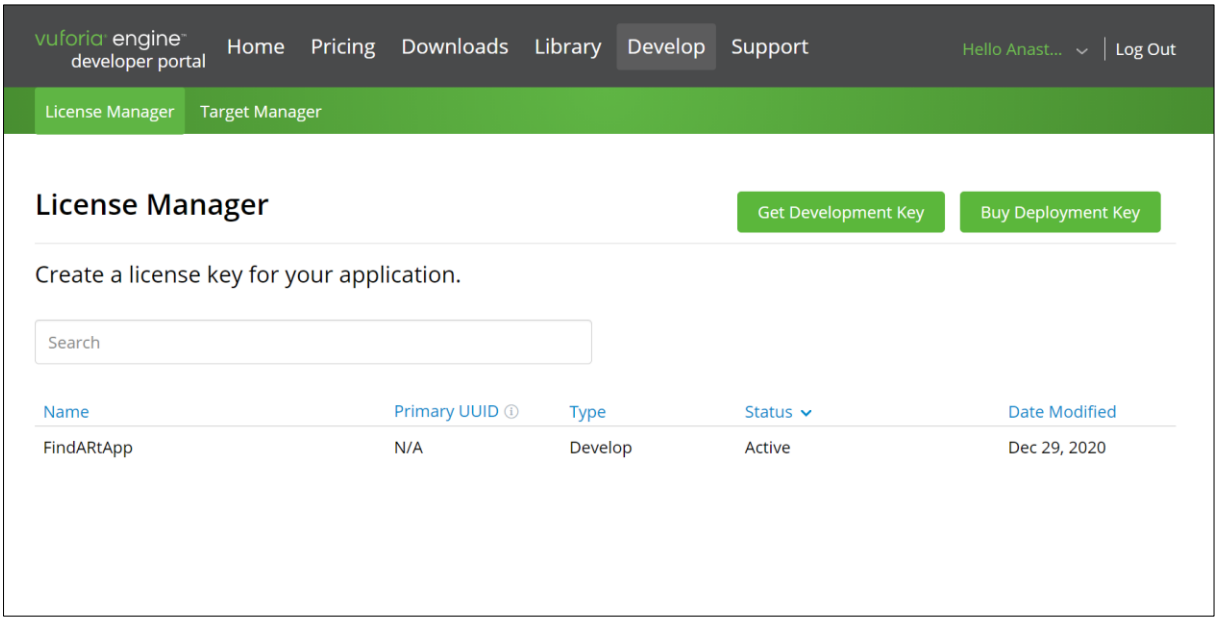


Рисунок 5.1 – Менеджер ліцензій на порталі розробників Vuforia Engine

Для використання Vuforia у Unity необхідно у проекті через Windows/Package Manager встановити пакет Vuforia Engine AR. У роботі використовується версія Vuforia Engine 8.1.12, яка є затвердженою версією для версій Unity 2019.4, а отже і найбільш стабільною. Після встановлення пакету необхідно додати до відповідної сцени (сцена сканування) об’єкт ARCamera у вікні ієрархії або через GameObject/Vuforia Engine/ARCamera. ARCamera – це об’єкт ігрової камери Unity, який включає VuforiaBehaviour для додавання підтримки застосунків доповненої реальності як для портативних пристроїв,

так і для цифрових окулярів. Після цього можна видалити камеру за замовчуванням, адже ARCamera містить власний компонент Camera.

В Інспекторі об’єкта ARCamera необхідно відкрити конфігурацію Vuforia та додати створений раніше ліцензійний ключ розробника Vuforia у поле ліцензійного ключа програми App License Key (рис. 5.2).

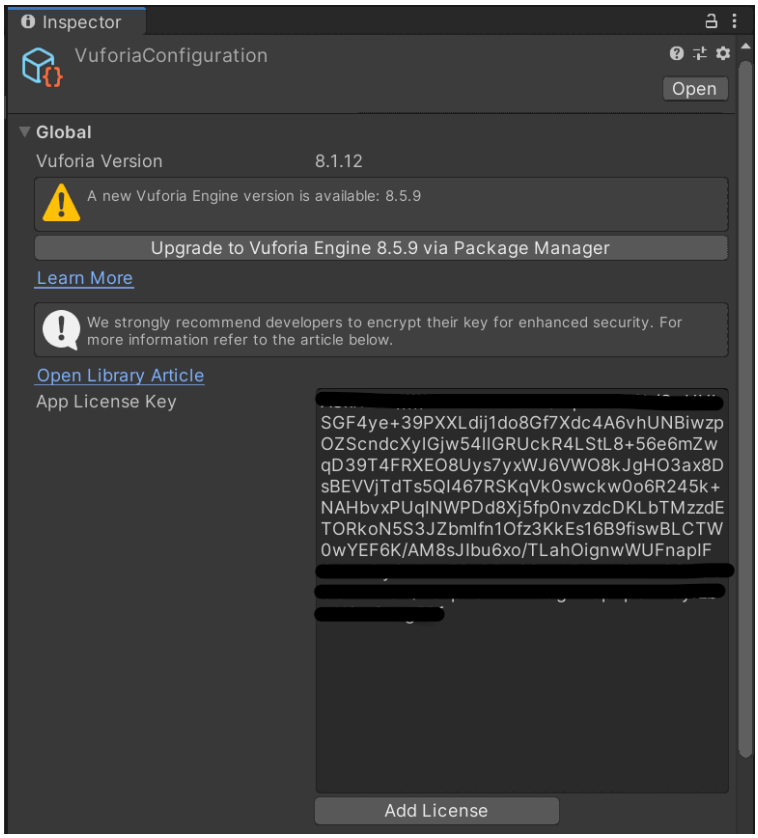


Рисунок 5.2 – Введений ліцензійний ключ програми

Наступний крок – створення бази даних. Vuforia пропонує три типи баз даних для зберігання цільових зображень: пристрою (Device), хмарні (Cloud) та бази VuMark.

Бази даних пристрою – це локальні бази даних цільових зображень або об’єктів, які зберігаються на пристрої користувача. Цей тип бази підтримує розпізнавання зображень, циліндрів мульти-цілей та об’єктних цілей. Він використовується, коли потрібно якнайшвидше розпізнавати цілі, і цільовий набір майже незмінний.

Бази даних VuMark – це локальні бази даних VuMarks, які зберігаються на пристрої користувача. Використовуються при роботі з VuMark (спеціальними зображеннямиVuforia, що поєднують QR-код та зображення).

Хмарні бази даних – це бази даних цільових зображень, що зберігаються в Інтернеті та запитуються через Інтернет. Використовуються, коли є велика кількість цільових зображень та потрібна можливість активного управління цільовим набором в Інтернеті. Хмарні бази даних не підтримують циліндрові або мульти-цілі.

З огляду на необхідність зберігання великої кількості цільових зображень для застосунку та постійне додавання даних було вирішено використовувати хмарну базу даних для зберігання цілей застосунку FindARt.

Для цього необхідно створити базу типу Cloud на порталі розробників Vuforia через менеджер цілей Target Manager і обрати ліцензійний ключ, який буде з нею пов’язаний (рис. 5.3).

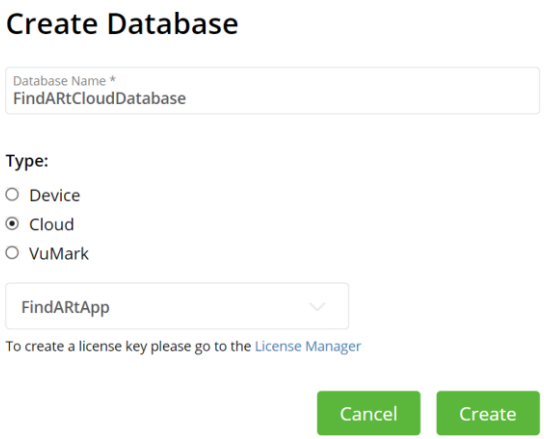


Рисунок 5.3 – Створення хмарної бази даних

В результаті буде створена хмарна база даних, що має декілька ключів доступу, які знадобляться пізніше (рис. 5.4).

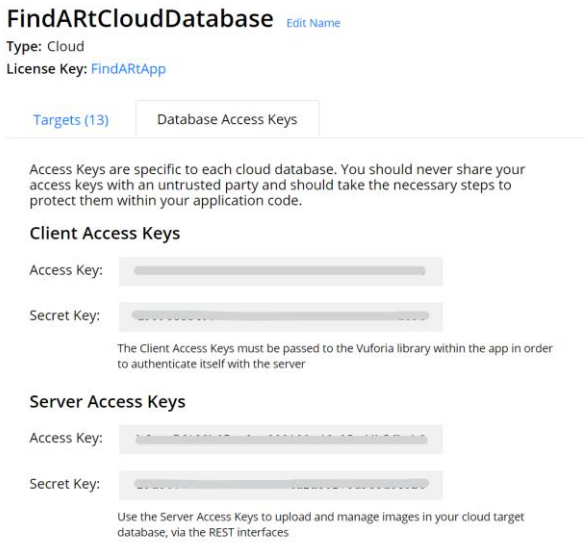


Рисунок 5.4 – Створена база даних з ключами

До сцени сканування необхідно додати об’єкти CloudRecognition та ImageTarget. ImageTarget – це зображення, які Vuforia Engine може виявляти та відстежувати. У вікні Інспектора об’єкта CloudRecognition необхідно ввести ключі хмарної бази даних у поля Ключ доступу (Access Key) та Секретний ключ (Secret Key) (рис. 5.5).

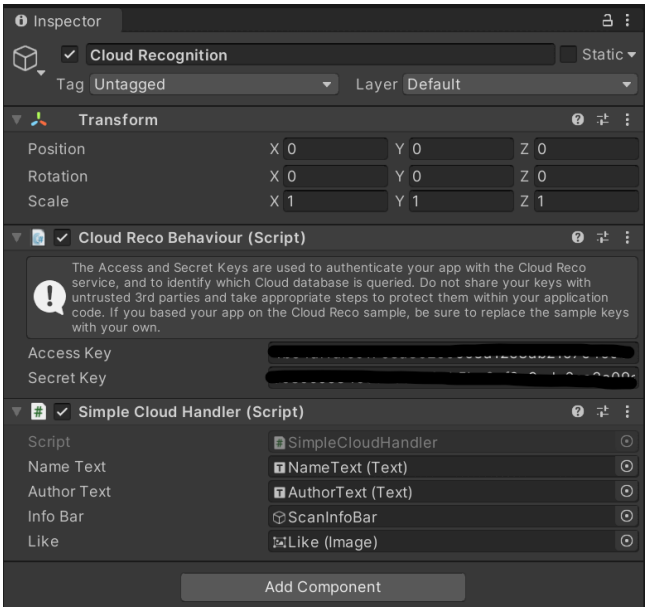


Рисунок 5.5 – Вікно Інспектора об’єкта CloudRecognition

Також необхідно створити скрипт C# та прикріпити його до об’єкта CloudRecognition. В цьому скрипті ініціалізується об’єкт класу CloudRecoBehaviour. Це основний клас поведінки, який інкапсулює поведінку хмарного розпізнавання Cloud Recognition. Він в свою чергу ініціалізуватиме пошук цілей (target finder) шукач і буде чекати нових результатів. Зміни стану та нові результати розпізнавання відстежуються за допомогою метода OnNewSearchResult(). Ім’я розпізнаного цільового зображення можна отримати, отримавши атрибут TargetName об’єкта класу TargetFinder.CloudRecoSearchResult.

```
public void OnNewSearchResult(TargetFinder.TargetSearchResult
targetSearchResult){
    TargetFinder.CloudRecoSearchResult cloudRecoSearchResult =
(TargetFinder.CloudRecoSearchResult) targetSearchResult;
    picName = cloudRecoSearchResult.TargetName; ...
}
```

Повний код застосування наведений у Додатку А.

Додавати цільові зображення до хмарної бази даних можна через Менеджер цілей на порталі розробників Vuforia Engine. Для цього необхідно обрати базу та натиснути «Add Target». При додаванні цілі необхідно завантажити зображення, вказати його ширину, за бажанням додати файл з метаданими та вказати ім’я цільового зображення.

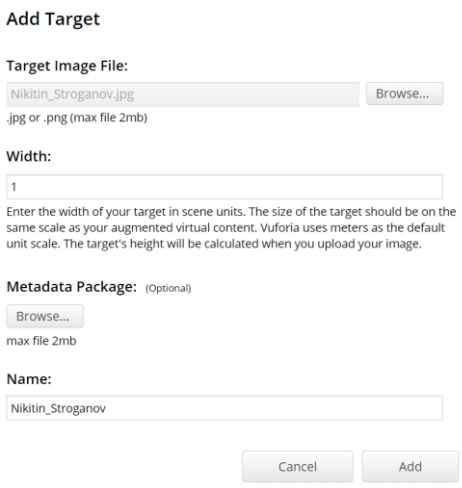


Рисунок 5.6 – Вікно додавання цільового зображення до хмарної бази

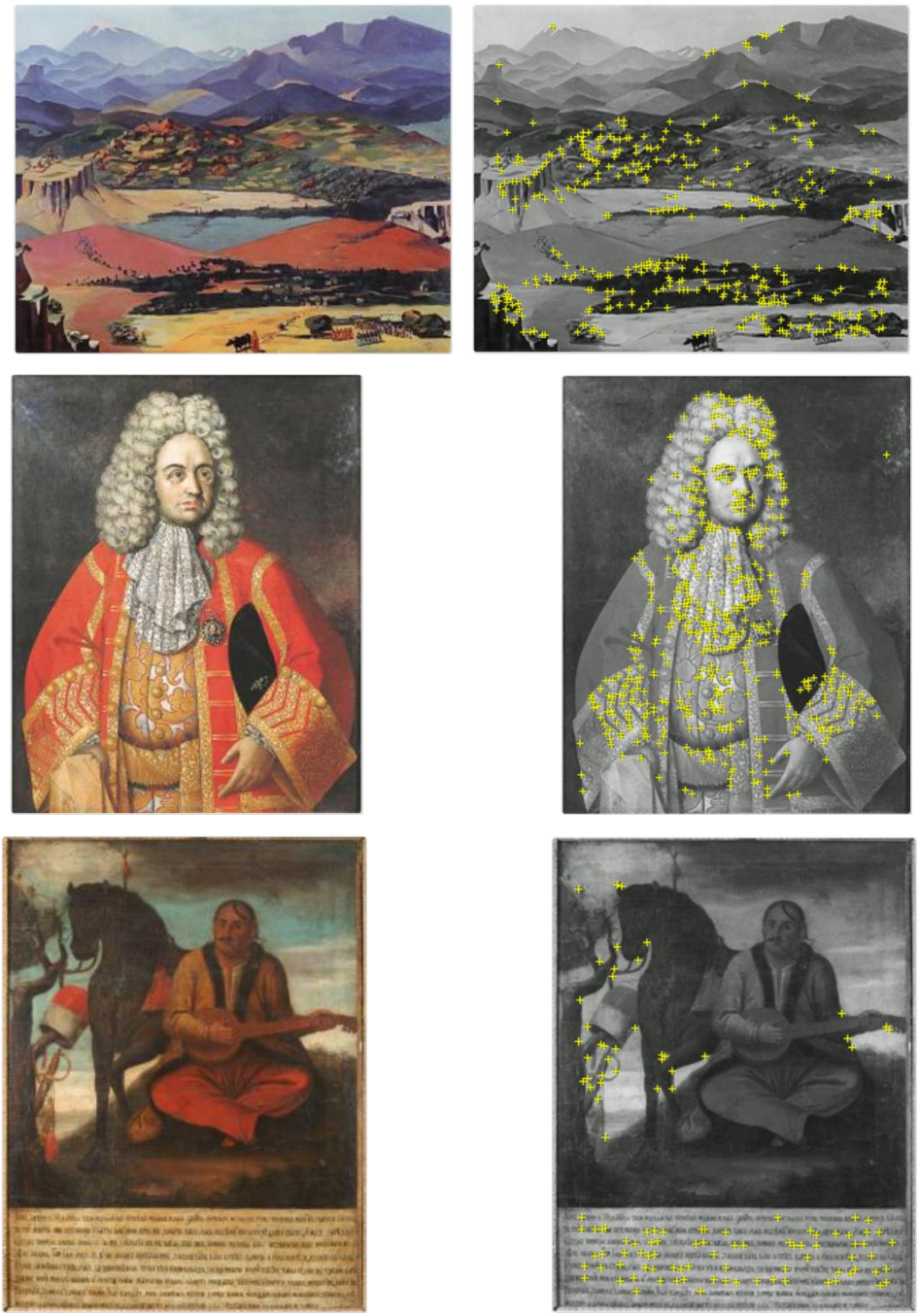
Цільові зображення повинні бути форматів JPG або PNG, у форматі RGB або у відтінках сірого (не CMYK). Розмір вхідних зображень повинен становити не більше 2,25 МБ і мати мінімальну ширину 320 пікселів. Для найкращої ефективності виявлення та відстеження Vuforia Engine також рекомендовано, щоб цільові зображення містили багато деталей та мали хороший контраст.

Цільові зображення виявляються на основі природних особливостей, які виділяються з цільового зображення, а потім порівнюються із зображенням камери в реальному часі. Рейтинг цілі може коливатися від 1 до 5 зірок. Хоча цілі з низьким рейтингом (1 або 2 зірки) зазвичай теж виявляються і відстежуються, для найкращих результатів ліпше використовувати цілі з 4 або 5 зірками.

Для кожного цільового зображення можна переглянути автоматично виявлені природні особливості (рис. 5.7).

У випадку застосунку FindARt до бази цільових зображень завантажуються об’єкти мистецтва – вже готові картини, на зміст яких розробник не може вплинути. Однак для отримання кращих рейтингів можна постаратися знайти більш якісні та контрастні зображення однієї і тієї ж картини. Це не завжди можливо, отже не всі картини-цільові зображення матимуть високий рейтинг (рис. 5.8).

Кожне цільове хмарне зображення може додатково мати пов'язані метадані. Метадані цілі - це визначені користувачем дані, які можуть бути пов’язані з ціллю та заповнені спеціальною інформацією до дозволених обмежень (до 2 МБ на ціль). Метадані можуть містити просте текстове повідомлення, яке відображатиметься на екрані пристрою при виявленні цілі; простий рядок URL-адреси, який вказуватиме на спеціальне мережеве розташування, де зберігається якийсь інший вміст, наприклад, 3D-модель, відео, зображення або будь-які інші користувацькі дані; деякий спеціальний текст, який програма може обробити та використовувати для виконання певних дій, наприклад, представлення об’єкта у форматі JSON.



а) б)

Рисунок 5.7 – Цільові зображення – а) завантажене зображення – б) виявлені природні особливості для розпізнавання

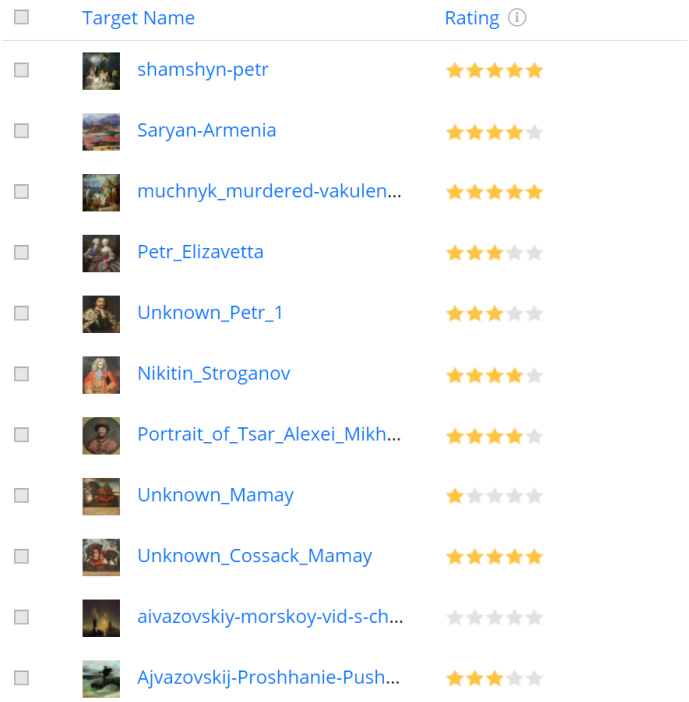


Рисунок 5.8 – Рейтинги деяких цільових зображень бази застосунку FindARt

Метадані можна завантажувати з цільовим зображенням під час створення цілі у хмарній базі даних або оновити метадані існуючої цілі пізніше через менеджер цілей. Для цього необхідно видалити поточний файл з метаданими та завантажити новий текстовий файл з оновленими даними. Для однієї цілі це може зайняти до декількох хвилин. Зрозуміло, що для постійного оновлення даних, додавання інформації про картини це дуже незручно. Тому було вирішено зберігати данні про картини не як метадані цільових зображень у хмарній базі даних Vuforia, а у хмарній базі Firebase Database.

5.2 Використання Firebase

Firebase – це платформа, розроблена для створення мобільних та веб-застосунків. Вона заснована в 2011 році і придбана Google у 2014 році.

На платформі Firebase 18 продуктів, розділених на три групи: Розробка, Випуск та моніторинг та Залучення (рис. 5.9).

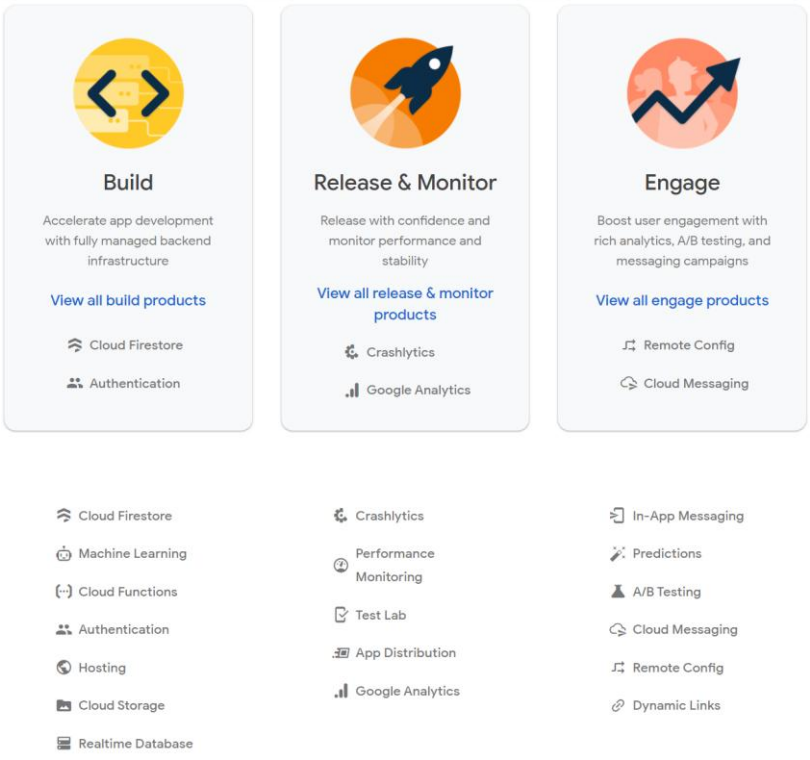


Рисунок 5.9 – Продукти платформи Firebase

База даних Firebase Realtime (Firebase Realtime Database) – це база даних, розміщена в хмарі. Дані зберігаються в форматі JSON і синхронізуються в реальному часі з кожним підключеним клієнтом.

База даних Firebase Realtime дозволяє створювати багатофункціональні програми для спільної роботи, забезпечуючи безпечний доступ до бази даних безпосередньо з клієнтського коду. Дані зберігаються локально, і навіть в автономному режимі події в реальному часі продовжують активуватися, що дає кінцевому користувачу можливість швидко реагувати. Коли пристрій відновлює з'єднання, база даних реального часу синхронізує зміни локальних даних з віддаленими оновленнями, які відбулися, коли клієнт був в автономному режимі, автоматично об'єднуючи будь-які конфлікти.

База даних Realtime надає гнучку мову правил Firebase Realtime Database Security Rules на основі виразів для визначення того, як дані повинні бути структуровані і коли вони можуть бути прочитані або записані. При ін-

теграції з Firebase Authentication розробники можуть визначати, хто має доступ до яких даних.

Firebase Realtime Database – це база даних NoSQL, тому вона відрізняється оптимізацією і функціональністю в порівнянні з реляційною базою даних. API бази даних реального часу призначені тільки для операцій, які можуть виконуватися швидко. Це дозволяє створювати відмінні умови для роботи в реальному часі, які можуть обслуговувати мільйони користувачів без шкоди для швидкості реагування.

Використання більшості продуктів Firebase безкоштовне, але з деякими обмеженнями. Наприклад, у Realtime Database можна зберігати не більше 1 ГБ даних та завантажувати не більше 10 ГБ щомісяця. Цього більше, ніж достатньо для невеликих проектів: близько 100 записів у базі займають лише 5 Кб.

Перш ніж додати Firebase в проект Unity, необхідно створити проект Firebase. Для цього необхідно зайти до Консолі Firebase [17] через свій аккаунт Google та натиснути «Додати проект» (рис. 5.10).

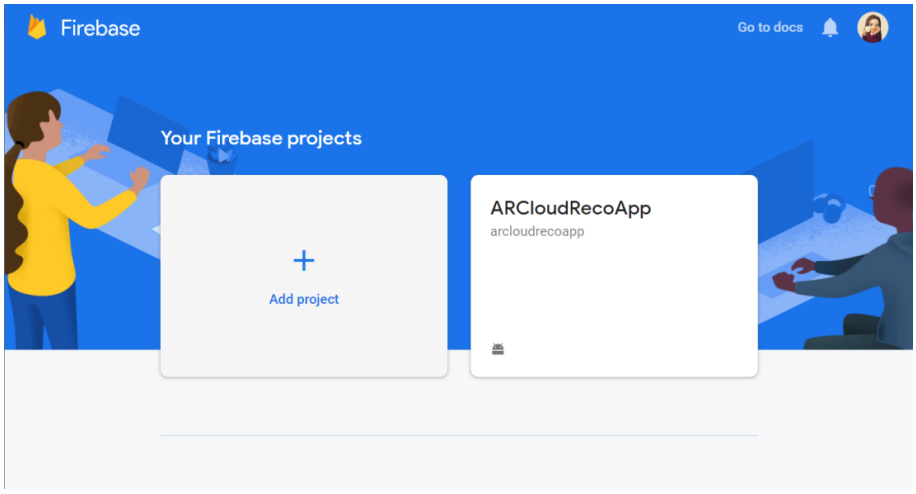


Рисунок 5.10 – Firebase Console

До існуючого проекту необхідно додати застосунок. Для цього в Unity в Build Settings/Player Settings/Other Settings необхідно скопіювати назву пакета та зареєструвати його у створеному проекті Firebase як застосунок для

Android. Firebase згенерує файл конфігурації проекту google-services.json, який необхідно помістити до папки Assets проекту в Unity.

При додаванні застосунку до проекту, який розроблюється за допомогою платформи Unity, необхідно також завантажити Firebase Unity SDK (близько 2 ГБ). Це файли, необхідні для роботи продуктів платформи Firebase в Unity. Через меню Assets/Import Package/Custom Package можна імпортувати тільки потрібні продукти.

Хоча база даних Realtime Database – нереляційна, основні сутності та їх відносини можна представити на схемі даних (рис. 5.11).

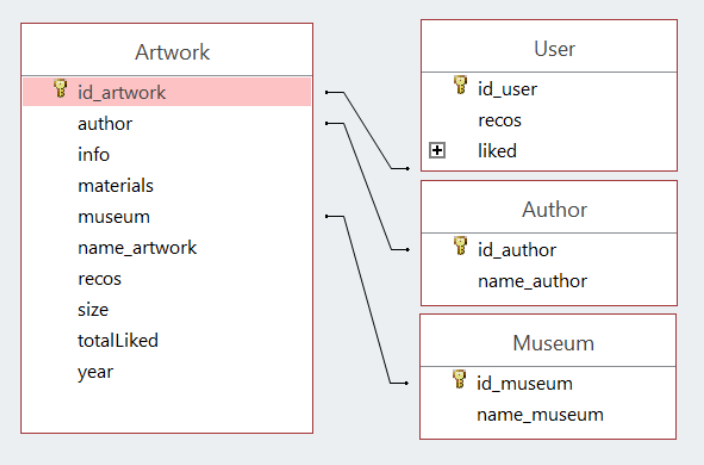


Рисунок 5.11 – Схема даних, створена в MS Access

Дані в Firebase Realtime Database зберігаються в форматі JSON. Приклад опису об’єкта у форматі JSON:

```
{
    "author" : "8",
    "materials" : "Холст, масло",
    "museum" : "1",
    "name" : "Армения",
    "size" : "168 x 132 см",
    "year" : "1957",
}
```

У випадку з базою даних картин використання бази типу NoSql зручніше за реляційні бази даних. При зберіганні даних у форматі JSON можна не турбуватись про визначення повної структури кожного елемента або про взаємодію типів даних, що зберігаються. Наприклад, не у всіх об’єктів Artwork (картин) може бути поле з описом або відомі матеріали чи розміри оригіналу. Також рік створення можна записати числом, а можна вказати текстом, наприклад «до 1715 року».

У роботі також використовується продукт Firebase Storage для зберігання зображень користувачів, портретів авторів, музеїв та картин для відображення у застосунку і Firebase Authentication для роботи з користувачами.

Було створено 8 сцен застосунку (рис. 5.12):

– AuthorScene – сторінка з роботами автора,
– HomeScene – головна сторінка застосунку,
– InfoScene – сторінка з додатковою інформацією про картину,
– MuseumScene – сторінка експозиції музею,
– ProfileScene – сторінка профіля користувача,
– RegistrationScene – сторінка реєстрації,
– ScanScene – сторінка сканування,
– SignInScene – сторінка входу до системи.

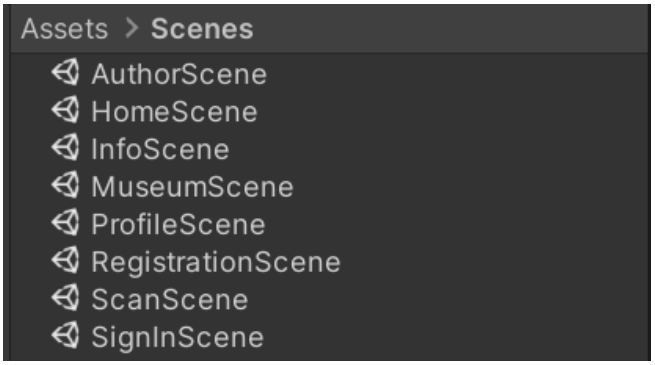


Рисунок 5.12 – Сцени застосунку «FindARt» в Unity

5.3 Опис та тестування застосунку

Значок застосунку зображений на рис. 5.13. Робота з застосунком починається з головної сторінки (рис. 5.14).

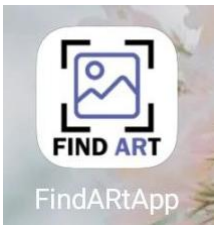


Рисунок 5.13 – Значок застосунку «FindARt»

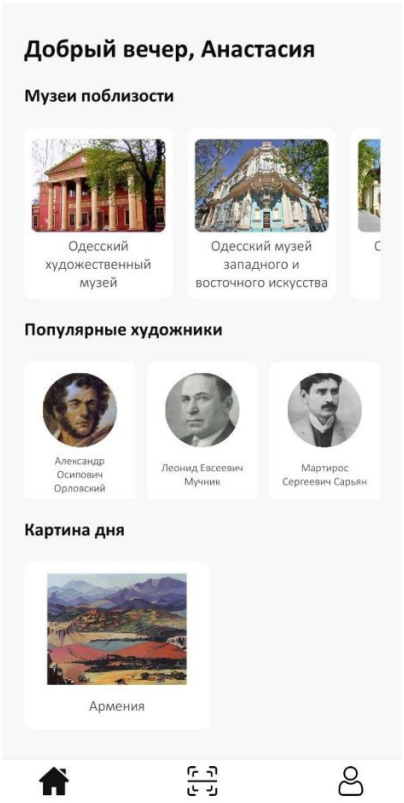


Рисунок 5.14 – Головна сторінка застосунку «FindARt»

Користувач може обрати один з музеїв та переглянути його експозицію (рис. 5.15).

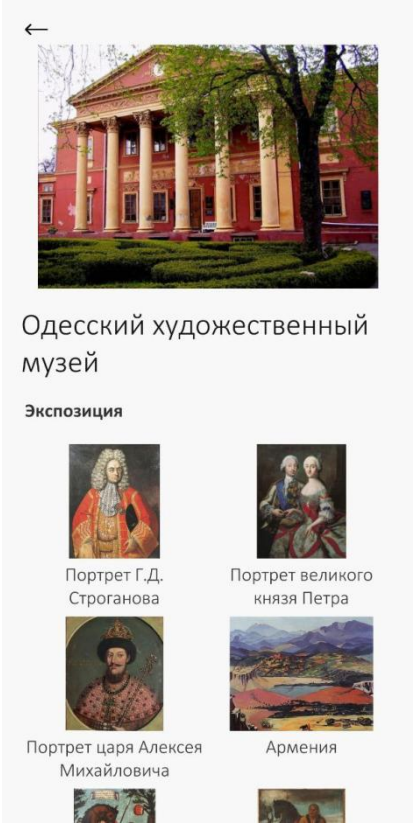


Рисунок 5.15 – Сторінка експозиції музею

З головної сторінки користувач також може перейти до перегляду робіт одного з популярних авторів (рис. 5.16).

На головній сторінці застосунку також відображається найбільш популярна картина з бази даних, яка має найбільшу кількість лайків. За допомогою нижнього меню можна перейти на сторінку сканування та профіля користувача.

На сторінці профіля користувач може зареєструватися або увійти до системи, якщо він ще цього не зробив. Після цього для нього доступна статистика його взаємодії з застосунком: загальна кількість картин, які він розпі-

знав, та кількість картин, які йому сподобались. Ці картини він може переглянути у своєму «Обраному» (рис. 5.17).

Рисунок 5.16 – Сторінка робіт автора

Рисунок 5.17 – Профіль користувача після входу до системи

При переході на сторінку сканування завантажується AR камера. Є можливість ввімкнути ліхтарик. Водяний знак Vuforia, що свідчить про безкоштовну ліцензію, майже не видно, бо його закриває нижнє меню застосунку (рис. 5.18).

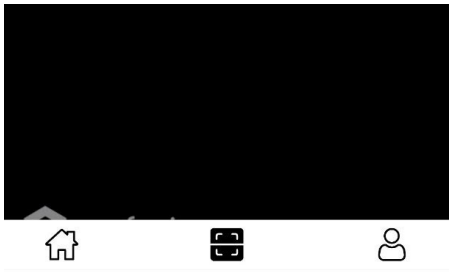


Рисунок 5.18 – Водяний знак Vuforia

При розпізнаванні зображення на сторінці сканування з’являється коротка інформація про розпізнану картину – назва та митець (рис. 5.19).

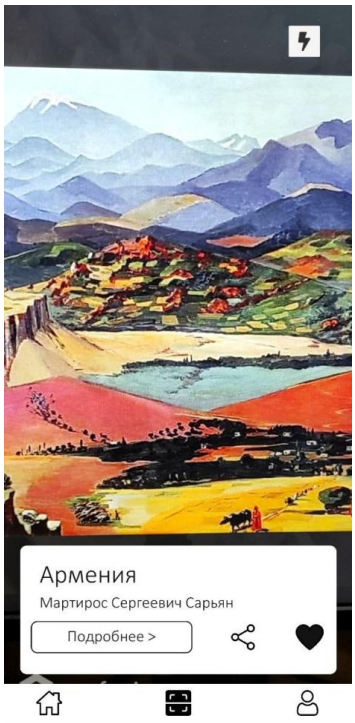


Рисунок 5.19 – Коротка інформація про розпізнану картину

Користувач може додати картину до «Обраного» або відправити інформацію про неї через соціальні мережі. Користувач також може переглянути більше відомості про картину, натиснувши «Подробнее». Відкривається сторінка з детальною інформацією про картину (рис. 5.20). Через неї також можна додати картину до «Обраного» або поділитися інформацію про неї через інші застосунки.

Під час карантину не було можливості протестувати роботу застосунку в реальному музеї, однак на веб-сайті Одеського художнього музею [18] можна переглянути всю експозицію онлайн у формі 3D-туру та «пройти» всіма залами. Застосунок «FindARt» був успішно використаний у цьому віртуальному музеї та були розпізнані всі додані до бази картини для тестування (рис. 5.21).

Рисунок 5.20 – Сторінка детальної інформації про картину

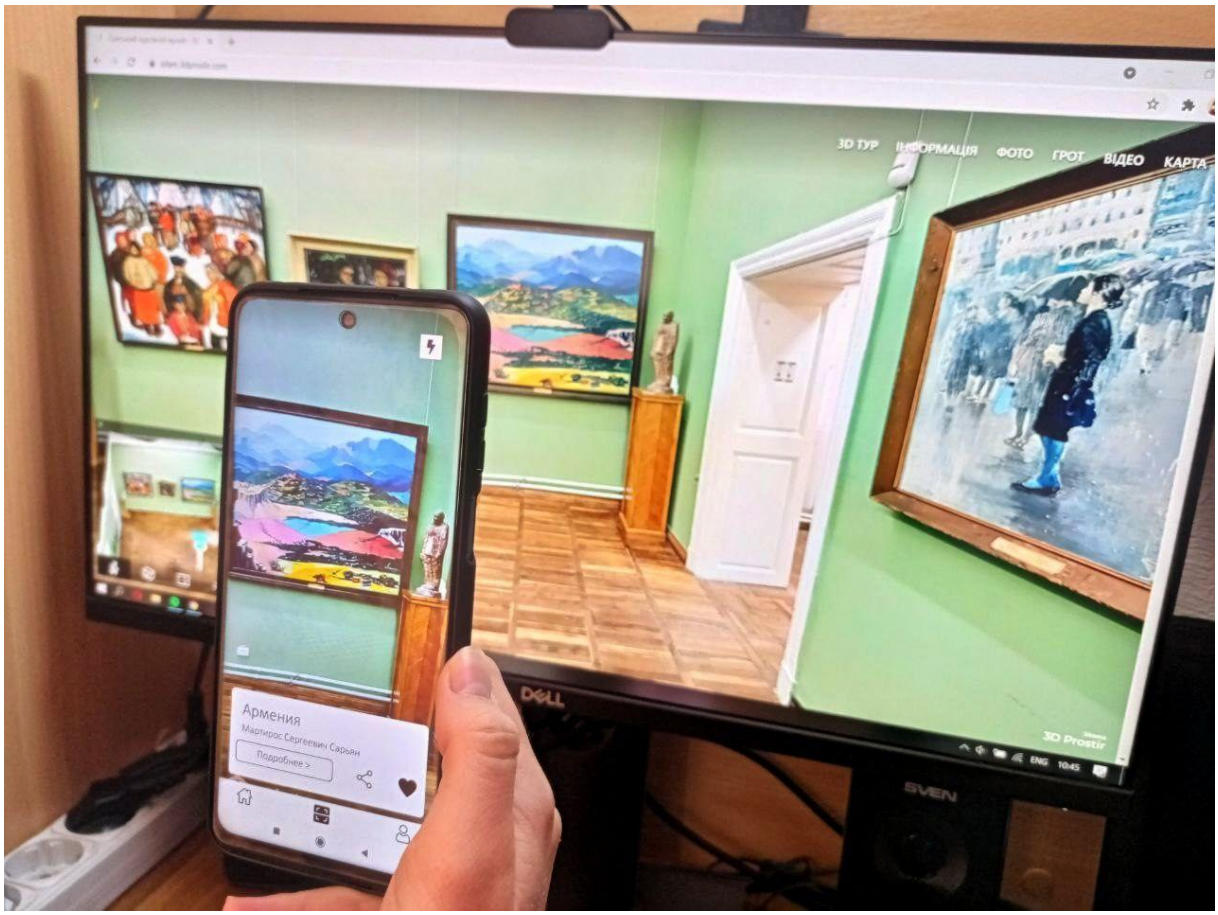


Рисунок 5.21 – Тестування роботи застосунку у музеї

Звичайно, застосунок «FindARt» можна використовувати не лише в музеях. Якщо розширити базу даних картин, користувач зможе розпізнавати предмети мистецтва, які надруковані на різних речах, наприклад, на сумках або блокнотах. Для перевірки до бази була додана картина «Зоряна ніч» Вінсента ван Гога, яку часто зображують на таких речах. На рисунку 5.22 зображений результат тестування розпізнавання цієї картини з сумки в інтернет-магазині. Звичайно, в такому випадку є деякі обмеження: поверхня предмету бажано повинна бути рівною, а зображення має бути максимально повним для коректного розпізнавання Vuforia.

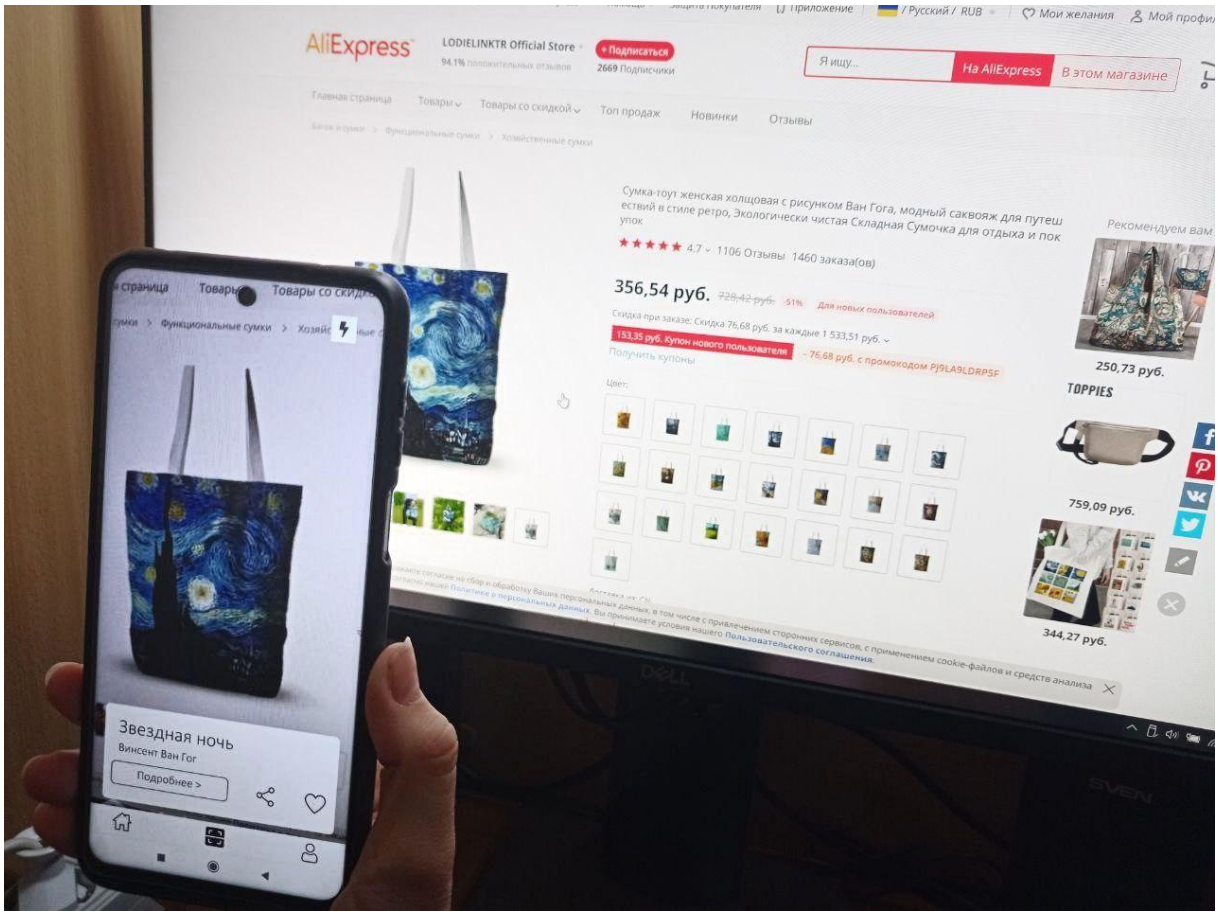


Рисунок 5.22 – Тестування роботи застосунку в інтернет-магазині

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра був створений мобільний застосунок «FindARt» для розпізнавання об’єктів мистецтва з використанням технології доповненої реальності. Для цього був проведений аналіз предметної області та порівняльний аналіз існуючих програм-аналогів у Play Market. В якості програмного засобу розробки було обрано середовище Unity з огляду на те, що більшість платформ для роботи з технологіями доповненої реальності призначені саме для Unity. Інструментом для розробки ПЗ з використанням AR було обрано Vuforia Engine. Вона вважається повним SDK з великим набором функцій для застосунків AR: ідентифікація та відстеження цільових зображень, текстів англійською мовою і 3D-об'єктів в режимі реального часу; розміщення віртуальних об'єктів, таких як 3D-моделі, в реальному середовищі; багатоцільові 3D-конфігурації; Vuforia Engine Area Targets разом з Area Target Generator; відсканованні Model Targets; розширені Model Targets; виявлення декількох моделей; продовження роботи при припиненні застосунків; режим симуляції; Vuforia Engine Tracking Scale тощо.

На етапі проектування були створені UML-діаграми варіантів використання (прецедентів), активності і послідовності та розроблені макети інтерфейсу застосунку. З огляду на необхідність зберігання великої кількості цільових зображень для застосунку та постійне додавання даних було вирішено використовувати хмарну базу даних Vuforia для зберігання цілей застосунку FindARt та інструмент розпізнавання з неї – Vuforia Cloud Recognition. Використання цього сервісу безкоштовне, але має деякі обмеження.

В роботі представлені можливості застосунку та поетапний приклад його використання. Було виконано тестування роботи застосунку у музеї та розпізнавання картин, які надруковані на різних речах. Проблем при використанні виявлено не було.

Результати кваліфікаційної роботи були приставлені на Міжнародній студентській конференції в університеті AGH (м.Краків, Польща). Презента-

ція докладу: «Development of a mobile application for museum tours using augmented reality technologies» отримала дипломом ІІ ступеня (додаток Б). Також результати роботи доповідалися на Студентській науковій конференції ОДЕКУ 19 – 23 квітня 2021.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Искусство: его суть, виды, жанры и история. URL: https://veryimportantlot.com/ru/news/blog/chto-takoe-iskusstvo (дата звернення 14.04.2021)
2. Мистецтво – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Мистецтво (дата звернення 14.04.2021)
3. Самые популярные мобильные операционные системы с 2000 по 2020 год. URL: https://www.youtube.com/watch?v=vr4q8u-asnI (дата звернення 14.04.2021)
4. Mobile Operating System Market Share Worldwide. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення 14.04.2021)
5. Объективное сравнение iOS и Android в 2019 году. URL: https://ilounge.ua/review/obektivnoe-sravnenie-ios-i-android-v-2019-godu (дата звернення 15.04.2021)
6. В третьем квартале 2020 года число загрузок из Google Play достигло 28,3 млрд. URL: https://www.ixbt.com/news/2020/10/15/2020-google-play-28-3.html (дата звернення 15.04.2021)
7. Android Studio: среда разработки мобильных приложений. URL: https://arduinoplus.ru/android-studio (дата звернення 15.04.2021)
8. Офіційний веб-сайт Unity. URL: https://unity.com/ru/ (дата звернення 15.04.2021)
9. Augmented Reality – Wikipedia. URL: https://en.wikipedia.org/wiki/Augmented_reality (дата звернення 17.04.2021)
10. Технология дополненной реальности AR. URL: https://funreality.ru/technology/augmented_reality/ (дата звернення 17.04.2021)

11. Дополненная реальность (AR). URL: https://www.it.ua/ru/knowledge-base/technology-innovation/dopolnennaja-realnost-ar (дата звернення 17.04.2021)
12. Топ 18 SDK для работы с AR. URL: https://medium.com/@grifer163/топ-18-sdk-для-работы-с-ar-44288137af76 (дата звернення 17.04.2021)
13. Wikitude Products. URL: https://www.wikitude.com/store/ (дата звернення 17.04.2021)
14. Vuforia: ведущее корпоративное решение дополненной реальности на рынке. URL: https://www.ptc.com/ru/products/vuforia (дата звернення 17.04.2021)
15. 600 000 + разработчиков выбрали Vuforia Engine. URL: https://www.irisoft.ru/products/produkty-vuforia-dopolnennaya-realnost-dlya-promyshlennosti/vuforia-engine/ (дата звернення 17.04.2021)
16. Vuforia Engine Developer Portal. URL: https://www.developer.vuforia.com (дата звернення 23.03.2021)
17. Firebase Console. URL: https://www.console.firebase.google.com (дата звернення 23.03.2021)
18. Одеський художній музей. 3D-тур. URL: https://ofam.3dprostir.com (дата звернення 29.04.2021)

ДОДАТОК А

Вихідний код програми

ARCameraManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class ARCameraManager : MonoBehaviour
{
    void Start()
    {
VuforiaARController.Instance.RegisterVuforiaStartedCallback(OnVu
foriaStarted);

VuforiaARController.Instance.RegisterOnPauseCallback(OnPaused);
    }

    private void OnVuforiaStarted()
    {
CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_
MODE_CONTINUOUSAUTO);
    }

    private void OnPaused(bool paused)
    {
        if (!paused) // resumed
        {
CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_
MODE_CONTINUOUSAUTO);
        }
    }
}
```

AuthorInfoLoader.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Firebase.Auth;
using Firebase.Database;
using Firebase.Storage;
using Firebase.Extensions;
using UnityEngine.EventSystems;

public class AuthorInfoLoader : MonoBehaviour
{
    public Image authImg;
    public Text authName;
    private StorageReference stRef;
    private DatabaseReference dbRef;
    private FirebaseUser user;
    private ArrayList keysArray;
```

```
    private ArrayList picsArray;
    private string authid;
    public GameObject expoPic;
    public GameObject group;
    public ScrollRect scrollRect;
    private int picsNum = 0;

    void Start()
    {
        stRef = FirebaseStorage.DefaultInstance.RootReference;
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        user = FirebaseAuth.DefaultInstance.CurrentUser;

        authid = Scenes.getParam("authid");
        stRef.Child("authors").Child(authid                    +
".jpg").GetBytesAsync(2            *            1024            *
1024).ContinueWithOnMainThread(task =>
        {
            if (task.IsFaulted || task.IsCanceled)
            {
                Debug.Log(task.Exception.ToString());
            }
            else
            {
                byte[] fileContents = task.Result;
                Texture2D texture = new Texture2D(1, 1);
                texture.LoadImage(fileContents);
                Sprite   sprite   =   Sprite.Create(texture,   new
Rect(0.0f,    0.0f,    texture.width,    texture.height),    new
Vector2(0.5f, 0.5f), 100.0f);
                authImg.sprite = sprite;
            }
        });

dbRef.Child("authors").GetValueAsync().ContinueWithOnMainThread(
task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                authName.text                                    =
snapshot.Child(authid).GetValue(true).ToString();
            }
        });
        LoadExpo();
    }

    private void LoadExpo()
    {
        dbRef.GetValueAsync().ContinueWithOnMainThread(task =>
        {
            if (task.IsFaulted)
            {
```

```
            Debug.LogWarning("Ошибка при получении данных из
БД");
        }
        else if (task.IsCompleted)
        {
            DataSnapshot snapshot = task.Result;

            picsArray = new ArrayList();
            keysArray = new ArrayList();
            IEnumerable<DataSnapshot>       allPics       =
snapshot.Child("artworks").Children;
            foreach (DataSnapshot pic in allPics)
            {
                if
(snapshot.Child("artworks").Child(pic.Key.ToString()).HasChild("
author"))
                {
                    string              auth              =
snapshot.Child("artworks").Child(pic.Key.ToString()).Child("auth
or").GetValue(true).ToString();
                    if (auth.Equals(authid))
                    {
                        string          nameOfPic         =
snapshot.Child("artworks").Child(pic.Key.ToString()).Child("name
").GetValue(true).ToString();
                        picsArray.Add(nameOfPic);
                        keysArray.Add(pic.Key.ToString());
                        picsNum++;
                    }
                }
            }
            Text                  nametext1               =
expoPic.transform.Find("Text").GetComponent<Text>();
            if (picsNum == 0)
            {
                nametext1.text = "К сожалению, тут пока пус-
то";
            }
            else
            {

expoPic.GetComponent<Button>().onClick.AddListener(ShowInfoScene
);
                nametext1.text = picsArray[0].ToString();
                stRef.Child(keysArray[0]                    +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                    ContinueWithOnMainThread(task1 =>
                    {
                        if        (task1.IsFaulted        ||
task1.IsCanceled)
                        {

Debug.Log(task1.Exception.ToString());
                        }
                        else
                        {
                            byte[]       fileContents       =
task1.Result;
```

```
                                Texture2D    texture    =    new
Texture2D(1, 1);
                                texture.LoadImage(fileContents);
                                Sprite          sprite          =
Sprite.Create(texture,   new   Rect(0.0f,   0.0f,   texture.width,
texture.height), new Vector2(0.5f, 0.5f), 100.0f);
                                Image           image           =
expoPic.transform.Find("Image").GetComponent<Image>();
                                image.sprite = sprite;

                            }
                        });

                    RectTransform              rt1              =
(RectTransform)group.transform;
                    rt1.sizeDelta = new Vector2(rt1.sizeDelta.x,
Mathf.CeilToInt(picsNum / 2.0f) * 440);

                    for (int i = 0; i < picsNum - 1; i++)
                    {
                        GameObject          cloneLiked          =
Instantiate(expoPic);

cloneLiked.transform.SetParent(group.transform, false);
                        cloneLiked.name = "expo" + (i + 1);


cloneLiked.GetComponent<Button>().onClick.AddListener(ShowInfoSc
ene);

                        Text               nametext              =
cloneLiked.transform.Find("Text").GetComponent<Text>();
                        nametext.text     =     picsArray[i     +
1].ToString();

                        stRef.Child(keysArray[i     +     1]     +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                        ContinueWithOnMainThread(task1 =>
                        {
                            if        (task1.IsFaulted        ||
task1.IsCanceled)
                            {

Debug.Log(task1.Exception.ToString());
                            }
                            else
                            {
                                byte[]       fileContents       =
task1.Result;
                                Texture2D    texture    =    new
Texture2D(1, 1);
                                texture.LoadImage(fileContents);
                                Sprite          sprite          =
Sprite.Create(texture,   new   Rect(0.0f,   0.0f,   texture.width,
texture.height), new Vector2(0.5f, 0.5f), 100.0f);
                                Image           image1          =
cloneLiked.transform.Find("Image").GetComponent<Image>();
                                image1.sprite = sprite;
                                StartCoroutine(ScrollToTop());
```

```
                        }
                    });

                }
            }
        }
    });
}
IEnumerator ScrollToTop()
{
    yield return new WaitForEndOfFrame();
    scrollRect.verticalNormalizedPosition = 1f;
}

void ShowInfoScene()
{
    string curBtn =
EventSystem.current.currentSelectedGameObject.name;
    Scenes.Load("InfoScene", "name",
keysArray[int.Parse(curBtn.Substring(curBtn.Length -
1))].ToString());
}

}
```

ExitManager.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ExitManager : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (SceneManager.GetActiveScene().buildIndex == 0)
            {
                Application.Quit();
            }
            else  if(SceneManager.GetActiveScene().buildIndex  ==
3) //from 3 info => 1 scan OR 2 profile
            {
                if (PlayerPrefs.GetInt("from")==1)
                {
                    SceneManager.LoadScene(1); //1 scan
                    PlayerPrefs.DeleteKey("from");
                }
                else
                {
                    SceneManager.LoadScene(2); //2 profile
                }
            }
            else  if(SceneManager.GetActiveScene().buildIndex  ==
4  ||  SceneManager.GetActiveScene().buildIndex  ==  5)  //from  5
signin, 4 register => 2 profile
            {
                SceneManager.LoadScene(2);
            }
```

```
            else if (SceneManager.GetActiveScene().buildIndex ==
6) //from 6 museum => 0 home OR 3 info
            {
                if (PlayerPrefs.GetInt("from") == 0)
                {
                    SceneManager.LoadScene(0); //0 home
                    PlayerPrefs.DeleteKey("from");
                }
                else
                {
                    //SceneManager.LoadScene(3);
                    Scenes.Load("InfoScene",              "name",
Scenes.getParam("name"));//3 info
                }
            }
            else //from 1 scan, 2 profile, 7 author => 0 home
            {
                SceneManager.LoadScene(0);
            }
        }
    }
}
```

FlashlightHandler.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class FlashlightHandler : MonoBehaviour
{
    public static bool isOn = false;
   public void TurnOnOff()
    {
        if (isOn)
        {
            CameraDevice.Instance.SetFlashTorchMode(false);
            isOn = false;
        }
        else
        {
            CameraDevice.Instance.SetFlashTorchMode(true);
            isOn = true;
        }

    }
}
```

FlexibleGridLayout.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class FlexibleGridLayout : LayoutGroup
{
```

```
    public int rows;
    public int columns;
    public Vector2 cellSize;
    public Vector2 spacing;

    public override void CalculateLayoutInputHorizontal()
    {
        base.CalculateLayoutInputHorizontal();
        float sqrRt = Mathf.Sqrt(transform.childCount);
        rows = Mathf.CeilToInt(sqrRt);
        columns = 2;// Mathf.CeilToInt(sqrRt);
        rows = Mathf.CeilToInt(transform.childCount / 2.0f);

        float parentWidth = rectTransform.rect.width;
        float parentHeight = rectTransform.rect.height;

        float cellWidth = parentWidth / (float)columns -
((spacing.x / (float)columns) * 2) - (padding.left /
(float)columns) - (padding.right / (float)columns);
        float cellHeight = parentHeight / (float)rows -
((spacing.y / (float)rows) * 2) - (padding.top / (float)rows) -
(padding.bottom / (float)rows);

        cellSize.x = cellWidth;
        cellSize.y = cellHeight;

        int columnCount = 0;
        int rowCount = 0;

        for(int i = 0; i < rectChildren.Count; i++)
        {
            rowCount = i / columns;
            columnCount = i % columns;

            var item = rectChildren[i];
            var xPos = (cellSize.x * columnCount) + (spacing.x *
columnCount) + padding.left;
            var yPos = (cellSize.y * rowCount) + (spacing.y *
rowCount) + padding.top;

            SetChildAlongAxis(item, 0, xPos, cellSize.x);
            SetChildAlongAxis(item, 1, yPos, cellSize.y);
        }

    }
}
```

HomeInfoLoader.cs

```
using Firebase.Auth;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Firebase.Database;
using Firebase.Storage;
using Firebase.Extensions;
using UnityEngine.EventSystems;
```

```
public class HomeInfoLoader : MonoBehaviour
{
    private DateTime dateTime;
    public Text helloText;
    private FirebaseAuth auth;
    private FirebaseUser user;
    private DatabaseReference dbRef;
    private StorageReference stRef;
    public GameObject authorParent;
    public GameObject group;
    public GameObject[] museums;
    private GameObject[] authors = new GameObject[3];
    public GameObject pic;

    void Start()
    {
        auth = FirebaseAuth.DefaultInstance;
        user = auth.CurrentUser;
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        stRef = FirebaseStorage.DefaultInstance.RootReference;
        CheckTime();
        LoadAuthors();
        LoadPics();
        for(int i = 0; i < museums.Length; i++)
        {

museums[i].GetComponent<Button>().onClick.AddListener(ShowMuseum
Scene);
        }

pic.GetComponent<Button>().onClick.AddListener(ShowPicInfo);

    }

    private void CheckTime()
    {
        dateTime = DateTime.Now;
        int hour = dateTime.Hour;

        if(hour >=5 && hour < 12)
        {
            helloText.text = "Доброе утро";
        }
        else if(hour >= 12 && hour < 16)
        {
            helloText.text = "Добрый день";
        }
        else if (hour >= 16 && hour < 23)
        {
            helloText.text = "Добрый вечер";
        }
        else
        {
            helloText.text = "Доброй ночи";
        }
        if (user != null)
        {
```

```
            helloText.text    =    helloText.text    +    ",    "
+user.DisplayName;
        }
    }

    void ShowMuseumScene()
    {
        PlayerPrefs.SetInt("from", 0);
        string                          curBtn                          =
EventSystem.current.currentSelectedGameObject.name;
        Scenes.Load("MuseumScene",                          "musid",
curBtn.Substring(curBtn.Length - 1));
        Debug.Log("Starting      with      musid      =      "+
curBtn.Substring(curBtn.Length - 1));
    }

    void ShowAuthorScene()
    {
        string                          curBtn                          =
EventSystem.current.currentSelectedGameObject.name;
        Scenes.Load("AuthorScene",                          "authid",
curBtn.Substring(curBtn.Length - 1));
        Debug.Log("Starting     with     authid     =     "     +
curBtn.Substring(curBtn.Length - 1));
    }

    void ShowPicInfo()
    {
        string                          curBtn                          =
EventSystem.current.currentSelectedGameObject.name;
        Scenes.Load("InfoScene", "name", "");
    }

    private void LoadPics()
    {
        Image                           img0                            =
pic.transform.Find("ImagePic").GetComponent<Image>();
        Text                           picName0                         =
pic.transform.Find("Text").GetComponent<Text>();


dbRef.Child("artworks").GetValueAsync().ContinueWithOnMainThread
(task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                string mostLikedName = "";
                int biggestlikedNum = 0;
                IEnumerable<DataSnapshot>       likedPics       =
snapshot.Children;
                foreach (DataSnapshot pic in likedPics)
                {
                    if (pic.HasChild("totalLiked"))
```

```
                    {
                        int             likedNum              =
int.Parse(pic.Child("totalLiked").GetValue(true).ToString());

                        if (likedNum > biggestlikedNum)
                        {
                            biggestlikedNum = likedNum;
                            mostLikedName = pic.Key.ToString();
                        }
                    }

                }
                SetPicImageText(img0, picName0, mostLikedName);

            }
        });
    }

    private void LoadAuthors()
    {
        Image                          img                          =
authorParent.transform.Find("RoundMask").GetComponent<Image>().g
ameObject.transform.Find("Image").GetComponent<Image>();
        Text                         authName                       =
authorParent.transform.Find("Text").GetComponent<Text>();
        SetImage(img, 6);
        SetName(authName, 6);
        authors[0] = authorParent;

        for (int i = 7; i <= 8; i++)
        {
            GameObject cloneAuthor = Instantiate(authorParent);
            cloneAuthor.transform.SetParent(group.transform,
false);
            cloneAuthor.name = "author" + i;
            Image                          mask                     =
cloneAuthor.transform.Find("RoundMask").GetComponent<Image>();
            Image                          img1                     =
mask.gameObject.transform.Find("Image").GetComponent<Image>();
            Text                         authName1                  =
cloneAuthor.transform.Find("Text").GetComponent<Text>();
            authors[i-6] = cloneAuthor;
            SetImage(img1, i);
            SetName(authName1, i);
        }
        for (int i = 0; i < authors.Length; i++)
        {

authors[i].GetComponent<Button>().onClick.AddListener(ShowAuthor
Scene);
        }
    }

    private void SetImage(Image img, int name)
    {

stRef.Child("authors").Child(name+"small.jpg").GetBytesAsync(2  *
1024 * 1024).ContinueWithOnMainThread(task =>
        {
```

```
            if (task.IsFaulted || task.IsCanceled)
            {
                Debug.Log(task.Exception.ToString());
            }
            else
            {
                byte[] fileContents = task.Result;
                Texture2D texture = new Texture2D(1, 1);
                texture.LoadImage(fileContents);
                Sprite sprite = Sprite.Create(texture, new
Rect(0.0f, 0.0f, texture.width, texture.height), new
Vector2(0.5f, 0.5f), 100.0f);
                img.sprite = sprite;
            }
        });
    }

    private void SetPicImageText(Image img, Text text, string
name) {

dbRef.Child("artworks").GetValueAsync().ContinueWithOnMainThread
(task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                text.text =
snapshot.Child(name).Child("name").GetValue(true).ToString();
            }
        });
        stRef.Child(name + ".jpg").GetBytesAsync(2 * 1024 *
1024).ContinueWithOnMainThread(task =>
        {
            if (task.IsFaulted || task.IsCanceled)
            {
                Debug.Log(task.Exception.ToString());
            }
            else
            {
                byte[] fileContents = task.Result;
                Texture2D texture = new Texture2D(1, 1);
                texture.LoadImage(fileContents);
                Sprite sprite = Sprite.Create(texture, new
Rect(0.0f, 0.0f, texture.width, texture.height), new
Vector2(0.5f, 0.5f), 100.0f);
                img.sprite = sprite;
            }
        });
    }



    private void SetName(Text text, int id)
    {
```

```
dbRef.Child("authors").GetValueAsync().ContinueWithOnMainThread(
task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                text.text                                          =
snapshot.Child(id+"").GetValue(true).ToString();
            }
        });
    }
}
```

InfoBarController.cs

```
using Firebase.Database;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using Firebase.Extensions;
using Firebase.Auth;
using Firebase.Storage;

public class InfoBarController : MonoBehaviour
{
    private DatabaseReference dbRef;
    private StorageReference stRef;
    private FirebaseAuth auth;
    private FirebaseUser user;
    public Image share, like;
    private string isLiked = "0";
    private int likedNum = 0;
    private string nameFromLiked = null;
    private bool isFocus = false;
    private bool isProcessing = false;

    private void Start()
    {
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        stRef = FirebaseStorage.DefaultInstance.RootReference;
        auth = FirebaseAuth.DefaultInstance;
        user = auth.CurrentUser;
        nameFromLiked = Scenes.getParam("name");
        CheckLiked();
    }

    public void CheckLiked()
    {
```

```
dbRef.Child("users").Child(user.UserId).GetValueAsync().Continue
WithOnMainThread(task => {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                likedNum                                        =
int.Parse(snapshot.Child("liked").ChildrenCount.ToString());

                if      (nameFromLiked      ==      null      ||
nameFromLiked.Equals(""))             nameFromLiked             =
SimpleCloudHandler.picId;

                if
(snapshot.Child("liked").HasChild(nameFromLiked))
                {
                    isLiked = "1";
                    like.color = new Color32(19, 19, 19, 255);

                }
                else
                {
                    isLiked = "0";
                    like.color  =  new  Color32(255,  255,  255,
255);

                }
            }
        });
    }


    public void GetMoreInfo()
    {
        PlayerPrefs.SetInt("from", 1);
        Scenes.Load("InfoScene",                                "name",
SimpleCloudHandler.picId);
    }

    public void Share()
    {
        dbRef.GetValueAsync().ContinueWithOnMainThread(task => {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                string picId = nameFromLiked;
                string nameText = "", authorText = "", yearText
= "неизвестный", materialsText = "неизвестные", sizeText = "неи-
звестный";
```

```
                //isLiked                                       =
snapshot.Child("artworks").Child(picId).Child("liked").GetValue(
true).ToString();

if(snapshot.Child("artworks").Child(picId).HasChild("name"))
nameText                                                        =
snapshot.Child("artworks").Child(picId).Child("name").GetValue(t
rue).ToString();
                string                    author                =
snapshot.Child("artworks").Child(picId).Child("author").GetValue
(true).ToString();
                if(snapshot.Child("authors").HasChild(author))
authorText                                                      =
snapshot.Child("authors").Child(author).GetValue(true).ToString(
);
                if
(snapshot.Child("artworks").Child(picId).HasChild("year"))
yearText                                                        =
snapshot.Child("artworks").Child(picId).Child("year").GetValue(t
rue).ToString();
                if
(snapshot.Child("artworks").Child(picId).HasChild("materials"))
materialsText                                                   =
snapshot.Child("artworks").Child(picId).Child("materials").GetVa
lue(true).ToString();
                if
(snapshot.Child("artworks").Child(picId).HasChild("size"))
sizeText                                                        =
snapshot.Child("artworks").Child(picId).Child("size").GetValue(t
rue).ToString();
                string msg = nameText + "\n" + authorText +
"\nМатериалы: " + materialsText + "\nГод создания: " + yearText
+ "\nОригинальный размер: " + sizeText;


stRef.Child(picId+".jpg").GetDownloadUrlAsync().ContinueWithOnMa
inThread(task1 => {
                    if (!task1.IsFaulted && !task1.IsCanceled)
                    {

                        string imgRef = task1.Result.ToString();

                        ShareText(msg, imgRef);

                    } else
                    {
                        Debug.Log("Download                    URL
ERROR"+task1.Result);
                    }
                });

            }
        });

    }

    public void ShareText(string msg, string imgRef)
    {
```

```
#if UNITY_ANDROID
        if (!isProcessing)
        {
            StartCoroutine(ShareTextInAnroid(msg, imgRef));
        }
#else
          Debug.Log("No sharing set up for this platform.");
#endif
    }

#if UNITY_ANDROID
    public  IEnumerator  ShareTextInAnroid(string  msg,  string
imgRef)
    {

        var shareMessage = msg;
        var shareSubject = "Посмотрите, что я нашёл!";
        isProcessing = true;
        string screenShotPath = imgRef;
        Debug.Log(screenShotPath);
        yield return new WaitForEndOfFrame();

        if (!Application.isEditor)
        {
            AndroidJavaClass         unity         =         new
AndroidJavaClass("com.unity3d.player.UnityPlayer");
            AndroidJavaObject          currentActivity          =
unity.GetStatic<AndroidJavaObject>("currentActivity");

            AndroidJavaClass       intentClass       =       new
AndroidJavaClass("android.content.Intent");
            AndroidJavaObject      intentObject       =       new
AndroidJavaObject("android.content.Intent");
            intentObject.Call<AndroidJavaObject>("setAction",
intentClass.GetStatic<string>("ACTION_SEND"));

            AndroidJavaObject       fileObject       =       new
AndroidJavaObject("java.io.File", screenShotPath);

            //create FileProvider class object
            AndroidJavaClass     fileProviderClass     =     new
AndroidJavaClass("android.support.v4.content.FileProvider");

            object[] providerParams = new object[3];
            providerParams[0] = currentActivity;
            providerParams[1]                                   =
"com.agrawalsuneet.unityclient.provider";
            providerParams[2] = fileObject;

            AndroidJavaObject             uriObject             =
fileProviderClass.CallStatic<AndroidJavaObject>("getUriForFile",
providerParams);

            intentObject.Call<AndroidJavaObject>("putExtra",
intentClass.GetStatic<string>("EXTRA_STREAM"), uriObject);
            intentObject.Call<AndroidJavaObject>("setType",
"image/png");
            intentObject.Call<AndroidJavaObject>("putExtra",
intentClass.GetStatic<string>("EXTRA_SUBJECT"), shareSubject);
```

```
            intentObject.Call<AndroidJavaObject>("putExtra",
intentClass.GetStatic<string>("EXTRA_TEXT"), shareMessage);

            intentObject.Call<AndroidJavaObject>("addFlags",
intentClass.GetStatic<int>("FLAG_GRANT_READ_URI_PERMISSION"));

            AndroidJavaObject              chooser              =
intentClass.CallStatic<AndroidJavaObject>("createChooser",
intentObject, "Поделитесь с друзьями понравившейся картиной");
            currentActivity.Call("startActivity", chooser);
        }

        yield return new WaitUntil(() => isFocus);
        isProcessing = false;
    }
#endif


public void Like()
    {
        if (user != null)
        {
        if (isLiked.Equals("1"))
            {

dbRef.Child("users").Child(user.UserId).Child("liked").Child(Sim
pleCloudHandler.picId).SetValueAsync(null);
                like.color = new Color32(255, 255, 255, 255);
                nameFromLiked = SimpleCloudHandler.picId;
                int totalLikedNum = 0;

dbRef.Child("artworks").Child(SimpleCloudHandler.picId).GetValue
Async().ContinueWithOnMainThread(task =>
                {
                    if (task.IsFaulted)
                    {
                        Debug.LogWarning("Ошибка  при  получении
данных из БД");
                    }
                    else if (task.IsCompleted)
                    {
                        DataSnapshot snapshot = task.Result;
                        if (snapshot.HasChild("totalLiked"))
                        {
                            totalLikedNum                      =
int.Parse(snapshot.Child("totalLiked").GetValue(true).ToString()
);
                            if      (totalLikedNum      >      0)
dbRef.Child("artworks").Child(SimpleCloudHandler.picId).Child("t
otalLiked").SetValueAsync(totalLikedNum - 1);
                        }
                    }
                });


                CheckLiked();
            }
            else
            {
```

```
dbRef.Child("users").Child(user.UserId).Child("liked").Child(Sim
pleCloudHandler.picId).SetValueAsync("liked");
                like.color = new Color32(19, 19, 19, 255);
                nameFromLiked = SimpleCloudHandler.picId;

dbRef.Child("artworks").Child(SimpleCloudHandler.picId).GetValue
Async().ContinueWithOnMainThread(task =>
                {
                    if (task.IsFaulted)
                    {
                        Debug.LogWarning("Ошибка  при  получении
данных из БД");
                    }
                    else if (task.IsCompleted)
                    {
                        DataSnapshot snapshot = task.Result;
                        if (snapshot.HasChild("totalLiked"))
                        {
                            int          totalLikedNum          =
int.Parse(snapshot.Child("totalLiked").GetValue(true).ToString()
)+1;

dbRef.Child("artworks").Child(SimpleCloudHandler.picId).Child("t
otalLiked").SetValueAsync(totalLikedNum);
                        }
                    }
                });

                CheckLiked();
            }
        }
    }
}
```

LoadingSceneLoader.cs

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LoadingSceneLoader : MonoBehaviour
{

    private AsyncOperation operation;
    private Canvas canvas;

    private void Awake()
    {
        canvas = GetComponentInChildren<Canvas>(true);
        DontDestroyOnLoad(gameObject);
    }

    public void LoadScene(string sceneName)
    {
        canvas.gameObject.SetActive(true);
        StartCoroutine(BeginLoad(sceneName));
    }
```

```
    private IEnumerator BeginLoad(string sceneName)
    {
        operation = SceneManager.LoadSceneAsync(sceneName);
        while (!operation.isDone)
        {
            yield return null;
        }
        operation = null;
        canvas.gameObject.SetActive(false);

    }
}
```

MoreInfoLoader.cs

```
using Firebase.Database;
using Firebase.Storage;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Firebase.Extensions;
using Firebase.Auth;

public class MoreInfoLoader : MonoBehaviour
{

    private DatabaseReference dbRef;
    private  Text  nameText,  authorText,  yearText,  sizeText,
materialsText, descrText;
    public Text museumName, likedText, scanText;
    public Image musImg;
    private Image img;
    public ScrollRect scrollRect;
    public VerticalLayoutGroup verticalLayoutGroup;
    public GameObject loadingImage;
    private FirebaseAuth auth;
    private FirebaseUser user;
    public Image like;
    string museumId = "";
    private string nameFromLiked = null;
    public GameObject museumObj;
    private StorageReference storageReference;


    void Start()
    {
        loadingImage.SetActive(true);
        auth = FirebaseAuth.DefaultInstance;
        user = auth.CurrentUser;
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        storageReference                                        =
FirebaseStorage.DefaultInstance.RootReference;

        nameText                                                =
GameObject.Find("NameText").GetComponent<Text>();
        authorText                                              =
GameObject.Find("AuthorText").GetComponent<Text>();
        yearText = GameObject.Find("year").GetComponent<Text>();
```

```
        materialsText                                                   =
GameObject.Find("materials").GetComponent<Text>();
        sizeText = GameObject.Find("size").GetComponent<Text>();
        descrText                                                       =
GameObject.Find("descrText").GetComponent<Text>();
        img = GameObject.Find("Image").GetComponent<Image>();
        nameFromLiked = Scenes.getParam("name");
        Debug.Log("nameFromLiked from start MoreInfoLoader = " +
nameFromLiked);
    }

    private void Awake()
    {
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        nameFromLiked = Scenes.getParam("name");

dbRef.Child("artworks").Child(nameFromLiked).GetValueAsync().Con
tinueWithOnMainThread(task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                LoadInfo(nameFromLiked);
                DataSnapshot snapshot = task.Result;
                if (snapshot.HasChild("museum"))
                {
                    museumId                                            =
snapshot.Child("museum").GetValue(true).ToString();

museumObj.GetComponent<Button>().onClick.AddListener(ShowInfoSce
ne);
                    Debug.Log("museumId   FROM   AWAKE   =   "   +
museumId);
                    LoadMuseum();
                }

            }
        });
    }

    void ShowInfoScene()
    {
        Scenes.Load("MuseumScene", "musid", museumId);
    }

    private void LoadInfo(string linkname)
    {
        dbRef.GetValueAsync().ContinueWithOnMainThread(task => {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
```

```
                DataSnapshot snapshot = task.Result;
                if (user != null) {
                    int                likedNum                =
(int)snapshot.Child("users").Child(user.UserId).Child("liked").C
hildrenCount;

                    if     (nameFromLiked     ==     null     ||
nameFromLiked.Equals(""))             nameFromLiked             =
SimpleCloudHandler.picId;
                    if
(snapshot.Child("users").Child(user.UserId).Child("liked").HasCh
ild(nameFromLiked))
                    {
                        like.color  =  new  Color32(19,  19,  19,
255);
                        Debug.Log("ЕСТЬ В ИЗБРАННОМ");
                    }
                    else
                    {
                        like.color = new Color32(255, 255, 255,
255);
                    }
                }
                nameText.text                                   =
snapshot.Child("artworks").Child(linkname).Child("name").GetValu
e(true).ToString();
                string                  author                  =
snapshot.Child("artworks").Child(linkname).Child("author").GetVa
lue(true).ToString();
                authorText.text                                 =
snapshot.Child("authors").Child(author).GetValue(true).ToString(
);

                if
(snapshot.Child("artworks").Child(linkname).HasChild("recos"))
scanText.text                                                   =
snapshot.Child("artworks").Child(linkname).Child("recos").GetVal
ue(true).ToString();
                if
(snapshot.Child("artworks").Child(linkname).HasChild("totalLiked
"))                          likedText.text                     =
snapshot.Child("artworks").Child(linkname).Child("totalLiked").G
etValue(true).ToString();
                if
(snapshot.Child("artworks").Child(linkname).HasChild("year"))
yearText.text                                                   =
snapshot.Child("artworks").Child(linkname).Child("year").GetValu
e(true).ToString();

if(snapshot.Child("artworks").Child(linkname).HasChild("material
s"))                         materialsText.text                 =
snapshot.Child("artworks").Child(linkname).Child("materials").Ge
tValue(true).ToString();

if(snapshot.Child("artworks").Child(linkname).HasChild("size"))
sizeText.text                                                   =
snapshot.Child("artworks").Child(linkname).Child("size").GetValu
e(true).ToString();
```

```
if(snapshot.Child("artworks").Child(linkname).HasChild("info"))
descrText.text                                                    =
snapshot.Child("artworks").Child(linkname).Child("info").GetValu
e(true).ToString();
                if   (!museumId.Equals(""))   museumName.text   =
snapshot.Child("museums").Child(museumId).GetValue(true).ToStrin
g();
                else museumName.text = "Нет информации о место-
нахождении картины";


            }
        });

        LoadMuseum();
        storageReference.Child(linkname                           +
".jpg").GetBytesAsync(2 * 1024 * 1024).
            ContinueWithOnMainThread(task =>
            {
                if (task.IsFaulted || task.IsCanceled)
                {
                    Debug.Log(task.Exception.ToString());
                    StartCoroutine(UpdateRect());
                }
                else
                {
                    byte[] fileContents = task.Result;
                    Texture2D texture = new Texture2D(1, 1);
                    texture.LoadImage(fileContents);
                    Sprite  sprite  =  Sprite.Create(texture,  new
Rect(0.0f,    0.0f,    texture.width,    texture.height),    new
Vector2(0.5f, 0.5f), 100.0f);
                    img.sprite = sprite;
                    StartCoroutine(UpdateRect());
                }
            });

    }

    private void LoadMuseum()
    {
        if (!museumId.Equals(""))
        {
            storageReference.Child("museums").Child(museumId   +
".jpg").GetBytesAsync(2 * 1024 * 1024).
            ContinueWithOnMainThread(task1 =>
            {
                if (task1.IsFaulted || task1.IsCanceled)
                {
                    Debug.Log(task1.Exception.ToString());
                }
                else
                {
                    byte[] fileContents = task1.Result;
                    Texture2D texture = new Texture2D(1, 1);
                    texture.LoadImage(fileContents);
```

```
                    Sprite sprite = Sprite.Create(texture, new
Rect(0.0f,    0.0f,    texture.width,    texture.height),    new
Vector2(0.5f, 0.5f), 100.0f);
                    musImg.sprite = sprite;

                }
            });
        }
    }

    IEnumerator ScrollToTop()
    {
        yield return new WaitForEndOfFrame();
        scrollRect.verticalNormalizedPosition = 1f;
        LeanTween.scale(loadingImage,  new  Vector3(0,  0,  0),
0.5f);
        loadingImage.SetActive(false);
    }
    IEnumerator UpdateRect()
    {
        if (verticalLayoutGroup != null)
        {
            verticalLayoutGroup.enabled = false;
            yield return new WaitForSeconds(0.1F);
            verticalLayoutGroup.enabled = true;
        }
        StartCoroutine(ScrollToTop());
    }
}
```

MuseumInfoLoader.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Firebase.Auth;
using Firebase.Database;
using Firebase.Storage;
using Firebase.Extensions;
using UnityEngine.EventSystems;

public class MuseumInfoLoader : MonoBehaviour
{
    public Image musImg;
    public Text musName;
    private StorageReference stRef;
    private DatabaseReference dbRef;
    private FirebaseUser user;
    private ArrayList keysArray;
    private ArrayList picsArray;
    private string musId;
    public GameObject expoPic;
    public GameObject group;
    public ScrollRect scrollRect;
    private int picsNum = 0;

    void Start()
    {
```

```
        stRef = FirebaseStorage.DefaultInstance.RootReference;
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        user = FirebaseAuth.DefaultInstance.CurrentUser;

        musId = Scenes.getParam("musid");
        stRef.Child("museums").Child(musId                          +
".jpg").GetBytesAsync(2            *            1024            *
1024).ContinueWithOnMainThread(task =>
        {
            if (task.IsFaulted || task.IsCanceled)
            {
                Debug.Log(task.Exception.ToString());
            }
            else
            {
                byte[] fileContents = task.Result;
                Texture2D texture = new Texture2D(1, 1);
                texture.LoadImage(fileContents);
                Sprite   sprite   =   Sprite.Create(texture,   new
Rect(0.0f,    0.0f,    texture.width,    texture.height),    new
Vector2(0.5f, 0.5f), 100.0f);
                musImg.sprite = sprite;
            }
        });

dbRef.Child("museums").GetValueAsync().ContinueWithOnMainThread(
task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                musName.text                                        =
snapshot.Child(musId).GetValue(true).ToString();
            }
        });
        LoadExpo();
    }

    private void LoadExpo()
    {
        dbRef.GetValueAsync().ContinueWithOnMainThread(task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;

                picsArray = new ArrayList();
                keysArray = new ArrayList();
```

```
                IEnumerable<DataSnapshot>       allPics       =
snapshot.Child("artworks").Children;
                foreach (DataSnapshot pic in allPics)
                {
                    if
(snapshot.Child("artworks").Child(pic.Key.ToString()).HasChild("
museum"))
                    {
                        string               mus               =
snapshot.Child("artworks").Child(pic.Key.ToString()).Child("muse
um").GetValue(true).ToString();
                        if (mus.Equals(musId))
                        {
                            string          nameOfPic          =
snapshot.Child("artworks").Child(pic.Key.ToString()).Child("name
").GetValue(true).ToString();
                            picsArray.Add(nameOfPic);
                            keysArray.Add(pic.Key.ToString());
                            picsNum++;
                        }
                    }
                }
                Text                   nametext1                =
expoPic.transform.Find("Text").GetComponent<Text>();
                if (picsNum == 0)
                {
                    nametext1.text = "К сожалению, тут пока пус-
то";
                }
                else
                {

expoPic.GetComponent<Button>().onClick.AddListener(ShowInfoScene
);
                    nametext1.text = picsArray[0].ToString();
                    stRef.Child(keysArray[0]                    +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                        ContinueWithOnMainThread(task1 =>
                        {
                            if        (task1.IsFaulted        ||
task1.IsCanceled)
                            {

Debug.Log(task1.Exception.ToString());
                            }
                            else
                            {
                                byte[]       fileContents       =
task1.Result;
                                Texture2D    texture    =    new
Texture2D(1, 1);
                                texture.LoadImage(fileContents);
                                Sprite          sprite          =
Sprite.Create(texture,   new   Rect(0.0f,   0.0f,   texture.width,
texture.height), new Vector2(0.5f, 0.5f), 100.0f);
                                Image           image           =
expoPic.transform.Find("Image").GetComponent<Image>();
                                image.sprite = sprite;
```

```
                        }
                    });

                RectTransform             rt1             =
(RectTransform)group.transform;
                rt1.sizeDelta = new Vector2(rt1.sizeDelta.x,
Mathf.CeilToInt(picsNum / 2.0f) * 440);

                for (int i = 0; i < picsNum - 1; i++)
                {
                    GameObject          cloneLiked          =
Instantiate(expoPic);

cloneLiked.transform.SetParent(group.transform, false);
                    cloneLiked.name = "expo" + (i + 1);


cloneLiked.GetComponent<Button>().onClick.AddListener(ShowInfoSc
ene);

                    Text              nametext              =
cloneLiked.transform.Find("Text").GetComponent<Text>();
                    nametext.text     =     picsArray[i     +
1].ToString();

                    stRef.Child(keysArray[i     +     1]     +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                    ContinueWithOnMainThread(task1 =>
                    {
                        if        (task1.IsFaulted        ||
task1.IsCanceled)
                        {

Debug.Log(task1.Exception.ToString());
                        }
                        else
                        {
                            byte[]       fileContents       =
task1.Result;
                            Texture2D    texture    =    new
Texture2D(1, 1);
                            texture.LoadImage(fileContents);
                            Sprite          sprite          =
Sprite.Create(texture,   new   Rect(0.0f,   0.0f,   texture.width,
texture.height), new Vector2(0.5f, 0.5f), 100.0f);
                            Image           image1          =
cloneLiked.transform.Find("Image").GetComponent<Image>();
                            image1.sprite = sprite;
                            StartCoroutine(ScrollToTop());
                        }
                    });

                }
            }


        }
    });
}
```

```
    IEnumerator ScrollToTop()
    {
        yield return new WaitForEndOfFrame();
        scrollRect.verticalNormalizedPosition = 1f;
    }

    void ShowInfoScene()
    {
        string                          curBtn                          =
EventSystem.current.currentSelectedGameObject.name;
        Scenes.Load("InfoScene",                                  "name",
keysArray[int.Parse(curBtn.Substring(curBtn.Length                -
1))].ToString());
    }
}
```

NavButtonHandler.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class NavButtonHandler : MonoBehaviour
{
    public void goHomeScene()
    {
        SceneManager.LoadScene("HomeScene");
    }

    public void goProfileScene()
    {
        SceneManager.LoadScene("ProfileScene");
    }

    public void goScanScene()
    {
        SceneManager.LoadScene("ScanScene");
    }

    public void goRegistrationScene()
    {
        SceneManager.LoadScene("RegistrationScene");
    }

    public void goSignInScene()
    {
        SceneManager.LoadScene("SignInScene");
    }
}
```

RegistrationHandler.cs

```
using UnityEngine;
using Firebase.Auth;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Firebase.Extensions;
using Firebase.Database;
```

```
public class RegistrationHandler : MonoBehaviour
{
    private FirebaseAuth auth;
    public Text nameText, emailText, passText;
    private string email, password;
    private DatabaseReference dbRef;

    void Start()
    {
        auth = FirebaseAuth.DefaultInstance;
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
    }

    public void Register()
    {
        email = emailText.text.ToString();
        password = passText.text.ToString();
        auth.CreateUserWithEmailAndPasswordAsync(email,
password).ContinueWithOnMainThread(task => {
            if (task.IsCanceled)
            {

Debug.LogError("CreateUserWithEmailAndPasswordAsync           was
canceled.");
                return;
            }
            if (task.IsFaulted)
            {

Debug.LogError("CreateUserWithEmailAndPasswordAsync   encountered
an error: " + task.Exception);
                return;
            }
            FirebaseUser newUser = task.Result;
            if (newUser != null)
            {
                UserProfile profile = new UserProfile
                {
                    DisplayName = nameText.text.ToString()
                };

newUser.UpdateUserProfileAsync(profile).ContinueWithOnMainThread
(task1 => {
                    if (task1.IsCanceled)
                    {
                        Debug.LogError("UpdateUserProfileAsync
was canceled.");
                        return;
                    }
                    if (task1.IsFaulted)
                    {
                        Debug.LogError("UpdateUserProfileAsync
encountered an error: " + task1.Exception);
                        return;
                    }
                    Debug.Log("User        profile        updated
successfully.");
                    Debug.LogFormat("Firebase    user    created
successfully: {0} ({1})", newUser.DisplayName, newUser.UserId);
```

```
dbRef.Child("users").Child(newUser.UserId).Child("recos").SetVal
ueAsync(0);

                    SceneManager.LoadScene("ProfileScene");
                });
            }

        });
    }

    public void SignIn()
    {
        email = emailText.text.ToString();
        password = passText.text.ToString();
        auth.SignInWithEmailAndPasswordAsync(email,
password).ContinueWithOnMainThread(task => {
            if (task.IsCanceled)
            {
                Debug.LogError("SignInWithEmailAndPasswordAsync
was canceled.");
                return;
            }
            if (task.IsFaulted)
            {
                Debug.LogError("SignInWithEmailAndPasswordAsync
encountered an error: " + task.Exception);
                ShowAndroidToastMessage("Неверный адрес или па-
роль! Пожалуйста, проверьте данные.");
                return;
            }
            FirebaseUser newUser = task.Result;
            Debug.LogFormat("User signed in successfully: {0}
({1})", newUser.DisplayName, newUser.UserId);
            SceneManager.LoadScene("ProfileScene");
        });
    }

    private void ShowAndroidToastMessage(string message)
    {
        AndroidJavaClass unityPlayer = new
AndroidJavaClass("com.unity3d.player.UnityPlayer");
        AndroidJavaObject unityActivity =
unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");

        if (unityActivity != null)
        {
            AndroidJavaClass toastClass = new
AndroidJavaClass("android.widget.Toast");
            unityActivity.Call("runOnUiThread", new
AndroidJavaRunnable(() =>
            {
                AndroidJavaObject toastObject =
toastClass.CallStatic<AndroidJavaObject>("makeText",
unityActivity, message, 0);
                toastObject.Call("show");
            }));
        }
```

```
    }
}
```

Scenes.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;

public static class Scenes
{

    private static Dictionary<string, string> parameters;

    public static void Load(string sceneName, Dictionary<string,
string> parameters = null)
    {
        Scenes.parameters = parameters;
        SceneManager.LoadScene(sceneName);
    }

    public static void Load(string sceneName, string paramKey,
string paramValue)
    {
        Scenes.parameters = new Dictionary<string, string>();
        Scenes.parameters.Add(paramKey, paramValue);
        SceneManager.LoadScene(sceneName);
    }

    public       static       Dictionary<string,       string>
getSceneParameters()
    {
        return parameters;
    }

    public static string getParam(string paramKey)
    {
        if (parameters == null) return "";
        return parameters[paramKey];
    }

    public   static   void   setParam(string   paramKey,   string
paramValue)
    {
        if (parameters == null)
            Scenes.parameters    =    new    Dictionary<string,
string>();
        Scenes.parameters.Add(paramKey, paramValue);
    }

}
```

SimpleCloudHandler.cs

```
using Firebase.Database;
using UnityEngine;
using UnityEngine.UI;
using Vuforia;
```

```
using Firebase.Storage;
using System.Collections;
using Firebase.Extensions;
using Firebase.Auth;

public class SimpleCloudHandler : MonoBehaviour,
IObjectRecoEventHandler
{
    private CloudRecoBehaviour mCloudRecoBehaviour;
    public Text nameText, authorText;
    private DatabaseReference dbRef;
    private StorageReference stRef;
    public GameObject infoBar;
    public static string picId;
    private FirebaseAuth auth;
    private FirebaseUser user;
    public UnityEngine.UI.Image like;
    public GameObject imageTargetObj;

    void Start()
    {
        mCloudRecoBehaviour =
GetComponent<CloudRecoBehaviour>();

        if (mCloudRecoBehaviour)
        {
            mCloudRecoBehaviour.RegisterEventHandler(this);
        }
        mCloudRecoBehaviour.CloudRecoEnabled = true;

        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        stRef =
FirebaseStorage.DefaultInstance.RootReference.Child("authors");
        auth = FirebaseAuth.DefaultInstance;
        user = auth.CurrentUser;
      }

    public void OnInitialized(TargetFinder targetFinder)
    {
        Debug.Log("Инициализировано Cloud Reco");
    }
    public void OnInitError(TargetFinder.InitState initError)
    {
        Debug.Log("Ошибка инициализации Cloud Reco " +
initError.ToString());
    }
    public void OnUpdateError(TargetFinder.UpdateState
updateError)
    {
        Debug.Log("Ошибка обновления Cloud Reco " +
updateError.ToString());
    }

    public void OnStateChanged(bool scanning)
    {
        //Debug.Log("<color=blue>OnStateChanged(): </color>" +
scanning);
        if (scanning)
        {
```

```
            ObjectTracker                tracker                =
TrackerManager.Instance.GetTracker<ObjectTracker>();

tracker.GetTargetFinder<ImageTargetFinder>().ClearTrackables(fal
se);
        } else
        {
            imageTargetObj.SetActive(false);
        }

    }

    public                                                  void
OnNewSearchResult(TargetFinder.TargetSearchResult
targetSearchResult)
    {
        TargetFinder.CloudRecoSearchResult  cloudRecoSearchResult
= (TargetFinder.CloudRecoSearchResult)targetSearchResult;
        picId = cloudRecoSearchResult.TargetName;
        Debug.Log(picId);

        imageTargetObj.SetActive(true);

        dbRef.GetValueAsync().ContinueWithOnMainThread(task => {

            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                infoBar.SetActive(true);
                DataSnapshot snapshot = task.Result;
                nameText.text                                   =
snapshot.Child("artworks").Child(picId).Child("name").GetValue(t
rue).ToString();
                string                    author                =
snapshot.Child("artworks").Child(picId).Child("author").GetValue
(true).ToString();
                authorText.text                                 =
snapshot.Child("authors").Child(author).GetValue(true).ToString(
);
                int recosNum = 0;
                if
(snapshot.Child("artworks").Child(picId).HasChild("recos"))
                {
                    recosNum                                    =
int.Parse(snapshot.Child("artworks").Child(picId).Child("recos")
.GetValue(true).ToString());
                }

dbRef.Child("artworks").Child(picId).Child("recos").SetValueAsyn
c(recosNum+1);

                if (user != null)
                {
                    CheckLiked();
```

```
                    int                    recos                    =
int.Parse(snapshot.Child("users").Child(user.UserId).Child("reco
s").GetValue(true).ToString())+1;

dbRef.Child("users").Child(user.UserId).Child("recos").SetValueA
sync(recos);


                }
            }
        });
    }


    public void CheckLiked()
    {

dbRef.Child("users").Child(user.UserId).GetValueAsync().Continue
WithOnMainThread(task => {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                string                  likedNum                =
snapshot.Child("liked").ChildrenCount.ToString();

                if (snapshot.Child("liked").HasChild(picId))
                {
                    like.color = new Color32(19, 19, 19, 255);
                }
                else
                {
                    like.color  =  new  Color32(255,  255,  255,
255);
                }
            }
        });
    }
}
```

UserInfoLoader.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Firebase.Auth;
using UnityEngine.UI;
using Firebase.Database;
using Firebase.Extensions;
using System;
using Firebase.Storage;
using UnityEngine.EventSystems;

public class UserInfoLoader : MonoBehaviour
{
```

```
    private FirebaseAuth auth;
    public GameObject buttons, emptyStats;
    public Text nameT, emailT;
    public Image userPhoto;
    public Button signOutBtn;
    public Text scanNum, likedNum;
    private DatabaseReference dbRef;
    private FirebaseUser user;
    public GameObject likedPic;
    public GameObject group;
    private StorageReference storageReference;
    public VerticalLayoutGroup verticalLayoutGroup;
    public ScrollRect scrollRect;
    ArrayList keysArray;

    void Start()
    {
        auth = FirebaseAuth.DefaultInstance;
        user = auth.CurrentUser;
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        storageReference                                          =
FirebaseStorage.DefaultInstance.RootReference;

        if (user != null)
        {
            buttons.SetActive(false);
            emptyStats.SetActive(false);
            nameT.text = user.DisplayName;
            emailT.text = user.Email;
            SetUserPhoto(user.UserId);
            signOutBtn.gameObject.SetActive(true);
            CheckStats();

        }
        else
        {
            buttons.SetActive(true);
            emptyStats.SetActive(true);
            nameT.text = "Вы не вошли в систему";
            emailT.text = "";
            signOutBtn.gameObject.SetActive(false);
        }
    }

    private void SetUserPhoto(string id)
    {
        storageReference.Child("users").Child(id                  +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                    ContinueWithOnMainThread(task =>
                    {
                        if (task.IsFaulted || task.IsCanceled)
                        {

Debug.Log(task.Exception.ToString());
                        }
                        else
                        {
                            byte[] fileContents = task.Result;
```

```
                            Texture2D     texture     =     new
Texture2D(1,1);
                            texture.LoadImage(fileContents);
                            texture = Resize(texture, 256, 341);
                            Sprite           sprite           =
Sprite.Create(texture,  new  Rect(0.0f,  0.0f,  256,  341),  new
Vector2(0.5f, 0.5f), 100.0f);
                            userPhoto.sprite = sprite;


                        }
                    });
    }

    Texture2D  Resize(Texture2D  texture2D,  int  targetX,  int
targetY)
    {
        RenderTexture  rt  =  new  RenderTexture(targetX,  targetY,
24);
        RenderTexture.active = rt;
        Graphics.Blit(texture2D, rt);
        Texture2D result = new Texture2D(targetX, targetY);
        result.ReadPixels(new  Rect(0,  0,  targetX,  targetY),  0,
0);
        result.Apply();
        return result;
    }
    private void CheckStats()
    {
        dbRef.GetValueAsync().ContinueWithOnMainThread(task =>
        {
            if (task.IsFaulted)
            {
                Debug.LogWarning("Ошибка при получении данных из
БД");
            }
            else if (task.IsCompleted)
            {
                DataSnapshot snapshot = task.Result;
                string                   likedT                =
snapshot.Child("users").Child(user.UserId).Child("liked").Childr
enCount.ToString();
                likedNum.text = likedT;
                scanNum.text                                     =
snapshot.Child("users").Child(user.UserId).Child("recos").GetVal
ue(true).ToString();
                int                   picsNum                    =
(int)Math.Ceiling(double.Parse(likedT));
                ArrayList picsArray = new ArrayList();
                keysArray = new ArrayList();
                IEnumerable<DataSnapshot>       likedPics       =
snapshot.Child("users").Child(user.UserId).Child("liked").Childr
en;
                foreach (DataSnapshot pic in likedPics)
                {
                    string               nameOfPic              =
snapshot.Child("artworks").Child(pic.Key.ToString()).Child("name
").GetValue(true).ToString();
                    picsArray.Add(nameOfPic);
```

```
                keysArray.Add(pic.Key.ToString());
            }

likedPic.GetComponent<Button>().onClick.AddListener(ShowInfoScen
e);
            Text                nametext1                =
likedPic.transform.Find("Text").GetComponent<Text>();
            nametext1.text = picsArray[0].ToString();
            storageReference.Child(keysArray[0]             +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                ContinueWithOnMainThread(task1 =>
                {
                    if (task1.IsFaulted || task1.IsCanceled)
                    {

Debug.Log(task1.Exception.ToString());
                    }
                    else
                    {
                        byte[] fileContents = task1.Result;
                        Texture2D texture = new Texture2D(1,
1);
                        texture.LoadImage(fileContents);
                        Sprite            sprite            =
Sprite.Create(texture,   new   Rect(0.0f,   0.0f,   texture.width,
texture.height), new Vector2(0.5f, 0.5f), 100.0f);
                        Image             image             =
likedPic.transform.Find("Image").GetComponent<Image>();
                        image.sprite = sprite;

                    }
                });

            RectTransform               rt1                =
(RectTransform)group.transform;
            rt1.sizeDelta   =   new   Vector2(rt1.sizeDelta.x,
Mathf.CeilToInt(picsNum / 2.0f) * 440);

            for (int i = 0; i < picsNum-1; i++)
            {
                GameObject            cloneLiked            =
Instantiate(likedPic);

cloneLiked.transform.SetParent(group.transform, false);
                cloneLiked.name = "liked" + (i + 1);


cloneLiked.GetComponent<Button>().onClick.AddListener(ShowInfoSc
ene);

                Text                nametext                =
cloneLiked.transform.Find("Text").GetComponent<Text>();
                nametext.text = picsArray[i+1].ToString();

                storageReference.Child(keysArray[i+1]       +
".jpg").GetBytesAsync(2 * 1024 * 1024).
                ContinueWithOnMainThread(task1 =>
                {
                    if (task1.IsFaulted || task1.IsCanceled)
```

```
                        {
Debug.Log(task1.Exception.ToString());
                        }
                        else
                        {
                            byte[] fileContents = task1.Result;
                            Texture2D texture = new Texture2D(1,
1);
                            texture.LoadImage(fileContents);
                            Sprite           sprite           =
Sprite.Create(texture,   new   Rect(0.0f,   0.0f,   texture.width,
texture.height), new Vector2(0.5f, 0.5f), 100.0f);
                            Image            image1           =
cloneLiked.transform.Find("Image").GetComponent<Image>();
                            image1.sprite = sprite;
                            StartCoroutine(ScrollToTop());

                        }
                    });

                }
            }
        });
    }

    void ShowInfoScene()
    {
        string                       curBtn                      =
EventSystem.current.currentSelectedGameObject.name;
        Scenes.Load("InfoScene",                              "name",
keysArray[int.Parse(curBtn.Substring(curBtn.Length-
1))].ToString());
    }

    IEnumerator UpdateRect()
    {
        if (verticalLayoutGroup != null)
        {
            verticalLayoutGroup.enabled = false;
            yield return new WaitForSeconds(0.1F);
            verticalLayoutGroup.enabled = true;
        }
       StartCoroutine(ScrollToTop());
    }

    IEnumerator ScrollToTop()
    {
        yield return new WaitForEndOfFrame();
        scrollRect.verticalNormalizedPosition = 1f;
    }

    public void SignOut()
    {
        auth.SignOut();
    }
}
```

AGH

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

2nd place

for

Anastasiya Molchanova

the author of presentation:

Development of a mobile application for museum tours using augmented reality technologies

delivered in section
Computer Science

58 of the Conference of Student Scientific Circles of the Metallurgical Division of AGH

Representative of the Rector for student research clubs

PhD **Joanna Augustyn-Nadzieja**

AGH UST Vice-Rector for Student Affairs

Professor **Rafał Dańko**

Kraków, 6 maja 2021