Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра мультимедійних систем

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Текстова частина до курсової роботи

за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник курсової роботи
с.в. Борозенний С.О.
(прізвище та ініціали)
_____
(підпис)
“____”_____ 2021 р.
Виконав студент БП ІПЗ-4
Сабадишин М.О.
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2021 р.

Київ-2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. Кафедри мультимедійних систем,

доцент, к.ф-м.н.

_____ О.П. Жежерун

(підпис)

“____” _____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

Студента Сабадишина Максима Олександровича факультету інформатики 4 курсу

ТЕМА розробка мобільного застосунку з використанням технології доповненої реальності

Зміст ТЧ до курсової роботи:

- Календарний план
- Вступ
- Аналіз предметної області та дослідження її актуальності
- Теоретичні відомості щодо використаних технологій
- Опис реалізації застосунку
- Висновки по роботі та аналіз шляхів розвитку
- Список джерел

Дата видачі “____” _____ 2021 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання роботи:

Тема: розробка мобільного застосунку з використанням технології доповненої реальності

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової	01.10.2020	
2.	Ознайомлення з предметною областю	04.01.2021	
3.	Пошук корисних ресурсів	08.01.2021	
4.	Проведення дослідження	10.01.2021	
5.	Закінчення дослідження та опрацювання результатів	20.01.2021	
6.	Проектування та підбір архітектури до практичної частини застосунку	23.01.2021	
7.	Створення початково налаштованого проекту	30.01.2021	
8.	Вивчення матеріалів по доповненій реальності	01.02.2021	
9.	Створення MVP з простою ідентифікацією в просторі	15.02.2021	
10.	Створення повноцінного функціоналу по доповненій реальності	17.02.2021	
11.	Реалізація функціоналу навчальних матеріалів	26.02.2021	
12.	Створення розділу з тестами	05.03.2021	
13.	Рефакторинг коду	09.03.2021	
14.	Написання текстової частини роботи	10.03.2021	
15.	Створення презентації та доповіді до неї	05.04.2021	

Студент Сабадишин М.О.

Керівник Борозенний С.О.

“____” _____

ЗМІСТ

Анотація

Метою цієї курсової роботи є створення застосунку під мобільні пристрої, що використовував би технології доповненої реальності, та ніс освітню ціль для визначеної аудиторії.

Проведено аналіз обраних сфер, та серед них обрано найбільш релевантну до початкової мети.

Проведено огляд актуальності технології доповненої реальності, її розвиток та перспективи, потенціал використання сьогодні.

Розглянуто актуальність використання мобільних застосунків в освітніх цілях, окремо проаналізовано використання додатків з використанням технології в освітніх проектах.

Розглянуто нові підходи в розробці додатків під iOS, проаналізовано їх сильні та слабкі сторони.

На основі попереднього аналізу остаточно визначено функціонал та ціль додатку, додано нові функціональні вимоги.

Ключові слова: мобільний додаток, створення додатку, доповнена реальність, iOS, освітній застосунок.

Вступ

Актуальність теми та її практичне значення

Попри тенденцію до збільшення середньої вартості смартфону[1], їх ринок збільшується з кожним роком, а прикладна користь тільки розширюється. З кожною новою ітерацією розвитку смартфонів, вони отримують нові функції та потужніше залізо. Разом з тим, цей потенціал часто не використовується, а найпопулярнішими додатками є месенджери, соц. мережі, сервіси перегляду відео[2], що мають в собі постійний незмінюваний функціонал.

Створювати додатки використовуючи нові технології може бути ризиковано, адже немає гарантії, що це буде саме те, чим захоче користуватися аудиторія. Погіршує стабільну в майбутньому зацікавленість користувачів в технології й те, що на моменті її виходу, а з цим і на моменті найбільшої органічної популярності в загальної аудиторії, виходить велика кількість несерйозних та безкорисних додатків.

Вони разом привертають на себе увагу через новий досвід для кінцевого користувача, проте разом з цим заповнюють маркет, не даючи шансу просунутись дійсно цікавим проектам. Це часто грає проти поширення технології в корисних цілях.

Проте комбінуючи потенціал з новими технологіями, з’являється можливість створити справді цікавий продукт.

Власна технологія доповненої реальності з’явилась на пристроях Apple у 2017 році. [3]

Така розповсюдженість смартфонів, звісно, змінила і їх роль в процесах змін. Вони дали легкий доступ до аудиторії з мільйонів користувачів по всьому світу. Такий доступ – чудова можливість для впливу на соціум. Проте вплив може бути як корисний, так і шкідливий.

В контексті даної роботи розглянуто використання смартфонів як інструменту поширення та впливу в корисних цілях, а саме сконцентровано увагу на використанні смартфону як освітнього інструменту.

З розвитком технологій змінився підхід і в освітніх процесах. Вони еволюціонували та підлаштувались під нові актуальні умови. Популярність отримали не тільки веб сервіси, що надавали послуги онлайн курсів, а і додатки для смартфонів. Велика різноманітність та нові формати вивчення чогось нового також привернули увагу і самих користувачів.

В роботі розглянуто використання технології доповненої реальності для досягнення інтересу в користувачів щодо використання смартфону як місця для отримання нової інформації.

Структура роботи

Дана робота складається з трьох основних розділів.

Перший розділ містить аналіз предметної області, дослідження можливості використання смартфонів у освітніх цілях. Також розглянуто технологію доповненої реальності та проаналізовано її прикладну користь.

У другому розділі проведено опис використаних під час розробки застосунку та дизайну технологій.

Третій розділ наповнений аргументацією вибору тієї чи іншої технології, проведено порівняння з конкурентами. Також розділ містить опис процесу розробки додатку, і загальні принципи в роботі з ним.

Розділ 1. Аналіз предметної області та дослідження її актуальності

1.1 Дослідження можливості використання смартфонів як інструменту для самоосвіти та саморозвитку

Освітні процеси доволі динамічно підлаштовуються та розвиваються під потреби сучасної людини. Крім того, що еволюціонують вже сформовані можливості для саморозвитку, також з'являються все нові варіанти того, як можна дізнаватись щось нове використовуючи при цьому смартфони чи планшети.

Одним з прикладів того, як раніше існуючий формат знаходить нове життя у цифровому світі, є книги. Індустрія книг вже давно поступово почала процес переходу в електронний формат, а деякі взагалі видають ексклюзивно в ньому.

Цьому є декілька причин. [4] По-перше, електронні книги значно більш портативні, адже по суті не займають жодного додаткового фізичного простору. Також вони значно доступніші з точки зору швидкості отримання. Їх не потрібно замовляти і чекати, вони стають доступними для прочитання відразу після покупки.

Враховуючи згадані переваги, очікуваним стало закріплення ринку електронних книг та їх становлення як одного з провідних прикладів використання пристроїв з точки зору самоосвіти.

Іншим прикладом переходу в нові технології є навчальні курси. Електронні платформи надають великий потенціал для розвитку таких курсів, саме тому сьогодні існує тенденція щодо збільшення популярності такого формату самоосвіти.

Звісно, пандемія додала свого впливу, тому популярність курсів в 2020-2021 році ще більш помітна. На сьогоднішній день існує багато окремо виділених під онлайн освіту ресурсів, таких як edX, Coursera, чи український Prometheus. Разом з цим багато офлайн шкіл також готують свої онлайн платформи для утримання існуючих користувачів та залучення нової потенційної аудиторії.

Онлайн курси мають багато своїх переваг[5]. Вони мають набагато більш гнучкий розклад, доходячи до того, що клієнтам надаються записи уроків у разі відсутності. Також такі курси часто дешевші за їх офлайн аналоги, адже багато потенційних витрат шкіл, як от оренда приміщення для класів, зникають. Це дає змогу запропонувати більш гнучкі тарифи, що будуть привабливішими для користувачів і все ще прибутковими для самої компанії.

З додаткових переваг такого підходу до освіти є і те, що перед користувачем лежить вибір з усіх можливих курсів без прив'язки до фактичної локації викладача. Це дає змогу слухати матеріали з багатьох країн світу без жодних зусиль.

Разом з цим, з розвитком смартфонів почали з'являтись і нові формати того, як люди почали отримувати знання. Деякі з цих форматів не довели свою ефективність та зникли, проте певні з них стали незамінними для мільйонів користувачів по всьому світу.

З популяризацією мобільних пристроїв та додатків для них, набув розквіту формат вивчення іноземних мов, базуючись на картковому збереженні слів, та регулярному повторенні задля того, аби вони відклались у довготривалій пам'яті[6].

Багато інших додатків також використовували доступність в смартфонах починаючи розробляти освітні додатки на багато наукових чи загальних тем. Набули популярності додатки з інформацією про людське тіло чи географію Землі.

У підтвердження щодо популярності смартфонів як освітніх інструментів можна навести статистику магазину мобільних додатків для смартфонів та планшетів компанії Apple – App Store, та магазину Android додатків Google Play. На сьогоднішній день категорія Education – третя за популярністю в App Store з часткою 8.68% та друга в Google Play з часткою 8.94%. [7]

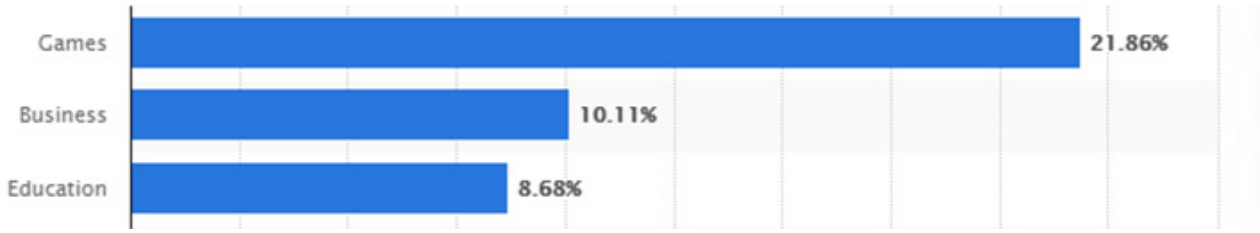

Рисунок 1.1 Статистика категорій в App Store

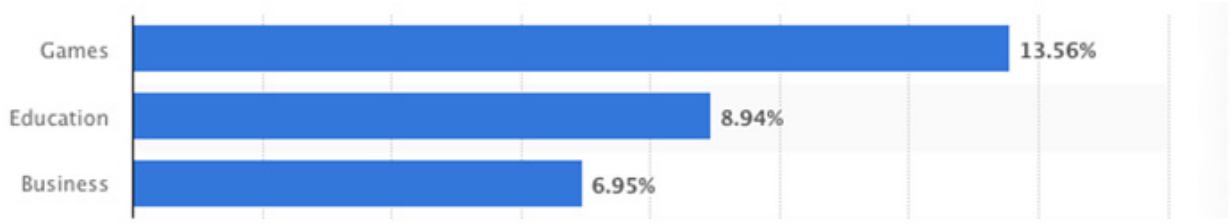

Рисунок 1.2 Статистика категорій в Google Play

Ринок освітніх додатків зростає і фінансово. Згідно даним Research and Markets, їх ринок буде коштувати 398 мільярдів доларів в 2026, що по суті значить 300% приріст у порівнянні з 2015 роком. [8]

Якщо не прогнозувати, а дивитись на поточні успіхи ринку, то можна звернути увагу на те, що в порівнянні з 2018 роком, в 2019 інвестиції в освітні стартапи збільшились на 14.2%. [9]

Стрімкий розвиток мобільних технологій та зацікавленість як інвесторів, так і аудиторії, робить створення мобільного додатку з освітніми цілями доволі актуальним в сьогоднішніх умовах.

1.2 Загальний огляд технології доповненої реальності в мобільних пристроях.

Доповнена реальність – це технологія, що дозволяє програмно поєднати два початково незалежних один від одного світи: реальний світ навколо людини, з його об’єктами та подіями, і віртуально запрограмований світ, з попередньо визначеними та створеними моделями.

Загалом, ідея доповненої реальності полягає в тому, що вхідний відеосигнал опрацьовується, і на нього накладається попередньо створене віртуальне середовище.

Це реалізується за допомогою камери смартфону, та, найчастіше, певних маркерів, які програма відшуковує на вхідному сигналі.

В основі технології доповненої реальності лежать математичні алгоритми, які в одне ціле поєднують камеру смартфону та маркери на екрані. Оскільки кінцева задача технології – накласти віртуальну модель на реальний відеопотік, то для її досягнення основним кроком є визначення положення моделі на екрані. Найчастіше для досягнення цього використовуються мітки. А тому програма отримуючи знімок з камери починає пошук певного патерну, що відповідав би заданому маркерові.

Після знаходження, задача полягає в тому, щоб визначити який саме маркер було знайдено, а після – накласти віртуальну модель на необхідне місце і з'єднати з маркером. Після цього з'єднання модель закріплюється в просторі та залишається на своїй позиції навіть за рухів користувача.

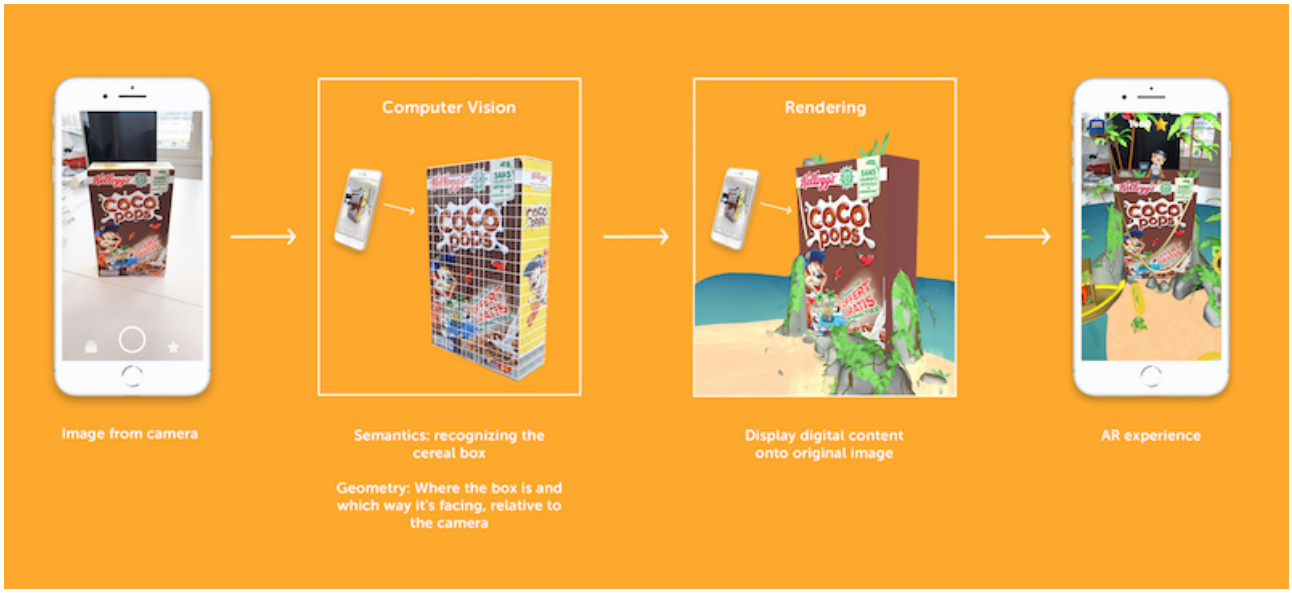

Рисунок 1.3 Приклад роботи AR додатку[10]

Загалом існує три основні напрями розвитку технології доповненої реальності: маркерна, безмаркерна, та просторова.

Маркерна технологія використовує певні мітки, які розпізнаються камерою, і потім використовуються для розміщення віртуальних об'єктів на відеопотоці.

Завдяки наявності маркерів стає легше чітко розмістити віртуальну модель в потрібному місці та «прив’язати» її до нього в просторі. Маркером може бути, наприклад, 2D зображення, візуальні риси якого під час аналізу буде легко відрізнити. [11]

Безмаркерна технологія, як можна зрозуміти з назви, не потребує використання міток в роботі. Її суть в спеціальних алгоритмах розпізнавання патернів. На вхідний відеопотік накладається віртуальна сітка, після чого вона аналізується на предмет певних опорних точок, знайшовши які й відбувається прив’язка моделі до місця на відео. Перевагою є те, що у якості таких точок можуть використовуватись об’єкти з реального світу.

Просторовий підхід діє інакше, за його використання використовується технологія, що базується на знаходженні пристрою в просторі. В роботі з нею використовуються різні датчики та сенсори, що містяться в смартфоні, як от гіроскоп, компас, GPS модуль. Відповідно до цього, місце віртуального об’єкта визначається реальними координатами в просторі й він стає видимим лише за збігу його координат з координатами користувача.

Порівнюючи ці три підходи можна сконкретизувати, що маркерний є найбільш надійним з усіх. При цьому він також має певні проблеми, які треба розуміти перед початком роботи. Всі вони зводяться в одну основну – коли пристрій перестає бачити мітку, у нього зникає розуміння локації об’єкта на камері, а тому він просто зникає до моменту відновлення інформації по міткам.

Є два основних чинники такої проблеми: розташування мітки на екрані та недостача світла.

Заглиблюючись в причини виникнення проблеми з ідентифікацією маркера, визначаємо, що камері необхідна чітка картинка мітки. Необхідно, щоб рамки маркеру були повністю видимі, не були під певним кутом нахилу чи прикриті іншими предметами. Також, в момент руху камери необхідно враховувати те, що камера має постійно тримати маркер чітко в кадрі, а тому за швидкого переміщення пристрою відносно мітки також можливе зникнення моделі.

Наявність світла є важливою для якісного досвіду з доповненою реальністю, адже його нестача напряму впливає на якість картинки, яка аналізуватиметься на

предмет знаходження в ній маркеру. Крім нестачі, так само погано впливає і його надмірність. Засвітлені зони також можуть приховувати мітку і заважати її ідентифікації.

1.3 Аналіз розвитку та актуальності технології доповненої реальності

Перед тим як використовувати ту чи іншу технологію в проекті, що має прямий вплив на досвід користувача, необхідно оцінити її перспективи, проаналізувати існуючі, якщо такі вже є, додатки на ринку, і загалом пропрацювати її прийняття користувачами.

Технологія доповненої реальності хороший кандидат для такого аналізу. Вона вже достатньо довго на ринку, щоб мати набір даних для роботи й прогнозування її подальших результатів.

Перш за все, було переглянуто світовий дохід ринку мобільних додатків з використанням технології доповненої реальності. З графіку чітко видно сильний щорічний зріст ринку й очікуваний ріст і надалі. Маючи ринок в 3.99 мільярди доларів у 2019 році, на 2020 рік оцінка ринку складала вже 6.16 мільярдів. За прогнозами, у 2024 році з ростом такою швидкістю вартість ринку складатиме 21.02 мільярд доларів США. [12]

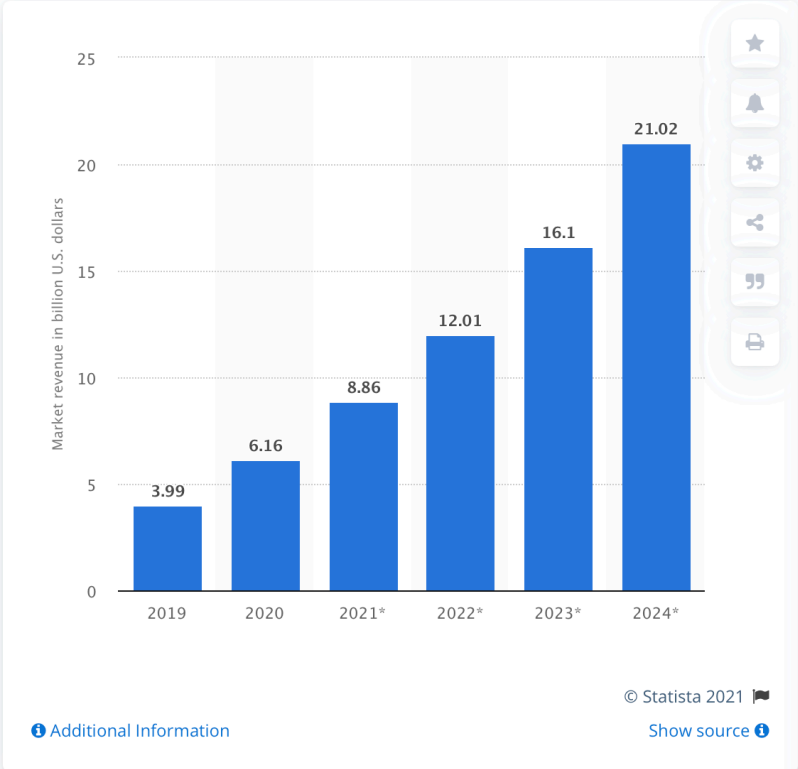

Рисунок 1.4 Вартість ринку мобільних AR додатків

Зважаючи на такі успішні результати зараз, а також на те, що технологія активно підтримується виробниками, можна прогнозувати такий успішний ріст і надалі.

Для створення чіткої картини інтересу користувачів, важливо також звернути увагу на те, скільки загалом активних користувачів AR додатків по світу.

Також важливо оцінити наскільки користувачі готові витрачати власних коштів на додатки з досвідом доповненої реальності.

Говорячи про кількість активних користувачів можна помітити, що зі зростом самого ринку, зростає і їх кількість. Варто зауважити, що в контексті даного дослідження до уваги бралась кількість саме активних користувачів, а не загальна. Адже людей, що завантажили той чи інший застосунок просто так набагато більше, в той час як репрезентативну вибірку складають саме ті, хто реально користується продуктом.

Згідно з дослідженням проведеним в період з 2019 до 2020 року, кількість користувачів складала 440 мільйонів у 2019 році, в той час, як в наступному році вона зросла вже до 600 мільйонів активних. За прогнозом, до 2024 року ця цифра сягне 1.73 мільярда користувачів. [13]

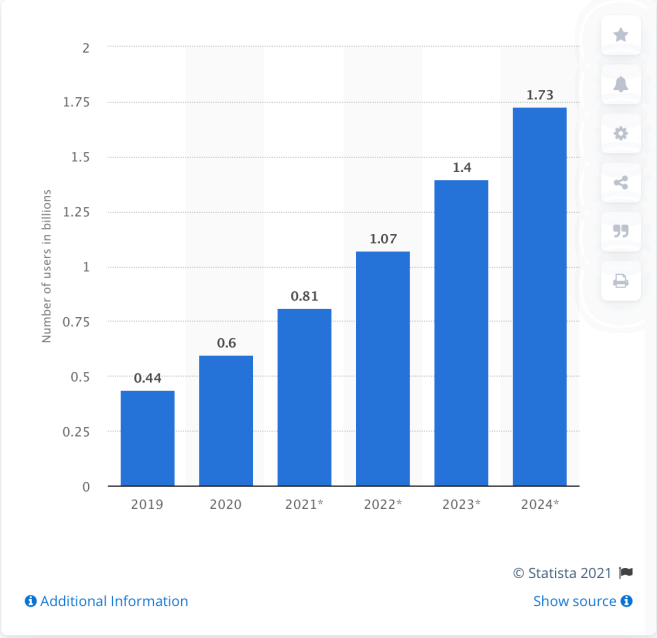

Рисунок 1.5 Графік росту активних користувачів AR мобільних AR додатків

Також стрімкий ріст видно і в тому, як багато користувачі готові витрачати на додатки з використанням технології доповненої реальності. Звісно, необхідно також зауважити, що ріст витрат корелює з загальним ростом аудиторії. Проте видно, що він хоч і не такий стрімкий в порівнянні з попередньо розглянутими показниками, все ж є. І маючи 1.17 мільярдів доларів витрат на 2019 рік створено прогноз на збільшення таких витрат до 4.18 мільярдів доларів у 2024 році. [14]

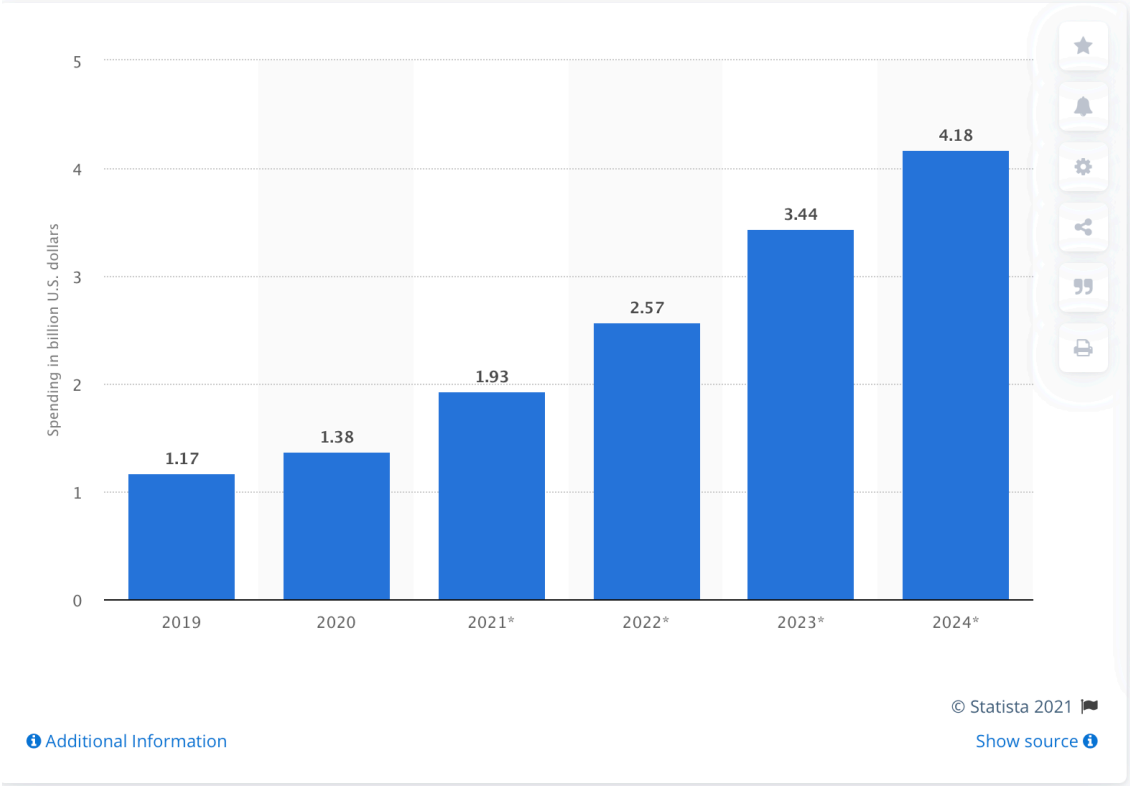

Рисунок 1.6 Графік витрат на AR додатки користувачами по світу

Підсумовуючи попередні ключові точки аналізу популярності ринку додатків з використанням доповненої реальності можна з упевненістю сказати, що ринок живий, а інтерес до такого типу застосунків тільки зростає і продовжуватиме свій ріст у майбутньому.

1.4 Аналіз можливостей використання технології доповненої реальності в освітніх додатках

Враховуючи попередньо визначений інтерес до додатків з функціоналом доповненої реальності не дивно, що ніша освітніх застосунків вже почала освоюватись такими стартапами, а також компаніями, що до цього вже були присутні на ринку.

Діапазон програм варіюється від тих, які мають функціонал, пов’язаний з доповненою реальністю, як частину від основного, до тих, вся робота яких повністю зав’язана на ній.

Проте загалом потенційні можливості, які надає доповнена реальність у сфері освіти, не мають обмежень.

Доповнена реальність покриває декілька базових потреб людини, що бажає дізнатись щось нове: вона допомагає отримати та запам’ятати інформацію. Крім цього, отримання нових знань через такий специфічний формат робить процес цікавішим та краще відкладає в пам’яті отриману інформацію. [15]

Разом з цим, доповнена реальність не ставить жодних вікових рамок щодо аудиторії додатку. Це можуть бути люди похилого віку, що вирішили дізнатись про еволюцію розвитку технологій, студенти, яким необхідно підготуватись до екзаменів по біології й вивчити анатомію людського тіла, або ж діти, які тільки вивчають правопис.

Така універсальність підходу і робить його потенційно дуже успішним. Також доповнена реальність дає й інші переваги в даній категорії застосунків.

На відміну від віртуальної реальності, яка вимагає додаткових аксесуарів, як самий мінімум, шолома, доповнена реальність не потребує додаткових капіталовкладень, крім смартфона чи планшета. Враховуючи, що сьогодні 73% підлітків володіють смартфоном, більшість отримує доступ до такого контенту відразу. [16]

73% of Teens Have Access to a Smartphone; 15% Have Only a Basic Phone

73% of Teens Have Access to a Smartphone; 15% Have Only a Basic Phone

% of all teens who have or have access to the following types of cell phones

	Smartphone	Basic phone only	No cell phone
All teens	73%	15%	12%
Sex			
a Boys	71	16	13
b Girls	74	14	12
Race / ethnicity			
c White, non-Hispanic	71	17[d]	12
d Black, non-Hispanic	85[ce]	7	8
e Hispanic	71	15	14
Age			
f 13-14	68	14	18[f]
g 15-17	76[f]	16	8
Sex by age			
h Boys 13-14	64	16	19[ik]
i Boys 15-17	75[h]	16	8
j Girls 13-14	72	11	17[ik]
k Girls 15-17	76[h]	16	8
Household income			
l <$30K	61	22[no]	17[o]
m $30K-$49,999	67	16	18[o]
n $50K-$74,999	76[l]	12	12
o $75K+	78[lm]	13	9
Parent educational attainment			
p Less than high school	60	21	19[s]
q High school	72	15	13
r Some college	76[p]	12	12
s College+	75[p]	16	9
Urbanity			
t Urban	73	16	11
u Suburban	74	14	12
v Rural	68	16	15

Source: Pew Research Center's Teens Relationships Survey, Sept. 25–Oct.9, 2014 and Feb. 10–Mar. 16, 2015 (N=1,060 teens ages 13 to 17).

Note: Percentages marked with a superscript letter (e.g., [a]) indicate a statistically significant difference between that row and the row designated by that superscript letter, among categories of each demographic characteristic (e.g. age).

PEW RESEARCH CENTER

Рисунок 1.7 Статистика доступу у підлітків до смартфону

Інтерактивний процес навчання має сильний вплив на тих, хто отримує новий матеріал, та дозволяє бути більш сконцентрованим на навчанні й не відволікатись. Візуалізація та 3D формат того про що розповідається також тільки сприяє кращому сприйняттю інформації.

Останні ітераційні оновлення фреймворків по роботі з доповненою реальністю на базі смартфонів під iOS та Android додали можливості колаборційного досвіду з доповненою реальністю, що дає можливість, за доступної програмної реалізації, об’єднуватись по групам та отримувати досвід доповненої реальності разом.

З прикладів успішних кейсів додатків з доповненою реальністю, що мали її в основі свого функціоналу, можна навести NarratorAR. [17] Це додаток, що розрахований на дітей, та за свою ціль ставить покращити процес вивчення

письма. NarratorAR також чудовий приклад того, як можна поєднати використання додатку з використанням фізичних матеріалів, адже надає друковані завдання, які мають бути використані дітьми.

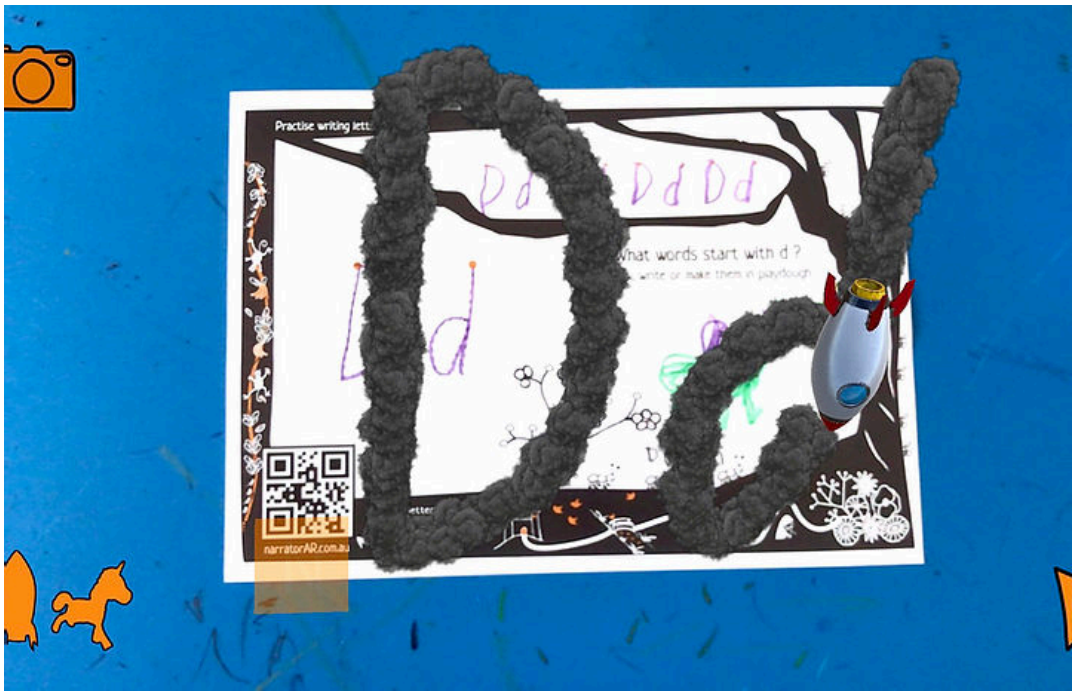

Рисунок 1.8 Приклад використання NarratorAI [18]

Іншим прикладом використання доповненої реальності в освітніх цілях є додатки музеїв. Доповнена реальність переносить експозиції в кімнату до користувачів, а, наприклад, додаток Jardin Botanique Grand Nancy [19] надає можливість пройти AR відео тур для того, щоб користувачі з різних куточків світу могли відвідати цей ботанічний сад.

Резюмуючи, доповнена реальність може бути застосовна в освітній сфері та приносить з собою багато переваг, при цьому будучи доступною для користувачів.

1.5 Висновки до розділу 1

В першому розділові було розглянуто можливості використання смартфонів як інструментів для самоосвіти, проаналізовано процеси розвитку навчальних форматів, їх шлях в онлайн.

Також в розділі було розглянуто загальну інформацію щодо технології доповненої реальності, базові принципи її роботи, актуальні техніки та підходи, а також можливі проблеми за використання.

В розділі проаналізовано розвиток та популярність технології, прогнози на її положення в майбутньому. Базуючись на проведеному аналізі, зроблено висновки щодо актуальності застосування доповненої реальності.

В кінці розділу було розглянуто можливості застосування доповненої реальності в освітніх додатках, розглянуто існуючі застосунки.

Розділ 2. Теоретичні відомості щодо використаних технологій

2.1 Сучасні підходи в розробці iOS додатків

Цьогоріч виповниться 14 років з моменту як iPhone був представлений публіці. Всі ці роки з еволюцією самих гаджетів еволюціонувала і розробка програмного забезпечення під них.

З виходом iPhone 2G з'явився App Store та базовий SDK для розробки додатків під смартфони компанії Apple, що складався з чотирьох базових компонентів, а багато очевидних сьогодні структур приходилось писати вручну.

Кожна ітерація вносила нові можливості й сьогодні засоби розробки під пристрої Apple зручні та повноцінні.

Враховуючи кількість користувачів iPhone, розробка під цю платформу стала популярною. Це може підтвердити статистика, яка стверджує, що з моменту запуску App Store у 2008 році й по 2017 рік користувачі здійснили близько 180 мільярдів завантажень. [20] Зараз ця цифра, звісно, суттєво більша. Така кількість завантажень створила потребу у якості додатків, і тому вже недостатньо просто створити застосунок з хорошою ідеєю.

Крім ідеї, продукт має швидко та надійно працювати, мати зручний користувацький інтерфейс та досвід, а також отримувати постійні оновлення. Для того, щоб він відповідав цим вимогам розробникам необхідно розуміти й застосовувати сучасні підходи при розробці iOS додатків.

Користувацький інтерфейс є важливою складовою успішного проекту, адже це те, що першим бачить користувач. Apple хотіла б мати певну поведінкову схожість у додатках на своїй платформі, для того, щоб користувач, незалежно від того яким застосунком він користується, постійно відчував себе в межах iOS.

Для виконання такої цілі були створені Human Interface Guidelines. Це формальна конвенція Apple, що стосується користувацького інтерфейсу та досвіду в рамках додатку під iOS. Цей набір гайдлайнів корисний як для дизайнерів, адже надає конкретні приклади та поради щодо застосування елементів користувацького інтерфейсу, так і для розробників, бо може слугувати

довідником по повному циклу розробки. Human Interface Guidelines також надають кращі приклади втілення різних функціональних елементів і детальну інформацію по компонентах користувацького інтерфейсу в екосистемі Apple.

HIG приносять користь і в розробці додатків з технологією доповненої реальності. В гадлайнах міститься інформація по взаємодії з нею, а також наводяться типові проблеми при роботі. До прикладу, довідники містять інформацію по тому як саме користувач взаємодіє з 3D моделями на екрані і в чому різниця між взаємодією з реальним контентом.

Більш того, Human Interface Guidelines постійно доповнюються та оновлюються згідно з останніми змінами в екосистемі, а також містять бібліотеку з ресурсами для використання та ознайомлення.

Слідування Human Interface Guidelines не тільки важливе і береться до уваги перевіркою при запуску додатку в App Store, але також дає змогу кінцевому користувачеві отримати найбільше комфорту від користування ним без необхідності додатково вивчати нові для себе патерни взаємодії.

Швидкість та надійність роботи, безумовно, також є одними з найважливіших складових для користувачів. Сьогодні мобільні додатки – це комплексний продукт з багатьма модулями, які постійно працюють та обмінюються інформацією: це зв’язок з мережею, обробка дій від користувача, надання нової інформації користувачеві, та інші.

Для того, щоб це працювало швидко та надійно, недостатньо просто вміти писати код, необхідно також звертати увагу та використовувати усталені підходи при розробці застосунків. Такі підходи та патерни існують і у світі мобільної розробки.

Перед написанням додатку необхідно обрати його архітектуру. На сьогодні є багато можливих підходів з архітектурної точки зору. Найпопулярнішим все ще залишається MVC, який є рекомендованим і самою Apple.

За реалізації MVC об’єкти в додатку умовно діляться на три основні рівні: Model, View, та Controller. Вони виконують виділені їм дії та комунікують між собою використовуючи усталені підходи.

Model – це частина додатку, що займається керуванням даними, які зберігаються. На рівні Model відбувається читання з баз даних, збереження їх туди, а також повідомлення Controller про якісь зміни з інформацією.

View відповідає за рівень відображення, в нього входить весь користувацький інтерфейс. Крім цього, View не має відповідати за будь-яку логіку пов'язану з ним.

Своєю чергою Controller відповідає за обробку подій, що надходять з View. Він з'єднує рівень логіки з рівнем даних, та використовує їх для того, щоб змінювати стан додатку відповідно до виконаних користувачем дій, або ж оновлень, що приходять зі сторони сервера.

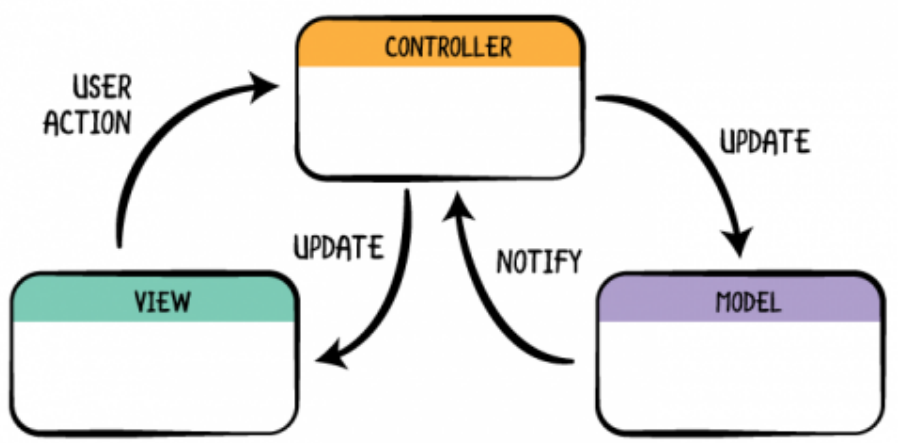

Рисунок 2.1 Схематичне зображення патерну MVC [21]

За даного підходу можна легко перевикористовувати об'єкти з Model та View рівнів, а також надати чітке розподілення обов'язків між ними.

Даний підхід був та залишається найпопулярнішим і найбільш зрозумілим в iOS розробці, проте має свої недоліки. Найбільш частий з них, це так званий "Massive ViewController". Він полягає в тому, що з часом розвитку додатку його логіка розширюється, і об'єми Controller рівня без належного контролю та додаткового розділення стають занадто великими. Це в свою чергу погіршує виявлення помилок та сповільнює роботу.

З розвитком мобільної розробки та збільшенням потреби у патернах, що задовольняли б вимогам високонавантажених систем, почали з'являтись і нові підходи. Сьогодні є багато інших способів як організувати архітектуру мобільного додатку. Найпопулярніші з них це MVVM, MVP, VIPER.

Суть MVVM полягає в додатковому рівні, який розділив би Controller у MVC. MVVM складається з Model, View, ViewModel, де View відповідає за відображення, а також за обробку дій користувача. ViewModel в свою чергу відповідає за маніпуляцію з даними на рівні View, а саме за надання необхідних даних для View, зміну даних через команди з View, а також зв'язок з Model, яка власне, як і у MVC, працює з інформацією.

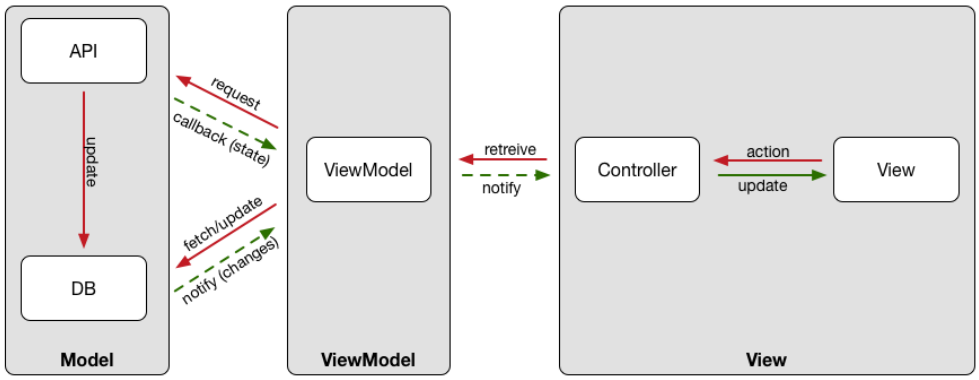

Рисунок 2.2 Схематичне зображення патерну MVVM [22]

MVP також є альтернативою для розв'язання проблеми занадто об'ємних контролерів. Даний патерн пропонує використання Model, View, та Presenter. Різниця з MVC полягає в тому, що View рівень в даному підході також відповідає за опрацювання подій пов'язаних з користувачем. Presenter в свою чергу містить в собі логіку й оновлення користувацького інтерфейсу через делегата.

Ще однією перевагою використання даного підходу є те, що в ньому Presenter не залежить від UIKit - фреймворку для користувацького інтерфейсу, а тому легко тестується.

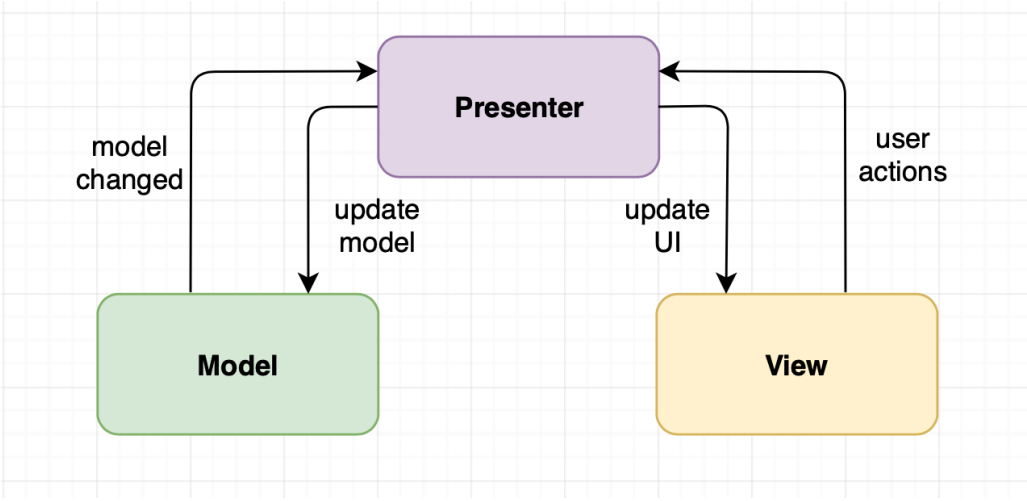

Рисунок 2.3 Схематичне зображення патерну MVP [23]

VIPER є більш комплексним підходом при розробці додатку і розрахований саме на високонавантажені системи. Проте його можна використовувати й для менш вимогливих додатків, при цьому відчуваючи переваги. VIPER – це акронім, що складається з View, Interactor, Presenter, Entity, та Routing.

View показує те, що повідомляється в Presenter і передає йому назад дії клієнта.

Interactor містить в собі бізнес-логіку, конкретну під випадок використання.

Presenter має в собі логіку View з підготовки інтерфейсу для відображення і для того, щоб реагувати на дії користувача.

Entity містить базові об’єкти моделі, що використовуються Interactor.

Routing в свою чергу містить логіку переходів між екранами.

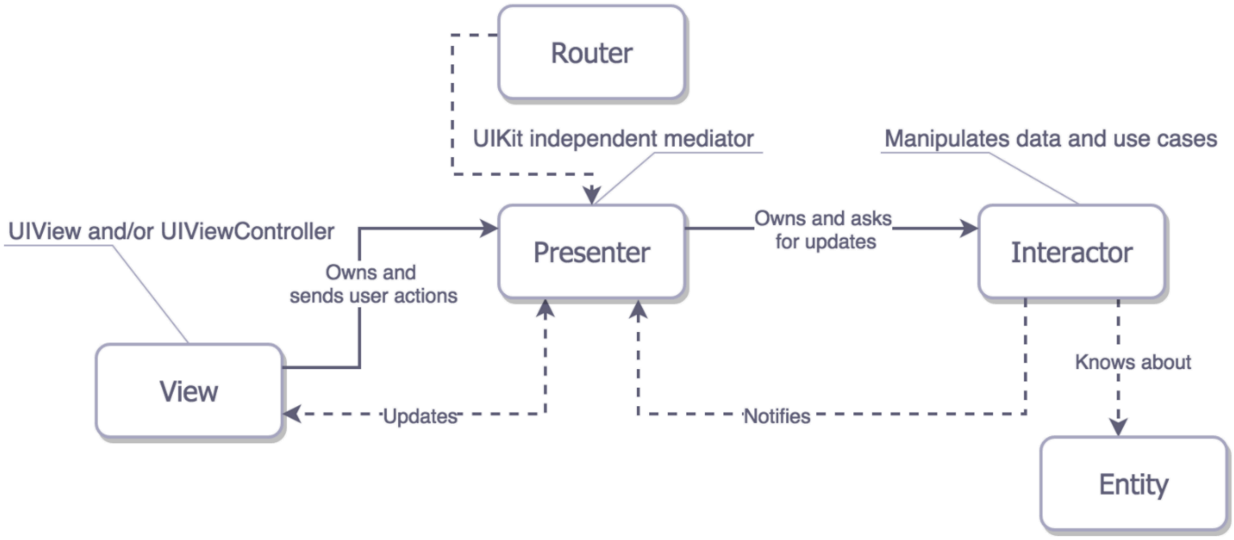

Рисунок 2.4 Схематичне зображення патерну VIPER [24]

Кожен новий підхід намагався виправити помилки попередніх та додати своїх переваг, проте на сьогоднішній день немає ідеального рішення, і розробникам все одно приходиться обирати залежно від того, який саме підхід буде зручніше використати команді.

2.2 Swift як актуальна мова програмування для написання iOS додатків

Починаючи розробку додатку під будь-яку платформу, вибір мов програмування завжди є важливим. У випадку з мобільною розробкою початковий вибір стоїть між тим, обрати засоби нативної розробки чи кросплатформової.

Нативна розробка – це тип створення продукту під кожну платформу окремо, використовуючи створені для цього інструменти. Кросплатформова розробка в свою чергу дає змогу створити додаток під iOS та Android одночасно, використовуючи спільну кодову базу.

Переваги нативної розробки у швидкодії роботи, розмірі додатку, підтримці низькорівневих функцій та використання нових технологій смартфонів.

У випадку написання нативного додатка під iOS вибір складається з двох мов програмування: Swift та Objective-C. Objective-C була першою на ринку і поява Swift була спричинена бажанням створити більш сучасну мову програмування за синтаксисом та можливостями. Незважаючи на активний розвиток Swift та те, що Apple зробила остаточну ставку саме на нього, Objective-C все ще використовується та підтримується багатьма командами по всьому світові. В останньому виданні рейтингу TIOBE за поширеністю ці дві мови відділяють соті відсотка.

Mar 2021	Mar 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	15.33%	-1.00%
2	1	↓	Java	10.45%	-7.33%
3	3		Python	10.31%	+0.20%
4	4		C++	6.52%	-0.27%
5	5		C#	4.97%	-0.35%
6	6		Visual Basic	4.85%	-0.40%
7	7		JavaScript	2.11%	+0.06%
8	8		PHP	2.07%	+0.05%
9	12	↑	Assembly language	1.97%	+0.72%
10	9	↓	SQL	1.87%	+0.03%
11	10	↓	Go	1.31%	+0.03%
12	18	⇈	Classic Visual Basic	1.26%	+0.49%
13	11	↓	R	1.25%	-0.01%
14	20	⇈	Delphi/Object Pascal	1.20%	+0.48%
15	36	⇈	Groovy	1.19%	+0.94%
16	14	↓	Ruby	1.18%	+0.13%
17	17		Perl	1.15%	+0.24%
18	15	↓	MATLAB	1.04%	+0.05%
19	13	⇊	Swift	0.95%	-0.28%
20	19	↓	Objective-C	0.91%	+0.17%

Рисунок 2.5 Рейтинг TIOBE популярності мов програмування [25]

Проте якщо розглядати питання вибору мови програмування для нового додатку, то Swift з його постійними оновленнями та набутою стабільністю виглядає як впевнений вибір.

Він був представлений Apple на щорічній конференції для розробників WWDC у 2014 році й з того часу отримав 5 стабільних версій. На сьогоднішній день останньою є Swift 5.3.

Swift – це мова програмування загального призначення з автоматичним керуванням пам’яттю та статичною типізацією. Це надає змогу розробнику сконцентруватись на інших аспектах роботи над продуктом, а також отримувати інформацію про помилки з типами ще на етапі компіляції. Наявність статичної типізації дозволяє уникнути багатьох проблем в процесі роботи програми, забезпечуючи правильність формату даних, що зберігаються в тих чи інших змінних.

Щодо автоматичного керування пам’яттю, то у Swift за це відповідає так звана система ARC – Automatic Reference Counter. Оскільки мається на увазі автоматичне керування, то в загальному випадку для розробника виділення та звільнення пам’яті «просто працює», без необхідності думати про те, які саме процеси відбуваються всередині.

В основі ARC лежить рахівник посилань на об’єкт. Якщо на об’єкт A посилається об’єкт B, то рахівник посилань об’єкту A збільшується на один і таким чином забезпечує збереження його в пам’яті. Об’єкт може бути вивільненим з пам’яті тоді, коли рахівник посилань на нього дорівнює нулю.

Не можна забувати й про можливі помилки з такою системою. Найбільш часто виникає проблема «циклу посилань». Вона трапляється тоді, коли умовний об’єкт A містить посилання на об’єкт B, а об’єкт B в свою чергу містить посилання на A. За такої умови жоден з них не буде звільнений з пам’яті, адже кількість посилань у кожного буде як мінімум дорівнювати одному.

Такі ситуації вирішуються використанням спеціальних ключових слів weak та unowned при створенні посилання. За їх використання рахівник посилань на об’єкт не збільшуватиметься, що дасть змогу ARC потім звільнити за потреби з пам’яті.

Підсумовуючи сказане, Swift – це сучасна мова програмування з великою кількістю можливостей та постійно зростаючою аудиторією розробників. Стабільна підтримка та розвиток зі сторони як самої Apple, так і спільноти, дає сильний фундамент для розкриття його повного потенціалу.

2.3 ARKit та основні аспекти роботи з ним

Бурхливий розвиток технології доповненої реальності не міг бути непоміченим Apple. Компанія розглядає технологію як дуже перспективну і спрямовує багато зусиль та ресурсів в її розвиток. В нещодавньому інтерв'ю виконавчий директор Apple Тім Кук сказав, що «AR принесе ще більше користі, тому це критично важлива частина майбутнього розвитку Apple». [26]

Дана віра в технологію підкріпилася релізом фреймворку, що надав можливість нативної розробки з підтримкою доповненої реальності. ARKit був представлений влітку 2017 року на WWDC – щорічній конференції по розробці в екосистемі Apple. Починаючи з того року, даний фреймворк ітеративно стабільно отримував нові релізи та можливості. Сьогодні ARKit 4 – це повноцінний інструмент для створення функціоналу пов'язаного з доповненою реальністю.

ARKit працює базуючись на так званій візуальній одометрії, він не відтворює повністю 3D простір середовища. ARKit максимально утилізує наявні в смартфонах та планшетах Apple сенсори й на основі їх вхідних даних знаходить у просторі точки відносно яких потім розміщує 3D модель.

В iPhone та iPad старіших версій ці сенсори – камера та сенсори руху, а в останніх версіях пристроїв до них також приєднався LiDAR, який доданий саме в цілях покращення ідентифікації простору для різних задач. Принципи роботи даного сенсора з'явились ще в 60-х роках минулого століття. [27] Вони полягають в тому, що він випускає промені лазеру, і вимірює проміжок часу за який вони повертаються. На основі цієї інформації й формуються всі інші знання LiDAR як от глибина кімнати чи відстань до речей.

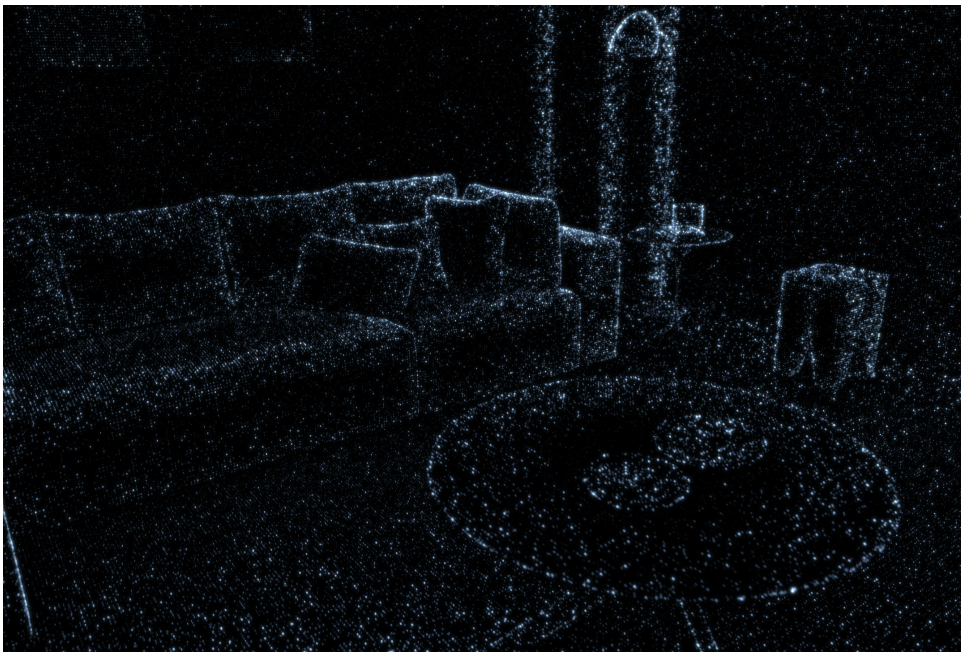

Рисунок 2.6 Приклад моделювання світу через LiDAR [28]

В роботі ARKit робить 3 загальні речі: відстеження, розуміння середовища, та відмальовування об’єктів.

Відстеження моніторить орієнтацію та позицію девайсу в реальному світі, а також може додатково слідкувати за різними об’єктами. В ARKit воно ділиться на два типи: світове та орієнтаційне.

Орієнтаційне відстеження полягає у моніторингу позиції пристрою в сферичному віртуальному середовищі, де позиція девайсу стає центром. Даний тип відслідковування використовує внутрішні сенсори пристрою.

Світове відстеження слідкує за фізичним положенням девайсу й орієнтацією його камери. Порівнюючи з орієнтаційним, світове відслідковує саме переміщення девайсу в фізичному світі.

Розуміння середовища сканує його та надає розробникові додаткову інформацію. ARKit сьогодні вміє визначати поверхню столу, підлоги, чи стіни. Крім того, він оцінює інтенсивність, температуру, напрямлення світла, а також створює відбивання від реальних об’єктів на віртуальні.

Рендеринг об’єктів, що присутній в ARKit, оптимізує процес розробки, адже фреймворк бере на себе логіку розміщення 3D об’єктів в сцені, яка поточно відображається.

За роботи з ARKit робота з 3D моделями стає важливим питанням, а сам фреймворк від Apple підтримує імпортовані. ARKit не займається читанням

моделей, це робота рендеринг двигуна. У його випадку цим займаються SceneKit та RealityKit. На сьогоднішній день є повноцінна підтримка шести форматів: .DAE, .OBJ, .ABC, .PLY, .STL, та найважливішого з них - .USDZ.

USDZ, Universal Scene Description Zip, це власний формат, що був створений Apple в колаборації з Pixar для доповненої реальності. Переваги USDZ в його універсальності серед екосистеми Apple. Модель в даному форматі може бути імпортована як до додатка з використанням технології доповненої реальності, так і для відображення у веб форматі.

2.4 Висновки до розділу 2

У другому розділові було розглянуто основні підходи в сучасній iOS розробці, визначено важливість Human Interface Guidelines, а також архітектурних патернів. Було оглянуто найрозповсюдженіші з них на сьогодні, проаналізовано їх слабкі та сильні сторони.

Також у розділові було розглянуто мову Swift - сучасну мову програмування під iOS, її основні особливості, місце в екосистемі Apple.

В кінці розділу було розглянуто ARKit та основні моменти роботи з ним, описано історію його розвитку та поточні можливості.

Розділ 3. Опис реалізації застосунку

3.1 Аналіз технічного завдання

Технічним завданням була розробка мобільного застосунку, що працював би на смартфонах під управлінням iOS і використовував технологію доповненої реальності.

В процесі визначення предметної області було також встановлено корисну складову додатку, що полягала в його освітній цілі. Під освітнім застосунком мається на увазі такий, що у своїй основі містить функціонал, задача якого надати користувачеві нову інформацію. В такому типові додатків також важливо обрати в якій саме галузі буде надаватись інформація. Загальна ідея та перевірка гіпотез стосуються саме освітнього додатка, конкретна тема була обрана виходячи з актуальних та цікавих на сьогодні речей.

Розглядаючи різні сфери було обрано сферу космосу базуючись на зацікавленості суспільства та поточному ознайомленні з темою. Згідно з дослідженням, проведеним у 2019 році, 80% людей рахують важливим дослідження космосу і 77% вважають, що ця галузь мотивує молодь. При цьому за результатами цього ж дослідження лише 5% опитаних відповіли, що є обізнаними в даній сфері. [29]

Базуючись на цих результатах, для користувацьких матеріалів, які використовуватимуться в додатку, було обрано тематику космосу.

Враховуючи специфіку розробки з технологією доповненої реальності, де необхідно розміщувати віртуальні об’єкти в реальному середовищі, також було вирішено в технічне завдання включити розробку та пропрацювання ідеї створення супутніх доповнень до застосунку.

Фіналізувавши вимоги, була проаналізована кінцева предметна область та визначені потреби користувачів, що покриватимуться застосунком.

До появи та розвитку інтернету у суспільства були певні проблеми з отриманням доступу до відкритої інформації. Спочатку це вирішувалось книгами та їх доступом в бібліотеках, а також живим обміном інформацією. Поступово можливості поповнювались іншими способами. Ключовим є факт

того, що інформацію потрібно було знайти і не було легкого способу отримати необхідні матеріали відразу.

З приходом інтернету та його поступовим розвитком проблема дефіциту інформації була заміщена проблемою його надлишковості. В сучасному світі матеріалу багато, і складність не в тому, щоб його знайти, а в тому, щоб знайти якісний та корисний.

Не є виключенням і тема космосу, яку було взято за основу освітнього функціонала застосунку. Проблематикою даної предметної області є ще й те, що існує багато контроверсійних та неточних джерел, які спекулюють на темі. Саме тому відбір якісного для читання матеріалу необхідний.

Іншою проблемою ознайомлення з чимось новим загалом було визначено утримування зацікавлення користувача та мотивація до отримання інформації. Згідно з дослідженням проведеним Microsoft, продовжуваність концентрації уваги у людей зменшилась з дванадцяти секунд у 2000 році до восьми. [30]

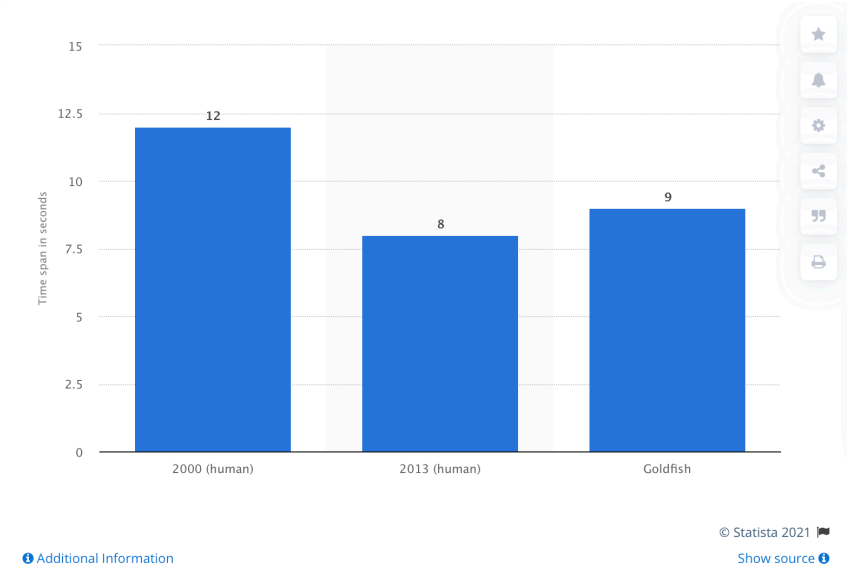

Рисунок 3.1 Статистика продовжуваності концентрації уваги[31]

Саме тому після проведеного аналізу було визначено звернути увагу на фактор зацікавленості користувачів та можливості його втримання.

Продовжуваність концентрації уваги впливає і на загальний досвід взаємодії з додатком. Вона корелює з вимогливістю користувача щодо того як виглядає та поводиться застосунок. Враховуючи тенденції в мобільній розробці, сьогодні під час проектування та створення дизайну необхідно звертати

особливу увагу на простоту та зрозумілість дизайну, а також поведінки його компонентів.

3.2 Пояснення вибору використаних технологій та інструментів

Попри певні стереотипні роздуми про те, що розробляючи мобільний додаток на смартфони під управлінням iOS, розробники не мають такої ж варіативності серед доступних технологій, як і колеги-розробники під Android, на обох платформах є великий простір для вибору.

Розробка мобільного застосунку ділиться на багато різних етапів та частин: розробка логіки, прототипування користувацького інтерфейсу, його створення пізніше, та інші.

Саме тому почали з’являтись різні фреймворки, що створюються членами спільноти з ідеєю полегшення розробки. Вони створюються як і великими командами, так і просто ентузіастами. З їх допомогою можна легше створювати комплексні інтерфейси або ж працювати з вищим рівнем абстракції при створенні мережевого рівня. Вони додаються через так звані менеджери залежностей.

Багато розробників не задумуючись користуються такими інструментами. Під час написання додатку свідомо не було використано жодних додаткових сторонніх фреймворків і весь функціонал написаний вручну.

Дане рішення аргументується тим, що такі інструменти хоч і можуть полегшити задачу в моменті, додатково вносять в проект багато зайвого і потенційно непотрібного в контексті коду. Крім того, такі фреймворки – це ще одна залежність в проекті, яку необхідно враховувати. Додатково до цього збільшується час компіляції програми, адже на моменті збірки необхідно зібрати ще й фреймворк, а також відбувається підвищення об’єму додатку. При цьому це не значить, що вони непотрібні, насправді є корисні та якісно розроблені фреймворки, проте необхідно завжди оцінювати релевантність їх використання.

Резюмуючи сказане вище, в проекті було використано лише надані Apple інструменти та технології.

Однією з використаних технологій є ARKit, який надає можливість створювати додатки з використанням технології доповненої реальності. Даний фреймворк з моменту свого виходу є найпопулярнішим та найбільш інтегрованим в екосистему рішенням що стосується доповненої реальності. Звісно, свою роль грає той факт, що він є розроблений Apple.

Проте крім ARKit існують і інші варіанти в даній галузі. Перед фінальним прийняттям рішення мною як потенційні кандидати додатково розглядались ARCore від Google, та Vuforia. Всі ці варіанти насправді якісно розроблені фреймворк з підтримкою.

ARCore – це пряма відповідь Google на реліз ARKit. Він з'явився на рік пізніше останнього з огляду на зростаючу конкуренцію на ринку. Враховуючи компанію, всередині якої розробили ARCore, очевидно, що початковий акцент було зроблено не на iOS, а на Android. Проте з його розвитком було пророблено роботу в сторону кросплатформенності рішення. Сьогодні ARCore також доступний і на iOS.

Переваги ARCore включають, проте не обмежуються тим, що він є безкоштовним для використання, полегшує розробникам роботу з доповненою реальністю, а також доступний в багатьох середовищах: Android, iOS, Unity3D та Unreal Engine 4.

Проте не дивлячись на наявні переваги, його функціонал на iOS все ще не повний, особливо порівнюючи з ARKit. Також ARCore підтримує імпорт у два рази меншої кількості форматів. Що найважливіше в контексті даної роботи, ARCore не підтримує роботу з форматом USDZ, 3D моделі якого і були використані.

Vuforia – це AR двигун з підтримкою Android, iOS, та UWP застосунків. [32] Даний двигун є одним з найстаріших, що використовуються для створення AR досвіду. Після того як компанію, що його створила, в 2015 році поглинули, Vuforia почала свою роботу в напрямку створення інструментарію для розробки додатків з використанням технології доповненої реальності. Також стала доступна пряма підтримка Unity 3D.

Попри те, що дана технологія не так на слуху в медіа просторі серед мобільної розробки, вона активно використовується та присутня в таких додатках як Mercedes Onboarding та Medical Diagnostics.

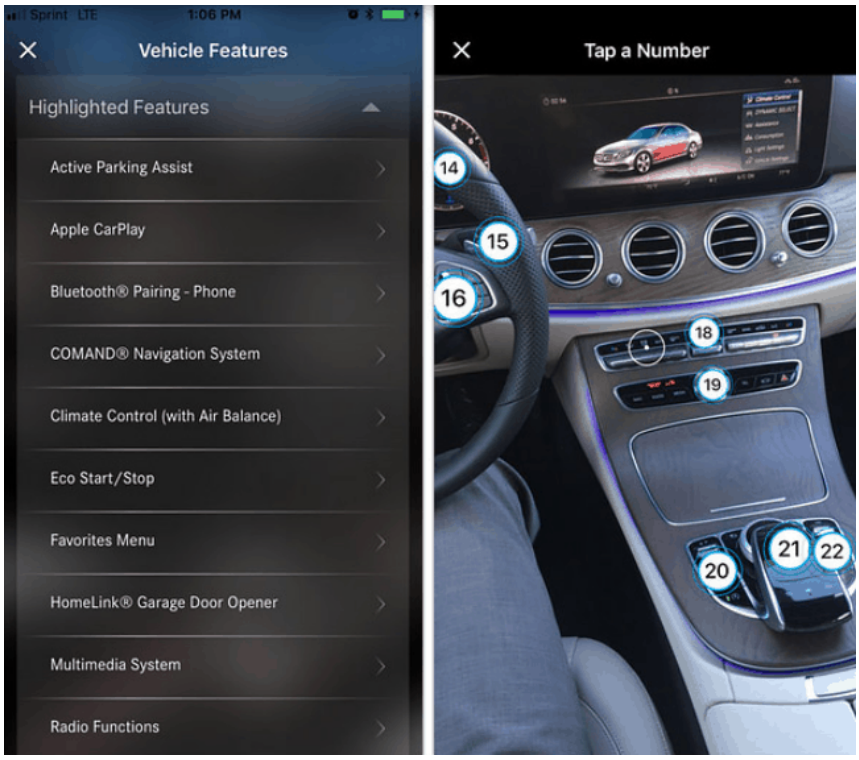

Рисунок 3.2 Приклад створеного додатку з AR на базі Vuforia

Серед переваг Vuforia можна виділити безболісну інтеграцію з Unity 3D, підтримку декількох мов програмування для розробки. Проте дані переваги не були релевантними з точки зору розробки продукту, що розглядається.

Крім цього, Vuforia має свої недоліки. Він не є безкоштовним і є відносно дорогим, особливо для малих або некомерційних проектів. Вартість використання Vuforia становить $42.

Провівши аналіз та розглянувши різні альтернативи було вирішено, що ARKit – найоптимальніше рішення враховуючи платформу розробки та контекст проекту.

3.3 Опис та пояснення вибору середовищ розробки

З точки зору вибору основного середовища створення нативних додатків під iOS, у розробників немає такого ж великого вибору, як у випадку з технологіями.

Звісно, є безліч редакторів коду як от Sublime Text, Atom, чи TextMate. Всі вони надають можливість писати код, в тому числі і на Swift. Проте якщо мова

йде саме про IDE, тобто інтегроване середовище розробки з різними інструментами для збірки, тестування, та полегшеної модифікації коду, то кількість варіантів є дуже обмеженою.

Першим з них є Xcode – IDE, розроблена самою Apple. Дане інтегроване середовище розробки не обмежується лише підтримкою Swift та Objective-C, проте саме розробка під платформи Apple є основною ціллю використання на сьогодні. Він містить всі необхідні інструменти розробника, починаючи від створення проекту і закінчуючи відправкою його в App Store.

В основі візуального інтерфейсу Xcode лежить принцип Single-Window Interface. Під цим мається на увазі, що всі елементи лежать в рамках одного вікна, що змінюється в залежності від поточних задач розробника.

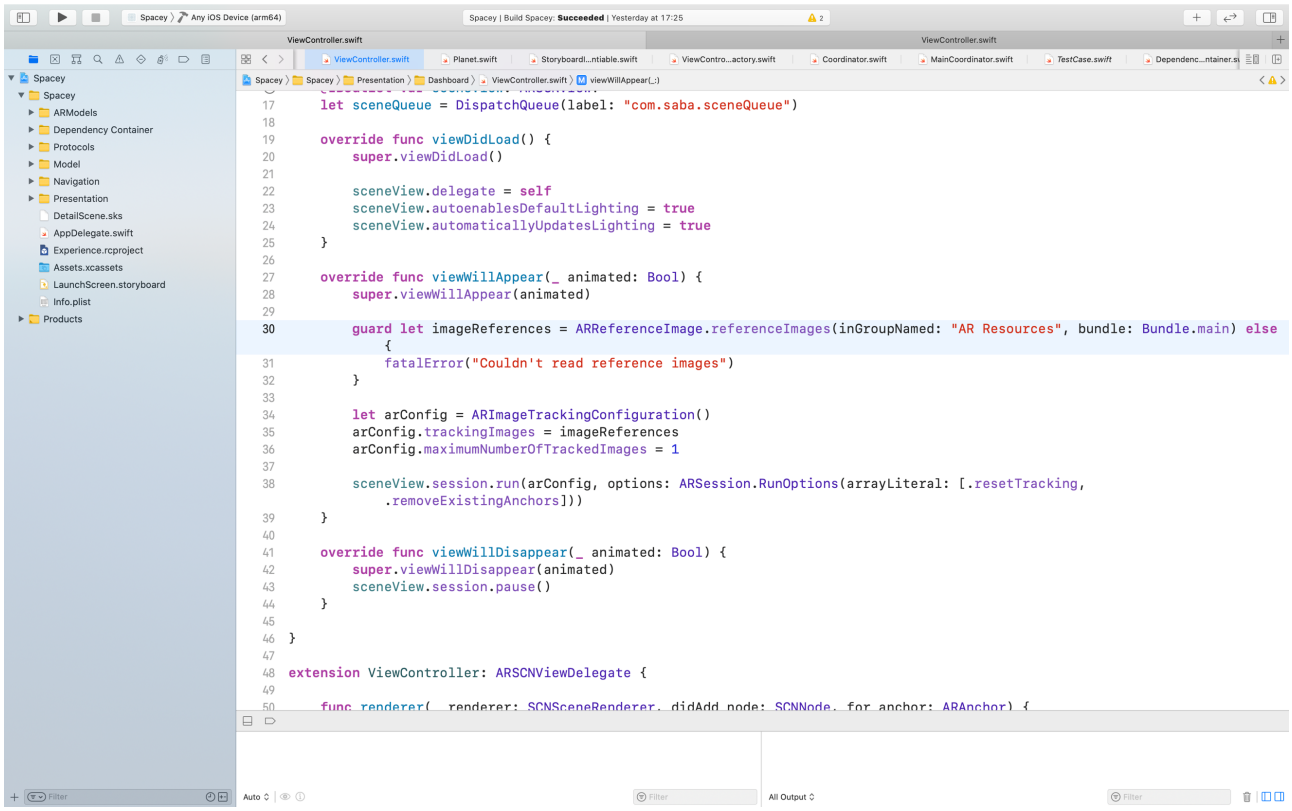

Рисунок 3.3 Основне вікно Xcode

Xcode надає зручну систему керування контролем версій, можливість розбивати основне вікно на декілька секторів, щоб писати код в декількох файлах не перемикаючись між ними, а також вбудовану в себе документацію по Swift та розробці під iOS. Останнє дає розробнику можливість не покидати IDE в разі виникнення запитань.

Окремо варто виділити інтегрований функціонал по тестуванні додатку. Розробник має весь необхідний по тестуванню інструментарій вже вбудованим в Xcode. В процесі Xcode також надає можливості по перевірці поточного навантаження на систему, такого як використання пам'яті, завантаження процесору, на якому тестується додаток. Дане середовище розробки надає функціонал по легкому виявленню витоків пам'яті та пошуку місць в коді, де система найбільш навантажена.

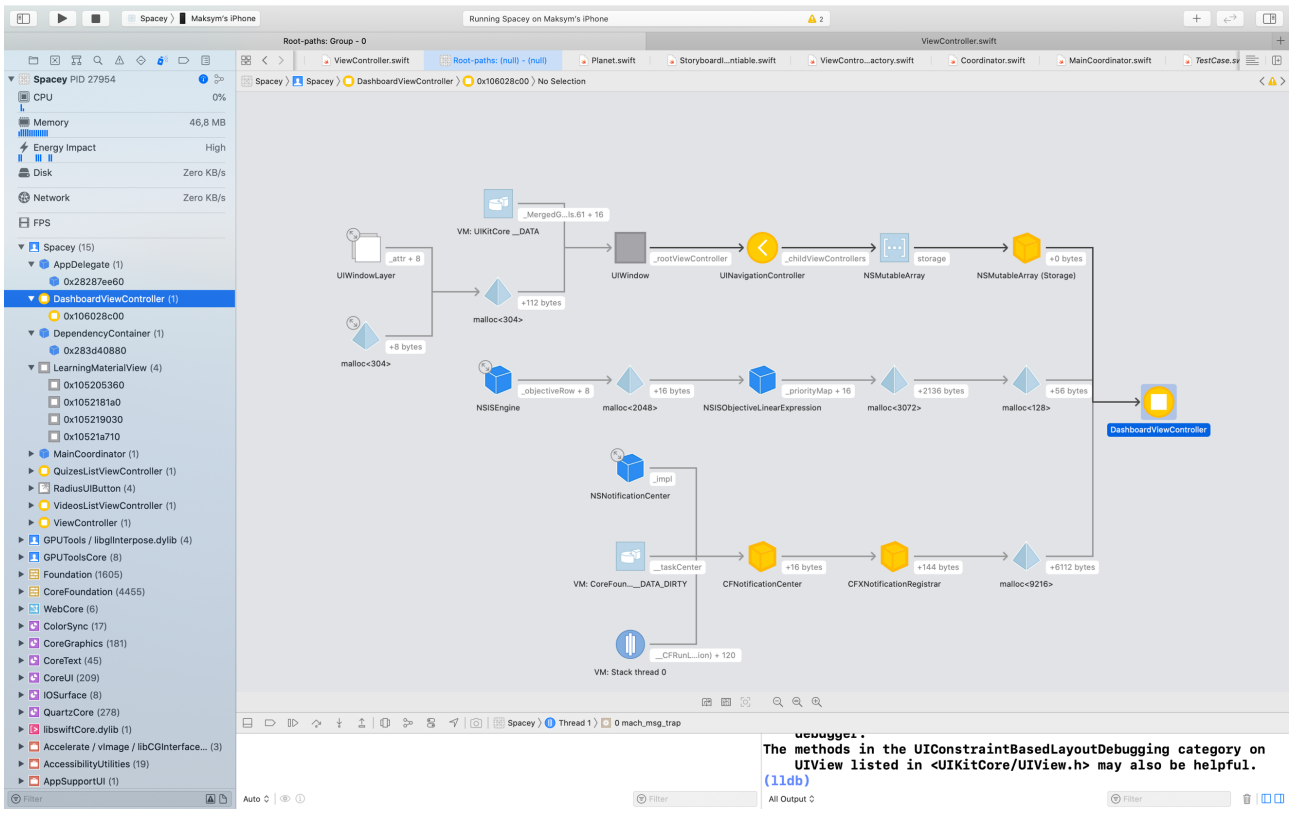

Рисунок 3.4 Пошук витоків пам'яті використовуючи збудований в Xcode граф пам'яті

Також в пакеті з Xcode йде багато додаткового інструментарію. Він винесений в окрему категорію Instruments, адже не є основним для розробки, проте надає багато корисних в тестуванні можливостей.

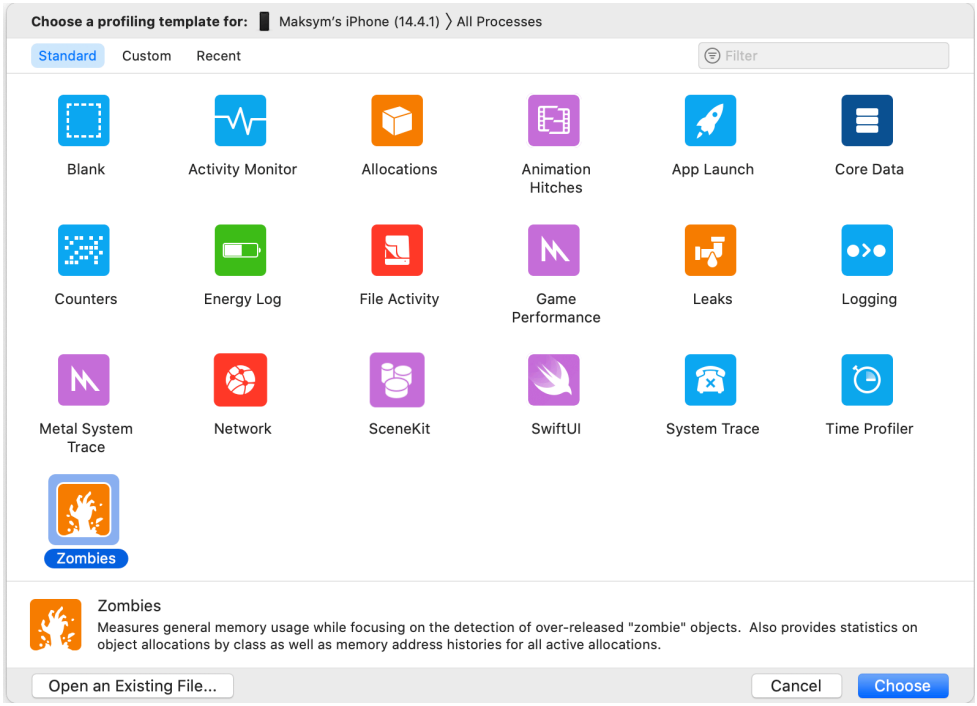

Рисунок 3.5 Додаткові інструменти для тестування

Поміж цим, Xcode має підтримку написання автоматизованих тестів моніторинг за їх виконанням, наданий у вигляді додаткового меню в панелі керування.

Попри те, що попередній функціонал є важливим, він не є найбільш вирішальним у світі iOS розробки. Xcode підтримує вбудовану систему по розробці інтерфейсів – Interface Builder, а також Live Preview в разі розробки використовуючи Swift UI. Надаючи даний інструментарій в руки розробників, Xcode робить себе майже незамінним середовищем розробки під iOS.

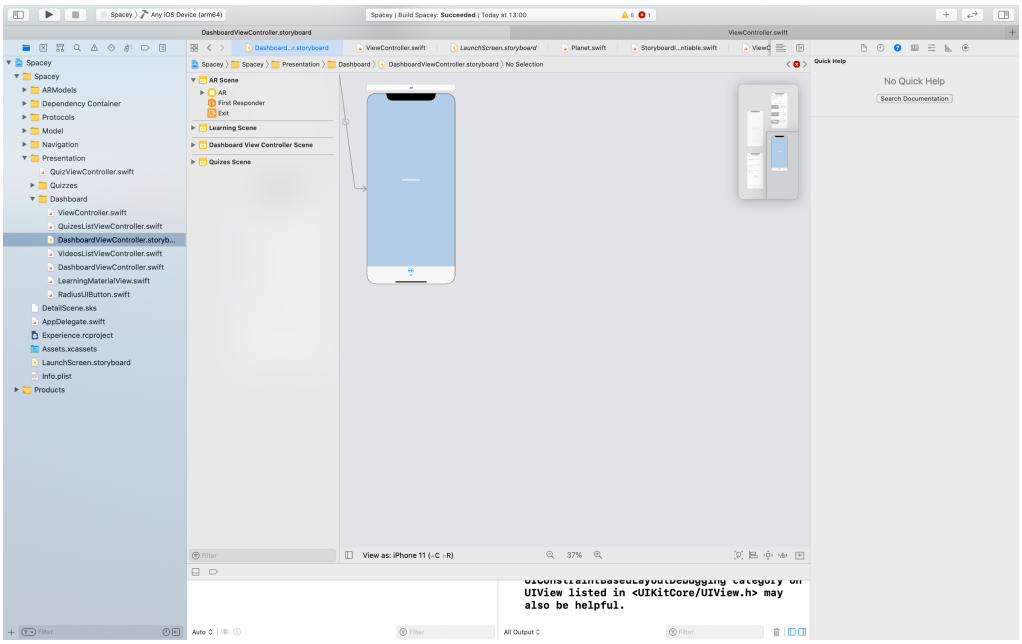

Рисунок 3.6 Interface Builder

Єдиним конкурентом Xcode в такому форматі є AppCode – IDE від JetBrains. Перша та очевидна перевага для тих, хто до цього користувався IDE від JetBrains – це схожість користувацького інтерфейсу. Переходячи на AppCode у розробника не виникне суттєвого адаптаційного періоду, адже логіка розміщення елементів у JetBrains спільна між продуктами.

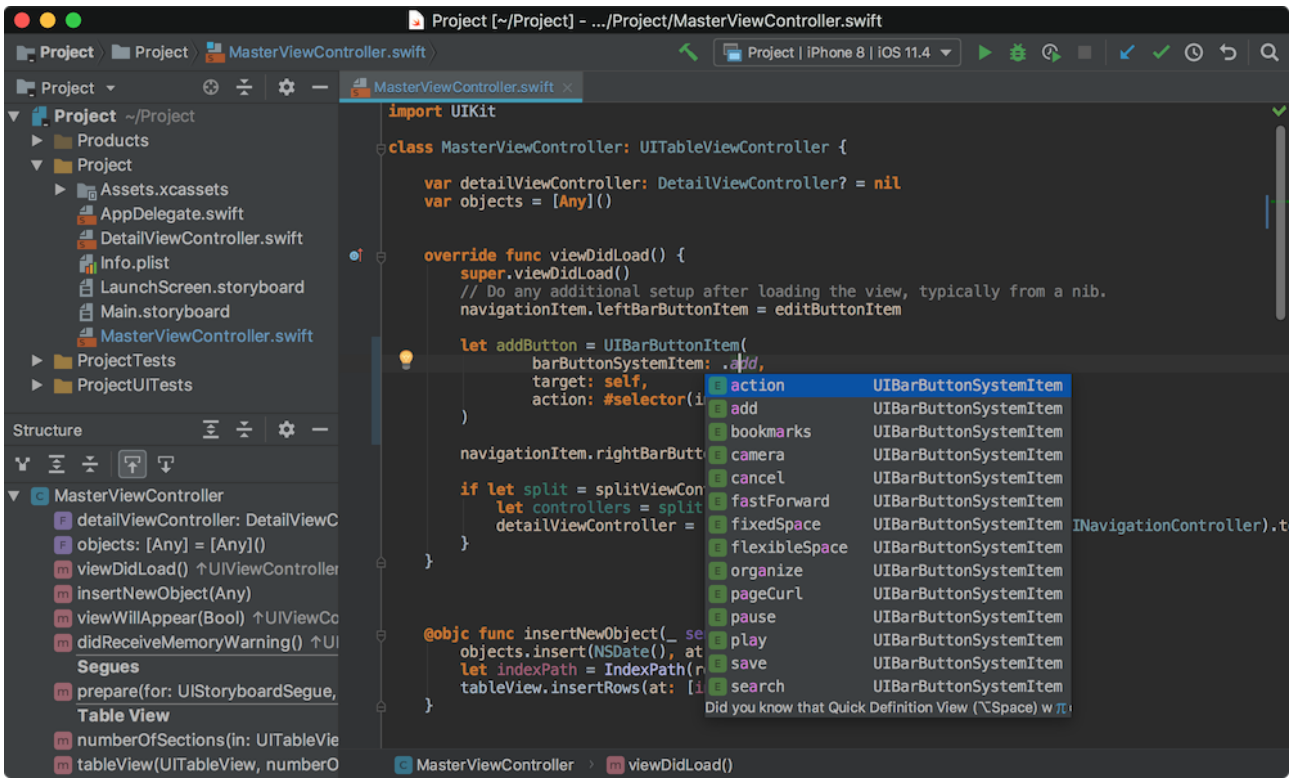

Рисунок 3.7 Основне вікно AppCode

AppCode також працює за принципом Single-Window Interface, має зручні підказки в коді, та організований простір для автоматизованого тестування додатків.

Також, що робить AppCode реальним конкурентом, є те, що він має 100% сумісність з Xcode, а також менеджерами залежностей такими як CocoaPods.

Проте є один суттєвий недолік, який не дає змогу AppCode спробувати витіснити Xcode з лідерів серед IDE в iOS розробці. В інструментарії AppCode немає аналога Interface Builder, а тому при розробці користувацького інтерфейсу розробникові все одно треба переходити в Xcode та налаштовувати його там. Також AppCode не є безкоштовним та коштує $59.

Враховуючи відсутність можливості розробляти користувацькі інтерфейси всередині AppCode, а також об’ємний функціонал Xcode і його підтримку від Apple, в розробці даного продукту було вирішено зупинитись саме на ньому.

3.4 Опис та обґрунтування вибору системи середовища дизайну

В контексті роботи над додатком було вирішено сконцентрувати увагу на створенні власних предметів по ідентифікації місця в просторі для активації технології доповненої реальності. Після аналізу області було визначено, що форма карток з унікальним на них зображенням дозволила б збільшити інтерес до додатка, а також бути маркером ідентифікації в реальному просторі.

Для реалізації цього постало питання вибору середовища дизайну для створення макетів. Вибір полягав між двома популярними на ринку продуктами – Sketch та Figma.

Sketch - це дизайн платформа, що дозволяє створювати як окремі макети, так і цілі дизайн системи для масштабних проектів. [33] Він містить повний функціонал для дизайнерів, а враховуючи, що для контексту задачі не виникало необхідності в чомусь специфічному, мав більше, ніж треба.

З певних недоліків для загальної аудиторії є те, що Sketch доступний тільки на пристроях під управлінням macOS, а також те, що він не є безкоштовним. Ціна на Sketch починається з $9 за місяць. [34]

Figma – це ще одна дизайн платформа, модель якої полягає в хмарному рішенні, що полегшує колаборацію в команді. Завдяки цьому Figma доступна більшій кількості людей та не вимагає macOS для роботи. Попри присутність веб рішення, задля більш оптимальної роботи командою Figma було також розроблене і десктопну версію. Створення нових дизайнів з використанням стандартних елементів доволі інтуїтивне, а основне вікно розробки не переповнене лишніми компонентами. Це дозволяє швидко увійти в процес дизайну і без попереднього досвіду в сфері.

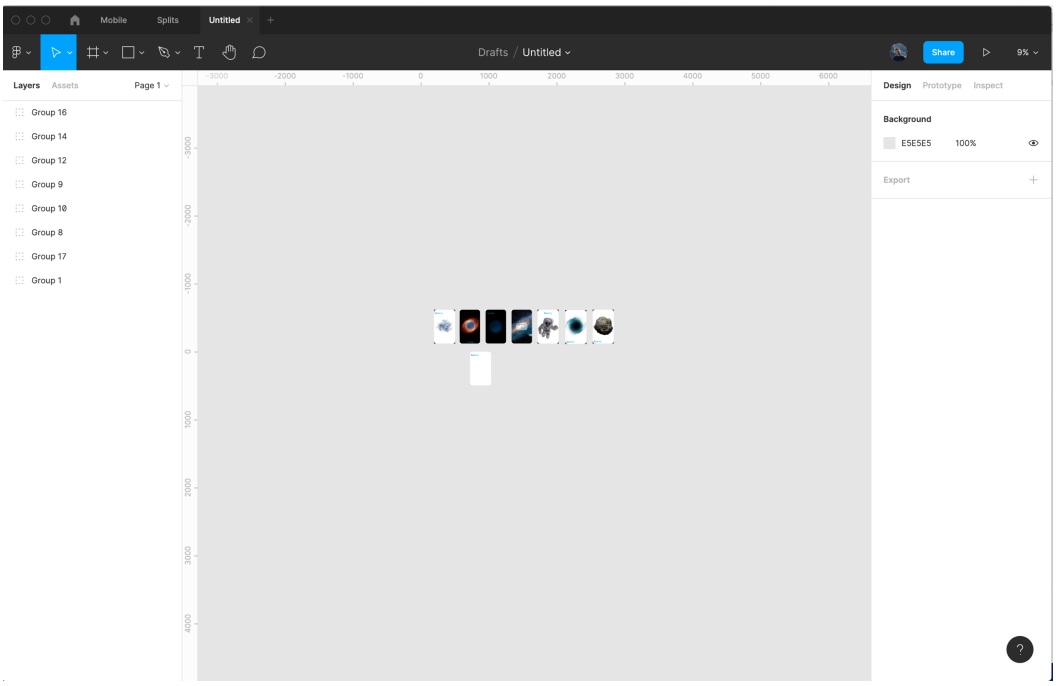

Рисунок 3.8 Основне вікно прототипування у Figma

Перевагою Figma також є те, що її більша аудиторія сприяла тому, що знайти необхідну документацію чи відповідь на запитання певною мірою легше, ніж у випадку зі Sketch.

При цьому її функціонал задовольняє вимоги даного проекту, а сама Figma є безкоштовною. Враховуючи дані фактори саме Figma було обрано в якості дизайн платформи для розробки дизайну карток.

3.5 Опис та обґрунтування архітектури проекту

Архітектура проекту – чи не найважливіша його складова, адже саме від неї залежить швидкість розробки, масштабованість додатку, та ймовірність появи помилок в ньому.

Як вже було згадано в попередньому розділові, сьогодні існує багато різних підходів до розробки. Всі з них намагаються розв’язат проблеми попередніх, та лише невелика частина не додає своїх.

В процесі підготовки до розробки цього застосунку було вирішено відійти від класичного MVC, адже контекст передбачав велику кількість логіки містити в контролерах. Це потенційно могло б привести до їх перенавантаження кодом, що в процесі сповільнило б час роботи, а також могло спричинити неочікувану поведінку в застосункові.

Було обрано скомбінувати різні підходи в розробці мобільних додатків для того, щоб використати збалансований та надійний патерн. В процесі рішень в основу додатку ліг MVVM + C.

Перед тим як його розглянути необхідно ввести розуміння про підходи в розробці, які тісно пов’язані з його реалізацією.

Однією з важливих концепцій, що лежить в суті мови програмування Swift є протоколо-орієнтоване програмування. Воно було вперше представлене у 2015 році на WWDC і є парадигмою, яка лежала в основі при створенні Swift.

Протокол у Swift – це тип даних, що описує необхідний набір функцій чи змінних, які має мати структура даних, що йому відповідає. У Swift вони наділені багатьма можливостями, такими як розширення, наслідування, та композиція. В інших мовах програмування є інтерфейси, що базовим функціоналом є схожими.

Суть протоколо-орієнтованого програмування в тому, щоб починати розробляти систему не створюючи клас чи структуру, а описуючи поведінку в протоколі, і вже потім створювати відповідну йому сутність. З приходом цієї парадигми змінюється і семантика.

Swift був створений як мова, де типи, які передаються за значенням, мають пріоритет у виборі, ніж ті, що передаються за посиланнями. При цьому концепти об’єктно-орієнтованого програмування гірше комбінуються з такими типами, адже, наприклад, наслідування в такому випадку не працює.

Використовуючи протокол ми уникаємо таких проблем, адже типи, що передаються за значенням, можуть наслідувати протоколи, і не обов’язково лише один.

Протоколо-орієнтоване програмування також полегшує тестування додатку і зміну його поведінки в рантаймі, адже виводить абстракцію на рівень вище, і дозволяє підставляти різні конкретні типи залежно від контексту. Це стає

можливим, адже змінні можуть мати тип протоколу. В такому випадку змінна може мати значення будь-якого конкретного типу, що відповідає протоколу.

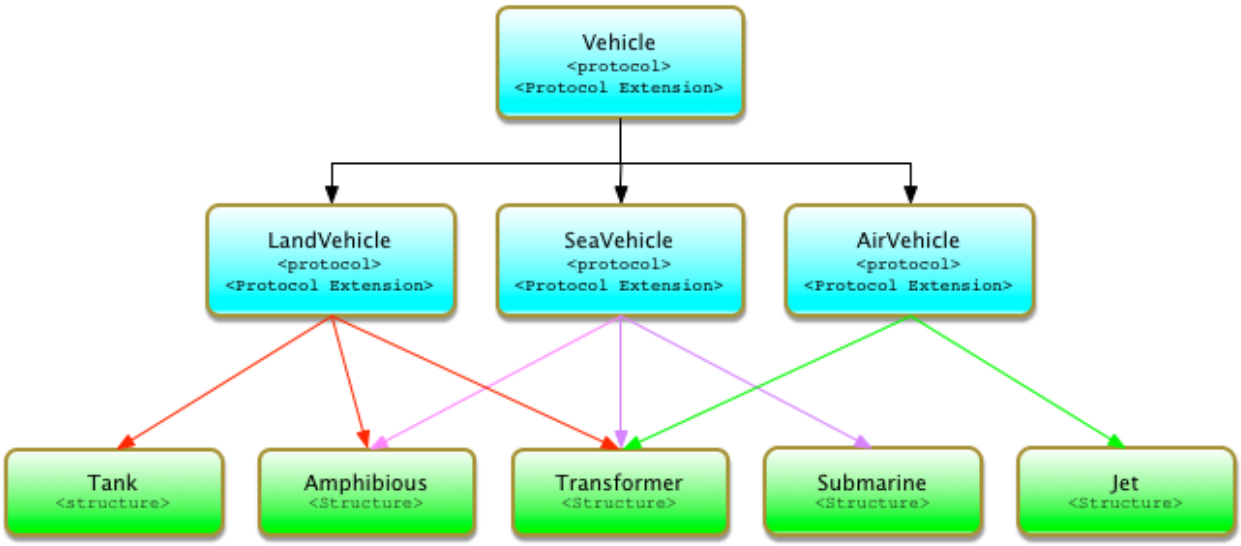

Рисунок 3.9 Приклад протокол-орієнтованої структури

Іншим важливим концептом у контексті розгляду створеного застосунку є ін'єкція залежностей. Це підхід за якого залежності в проекті створюються в одному місці – контейнері залежностей, і вже потім додаються за необхідності до різних структур.

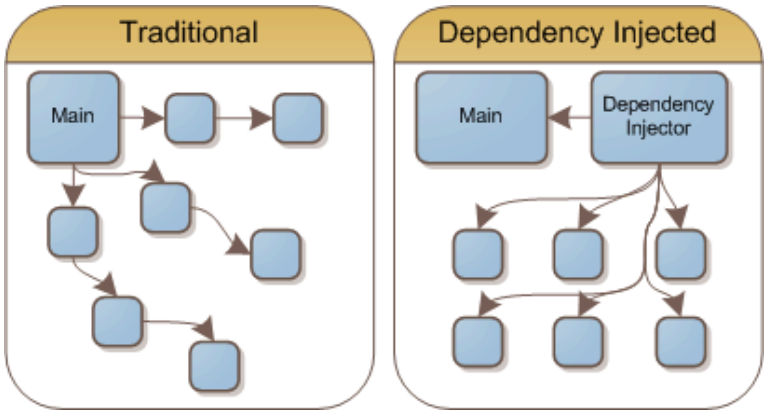

Рисунок 3.10 Порівняння стандартного підходу і ін'єкції залежностей

У розробці не було використано стороннього контейнеру залежностей, який би контролював ін'єкції та їх стан. У використаному підході композитним коренем, тобто місцем, що зберігає стан залежностей, було використано клас

Dependency Container. У ньому відбувається створення об’єктів, яким вже тоді передаються необхідні залежності, що також є створеними всередині контейнеру.

```
func createQuizScreen(forTest test: TestCase) -> QuizViewController {
    let vc = QuizViewController.instantiate()
    vc.viewModel = QuizViewModel(test: test)
    return vc
}
```

Лістинг 3.1 Приклад створення об’єкту в контейнері і ін’єкції залежності viewModel

Комбінація протоколо-орієнтованого програмування з ін’єкцією залежностей робить тестування легким та зручним.

Створення нових об’єктів у даному підході лежить на контейнері й викликається з координатора, проте він не має знати нічого про контейнер залежностей, адже порушаться SOLID принципи та область того, про що знає клас. Саме тому було створено протокол ViewControllerFactory, в якому описані необхідні функції для створення екземплярів контроллерів. Саме його тип і приймає координатор.

```
protocol ViewControllerFactory {
    func createDashboardScreen() -> DashboardViewController
    func createARScreen() -> ViewController
    func createQuizScreen(forTest: TestCase) -> QuizViewController
    func createQuizResults(score: String) -> QuizResultsViewController
}
```

Лістинг 3.2 Фабрика контроллерів

Даний протокол також описує використання патерну Фабрика, що визначає поведінку об’єкту, який має можливість створювати екземпляри контроллерів. Це можуть бути різні класи або їх може бути декілька, проте в контексті розробки даного додатку фабрикою є контейнер залежностей.

Про основні принципи роботи MVVM вже було розказано в розділі про теоретичні відомості. В загальному випадку MVVM працює з парадигмою реактивного програмування, найчастіше зокрема з фреймворком RxSwift. В поточному підході дану парадигму було опущено через більш інтенсивне

використання нею пам’яті. Враховуючи те, що доповнена реальність доволі ресурсомістка сама по собі, не було логічно створювати умови для ще більшого використання пам’яті. В даній MVVM було використано так звані замикання – блоки з кодом, які передаються і викликаються в разі, якщо необхідно зробити щось архітектурно на іншому місці.

С у MVVM + C відповідає за Coordinator. Це додатковий патерн, що бере на себе навігацію в проекті. Логіка переходів на наступні чи попередні екрани переходить від інших сутностей, зокрема контролерів, до об’єктів, що відповідають протоколу Coordinator. Координатор є точкою входу на перший екран додатку, після чого чекає від нього інструкцій по переходу на інше вікно або ж поверненню на попереднє.

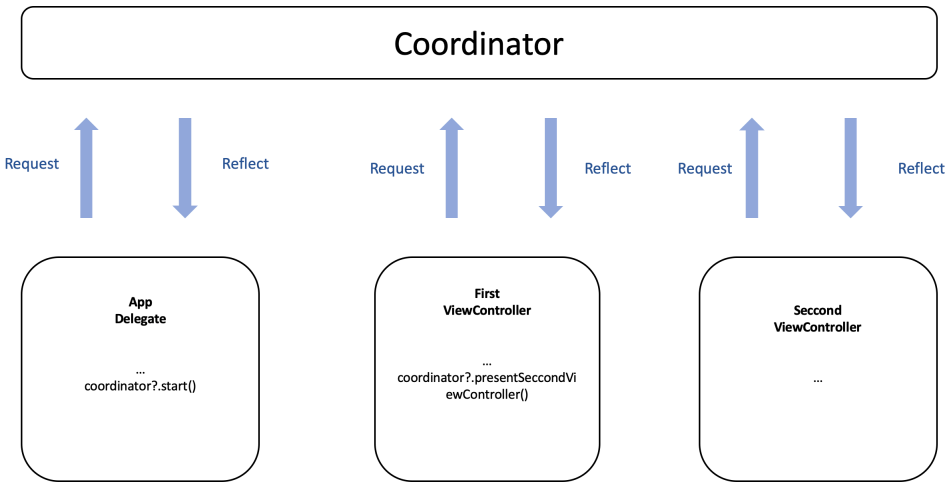

Рисунок 3.11 Загальний принцип роботи патерну Координатор

Перевагою такого підходу є те, що ми звільняємо контроллери від обов’язку знати будь-що про процес переходу на інше вікно та брати в цьому учать. Разом з цим існує виділений рівень абстракції, що знає та розуміє як дані переходи робити.

Щоб вивести рівень абстракції ще далі, координатор – це протокол, який тільки визначає що мають вміти конкретні типи координаторів.

```
protocol Coordinator {
    var navigationController: UINavigationController { get }
    var viewControllerFactory: ViewControllerFactory { get }

    func start()
    func startTest(withType type: TestCase)
}
```

Лістинг 3.3 Структура координаторів

З таким підходом розробник може передавати абстрактний тип Coordinator, всередині якого можуть перебувати різні конкретні реалізації з різною поведінкою. Важливо те, що клієнта координатора цікавить лиш наявність відповідних функцій, а не їх конкретна реалізація.

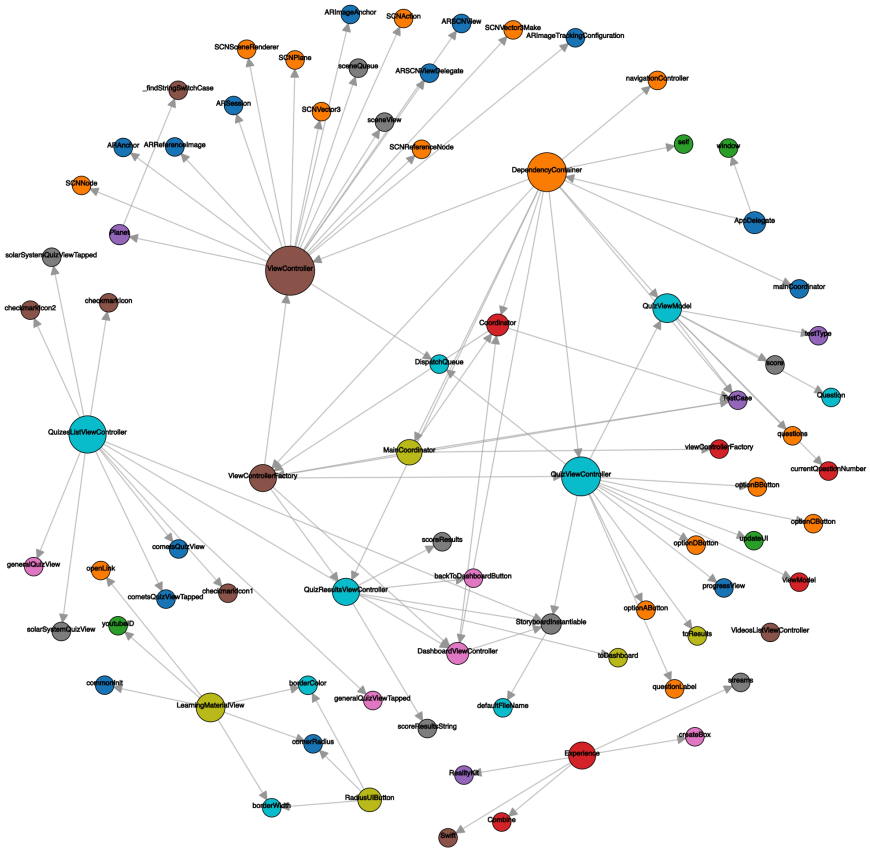

Рисунок 3.12 Повна структура проекту

3.6 Опис розробки застосунку

Після фіналізації технічного завдання та аналізу предметної області було визначено особливостей та потенційні проблеми користувачів.

Задля вирішення проблеми великої кількості матеріалу в додаток, крім його основної функції з доповненою реальністю, було також включено секцію з

матеріалами по вивченню теми. Початковий набір матеріалів був відібраний базуючись на популярності та фаховості розповідачів і складається з відео від National Geographic. В майбутньому життєвому циклі проекту набір матеріалів має поповнюватись регулярно, при цьому не втрачаючи в якості.

Питання зацікавленості користувачів гостро стоїть у світі мобільної розробки. В контексті освітнього застосунку, зацікавленість проявляється не тільки в якості матеріалів, а і в тому, як саме вони подаються користувачеві. Звісно, сам формат доповненої реальності вже сприяє більшому рівню уваги, проте в даній роботі було вирішено зробити крок вперед і також додати інші точки утримання користувача.

В рамках доповненої реальності в Spacey, саме така фінальна назва застосунку, відображається не тільки модель потрібної планети, а ще й інтерактивна дошка з інформацією. Також була розглянута можливість зацікавлення користувача через колекціонування та пропрацьований вигляд супутніх продуктів.

Як згадувалось раніше, для хорошої ідентифікації розміщення об’єкту в просторі можна використовувати маркери. Маркерами може бути будь-що, що легко ідентифікувати. Розробляючи Spacey, була звернена увага на колекційні карткові ігри, і звідти взята ідея щодо створення унікального дизайну карток, які і були б маркерами в реальному світі.

Всього було розроблено 8 унікальних дизайнів, що відповідають кількості моделей планет, які присутні в додатку. Сьогодні Spacey може розповісти користувачеві про кожну з планет Сонячної системи з потенціалом на розширення кількості моделей.

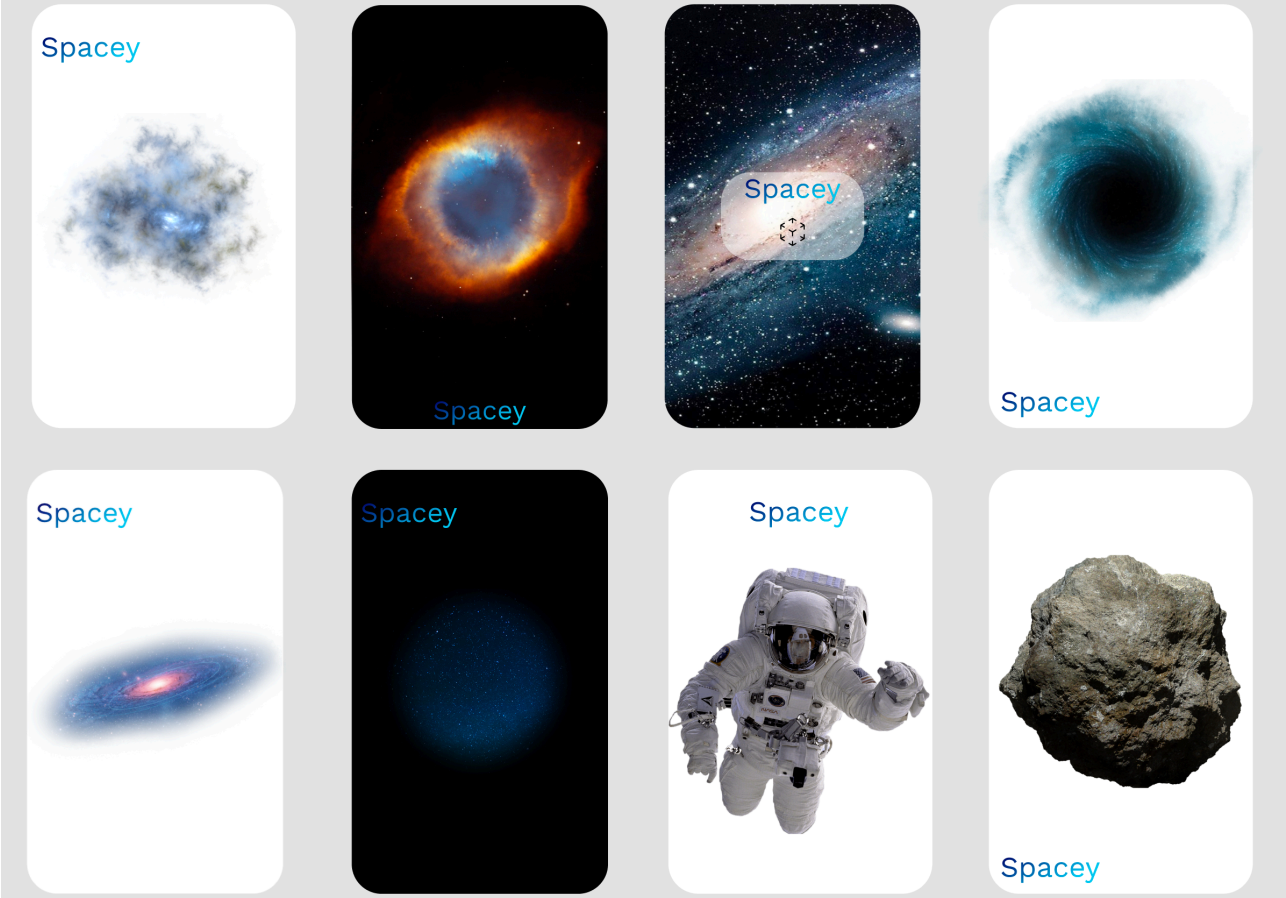

Рисунок 3.13 Створені дизайни карток

Також як доповнення був реалізований функціонал тестів, що відповідають предметній області.

В розробці користувацького інтерфейсу увага зверталась на його простоту та лаконічність, а також на те, щоб користувачеві були інтуїтивно зрозумілі навігація та функціонал. В режимі доповненої реальності, де відеопотік з камери транслюється на весь екран, було звернено увагу на те, щоб користувачеві було зрозуміло стадію роботи з технологією. Саме тому при ідентифікації картки на неї накладається маска, що ідентифікує те, що маркер було знайдено.

3.7 Принципи роботи готового застосунку

Застосунок містить в собі 3 основні сторінки: навчання, AR, та тести. Перехід по сторінках було виконано за допомогою нижнього меню.

На сторінці з навчальними матеріалами представлено список відібраних матеріалів з функцією гортання в разі поповнення новими. Користувач може

прогортати та переглянути наявні матеріали, почитати опис, та перейти до них на сторінку для детального ознайомлення.

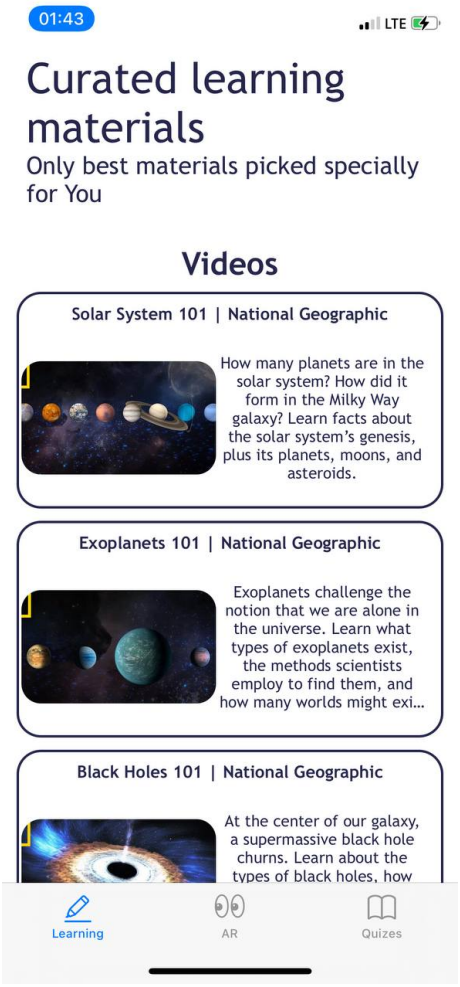

Рисунок 3.14 Меню Learning

У разі якщо це Youtube відео, користувача перенаправляє в додаток. Якщо ж додатку у користувача не встановлено, відбувається переадресація на сторінку в браузері, який встановлено за замовченням.

Наступною є сторінка AR. Вона і містить в собі основний функціонал застосунку, а саме використання технології доповненої реальності. Основну частину екрану займає відеопотік, внизу все ще залишаються пункти переходу на інші сторінки. Додаток в режимі AR готовий ідентифікувати картку в момент її появи у відеопотоці.

Після появи картки вона підсвічується, показуючи те, що є знайденою. Після цього відбувається процес відображення моделі доповненої реальності, а також супутніх матеріалів. Картка замінюється на інтерактивну дошку з інформацією про поточну планету. Користувач отримує можливість читати

детальні відомості відразу в режимі доповненої реальності, без необхідності переходу на окрему веб сторінку.

Також для кожної планети справа від картки розміщується довідкова інформація про розміщення планет в Сонячній системі.

Власне над карткою з’являється 3D модель самої планети, яку користувач може детально оглянути. Модель є прикріпленою до маркеру, а тому весь час поки маркер є ідентифікованим, користувач може роздивлятись її з різних ракурсів. У разі втрати маркера з поля зору 3D модель та інша інформація зникають з екранів.

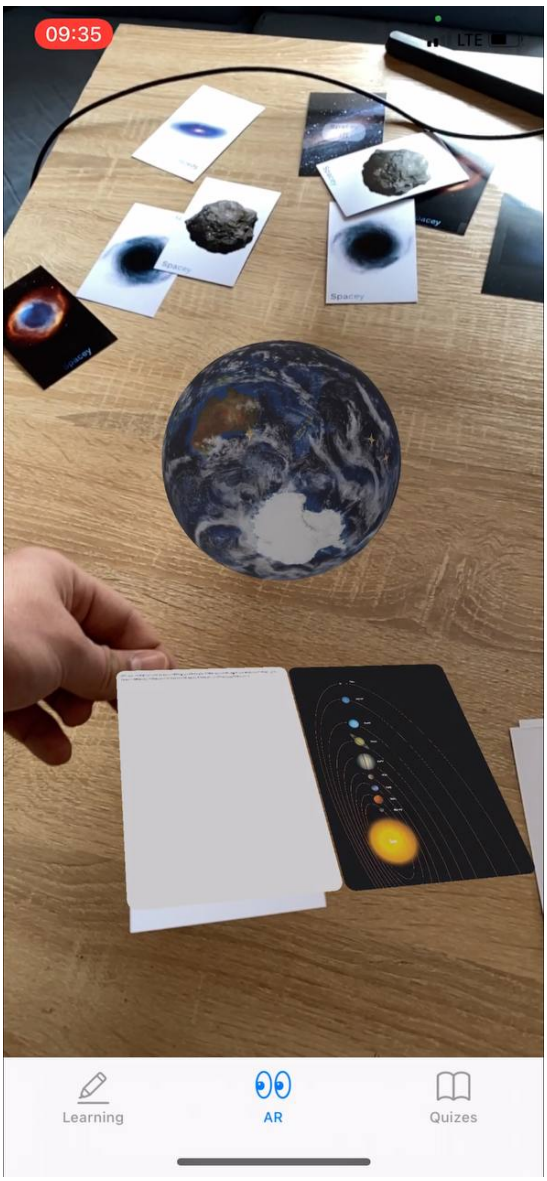

Рисунок 3.15 Приклад ідентифікованої планети

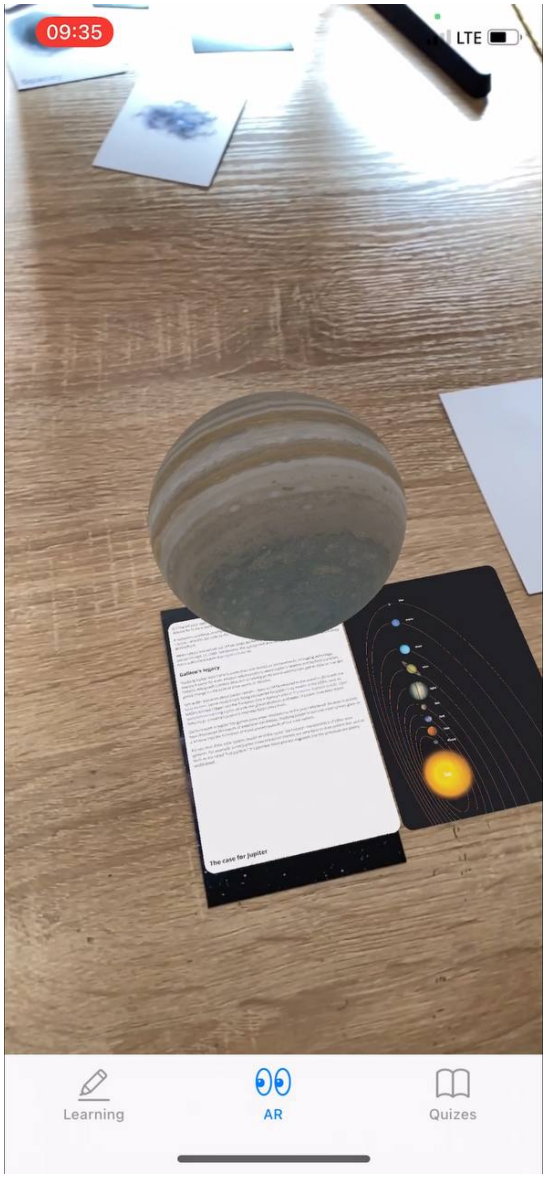

Рисунок 3.16 Приклад ідентифікованої планети

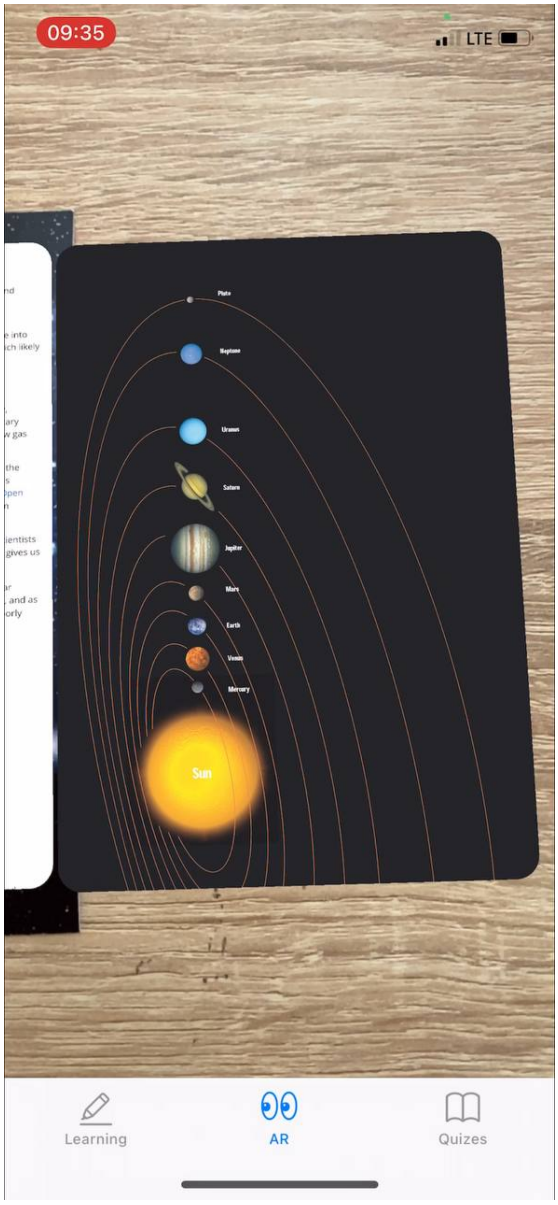

Рисунок 3.17 Зображення Сонячної системи в AR

Рисунок 3.18 Інтерактивна сторінка з додатковою інформацією

Третій пункт – це тести. Вони дають можливість користувачеві перевірити свої знання. Тести містять довільну кількість запитань та різні рівні складності. У рамках роботи було додано три тести, проте в життєвому циклі додатку їх кількість може легко поповнюватись.

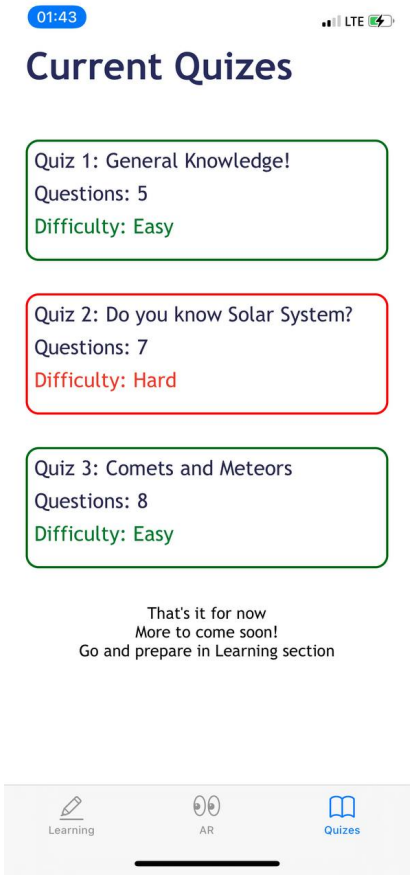

Рисунок 3.19 Сторінка тестів

3.8 Висновки до розділу 3

В третьому розділові було проаналізовано початкове технічне завдання. Базуючись на аналізі, було виділено нові вимоги щодо застосунку.

Також було пропрацьовано предметну область додатку, сформовано його основну ціль як освітню. Після формування цілі додатку, було визначено основні потенційні проблеми в освітніх застосунках, на вирішення або ж уникнення яких варто звернути увагу в розробці.

Після цього було пояснено причини вибору технологій, що використовувались в проекті. Було проведено порівняння популярних фреймворків для роботи з додатковою реальністю та обґрунтовано вибір саме ARKit.

Також в третьому розділові було сформовано задачу розробки карток, що працювали б в парі з додатком та слугували маркером розпізнавання для доповненої реальності.

Були порівняні дві популярні дизайн платформи Sketch та Figma, наведені їх переваги та недоліки. Базуючись на наведених характеристиках було обґрунтовано вибір Figma для розробки дизайну карток.

В розділові було покрито опис архітектури додатку, причини її вибору, а також принципи роботи. В описі архітектури також було зачеплено інші популярні існуючі підходи в розробці, комбінація яких присутня в додатку.

Також було описано процес розробки застосунку та дизайну карток.

В кінці розділу було описано та показано принципи роботи додатку.

Висновки по роботі та аналіз шляхів розвитку

По закінченню роботи було створено iOS додаток Spacey, що використовує технологію доповненої реальності. Початково поставлену ціль було повністю виконано.

Під час проектування застосунку була проаналізована предметна область та сформовано додаткові цілі проекту. Серед цілей стала розробка саме освітнього додатка. Додатково в процесі аналізу знайдених потенційних проблем було додано нові функції, а саме можливість проходити тести та курований список навчальних матеріалів.

Враховуючи розробку додатку з використанням технології доповненої реальності, а також розглядаючи це як можливість зробити продукт більш привабливим та цікавішим для користувача, було вирішено створити унікальний дизайн карток, які слугували б маркерами в просторі. Дизайн було створено та додано для ідентифікації, а самі картки в рамках тестування роздруковано у фізичному форматі.

Під час розробки виникали певні проблеми, як от дефіцит 3D моделей у вільному доступі. Дана проблема була вирішена дослідженням доступних ресурсів, а моделі були знайдені на сайті NASA.

Ще однією проблемою, яку прийшлось вирішувати в процесі розробки, був збір навчальних матеріалів та створення тестів. В процесі роботи було розглянуто, які ресурси є надійними для надання інформації, звідти було взято і потім сформовано кінцеві матеріали.

Даний застосунок має хороший потенціал для розвитку, а також бізнесову привабливість. Першим кроком в його розвитку має бути наповнення більшою кількістю контенту, проте в майбутньому, завдяки архітектурі та використаним підходам, його буде легко підтримувати та поповнювати новою інформацією.

Використання індивідуальних карток може стати частиною бізнес-моделі додатку, адже їх призначення також може бути спрямоване в сторону колекціонування. Саме розповсюдження може організовуватись через продаж.

Список джерел

1. Global average selling price (ASP) of smartphones from 2016 to 2021 [Електронний ресурс] https://www.statista.com/statistics/788557/global-average-selling-price-smartphones/

2. Top Ranked iOS App Store Apps [Електронний ресурс] https://appfigures.com/top-apps/ios-app-store/united-states/iphone/top-overall

3. Breaking down Apple's new augmented reality platform [Електронний ресурс] https://www.theverge.com/2017/6/6/15742736/apple-arkit-augmented-reality-platform-wwdc-breakdown

4. 7 reasons why ebooks are better than printed books, and where to download new titles to read right now [Електронний ресурс] https://www.businessinsider.com/ebooks-vs-books

5. Why Are Online Programs So Popular Right Now? [Електронний ресурс] https://www.chartercollege.edu/news-hub/why-are-online-programs-so-popular-right-now

6. Repetition Works! How To Learn a New Language with Spaced Repetition [Електронний ресурс] https://lingvist.com/blog/spaced-repetition-in-learning/

7. Educational Apps: E-Learning Market Research And Statistics [Електронний ресурс] https://shakuro.com/blog/educational-apps-e-learning-market-research-and-statistics

8. Corporate E-Learning - Global Market Outlook (2017-2026) [Електронний ресурс] https://www.marketresearch.com/Stratistics-Market-Research-Consulting-v4058/Corporate-Learning-Global-Outlook-12571841/

9. 2019 Global Edtech Investments Reach a Staggering $18.66 Billion [Електронний ресурс] https://markets.businessinsider.com/news/stocks/2019-global-edtech-investments-reach-a-staggering-18-66-billion-1028800669#

10. What is augmented reality (AR) and how does it work? [Електронний ресурс] https://www.blippar.com/blog/2018/08/21/what-is-augmented-reality-and-how-does-augmented-reality-work

11. Comparison of marker-based AR and markerless AR: A case study on indoor decoration system [Електронний ресурс] https://www.researchgate.net/publication/318440535_Comparison_of_marker-based_AR_and_markerless_AR_A_case_study_on_indoor_decoration_system

12. Mobile augmented reality (AR) market revenue worldwide from 2019 to 2024 [Електронний ресурс] https://www.statista.com/statistics/282453/mobile-augmented-reality-market-size/

13. Number of mobile augmented reality (AR) active users worldwide from 2019 to 2024 [Електронний ресурс] https://www.statista.com/statistics/1098630/global-mobile-augmented-reality-ar-users/

14. Consumer mobile augmented reality (AR) experiences spending worldwide from 2019 to 2024 [Електронний ресурс] https://www.statista.com/statistics/1222728/consumer-mobile-augmented-reality-spending/

15. How Can Teachers Benefit From Augmented Reality In Education? [Електронний ресурс] https://elearningindustry.com/how-teachers-benefit-from-augmented-reality-in-education

16. Teens, social media & technology overview 2015 [Електронний ресурс] https://www.pewresearch.org/internet/2015/04/09/teens-social-media-technology-2015/pi_2015-04-09_teensandtech_06/

17. NarratorAR [Електронний ресурс] https://www.narratorar.com.au

18. NarratorAR [Електронний ресурс] https://www.narratorar.com.au/download-form

19. Jardin Botanique Grand Nancy [Електронний ресурс] https://apps.apple.com/us/app/jardin-botanique-grand-nancy/id1444413229?l=uk

20. Cumulative number of apps downloaded from the Apple App Store from July 2008 to June 2017 [Електронний ресурс] https://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store/

21. Model-View-Controller (MVC) in iOS – A Modern Approach [Електронний ресурс] https://www.raywenderlich.com/1000705-model-view-controller-mvc-in-ios-a-modern-approach

22. MVVM-Swift [Електронний ресурс] http://cocoadocs.org/docsets/MVVM-Swift/1.1.0/

23. Lightweight MVP architecture in iOS [Електронний ресурс] https://medium.com/omisoft/lightweight-mvp-architecture-in-ios-a16b3da01563

24. iOS Project Architecture: Using VIPER [Електронний ресурс] https://cheesecakelabs.com/blog/ios-project-architecture-using-viper/

25. TIOBE Index for April 2021 [Електронний ресурс] https://www.tiobe.com/tiobe-index/

26. AR is 'critically' important for Apple's future, says Tim Cook [Електронний ресурс] https://9to5mac.com/2021/04/05/tim-cook-talks-ar-interview-kara-swisher/

27. Chapter 2: History of LiDAR [Електронний ресурс] https://www.spiedigitallibrary.org/ebooks/PM/LiDAR-Technologies-and-Systems/2/History-of-LiDAR/10.1117/3.2518254.ch2?SSO=1

28. LiDAR Night Vision: Nachtsicht-App für iPhone 12 und iPad Pro [Електронний ресурс] https://www.iphone-ticker.de/lidar-night-vision-nachtsicht-app-fuer-iphone-12-und-ipad-pro-167627/

29. Attitudes Toward Space Exploration [Електронний ресурс] https://www.ipsos.com/en-us/news-polls/cspan-space-exploration-2019

30. Humans have shorter attention span than goldfish, thanks to smartphones [Електронний ресурс] https://www.telegraph.co.uk/science/2016/03/12/humans-have-shorter-attention-span-than-goldfish-thanks-to-smart/

31. Attention span of humans compared to a goldfish in 2000 and 2013 [Електронний ресурс] https://www.statista.com/statistics/689457/human-attention-span-worldwide/

32. About Vuforia Engine [Електронний ресурс] https://docs.unity3d.com/Packages/com.ptc.vuforia.engine@8.5/manual/index.html

33. Sketch [Електронний ресурс] https://www.sketch.com

34. Sketch Pricing [Електронний ресурс] https://www.sketch.com/pricing/

Додаток А. Функціонал ідентифікації та відображення в рамках доповненої реальності

```
extension ViewController: ARSCNViewDelegate {

    func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {

        guard let imageAnchor = anchor as? ARImageAnchor else { return }

        print("Planet detected: \(imageAnchor.referenceImage.name ?? "")")

        let planet = Planet(rawValue: imageAnchor.referenceImage.name ?? "")

        sceneQueue.async {
            let imageWidth = imageAnchor.referenceImage.physicalSize.width
            let imageHeight = imageAnchor.referenceImage.physicalSize.height

            let scnPlane = SCNPlane(width: imageWidth, height: imageHeight)
            scnPlane.firstMaterial?.colorBufferWriteMask = .alpha

            let mainNode = SCNNode(geometry: scnPlane)
            mainNode.eulerAngles.x = -.pi / 2
            mainNode.renderingOrder = -1
            mainNode.opacity = 1

            node.addChildNode(mainNode)

            self.highlightDetection(on: mainNode, width: imageWidth, height: imageHeight, completionHandler: {

                if let planet = planet {
                    self.displayWebView(planet: planet, on: mainNode, xOffset: imageWidth)
                    self.displayModel(planet: planet, on: mainNode, xOffset: imageWidth)
                }
            })
        }
    }

    func displayModel(planet: Planet, on rootNode: SCNNode, xOffset: CGFloat) {
        guard let url = Bundle.main.url(forResource: planet.rawValue, withExtension: "usdz") else {
            fatalError()
        }

        let referenceNode = SCNReferenceNode(url: url)!
        referenceNode.load()
        referenceNode.position = SCNVector3(0.0, -0.02, 10)

        referenceNode.scale = SCNVector3Make(0.015, 0.015, 0.015)

        rootNode.addChildNode(referenceNode)

    }

    func displayWebView(planet: Planet, on rootNode: SCNNode, xOffset: CGFloat) {
        DispatchQueue.main.async {

            let webViewPlane1 = SCNPlane(width: xOffset, height: xOffset * 1.4)
            webViewPlane1.cornerRadius = 0.25

            let webViewNode1 = SCNNode(geometry: webViewPlane1)
            webViewNode1.geometry?.firstMaterial?.diffuse.contents = UIImage(imageLiteralResourceName: "planetLocation")
            webViewNode1.position.z -= 0.5
            webViewNode1.opacity = 0



            rootNode.addChildNode(webViewNode1)
            webViewNode1.runAction(.sequence([
                .wait(duration: 0.5),
                .fadeOpacity(to: 1.0, duration: 1.5),
                .moveBy(x: xOffset + 0.05, y: -0.05, z: 1, duration: 1.5)
            ])
            )


            let request = URLRequest(url: URL(string: planet.planetURL)!)
            let webView = UIWebView(frame: CGRect(x: 0, y: 0, width: 800, height: 1200))  'UIWebView' was deprecated in iOS 12.0: No long...
            webView.loadRequest(request)

            let webViewPlane = SCNPlane(width: xOffset, height: xOffset * 1.4)
            webViewPlane.cornerRadius = 0.25

            let webViewNode = SCNNode(geometry: webViewPlane)
            webViewNode.geometry?.firstMaterial?.diffuse.contents = webView
            webViewNode.position.z -= 0.5
            webViewNode.opacity = 0

            rootNode.addChildNode(webViewNode)
            webViewNode.runAction(.sequence([
                .wait(duration: 2.0),
                .fadeOpacity(to: 1.0, duration: 1.5),
                .moveBy(x: 0, y: 0, z: 1, duration: 1.5)
            ])
            )
        }
    }
```

```
    func highlightDetection(on rootNode: SCNNode, width: CGFloat, height: CGFloat, completionHandler block: @escaping (() ->
        Void)) {
        let planeNode = SCNNode(geometry: SCNPlane(width: width, height: height))
        planeNode.geometry?.firstMaterial?.diffuse.contents = UIColor.white
        planeNode.position.z += 0.1
        planeNode.opacity = 0

        rootNode.addChildNode(planeNode)
        planeNode.runAction(self.imageHighlightAction) {
            block()
        }
    }

    var imageHighlightAction: SCNAction {
        return .sequence([
            .wait(duration: 0.25),
            .fadeOpacity(to: 0.85, duration: 0.25),
            .fadeOpacity(to: 0.15, duration: 0.25),
            .fadeOpacity(to: 0.85, duration: 0.25),
            .fadeOut(duration: 0.5),
            .removeFromParentNode()
        ])
    }
}
```