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: «Веб-сервіс культурних заходів міста»

Виконав
(Підпис)
Доценко Микита Данилович
(прізвище, ім'я, по батькові)

Керівник Гарко Ірина Ігорівна
(прізвище, ім'я, по батькові)

(Резолюція «До захисту»

Попередній захист:

_____
_
(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 - 96 %

Київ – 2022

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	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	
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	21.10.2021	
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	21.12.2022	
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	31.01.2022	
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	30.03.2022	
9.	Подання роботи у першому варіанті	28.04.2022	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	23.05.2022	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	27.05.2022	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	10.06.2022	
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022 23.06.2022 24.06.2022	

Здобувач вищої освіти

Керівник

ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Складові частини кваліфікаційної роботи	Обсяг, арк.
Титульний аркуш	1
Календарний план кваліфікаційної роботи	1
Відомість кваліфікаційної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	0
Вступ	3
Розділ 1	7
Розділ 2	12
Розділ 3	21
Висновки	2
Список використаних джерел	2
Додатки	48

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Доценко М.Д.			Відомість кваліфікаційної роботи	Лист	Листів
Керівн.	Гарко І.І.					
Н/контр.	Базилюк А.М.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Кваліфікаційна робота бакалавра містить: 103 сторінки, 36 рисунків, 20 використаних джерел. Метою роботи є підвищення ефективності пошуку та відслідковування культурних заходів міста з використанням веб-сервісу. Об’єктом дослідження кваліфікаційної роботи бакалавра є процеси пошуку та відслідковування культурних заходів. Предметом дослідження кваліфікаційної роботи бакалавра є програмно-технічні, організаційні засади, принципи, підходи побудови веб-застосунку культурних заходів міста. Методи дослідження — аналітичний (допомагає в проведенні досліджень щодо вибору оптимальних інструментів і підходів до проектування сервісу, за рахунок наявних даних), порівняльний (використовується при вивченні існуючих аналогів веб-сервісів та їх клієнтів і технічних засобів, і полягає у виборі кращого варіанту за рахунок порівняльного аналізу даних), абстрагування (використовується для зосередження на головних аспектах існуючих аналогів веб-сервісів з перегляду культурних заходів міста, для виділення і знаходження існуючих проблем, що можуть бути вирішені), моделювання (використовується для формування висновків щодо об’єкту дослідження на основі вивчення аналогів існуючих сервісів), прогнозування (використовується для формування припущення щодо об’єкту дослідження). У результаті виконання роботи було розроблено веб-сервіс для перегляду культурних заходів міста, а також зручний і сучасний мобільний клієнт для нього, що має на меті допомогти користувачеві знаходити цікаві йому культурні заходи, переглядати і додавати цікаві йому культурні заходи у обране. Програмна реалізація веб-сервісу виконана на ASP.NET Core Web API, програмна реалізація мобільного клієнту виконана на Xamarin.Forms, використано мову програмування C#.

Ключові слова: мобільний застосунок, веб-сервіс, культурний захід, патерни програмування, розробка застосунків на платформі ASP.NET Core Web API/ Xamarin.Forms.

ABSTRACT

The bachelor's thesis contains: 103 pages, 36 drawings, 20 sources used. The aim of the work is to increase the efficiency of searching and tracking cultural events of the city using a web service. The object of research of the bachelor's thesis is the processes of search and tracking of cultural events. The subject of research is software and technical, organizational bases, principles, approaches of construction of web application of cultural events of the city. Research methods - analytical (helps to conduct research on the choice of optimal tools and approaches to service design, based on available data), comparative (used in the study of existing analogues of web services and their clients and technical means, and is to choose the best option comparative data analysis), abstraction (used to focus on the main aspects of existing analogues of web services for reviewing cultural events of the city, to identify and find existing problems that can be solved), modeling (used to draw conclusions about the object of study based on study of analogues of existing services), forecasting (used to form assumptions about the object of study). As a result, a web service for viewing cultural events of the city was developed, as well as a convenient and modern mobile client for him, which aims to help the user find interesting cultural events, view and add interesting cultural events to favorites. The software implementation of the web service is made on the ASP.NET Core Web API, the software implementation of the mobile client is made on Xamarin.Forms, the C # programming language is used.

Keywords: mobile application, web service, cultural event, programming patterns, application development on the ASP.NET Core Web API / Xamarin.Forms platform.

ЗМІСТ

ВСТУП

Актуальність дослідження цієї кваліфікаційної роботи бакалавра полягає у створенні сучасного і зручного веб-сервісу для перегляду культурних заходів міста і забезпечення користувача комфортним інструментом для пошуку і відслідковування культурних заходів на мобільному пристрої. Зважаючи на події сьогодення, може здатися, що людям зараз не до культурних заходів, проте саме культурні заходи, як можливість зібратися десь в компанії і повеселитися або розширити світогляд, є однією з можливостей трохи покращити психологічний стан людини.

Також, важливо зауважити, що наразі на ринку майже всі веб-сервіси з перегляду культурних заходів міста мають лише веб-клієнт, що наштовхує нас на думку про створення мобільного клієнту для нашого веб-сервісу, адже всі ми завжди маємо при собі смартфон. А, отже, актуальним є створення мобільного клієнту для веб-сервісу з перегляду культурних заходів міста для того, щоб використати також і можливості мобільної системи.

Метою дослідження є підвищення ефективності пошуку та відслідковування культурних заходів міста з використанням веб-сервісу.

Завдання дослідження:

- ознайомитися з сучасними мобільними технологіями (для розроблення зручного інтерфейсу);
- ознайомитися з сучасними веб-технологіями (з можливістю використання їх при створенні веб-сервісу);
- розроблення бекенду (отримання даних з бази даних, обробка цих даних, відправка їх на фронт), сучасні підходи до розроблення швидких і оптимізованих алгоритмів з отримання та обробки даних;
- проектування архітектури бази даних (дотримання нормальних форм, для уникнення перевантаження бази даних), а також побудова оптимізованих запитів до бази даних.

Об’єктом дослідження кваліфікаційної роботи бакалавра є процеси пошуку та відслідковування культурних заходів.

Предметом дослідження кваліфікаційної роботи бакалавра є програмно-технічні, організаційні засади, принципи, підходи побудови веб-застосунку культурних заходів міста.

Методи дослідження:

- аналітичний – використовується у практичній частині кваліфікаційної роботи бакалавра, допомагає в проведенні досліджень щодо вибору оптимальних інструментів і підходів до проектування сервісу, за рахунок наявних даних;
- порівняльний – використовується при вивчені існуючих аналогів веб-сервісів та їх клієнтів і технічних засобів, і полягає у виборі кращого варіанту за рахунок порівняльного аналізу даних;
- абстрагування – використовується для зосередження на головних аспектах існуючих аналогів веб-сервісів з перегляду культурних заходів міста, для виділення і знаходження існуючих проблем, що можуть бути вирішені;
- моделювання – використовується для формування висновків щодо об’єкту дослідження на основі вивчення аналогів існуючих сервісів;
- прогнозування – використовується для формування припущення щодо об’єкту дослідження.

У процесі виконання кваліфікаційної роботи бакалавра було використано програмне забезпечення Visual Studio, Figma, SSMS, Android Emulator, документація C#/Xamarin Forms/Sql Server/Web API, міжнародні стандарти з побудови веб-сервісів, стандарти з побудов Android застосунків та зручних і сучасних користувацьких інтерфейсів.

Практичне значення одержаних результатів полягає у створенні веб-сервісу культурних заходів міста, та мобільного клієнту до нього, за допомогою якого можна:

- переглянути культурні заходи, додаткову інформацію, їх місцезнаходження;

- зберегти культурний захід, що вас зацікавив, і налаштувати нотифікацію для нього;
- фільтрувати існуючі заходи за декількома ознаками (назвою, описом, датою).

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, поділених на підрозділи, висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 103 сторінки. Список використаних джерел охоплює 20 найменувань.

РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ ПІДХОДІВ ДО ПРОЕКТУВАННЯ ВЕБ-СЕРВІСІВ ТА АНАЛІЗ ІСНУЮЧИХ ВЕБ-СЕРВІСІВ КУЛЬТУРНИХ ЗАХОДІВ МІСТА

1.1 Огляд вимог до побудови веб-сервісу для перегляду культурних заходів міста

Культурні заходи завжди відігравали величезну роль у життєдіяльності будь-якого суспільства з давніх часів. На сьогодні, актуальність культурних заходів анітрохи не зменшилась, адже беручи участь у подібних івентах людина піднімає свій настрій, знаходить нові корисні зв’язки, а також розширює власний світогляд.

З появою інтернету люди отримали можливість користуватись широким спектром послуг, зокрема це стосується і області культурних заходів. Веб-застосунки культурних заходів надають можливість переглянути свіжі події, відфільтрувати їх за певними показниками, побачити місце та дату проведення, іноді, надають можливість замовити квитки на ці заходи.

Загалом, основними вимогами до побудови веб-сервісу для перегляду культурних заходів міста є актуальність і коректність інформації щодо тих чи інших культурних заходів, можливість побачити місцезнаходження проведення культурного заходу та наявність зручного і легкого у користуванні клієнта, адже користувач буде взаємодіяти безпосередньо з ним, а не веб-сервісом.

1.2 Аналіз існуючих аналогів. Обґрунтування необхідності вирішення даної задачі

Наразі на українському веб-просторі існує достатня кількість веб-застосунків з перегляду культурних заходів. Але більшість з них мають майже ідентичний функціонал і можливості, тому, задля кращого аналізу, серед усіх застосунків було обрано 4 застосунки, а саме:

- MoeMisto;

- KyivMaps;
- VechirniyKyiv;
- Офіційний портал Києва, Розділ культурно-мистецьких заходів.

Почнемо із застосунку MoeMisto, перейшовши на головну сторінку нас зустрічає стандартний і якісний, сучасний інтерфейс, який можна побачити на рис. 1.1.

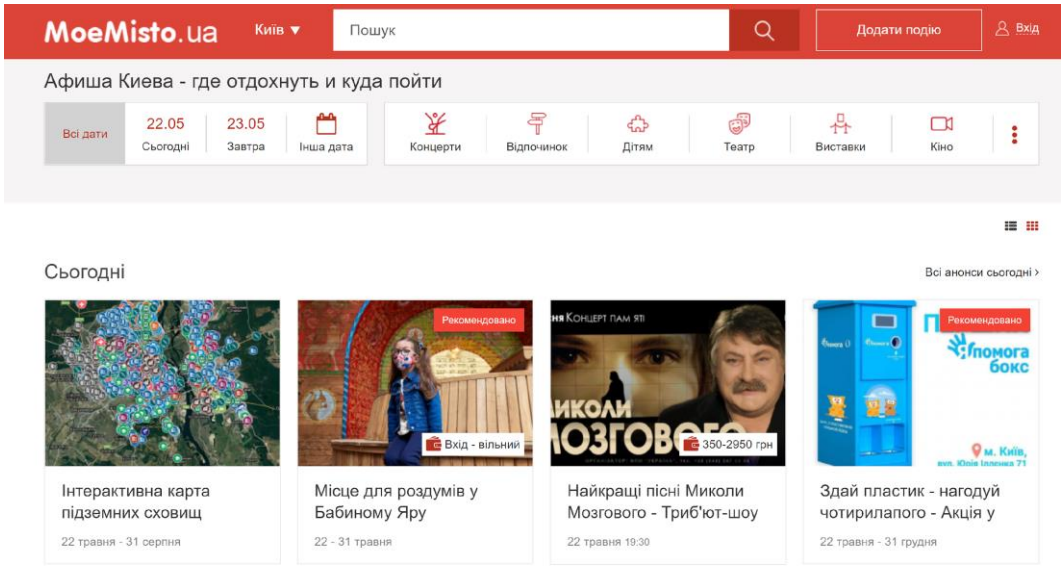


Рисунок 1.1 – Головна сторінка MoeMisto

З позитивних сторін можна зазначити:

- гнучкий пошук за різноманітними категоріями, датою або назвою;
- відображення як платних заходів (із зазначенням ціни квитка), так і безкоштовних;

Але варто звернути увагу і на слабкі сторони цього застосунку: якщо перевести відображення подій у форматі вертикального списку – зображення зникають, це можна побачити на рис. 1.2.

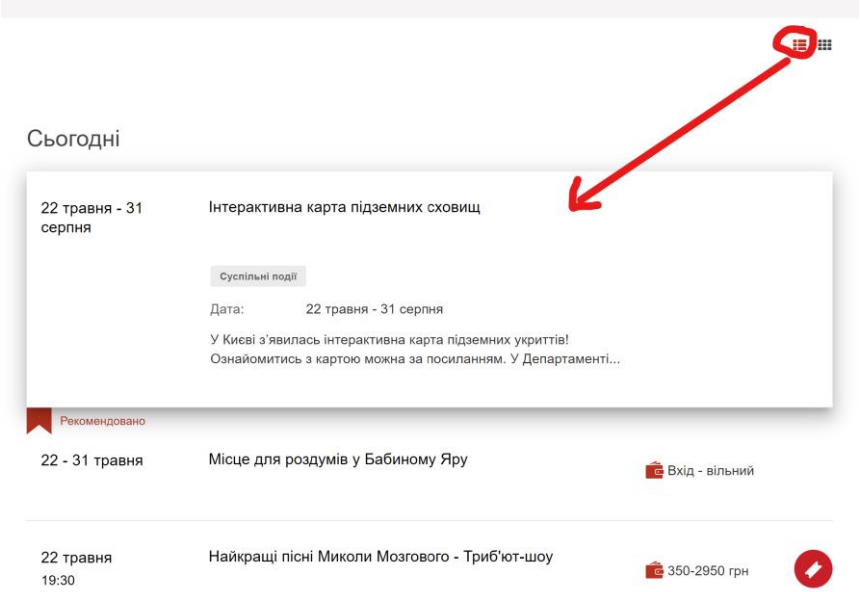


Рисунок 1.2 – Відсутність зображення при зміні формату

Перейшовши на певну подію перед нами відкривається більш детальна інформація стосовно події, серед важливої інформації варто виділити адресу та інтерактивну карту, на якій одразу можна побачити де саме знаходиться місце проведення події (рис. 1.3).

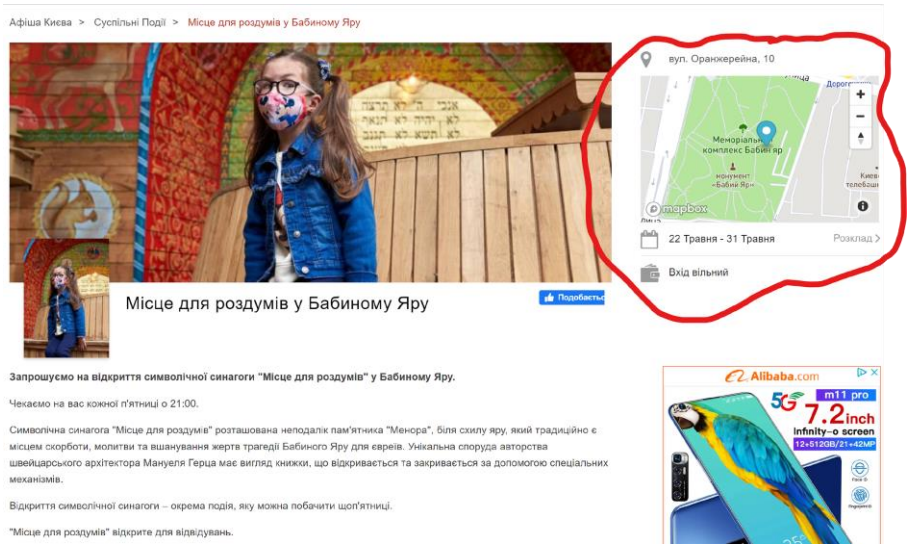


Рисунок 1.3 – Блок з інтерактивною картою

Нижче можна залишити свій коментар стосовно цієї події, або поділитися нею з іншими людьми.

Наступний на черзі у нас застосунок KyivMaps, головною особливістю якого (стає зрозуміло з назви) є більш просунута можливість роботи з картами.

При переході до головної сторінки нас зустрічає аналогічний до попередника інтерфейс, проте якщо натиснути кнопку «Більше подій», або обрати певну подію, ми бачимо подібний до Гугл Мепс інтерфейс, де зліва знаходиться список подій, а справа – інтерактивна карта, яка показує позначки всіх подій, а також деяку інформацію, якщо натиснути на маркер (рис. 1.4).

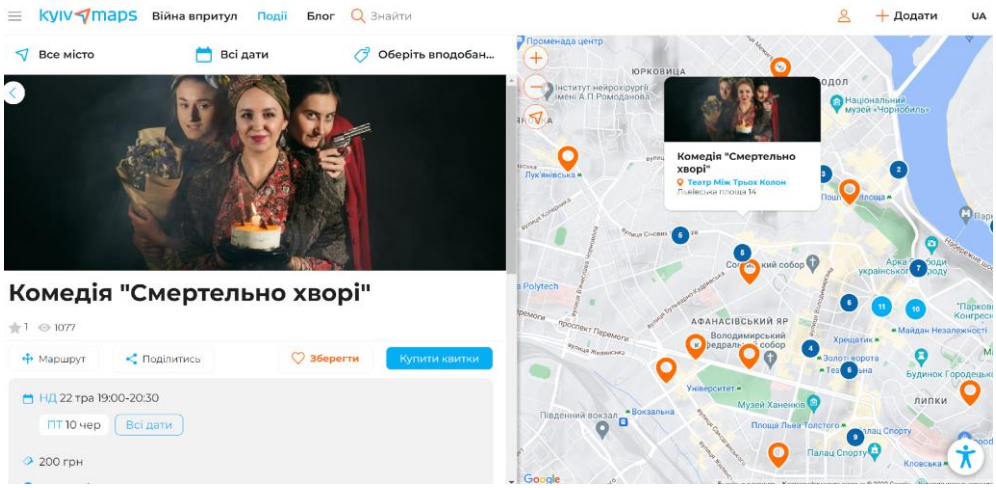


Рисунок 1.4 – Приклад роботи застосунку KyivMaps при виборі певної події

Це можна вважати головною особливістю цього застосунку на противагу іншим. Загалом KyivMaps виглядає потужним конкурентом всім іншим застосункам, проте його головна перевага є одразу і недоліком — не всім користувачам потрібна настільки просунута карта.

Наступним аналогом є «Вечірній Київ», застосунок одразу зустрічає нас надто перенасиченим і дещо «несучасним» інтерфейсом (рис. 1.5).

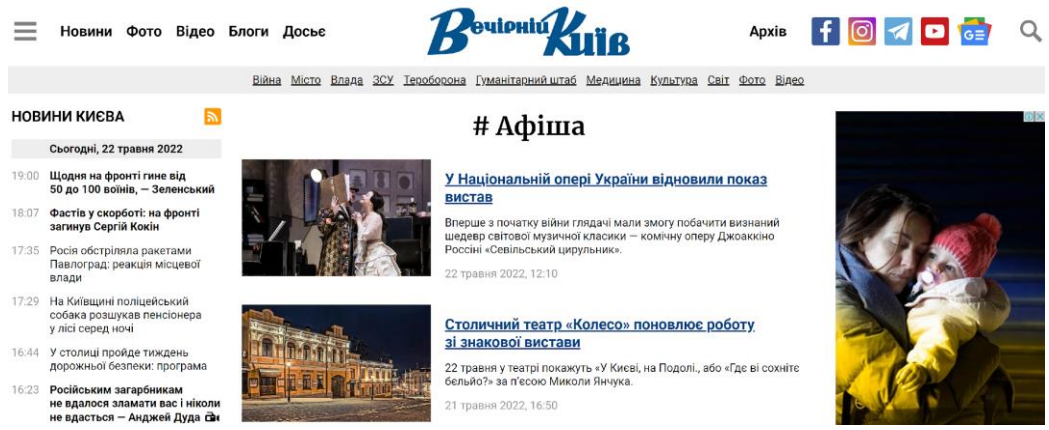


Рисунок 1.5 – Головна сторінка «Вечірній Київ»

Якщо перейти на будь-яку подію, то можна побачити таке саме нагромадження інформації, що не сприяє її розумінню, розмір зображень не регулюється відносно екрану, тому виглядає все дещо криво. Бракує фільтрів. Загалом відношення до застосунку негативне.

Останнім піддослідним є Офіційний портал Києва, Розділ культурно-мистецьких заходів. Головну сторінку можна побачити на рис.1.6.

Поглянувши на сторінку, можна одразу помітити, що на офіційному порталі не використовують зображення, що не є мінусом. Присутній стандартний пошук за назвою, не вистачає фільтрів. Серед переваг можна зазначити можливість переключення на версію для людей з порушеннями зору, що виділяє портал на фоні інших застосунків.

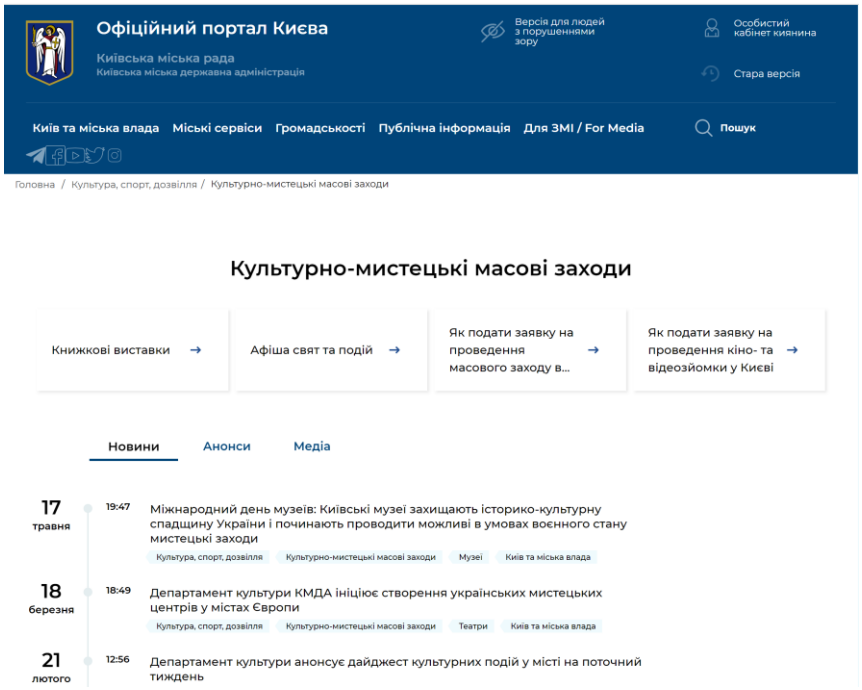


Рисунок 1.6 – Головна сторінка Офіційного порталу Києва

Переглянувши існуючі аналоги майбутнього застосунку і проаналізувавши їх сильні та слабкі сторони, приходимо до висновку, що більшість застосунків мають дуже схожий інтерфейс, що негативно впливає на їх пізнаваність (що важливо), функціонал в основному однаковий.

Серед зазначених проблем, які зустрічаються в веб-застосунках з перегляду культурних заходів міста можна виділити такі:

- орієнтація лише на веб, під час пошуку аналогів не було знайдено застосунків з мобільним або десктопним клієнтом;
- деякі застосунки перенасичені різними елементами, що призводить до більш повільної роботи, або призводить до незручності у користуванні;
- неможливість налаштування нагадувань щодо цікавих користувачеві подій.

1.3 Постановка цілей і вимог, які мають бути реалізовані у застосунку з перегляду культурних заходів міста

Відповідно до теми кваліфікаційної роботи бакалавра, маємо реалізувати веб-застосунок з перегляду культурних заходів міста, спробувавши надати користувачеві якісь нові та корисні можливості.

Отже, застосунок повинен:

- бути мобільним застосунком, що надає цілий спектр можливостей, недоступних для тільки веб-застосунків, серед яких можна виділити можливість нотифікації, підключення камери, відслідковування геолокації тощо;
- бути невеликого розміру, для заощадження місця на пристрої користувача;
- бути простим і зрозумілим в плані інтерфейсу, щоб користувач міг користуватись застосунком однією рукою;
- мати можливість бути розширеним до клієнту на інших типах систем (IOS, Windows).

Серед головного функціоналу можна виділити:

- перегляд культурних заходів;
- пошук культурних заходів за різними параметрами;

- можливість зберігання інформації про цікаві клієнту заходи на пристрої, де він матиме доступ до них без використання інтернету;
- можливість налаштування нотифікації, для нагадування користувачеві про культурний захід у зручний для нього час.

Відповідно до теми кваліфікаційної роботи бакалавра, маємо реалізувати веб-застосунок з перегляду культурних заходів міста, спробувавши надати користувачеві якісь нові та корисні можливості.

У підсумку, провівши аналіз існуючих веб-застосунків подібних до того, який буде створено в рамках даної кваліфікаційної роботи бакалавра, було описано необхідний набір функцій, які будуть у застосунку і функцій, які покращать досвід користування клієнта.

РОЗДІЛ 2. АНАЛІЗ І ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА АРХІТЕКТУРНИХ РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

2.1 Аналіз та вибір підходів до проектування веб-застосунків

Серед сучасних підходів до розробки веб-застосунків можна виділити [4]:

- статичний − застосунки розроблені переважно на HTML CSS у тому вигляді, у якому були створені адміністратором чи веб-майстром, статичні веб-застосунки зазвичай важко підтримувати, проте вони можуть бути практичним вибором, якщо передається дуже коротка інформація і взаємодія з клієнтом не потрібна;
- динамічний – на противагу статичному, контент динамічних сайтів генерується за рахунок запиту від клієнту та буде відрізнятися залежно від типу запиту, серед найпопулярніших програмних засобів до розробки бекендової частини можна виокремити PHP і ASP.NET.

Так як ми маємо створити застосунок, де користувач зможе взаємодіяти із застосунком, переглядаючи і зберігаючи культурні заходи, то нашим вибором буде розроблення динамічного веб-застосунку. Однак, динамічні веб-застосунки теж поділяються на декілька типів.

Серед підходів до розробки динамічних застосунків можна визначити такі:

- SPA – або односторінкові застосунки, дозволяють клієнту безперешкодно спілкуватися зі сторінкою веб-сайту, запити та відповіді виконуються ефективно через обмежену кількість інформації, за рахунок виконання логіки на рівні клієнту працюють швидше в порівнянні з іншими застосунками.
- MPA – або багатосторінкові застосунки, працюють так само, як і традиційні веб-застосунки, коли застосунок перезавантажується і показує іншу сторінку з сервера в програмі щоразу, коли клієнти відтворюють деякі дії. Логіка таких застосунків зазвичай зберігається

на рівні бекенду, який відправляє запити клієнтів на сервер і повертає відповідь.

- RIA – тип веб-застосунків, який в основному має кілька функцій настільних застосунків, але працює швидше і покладається на плагіни на стороні клієнта (Flash, Shockwave, Silverlight). Подібні застосунки можна використовувати навіть в автономному режимі.

Крім підходів до розроблення важливо також виокремити підходи до рендеру застосунків [5]:

- SSR (Server-Side Rendering, серверний рендеринг) – при серверному типу рендеринга у відповідь на запит на сервері генерується весь HTML-код сторінки, це виключає необхідність додаткових запитів даних з боку клієнта, так як сервер бере всю роботу на себе, перш ніж відправити відповідь.
- CSR (Client-Side Rendering, рендеринг на клієнті) – при клієнтському типу, рендеринг сторінок відбувається прямо в браузері за допомогою JS, вся логіка, отримання даних, шаблонізація і маршрутизація оброблюється на клієнті.

Зважаючи на те, що у першому розділі було зазначено, що наш застосунок повинен мати мобільний клієнт, необхідність у традиційному веб-застосунку відсутня, а отже постає необхідність у веб-застосунку, який може передавати якісь дані у форматі повідомлень, які будуть прийматися, опрацьовуватися і відображатися на мобільному клієнті. Поглянемо на Web API.

Web API є універсальним засобом для підключення різноманітних програмних застосунків, який допомагає будувати важкі функціональні одиниці прикладаючи менше зусиль. Їх особливістю є приховування великої кількості коду від програміста. Принцип роботи можна побачити на рис.2.1.

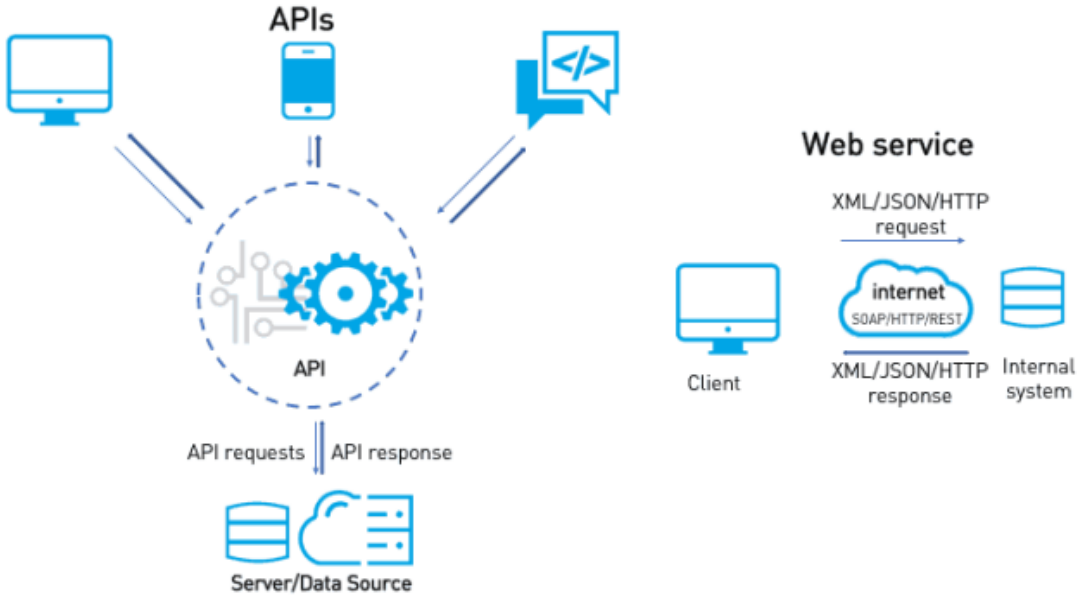


Рисунок 2.1 – Зображення традиційної взаємодії з API

Загалом можна виділити 4 типи API:

- Публічні (доступні для всіх, зазвичай використовуються для монетизації);
- Партнерські (доступні лише для конкретних споживачів);
- Внутрішні (призначені для внутрішнього користування, а тому зазвичай мають слабкий захист);
- Композитні (поєднують у собі два або більше API).

За протоколом передачі даних на сьогодні займають лідерство три протоколи, а саме:

- REST – мабуть, найпопулярніший підхід до створення API. REST спирається на підхід клієнт/сервер, який розділяє передній і задній частини API і забезпечує значну гнучкість у розробці та впровадженні. REST має статус «без стану», що означає, що API не зберігає жодних даних або статусу між запитами. REST підтримує кешування, яке зберігає відповіді для повільних або нечутливих до часу API. API REST, які зазвичай називають «RESTful».
- RPC – протокол віддаленого процедурного виклику (RPC), є простим засобом надсилання кількох параметрів та отримання результатів. API

RPC викликають виконувані дії або процеси, тоді як REST API в основному обмінюються даними або ресурсами, такими як документи. RPC може використовувати дві різні мови, JSON і XML, для кодування; ці API мають назву JSON-RPC і XML-RPC відповідно.

- SOAP – простий протокол доступу до об’єктів (SOAP), це стандарт обміну повідомленнями, визначений Консорціумом World Wide Web і широко використовується для створення веб-API, зазвичай із XML. SOAP підтримує широкий спектр протоколів зв’язку, доступних в Інтернеті, таких як HTTP, SMTP і TCP. SOAP також є розширюваним і незалежним від стилю, що дозволяє розробникам писати SOAP API різними способами та легко додавати функції та модулі.

Зважаючи на описані вище дані, можна зробити висновок, що найбільш вдалим вибором типу API буде REST WEB API, адже воно чудово підходить для простих мобільних застосунків.

Rest API являє собою архітектурний підхід до побудови стандартизованих інтерфейсів. Основними особливостями цього підходу є:

- Уніфікований інтерфейс (спрощення загальної архітектури досягається за рахунок, наприклад: однозначної ідентифікації кожного ресурсу у взаємодії клієнта-сервера; одного представлення для всіх ресурсів; достатності кожного ресурсу як одиниці інформації; надавання клієнту лише початкового URL, а подальші шляхи динамічно змінювати у програмі).
- Клієнт-серверна архітектура (розділення клієнта і сервера на різні компоненти, що дозволяє створювати клієнта на декількох платформах з єдиним сервером для них).
- Відсутність станів (сервер не повинен зберігати контекстну інформацію про користувача, клієнт це робить і направляє на сервер всю інформацію необхідну для розуміння та виконання запиту).
- Кешування (використання частини даних, які були використані клієнтом в попередніх запитах, в поточному запиті).

- Багаторівнева система застосунку (означає, що система буде складатися з ієрархічних шарів шляхом обмеження поведінки компонентів).
- Код за запитом (дозволяє розширити можливості клієнта за рахунок завантаження скриптів з серверу і їх виконання).

Побудову архітектури REST API можна звести до чотирьох базових операцій (самих операцій може бути більше, однак чотири – це мінімум для того, щоб веб-сервіс можна було вважати повноцінним). Ці операції корелюють з методами HTTP-запитів:

- GET – отримання даних з сервера, зазвичай у форматі JSON.
- POST – додавання на сервер нових даних.
- PUT – модифікація даних на сервері.
- DELETE – видалення даних з сервера.

2.2 Вибір технології для створення веб-застосунку

За рахунок розвитку цифрових технологій на сьогоднішній день існує безліч програмних і технічних засобів для розробки застосунків різної величини і складності, створення повноцінного застосунку поділяється на:

- клієнт (фронтенд) – відповідає за зовнішній вигляд веб-застосунку, а також надає користувачу інтерфейс взаємодії з бекендом, серед популярних мов можна зазначити HTML/CSS/JS, а також безліч різних фреймворків для JS: Angular, React, Vue.js;
- сервер (бекенд) – відповідає за логіку і функціонал застосунку, відправку та обробку даних, зв’язок з базою даних, серед популярних наразі мов програмування є PHP, Python, Ruby, C#;
- база даних – відповідає за створення бази даних (реляційна чи не реляційна), може містити різноманітні процедури і транзакції, якщо логіка роботи з базою даних не перенесена на бекенд. Серед популярних реляційних баз даних можна назвати MySql, Sql Server, PostgreSql, серед нереляційних: MongoDB, Redis, Cassandra.

Так як маємо необхідність розробити мобільний застосунок, то варіант HTML/CSS/JS нам не підходить (хіба що було б вирішено використати React Native). А отже, для розробки клієнту необхідно подивитись на відповідні мови/фреймворки.

Загалом, для розробки мобільних застосунків на сьогодні використовують низку мов програмування в парі з відповідними фреймворками, а саме:

- Java – в минулому був найпопулярнішим варіантом з розробки андроїд застосунків, наразі потрохи витісняється Kotlin.
- Kotlin – розширення мови Java спеціалізоване на створенні Android-застосунків, наразі дуже актуальне.
- C# – за рахунок потужної бази можливостей Microsoft, маємо можливість розробляти будь-яке програмне забезпечення на C#, для мобільної розробки можна використати Xamarin.Forms, який дозволяє створювати кросплатформерні застосунки для будь-якої системи (Android, IOS, Windows) пишучи одну кодову базу для всіх.
- Python – добре відомий своєю читабельністю та простотою використання, що може призвести до швидшої розробки. Для розроблення мобільних застосунків на Python використовується фреймворк Kivy, він має графічну реалізація обробки, створеної на основі OpenGL, тому він може обробляти робочі навантаження, пов’язані з GPU, коли це необхідно. Він також має проект python-to-android, який дозволяє переносити програми Python на Android. Він має подібний набір інструментів для iOS, хоча на даний момент пакунки для iOS можна створювати лише за допомогою Python 2.7.
- React Native – JS-бібліотека, дозволяє створювати нативні застосунки на Android/IOS, що є надзвичайно зручним для розробників на JS.
- Dart – сучасна мова програмування, має Java-подібний синтаксис, розроблена компанією Google, направлена на розроблення кроссплатформерних застосунків на різні ОС, для розроблення мобільних застосунків використовує фреймворк Flutter.

Зважаючи на переваги платформи .NET і наявність досвіду розробки на ньому, наш вибір падає на Xamarin.Forms, адже це дуже зручно писати один код на C# (на бекенді), та XAML (на фронтенді) і компілювати це у застосунок на будь-якій ОС з незначними змінами у нативі.

2.3 Аналіз і вибір БД і технологій для зв'язку з БД

При створенні застосунку з перегляду культурних заходів міста необхідно обрати базу даних, в якій будуть зберігатися культурні заходи та інформація про них. Так як для створення веб-застосунку було обрано платформу ASP.NET Core Web API з усіма наслідками, що випливають,, базою даних, яка буде використовуватися є Sql Server, тому що теж є продуктом Microsoft і має налагоджений процес роботи з платформою .NET.

Для зв’язку з базою даних зазвичай використовують ORM – технологія програмування, яка пов’язує бази даних з концепціями об’єктно-орієнтованих мов програмування, що дозволяє абстрагуватися від бази даних і працювати з даними незалежно від типу сховища. Серед найпопулярніших ORM можна зазначити Entity Framework 6, яка дозволяє працювати з базою даних, використовуючи LINQ і об’єкти, замість таблиць і запитів. Проте використання ORM не завжди добре впливає на швидкість роботи ПЗ, тому в проекті планується використати чистий ADO.NET із запитами на T-SQL.

ADO.NET надає собою технологію роботи з даними, яка заснована на платформі .NET Framework. Ця технологія надає набір класів, через які ми можемо відправляти запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних і виробляти ряд інших операцій. Функціонал ADO.NET побудований таким чином, щоб надати розробникам уніфікований інтерфейс для роботи з самими різними СУБД.

Схематично архітектуру ADO.NET можна представити таким чином (рис. 2.2):

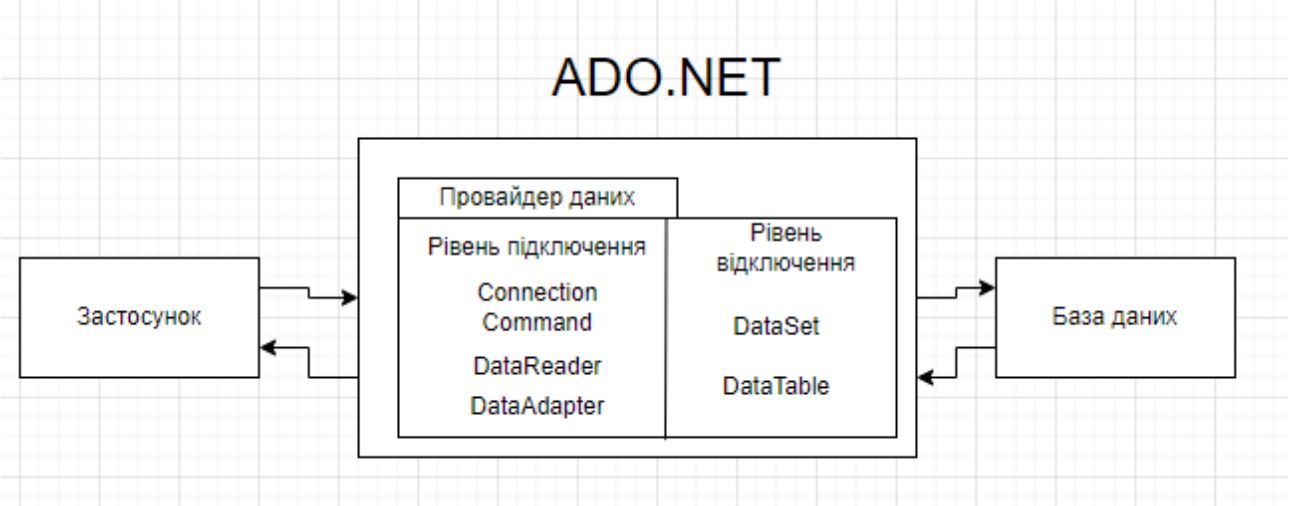


Рисунок 2.2 – Зображення зв’язку додатку і БД за допомогою ADO.NET [6]

Основу інтерфейсу взаємодії з базами даних в ADO.NET представляє обмежене коло об’єктів: Connection, Command, DataReader, DataSet і DataAdapter.

2.4 Проектування користувальницького інтерфейсу

Відповідно до завдання поставленого у даній кваліфікаційній роботі, клієнт для веб-сервісу культурних заходів міста має бути зручним і зрозумілим. Після переглянутих аналогів прийшли до висновку, що застосунок для перегляду культурних заходів міста має складатися з чотирьох модулів:

- головного вікна;
    - вікна з культурними заходами;
    - вікна зі збереженими культурними заходами;
- вікна перегляду додаткової інформації про культурний захід;
    - вікно додаткової інформації;
    - вікно мапи.

Для проектування користувальницького інтерфейсу було використано застосунок Figma, який надає широкий спектр можливостей навіть на безкоштовній версії.

Головне вікно і вікно користувача можна побачити на рис. 2.3 і рис. 2.4.

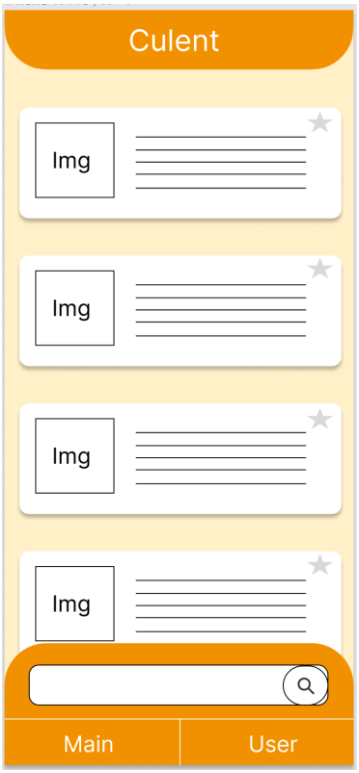


Рисунок 2.3 – Головне вікно

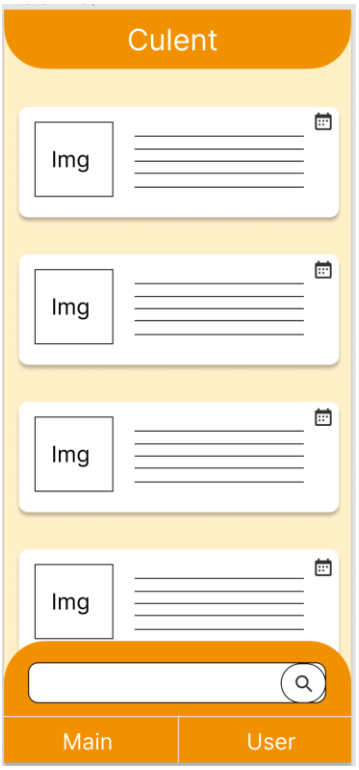
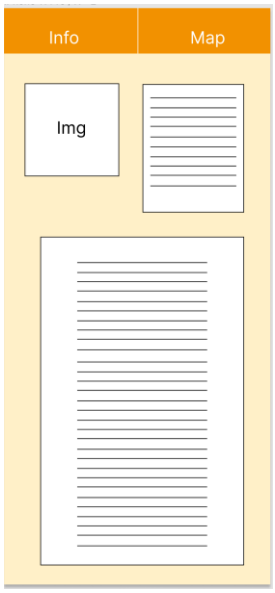


Рисунок. 2.4 – Вікно користувача

Головне вікно і вікно користувача мають схоже наповнення, крім одного — у вікні користувача присутнє поле для вибору дати, під час якої він буде сповіщений про конкретний культурний захід.

Вікно додаткової інформації про культурний захід і вікно з інтерактивною мапою, на якій можна побачити місцезнаходження заходу можна побачити на рис. 2.5 і рис. 2.6.



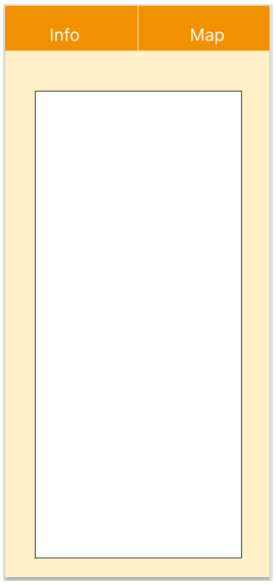


Рисунок 2.5 – Вікно повної інформації Рисунок 2.6 – Вікно інтерактивної мапи

2.5 Структура бази даних, принцип роботи інформаційної системи

Так як у даній кваліфікаційній роботі бакалавра ми в більшій мірі працюємо з фронтендом, а саме клієнтом на мобільному пристрої, в плані бази даних робота має мало моделей даних, а отже і таблиць відповідно. Необхідно створити три таблиці:

Таблиця 2.1 — Структура таблиці Category

Поле	Тип	Призначення
CategoryId	INT	Айді категорії
CategoryName	NVARCHAR(100)	Ім’я категорії

Таблиця 2.2 — Структура таблиці Event

Поле	Тип	Призначення
EventId	INT	Айді події
EventName	NVARCHAR(100)	Ім’я події
EventDescription	NVARCHAR(3000)	Опис події
EventDate	DATETIME	Дата проведення події
EventNotif	DATETIME	Дата нагадування про подію
EventAddress	NVARCHAR(100)	Адреса події
EventLatitude	NVARCHAR(50)	Довгота
EventLongtitude	NVARCHAR(50)	Широта

Таблиця 2.3 — Структура таблиці EventCategory

Поле	Тип	Призначення
EventCategoryId	INT	Айді відношення
EventId	INT	Айді події
CategoryId	INT	Айді категорії

Як було зазначено у попередньому розділі для зв’язку з базою даних, яка локально розташована в проекті використовується технологія ADO.NET. Логіка взаємодії з базою даних розроблена у вигляді методів, які надсилають запит у форматі T-SQL запитів, зроблено це для оптимізації швидкості отримання даних з бази даних, адже компілятору немає необхідності у перекладі методів C# у SQL.

2.6 Опис структури інформаційної системи

Створений застосунок для перегляду культурних заходів міста складається з низки файлів різного типу, що мають певну ієрархію і взаємодіють між собою.

На рис. 2.7 представлено загальну структуру проекту у форматі інтуїтивно зрозумілої діаграми, побудованої за допомогою застосунку app.diagrams.net:

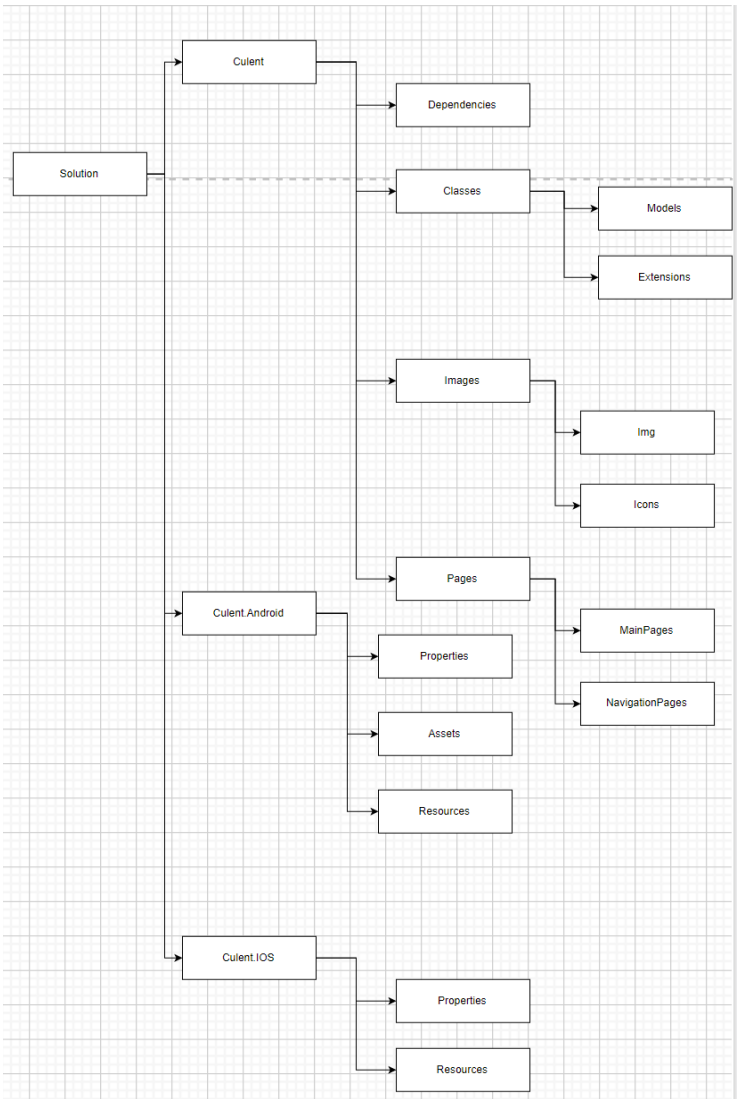


Рисунок 2.7 – Схематичне зображення структури проекту

Кожна з цих папок містить класи, які відповідають за логіку застосунку, а також файли, які відповідають за зовнішній вигляд (інтерфейс) застосунку.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ВПРОВАДЖЕННЯ ВЕБ-СЕРВІСУ КУЛЬТУРНИХ ЗАХОДІВ МІСТА

3.1 Програмна реалізація веб-сервісу культурних заходів міста

Для початку розробимо мобільний клієнт. Для створення проекту необхідно скачати собі Microsoft Visual Studio, як робоче середовище та встановити декілька необхідних компонентів. Файл для завантаження можна знайти на їх сайті, вибираємо Community версію, як зручний і безкоштовний варіант.

Після завантаження встановлюємо, після чого необхідно перейти в Visual Studio Installer і натиснути «Змінити» і вибрати декілька нових компонентів для встановлення, а саме:

- ASP.NET і розробка веб-застосунків;
- Розробка мобільних застосунків на .NET.

Далі створюємо проект і обираємо тип проекту «Mobile App(Xamarin.Forms)”, серед вибраних шаблонів обираємо пустий. Операційні системи, для яких буде створюватися застосунок, залишаємо дефолтними (Andoid, IOS).

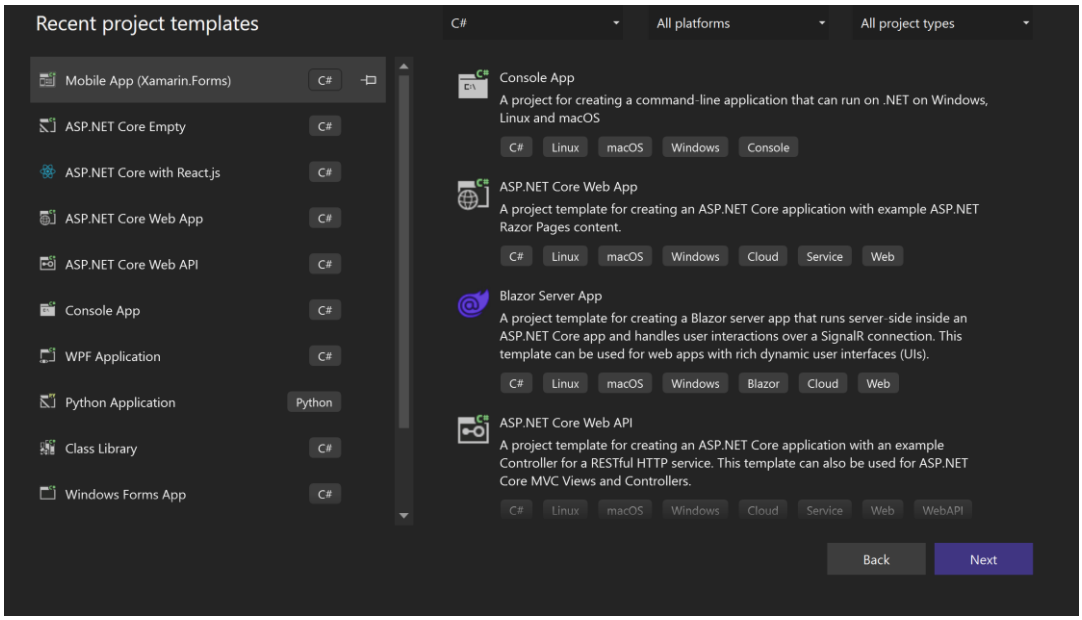


Рисунок 3.1 – Проект для створення

Після створення проекту у солюшині бачимо три базові проекти:

- Culent – головний проект, в якому зосереджена базова для усіх систем логіка і фронт, більшість часу будемо писати у ньому;
- Culent.Android – проект для додаткового налаштування застосунку під Android систему, підключення стилів і ресурсів;
- Culent.IOS – проект для додаткового налаштування застосунку під Android систему, підключення стилів і ресурсів, його використовувати не будемо, адже для роботи з IOS проектами необхідно мати Macbook і аккаунт розробника.

На початку роботи з проектом варто звернути увагу на основні (на даний момент часу) сторінки (рис. 3.2).

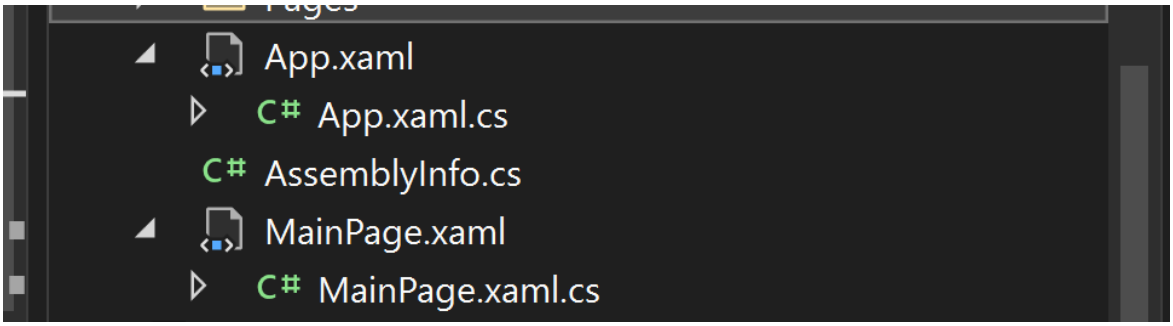


Рисунок 3.2 – Перші сторінки

Кожна сторінка складається XAML-файлу, а також з CS-файлу, в яких зберігається їх зовнішній вигляд і логіка роботи відповідно.

Файл App.xaml.cs по суті являє собою вхідну точку у наш застосунок.

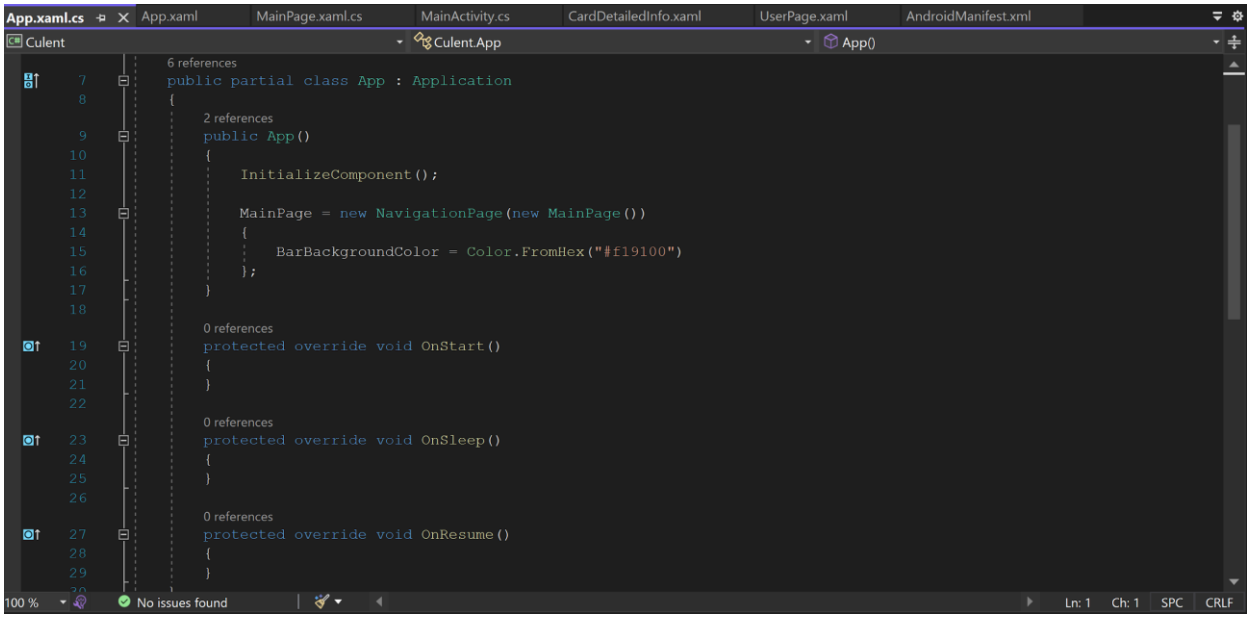


Рисунок 3.3 – Код файлу App.cs

По факту, він тримає у собі лише низку методів, які будуть відбуватись при відповідних станах застосунку (на запуску роботи застосунку, при переході в режим сну і при поверненні назад), а також конструктор, який запускає метод ініціалізації компоненту, який поєднує замл-файл і cs-файл у одну збірку, а вже її перетворює у нативний для відповідної системи код.

На рис. 3.3 видно, що крім виклику методу ми також ініціалізуємо об’єкт класу «Головна сторінка», саме він буде першим, що побачить користувач при запуску застосунку.

В xaml-файлі застосунку записали декілька базових кольорів, для усього застосунку, щоб було зручніше ними користуватись в будь-якому класі (рис. 3.4).

```
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Culent.App">
    <Application.Resources>
        <Color x:Key="navColor">#f19100</Color>
        <Color x:Key="backColor">#fff0c7</Color>
        <Color x:Key="cardColor">#FAF7EE</Color>
    </Application.Resources>
</Application>
```

Рисунок 3.4 – Основні кольори застосунку

Перейдемо до файлу «Головної сторінки», базовий файл містить пару лейблів і посилання на документацію, змінимо його на необхідну нам, відповідно до стилю, замальованого у Figma.

Так як наш застосунок складається з декількох табів, а Shell в цій роботі ми не будемо використовувати — маємо необхідність створити спеціальну сторінку типу «Таб», а наповнення буде складатися з посилань на інші звичайні сторінки. Для цього зробимо так, щоб головна сторінка наслідувала клас “Tabbed Page” і змінимо замл-файл відповідно до рис. 3.5.

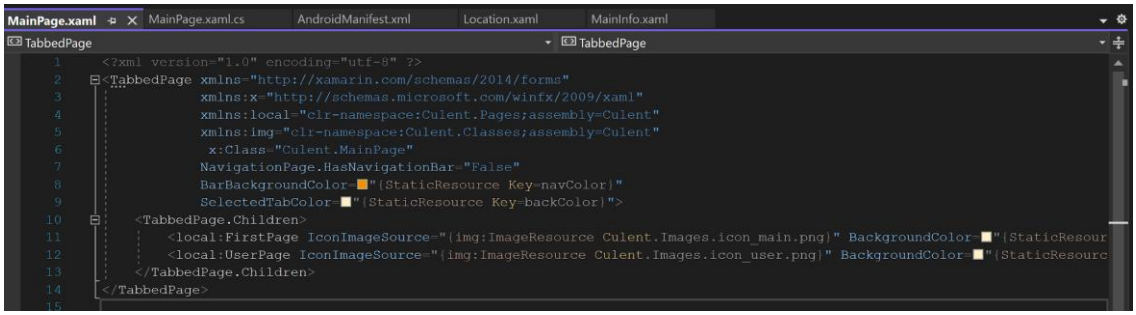


Рисунок 3.5 – Сторінка посилається на інші сторінки-таби

Зокрема необхідно виділити, що кожна система інтерпретує код збірки по-своєму, наприклад базовим для Android-застосунку є розташування табів угорі. Це не дуже підходить для нашого застосунку, адже маємо бажання, щоб користувач міг легко користуватись застосунком однією рукою, тому розташуємо їх знизу. Для цього необхідно перейти в файл логіки головної сторінки і імпортувати спеціальний простір імен для андроїд специфікації і після цього у конструкторі класу вказати розташування табів унизу (рис. 3.6).

Рисунок 3.6 – Команда для специфікації застосунку для андроїд-систем

Після цього все має працювати як належне. Перейдемо до пункту зі стилями і зовнішнім виглядом застосунку, адже дизайн теж є важливою частиною для зручності користування застосунком.

Так, для використання стилів, записаних у App-замл файлу необхідно користуватися біндингом, а саме командами Static Resource/Dynamic Resource.

Рисунок 3.7 – Використовуємо стиль для навбару таким чином

Може виникнути питання — завантажити ресурс як статичний чи динамічний? Відповідь полягає у тому, що динамічні ресурси необхідно використовувати лише тоді, коли в рантаймі (безпосередньо під час роботи застосунку) має змінюватись стиль, прикладом цього може бути заміна базової теми на нічну, коли настає темрява.

Також можна побачити на рис. 3.7, що ми використовуємо картинку як лого, є правила для використання подібних ресурсів у Xamarin.Forms, для того, щоб воно працювало належним чином необхідно проставити опцію «побудови» для картинки як «вбудований ресурс», що покращить продуктивність. Для того щоб не плодити однакові файли по проектам зручніше за все створити одну папку за ресурсами у головному проекті, у даному випадку вона називається Images і тримає у собі лого, іконки і стокові зображення (рис. 3.8).

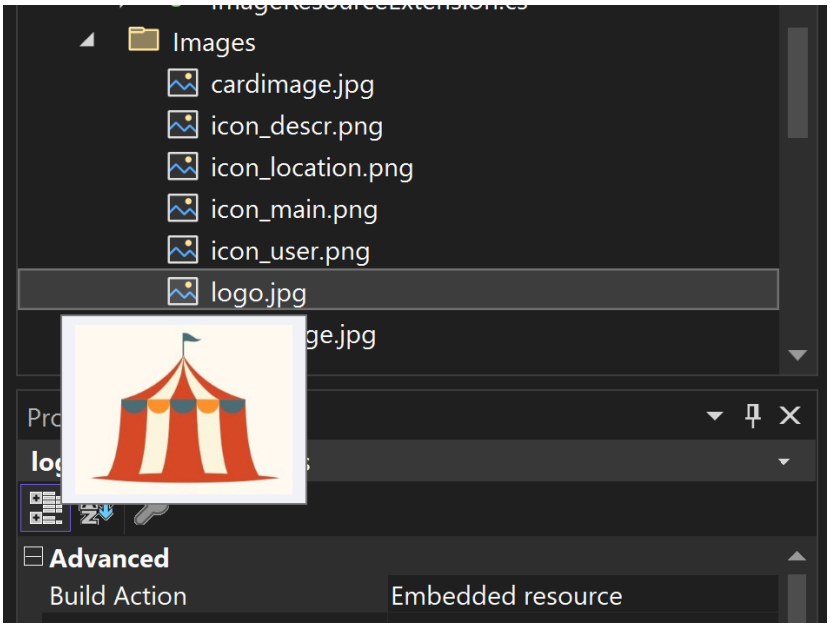


Рисунок 3.8 – Лого застосунку, помічене як «вбудоване»

Важливо також звернути увагу на використання зображення у застосунку. Існує багато методів для завантаження зображень з інтернету, локальної папки чи бази даних, проте існує і проблема – ці методи знаходяться на рівні cs-файлів і логіки, а на стороні фронту їх використати не можна. Для того щоб мати можливість прописувати шлях до ресурсів прямо у замл-файлі необхідно створити метод розширення, який буде повертати ресурс на стороні фронту за вказаним шляхом. Для цього створимо ImageResourceExtension і заповнимо його відповідним кодом (рис. 3.9).

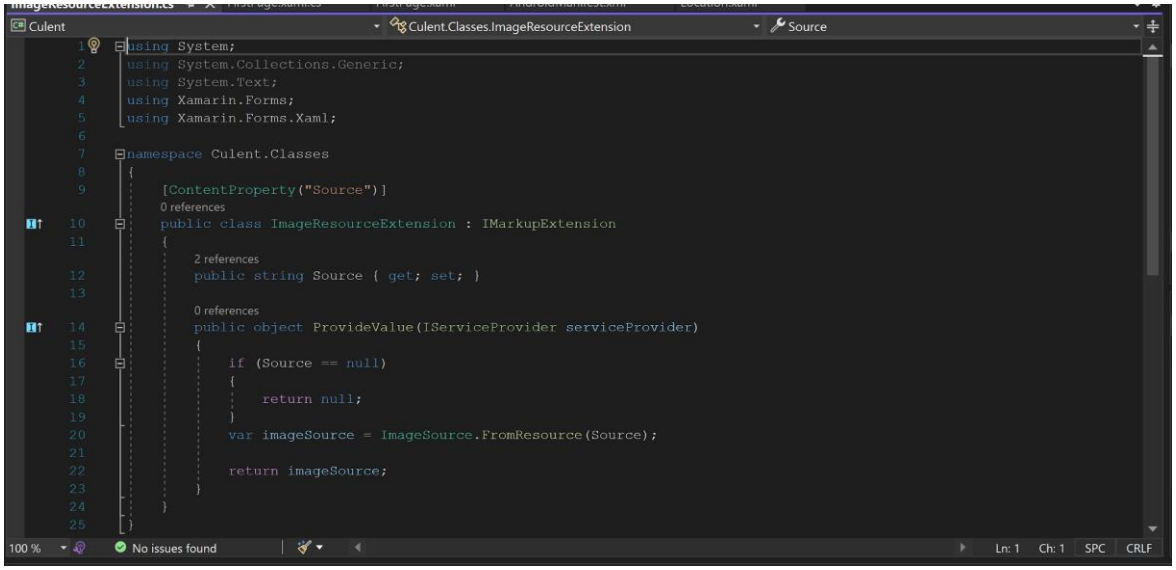


Рисунок 3.9 – Код классу ImageResourceExtension

Як можна побачити в коді – для доступу до методу будемо використовувати властивість «Source». Після чого прописуємо шлях до класу, попередньо оголосивши для нього зміну «img»

```
Title="Main"
x:Class="Culent.Pages.FirstPage"
xmlns:img="clr-namespace:Culent.Classes;assembly=Culent"
```

Рисунок 3.10 – Імпорт методу розширення у xaml-файл

Далі перейдемо до важливої частини головної сторінки, а саме контейнерів компонування. Це компоненти, які тримають у собі інші компоненти і відображають їх у специфічному форматі. Загалом є декілька основних контейнерів:

- StackLayout – стандартний контейнер, якій відображає компоненти послідовно, один за одним, вертикально або горизонтально;
- AbsoluteLayout – контейнер, який дозволяє задавати вкладеним елементами абсолютні кординати розташування на сторінці;
- RelativeLayout – дозволяє позиціонувати елементи відносно країв контейнеру, або інших елементів;
- Grid – стандартний контейнер грід, який розташовує елементи у форматі таблиці.

Найзручнішим тут буде контейнер Grid, адже надає змогу позиціонувати елементи в 2 вимірах, а простір, який має той чи інший елемент коректується кількістю колонок або рядків, які він тримає у собі.

```
<Grid RowSpacing="0" ColumnSpacing="0">
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>

    <Frame BackgroundColor="■{StaticResource Key=navColor}" Grid.Row="0" Grid.RowSpan="1">
        <Frame CornerRadius="20"
               HorizontalOptions="Center"
               Padding="0"
               IsClippedToBounds="True">
            <Image
            Source="{img:ImageResource Culent.Images.logo.jpg}"
            VerticalOptions="Center"
            HorizontalOptions="Center"
            />
        </Frame>
    </Frame>
```

Рисунок 3.11 – Перший елемент фрейм займає першу колонку і перший рядок

Перейдемо до основної частину головної сторінки, а саме компоненту «список», одного з компонентів-колекцій для відображення масиву даних в одному специфічному форматі. Найбільш вдалим є “ListView”, адже надає можливість просто і зручно відобразити дані у форматі вертикального спадаючого списку.

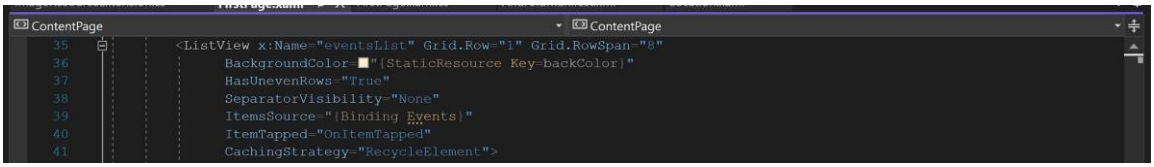


Рисунок 3.12 – Базові налаштування для компоненту списку

Бачимо на зображенні, що колекція займає 1/8 від загального місця, за рахунок гріда як контейнера – вона буде однаково добре виглядати на пристрої будь-якого розміру. Як джерело даних вказана колекція Events, яка реагує на зміни, але про це поговоримо пізніше. Також бачимо підв’язаний обробник подій, який буде спрацьовувати при натисканні на комірку. Бачимо також увімкнення кешування даних колекції, що оптимізує процес завантаження даних і покращує швидкість роботи.

Тепер перейдемо до налаштування комірки колекції (одна комірка являє собою один культурний захід).

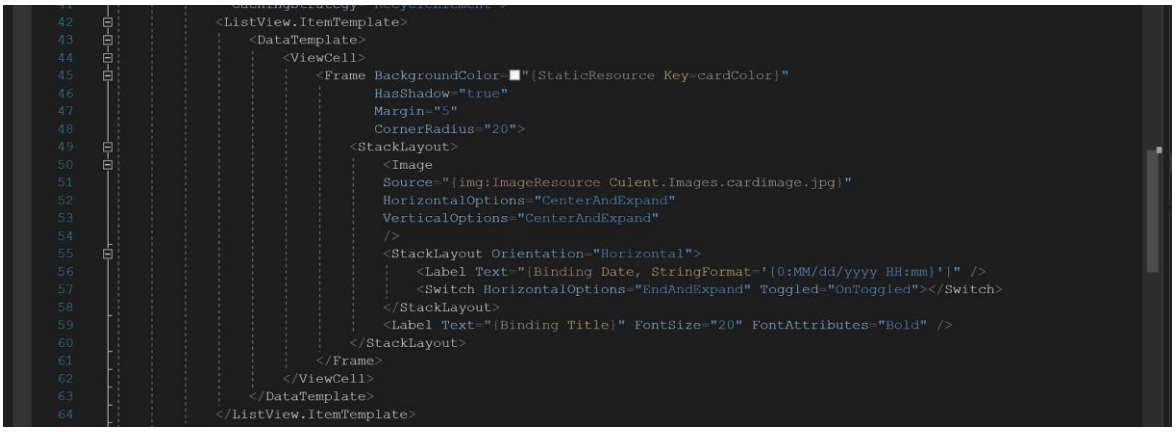


Рисунок 3.13 – Налаштування комірки колекції

Як можемо бачити, у тезі ViewCell можна записувати як елементи, так і контейнери, що дозволяє робити дуже потужний дизайн.

Тепер маємо компонент для відображення даних, проте необхідно мати якусь можливість фільтрувати дані в колекції. Для цього інтегруємо у сторінку компонент “SearchBar”, що слугує полем для вводу різної інформації (можна вибирати тип клавіатури, яка буде відображатись в залежності від типу даних), і підв’яжемо відповідний обробник подій “Text_Changed”.

Рисунок 3.14 – Налаштування компоненту “SearchBar”

Далі перейдемо до логіки головної сторінки, яка знаходиться у “FirstPage.xaml.cs”. В цьому файлі на поточний момент зосереджено логіка обробки тапів користувача, а також ініціалізація стартових даних.

Рисунок 3.15 – Ініціалізація даних

Як ми бачимо, в коді створюється змінна типу ObservableCollection з дженеріком типу Event. Особливістю цієї колекції є те, що вона, при зміні свого стану, наприклад додаванні або видалені елементів колекції, посилає відповідні повідомлення уверх по ієрархії. Це використовується для того, щоб при зміні стану колекції на бекенді ми могли бачити зміни і на фронтенді. Дженерік використовується для того, щоб програма на етапі компіляції зрозуміла, що за тип буде використовуватись і працювала з ним відповідно до його можливостей у дженеріку. В даному випадку використано модель Event, властивості якої можна побачити нижче.

Рисунок 3.16 – Властивості моделі

Проте це тільки модель даних, безпосередньо колекції створюються у відповідному класі EventLists.cs, код якого можна побачити нижче.

Рисунок 3.17 – Клас колекцій

Можна звернути увагу на те, що колекції тут створюються з розширенням «статік», це робиться для того, щоб мати один, загальний стан колекцій для всього застосунку, адже статичні об’єкти створюються один раз при зверненні до них, або першої ініціалізації об’єкту класу. Для доступу для цих колекцій достатньо лише прописати повний шлях до неї, без ініціалізації об’єкту.

Перевагою використання подібного типу ініціалізації даних є те, що за необхідності можна легко розширити модель необхідними властивостями, або створити зовсім нову модель і замінити колекції двома кліками. Також дозволяє використовувати LINQ, технологія доступу і фільтрації даних в різних сховищах

(бази даних, об’єкти, масиви і тд) у форматі методів розширення, або SQL-подібного синтаксису.

Перейдемо до наступного шматку логіки у класі, а саме обробки натискання на цікавий користувачеві елемент.

```
26      private async void OnItemTapped(object sender, ItemTappedEventArgs e)
27      {
28          Event selectedEvent = e.Item as Event;
29          if (selectedEvent != null)
30          {
31              eventsList.SelectedItem = -1;
32              CardDetailedInfo infoPage = new CardDetailedInfo();
33              infoPage.BindingContext = selectedEvent;
34              await Navigation.PushAsync(infoPage);
35          }
36      }
```

Рисунок 3.18 – Обробка натискання на якусь подію

Як ми можемо бачити, при натисканні ми отримуємо безпосередньо об’єкт типу Event і потім викликаємо сторінку CardDetailedInfo.xaml, яка дозволяє переглянути інформацію про подію більш детально. Інформацію передаємо за рахунок властивості BindingContext і асинхронно викликаємо сторінку за допомогою вбудованої у Xamarin.Forms технології навігації.

Далі перейдемо до обробки події Text_Changed, яку використовує компонент SearchBar.

```
38      private void SearchBar_TextChanged(object sender, TextChangedEventArgs e)
39      {
40          if (string.IsNullOrEmpty(e.NewTextValue))
41          {
42              eventsList.ItemsSource = Events;
43          }
44          else
45          {
46              eventsList.ItemsSource = Events.Where(x => x.Title.StartsWith(e.NewTextValue));
47          }
48
49          //var coll = from p in Phones
50          //           where p.Title.ToLower().StartsWith(input) || p.Title.ToLower().Contains(input)
51          //           || p.Title.ToLower().EndsWith(input) ||
52          //           p.Company.ToLower().StartsWith(input) || p.Company.ToLower().Contains(input)
53          //           || p.Company.ToLower().EndsWith(input) ||
54          //           p.Date.ToString("dd.MM.yyyy").StartsWith(input) || p.Date.ToString("dd.MM.yyyy").Contains(input)
55          //           || p.Date.ToString("dd.MM.yyyy").EndsWith(input)
56          //           select p;
57      }
```

Рисунок 3.19 – Пошук подій в колекції

Так як ми працюємо з колекціями і можемо використовувати LINQ — робимо це. Таким чином при зміні стану поля пошуку програма буде перевіряти поле на наявність там якогось значення і якщо воно не дорівнює null, воно буде

шукати співпадіння підстроки у колекції. Наразі воно шукає лише за початком назви події, проте нижче у коменті можна побачити шматок коду, який дозволяє шукати події за всіма ознаками одразу (назві, опису, даті, адресі).

І останнім, але не менш важливим є функціонал зберігання цікавих події у іншому місці, це робиться за рахунок компоненту switch і активації події OnToggled.

Рисунок 3.20 – Логіка додавання події у бажане

При активації світча програма дістає інформації про цю події через приведення типів, а далі йде перевірка, якщо подібна подія вже є у колеції – нічого не буде, але якщо немає – буде додано новий елемент.

В свою чергу при деактивації світчу робиться перевірка навпаки і якщо елемент присутній – його видалять, якщо нема – нічого не станеться.

Перейдемо до вікна більш детальної інформації. Після натискання на цікаву подію буде викликано табсторінку CardDetailedInfo, яка підтягує з собою дві звичайні сторінки MainInfo і Location.

Рисунок 3.21 – Код табсторінки

Оглянемо сторінку MainInfo, яка містить у собі повну інформацію про подію, загалом, її структура проста і складається з декількох блоків (зображення,

опис, дата проведення). Для побудови структури сторінки знову використовувались гріди.

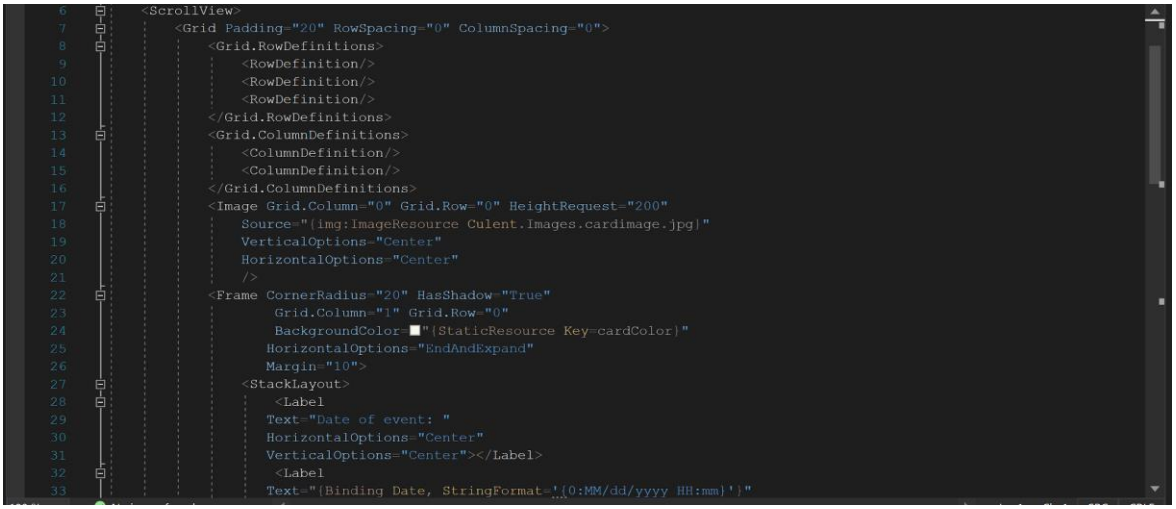


Рисунок 3.22 – Структура сторінки

Далі розглянемо сторінку Location, яка містить у собі інтерактивну карту, що дозволяє подивитись розташування місця проведення події. Тут все цікавіше за попередню сторінку.

Для початку необхідно отримати ключ Google API, для використання карт, для цього необхідно перейти в гугл клауд, створити проект, підключити до проекту Maps API і згенерувати для проекту ключ доступу.

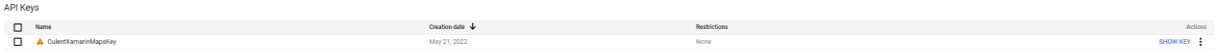


Рисунок 3.23 – Ключ згенеровано

Далі необхідно перейти до нашого проекту. Для кожної операційної системи налаштування карт робиться різними способами. В даному випадку працюємо з Android. Для цього перейдемо в проект Culent.Android у клас MainActivity, який є стартовою точкою у побудові застосунку на андроїді і напишемо ініціалізацію карт.

```
0 references
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);

    Xamarin.Essentials.Platform.Init(this, savedInstanceState);
    global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
    Xamarin.FormsMaps.Init(this, savedInstanceState);
    LoadApplication(new App());
}
0 references
public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Cor
{
    Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);

    base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
}
}
```

Рисунок 3.24 – Ініціалізація карт у Андроїд-проекті

Можемо помітити також метод OnRequestPermissions, який робить запит при першому запуску програми про можливість використання деяких даних, які будуть використовуватись застосунком.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0"
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="31" />
    <application android:label="Culent.Android" android:theme="@style/MainTheme">
      <meta-data android:name="com.google.android.geo.API_KEY"
           android:value="AIzaSyAh-xpO-LJSSIDVQFHocORjO_dkofpqdG0" />
      <uses-library android:name="org.apache.http.legacy" android:required="false"/>
    </application>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
</manifest>
```

Рисунок 3.25 – Список запитів на доступ

В цьому випадку перейдемо у файл AdroidManifest і зробимо запит на низку можливостей (про приблизне розташування користувача і про точне розташування користувача). Це дозволить розширити можливості користування Google Maps API.

Налаштування карт готове. Залишилось додати її на відповідну сторінку. Для цього перейдемо до сторінки Location і напишемо відповідний код.

```
<maps:Map x:Name="map"
  IsShowingUser="True"
  MoveToLastRegionOnLayoutChange="False">
   <x:Arguments>
      <maps:MapSpan>
         <x:Arguments>
            <maps:Position>
               <x:Arguments>
                  <x:Double>50.45072581718367</x:Double>
                  <x:Double>30.522969040670 67</x:Double>
               </x:Arguments>
            </maps:Position>
            <x:Double>0.01</x:Double>
            <x:Double>0.01</x:Double>
         </x:Arguments>
      </maps:MapSpan>
   </x:Arguments>
   <maps:Map.Pins>
      <maps:Pin Label="{Binding Title}"
        Address="{Binding Date, StringFormat='{0:MM/dd/yyyy HH:mm}'}"
        Type="Place">
         <maps:Pin.Position>
            <maps:Position>
               <x:Arguments>
                  <x:Double>50.45072581718367</x:Double>
                  <x:Double>30.522969040670 67</x:Double>
               </x:Arguments>
            </maps:Position>
         </maps:Pin.Position>
      </maps:Pin>
```

Рисунок 3.26 – Додавання компоненту карти до застосунку

Як бачимо, їх дуже легко налаштувати, можна додавати координати прямо у тегах, а також створити компоненту маркер, з контекстною інформацією про подію.

3.2 Документація веб-сервісу культурних заходів міста

Розроблений веб-сервіс культурних заходів міста, а також мобільний клієнт для нього надає змогу продивлятися різноманітні культурні заходи, шукати їх за різними ознаками, а також зберігати їх локально на пристрої і виставляти дату нотифікації.

Зручність інтерфейсу була досягнута за рахунок аналізу існуючих аналогів та вибору найкращих їх якостей.

Для початку роботи з сервісом користувачу необхідно відкрити мобільний застосунок “Culent”. В даному випадку сам веб-сервіс запущено на локальному сервері.

Так як ми перевіряємо роботу розробленого застосунку на Android Emulator — побачений результат буде зафіксовано на моделі телефону Pixel 5, проте можна використовувати і справжній телефон і будь-яку модель, застосунок буде коректно працювати.

Після відкриття застосунку ми бачимо стартове вікно застосунку, яке наведено на рис. 3.27.

На ньому можна виділити кілька основних розділів, а саме:

- головне вікно – при відкритті застосунку головне вікно відкривається першим, тут можна побачити список культурних заходів, переглянути більш детальну інформацію і додати культурні заході до бажаних;
- поле для пошуку – дозволяє знаходити бажані події за ознаками (назва, опис, дата проведення, категорії);
- панель табів – дозволяє робити перехід від головного вікна, де можна бачити всі культурні заходи, до вікна користувача, на якому

зберігаються цікаві користувачеві заходи, де він, також, має можливість виставити собі дату, при досягненні якої буде приходити сповіщення.

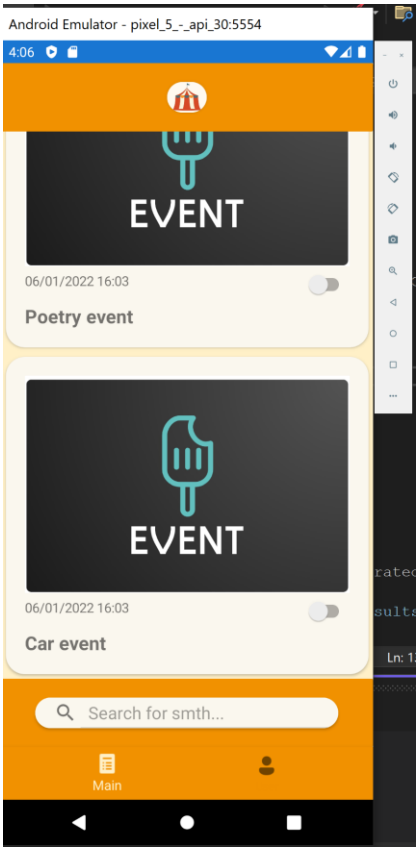


Рисунок 3.27 – Головне вікно, яке зустрічає користувача

Якщо у користувача виникає бажання знайти конкретну подію — він може скористатися спеціальним полем для пошуку. Пошук відбувається за декількома ознаками (назва, опис, дата проведення), на майбутнє може бути додано більше категорій.

Тож, після натискання на поле пошуку перед користувачем відкривається зручна клавіатура, з можливістю вводу як символів, так і цифр. Можна побачити на рис. 3.28.

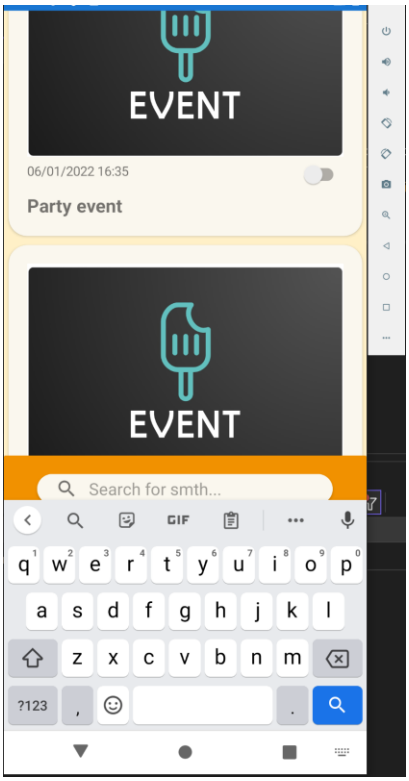


Рисунок 3.28 – Активація поля для пошуку

Для прикладу, давайте пошукаємо якісь культурний захід пов’язаний з машинами і напишемо “Car”, для того щоб знайти зв’язані події. Як можемо побачити на рис. 3.29 – подібна подія існує.

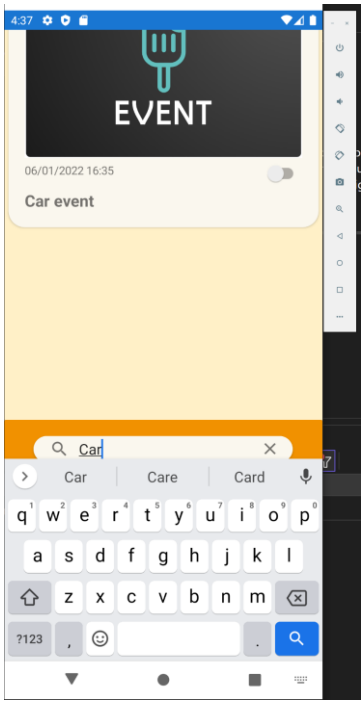


Рисунок 3.29 – Вдалий пошук

За бажанням, можна відмінити пошук натиснувши на хрестик, який зітре всі введені дані, рис. 3.30.

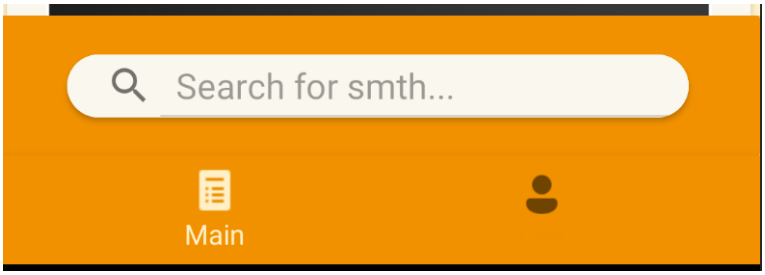


Рисунок 3.30 – Застосунок повернувся до попереднього стану

Якщо користувач зацікавився якоюсь подією і бажає почитати про неї більш детально, то необхідно натиснути на цікаву подію і відкриється вікно «Додаткової інформації», яке складається з табів:

- повна інформації про подію – містить опис і більш детальну інформацію;
- місце розташування – містить адресу і інтерактивну мапу, яка показує місцезнаходження події на ній.

Побачити це можна на рис. 3.31 і рис. 3.32.

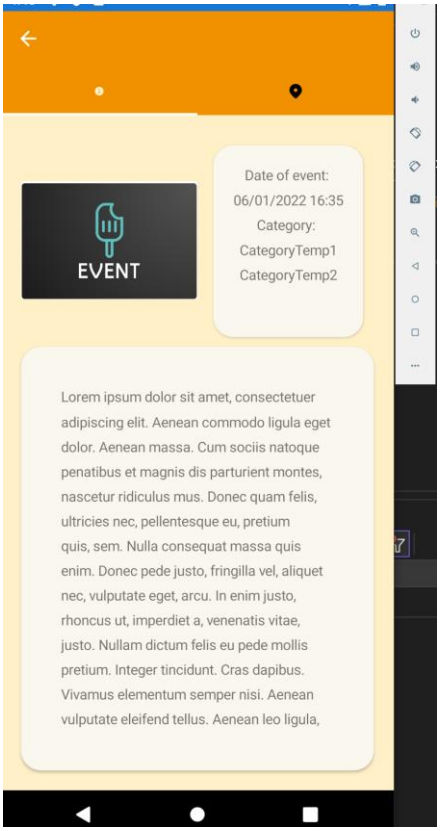


Рисунок 3.31 – Вікно повної інформації про подію

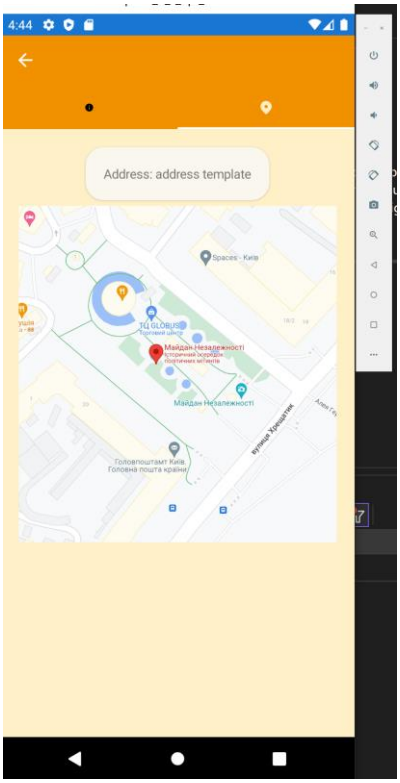


Рисунок 3.32 – Вікно інтерактивної мапи

Якщо користувач, прочитавши всю інформацію виявив побажання відвідати цей захід, він може додати цей культурний захід до бажаних, за рахунок Switch-тригеру (рис. 3.33).

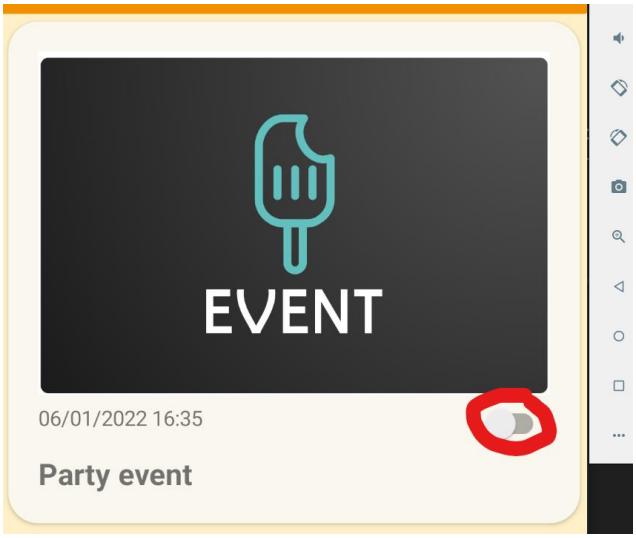


Рисунок 3.33 – Зручний і мінімалістичний чекбокс

Після активація цього чекбоксу даний захід локально збережеться на пристрої користувача і побачити його можна перейшовши на таб «Користувача» (рис. 3.34).

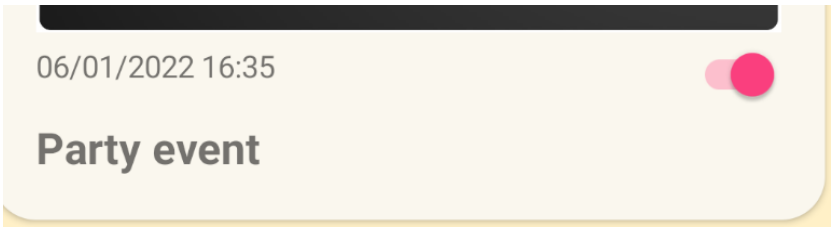


Рисунок 3.34 – Активований чекбокс

Для переходу на цей таб необхідно лише натиснути кнопку “User”, на панелі табів.

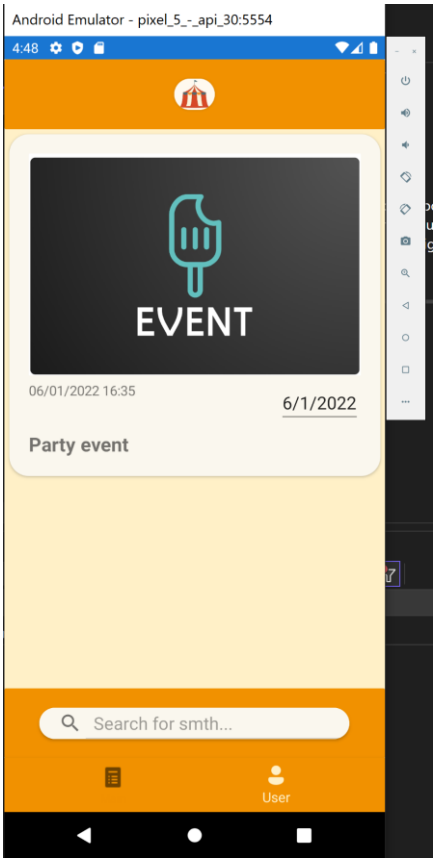


Рисунок 3.35 – Таб користувача, бачимо додану подію

Дивлячись на вікно користувача можна прийти до висновку, що дане вікно майже нічим не відрізняється від головного, крім того, що має відповідне поле для вибору дати.

Якщо користувач боїться забути про цікавий йому захід, то він має можливість обрати дату, при досягненні якої для нього прийде сповіщення на телефон про подію. Для цього необхідно обрати поле для вибору дати і обрати бажану дату (рис. 3.36).

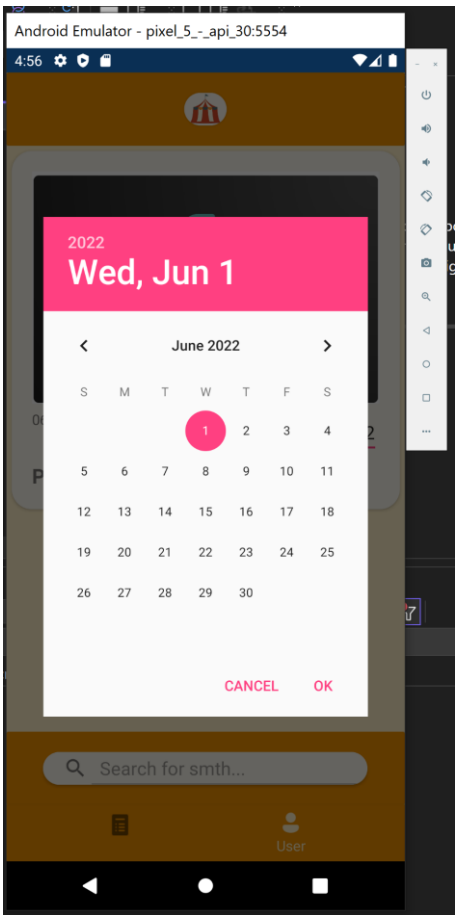


Рисунок 3.36 – Вікно для вибору дати

ВИСНОВКИ

За час роботи над кваліфікаційною роботою бакалавра було проведено детальний аналіз існуючих аналогів веб-сервісу культурних заходів міста, а також огляд їх недоліків і переваг. Відповідно до огляду було проведено вибір найбільш сучасних і ефективних технологій для розробки у сфері фронтенду, бекенду, баз даних і забезпечення веб-сервісу культурних заходів міста корисними можливостями, які могли б покращити досвід користування подібними застосунками.

Веб-сервіс та його клієнт були реалізовані за допомогою цілого списку сучасних і актуальних підходів і технологій. Розробка застосунку була проведена у програмному забезпеченні Visual Studio, компанії Microsoft. Основною мовою розробки у цій роботі була мова C#, на платформі .NET, XAML – мова розмітки, для побудови фронтенду, а також широкий спектр технологій, які надає розробникам компанія Microsoft. Серед використаних технологій, зокрема, можна зазначити ASP.NET Core WEB API, яка надає змогу будувати веб-сервіси зручно і швидко на базі технологій .NET, ADO.NET – технологія, що надає можливість зв’язку програми і бази даних у форматі уніфікованого інтерфейсу.

Особливостями даного веб-сервісу є невеликий розмір, мобільний клієнт, який надає цілу низку можливостей, наприклад – нотифікація, наприклад, яких не було у веб-клієнтах, а також швидкість роботи і зручність використання.

Загалом застосунок дозволяє:

- переглянути культурні заходи, додаткову інформацію, їх місцезнаходження;
- зберегти цікавий вам культурний захід і налаштувати нотифікацію для нього;
- фільтрувати існуючі заходи за декількома ознаками.

Серед можливостей для подальшого розвитку можна виокремити такі функціональності:

- підключення функціоналу відгуків і комментарів, щоб розширити базу даних при виборі культурного заходу для користувача;
- надання можливості роботи фотографії на згадку, прямо у застосунку, з можливістю використання різних фільтрів і зберігання фотографій у відповідних подіях на згадку;
- можливість поширення фотографій для інших людей, щоб вони могли оцінити їх;
- додати можливість створення локальних подій і запрошувати туди близьких вам людей.

У результаті виконання кваліфікаційної роботи бакалавра отримано широкий спектр актуальних знань, не лише у розробці програмного забезпечення, а й також у проектування UI/UX дизайну, формуванні правильної документації для користувача, а також правильному логічному формуванню і оформленню ідей, відповідно до поставлених вимог роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Підходи і інструменти до розробки веб-застосунків: веб-сайт. URL: https://habr.com/ru/post/539172/ (дата звернення: 10.02.2022).
2. Сучасні підходи до розробки веб-застосунків: веб-сайт. URL: https://prog-help.ru/javascript/samye-sovremennye-podhody-pri-razrabotke-sajtov-i-veb-prilozhenij/ (дата звернення: 10.02.2022).
3. Типи веб-застосунків: веб-сайт. URL: https://www.clustox.com/10-different-types-of-web-application-development/ (дата звернення: 11.02.2022).
4. Підходи до розробки веб-застосунків: веб-сайт. URL: https://ichi.pro/ru/10-razlicnyh-tipov-razrabotki-veb-prilozenij-186637186227438 (дата звернення: 13.02.2022).
5. SSR or CSR: веб-сайт. URL: https://tproger.ru/translations/rendering-on-the-web/ (дата звернення: 10.02.2022).
6. Поняття ORM: веб-сайт. URL: https://habr.com/ru/post/237889/ (дата звернення: 15.02.2022).
7. ADO.NET документація: веб-сайт. URL: https://metanit.com/sharp/adonet/1.1.php (дата звернення: 17.02.2022).
8. Документація C#: веб-сайт. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата звернення: 10.03.2022).
9. Andrew Troelsen, Philip Japikse - Pro C# 7: With .NET and .NET Core: підручник, 8th edition, 2017, 456 с.
10. Підручник з розробки на Xamarin.Forms: веб-сайт. URL: https://metanit.com/sharp/xamarin/4.7.php (дата звернення: 08.02.2022).
11. Типи Веб АПІ: веб-сайт. URL: https://www.techtarget.com/searchapparchitecture/tip/What-are-the-types-of-APIs-and-their-differences (дата звернення: 05.02.2022).
12. Введення в Web API: веб-сайт. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення: 17.04.2022).

13. Що таке REST: веб-сайт. URL: https://restfulapi.net/ (дата звернення: 19.04.2022).
14. Про ФІТ: веб-сайт. URL: http://fit.univ.kiev.ua/about_us (дата звернення: 04.02.2022).
15. Xamarin.Forms підручник: веб-сайт. URL: https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/ (дата звернення: 02.05.2022).
16. Підключення Google Maps API: веб-сайт. URL: https://docs.microsoft.com/ru-ru/xamarin/android/platform/maps-and-location/maps/obtaining-a-google-maps-api-key?tabs=windows (дата звернення: 15.05.2022).
17. Про Google Cloud: веб-сайт. URL: https://cloud.google.com/ (дата звернення: 15.05.2022).
18. C# підручник онлайн: веб-сайт. URL: https://metanit.com/sharp/tutorial/ (дата звернення: 11.03.2022).
19. Daniel Hindrikes, Johan Karlsson “Build multiplatform mobile apps and a game from scratch using C# and Visual Studio 2019, 2nd Edition”, 236 c.
20. Can Bilgin “Mobile Development with .NET: Build cross-platform mobile applications with Xamarin.Forms 5 and ASP.NET Core 5, 2nd Edition” 158 c.

ДОДАТКИ

ДОДАТОК А

А.1 — App.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Culent.App">
    <Application.Resources>
        <Color x:Key="navColor">#f19100</Color>
        <Color x:Key="backColor">#fff0c7</Color>
        <Color x:Key="cardColor">#FAF7EE</Color>
    </Application.Resources>
</Application>
```

А.2 — App.xaml.cs

```
using Culent.Classes;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent
{
    public partial class App : Application
    {
        public ApplicationViewModel viewModel;
        public App()
        {
            InitializeComponent();
            viewModel = new ApplicationViewModel();
```

```
            MainPage = new NavigationPage(new MainPage())
            {
                BarBackgroundColor = Color.FromHex("#f19100")
            };
        }

        protected async override void OnStart()
        {
            await viewModel.GetEvents();
            base.OnStart();
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
}
```

A.3 — Main.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            xmlns:local="clr-namespace:Culent.Pages;assembly=Culent"
            xmlns:img="clr-namespace:Culent.Classes;assembly=Culent"
             x:Class="Culent.MainPage"
```

```
        NavigationPage.HasNavigationBar="False"
        BarBackgroundColor="{StaticResource Key=navColor}"
        SelectedTabColor="{StaticResource Key=backColor}">
    <TabbedPage.Children>
        <local:FirstPage IconImageSource="{img:ImageResource
Culent.Images.icon_main.png}" BackgroundColor="{StaticResource
Key=backColor}"/>
        <local:UserPage IconImageSource="{img:ImageResource
Culent.Images.icon_user.png}" BackgroundColor="{StaticResource
Key=backColor}" />
    </TabbedPage.Children>
</TabbedPage>
```

A.4 — Main.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.PlatformConfiguration.AndroidSpecific;

namespace Culent
{
    public partial class MainPage : Xamarin.Forms.TabbedPage
    {
        public MainPage()
        {
            InitializeComponent();
```

```
On<Xamarin.Forms.PlatformConfiguration.Android>().SetToolbarPlacement(Toolba
rPlacement.Bottom);
        }
    }
}
```

A.5 — FirstPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             Title="Main"
             x:Class="Culent.Pages.FirstPage"
             xmlns:img="clr-namespace:Culent.Classes;assembly=Culent"
             NavigationPage.HasNavigationBar="False">

    <Grid RowSpacing="0" ColumnSpacing="0">
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
```

```
        <Frame BackgroundColor="{StaticResource Key=navColor}" Grid.Row="0"
Grid.RowSpan="1">
            <Frame CornerRadius="20"
                HorizontalOptions="Center"
                Padding="0"
                IsClippedToBounds="True">
              <Image
              Source="{img:ImageResource Culent.Images.logo.jpg}"
              VerticalOptions="Center"
              HorizontalOptions="Center"
              />
            </Frame>
        </Frame>
        <ListView x:Name="eventsList" Grid.Row="1" Grid.RowSpan="8"
            BackgroundColor="{StaticResource Key=backColor}"
            HasUnevenRows="True"
            SeparatorVisibility="None"
            ItemsSource="{Binding Events}"
            ItemTapped="OnItemTapped"
            >
            <ListView.ItemTemplate>
              <DataTemplate>
                 <ViewCell>
                    <Frame BackgroundColor="{StaticResource Key=cardColor}"
                        HasShadow="true"
                        Margin="5"
                        CornerRadius="20">
                      <StackLayout>
                        <Image
```

```
                    Source="{Binding Image}"
                    HorizontalOptions="FillAndExpand"
                    VerticalOptions="CenterAndExpand"
                    HeightRequest="200"

                    />
                    <StackLayout Orientation="Horizontal">
                        <Label Text="{Binding Date, StringFormat='{0:dd.MM.yyyy
HH:mm}'}" />
                        <Switch HorizontalOptions="EndAndExpand"
Toggled="OnToggled"></Switch>
                    </StackLayout>
                    <Label Text="{Binding Name}" FontSize="20"
FontAttributes="Bold" />
                </StackLayout>
            </Frame>
        </ViewCell>
    </DataTemplate>
</ListView.ItemTemplate>
</ListView>
<Frame BackgroundColor="{StaticResource Key=navColor}" Grid.Row="9">
    <Frame CornerRadius="20"
        HorizontalOptions="Center"
        Padding="0"
        IsClippedToBounds="True">
        <SearchBar TextChanged="SearchBar_TextChanged"
BackgroundColor="{StaticResource Key=cardColor}"  Placeholder="Search for
smth..."></SearchBar>
    </Frame>
</Frame>
```

```
    </Grid>
</ContentPage>
```

A.6 — FirstPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Culent.Classes;
using Culent.Classes.Models;
using Culent.Pages.NavigationPages;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent.Pages
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FirstPage : ContentPage
    {
        public ObservableCollection<Event> Events { get; set; }
        public FirstPage()
        {
            InitializeComponent();
            Events = ApplicationViewModel.Events;
            this.BindingContext = this;
        }
        private async void OnItemTapped(object sender, ItemTappedEventArgs e)
```

```
        {
            Event selectedEvent = e.Item as Event;
            if (selectedEvent != null)
            {
                eventsList.SelectedItem = -1;
                CardDetailedInfo infoPage = new CardDetailedInfo();
                infoPage.BindingContext = selectedEvent;
                await Navigation.PushAsync(infoPage);
            }
        }

        private void SearchBar_TextChanged(object sender, TextChangedEventArgs e)
        {
            if (string.IsNullOrEmpty(e.NewTextValue))
            {
                eventsList.ItemsSource = Events;
            }
            else
            {
                eventsList.ItemsSource = from p in Events
                                         where
p.Name.ToLower().StartsWith(e.NewTextValue.ToLower()) ||
p.Name.ToLower().Contains(e.NewTextValue.ToLower())
                                         ||
p.Name.ToLower().EndsWith(e.NewTextValue.ToLower()) ||

p.Description.ToLower().StartsWith(e.NewTextValue.ToLower()) ||
p.Description.ToLower().Contains(e.NewTextValue.ToLower())
                                         ||
p.Description.ToLower().EndsWith(e.NewTextValue.ToLower()) ||
```

```
p.Date.ToString("dd.MM.yyyy").StartsWith(e.NewTextValue) ||
p.Date.ToString("dd.MM.yyyy").Contains(e.NewTextValue)
                                    ||
p.Date.ToString("dd.MM.yyyy").EndsWith(e.NewTextValue)
                                    select p;
        }
    }
    private void OnToggled(object sender, ToggledEventArgs e)
    {
        var currSwitch = (Switch)sender;
        Event currEvent = (Event)currSwitch.BindingContext;

        if (currSwitch.IsToggled)
        {
            if (!ApplicationViewModel.FavouriteEvents.Contains(currEvent))
                ApplicationViewModel.FavouriteEvents.Add(currEvent);
        }
        else
        {
            if (ApplicationViewModel.FavouriteEvents.Contains(currEvent))
                ApplicationViewModel.FavouriteEvents.Remove(currEvent);
        }
    }
  }
}
```

A.7 — UserPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        Title="User"
        x:Class="Culent.Pages.UserPage"
        xmlns:img="clr-namespace:Culent.Classes;assembly=Culent"
        NavigationPage.HasNavigationBar="False">

    <Grid RowSpacing="0" ColumnSpacing="0">
      <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
      </Grid.RowDefinitions>

      <Frame BackgroundColor="{StaticResource Key=navColor}" Grid.Row="0"
Grid.RowSpan="1">
        <Frame CornerRadius="20"
            HorizontalOptions="Center"
            Padding="0"
            IsClippedToBounds="True">
          <Image
          Source="{img:ImageResource Culent.Images.logo.jpg}"
          VerticalOptions="Center"
```

```
            HorizontalOptions="Center"
            />
        </Frame>
    </Frame>
    <ListView x:Name="eventsList" Grid.Row="1" Grid.RowSpan="8"
        BackgroundColor="{StaticResource Key=backColor}"
        HasUnevenRows="True"
        SeparatorVisibility="None"
        ItemsSource="{Binding Events}"
        ItemTapped="OnItemTapped"
        >
        <ListView.ItemTemplate>
            <DataTemplate>
                <ViewCell>
                    <Frame BackgroundColor="{StaticResource Key=cardColor}"
                        HasShadow="true"
                        Margin="5"
                        CornerRadius="20">
                        <StackLayout>
                            <Image
                            Source="{Binding Image}"
                            HorizontalOptions="CenterAndExpand"
                            VerticalOptions="CenterAndExpand"
                            />
                            <StackLayout Orientation="Horizontal">
                                <Label Text="{Binding Date, StringFormat='{0:MM/dd/yyyy
HH:mm}'}" />
                                <Button Text="Notify date" Clicked="OnScheduleClick"
HorizontalOptions="EndAndExpand" CornerRadius="10"
```

```
                                BackgroundColor="{StaticResource Key=navColor}"
TextColor="{StaticResource Key=cardColor}"></Button>
                        </StackLayout>
                        <Label Text="{Binding Name}"  FontSize="20"
FontAttributes="Bold"  />
                    </StackLayout>
                </Frame>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
<Frame BackgroundColor="{StaticResource Key=navColor}" Grid.Row="9">
    <Frame CornerRadius="20"
        HorizontalOptions="Center"
        Padding="0"
        IsClippedToBounds="True">
        <SearchBar TextChanged="SearchBar_TextChanged"
BackgroundColor="{StaticResource Key=cardColor}"  Placeholder="Search for
smth..."></SearchBar>
    </Frame>
</Frame>
</Grid>
</ContentPage>
```

A.8 — UserPage.xaml.cs

```
using Culent.Classes;
using Culent.Classes.Models;
using Culent.Pages.NavigationPages;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.CommunityToolkit.Extensions;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent.Pages
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class UserPage : ContentPage
    {
        INotificationManager notificationManager;
        public ObservableCollection<Event> Events { get; set; }
        public UserPage()
        {
            InitializeComponent();
            Events = ApplicationViewModel.FavouriteEvents;
            this.BindingContext = this;
            notificationManager = DependencyService.Get<INotificationManager>();
        }

        private async void OnItemTapped(object sender, ItemTappedEventArgs e)
        {
            Event selectedEvent = e.Item as Event;
            if (selectedEvent != null)
            {
                eventsList.SelectedItem = -1;
```

```
            CardDetailedInfo infoPage = new CardDetailedInfo();
            infoPage.BindingContext = selectedEvent;
            await Navigation.PushAsync(infoPage);
        }
    }

    private void SearchBar_TextChanged(object sender, TextChangedEventArgs e)
    {
        if (string.IsNullOrEmpty(e.NewTextValue))
        {
            eventsList.ItemsSource = Events;
        }
        else
        {
            eventsList.ItemsSource = from p in Events
                                     where p.Name.ToLower().StartsWith(e.NewTextValue) ||
p.Name.ToLower().Contains(e.NewTextValue)
                                     || p.Name.ToLower().EndsWith(e.NewTextValue) ||
                                     p.Description.ToLower().StartsWith(e.NewTextValue) ||
p.Description.ToLower().Contains(e.NewTextValue)
                                     || p.Description.ToLower().EndsWith(e.NewTextValue) ||

p.Date.ToString("dd.MM.yyyy").StartsWith(e.NewTextValue) ||
p.Date.ToString("dd.MM.yyyy").Contains(e.NewTextValue)
                                     ||
p.Date.ToString("dd.MM.yyyy").EndsWith(e.NewTextValue)
                                     select p;
        }
    }
    async void OnScheduleClick(object sender, EventArgs e)
```

```
        {
            DatePopup datepopup = new DatePopup();
            DateTime? date;
            try
            {
                date = (DateTime)await Navigation.ShowPopupAsync(datepopup);
            }
            catch (NullReferenceException)
            {
                return;
            }
            if (date == null)
                return;
            var button = sender as Button;
            Event selectedEvent = button.BindingContext as Event;
            string title = $"Culent Notification!";
            string message = $"Don't forget about:{selectedEvent.Name} event!";
            notificationManager.SendNotification(title, message, date);
        }
    }
    public class NotificationEventArgs : EventArgs
    {
        public string Title { get; set; }
        public string Message { get; set; }
    }
}
```

A.9 — CardDetailedInfo.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:local="clr-
namespace:Culent.Pages.NavigationPages;assembly=Culent"
        xmlns:img="clr-namespace:Culent.Classes;assembly=Culent"
         x:Class="Culent.Pages.NavigationPages.CardDetailedInfo"
        BarBackgroundColor="{StaticResource Key=navColor}"
        SelectedTabColor="{StaticResource Key=backColor}">
   <TabbedPage.Children>
      <local:MainInfo IconImageSource="{img:ImageResource
Culent.Images.icon_descr.png}" BackgroundColor="{StaticResource
Key=backColor}"/>
      <local:Location IconImageSource="{img:ImageResource
Culent.Images.icon_location.png}" BackgroundColor="{StaticResource
Key=backColor}" />
   </TabbedPage.Children>
</TabbedPage>
```

A.10 — CardDetailedInfo.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent.Pages.NavigationPages
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
```

```
    public partial class CardDetailedInfo : TabbedPage
    {
        public CardDetailedInfo()
        {
            InitializeComponent();
        }
    }
}
```

A.11 — MainInfo.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Culent.Pages.NavigationPages.MainInfo"
             xmlns:img="clr-namespace:Culent.Classes;assembly=Culent">
    <ScrollView>
        <Grid Padding="20" RowSpacing="0" ColumnSpacing="0">
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
            </Grid.RowDefinitions>
            <Image Grid.Row="0"
                HeightRequest="200" WidthRequest="300"
                Source="{Binding Image}"
                VerticalOptions="FillAndExpand"
                HorizontalOptions="FillAndExpand"
                />
```

```
<Frame CornerRadius="20" HasShadow="True"
       Grid.Row="1"
       BackgroundColor="{StaticResource Key=cardColor}"
      HorizontalOptions="FillAndExpand"
      Margin="10">
   <StackLayout>
       <Label
      Text="Date of event:"
      HorizontalOptions="Start"
      VerticalOptions="Center"
      FontAttributes="Bold"></Label>
       <Label
      Text="{Binding Date, StringFormat='{0:dd.MM.yyyy HH:mm}'}"
      HorizontalOptions="Start"
      VerticalOptions="Center"></Label>
       <Label
      Text="Adrress:"
      HorizontalOptions="Start"
      VerticalOptions="Center"
      FontAttributes="Bold"></Label>
       <Label
      Text="{Binding Address}"
      HorizontalOptions="Start"
      VerticalOptions="Center"></Label>
   </StackLayout>
</Frame>

<Frame CornerRadius="20" HasShadow="True"
```

```
                Grid.Row="2" Grid.RowSpan="2"
                BackgroundColor="{StaticResource Key=cardColor}">
              <Label
                Text="{Binding Description}"
                HorizontalOptions="CenterAndExpand"
                Margin="20"
                LineHeight="1.5"/>
          </Frame>
        </Grid>
    </ScrollView>
</ContentPage>
```

A.12 — MainInfo.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent.Pages.NavigationPages
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainInfo : ContentPage
    {
        public MainInfo()
        {
```

```
            InitializeComponent();
        }
    }
}
```

A.13 — Location.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Culent.Pages.NavigationPages.Location"
             xmlns:maps="clr-
namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
             xmlns:img="clr-namespace:Culent.Classes;assembly=Culent">
    <ContentPage.Content>
        <StackLayout Padding="20">
            <Frame VerticalOptions="Start" HorizontalOptions="CenterAndExpand"
                   CornerRadius="20" HasShadow="True"
                   BackgroundColor="{StaticResource Key=cardColor}">
                <Label
                Text="Address: address template"></Label>
            </Frame>
            <maps:Map x:Name="map"
              IsShowingUser="True"
              MoveToLastRegionOnLayoutChange="False">
               <x:Arguments>
                  <maps:MapSpan>
                     <x:Arguments>
                        <maps:Position>
                           <x:Arguments>
                              <x:Double>50.45072581718367</x:Double>
```

```
                        <x:Double>30.52296904067067</x:Double>
                    </x:Arguments>
                </maps:Position>
                <x:Double>0.01</x:Double>
                <x:Double>0.01</x:Double>
            </x:Arguments>
        </maps:MapSpan>
    </x:Arguments>
    <maps:Map.Pins>
        <maps:Pin Label="{Binding Title}"
            Address="{Binding Date, StringFormat='{0:MM/dd/yyyy HH:mm}'}"
            Type="Place">
            <maps:Pin.Position>
                <maps:Position>
                    <x:Arguments>
                        <x:Double>50.45072581718367</x:Double>
                        <x:Double>30.52296904067067</x:Double>
                    </x:Arguments>
                </maps:Position>
            </maps:Pin.Position>
        </maps:Pin>
    </maps:Map.Pins>
</maps:Map>
</StackLayout>
</ContentPage.Content>
</ContentPage>
```

A.14 — Location.xaml.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent.Pages.NavigationPages
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class Location : ContentPage
   {
      public Location()
      {
         InitializeComponent();
      }
   }
}
```

A.15 — DatePopup.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<xct:Popup xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:xct="clr-
namespace:Xamarin.CommunityToolkit.UI.Views;assembly=Xamarin.CommunityT
oolkit"
        Size="250, 200"
        x:Class="Culent.Pages.NavigationPages.DatePopup"
        IsLightDismissEnabled="True">
```

```
   <StackLayout VerticalOptions="FillAndExpand"
HorizontalOptions="FillAndExpand" Padding="0" Margin="0">
       <Frame CornerRadius="15"
            HorizontalOptions="FillAndExpand"
            VerticalOptions="FillAndExpand"
            BackgroundColor="{StaticResource Key=cardColor}">
         <StackLayout>
            <Label Text="Choose date and time" HorizontalOptions="Center"
VerticalOptions="Start" FontAttributes="Bold" FontSize="19" />
            <DatePicker x:Name="popD" HorizontalOptions="Center"
VerticalOptions="Start"></DatePicker>
            <TimePicker x:Name="popT" Format="HH:mm"
HorizontalOptions="Center" VerticalOptions="Start"></TimePicker>
            <Button CornerRadius="10" BackgroundColor="{StaticResource
Key=navColor}" TextColor="{StaticResource Key=cardColor}"
                  Text="Apply" HorizontalOptions="Center" VerticalOptions="Start"
Clicked="Button_Clicked"></Button>
         </StackLayout>
       </Frame>
   </StackLayout>
</xct:Popup>
```

A.16 — DatePopup.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.CommunityToolkit.UI.Views;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
```

```
namespace Culent.Pages.NavigationPages
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class DatePopup : Popup
    {
        public DatePopup()
        {
            InitializeComponent();
        }

        private void Button_Clicked(object sender, EventArgs args)
        {
            DateTime dateTime = popD.Date;
            dateTime = dateTime.Add(popT.Time);
            Dismiss(dateTime);
        }
    }
}
```

A.17 — Event.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Culent.Classes.Models
{
    public class Event
    {
```

```
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; }
        public string Address { get; set; }
        public string Image { get; set; }
        public decimal Latitude { get; set; }
        public decimal Longitude { get; set; }
        public DateTime NotifDate { get; set; }
    }
}
```

A.18 — EventService.cs

```
using Culent.Classes.Models;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;

namespace Culent.Classes
{
    public class EventService
    {
        const string Url = "http://192.168.0.103:3000/api/event/";

        JsonSerializerOptions options = new JsonSerializerOptions
        {
```

```
            PropertyNameCaseInsensitive = true,
        };

        private HttpClient GetClient()
        {
            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return client;
        }

        public async Task<IEnumerable<Event>> Get()
        {
            HttpClient client = GetClient();
            string result = await client.GetStringAsync(Url);
            return JsonSerializer.Deserialize<IEnumerable<Event>>(result, options);
        }
    }
}
```

A.19 — ApplicationViewModel.cs

```
using Culent.Classes.Models;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Culent.Classes
{
    public class ApplicationViewModel
    {
        public static ObservableCollection<Event> Events { get; set; }
        public static ObservableCollection<Event> FavouriteEvents { get; set; }
        EventService eventsService = new EventService();

        static ApplicationViewModel()
        {
            Events = new ObservableCollection<Event>();
            FavouriteEvents = new ObservableCollection<Event>();
        }


        public async Task GetEvents()
        {
            IEnumerable<Event> events = await eventsService.Get();

            while (Events.Any())
                Events.RemoveAt(Events.Count - 1);


            foreach (Event f in events)
                Events.Add(f);
        }
    }
```

A.20 — ImageResourceExtension.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Culent.Classes
{
   [ContentProperty("Source")]
   public class ImageResourceExtension : IMarkupExtension
   {
      public string Source { get; set; }

      public object ProvideValue(IServiceProvider serviceProvider)
      {
         if (Source == null)
         {
            return null;
         }
         var imageSource = ImageSource.FromResource(Source);

         return imageSource;
      }
   }
}
```

А.21 — INotificationManager.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Culent.Classes
{
   public interface INotificationManager
   {
      event EventHandler NotificationReceived;
      void Initialize();
      void SendNotification(string title, string message, DateTime? notifyTime =
null);
      void ReceiveNotification(string title, string message);
   }
}
```

ДОДАТОК Б

Б.1 — MainActivity.cs

```
using System;

using Android.App;
using Android.Content.PM;
using Android.Content;
using Android.Runtime;
using Android.OS;
using Xamarin.Forms;
using Culent.Classes;
using Android;
```

```
namespace Culent.Droid
{
    [Activity(Label = "Culent", Icon = "@mipmap/icon", Theme =
"@style/MainTheme", MainLauncher = true, ConfigurationChanges =
ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode |
ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize, LaunchMode =
LaunchMode.SingleTop)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        const int RequestLocationId = 0;
        readonly string[] LocationPermissions =
        {
            Manifest.Permission.AccessCoarseLocation,
            Manifest.Permission.AccessFineLocation
        };
        protected override void OnStart()
        {
            base.OnStart();
            if((int)Build.VERSION.SdkInt >= 23)
            {
                if(CheckSelfPermission(Manifest.Permission.AccessFineLocation) !=
Permission.Granted)
                {
                    RequestPermissions(LocationPermissions, RequestLocationId);
                }
            }
        }
        protected override void OnCreate(Bundle savedInstanceState)
```

```
        {
            base.OnCreate(savedInstanceState);

            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
            Xamarin.FormsMaps.Init(this, savedInstanceState);
            LoadApplication(new App());
            CreateNotificationFromIntent(Intent);
        }
        protected override void OnNewIntent(Intent intent)
        {
            CreateNotificationFromIntent(intent);
        }

        void CreateNotificationFromIntent(Intent intent)
        {
            if (intent?.Extras != null)
            {
                string title = intent.GetStringExtra(AndroidNotificationManager.TitleKey);
                string message =
intent.GetStringExtra(AndroidNotificationManager.MessageKey);

DependencyService.Get<INotificationManager>().ReceiveNotification(title,
message);
            }
        }
        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
            if(requestCode == RequestLocationId)
```

```
        {
            if((grantResults.Length == 1) && (grantResults[0] ==
(int)Permission.Granted))
            {
                //smth documentation
            }
            else
            {
                //writeline a message
            }
        }
        else
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);
        }
        base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
    }
  }
}
```

Б.2 — AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using Android.App;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
```

```
[assembly: AssemblyTitle("Culent.Android")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Culent.Android")]
[assembly: AssemblyCopyright("Copyright ©  2014")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(false)]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]

// Add some common permissions, these can be removed if not needed
[assembly: UsesPermission(Android.Manifest.Permission.Internet)]
[assembly: UsesPermission(Android.Manifest.Permission.WriteExternalStorage)]
```

Б.3 — AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1" android:versionName="1.0"
package="com.companyname.culent">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="31" />
```

```
   <application android:label="Culent.Android"
android:theme="@style/MainTheme">
     <meta-data android:name="com.google.android.geo.API_KEY"
         android:value="API KEY" />
     <uses-library android:name="org.apache.http.legacy" android:required="false"/>
   </application>
   <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
</manifest>
```

Б.4 — AndroidNotificationManager.cs

```
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Xamarin.Forms;
using AndroidX.Core.App;
using AndroidApp = Android.App.Application;
using Culent.Pages;
```

```
using Android.Graphics;
using Culent.Classes;

[assembly: Dependency(typeof(Culent.Droid.AndroidNotificationManager))]
namespace Culent.Droid
{
    public class AndroidNotificationManager : INotificationManager
    {
        const string channelId = "default";
        const string channelName = "Default";
        const string channelDescription = "The default channel for notifications.";

        public const string TitleKey = "title";
        public const string MessageKey = "message";

        bool channelInitialized = false;
        int messageId = 0;
        int pendingIntentId = 0;

        NotificationManager manager;

        public event EventHandler NotificationReceived;

        public static AndroidNotificationManager Instance { get; private set; }

        public AndroidNotificationManager() => Initialize();

        public void Initialize()
```

```
    {
        if (Instance == null)
        {
            CreateNotificationChannel();
            Instance = this;
        }
    }

    public void SendNotification(string title, string message, DateTime? notifyTime
= null)
    {
        if (!channelInitialized)
        {
            CreateNotificationChannel();
        }

        if (notifyTime != null)
        {
            Intent intent = new Intent(AndroidApp.Context, typeof(AlarmHandler));
            intent.PutExtra(TitleKey, title);
            intent.PutExtra(MessageKey, message);

            PendingIntent pendingIntent =
PendingIntent.GetBroadcast(AndroidApp.Context, pendingIntentId++, intent,
PendingIntentFlags.CancelCurrent);
            long triggerTime = GetNotifyTime(notifyTime.Value);
            AlarmManager alarmManager =
AndroidApp.Context.GetSystemService(Context.AlarmService) as AlarmManager;
            alarmManager.Set(AlarmType.RtcWakeup, triggerTime, pendingIntent);
        }
```

```
        else
        {
            Show(title, message);
        }
    }

    public void ReceiveNotification(string title, string message)
    {
        var args = new NotificationEventArgs()
        {
            Title = title,
            Message = message,
        };
        NotificationReceived?.Invoke(null, args);
    }

    public void Show(string title, string message)
    {
        Intent intent = new Intent(AndroidApp.Context, typeof(MainActivity));
        intent.PutExtra(TitleKey, title);
        intent.PutExtra(MessageKey, message);

        PendingIntent pendingIntent =
PendingIntent.GetActivity(AndroidApp.Context, pendingIntentId++, intent,
PendingIntentFlags.UpdateCurrent);

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(AndroidApp.Context, channelId)
            .SetContentIntent(pendingIntent)
```

```
            .SetContentTitle(title)
            .SetContentText(message)

.SetLargeIcon(BitmapFactory.DecodeResource(AndroidApp.Context.Resources,
Resource.Mipmap.icon_round))
            .SetSmallIcon(Resource.Mipmap.icon_round)
            .SetDefaults((int)NotificationDefaults.Sound |
(int)NotificationDefaults.Vibrate);

        Notification notification = builder.Build();
        manager.Notify(messageId++, notification);
    }

    void CreateNotificationChannel()
    {
        manager =
(NotificationManager)AndroidApp.Context.GetSystemService(AndroidApp.Notificat
ionService);

        if (Build.VERSION.SdkInt >= BuildVersionCodes.O)
        {
            var channelNameJava = new Java.Lang.String(channelName);
            var channel = new NotificationChannel(channelId, channelNameJava,
NotificationImportance.Default)
            {
                Description = channelDescription
            };
            manager.CreateNotificationChannel(channel);
        }
```

```
            channelInitialized = true;
        }

        long GetNotifyTime(DateTime notifyTime)
        {
            DateTime utcTime = TimeZoneInfo.ConvertTimeToUtc(notifyTime);
            double epochDiff = (new DateTime(1970, 1, 1) -
DateTime.MinValue).TotalSeconds;
            long utcAlarmTime = utcTime.AddSeconds(-epochDiff).Ticks / 10000;
            return utcAlarmTime;
        }
    }
    [BroadcastReceiver(Enabled = true, Label = "Local Notifications Broadcast
Receiver")]
    public class AlarmHandler : BroadcastReceiver
    {
        public override void OnReceive(Context context, Intent intent)
        {
            if (intent?.Extras != null)
            {
                string title = intent.GetStringExtra(AndroidNotificationManager.TitleKey);
                string message =
intent.GetStringExtra(AndroidNotificationManager.MessageKey);

                AndroidNotificationManager manager =
AndroidNotificationManager.Instance ?? new AndroidNotificationManager();
                manager.Show(title, message);
            }
        }
    }
```

```
}
```

ДОДАТОК В

В.1 — Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CulentApi
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
```

```
}
```

B.2 — Startup.cs

```
using CulentApi.Model;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CulentApi
{
   public class Startup
   {
      // This method gets called by the runtime. Use this method to add services to the container.
      // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
      public void ConfigureServices(IServiceCollection services)
      {
         services.AddControllers();


         string con = "Server=(localdb)\\mssqllocaldb;Database=eventsdb1;Trusted_Connection=True;";
         services.AddDbContext<EventContext>(options => options.UseSqlServer(con));
```

```
        }

        // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

B.3 — EventController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using CulentApi.Model;
namespace CulentApi.Controller
```

```
{
    [Route("api/[controller]")]
    [ApiController]
    public class EventController : ControllerBase
    {
        EventContext db;
        public EventController(EventContext context)
        {
            db = context;
        }

        // GET api/event/
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Event>>> Get()
        {
            return await db.Events.ToListAsync();
        }

        // GET api/event/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Event>> Get(int id)
        {
            Event eventS = await db.Events.FirstOrDefaultAsync(x => x.Id == id);
            if (eventS == null)
                return NotFound();
            return new ObjectResult(eventS);
        }
```

```
    }
}
```

B.4 — EventContext.cs

```
using Microsoft.EntityFrameworkCore;
namespace CulentApi.Model
{
    public class EventContext : DbContext
    {
        public DbSet<Event> Events { get; set; }

        public EventContext(DbContextOptions<EventContext> options)
            : base(options)
        {
            //Database.EnsureDeleted();
            Database.EnsureCreated();
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Event>().HasData(
                new Event[]
                {
                new Event {
                    Id=1,
                    Name="Шопен у саду",
                    Description="Запрошуємо вас на чарівний вечір у саду. " +
                    "Для вас прозвучить найромантичніша і найкрасивіша фортепіанна музика в історії. " +
                    "Прекрасні Ноктюрні романтика Шопена, занурюючись у які можна забути про все, крім любові та краси.",
```

```
                Date = new System.DateTime(2022, 7, 10, 19, 0, 0),
                Address = "Ботанічний сад ім. ГришкаКиїв, вул. Тимирязівська, 1",
                Image = "https://iili.io/heKUu4.jpg",
                Latitude = 50.41506214087533M,
                Longitude = 30.56279749808595M,
                NotifDate = System.DateTime.Now
                },
            new Event {
                Id=2,
                Name="ХАММЕРTIME",
                Description="Це ХаммерTime! Безкровні тіла призначені для сумних могил… " +
                "Ім’я та Заслуги уникають смерті! " +
                "В 16й день липня в братському колі вшануємо легендарного Воїна Тараса \"Хаммера\" Бобанича.",
                Date = new System.DateTime(2022, 7, 16, 17, 0, 0),
                Address = "Bel etage Київ, вул. Шота Руставелі, 16А",
                Image = "https://iili.io/heKkas.png",
                Latitude = 50.439328392241755M,
                Longitude = 30.519921240416657M,
                NotifDate = System.DateTime.Now
                },
            new Event {
                Id=3,
                Name="Стендап на Воздвиженці",
                Description="«STAND UP PRO» продовжує підтримувати моральний дух українців! " +
                "«Стендап на Воздвиженці» - це чудова нагода підтримати власний моральний дух " +
```

```
            "в компанії популярних українських коміків. Концерт рекомендовано для глядачів старше 18 років.",
            Date = new System.DateTime(2022, 8, 16, 18, 0, 0),
            Address = "32 Jazz ClubКиїв, ул. Воздвиженская, 32",
            Image = "https://iili.io/heKeFn.jpg",
            Latitude = 50.460123397531646M,
            Longitude = 30.511452526922707M,
            NotifDate = System.DateTime.Now
            },
        new Event {
            Id=4,
            Name="KADNAY",
            Description="«Kadnay у Києві! Чекаємо на вас 2022-07-01 в 18:00 на локації Bel etage, Київ. " +
            "Ви можете придбати квитки на Kadnay на Concert.ua онлайн.",
            Date = new System.DateTime(2022, 7, 1, 18, 0, 0),
            Address = "Bel etage Київ, вул. Шота Руставелі, 16А",
            Image = "https://iili.io/heKN6X.jpg",
            Latitude = 50.439328392241755M,
            Longitude = 30.519921240416657M,
            NotifDate = System.DateTime.Now
            },
        new Event {
            Id=5,
            Name="Ваня Якимов - читання",
            Description="Ваня Якимов - Читання у Києві! " +
            "Чекаємо на вас 2022-07-09 в 17:00 на локації Будинок архітектора, Київ. " +
            "Ви можете придбати квитки на Ваня Якимов - Читання на Concert.ua онлайн.",
```

```
                Date = new System.DateTime(2022, 7, 9, 17, 0, 0),
                Address = "Будинок архітектора Київ, вул. Бориса Грінченка, 7",
                Image = "https://iili.io/heKv8G.jpg",
                Latitude = 50.44953252394175M,
                Longitude = 30.520498255797598M,
                NotifDate = System.DateTime.Now
                },
            new Event {
                Id=6,
                Name="Французький Джаз у затишному дворику",
                Description="Французький джаз у затишному дворику у Києві! " +
                "У затишному дворику у самому серці міста відбудеться прозвучить легендарний французський " +
                "джаз у виконанні тріо APASIONADO (випускники Університету Мистецтв м. Грац, Австрія).",
                Date = new System.DateTime(2022, 8, 18, 17, 0, 0),
                Address = "Дворик МЦКМ (Жовтневий палац) Київ, Алея Героїв Небесної Сотні, 1",
                Image = "https://iili.io/heKS9f.jpg",
                Latitude = 50.4498199166991M,
                Longitude = 30.5276518502335M,
                NotifDate = System.DateTime.Now
                },
            });
        }
    }
}
```

B.5 — Event.cs

```
using System;
```

```
namespace CulentApi.Model
{
    public class Event
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; }
        public string Address { get; set; }
        public string Image { get; set; }
        public decimal Latitude { get; set; }
        public decimal Longitude { get; set; }
        public DateTime NotifDate { get; set; }
    }
}
```

B.6 — launchSettings.json

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:51191",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
```

```
    "commandName": "IISExpress",
    "launchBrowser": true,
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "CulentApi": {
    "commandName": "Project",
    "dotnetRunMessages": "true",
    "launchBrowser": true,
    "applicationUrl": "http://localhost:5000",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
 }
}
```