КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Мобільний застосунок із пошуку культурних пам'яток України»

Виконав
(Підпис)
Кадиров Кадір Бєкєтович
(прізвище, ім'я, по батькові)

Керівник Плескач В. Л.
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист: До захисту

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Засвідчую, що у цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.

Унікальність тексту - 79 %

Київ – 2022

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

Назва теми: «Мобільний застосунок із пошуку культурних пам'яток України»

Освітня програма: Прикладне програмування

Спеціальність: Комп’ютерні науки

ПІБ	 Підпис
Кадиров Кадір Бєкєтович	

Назва роботи українською та англійською мовами:

Мобільний застосунок із пошуку культурних пам'яток України
Mobile application for search of cultural monuments of Ukraine

Мета бакалаврської роботи: швидкий пошук культурних пам'яток України на основі мобільного застосунку..
План роботи: 1. Теоретичні основи з розвитку культурних пам'яток і відповідні застосунки 2. Архітектура iOS застосунку 3. Проектування, розроблення, тестування, впровадження власного мобільного застосунку пошуку культурних пам'яток

ПІБ, ступінь, звання наукового керівника роботи:

Плескач Валентина Леонідівна д.е.н., к.т.н., проф.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	09.10.2021	виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	21.10.2021	виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	21.12.2022	виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	31.01.2022	виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	30.03.2022	виконано
9.	Подання роботи у першому варіанті	28.04.2022	виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2022**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	27.05.2022	Виконано
	Затвердження роботи в цілому (підготовка письмового відгуку	10.06.2022	виконано

13.	керівника, письмова рецензія на бакалаврської роботу)		
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022 23.06.2022 24.06.2022	

Здобувач вищої освіти  (підпис)

Керівник 

(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи(календарний план дипломної роботи)	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	1
Розділ 1	15
Розділ 2	13
Розділ 3	14
Висновок	1
Перелік використаних джерел	7
Додатки	12

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Кадиров К.Б..				Лист	Листів

Керівн.	Плескач В.Л.			Відомість дипломної роботи	1	73
Н/контр.	Базилюк А.М.					
Зав.каф.	Плескач В.Л.					

Перелік термінів, скорочень

Діловий туризм – це вид туризму, який здійснюють представники компаній/організацій з діловими (комерційними) цілями, або організація корпоративних заходів.

Мобільний застосунок – це програмне забезпечення, спеціально розроблене для певної мобільної платформи (iOS, Android, Windows Phone тощо). Призначений для використання на смартфонах, планшетах, смарт-годинниках та інших мобільних пристроях.

Івент – захід, подія.

Персоналізація – це процес адаптації програмного забезпечення до потреб конкретних осіб з урахуванням їх побажань.

Локалізація – це процес адаптації програмного забезпечення до культури країни. Зокрема, переклад інтерфейсу користувача, документації та відповідних програмних файлів з однієї мови на іншу.

Юзабіліті – простота використання, (англ. usability – буквально «Можливість використання», «Здатність використовувати», «Корисність») — поняття в мікроекономіці, ергономічна характеристика ступеня зручності об’єкта для використання користувачами при досягненні певних цілей у певному контексті.

Інтерфейс – загальна межа між двома функціональними об'єктами, вимоги до яких визначені стандартом; сукупність засобів, методів і правил взаємодії (управління, контролю тощо) між елементами системи.

ОС – операційна система - набір взаємопов'язаних програм, призначених для управління ресурсами комп'ютера та організації взаємодії користувача.

Зміст

АНОТАЦІЯ

Кваліфікаційна робота: ХХ с., Х рис., Х табл., ХХ джерел, Х дод.

Об’єктом дослідження є діловий туризм як явище.

Предметом дослідження є засоби побудови мобільного застосунку ділового туризму.

Проблему дослідження можна сформулювати у питанні: Як підвищити популярність туризму та пошук культурних пам’яток за допомогою розробленого застосунку?

Метою роботи є швидкий пошук культурних пам'яток України на основі мобільного застосунку.

У ході дослідження було проведено: аналіз літератури з проблеми, контент-аналіз мобільних застосунків, аналіз зарубіжних та вітчизняних мобільних подій-застосунків, SWOT-аналіз створюваного продукту.

Новизна роботи полягає в тому, що аналіз зарубіжних та вітчизняних аналогів виявив відсутність цього ІТ-продукту на ринку цифрових сервісів, а також фокус-групове опитування виявило необхідність створення єдиного мобільного інформаційного простору для пошуку культурних пам’яток.

Практичне значення полягає в тому, що створюється унікальний продукт за зарубіжними аналогами, адаптований до інфраструктури міст України, що забезпечує інформаційну зручність і комфорт діловим туристам міста.

Ключові слова: бізнес-туризм, мобільні застосунки, єдиний інформаційний простір пам’яток.

ANNOTATION

Qualification work: XX pp., X fig., X table., XX sources, X appendix.

The object of research is business tourism.

The subject of research is the impact of information technology on the development of business tourism.

The problem of the research can be formulated in the question: How to increase the popularity of tourists and the search for cultural monuments with the help of the developed application?

The aim of the work is to develop the concept of a mobile application for searching cultural monuments.

In the course of the research the following was conducted: analysis of the literature on the problem, content analysis of mobile applications, analysis of foreign and domestic mobile events-applications, SWOT-analysis of the created product.

The novelty of the work is that the analysis of foreign and domestic counterparts revealed the absence of this IT product on the market in a wide range, and the focus group survey revealed the need to create a single mobile information space for searching cultural monuments.

The practical significance lies in the fact that a unique product based on foreign analogues is created, adapted to the infrastructure of Ukrainian cities, which provides informational convenience and comfort to business tourists of the city.

Key words: business tourism, mobile applications, electronic space

ВСТУП

Актуальність дипломної роботи полягає в тому, що мобільні технології з кожним роком набирають обертів і набувають популярності на ринку ділових подорожей. Сьогодні вони широко використовуються серед ділових мандрівників, які за характером роботи мобільні. Згідно з дослідженням, проведеним у 2021 році під час щорічного Business Travel Show в Лондоні, мобільні застосунки займають одну з лідируючих позицій у списку технологій, які найбільше впливають на індустрію ділових подорожей.

Електронний застосунок із пошуку культурних пам’яток України, унікальний ІТ-продукт, який допоможе вітати гостей міста, які займаються діловим туризмом, створить зручний, комфортний та сучасний електронний простір, який зможе ознайомити людей з культурою Українських міст.

Ступінь теоретичної розробленості проблеми: Роботи таких вітчизняних авторів, як Маклашина Л., Морозова Н., Дашкова Є., а також Цацуліна І., Байков В., Дронов В. присвячені культурним пам’яткам України. Аналіз і роботу в предметній області туризму проводили такі зарубіжні автори, як Фрейен Б., Мітчелл С.

Об’єктом дослідження є діловий туризм як явище.

Предметом дослідження є засоби побудови веб-застосунку ділового туризму.

Проблему дослідження можна сформулювати у питанні: Як підвищити популярність туризму та пошук культурних пам’яток за допомогою розробленого застосунку?

Метою роботи є швидкий пошук культурних пам'яток України на основі мобільного застосунку.

Для досягнення цієї мети необхідно виконати такі завдання:

1. Проаналізувати досвід розвитку культурних пам’яток в Україні.
2. Вивчити вплив інформаційних технологій на культурні пам’ятки.

3. Дослідити вітчизняний та зовнішній ринок мобільних застосунків для пошуку культурних пам’яток.

4. Визначити особливості культурних пам’яток в Україні.

5. На основі вивченого створити проект мобільного застосунку для пошуку культурних пам’яток України.

Структура роботи визначається завданнями та матеріалом, що вивчається. Ця підсумкова кваліфікаційна робота складається із вступу, трьох розділів, висновку та списку використаних джерел. У першому розділі розкривається поняття культурної пам’ятки, його передумови, історія та сучасний стан. У першому розділі також розглядається вплив інформаційно-комунікаційних технологій на розвиток культурних пам’яток в Україні та за кордоном, зокрема мобільних застосунків. У другому розділі кваліфікаційної роботи аналізується ринок мобільних застосунків для пошуку культурних пам’яток. На основі дослідження було створено концепцію мобільного застосунку для пошуку культурних пам’яток України.

Новизна роботи полягає в тому, що аналіз зарубіжних і вітчизняних аналогів виявив відсутність цього IT-продукту на ринку мобільних застосунків для користувачів. Ключовими моментами програми є локалізація (продукт створений спеціально для ринку України) і персоналізація (цілеспрямована звернення до користувачів, можливість налаштувати застосунок під свої потреби).

Теоретичне та практичне значення роботи полягає у створенні унікального мобільного застосунка, робота якого може уніфікувати та систематизувати наявну інформацію з різних джерела в одне, його зручно зібрати і подати кінцевому користувачеві. Це дозволить більш продуктивно та ефективно використовувати існуючу інфраструктуру для пошуку культурних пам’яток.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ З РОЗВИТКУ КУЛЬТУРНИХ ПАМ’ЯТОК І ВІДПОВІДНІ ЗАСТОСУНКИ

1.1 Теоретичні основи появи і розвитку культурних пам’яток

Зародження і розвиток культурних пам’яток не тільки в Україні, але і в за кордоном, сприятливо впливає на формування таких факторів, як:

- соціальних;
- політичних;
- економічних;
- культурних тощо.

Культурна пам'ятка — визначна споруда, археологічний об'єкт або витвір мистецтва, що є частиною культурного надбання (культурної спадщини) країни, людства загалом (пам'ятка історії, літератури, мистецтва, мови, права тощо) і охороняється законом. Залежно від наукової, історичної і художньої цінності пам'ятки культури бувають світового, державного і місцевого значення.

Це одна з найбільш високорентабельних і перспективних галузей туризму в цілому, що характеризується стабільним зростанням. Український ринок культурних пам’яток менш відомий серед професіоналів, які традиційно займаються сферою розваг. Але в останні кілька років ця ситуація почала змінюватися на краще. Адже культурні пам’ятки забезпечують інтерес не тільки широкої цільової аудиторії, а й стабільні фінансові інвестиції. За великим рахунком, туризм вже давно невіддільний від економіки більшості країн. Це одна з найбільш прибуткових сфер культури. Вона посідає провідне місце в програмах соціально-економічного розвитку країн, що розвиваються на державному рівні, поряд з такими галузями, як освіта, медицина, сільське господарство та промислова промисловість [1].

Більшість культурних пам’яток виникли в Африці. Нині країни Європейського Союзу (західні компанії та організації) витрачають величезні суми (понад 1 млн доларів на рік) на поїздки для своїх співробітників, які носять

мотиваційний або заохочувальний характер. У будь-якій іноземній компанії стаття витрат на відрядження займає лідируючі позиції після таких оподаткування, кадрове забезпечення, оренда приміщень, зв'язок та інформаційні технології. Це, безумовно, сприяє швидкому зростанню розвитку культури в країні.

1.2 Мобільні застосунки та їх класифікація

Поняття мобільного застосунку (Mobile app) пов'язано з розробленням спеціального програмного забезпечення, основним завданням якого є управління. Це також означає, що мобільні застосунки можуть мати різні платформи та операційні системи, для яких вони написані. Але всіх їх об’єднує те, що вони покликані слугувати для виконання певних функційних завдань. Більшість програм можна не тільки попередньо встановити на пристрій за замовчуванням, але і завантажити з інтернет-магазинів. Наприклад, такі як:

- App Store;
- BlackBerry App World;
- Google Play;
- Магазин Windows phone тощо.

До того ж багато застосунків можна завантажити не тільки платно, але і безкоштовно.

Основним призначенням мобільних застосунків колись була перевірка листів на поштових скриньках. Але попит на них був настільки великий, що їх можливості значно розширилися. Тепер мобільні застосунки допомагають виконувати операції, пов’язані з переглядом відео, ігор, спілкуванням, GPS та використанням Інтернет-ресурсів.

Термін «мобільний застосунок» став настільки популярним, що в 2010 році Американське діалектичне товариство прийняло рішення зарахувати це слово до словника («слово року»).

Мобільний застосунок — спеціалізоване програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях.

Сьогодні ринок постачання та впровадження мобільних застосунків дуже розвинений. Згідно з аналітикою App Annie, вона продовжує зростати. Таким чином, на 2022 рік запланований приріст складатиме близько 30% і сягне $166 млрд. Крім того, $65 млрд – це витрати користувачів на придбання застосунків або підписку на них, а $101 млрд – гроші, вкладені розробниками чи рекламодавцями з метою просування продукту.

За словами Саміра Сінхи, директора з ринкової розвідки App Annie, у 2021 році користувачі витратили на додатки $52 млрд, а рекламодавці – $77 млрд. Звідси слідує, що основним драйвером зростання ринку мобільних застосунків є реклама. При цьому левова частка доходу розподіляється між соціальними мережами, іграми та відео сервісами. Величезну роль у зростанні доходів відіграють такі гіганти як соціальна мережа Facebook або безкоштовні платформи потокового відео (наприклад, YouTube). Велике значення має і реклама, вбудована в мобільні ігрові застосунки, найпопулярнішим форматом яких є відеокліпи. З більшої частини реклами просування відомих брендів становить близько 12,5%. Але обіг зростає з кожним роком, виводячи мобільні застосунки на все більше використання людством.

Наведемо класифікацію мобільних застосунків:

• ***бізнес-застосунки***. За допомогою таких застосунків компанія-замовник може керувати персоналом, розробляти нові бізнес-стратегії, отримувати більший прибуток і організовувати роботу за планом. Зазначимо, що багато з цих програм використовують частину ігрової структури. Переваги таких застосунків для керівників також важко переоцінити, оскільки саме з них можна отримати велику кількість необхідної інформації;

• ***контенти***. Вони користуються великим попитом як у споживчій сфері, так і серед споживачів. Ця категорія включає всі види плеєрів і застосунків для перегляду відео або фотографій. Також в цій категорії є всі види музичних програм;

• ***ігрові***. Це найпоширеніший тип програми, оскільки з усіх вимог важлива лише одна – установка програми на мобільний пристрій. Це і аркади, і настільні ігри (шашки, карти, шахи або доміно), і популярні стратегії. Великим попитом користуються також спортивні, сімейні або розвиваючі ігри;

• ***промо-застосунки***. Їх основні відмінні риси – високий рівень креативності при мінімальному смисловому навантаженні. Наприклад, у цій категорії цілком може з'явитися віртуальна запальничка. Їхнє головне завдання – розважити та відволікти клієнтів від повсякденних та побутових труднощів;

• ***нативні***. Такі застосунки пишуться з використанням новітніх технологій в одній операційній системі. Основними і найпоширенішими на сьогодні є: програми Windows, Android, IOS, Blackberry тощо відрізняються способом написання. Так, наприклад, для Android найпопулярнішим мовою програмування є Java, хоча в той же час додатки для систем iOS пишуться на основі Swift або Objective C;

• ***соціальні мережі***. Цей вид, безумовно, користується великим попитом у споживачів, яких дуже багато. Розвиток нових мереж активно ведеться як у нас, так і за кордоном. Вони дуже зручні не тільки для спілкування, але і для розміщення супутньої реклами. Багато компаній дуже активно використовують соціальні мережі, оцінивши всі їх переваги та переваги.

Використання гібридних програм зараз надзвичайно поширене. Вони можуть поєднувати основні функції нативних застосунків і при цьому максимально наближені до веб-застосунків (також можуть використовувати програмне забезпечення мобільного пристрою у своїх цілях і базуються на кросплатформенності). Вони також мають ще одну особливість.

Незважаючи на те, що оболонка для гібридних застосунків пишеться так само, як і для нативних, створення гібридних обходиться набагато дешевше, і вони пишуться набагато швидше за часом. Оскільки начинка для таких застосунків написана на HTML5, вони мають можливість функціонувати на різних пристроях і під різними операційними системами.

Нині завдання розробки нових застосунків залишається актуальним. Адже, це найбільш затребуваний сегмент ринку мобільної коменції. Це, безсумнівно,

пов’язано з потужним входженням мобільних пристроїв у повсякденне життя людини. За допомогою мобільних пристроїв люди звикли отримувати інформацію та ділитися нею з іншими, робити покупки, спілкуватися та знайомитися один з одним. Крім того, очевидним є і практичне використання їх у бізнес-середовищі. Сюди входить моніторинг стану прибутків і справ, відстеження важливих повідомлень і можливість обробки великих обсягів інформації, доступ до платіжних сервісів і перегляд аналітичних звітів.

Але цим переліком функціональність мобільних застосунків не вичерпується. Вони ще більш незамінні в туристичній сфері. Через них можна замовити або попередньо забронювати номер, дізнатися програму культурних заходів певного міста, викликати таксі та вибрати столик у ресторані, зробивши замовлення до приїзду.

1.3 Інформаційні технології розробки мобільних застосунків

Технологічний стек, який використовується при розробці мобільного застосунку, включає фреймворки та бібліотеки, необхідні для ефективної та швидкої розробки, що дозволяє з мінімальною кількістю даних та обмеженими системними вимогами інтегрувати мобільний застосунок з іншими сервісами для відображення даних у інтерфейс в режимі реального часу.

Swift став мовою програмування мобільних застосунків. За твердженнями TIOBE, зараз Swift займає 14 позицію у світовому рейтингу мов програмування. Він запозичив ідеї з Objective-C, Rust, Haskell, Ruby, Python, C#, CLU та інших мов. Його творцем є Кріс Латтнер. Компанія позиціонує цю мову як швидший і ефективніший спосіб створення програм для iPhone, iPad, Mac, Apple Watch і Apple TV. Google оголосила про плани зробити Swift так званою «першою мовою» для Android.

Рисунок 1.1 – Swift

Apple є однією з найбільших компаній, що виробляють смартфони, ноутбуки та іншу електроніку, але на відміну від більшості інших компаній, Apple сама пише програмне забезпечення для своїх пристроїв.

Apple надає інфраструктуру для легкого та зручного написання швидких та енергоефективних програм для iOS або macOS. Ця інфраструктура включає середовище розробки Xcode, мови Swift і Objective-C, а також різні бібліотеки та фреймворки.

Також щорічно в Каліфорнії проходить Всесвітня конференція розробників (WWDC) – всесвітня конференція для розробників на платформах Apple. На цих конференціях оголошуються нові продукти, розробки та оновлення платформ Apple.

Swift замінив старий і громіздкий Objective-C, який був розроблений ще в 1980-х. Apple запускає навчальні посібники Swift, а на iPad є застосунок для навчання дітей. На WWDC про оновлення та вдосконалення бібліотек і фреймворків розповідають на Swift, а не на Objective-C, тому вам потрібно хоча б трохи знати Swift, щоб дізнатися про все нове. Він вимагає менше коду, є більш читабельним, безпечним та інтерактивним.

Swift також має свої особливості, які впливають на продуктивність. Тому для того, щоб написати швидку та ефективну програму на Swift, вам потрібно з’ясувати, як вона працює всередині – як керує пам’яттю, які сутності використовує. Час компіляції також є важливою властивістю мови - якщо він великий, то розробка програми чи іншого продукту сильно сповільнюється. Тому вам потрібно знати, що може уповільнити компіляцію.

1.4 Програмні продукти сфери туризму

Для аналізу зовнішнього та вітчизняного ринку мобільних застосунків для пошуку культурних пам’яток необхідно визначити критерії, за якими буде проводитися аналіз і за якими характерезують якість ІТ-продукту.

Масовість – один з головних критеріїв. Розраховується за кількістю завантажень в електронному магазині.

Юзабіліті – це міра якості користувацького досвіду, отриманого під час взаємодії з продуктом або системою, наприклад веб-сайтом, програмним додатком тощо. Це досвід, отриманий під час використання програми, його необхідність. Дослівно «можливість використання».

Ергономіка – це простота використання програмного забезпечення. Правильне розташування деталей інтерфейсу. Логіка дій для отримання результату використання.

Рейтинг магазину – це відгуки клієнтів, які завантажили застосунок, а також «зірковий рейтинг» - оцінка заявки в електронному магазині. У той же час кількість завантажень і рейтинг можуть мати велике значення.

Також для аналізу необхідно мати перелік заявок на розгляд. Заявки взяті з електронного магазину застосунків Google Play, у добірці є як зарубіжні, так і вітчизняні програми. Надано 10 заявок зі сфери туризму, п’ять із них стосуються подій - CMMA2017, Город.IT, Афіша Омська, Виставки Одеси та AroundMe, а п’ять створені для більш широкого поняття, ніж просто для бізнесу - Evernote, Google Maps, TripIt, TripAdvisor і AirBNB. Спочатку давайте зробимо короткий опис цих програм і подивимося, як вони виглядають.

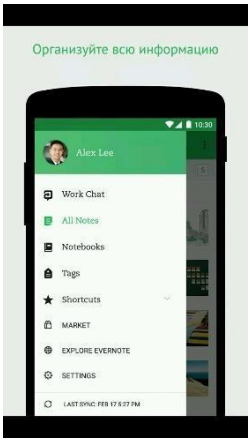

Рисунок 1.2 – Мобільний застосунок Evernote

1. Evernote. Як тільки ви починаєте готуватися до подорожі, ви вже відчуваєте потребу в якомусь єдиному сховищі всіх тих підтверджень бронювання, електронних квитків та рекомендацій щодо того, де жити, що їсти тощо. Застосунок Evernote легко синхронізує інформацію, що зберігається на різних пристроях. Навіть, якщо з вашим смартфоном або планшетом щось трапиться (наприклад, він загубиться або перестане функціонувати), усі необхідні дані можна знайти в Evernote.

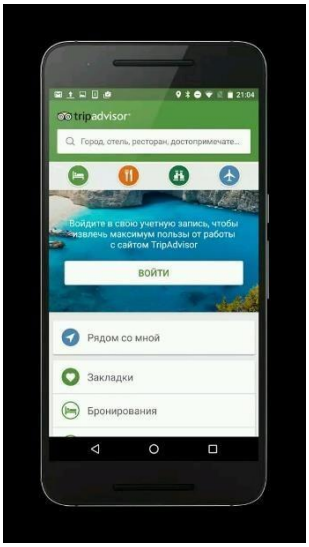

Рисунок 1.3 – Мобільний застосунок TripAdvisor

2. TripAdvisor – ця програма дуже інформативна та проста у використанні. Пропонує нормальний контент та відгуки інших користувачів. Вона має ряд великих незручностей з боку дружності інтерфейсу для кінцевого користувача.

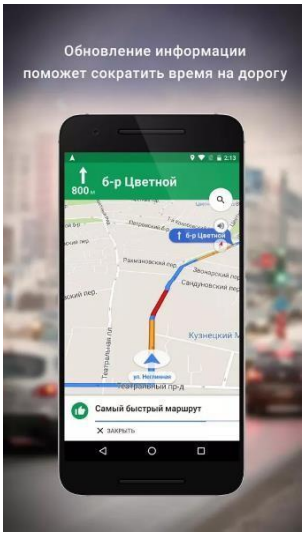

Рисунок 1.4 – Мобільний застосунок Google Maps

3. Google Maps – Офлайн-карти в найпопулярнішому сервісі Google дозволяють заздалегідь завантажити дані про певну місцевість і позначити місця, які ви збираєтеся відвідати. Це дозволяє орієнтуватися в просторі, не завантажуючи байти через мобільний Інтернет.

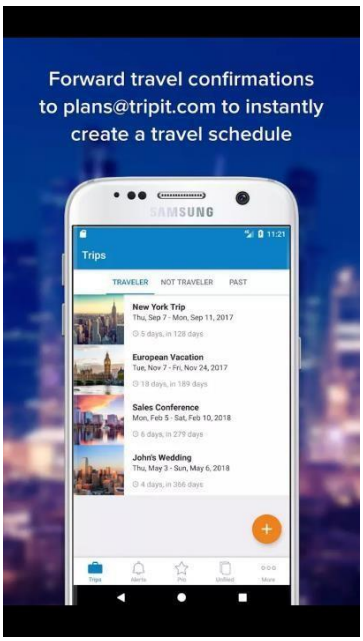

Рисунок 1.5 – Мобільний застосунок TripIt

4. TripIt - дозволяє зручно розташовувати плани так, щоб вони зберігалися в одному місці.

Рисунок 1.6 – Мобільний застосунок AirBNB

5. AirBNB – зручний дизайн, багато ідей, щоб отримати натхнення та знайти дивовижне місце для відпочинку. Зробивши бронювання, ви можете продовжувати використовувати застосунок, щоб зв’язатися з власником помешкання (наприклад, щоб підтвердити дату та час прибуття). За допомогою цієї ж системи ви можете залишатися на зв'язку з власником протягом усього перебування в його будинку.

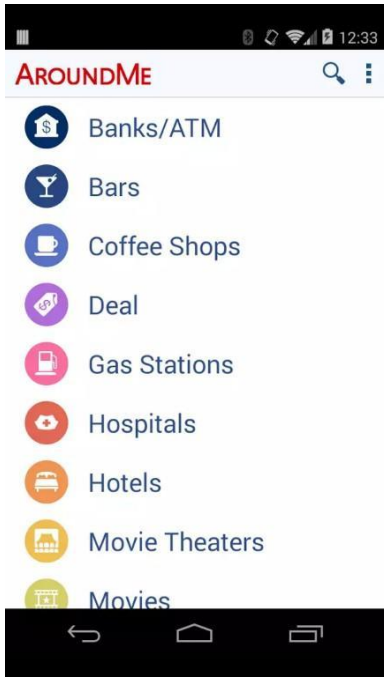

Рисунок 1.7 – Мобільний застосунок AroundMe

6. AroundMe - ще один варіант для мобільного застосунку, який дозволяє користувачам шукати поблизу цікаві та потрібні місця: театри, ресторани, готелі, автостоянки та лікарні.

Всі вище розглянуті застосунки є відомим програмними продуктами на ринку послуг, їх корисно мати під час подорожей до багатьох різних міст і країн, але, на жаль, кожний з них відповідає лише за певну функцію і відповідно до згаданої класифікації. Можна стверджувати, наприклад, що Evernote — організатор, TripAdvisor — путівник, Google maps — карта, TripIt — планувальник, а AroundMe — хороший агрегатор актуальної інформації за певними критеріями.

Крім цього, жодна з перерахованих програм не відноситься до категорії подій. Зараз ми їх розглянемо окремо.

Рисунок 1.8 – Мобільний застосунок SMMA2017

7. У рамках міжнародного китайського саміту CMMA2017 створено застосунок для показу подій, висвітлення питань, які будуть підніматися на саміті. Є можливість побачити розклад, доповідачів, карту, а також спланувати час перебування на саміті.

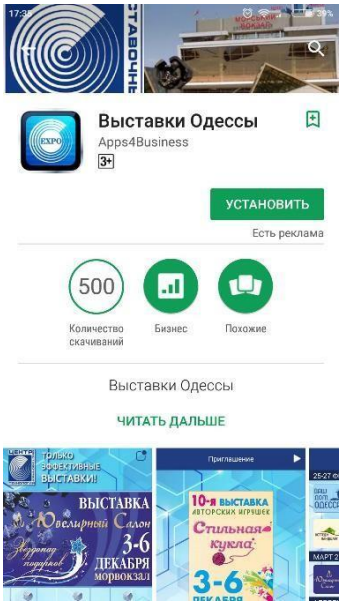

Рисунок 1.9 – Мобільний застосунок Виставки Одеси

8. Мобільний застосунок «Виставки Одеси» - охоплює майже всі афіші виставок міста Одеси. Простий дизайн і слабка функціональність відлякують використання цієї програми.

Висновки до розділу 1

У першому розділі надано визначення культурної пам'ятки та мобільного застосунку. Описано програмні продукти у вигляді мобільних застосунків, які пов'язані зі сферою туризму і культурних пам'яток.

РОЗДІЛ 2

АРХІТЕКТУРА iOS ЗАСТОСУНКУ

2.1 Види архітектур застосунку (MVC, MVVM, VIPER, Clean Swift)

Архітектура MVC

Статична сторінка на HTML не вміє реагувати на дії користувача. Для двосторонньої взаємодії потрібні динамічні веб-сторінки. MVC - ключ до розуміння розробки динамічних веб-застосунків, тому розробнику потрібно знати цю модель.

MVC розшифровується як модель-подання-контролер (Model-view-controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за ці застосунки, інший відповідає за зовнішній вигляд, а третій контролює роботу застосунка. Компоненти MVC відображено на рисунку 2.1.

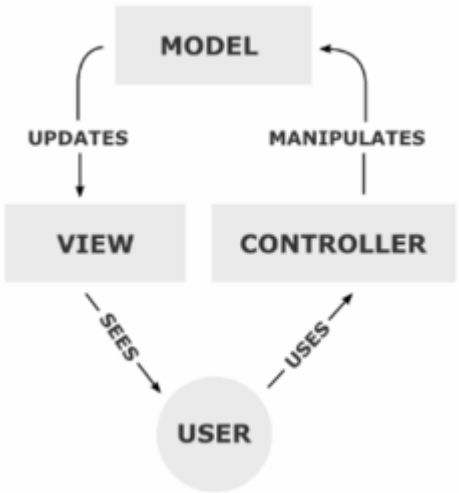

Рисунок 2.1 – Компоненти MVC

- Модель - це компонент, що відповідає за дані, а також визначає структуру програми. Наприклад, якщо розробник створює To-Do застосунок, код компонента model визначатиме список завдань і окремі завдання.

- Подання - це компонент, що відповідає за взаємодію з користувачем. Тобто код компонента view визначає зовнішній вигляд програми і способи його використання.

- Контролер - цей компонент, що відповідає за зв'язок між model та view. Код компонента controller визначає, як сайт реагує на дії користувача. По суті, це мозок MVC-застосунку.

Архітектура MVVM

Паттерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку застосунки від візуальної частини (подання). Цей патерн є архітектурним, тобто він задає загальну архітектуру програми.

Даний паттерн був представлений Джоном Госсманом (John Gossman) в 2005 році як модифікація шаблону Presentation Model та був спочатку націлений на розробку застосунків в WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в самих різних технологіях, в тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і подання (View), взаємодія яких відображена на рисунку 2.2.

Рисунок 2.2 – Компоненти архітектури MVVM

Model

Модель описує використовувані в застосунку дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити ніякої логіки,

пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомляти системі про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до подання, хоча знову ж пряма взаємодія між моделлю і представленням відсутня.

View

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з застосунком. Стосовно до WPF подання - це код в xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) в WPF може містити як інтерфейс в xaml, так і прив'язаний до нього код C #, проте код C # не повинен містити логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна. Вся ж основна логіка застосунку виноситься в компонент ViewModel.

Однак іноді в файлі пов'язаного коду може знаходитися певна логіка, яку важко реалізувати в рамках паттерна MVVM у ViewModel.

Подання не виконує жодних подій за деякими виключеннями, а виконує дії в основному за допомогою команд.

ViewModel

ViewModel або модель подання пов'язує модель і подання через механізм прив'язування даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна відображуваних даних в поданні, хоча безпосередньо модель і подання не пов'язані.

ViewModel також містить логіку з отримання даних з моделі, які потім передаються до подання. А потім ViewModel визначає логіку щодо оновлення даних в моделі.

Оскільки елементи подання, тобто візуальні компоненти типу кнопок, не використовують події, то подання взаємодіє з ViewModel за допомогою команд.

Наприклад, користувач хоче зберегти введені в текстове поле дані. Він натискає на кнопку і тим самим відправляє команду під ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель.

Підсумком застосування патерну MVVM є функціональний розподіл програми на три компонента, які простіше розробляти і тестувати, а також в подальшому модифікувати і підтримувати.

Архітектура VIPER

Архітектура VIPER складається з п’яти складових (рисунок 2.3), а саме:

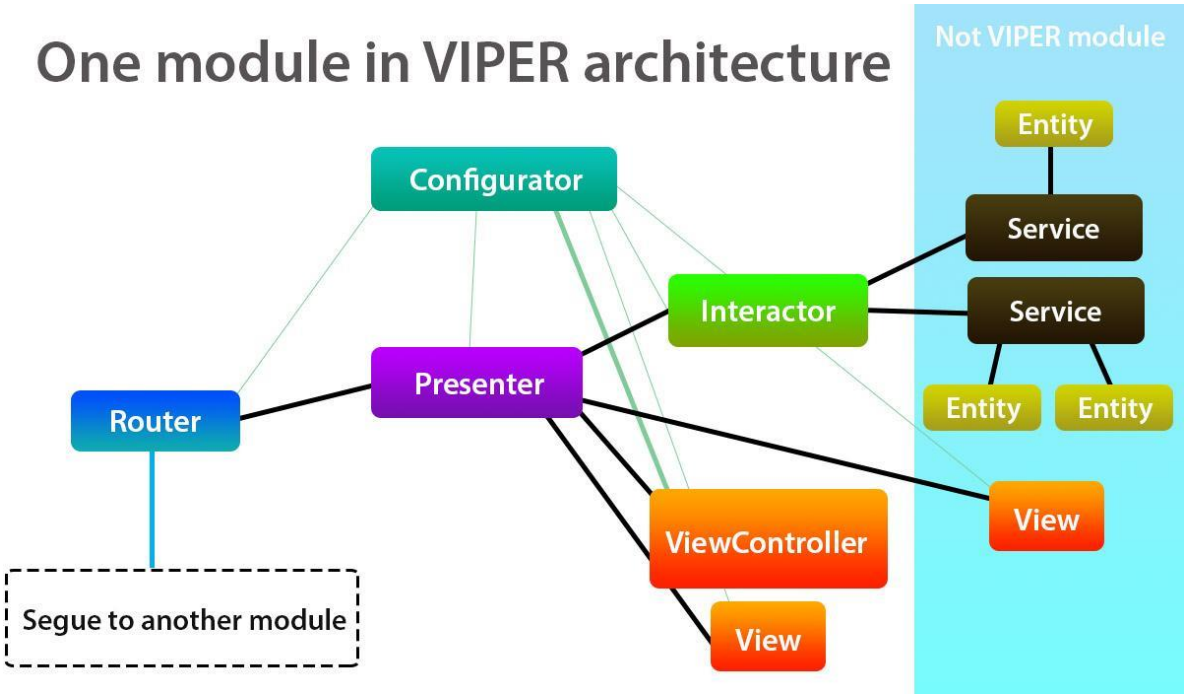

Рисунок 2.3 – Архітектура VIPER

View: відповідає за відображення даних на екрані й оповіщає Presenter про дії користувача. Він ніколи не запитує дані, тільки отримує їх від презентера.

Interactor: містить всю бізнес-логіку, необхідну для роботи поточного модуля.

Presenter: отримує від View інформацію про дії користувача і перетворює її в запити до Router'у, Interactor'у, а також отримує дані від Interactor'а, готує їх і відправляє View для відображення.

Entity: об'єкти моделі, що не містять нікакой бізнес-логіки.

Router: відповідає за навігацію між модулями.

Архітектура Clean Swift

Для початку потрібно розібрати основну термінологію архітектури. В Clean Swift застосунок складається зі сцен, тобто кожне вікно керування - це одна сцена. Основна взаємодія в сцені йде через послідовний цикл між компонентами ViewController -> Interactor -> Presenter. Це називається VIP цикл.

Мостом між компонентами виступає файл Models, який зберігає в собі дані, що передаються. Router, який відповідає за перехід і передачу даних між сценами, і Worker, який бере частину логіки Interactor'а на себе. Компоненти архітектури відображені на рисунку 2.4.

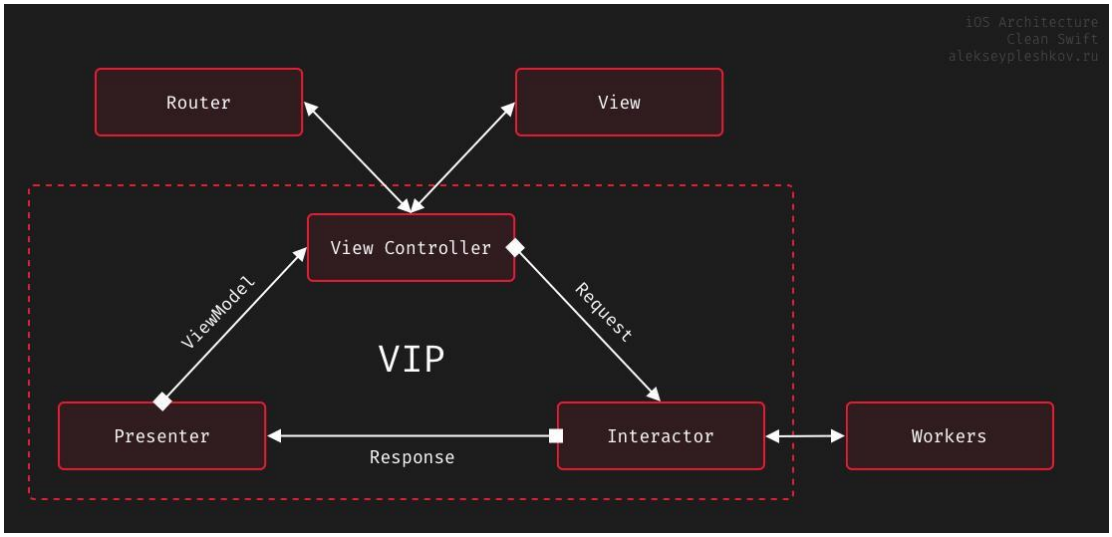

Рисунок 2.4 – Компоненти архітектури

View

Storyboard'и, XIB'и або UI елементи, написані через код.

ViewController

Відповідає тільки за конфігурацію і взаємодія з View. У контролері не повинно знаходитися ніякої бізнес логіки, взаємодії з мережею, обчислень і так далі.

Його завдання обробляти події з View, відображати або відправляти дані (без обробки і перевірок) в Interactor.

Interactor

Містить в собі бізнес логіку сцени.

Він працює з мережею, базою даних і модулями пристрою.

Interactor отримує запит з ViewController'a (з даними або порожній), обробляє його і, якщо це потрібно, передає нові дані в Presenter.

Presenter

Займається підготовкою даних для відображення.

Як приклад, додати маску на номер телефону або зробити першу букву в назві головної.

Обробляє дані, отримання з Interactor'a, після чого відправляє їх назад у ViewController.

Models

Набір структур для передачі даних між компонентами VIP циклу. Кожне коло циклу має в собі 3 види структур:

Request - Структура з даними (текст з TextField і т.д.) для передачі з ViewController'a в Interactor

Response - Структура з даними (завантаженими з мережі і т.д.) для передачі з Interactor в Presenter

ViewModel - Структура з обробленими даними (форматування тексту і т.д.) в Presenter'е для передачі назад у ViewController

Worker

Розвантажує Interactor, забираючи на себе частину бізнес логіки застосунка, якщо Interactor стрімко розростається.

Так само можна створювати загальні для всіх сцен Worker'и, якщо їх функціонал використовується в декількох сценах.

Як приклад, в Worker можна виносити логіку роботи з мережею або базою даних.

Router

У Router виноситься вся логіка, що відповідає за переходи і передачу даних між сценами.

2.2 MVC як основне архітектурне рішення для побудови застосунку

Для розробки була обрана концепція MVC.

Переваги концепції MVC

1. Єдина концепція системи. Безсумнівним плюсом MVC є єдина глобальна архітектура програми. Навіть в складних системах, розробники (як ті, які розробляли систему, так і ті, що знову приєдналися) можуть легко орієнтуватися в програмних блоках. Наприклад, якщо виникла помилка в логіці обробки даних, розробник відразу відкидає 2-блоку програми (controller і view) і займається дослідженням 3-го (model). Слідчо, в контексті MVC сильно спрощується локалізація проблем.

2. Спрощено механізм налагодження програми, тобто весь механізм візуалізації тепер сконцентрований в одному програмному блоці, спростилися механізми опціонального виведення графічних елементів.

Сама очевидна перевага, яку ми отримуємо від використання концепції MVC - це чіткий поділ логіки подання (інтерфейсу користувача) й логіки програми.

Підтримка різних типів користувачів, які використовують різні типи пристроїв є спільною проблемою наших днів. Наданий інтерфейс повинен відрізнятися, якщо запит приходить з персонального комп'ютера або з мобільного телефону. Модель повертає однакові дані, єдина відмінність полягає в тому, що контролер вибирає різні види для виведення даних.

Крім ізолювання видів від логіки застосунки, концепція MVC істотно зменшує складність великих застосунків. Код виходить набагато більш структурованим, і, тим самим, полегшується підтримка, тестування і повторне використання рішень.

Ідея, яка лежить в основі конструкційного шаблону MVC, дуже проста: потрібно чітко розділяти відповідальність за різне функціонування в наших програмах:

Застосунок розподіляється на три основних компоненти, кожен з яких відповідає за різні завдання.

2.3 Аналіз KeyChain як основного безпечного середовища даних в iOS

Згідно з документацією Apple, Keychain являє собою безпечний контейнер для зберігання конфіденційної інформації (імен користувачів, паролів, мережевих паролів, аутентифікаційних токенов різних застосунків). Засоби операційної системи Apple зберігають в Keychain паролі для Wifi, облікові записи Vpn тощо. Ці дані зашифровані і зберігаються в базі даних sqlite. Розробники зазвичай використовують цю можливість для зберігання облікових записів, замість використання NSUserDefaults, plist-файлів тощо. Причиною тому може бути бажання розробника зберегти інформацію про аутентифікацію, щоб користувачеві при повторному відкритті програми не доводилося щоразу вводити облікові дані. Дані keychain для кожної програми зберігаються поза ізольованим середовищем (пісочниці).

Також можливе спільне використання між застосунками даних keychain через групи доступу (keychain access groups). Ця група повинна бути визначена під час збереження інформації в keychain. Найкращий спосіб збереження інформації в keychain - використовувати клас KeychainItemWrapper. На сайті розробників Apple можна знайти тестовий проект. Для початку необхідно створити екземпляр класу.

```
KeychainItemWrapper * wrapper = [[KeychainItemWrapper alloc] initWithIdentifier: @ "Password" accessGroup: nil];
```

Ідентифікатор допоможе в майбутньому в отриманні інформації з keychain. Якщо ви хочете спільно використовувати інформацію з keychain з іншими застосунками, то необхідно визначити групу доступу. Застосунки з однієї і тієї групою доступу можуть отримувати доступ до однієї і тієї ж інформації з keychain.

```
KeychainItemWrapper * wrapper = [[KeychainItemWrapper alloc] initWithIdentifier: @ "Account Number"
    accessGroup: @ "YOUR_APP_ID_HERE.com.yourcompany.GenericKeychainSuite"];
```

Для збереження інформації в keychain використовуйте метод setObject: forKey :. У цьому випадку (id) kSecAttrAccount є зумовлений ключ, який ми можемо використовувати для оголошення імені облікового запису, для якої ми зберігаємо інформацію. Ключ kSecClass - визначає вид інформації, що зберігається (в нашому випадку це груповий пароль (generic password)). Ключ kSecValueData можна використовувати для зберігання будь-яких даних (в нашому випадку це пароль).

```
[KeychainItemWrapper setObject: kSecClassGenericPassword forKey: (id) kSecClass];
    [Wrapper setObject: @ "username" forKey: (id) kSecAttrAccount];
    [KeychainItemWrapper setObject: @ "password" forKey: (id) kSecValueData];
    [Wrapper setObject: (id) kSecAttrAccessibleAlwaysThisDeviceOnly forKey: (id) kSecAttrAccessible];
```

Мінлива kSecAttrAccessible використовується для регулювання доступу до даних keychain. Слід бути обережним при використанні цієї змінної. Як сказано

в документації Apple, змінна kSecAttrAccessible може приймати шість можливих значень.

Висновки до розділу 2

У другому розділі йдеться про архітектуру застосунку. Він також описує середовище та системи, необхідні для створення власного застосунку.

РОЗДІЛ 3

ПРОЕКТУВАННЯ, РОЗРОБЛЕННЯ, ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ВЛАСНОГО МОБІЛЬНОГО ЗАСТОСУНКУ ПОШУКУ КУЛЬТУРНИХ ПАМ’ЯТОК

3.1 Технічне завдання та постановка задачі мобільний застосунок з пошуку архітектурних пам'яток

Головна мета дипломного проекту є швидкий пошук культурних пам'яток України на основі мобільного застосунку..

Головне призначення мобільного додатку – надання інформації про культурні пам'ятки України. Мобільний додаток повинен надати можливість користувачеві швидко та легко здійснювати пошук архітектурних пам'яток України.

Вимоги до функціональних характеристик.

Мобільний додаток з пошуку культурних пам'яток повинен складатися з карти та розподіленого за областями списку культурних пам'яток України.

Мобільний додаток з пошуку культурних пам'яток повинен забезпечувати можливість виконання наведених нижче функцій:

- перегляд інформації про культурну пам'ятку;
- пошук культурної пам'ятки за геолокацією;
- пошук пам'ятки на карті.

Вимоги до інтерфейсу користувача.

Користувач повинен мати можливість:

- завантажити додаток з магазину;
- чітко розуміти, як необхідно шукати культурні пам'ятки;
- надсилати запити для отримання інформації про пам'ятку та оперативно отримувати результати цих запитів.

Стадії і етапи розробки.

Розробка має бути проведена в чотири стадії:

- розробка технічного завдання;
- розробка робочого проекту;
- тестування
- впровадження.

3.2 Проектування застосунку та його інформаційне забезпечення

Формування бази даних

Оскільки база даних є основним етапом розробки застосунку, то першим етапом реалізації було стало формування бази даних культурних пам'яток.

Для формування бази даних були знайдені і проаналізовані культурні пам'ятки областей України, потім відібрано і додано у застосунок найкращі 3-5 пам'яток кожної області України на основі відгуків людей і експертів.

Підтримка бази даних

Оскільки при туризмі, інколи, користувач має проблеми з інтернетом, то було вирішено використовувати базу даних на пристрої. Для мобільних пристроїв на базі системи iOS використано Core Data.

Core Data — це фреймворк для збереження даних, наданий Apple в операційних системах macOS та iOS. Він був представлений в Mac OS X 10.4 Tiger і iOS з iPhone SDK 3.0. Це дозволяє даним, організованим реляційною моделлю сутність-атрибут, бути серіалізовані в XML, двійкові або SQLite сховища. Даними можна маніпулювати за допомогою об’єктів вищого рівня, що представляють сутності та їх зв’язки. Core Data керує серійною версією, забезпечуючи керування життєвим циклом об’єкта та графом об’єктів, включаючи збереження. Core Data взаємодіє безпосередньо з SQLite, ізолюючи розробника від базового SQL.

Мова програмування

Swift - це один з найбільш актуальних способів писати програми для телефонів, для десктопних комп'ютерів, серверів. Swift - безпечна, швидка і інтерактивна мову програмування. Swift увібрала в собі кращі ідеї сучасних мов з мудрістю інженерної культури Apple. Компілятор оптимізований для продуктивності, а мова оптимізована для розробки, без компромісів з однієї чи іншої сторони.

Swift виключає великий пласт поширених програмних помилок за допомогою застосування сучасних програмних паттернів:

Змінні завжди ініціалізовані до того, як будуть використані.

Індекси масивів завжди перевіряються на out-of-bounds помилки.

Цілі числа перевіряються на переповнення.

Опціональність гарантує, що значення nil будуть явно оброблені.

Автоматичне управління пам'яттю

Обробка помилок дозволяє здійснювати контрольоване відновлення від непередбачених помилок.

Код на Swift скомпільований і оптимізований, щоб отримувати максимальну віддачу від сучасного обладнання. Синтаксис стандартної бібліотеки спроектований грунтуючись на керівництві, що є найочевиднішим і простішим способом написання коду.

Swift поєднує висновок типів і патерн-матчінг з сучасним простим синтаксисом, дозволяючи складним ідеям бути вираженим просто і коротко. І як результат не тільки стає простіше писати код, але і читати його і підтримувати так само стає просто.

Swift вже має за плечима роки розвитку, і він продовжує розвиватися, включаючи в себе все нові і нові можливості.

Саме тому, для реалізації мобільного застосунку було обрано мову програмування Swift.

3.3 Розробка власного продукту

Розроблено демонстраційний мобільний застосунок на платформі iOS, який повністю відповідає запропонованій у цій роботі технології розробки додатків.

При першому запуску необхідно вказати ідентифікатор програми, розробленої в рамках веб-конструктора, структура та вміст якої буде синхронізовано мобільним клієнтом. Модель даних клієнтського рішення повністю збігається з моделлю даних серверної частини, що згодом дозволяє модифікувати клієнтську програму при синхронізації даних із сервером.

Розробка будь-якого застосунку починається з логотипу. Для нашого застосунку було розроблено наступний логотип (рис. 3.1).

Рисунок 3.1 – Логотип застосунку

Наступним кроком була розробка меню програми. Спочатку було розроблено розбиття за областями. Наприклад, Тернопільська область (рис. 3.2) чи Київ (рис. 3.3).

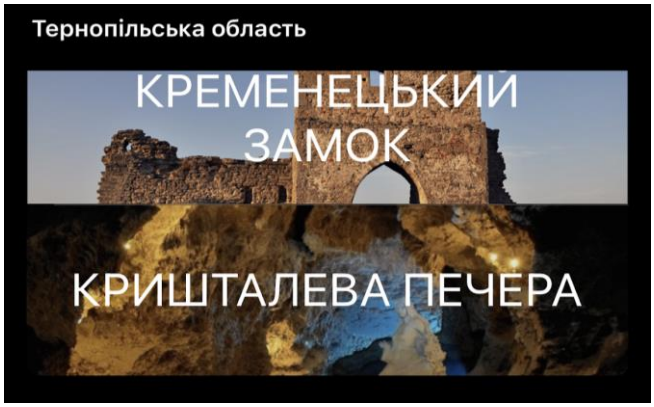

Рисунок 3.2 – Тернопільська область

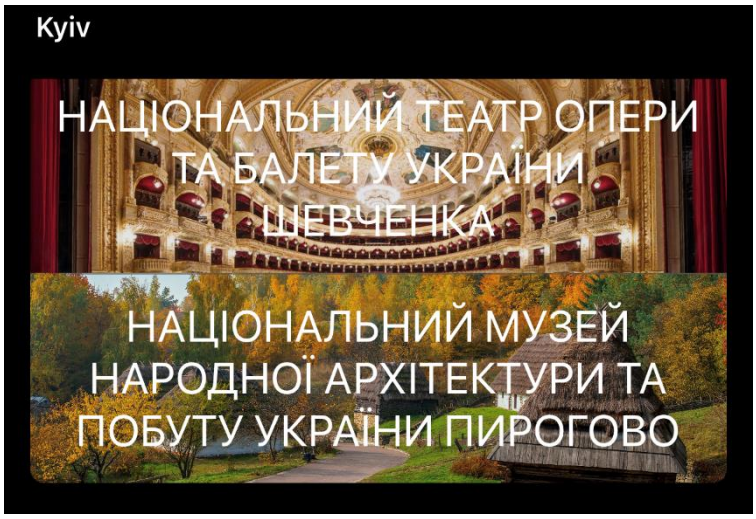

Рисунок 3.3 – Київ

Синхронізація відбувається у фоновому режимі при запуску програми, а також при отриманні сповіщення про зміну даних.

Повідомлення надсилаються всім встановленим мобільним застосункам, після отримання яких вони самостійно завантажують поточні налаштування конфігурації з сервера.

Далі було розроблено список культурних пам’яток в заданій області (рис. 3.4)

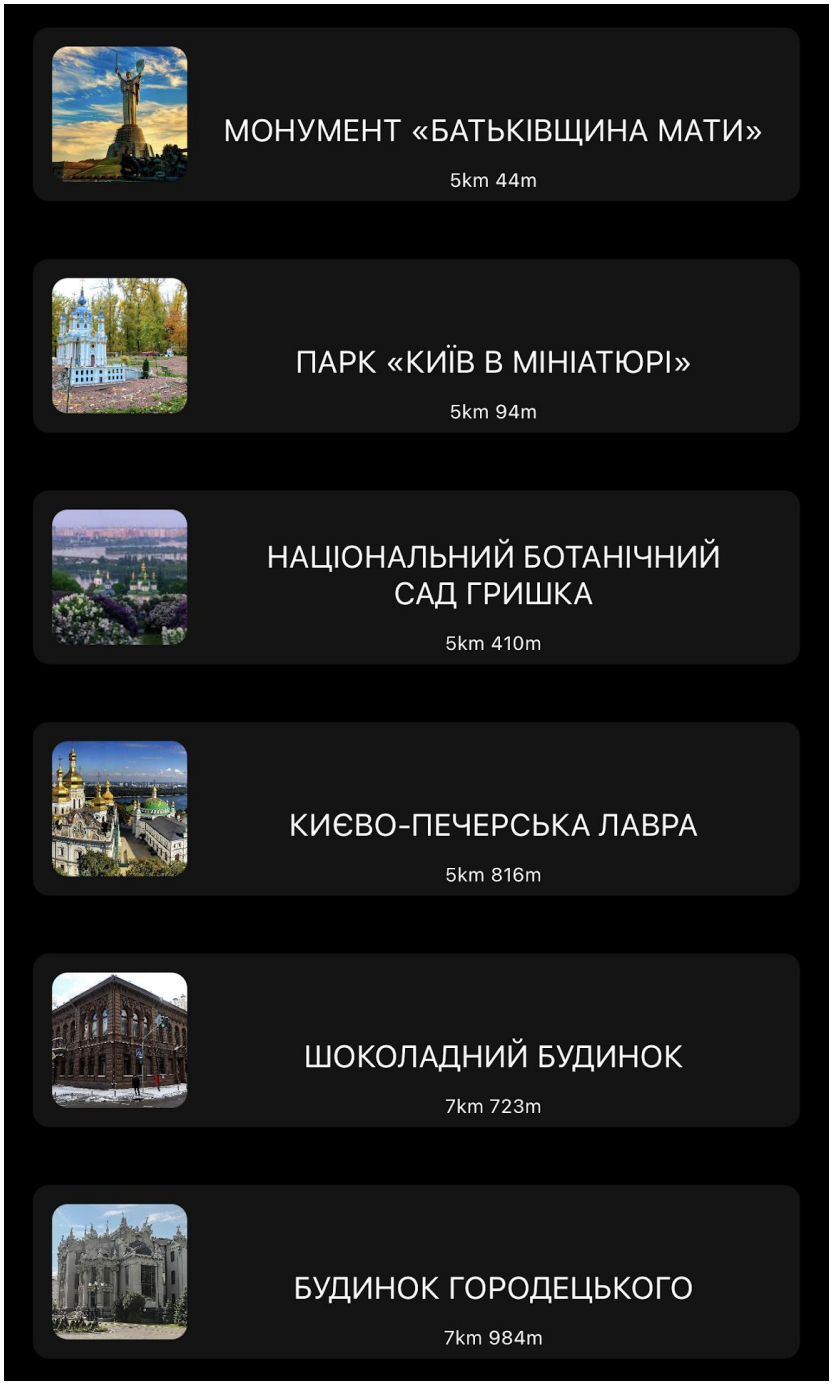

Рисунок 3.4 – перелік культурних пам’яток

Останнім кроком при розробці було додавання опису культурної пам’ятки (рис. 3.5)

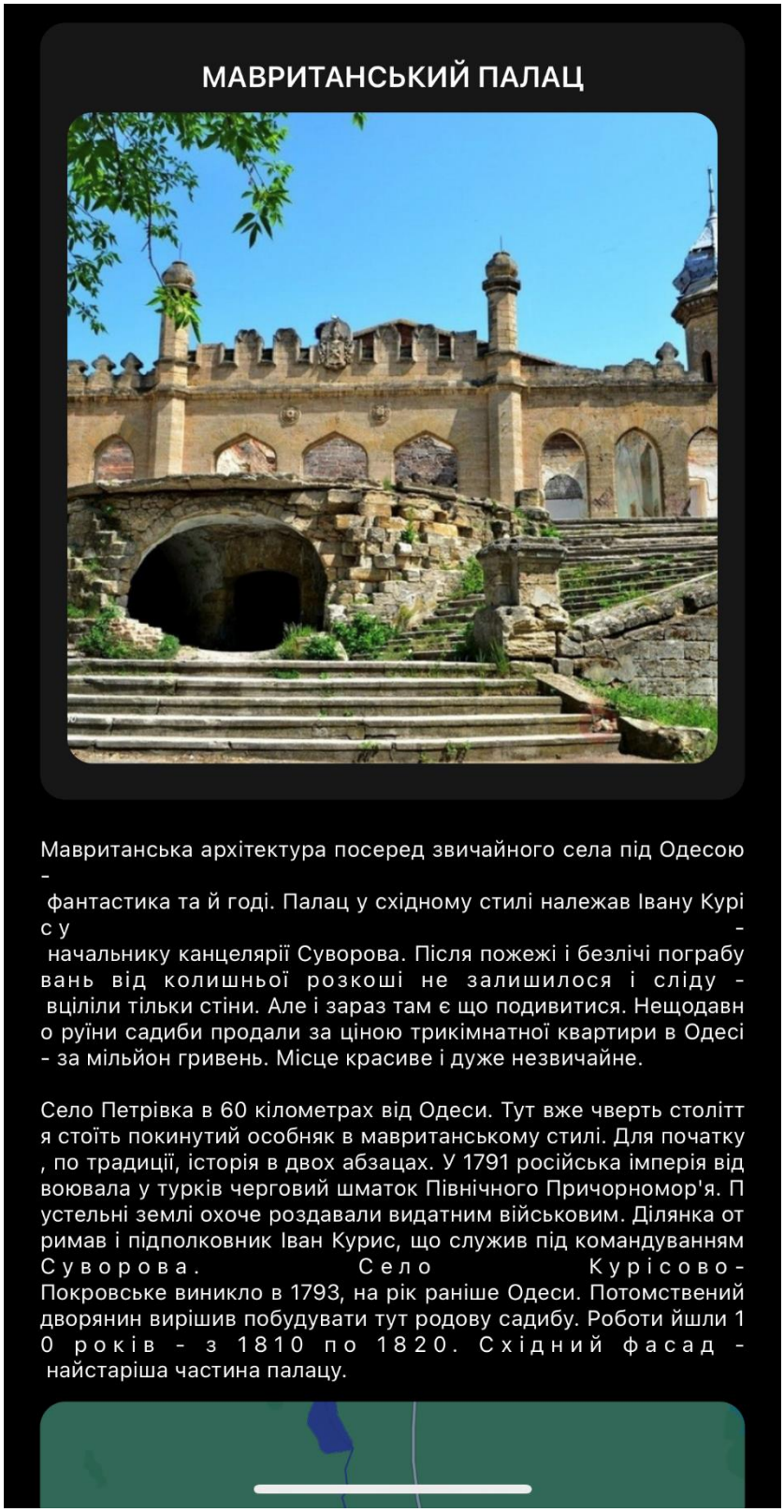

Рисунок 3.5 – Опис культурної пам’ятки

3.4 Опис функціоналу

Запустивши додаток перед вами з’явиться наступне зображення (рис. 3.6)

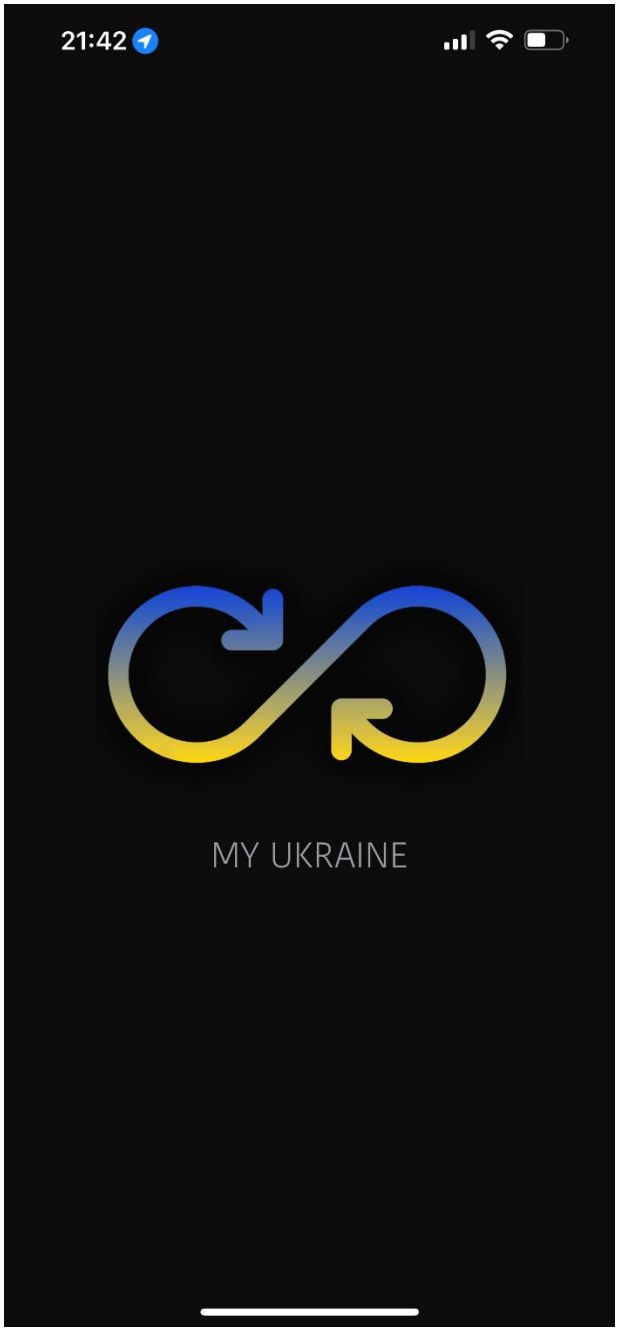

Рисунок 3.6 Стартове вікно

Перейшовши до меню застосунку ми можемо вибрати область України яка нас цікавить (рис. 3.7)

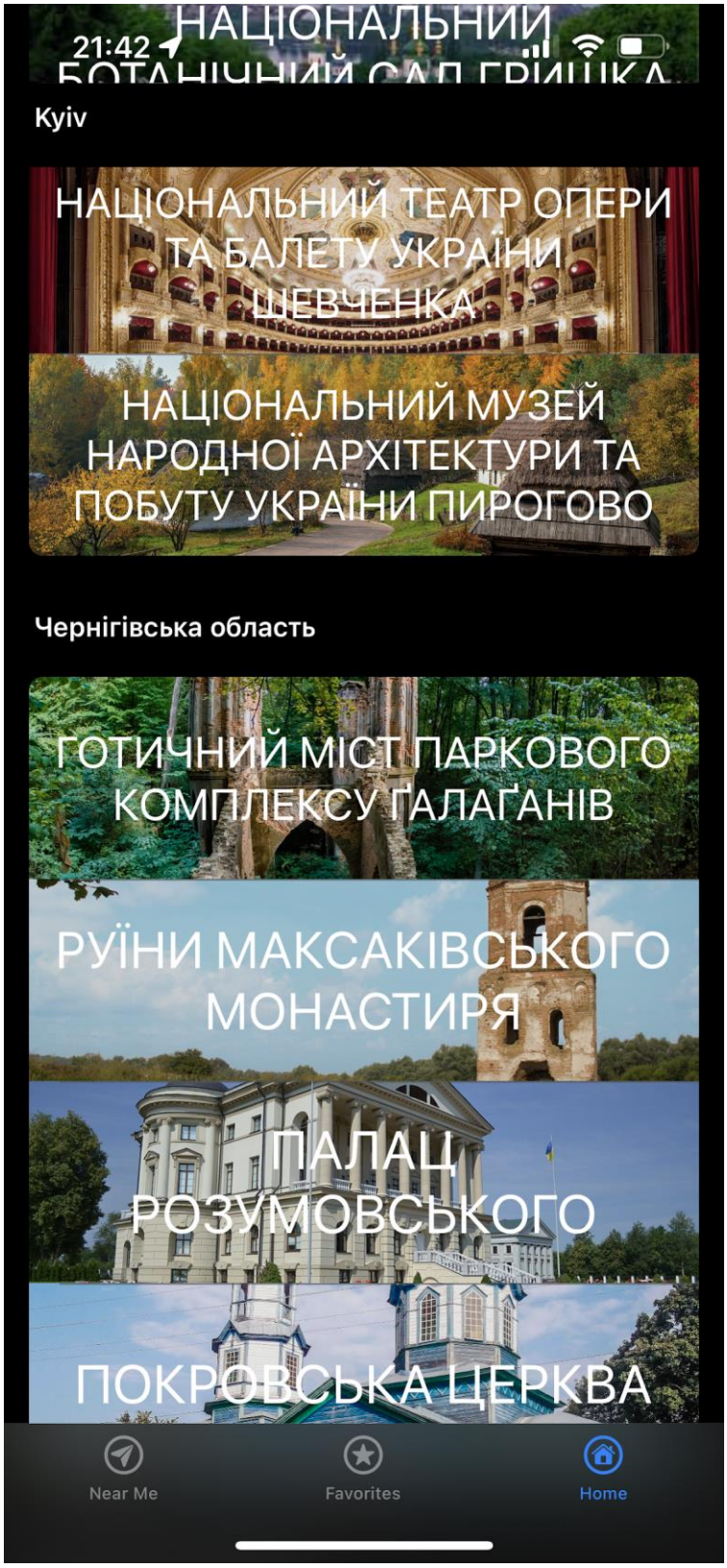

Рисунок 3.7 – Області України

Пам'ятки культури України згідно поділу областей.

Перейшовши до обраної області можна вибрати культурну пам’ятку, яка знаходиться в обраній області (рис. 3.8)

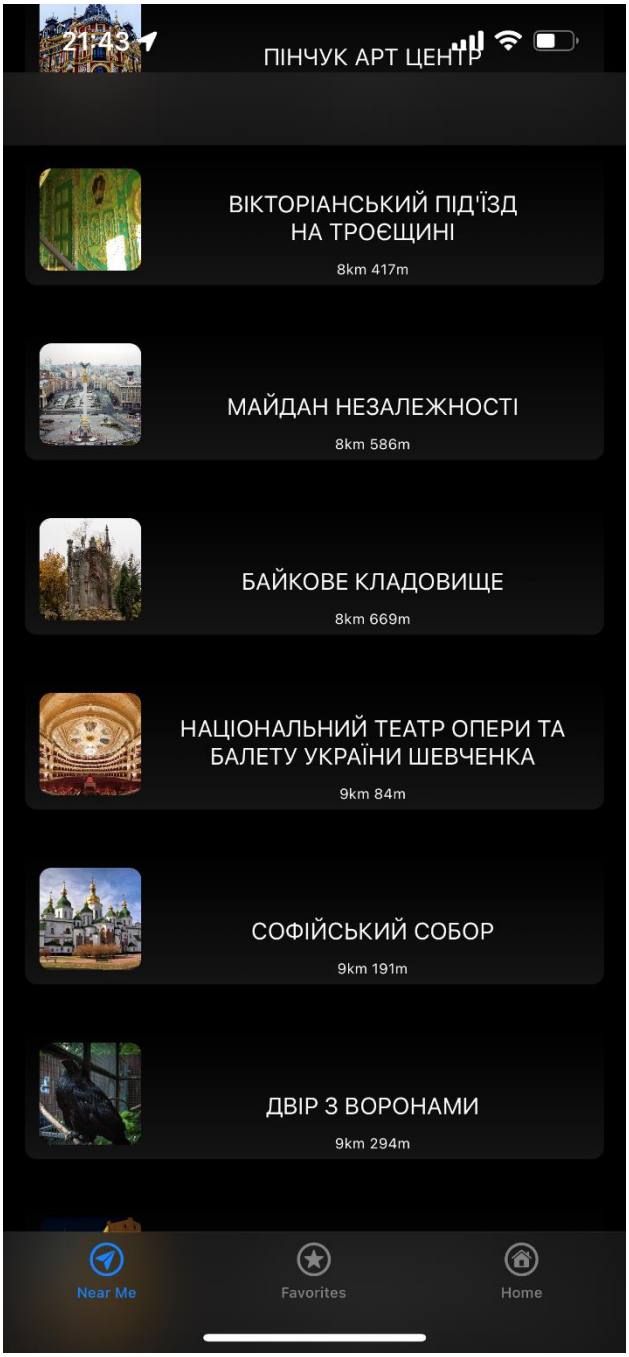

Рисунок 3.8 – Вибір культурної пам’ятки

Перейшовши до обраної культурної пам’ятки ви зможете ознайомитися з нею (рис. 3.9)

Рисунок 3.9 Опис культурної пам’ятки

3.5 Тестування

Під час тестування усі виникаючі помилки одразу виправлялися. Нижче наведено декілька прикладів тестування (рис. 3.10-3.14).

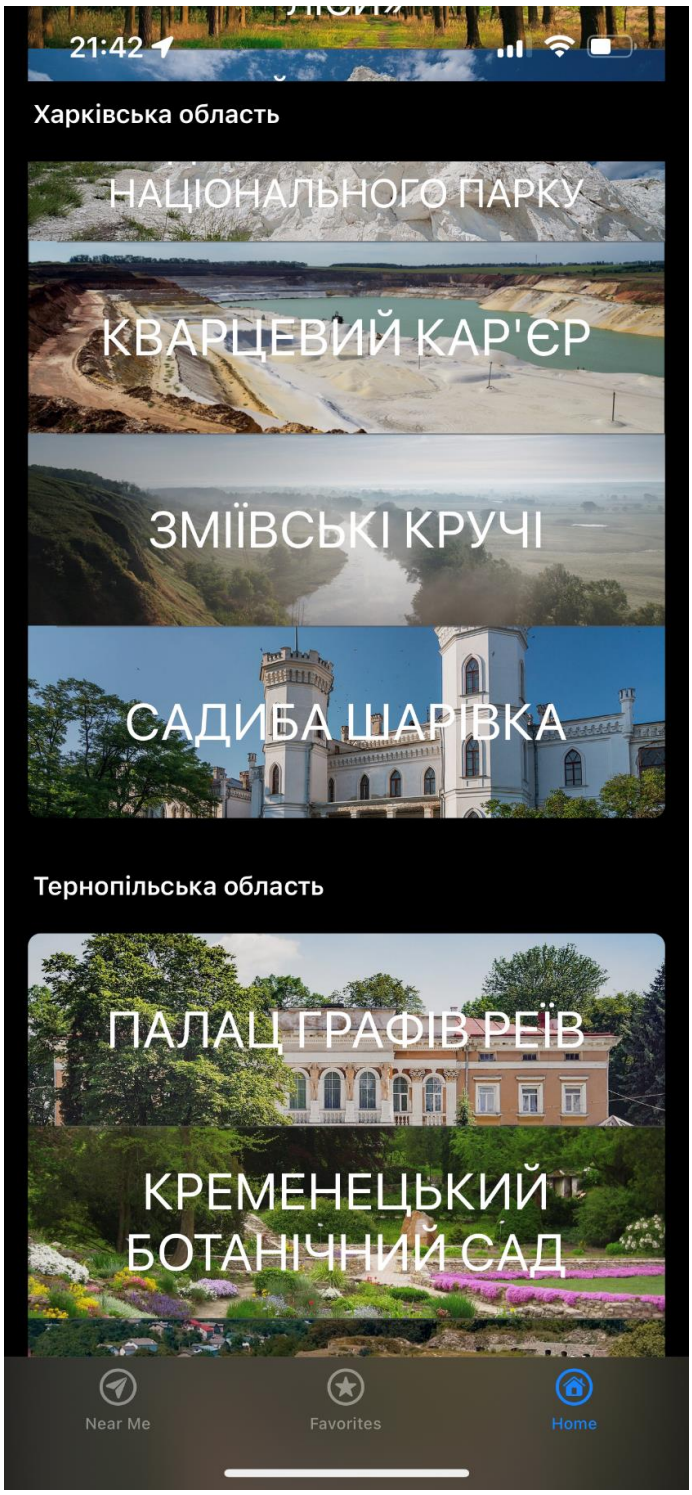

Рисунок 3.10 – Області України

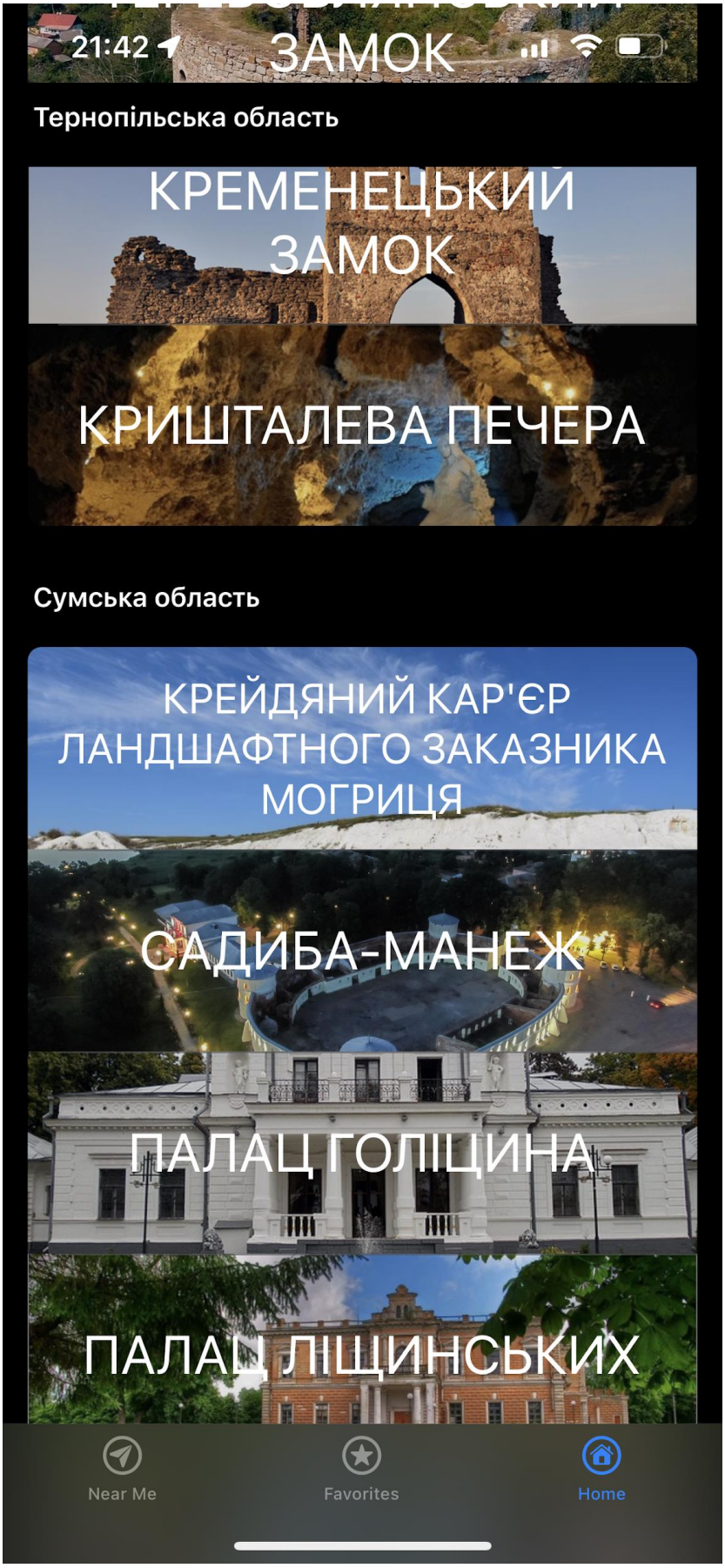

Рисунок 3.11 –Області України

Рисунок 3.12 – Опис культурної пам’ятки

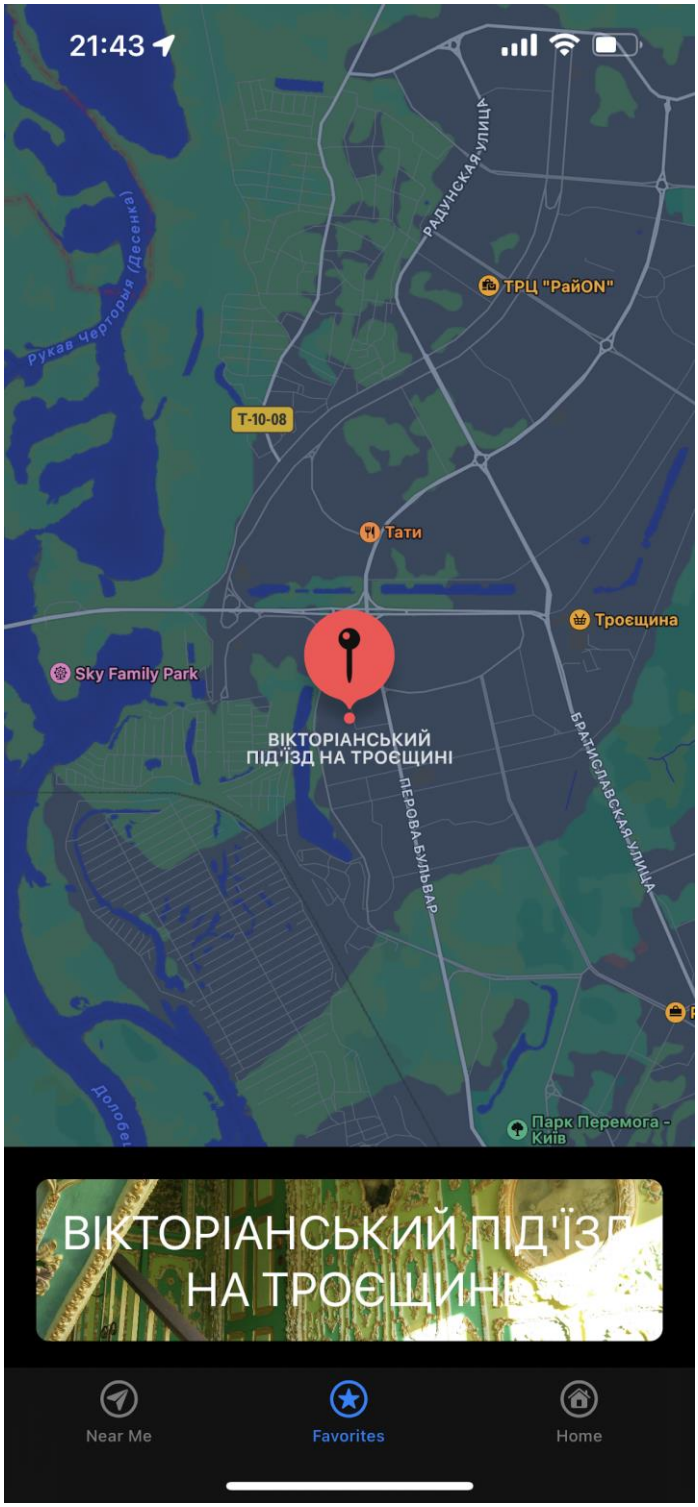

Рисунок 3.13 – Розташування культурної пам’ятки

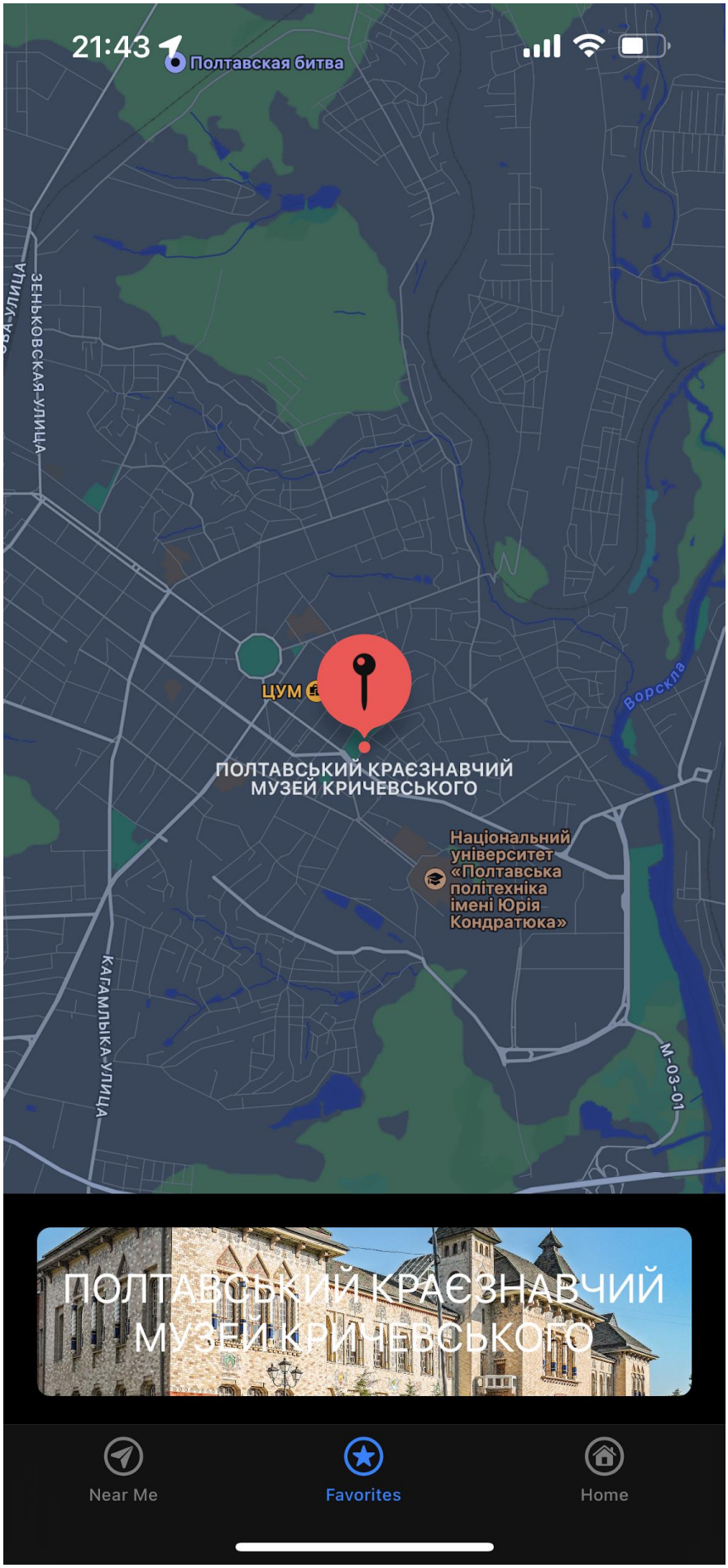

Рисунок 3.14 – Розташування культурної пам’ятки

Висновки до розділу 3

У цьому розділі розповідається про розроблену структуру застосунку. Описують основні форми інтерфейсу застосунку. Описується його функціонал та його тестування.

ВИСНОВКИ

У цій роботі ми пропонуємо та реалізуємо мобільний застосунок з пошуку культурних пам’яток України.

Унікальність цієї роботи полягає в тому, що створений контент-орієнтований мобільний додаток: структура і зміст контенту повністю налаштовуються у створеному додатку.

Обраний фундаментальний архітектурний зразок є реалізацією підходу Model-View-Controller до проектування та розробки мобільних додатків.

Аналіз варіантів реалізації такого підходу на інших мобільних платформах показує можливу масштабованість рішення. Важливою є можливість зміни опису моделі, а також надані варіанти міграції. Реалізований метод встановлює високий рівень гнучкості моделі, що важливо при розробці динамічних систем такого типу.

Розроблений демонстраційний мобільний клієнтський додаток доводить ефективність запропонованої архітектури.

Подальший розвиток технології передбачає створення клієнтських додатків для основних мобільних платформ (Android, Windows Phone).

Завдання полегшується тим, що для функціонування системи достатньо розробити лише одну програму, яка б реалізувала всі аспекти запропонованої технології. Завдяки динамічності рішення такий додаток може самостійно адаптуватися до моделі даних і структури вмісту, які згодом можуть змінюватися без необхідності перекомпіляції або збірки. Він також передбачає розробку вбудованого компонента (SDK), який сторонні розробники мобільних додатків можуть додати до своїх проектів для швидкого створення прототипів динамічних модулів.

СПИСОК ЛІТЕРАТУРИ

1. Культурна спадщина в контексті «Зводу пам’яток історії та культури України» / Кот С.І. (відповідальний редактор), Денисенко Г.Г., Івакін Г.Ю., Катаргіна Т.І., Ковпаненко Н.Г., Скрипник П.І., Тимофієнко В.І., Титова О.М., Федорова Л.Д. К.: Інститут історії України, 2015. 486 с.
2. История веб-дизайна, 2017. — RedKrab — URL: https://webevolution.ru/blog/sajti/istoriya-veb-dizajna/
3. История веб-дизайна: от каменного века до эпохи современных технологий, 2016. — WEBFORMYSELF — URL: https://webformyself.com/istoriya-veb-dizajna-ot-kamennogo-veka-do-epoxi-sovremennyx-texnologij/
4. Веб 2.0, 2021. — Wikipedia — URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1_2.0
5. Что такое веб-дизайн и с чем его едят?, 2016. — WEBFORMYSELF — URL: https://webformyself.com/chto-takoe-veb-dizajn-i-s-chem-ego-edyat/
6. Что такое веб-дизайн и чем занимается веб-дизайнер?, 2021. — E11EVEN — URL: https://e11evenmarketing.com/blog/web-design/chto-takoe-veb-dizajn-i-chem-zanimaetsya-veb-dizajner/
7. Вёрстка веб-страниц, 2021. — Wikipedia — URL: https://ru.wikipedia.org/wiki/%D0%92%D1%91%D1%80%D1%81%D1%82%D0%BA%D0%B0_%D0%B2%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86
8. Google Класс, 2021. — Wikipedia — URL: https://ru.wikipedia.org/wiki/Google_%D0%9A%D0%BB%D0%B0%D1%81%D1%81
9. Microsoft Teams, 2021. — Wikipedia — URL: https://ru.wikipedia.org/wiki/Microsoft_Teams
10. SunRav BookOffice, 2021. — SunRav — URL: https://sunrav.ru/bookoffice.html

11. WEB-дизайн як навчальна дисципліна, 2021. — KURSOVIKS — URL: https://ua.kursoviks.com.ua/kompyuterni/web-dizayn
1. Педагогічний дизайн у сучасному освітньому процесі, 2015. — eprints — URL: http://eprints.zu.edu.ua/18617/1/17.pdf
2. Професійна підготовка студентів соціально-педагогічної сфери – освітня складова суспільного розвитку [Електронний ресурс] : матеріали II Міжнародної науково-практичної конференції (дистанційної) (Київ, 1 квітня 2014 року) / Національний педагогічний університет імені М. П. Драгоманова, Бердянський державний педагогічний університет, Хмельницька гуманітарно-педагогічна академія, Українсько-американський гуманітарний інститут «Вісконсинський міжнародний університет (США) в Україні» ; за заг. ред. І. М. Ковчиної. - Київ : [б. в.], 2014. - 120 с.
3. Педагогіка вищої школи [Електронний ресурс] : навч. посібник / за ред. З. Н. Курлянд. - К. : Знання, 2005. - 399 с.
4. Теорія і методи соціальної роботи [Електронний ресурс] : навчальний посібник / М. П. Лукашевич, І. І. Мигович ; Міжрегіональна академія управління персоналом. - 2-ге вид., допов. і випр.. - К. : [б. в.], 2003. - 168 с.
5. Педагогіка у запитаннях і відповідях [Електронний ресурс] : навч. посіб. / А. І. Кузьмінський, В. Л. Омеляненко. - К. : Знання, 2006. - 311 с.. - (Навчально-методичний комплекс з педагогіки)
6. Інформаційно-комунікаційні технології у післядипломній освіті [Електронний ресурс] / В. С. Назаренко, В. В. Кузьменко. - Херсон : Херсонська академія неперервної освіти, 2013. - 171 с.
7. A history of the university in Europe [Electronic resource] : in 4 vol. / ed. Walter Ruegg. - Cambridge : Cambridge university press, 2003
8. Проблеми та перспективи розвитку освіти, науки і техніки в Україні та світі [Електронний ресурс] : зб. праць за матеріалами Всеукраїнської науково-практичної конференції, 20-21 травня 2016 р. / Інститут історії України НАН України ; уклад. С. М. Ховрич. - Київ : [б. в.], 2016. - 139 с.

ДОДАТОК А

Список культурних пам'яток у розрізі областей

```
import UIKit
import SafariServices

class ViewController: UIViewController {

    private var tableView: UITableView = .init()

    internal var tableViewData: [MainTown: [HomeCellModel]] = .init()

    override func viewDidLoad() {
        super.viewDidLoad()

        setup()
    }
}

private extension ViewController {

    func setup() {
        setupData()
        setupView()
        setupTableView()
        setupLayout()
    }

    func setupView() {
```

```
        view.backgroundColor = .black
    }

    func setupTableView() {

        tableView.backgroundColor = .clear
        tableView.delegate = self
        tableView.dataSource = self
        tableView.separatorStyle = .none
        tableView.estimatedRowHeight = 0
        tableView.estimatedSectionFooterHeight = 0
        tableView.estimatedSectionHeaderHeight = 0
        tableView.sectionFooterHeight = 0.0
        let inset: UIEdgeInsets = .init(top: 0.0, left: 0.0, bottom: 0.0, right: 0.0)
        tableView.contentInset = inset
        tableView.scrollIndicatorInsets = inset
        tableView.showsVerticalScrollIndicator = false
        tableView.isScrollEnabled = true
        tableView.tableFooterView = UIView(frame: CGRect(x: 0, y: 0, width:
tableView.frame.size.width, height: 1))

        tableView.register(
            HomeTabelViewCell.self,
            forCellReuseIdentifier: HomeTabelViewCell.reuseIdentifier
        )
    }

    func setupLayout() {
        view.addSubview(tableView)
```

```
        tableView.snp.makeConstraints { make in
            make.edges.equalToSuperview()
        }
    }
}

//MARK: - UITableViewDelegate

extension ViewController: UITableViewDelegate {

    func tableView(
        _ tableView: UITableView,
        heightForHeaderInSection section: Int
    ) -> CGFloat {
        50
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath) {
        let cell = tableView.cellForRow(at: indexPath)

        if let cell = cell as? HomeTabelViewCell {

            let vc: MoreViewController = .init()
            vc.modalPresentationStyle = .fullScreen

            vc.model = cell.model

            self.present(vc, animated: true)
        }
```

```
    }

    func tableView(
        _ tableView: UITableView,
        heightForRowAt indexPath: IndexPath
    ) -> CGFloat {

        return 120.0
    }
}

//MARK: - UITableViewDataSource

extension ViewController: UITableViewDataSource {

    func numberOfSections(
        in tableView: UITableView
    ) -> Int {
        tableViewData.count
    }

    func tableView(
        _ tableView: UITableView,
        numberOfRowsInSection section: Int
    ) -> Int {
        guard let headerType = section.headerType else { return 0 }

        return tableViewData[headerType]?.count ?? 0
    }
```

```
func tableView(
    _ tableView: UITableView,
    cellForRowAt indexPath: IndexPath
) -> UITableViewCell {

    if let cell = tableView.dequeueReusableCell(
        withIdentifier: HomeTabelViewCell.reuseIdentifier,
        for: indexPath
    ) as? HomeTabelViewCell,
    let headerType = indexPath.section.headerType,
    let models = tableViewData[headerType] {

        cell.updateCell(
            model: models[indexPath.row]
        )

        cell.accessibilityIdentifier = models[indexPath.row].id.rawValue

        return cell
    }

    return .init()
}

func tableView(
    _ tableView: UITableView,
    viewForHeaderInSection section: Int
) -> UIView? {
```

```
        let view: UIView = .init()
        let titleLabel: UILabel = .init()

        view.backgroundColor = .black

        titleLabel.font = UIFont.boldSystemFont(ofSize: 16.0)
        titleLabel.textColor = .white

        view.addSubview(titleLabel)

        titleLabel.snp.makeConstraints { make in
            make.leading.equalToSuperview().inset(18.0)
            make.bottom.equalToSuperview().inset(20.0)
        }

        titleLabel.text = section.headerType?.title

        return view
    }
}
```