Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Мобільний додаток для обліку щоденних та щотижневих задач
Назва теми
з додаванням тегів та з можливістю створення резервної копії

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПІПЗ.170104.01.04.ПЗ

Виконав студент IV курсу група ІПЗ-17-1 _____ А. І. Дьоміна
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ І. В. Гурман
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Радельчук
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

4 06 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
05 02 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Дьоміній Анастасії Іванівні
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії

Керівник проекту (роботи) Гурман Іван Васильович
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
кандидат технічних наук, доцент

Затверджена наказом ректора університету від 05.02.2021 р. № 11
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2021 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування мобільного додатку
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди, 14 шт.)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Радельчук Г.І., доцент кафедри ІПЗ		
Антиплагіат	Гурман І.В., доцент кафедри ІПЗ		

7. Дата видачі завдання «05» лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем ДП	01.12– 30.12.2020	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2021	
3 Проектування програмного забезпечення	01.02 – 28.02 2021	
4 Програмна реалізація	01.03 – 10.04.2021	
5 Тестування програмного забезпечення	11.04 – 30.04.2021	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2021	
7 Попередній захист ДП	Травень 2021 (згідно графіка)	
8 Перевірка ДП на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2021	
9 Підготовка до захисту та захист ДП	з 01.06.2021	

Студент _____ А.І. Дьоміна
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Гурман І.В.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту «Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії».

Автор проекту: Дьоміна Анастасія Іванівна.

Керівник проекту: Гурман Іван Васильович.

Пояснювальна записка: 125 с., 36 рис., 14 табл., 3 дод., 19 джерел.

Графічна частина: 14 презентаційних слайдів.

ЩОДЕННИК, ЗАПИСНИК, МОБІЛЬНИЙ ДОДАТОК, БАЗА ДАНИХ, ТЕГ, РЕЗЕРВНА КОПІЯ.

Мета дипломного проекту: розробка мобільного додатку для платформи Android, для вирішення проблем тайм-менеджменту, правильної організації задач та записів, а також зручного пошуку та зберігання попередніх записів.

У дипломному проекті виконано аналіз різних методів ведення обліку щоденних та щотижневих задач та особистих записів, а також існуючих рішень на ринку мобільних додатків. Визначено функціональні задачі та розроблено ТЗ. Проведено аналіз існуючих інструментів та технологій розробки Android-додатків та спроектовано архітектуру застосунку та унікальний функціональний інтерфейс. На основі яких розроблено додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії. Після розробки було проведено модульне, функціональне та конфігураційне тестування готового програмного продукту

Впровадження розробленого програмного продукту дозволяє швидко та зручно організовувати свої задачі та записи, а також сприяє легкому пошуку та зберіганню попередніх записів.

1.06.2021 р.
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПІПЗ.170104.01.04.ПЗ	Пояснювальна записка	12		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	14		

Змн.	Арк.	№ докум.	Підпис	Дата	ДПІПЗ.1700104.01.04.ВД			
Виконав		Дьоміна А. І.		1.06	Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії Відомість документів	Літ.	Арк.	Аркушів
Керівник		Гурман І. В.		2.06			1	1
Н. контр.		Радельчук Г. І.		3.06		ХНУ, ІПЗ-17-1		
Зав. каф.		Бедратюк Л.П.		4.06				

ЗМІСТ

					ДПІПЗ.1700104.01.04.ВД			
Змн.	Арк.	№ докум.	Підпис	Дата				
Виконав		Дьоміна А. І.		1.06	Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії. Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		Гурман І. В.		2.06			4	125
						ХНУ, ІПЗ-17-1		
Н. контр.		Радельчук Г. І.		3.06				
Зав. каф.		Бедратюк Л.П.		4.06				

ВСТУП

Проблеми управління часом, організації справ, часу та впорядкування думок є належать до списку найактуальніших проблем сучасності. Особливо складно з ними впоратись творчим особистостям та тим людям, чиї руки не постигають за їх думками. І саме тоді, коли людина мрійливо літає серед хмар, або намагається втримати в голові весь свій список справ, часто, хоч і ненароком стається так, що забувається щось дуже важливе. Або ж ще гірша ситуація – це коли всі справи на сьогодні готові, і наче все чудово, але завтра є така справа, до якої потрібно було підготувати щось заздалегідь. І людина стрімголов перевертає все шкереберть, аби без запізнення зробити все необхідне.

Здається, ми живемо в епоху технологій, коли вся існуюча інформація в світі завжди у нас в кишені та під рукою – у нашому мобільному телефоні, і на будь-яке побажання чи тему знайдуться десятки, сотні, а іноді і тисячі програм у AppStore та Play Market. А для спрощення організації часу і задач створено більше сотні застосунків, деякі з них встановлено за замовчуванням, інші ми завантажуємо на смартфон самостійно, але всі вони однотипні, і допомагають вирішити проблему тільки частково, тому що в головному вікні додатків відображаються або звичайні списки існуючих задач, або завдання на один єдиний день.

Саме тому велика кількість людей сьогодні користуються звичайними щоденниками та записниками з готовою або зробленою власноруч розміткою, адже вони хочуть мати повну картину своїх актуальних задач та планів на день чи тиждень. Але і у такого рішення існують певні недоліки, адже паперові носії недовговічні, та мають здатність псуватись та закінчуватись в самий непідходящий момент. Також їхня недовговічність та скінченність спричиняє інші недоліки, наприклад - неможливість постійно мати при собі всю картину своїх минулих та майбутніх задач, планів чи думок. Значним недоліком для багатьох також є те, що сторінки паперового щоденника «не вибачають

помилок», а це означає, що будь-яке перекреслення, виправлення чи помилка псують естетику зовнішнього вигляду записів та породжують деяку плутанину в записах. Тому мною було прийняте рішення створити застосунок, який об’єднає переваги звичайного паперового щоденника і можливості, які надає смартфон.

Актуальність обраної теми полягає у тому, що існує велика кількість застосунків що мають майже однаковий функціонал, та не надають зручне та ефективне вирішення вищевказаних проблем, а отже існує необхідність у створенні мобільного додатку, що запропонує користувачам новий зручний функціонал, який зможе задовольнити необхідність користувачів в організації часу та своїх думок.

Мета проекту – розробити додаток для мобільних пристроїв з операційною системою Android, який допоможе вирішити проблему тайм-менеджменту, правильної організації завдань та записів користувачів, а також зручного пошуку і зберігання минулих та майбутніх записів.

Для досягнення мети проекту потрібно виконати наступні завдання:

– провести дослідження предметної області, визначити особливості та специфіку процесу організації часу та створення списків задач;

– виконати детальний аналіз існуючих рішень;

– визначити функціональні задачі;

– розробити технічне завдання;

– провести аналіз інструментів та технологій, що використовуються для створення Android-додатків;

– проаналізувати та порівняти різні методи для запису та збереження інформації;

– розробити архітектуру додатку;

– розробити сучасний та функціональний інтерфейс;

– виконати програмну реалізацію мобільного застосунку;

– провести тестування готового програмного продукту.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Мобільний додаток – це програмне забезпечення розроблене для використання на мобільних пристроях, таких як смартфон, планшет чи смарт-годинник. Існує багато методів класифікації мобільних застосунків але найчастіше використовується розподіл мобільних програм на три типи: нативні, мережеві та гібридні.

Нативні застосунки – це програми, що створені для певної мобільної платформи. Створення додатку такого типу дозволяє застосовувати унікальні функції операційної системи, забезпечують кращу продуктивність та досвід взаємодії користувача з системою. Подібним чином, великою перевагою нативних додатків є те, що в порівнянні з іншими типами, нативні застосунки надають розробникам найбільше різноманітних можливостей, наприклад: широкий доступ до прикладних програмних інтерфейсів (API), можливість проводити низькорівневий контроль девайсів і значно менша, в порівнянні з іншими типами додатків кількість проблем та багів.

Основним недоліком нативних програм є те, що вони не можуть охопити велику частину потенційних користувачів, тому що застосунок створений для пристроїв, з ОС Android не може працювати на смартфоні, що використовує iOS.

Мережеві додатки створюються з використанням звичайних веб-технологій HTML, CSS та JavaScript. Для повноцінної роботи такої програми необхідний постійний доступ до інтернет мережі. Характерним для цього типу програм є те що більшість користувацьких даних зберігаються в хмарному сховищі. Ефективність роботи таких мобільних веб-застосунків схожа до швидкодії звичайних веб-додатків, що використовуються у браузері, і часто є значно нижчою ніж у аналогічного додатку нативного типу.

Гібридні застосунки – це поєднання ідеї роботи нативних і мережевих додатків. Ця категорія включає застосунки розроблені з використанням фреймворків Xamarin, React Native, Apache Cordova та подібних до них. Програми такого типу розробляються для використання веб та нативних технологій на багатьох платформах одразу. Їх перевага полягає в тому, що розробка таких додатків відбувається швидше та легше, оскільки вони забезпечують використання однакового коду, що працює на декількох мобільних ОС. Тим не менше, гібридні застосунки мають достатню кількість недоліків у порівнянні з нативними програмами, а саме: нижчу швидкодію, гіршу стабільність та різний вигляд інтерфейсу при роботі на смартфонах з різними операційними системами.

Зваживши всі переваги і недоліки різних типів мобільних застосунків я обрала нативний тип додатку, адже він забезпечує найкращу швидкодію застосунку, надає найбільшу кількість робочих інструментів, повністю функціонує без підключення до інтернету та гарантує найкоректніше виконання програми з найменшою кількістю багів..

Оскільки нативні застосунки призначені для створення програми тільки для однієї операційної системи, тому її варто вибрати.

Операційна система (ОС) – це програмне забезпечення, що реалізує зв'язок між прикладними програмами й апаратними засобами комп'ютера, яке забезпечує зручність використання комп'ютерної системи, а також її ефективність і роботу [2]. Наявність операційної системи – головна особливість, що відрізняє смартфон від звичайного мобільного телефону.

Найпоширеніші мобільні платформи на поточний момент є:

- Android;
- IOS;
- Kai OS;
- Windows 10 Mobile (не підтримується з 14 січня 2020);
- Blackberry OS (остання версія випущена в травні 2018).

Платформа Android це програмний стек операційної системи на основі Linux, призначений для керування пристроєм (апаратурою), пам'яттю та процесами. Система є відкритою та надає розробнику майже повну свободу дії і при цьому забезпечує багатозадачність і ізольованість процесів один від одного. На теперішній момент Android – найпоширеніша та найпопулярніша мобільна платформа в світі. З недоліків цієї системи можна відмітити, те, що її надмірна відкритість приводить до підвищених ризиків безпеки, а велика кількість пристроїв в модельному ряді, який підтримується системою, поступово негативно впливає на її стабільність на окремих пристроях.

iOS – це мобільна ОС, створена та розроблена компанією Apple спеціально для своїх пристроїв. Спочатку вона розроблялась тільки для мобільних пристроїв iPhone, пізніше iOS було дороблено та адаптовано для використання на інших пристроях компанії, таких як iPad, iPod Touch та Apple TV. Основними перевагами системи вважається її швидкодія, стабільна робота та безпека пристроїв та даних. Всі ці переваги вдалось впровадити через дуже сильну закритість системи та вузьку модельну лінійку пристроїв, на яких використовується система. Проте її закритість спричиняє деяку кількість проблем, які можна віднести до недоліків системи, наприклад:

– незважаючи на те, що додатки загалом захищені від зміни обладнання, розробнику все одно необхідно враховувати різницю між гаджетами при написанні коду, до прикладу, у деяких гаджетів є камера, а у деяких її немає;

– відсутність зручного та швидкого обміну даними між різними пристроями, через значну закритість системи;

– майже відсутність можливостей персоналізації, через зацикленість розробників на швидкодії смартфону.

KaiOS – мобільна ОС, створена на базі ядра Linux, що «поєднує потужність та переваги смартфонів з доступністю фічерфонів» і спеціально розроблена для кнопкових стільникових телефонів. Є гілкою B2G (Boot to Gecko), що розвивається ентузіастами простого послідовника операційної системи Firefox

OS, розробку якої припинено Mozilla в 2016 році. Цю мобільну операційну систему можна назвати народною, по причині її некомерційного розвитку.

При розробці нативного додатку головним недоліком вибору цієї операційної системи буде те, що вона застосовується на дуже невеликій кількості пристроїв, а тому кількість користувачів невеликою.

Також нераціонально створювати нативний додаток для Windows 10 Mobile та Blackberry OS, оскільки ці операційні системи на теперішній час, вже не підтримуються розробниками.

Тому залишається обрати тільки між операційними системами iOS та Android. Порівнявши всі переваги та недоліки цих мобільних ОС я зробила висновок, що кращим вибором буде розробка нативного додатку для ОС Android, адже вона надає величезну кількість можливостей для розробника, і є найпопулярнішою з усіх розглянутих вище мобільних платформ, а отже має найбільше користувачів.

Наступним кроком, після того, як обрано тип додатку та операційну систему для якої буде вестись розробка, необхідно дізнатись все про записники та щоденники, а також про різні методи ведення записів.

Сьогодні в більшій частині випадків люди заводять щоденники та записники для двох цілей: для тайм-менеджменту і організації планів або ж , в іншому випадку для впорядкування якихось своїх думок та ідей.

Люди, які заводять особистий щоденник для тайм-менеджменту в більшості використовують продатовані блокноти з надрукованою розміткою. Проте буває так, що знайти ідеальний записник стає завданням високої складності, тому в такому випадку людина створює власну розмітку та графить датування самостійно в пустому записнику, переважно для таких цілей використовується блокнот в крапку, або взагалі без розмітки.

Творчі люди, навпаки ж переважно користуються щоденником для запису і впорядкування своїх творчих ідей або просто думок. Часто вони стверджують, що ведення щоденника допомагає розібратись в своїх думках та собі, а інколи

щоденник і зовсім заміняє людині сеанс у психолога. Зрозуміло, що для ведення таких записів зовсім нераціонально використовувати датовані блокноти, адже у них обмежено місце для запису за один день, та й іноді буває, що за день зовсім не трапилось нічого, що було б варто записати до щоденника, а отже значна частина аркуша в датованому щоденнику залишиться пустою. В такому випадку зручніше і раціональніше використати пустий записник, без зайвих елементів.

Певним гібридним компромісом для людей, які не хочуть обмежувати використання щоденника тільки тайм-менеджментом, списками завдань чи тільки записами особистих думок є , так звана, система організації ведення особистих записів – bullet journal.

Bullet journal – це метод персональної організації розроблений та описаний дизайнером Райдером Керроллом на його особистому сайті [3], а потім і у книзі [4]. Запропоноване рішення дозволяє поєднувати та організовувати планування, списки справ, свої ідеї та інші задачі в один записник. Назва bullet journal походить від використання маркування крапками, які допомагають позначати різні типи записів в щоденнику. Ця система є чудовим компромісом для всіх хто не зміг зупинити вибір на якомусь певному стилі ведення щоденника. Також вона дозволяє наочно бачити свої поточні, майбутні та минулі задачі, що є чудовим рішенням проблем описаних у вступі. Єдиним значним недоліком цієї системи можна відмітити необхідність використання паперового блокноту.

Зважуючи вище написане можна зробити висновок, що правильним рішенням буде розробка додатку, який підходить для ведення записів за системою bullet journal, але в той же час є незалежним від неї. Тому, на мою думку, необхідно в додатку створити можливість виокремлення головного тексту, куди потрібно записувати важливі задачі чи думки, які завжди будуть доступними та помітними користувачу на головній сторінці тижня. А також виділити місце для запису певних спостережень, переживань, чи можливо додаткових пояснень до справ, які буде видно тільки після натискання на задачі певного дня.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Головною платформою для поширення застосунків для гаджетів з операційною системою Android на теперішній час є Play Market – офіційний додаток від Google, який зазвичай є попередньо встановлений на смартфони з ОС Android. Тому для проведення аналізу ринку, скористаємося рекомендаціями, які запропонує Play Market. Після введення відповідних запитів можна виділити наступний список програм, які є найпопулярнішими серед користувачів:

- My Diary;
- Universum;
- Pi Journal;
- SplenDO;
- MyNotes;
- Diary;
- Notes.

Далі проведемо детальний аналіз вищеперерахованих додатків, зважаючи на їхню вартість, функціональні можливості, переваги та недоліки, з врахуванням відгуків користувачів в Play Market та особистого досвіду використання.

У застосунка My Diary на його сторінці в Play Market [5] розробником описано такі функціональні особливості:

- можливість встановлення паролю;
- зміна теми інтерфейсу;
- щоденник настрою;
- додавання зображень в записи;
- можливість додавання тегів;
- синхронізація з хмарним сховищем та створення бекапу в ньому;
- можливість експорту запису у форматі .txt та .pdf.

Інтерфейс програми на перший погляд та до використання здається мінімалістичним, але після використання складається враження, що він занадто строкатий, перевантажений функціоналом та має надмірну кількість рекламних вставок (рисунок 1.1).

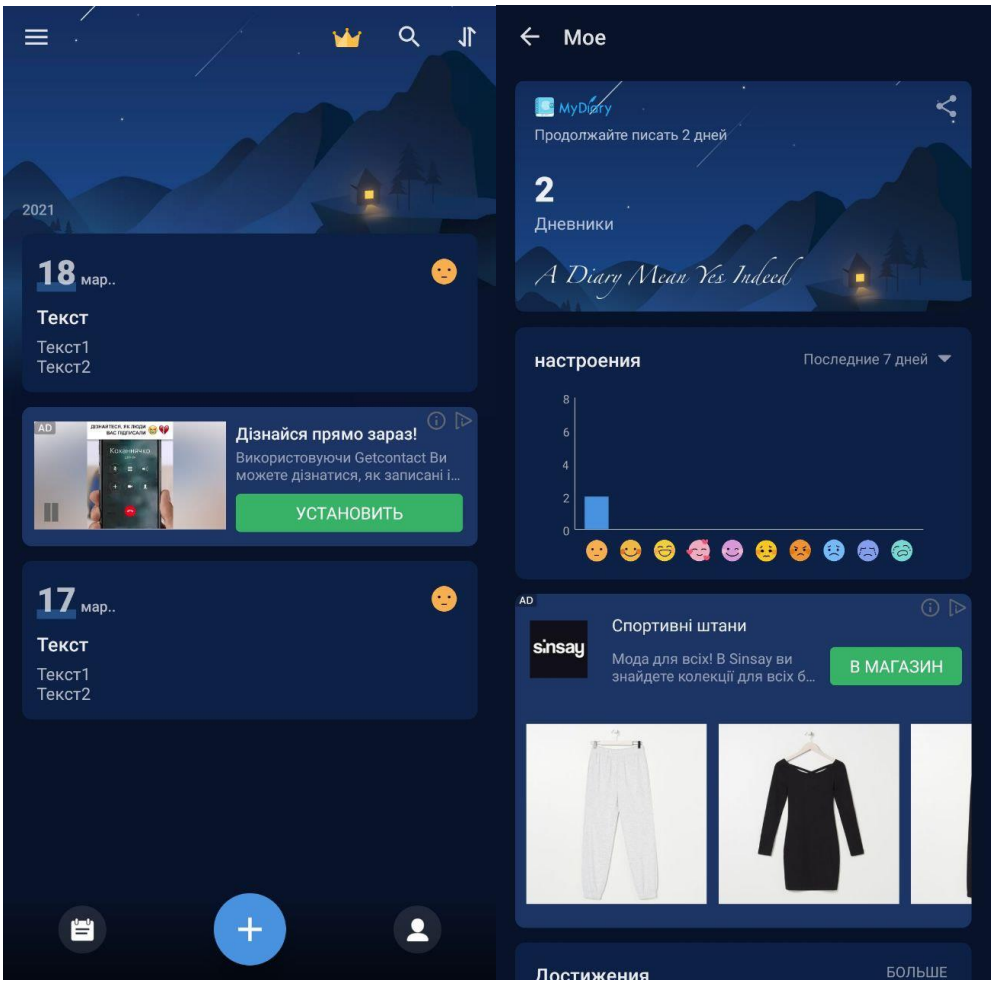


Рисунок 1.1 – інтерфейс додатку My Diary

Додаток працює на смартфонах з версією ОС 5.0 та вище. Продукт умовно-безкоштовний, тобто завантажити додаток можна безкоштовно, але для повнофункціонального використання необхідно купувати підписку, також без підписки в додатку занадто багато реклами яка заважає, забирає час та відволікає.

Синхронізація та створення бекапу можливе тільки за умови придбання підписки та використання Google Drive або Dropbox, що є не самим безпечним методом зберігання даних, адже зловмисники можуть легко отримати особисті

записи користувача. Тому краще створювати систему резервного копіювання для щоденнику, з можливістю збереження резервної копії без використання мережі Інтернет, і далі сам користувач має обирати чи зберігати резервну копію на поточному пристрої, чи завантажувати свій бекап до хмарного сховища, чи перенести його на інший фізичний носій.

Наступним розглянемо додаток Universum. Відповідно до інформації про додаток [6] у Play Market: Universum – це універсальний особистий щоденник для запису думок, ідей, фотографій, фінансів і завдань.

Розробник пропонує використовувати його застосунок як: журнал подорожей, журнал відслідковування дієти, класичний особистий щоденник, фінансовий щоденник, звичайний блокнот і багато інших варіантів. З поміж всіх функціональних можливостей програми розробник виділяє наступні:

– захист паролем;
– автоматичне резервне копіювання;
– вивантаження даних в PDF або CSV для подальшого редагування в Excel;
– пошук за всіма типами записів;
– створення хештегів;
– можливість створювати періодичні завдання;
– можливість отримувати нагадування;
– різні теми оформлення;
– ненав’язливий дизайн.

Як і попередній розглянутий додаток, програма доступна для смартфонів з версією ОС 5.0 та новіших версій, і також є умовно-безкоштовним. На відміну від попередника, інтерфейс (рисунок 1.2) і справді приємний та мінімалістичний, не перевантажений непотрібною інформацією та є достатньо зрозумілим для використання новим користувачем. В безкоштовній версії розглянутого застосунку також наявна реклама, але вона менш активна і докучлива, займає меншу площу екрану.

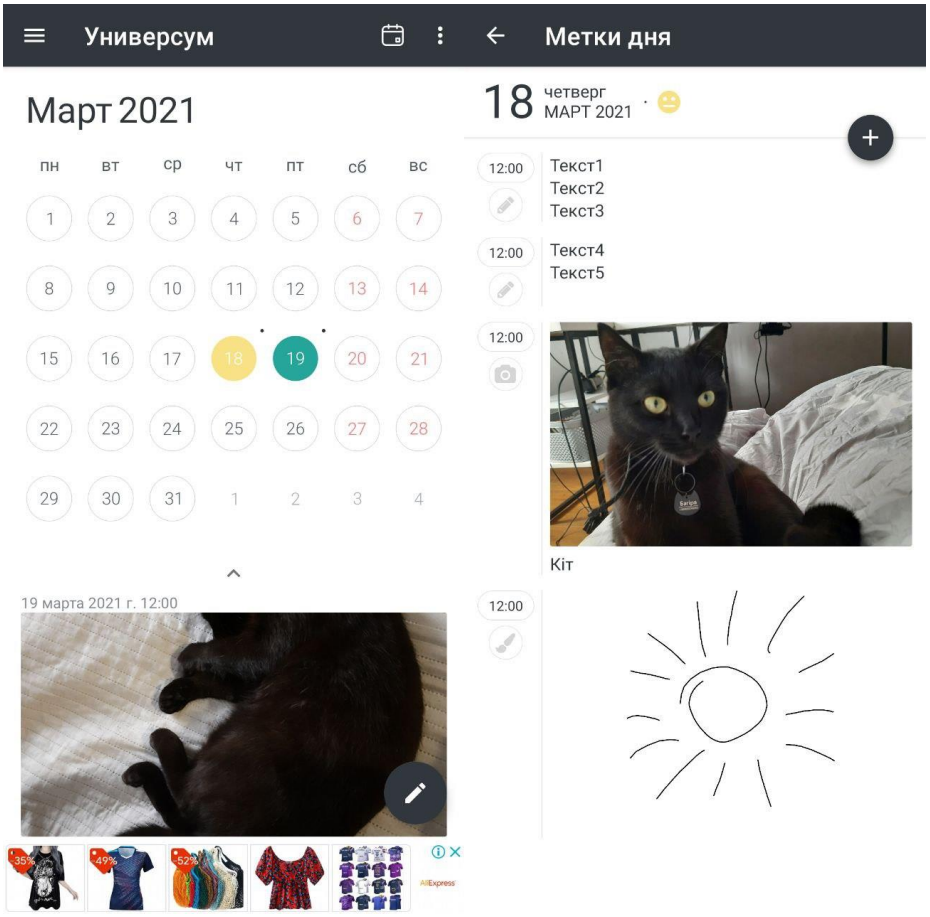


Рисунок 1.2 – Інтерфейс додатку Universum

Вагомим недоліком є те, що переважна більшість функціоналу, про який заявляє автор додатку доступна тільки в платно. Одна із найважливіших особливостей – автоматичне створення резервної копії – також наявна тільки в платній версії, і аналогічно попередньо розглянутій програмі працює тільки за наявності інтернет-з'єднання. А створення бекапу ще і з заміток з медіа файлами необхідно додатково мати місце в хмарному сховищі. Також сам редактор тексту примітивний і не надає можливості певним чином виокремлювати чи виділяти текст. Функція створення малюнків, адже нема мінімального набору інструментів, такого як ластик, ні жодного іншого, окрім звичайного олівця.

З переваг цього застосунку можна відмітити наявність україномовного інтерфейсу і цікавий функціонал, який виділяє програму серед інших.

Наступним розглянемо додаток Pi Journal. Розробник описує цей додаток у Play Market [7] наступним чином: Pi Journal – це дивовижний спосіб вести

щоденник самообслуговування з відповідним посібником по глибині розуму, додаток зробить ваше життя більш пам’ятним та незабутнім. Також на сторінці [7] виділено наступні особливості цього застосунку:

– можливість встановлення паролю;
– можливість записувати примітки;
– можливість додавати фотографії до записів;
– наявність різних тем інтерфейсу;
– трекер настрою;
– аналіз психічного здоров’я;
– журнал вдячності;
– наявність підказок, що допомагають знайти тему для письма.

Інтерфейс додатку (рисунок 1.3) використовує в спокійні кольори, що чудово підходять один до одного, і приємні оку, але у світлій темі оформлення в певних місцях не видно текст на світлому тлі.

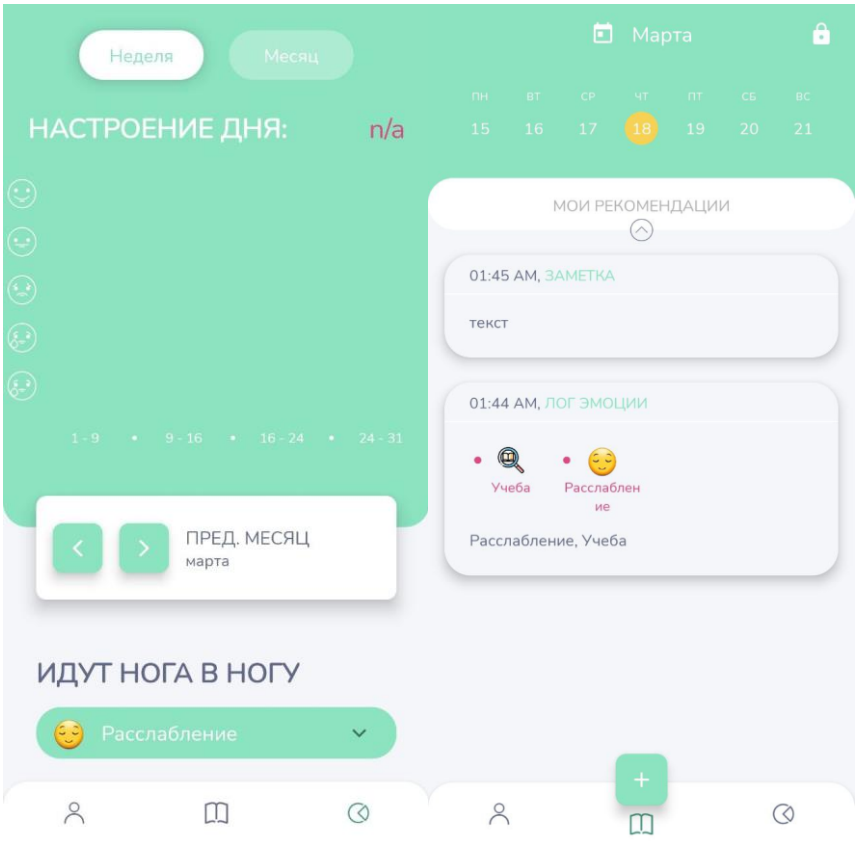


Рисунок 1.3 – Інтерфейс додатку Pi Journal

Також цікавим є позиціонування додатку розробником, а саме – щоденник, який допоможе відслідковувати та покращувати свій емоційний стан. Ця особливість є унікальною та вирізняє продукт серед однотипних додатків на ринку. Хоча й здається, що сьогодні всі щоденники мають функцію відслідковування настрою, проте вони не аналізують та не обробляють ці дані, максимум на що вони здатні – вивести кількісний того, який був настрій за певний період часу. У розглянутого додатку, інший підхід, він допомагає користувачу проаналізувати причини виникнення поганого настрою, та допомогти їх позбутись. При початку роботи з додатком користувачу задаються ознайомчі питання, які допомагають персоналізувати поради, що надаються у вигляді голосового повідомлення (рисунок 1.4).

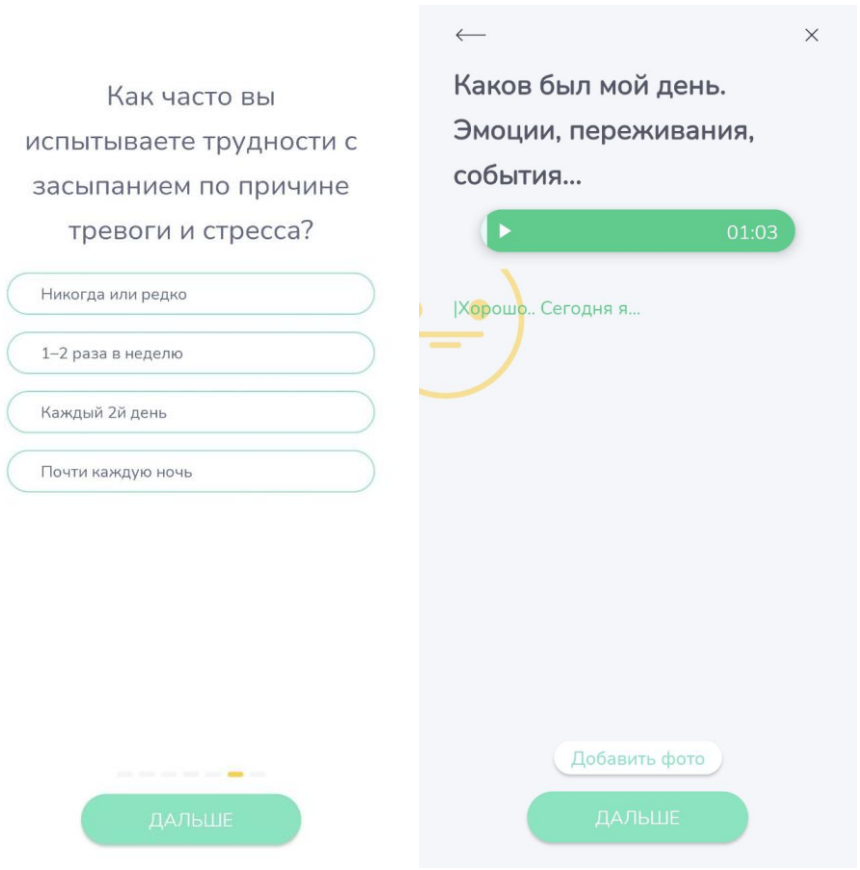


Рисунок 1.4 – Приклади уточнюючих питань та голосових записів

Недоліком можна назвати неправильну анімацію для перегортання сторінок, а саме – перегортання представлено невеликим зміщенням екрану.

Подібна анімація зазвичай використовується для позначення неможливості перегорнути сторінку, а не для самого перегортання, і через те, що вміст сторінки змінюється з деякою затримкою, зрозуміти чи сторінка перегорнулась не можливо. Це суттєво знижує зручність використання додатку.

Подібно до вище розглянутих продуктів, Pi Journal доступний для смартфонів з версією ОС Android 5.0 та новіших версій, і розповсюджується за умовно-безкоштовною моделлю. Проте на відміну від попередньо розглянутих, у безкоштовній версії додатку нема реклами, а преміум-версія тільки доповнює існуючий функціонал, що безперечно є перевагою.

Наступним розглянемо додаток SplenDO. На сторінці у Play Market [8] розробник описав його як «розумний список завдань для повсякденного використання», та виділив такі основні характеристики:

- зручне управління завданнями;
- завдання групуються в зручних списках завдань;
- віджети на головному екрані показують відразу, що робити;
- інтелектуальні повідомлення, в зазначений час;
- синхронізація з Google;
- голосове створення завдань;
- повторювані завдання;
- завдання без дати;
- застосовування дій до групи завдань.

Інтерфейс застосунку SplenDO (рисунок 1.5) є пустим і незавершеним, а кольорова гамма застарілою. Ймовірно при розробці автор додатку не зміг грамотно застосувати принципи UI/UX проектування Незважаючи на свою пустоту інтерфейс, він переповнений непропорційно великими рекламними банерами. При створенні завдань така реклама закриває значну частину робочого простору та потрібну інформацію. Для того щоб позбутись надокучливої реклами як і у розглянутих вище програм потрібно купити преміум-версію додатку. Також у програми відсутнє резервне копіювання завдань.

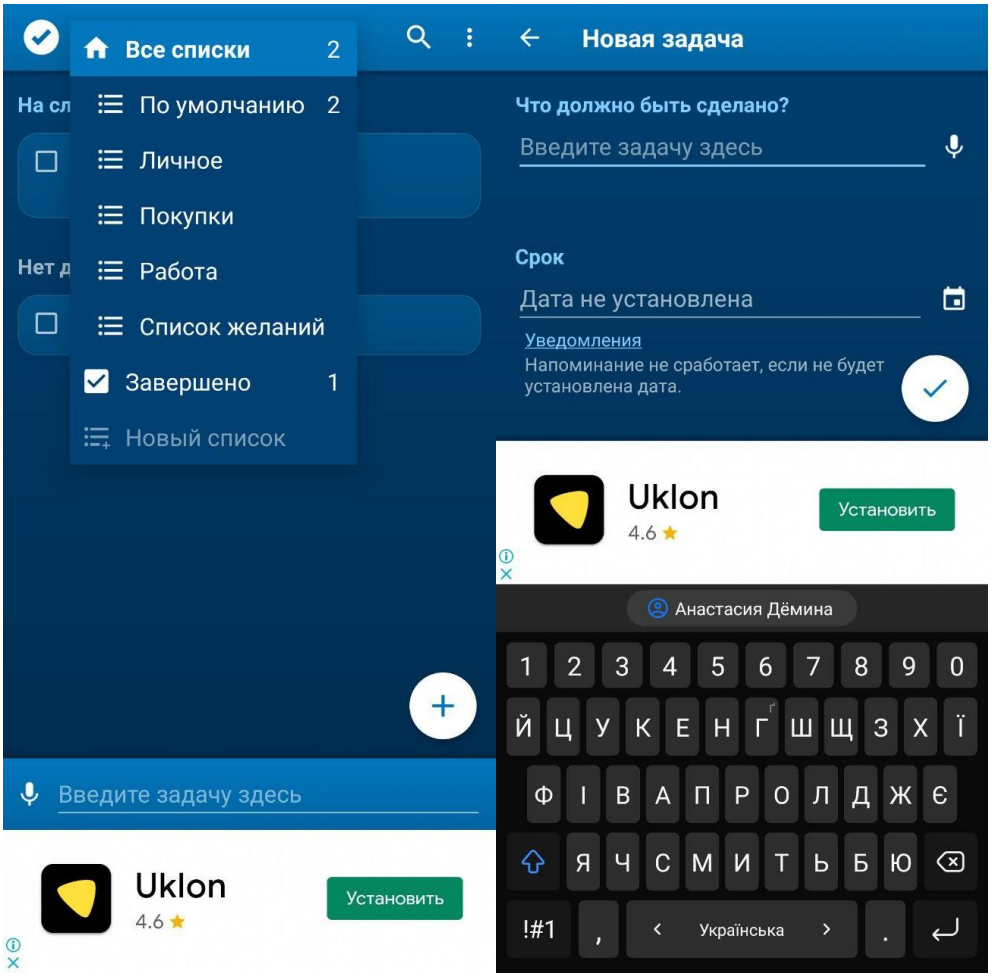


Рисунок 1.5 – Інтерфейс додатку SplenDO

На відміну від інших розглянутих застосунків, SplenDO доступний для смартфонів з версією Android 4.1 та новіших версій, що дозволяє охопити більше користувачів, проте накладає обмеження на можливий функціонал.

З переваг застосунку можна виділити наявність україномовного інтерфейсу та голосового вводу, а також те, що виконані завдання не зникають, а залишаються з поміткою "зроблено", як відмічають користувачі – це важливо з психологічної точки зору.

Самим відчутним недоліком додатку є те, що він сильно садить заряд батареї пристрою, за відгуками користувачів: «раніше заряду батареї вистачало на день, то з програмою SplenDO вистачає тільки до обіду». Також до недоліків можна записати відсутність можливості робити резервну копію хоч якось переносити нотатки на інший пристрій.

Перейдемо до аналізу застосунку MyNotes. Ось так описано додаток в Play Market [9]: «MyNotes – це простий в використанні, інтуїтивно зрозумілий, швидкий елегантний та безпечний блокнот з хмарною синхронізацією». Розробник вділяє в застосунку такі ключові можливості:

- блокування програми (Пароль або PIN-код + Відбиток пальця);
- збереження, перегляд, пошук та поширення нотаток;
- впорядкування нотаток за папками;
- сортування записів за датою створення, редагування та назві;
- додавання нагадувань;
- керування папками;
- навігація між записами в горизонтальному положенні;
- керування резервними копіями;
- ручна синхронізація нотаток за допомогою Google Drive;
- збереження в хмарі;
- збереження і відображення тисячі записів без втрати швидкодії;
- збереження великих записів;
- темна тема;
- колір теми;
- віджети;
- українська мова.

А також такі преміум-функції, що наявні тільки в платній версії:

- відсутність реклами;
- автоматична синхронізація;
- попередній перегляд резервної копії;
- експортування (Текстовий файл і HTML).

Застосунок є мінімалістичним, з доволі обмеженим функціоналом. У програмі зовсім не передбачено ніяких можливостей форматування тексту, що є значним недоліком. Також відсутня можливість додавання медіа-контенту.

Інтерфейс (рисунок 1.6) виконаний у стилі додатків від Google.

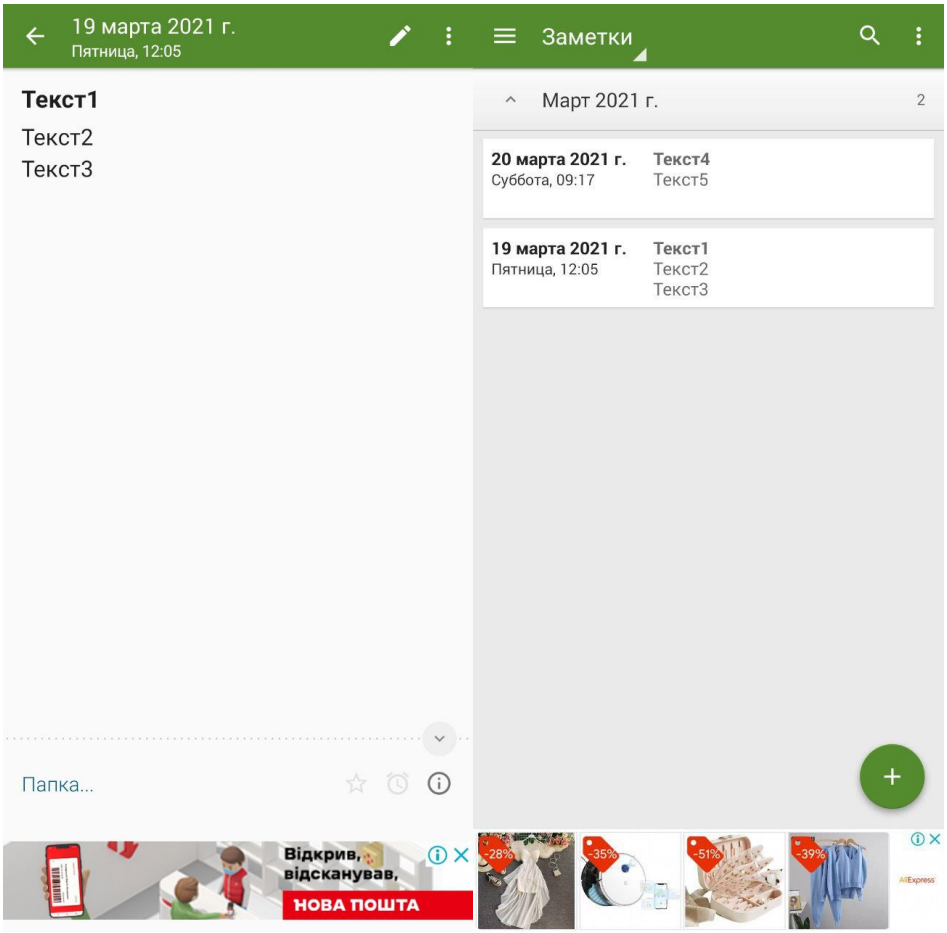


Рисунок 1.6 – Інтерфейс MyNotes

Додаток працює на ОС Android 4.0.3 і новіших версій. Реклама в безкоштовній версії займає невелику частину екрану і практично не заважає використанню основного функціоналу.

Далі розглянемо додаток Diary. Ось такий опис у нього на сторінці у Play Market [10]: «Diary – це захищений паролем особистий щоденник, який Ви можете використовувати для нотування гарних спогадів або особистих записів. Використовуйте прекрасні емоджі та смайли для вистежування Вашого настрою та зберігайте різноманітні спогади у вигляді світлин. Завдяки паролю Ваші нотатки та записи будуть надійно захищенні у застосунку. Проявіть індивідуальність в оформленні щоденника використовуючи теми та шрифти, які можна налаштувати.»

Також розробник виділяє наступний функціонал:

– захист даних паролем;

– різноманітні теми;
– синхронізація записів;
– резервне копіювання;
– можливість змінювати шрифти в записах;
– відстеження настрою;
– можливість прикріпити світлини;
– нагадування;
– експорт у форматі PDF;
– зручний пошук.

Інтерфейс програми(рисунок 1.7) мінімалістичний, приємний, дозволяє обрати кольорову схему та добавляти зображення до заміток, що формує візуальну частину інтерфейсу.

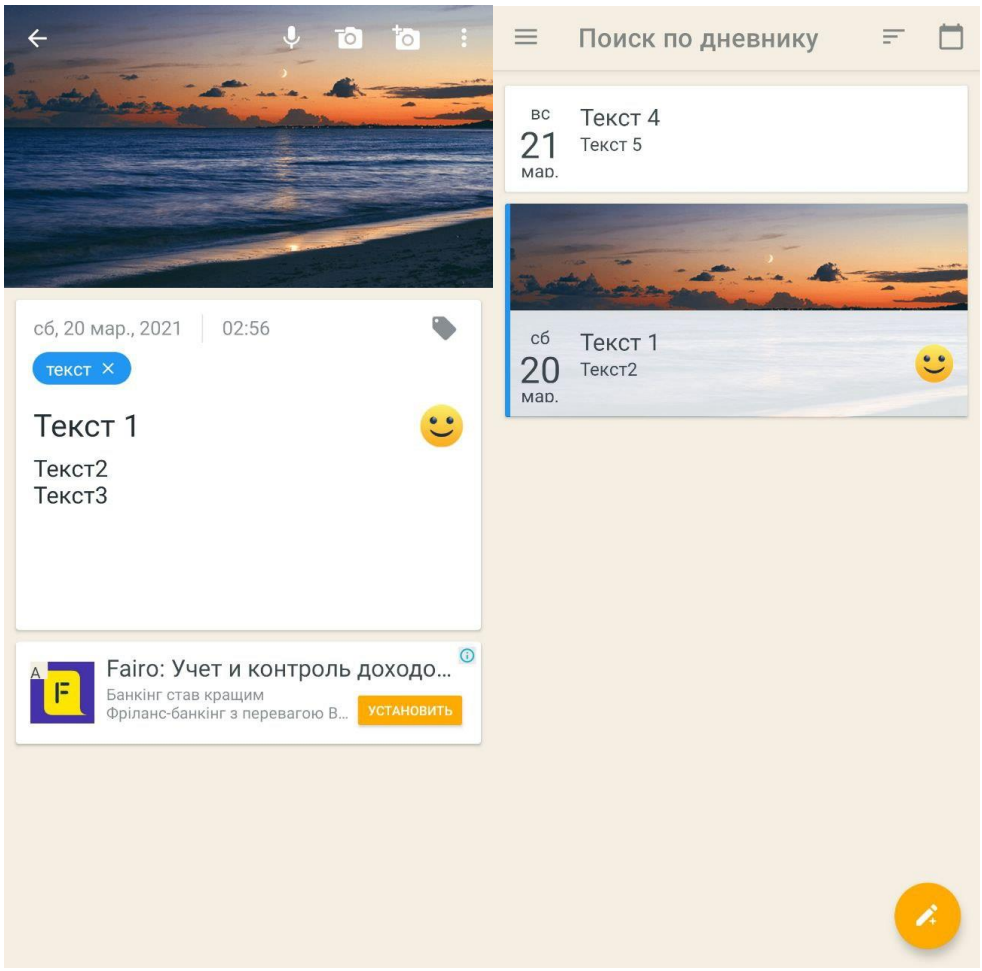


Рисунок 1.7 – Інтерфейс програми Diary

Реклама присутня тільки на одному з екранів, і не заважає при користуванні. Як і в інших додатках її можна прибрати купивши преміум-версію. Цікавим є те, як розробник вирішив продавати свій додаток, а саме – не обмежувати функціонал безкоштовної версії, а навпаки трохи розширити його в платній версії.

Як і більшість розглянутих програм, Diary доступний для пристроїв з версією 5.0 та вище. Також у застосунок має україномовну версію.

Останньою з комерційних прогам розглянемо Notes. Цей додаток найбільш подібний до попередньо встановлених програм для нотаток як за зовнішнім виглядом так і за можливостями, що він надає. На сторінці додатку в Play Market [11] заявлено наступний функціонал:

- створення звичайних заміток;
- створення TODO листів;
- прикріплення зображень;
- прикріплення аудіо-файлів та голосових нотаток;
- закріплення важливих записів у верху списку;
- створення нагадувань;
- додавання категорій, що позначаються кольором;
- фільтрування нотаток за кольором;
- переміщення нотаток у віджет;
- додавання паролю;
- звичайний пошук по тексту;
- створення нотаток з програм за допомогою кнопки «поділитись»;
- можливість створення секретних записів;
- можливість архівувати непотрібні записи;
- перенос записів в Google Drive.

Інтерфейс додатку (рисунок 1.8) простий, мінімалістичний, має просту кольорову схему та подібний до вбудованих програм для заміток. У Notes нема можливості змінити тему оформлення, чи головного кольору додатку. З мінусів

можна відмітити незручний спосіб відображення «важливих» заміток – відображення на сторінці з звичайними, і коли важливих записів стає багато, то нові створенні замітки губляться десь в низу списку і потрібно довго прогортувати сторінку, щоб їх знайти. Також недоліком є те, що у додатка відсутній україномовний переклад.

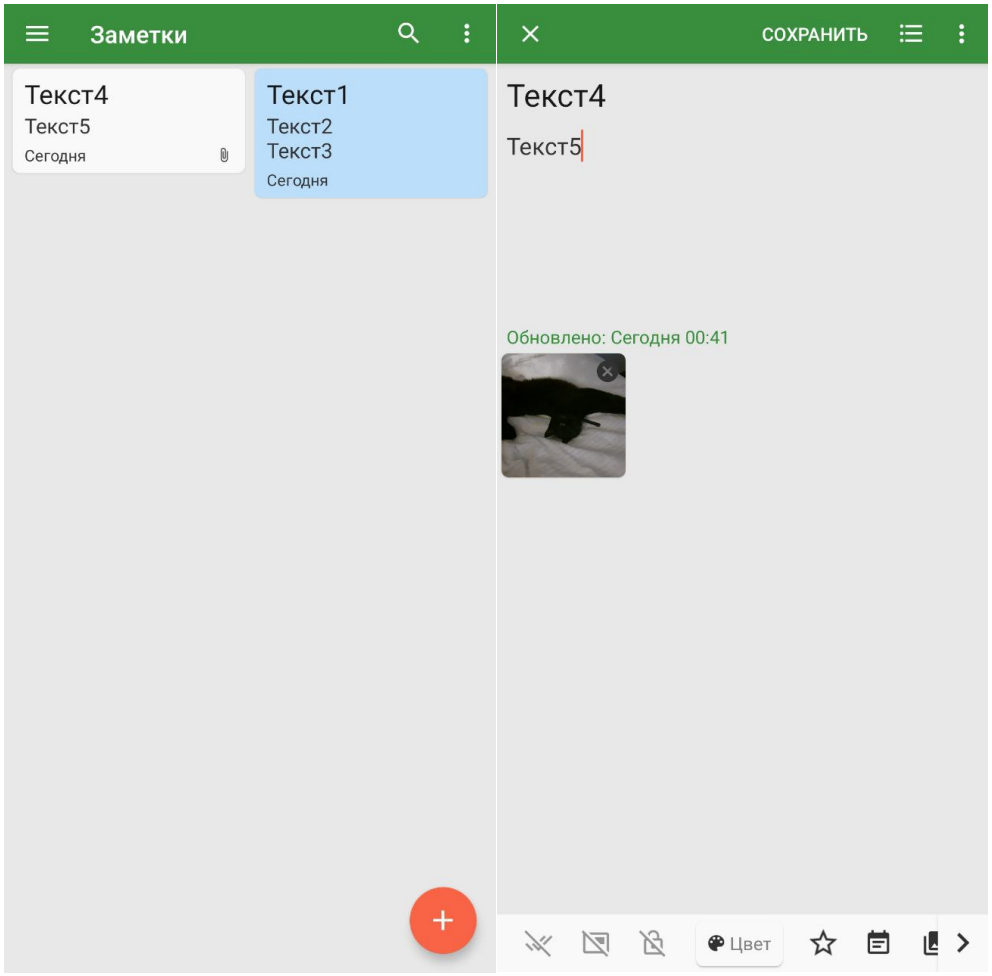


Рисунок 1.8 – Інтерфейс програми Notes

З переваг Notes можна відзначити відсутність реклами, малий розмір додатку та доступність для пристроїв з версією операційної системи від 4.2.

Окрім всіх комерційних застосунків зазвичай мають попит і програми для заміток, які попередньо встановлено на смартфони, але провести їх досконалий аналіз не видається нагоди, адже у кожного виробника гаджетів ці додатки відрізняються, і просто так встановити їх, відповідно до політики компаній не є

можливим. Проте можна виділити головні функціональні особливості, які притаманні програмам такого типу, а саме:

– створення списків з маркерами
– додавання зображень;
– поділ на категорії;
– можливість додавати малюнки;
– можливість позначати «обрані» записи;
– пошук по категоріям та звичайному тексту;
– можливість форматування тексту.

Весь цей функціонал можна побачити майже у всіх додатків для нотаток незалежно від марки смартфону. Для прикладу на рисунках 1.9 та 1.10 приведено інтерфейси вбудованих програм на смартфонах від «Samsung» та «Poco».

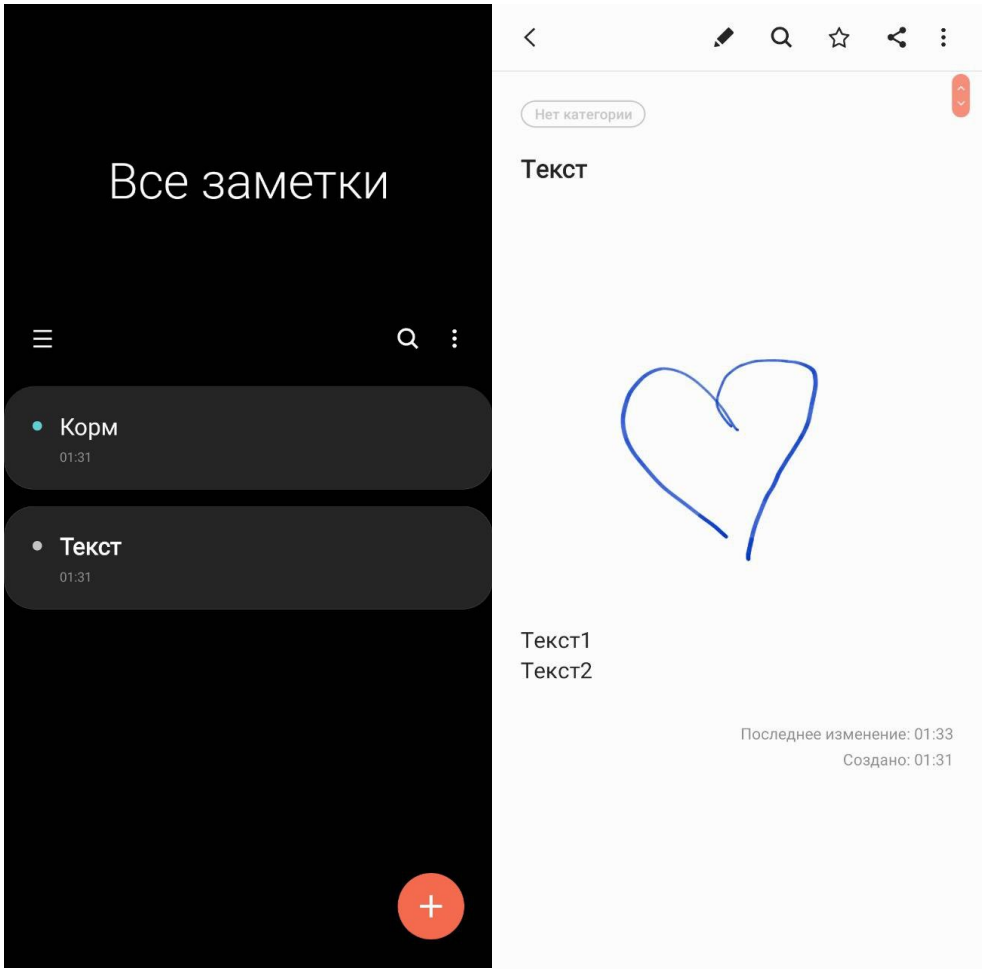


Рисунок 1.9 – Приклад інтерфейсу додатку для нотаток від «Samsung»

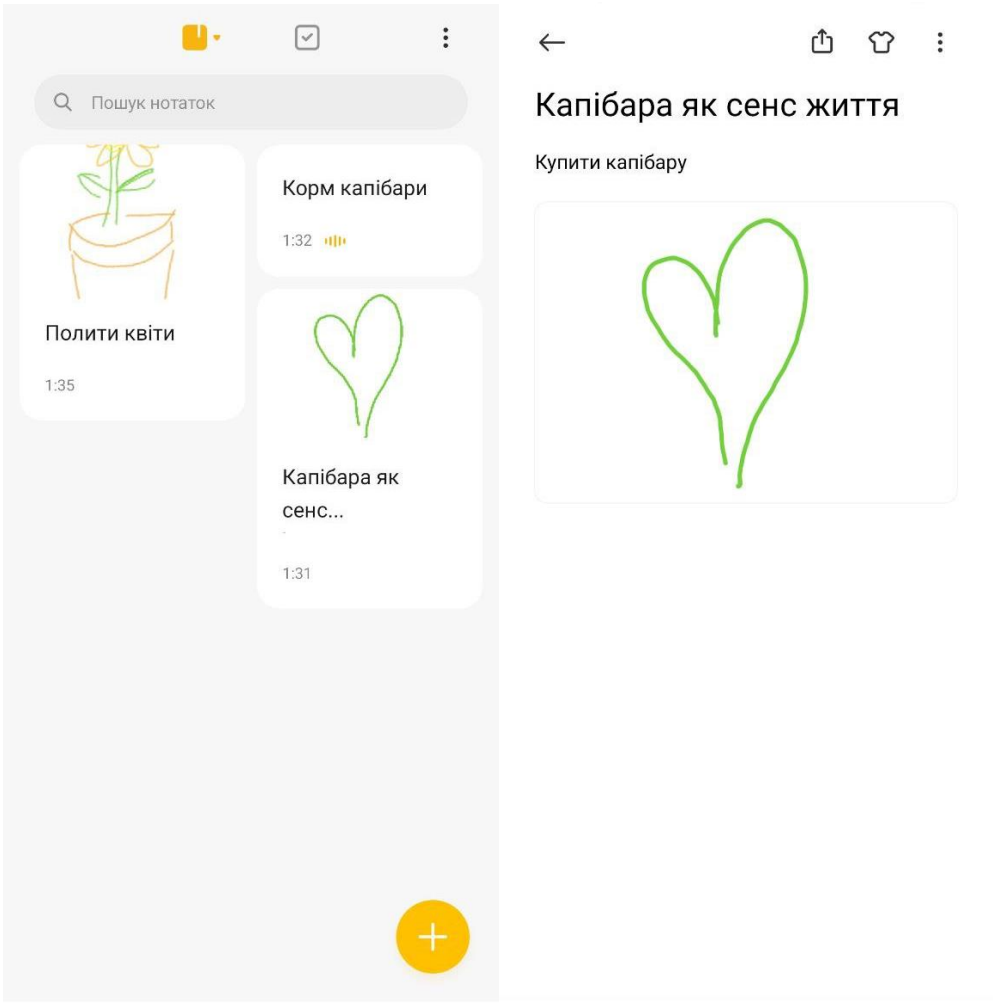


Рисунок 1.10 – Приклад інтерфейсу вбудованого додатку для нотаток на смартфоні «Poco »

Тепер можна провести порівняльний аналіз всіх розглянутих додатків за різними характеристиками. Так як основною особливістю додатків є робота з текстом, то важливим критерієм є те, чи надає додаток можливість форматувати текст. Далі важливо оцінити зручність інтерфейсу, можливості його кастомізації, і те, наскільки заважає реклама роботі з застосунком та способи відображення записів на головній сторінці додатку. Також важливо порівняти різні можливості збереження записів, адже особистий щоденник, це те, що не хочеться втратити. По тій же причині варто знати чи можливо захистити записи, за допомогою паролю, від перегляду чужаками. Також важливо чи є додаток платним. В таблиці 1.1 виконано порівняння додатків за перерахованими критеріями.

Таблиця 1.1 – Порівняльна таблиця додатків

Назва додатку	Засоби форматування тексту	Зручність інтерфейсу	Можливість кастомізації	Реклама	Відображення записів	Резервне копіювання та експорт	Пароль	Безкоштовний
1	2	3	4	5	6	7	8	9
My Diary	Обмежені в безкоштовній версії	4/10	Так, обмежена в безкоштовній версії	Дуже багато, займає значну площу	З прив'язкою до дати	Тільки в платній версії	Так	Ні
Universum	Відсутні	8/10	Зміна головного кольору	Помірно	За місяць	Архів в безкоштовній версії, експорт у PDF та хмару в платній	Так	Ні
Pi Journal	Відсутні	6/10	Відсутня	Помірно	За місяць, тиждень, день	Відсутня	Так	Ні
SplenDO	Відсутні	3/10	Відсутня	Дуже багато, займає значну площу	З прив'язкою до дати	Відсутня	Ні	Ні
MyNotes;	Відсутні	4/10	Відсутня	Помірно	З прив'язкою до дати	Ручна синхронізація через Google в безкоштовній версії, автоматична та експорт в платній	Так	Ні
Diary	Обмежені в безкоштовній версії	6/10	Так, обмежена в безкоштовній версії	Тільки на одному з екранів	З прив'язкою до дати	Ручна синхронізація через Google в безкоштовній версії, автоматична та експорт в платній	Так	Ні

Кінець таблиці 1.1

1	2	3	4	5	6	7	8	9
Notes	Доступно тільки в платній версії	5/10	Відсутня	Відсутня	Звичайні замітки	Експорт в файл та Google диск	Так	Ні
Нотатки «Samsung»	Хороші	7/10	Відсутня	Відсутня	З прив'яз-кою до дати	В хмару Samsung	Так	Так
Нотатки «Росо»	Хороші	6/10	Відсутня	Відсутня	З прив'яз-кою до дати	Відсутня	Ні	Так

Отже після вивчення ринку додатків для ведення щоденника, можна зробити висновок, що на даний момент переважна більшість продуктів, які існують на ринку мають функціонал направлений на створення звичайних заміток, або створення заміток з простою прив'язкою до дати. Також було виявлено, що майже всі додатки дозволяють створювати бекапи тільки після придбання платної версії, і при тому абсолютна більшість з них робить бекап в хмару, що є не самим безпечним методом для збереження таких інтимних речей, як особистий щоденник.

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Після аналізу предметної області та наявного програмного забезпечення було прийнято рішення про розробку нового додатку.

На відміну від наявних на ринку продуктів, розроблюваний додаток має унікальний спосіб відображення заміток. А саме – вони відображаються таким чином, щоб бачити всі замітки на поточний тиждень. Також можна переглядати задачі наступних та минулих тижнів. При тому замітка на кожен конкретний день

редагується окремо. В цю замітку можна написати будь яку кількість тексту і визначити який текст буде відображатись на сторінці тижня.

Також для зручності буде добавлено можливість пошуку записів по спеціальних мітках(тегах), не багато з існуючих на ринку продуктів, мають подібний функціонал.

Більшість конкурентних продуктів передбачають тільки платну можливість створення бекапу, що відразу відсіює значну частину потенційних користувачів, адже не всі мають бажання платити за можливість перенесення своїх заміток в інше місце. Інші додатки не мають можливості створити бекап не в хмарі, а просто на пристрої, а отже і не передбачають ніякого механізму для подальшого збереження його заміток і перенесення на інші пристрої. На відміну від інших, розроблюваний додаток буде мати переваги обох методів, обходячи їх недоліки. Можливість створення бекапу на пристрої та будь-якому носії дозволить відмовитись від необхідності створення облікового запису в хмарних сховищах, тобто додаток зможе використовувати неавторизований користувач, але при необхідності він зможе створити бекап, і завдяки ньому перенести свої записи на інші пристрої, або якщо необхідно – зберегти його у хмарі.

Враховуючи всі вищезазначені особливості, можна скласти перелік функціональних можливостей, які має надавати програма:

- перегляд головних задач тижня ;
- відкриття задачі кожного дня окремо;
- можливість написати для кожного дня текст, який не буде відображатись на сторінці тижня;
- перегляд старих записів;
- створення записів на перед;
- пошук записів за ключовими словами та тегами;
- при випадковому згортанні записані дані зберігаються;
- при виході з програми записані дані автоматично зберігаються;
- імпорт резервної копії;

– експорт даних в файл;

– створення резервної копії в пам’яті телефону.

Також було сформовано такі вимоги до інтерфейсу:

– стримані або базові кольори (сірий, білий, блакитний тощо);

– мінімалістичні іконки;

– при відкритті додатку має відображатись сторінка з замітками на дні поточного тижня;

– дні поточного тижня мають розміщуватись на головному екрані ергономічно та інтуїтивно зрозуміло (як приклад: розміщення днів тижня в шкільному щоденнику);

– при натисканні на день тижня має відкриватись сторінка обраного дня, на якій відображається головні задачі, та додаткове поле для тексту, який не виводиться на головну сторінку;

– на головній сторінці має бути перехід до сторінки для пошуку заміток по тегам та часині тексту замітки

– при перегортанні вправо-вліво на головній сторінці поточний тиждень має змінюватись на попередній та наступний відповідно.

На основі проведеного аналізу вимог до програмного забезпечення розроблено технічне завдання, яке подано у додатку А.

В цьому розділі було проведено детальний аналіз та огляд існуючих методів ведення щоденника та типів мобільних додатків, а також досліджено існуюче програмне забезпечення. Після чого було проведено аналіз вимог до розроблюваного додатку, на основі якого складено ТЗ.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКА

2.1 Архітектура та функціональна структура додатка

Архітектура системи – це фундаментальні поняття або властивості системи в її середовищі, що втілюються в її елементах, взаємозв'язках та в принципах її проектування та розвитку. Архітектура – це те, що є фундаментальним для системи – не обов'язково все, що пов'язано з нею, а тільки найважливіше. [12]

Проектування є одним з найважливіших етапів розробки програмного забезпечення, адже продумана архітектура додатку забезпечує легку розробку та подальшу підтримку розробленого ПЗ. Проте не існує такої архітектури яка буде завжди однозначно хорошою чи поганою, адже її оцінка є дуже суб'єктивним процесом, і сильно залежить від поставлених завдань. Тому той варіант архітектури, що є вдалим рішенням для одного проекту, може бути жахливим вибором для іншого.

Для полегшення процесу створення архітектури та вирішення поширених проблем при розробці програм прийнято використовувати певні шаблони проектування – патерни. Для їх застосування необхідно виділити основні частини та процеси програми.

На сьогоднішній день існують такі основні підходи до проектування архітектури мобільних додатків:

- Модель-Представлення-Контролер (MVC);
- Модель-Представлення-Представник (MVP);
- Модель-Представлення-Модель виду (MVVM);
- Представлення-Взаємодія-Ведучий-Сутність-Провідник (VIPER).

MVC – це класичний архітектурний шаблон, який застосовується при проектуванні мобільних застосунків. Основною його ідеєю є поділ системи на три частини, які взаємопов'язані між собою: модель даних, представлення (інтерфейс користувача) та контролер (модуль керування). Схема взаємодії даних в моделі MVC представлена на рисунку 2.1.

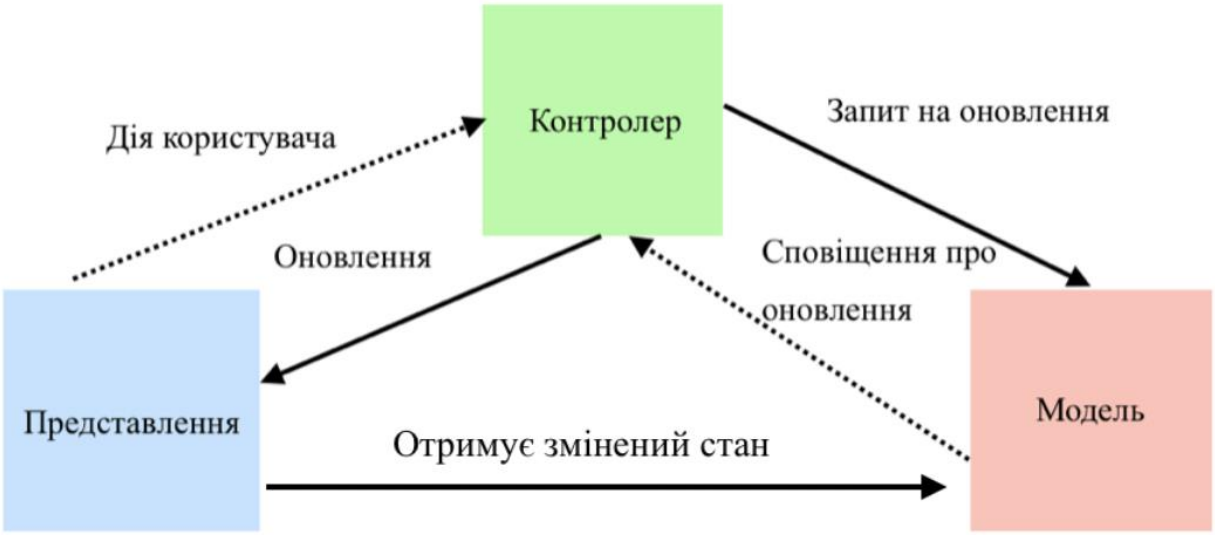


Рисунок 2.1 – Діаграма взаємодії між компонентами шаблону MVC

Модель (Model) – це головний компонент, який відповідає за зберігання, керування та структуру даних, які відображатимуться в представленні.

Представлення (View) – це візуальна проекція даних, які зберігаються в моделі, і відповідальна за відображення даних користувачу, тобто – являє собою користувацький інтерфейс.

Контролер (Controler) – це компонент, який пов'язує модель та представлення, та відповідає за обмін даними між системою та користувачем.

У MVC є багато переваг, основні з яких: легкість реалізації, швидкість розробки, непотрібність дублювання коду та висока швидкодія додатку. Проте є і один недолік – слабо розподілені обов'язки між компонентами призводять до проблем з масштабуванням проекту, адже більшу частину функціоналу реалізують у контролері.

MVP – це архітектурний шаблон, похідний від MVC, що відділяє поведінку роботи інтерфейсу, тобто події та його візуальне відображення у різні класи: Представник (Presenter) та Представлення (View) відповідно [13]. Виходить що представник та представлення розподіляють між собою роботу, що в MVC виконує контролер. Завдяки такому підході вирішується основна проблема, яка виникала в MVC – сильна зв'язність компонентів. Це відбувається завдяки інверсії контролю (Inversion of Control), яка дозволяє представнику, на відміну

від контролера, стати незалежним від представлення і навпаки взяти на себе частину роботи представлення. Принцип роботи MVP зображено на рисунку 2.2.

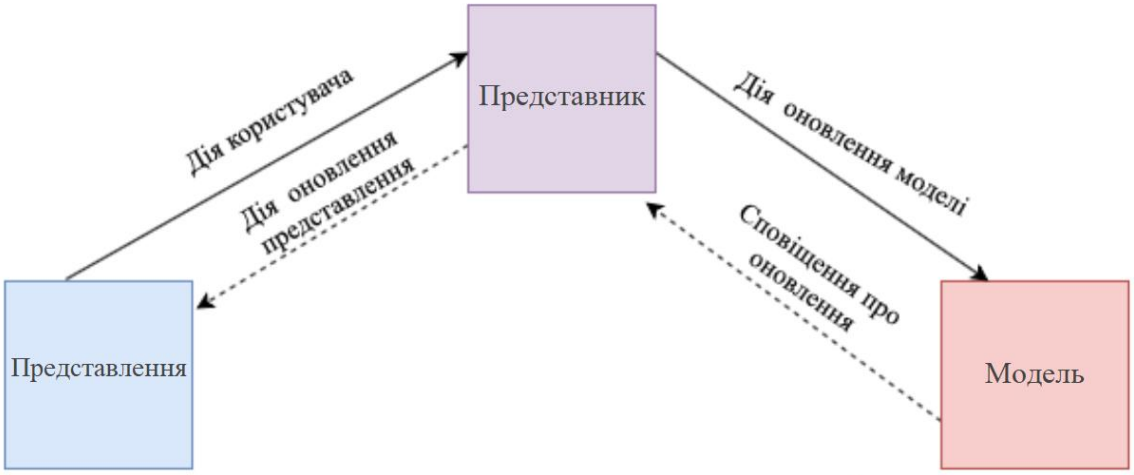


Рисунок 2.2 – Діаграма взаємодії між компонентами шаблону MVP

Основними перевагами моделі MVP можна назвати вирішення проблем зв'язності, а отже і масштабування проекту, а також спрощення процесу модульного тестування програми.

MVVM – це архітектурний шаблон, який є альтернативною модифікацією вищерозглянутого патерну MVP. Головною його задачею також э відокремлення користувацького інтерфейсу (Представлення) від бізнес-логіки (Моделі). Принцип роботи моделі зображено на рисунку 2.3.

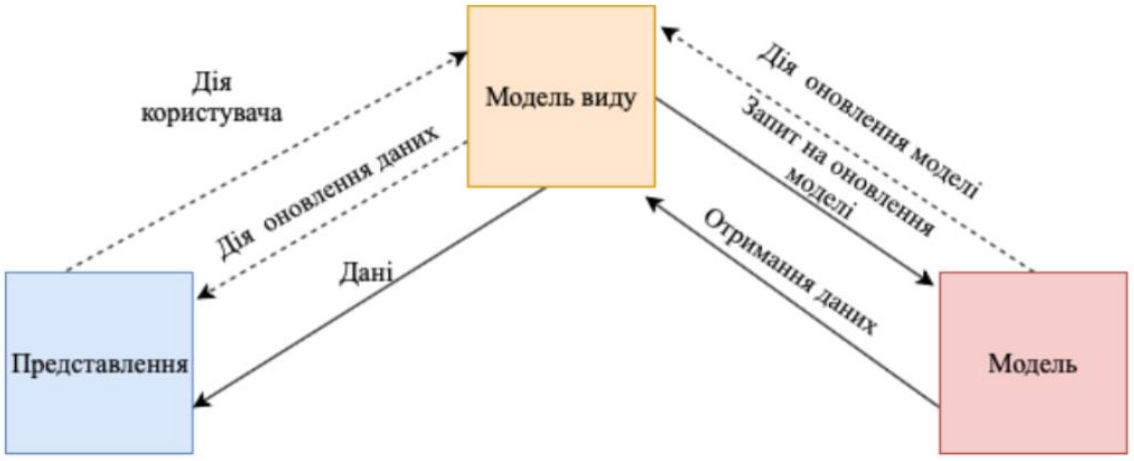


Рисунок 2.3 – Діаграма взаємодії між компонентами шаблону MVVM

На відміну від MVP у MVVM у ролі посередника виступає Модель виду. Вона відповідає за представлення з моделі та перетворення формату даних, і не змінює елементів представлення.

Головною перевагою MVVM є над MVP є те, що зв'язування представлення з моделлю виду відбувається автоматично, а для MVP даний зв'язок доводиться створювати самостійно. Проте це створює і головний недолік такої архітектури, адже для реалізації цього автоматичного зв’язку необхідно використовувати певні фреймворки, що негативно впливає на швидкодію та розмір застосунку.

VIPER – це найновіший архітектурний шаблон, що застосовується в розробці мобільних додатків та заснований на принципі єдиної відповідальності (Single Responsibility Principle) парадигми SOLID. Він складається з п’яти основних компонентів: Представлення (View), Взаємодія (Interactor), Представник (Presenter), Сутність (Entity), Провідник (Router). Принципи роботи компонентів представлені на рисунку 2.4.

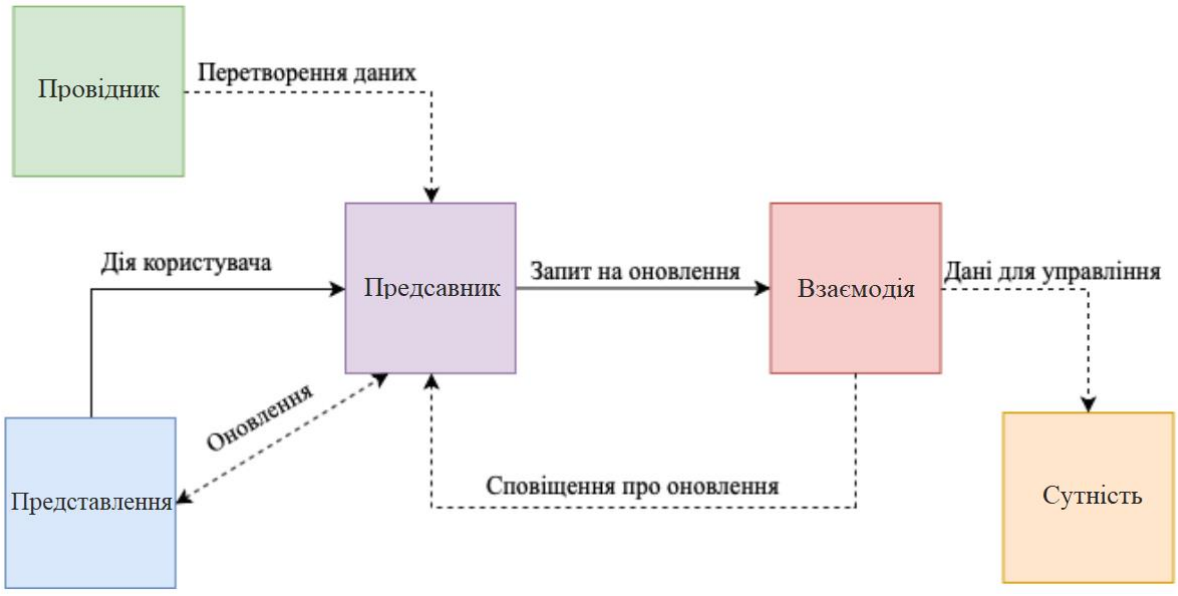


Рисунок 2.4 – Діаграма взаємодії між компонентами шаблону VIPER

Компоненти представлення та представник виконують ті ж самі задачі що і в MVP. Компонент взаємодія містить в собі бізнес логіку для керування об’єктами (Entity), та відповідає роботу з даними, та має подібний фунціонал до

компонента модель (Model) у вищеописаних архітектурних шаблонах. Компонент сутність – це об’єкти даних, якими керує Взаємодія (Interactor), та є найменшими структурними одиницями додатку. Провідник – це компонент, який відповідає за маршрутизацію між всіма іншими компонентами [14].

Головною перевагою VIPER над іншими архітектурними шаблонами з забезпечення повної незалежності усіх компонентів, з якої також випливає легкість масштабування проекту та проведення модульного тестування. Мінусами є збільшення складності проекту та кількості надлишкового коду, а отже і розміру додатку, а також негативний вплив на швидкодію.

Виконавши аналіз вищезазначених архітектурних шаблонів, було обрано найоптимальнішу архітектуру для розроблюваного додатка-щоденника, а саме – MVP, адже вона вирішує проблеми зв’язності, які є в MVC та не ускладнює код програми, а також підходить для невеликих проектів і не впливає на їх швидкодію негативним чином.

Так як мобільні додатки загалом використовують зміни стану об’єктів (події) у своїй роботі, то в такому разі доцільно додатково використовувати патерн EDA. Подійно-орієнтована архітектура (EDA) – архітектурний патерн ПЗ, що використовується для роботи з подіями, тобто створення подій, їх виявлення, використання і реагування на них.

EDA доцільно використати для реалізації процесу збереження заміток, адже він має відбуватись автоматично. Тобто в додатку буде відслідковуватись подія згортання та виходу з додатку, для того щоб зберегти записи, а також відслідковування події зміни тексту у полях, перегортання сторінок та натискання кнопок.

Головною частиною розроблюваного додатку є збереження самого вмісту записів, тому важливо правильно спланувати що як і куди записувати.

Збереження можна реалізовувати 2 методами, а саме:

– збереження в файл;

– збереження в базу даних.

В обох методів є свої переваги і недоліки, наприклад, в першого методу набагато легша реалізація, але в той же час, набагато менші можливості.

Тому для розробки було обрано запис в БД, адже в такому випадку у нас є можливість запису додаткової службової інформації без складних форматувань та парсингу тесту. Також запис в БД дозволить зробити пошук записів по тегам.

2.2 Проектування структури бази даних

База даних (БД) – сукупність даних, організованих відповідно до концептуальної структури, яка описує характеристики цих даних і взаємозв'язки між відповідними сутностями, підтримуючи одну або кілька областей застосування [15]. Для того щоб спроектувати базу даних, необхідно проаналізувати різні типи існуючих БД, та обрати таку, яка найкраще підходить для поставленого завдання, а саме – зберігання записів користувачів.

За розміщенням даних виділяють такі види БД:

– локальна, або централізована – це база даних, яка підтримується на одному пристрої.

– розподілена – це база даних, частини якої розміщують на різних пристроях мережі.

Оскільки в розроблюваному додатку БД необхідна для зберігання записів та різної службової інформації про них, і нема необхідності для обміну даними між різними пристроями та не використовується клієнт-серверна архітектура самого застосунку, розумним рішенням буде обрати локальну БД.

Також бази даних поділяють за моделлю організації даних на:

– ієрархічні;

– мережні;

– реляційні;

– об’єктно-орієнтовані.

Найпоширенішими на сьогоднішній день є реляційні бази даних. Вони зберігають дані у вигляді набору формально описаних таблиць, які мають різні відношення між собою, з них дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць БД. Свою популярність реляційна модель даних досягла завдяки тому, що у ній досягається більш високий рівень абстракції даних, ніж в інших моделях баз даних.

А отже найдоцільніше буде використовувати саме реляційну модель організації даних в БД.

Невід'ємною частиною процесу розробки таких БД є нормалізація, тобто усунення структурних недоліків, які призводять до надмірності даних. Нормалізація бази даних відбувається поетапно, завдяки послідовному приведенні БД в нормальні форми.

Нормальна форма являє собою обмеження схеми бази даних від деяких небажаних властивостей [16]. Відношення знаходиться в деякій нормальній формі, якщо задовольняє певний набір обмежень. Існує шість нормальних форм та дві додаткові (Бойса-Кодда та домен/ключ), проте на практиці зазвичай достатньо привести базу даних до 3 нормальної форми.

Відношення знаходиться в першій нормальній формі (1НФ), якщо воно задовольняє такі вимоги:

- відношення не має однакових кортежів;
- кортежі не впорядковані зверху вниз;
- атрибути не впорядковані зліва направо;
- всі значення атрибутів відношення є атомарними.[17]

Спочатку необхідно виділити сутності предметної області, які і будуть складати наше відношення. З функціональних вимог до розроблюваного додатку виокремлюємо, що наведені в ТЗ дані які необхідно зберігати і отримуємо наступний набір атрибутів:

- дата за яку відповідає запис;
- теги, які було додано до запису;

– головні задачі дня, тобто текст що відображається на головній сторінці тижня;

– додатковий текст дня.

В ході логічного моделювання пропонується зберігати всі дані в одній таблиці (таблиця 2.1). Оскільки відношення задовольняє всі вищеперераховані вимоги, то воно знаходиться в 1НФ

Таблиця 2.1 – Відношення в 1НФ

Ідентифікатор запису	Дата за яку відповідає запис	Головні задачі дня, текст що відображається на сторінці тижня	Додатковий текст дня	Доданий до запису тег
1	12.04.2021	Купити продукти; Випрати речі; Спекти торт.	Купити молоко, яйця, борошно, хліб, свічки	Покупки
2	13.04.2021	Зайти в зоомагазин Покупати кота; Обробити кота від паразитів Привітати брата.	Купити коту корм, краплі, таблетку та шампунь	День народження брата
3	13.04.2021	Зайти в зоомагазин Покупати кота; Обробити кота від паразитів Привітати брата.	Купити коту корм, краплі, таблетку та шампунь	Обробка кота від паразитів
4	14.04.2021	Купити продукти; Зашити рюкзак.	Купити сир, молоко, печиво, м’ясо.	Покупки

Аналізуючи таблицю, можна помітити, що дані зберігаються з великою надлишковістю, при цьому деякі дані є незалежними між собою. Якщо ніяких дій над таблицею не відбувається, то можна залишити її у такому вигляді, але якщо зміни відбуваються, то виникає велика кількість проблем (аномалій), які виникають при зміні стану БД.

Існують такі типи аномалій:

– аномалія вставки (Insert);

– аномалія оновлення (Update);

– аномалія видалення (Delete).

Всі три аномалії виникають через зберігання різнорідної інформації в одній таблиці. Отже, бачимо, що дана таблиця потребує нормалізації.

Правило приведення до 2НФ: Ті атрибути, які залежать від частини складного ключа, виносять в окремі відношення разом з частиною складного ключа. В початковому відношенні залишається частина складного ключа.

Для того щоб привести базу даних до 2НФ і усунути повторення необхідно створити додаткові таблиці для потенційних ключів. Тому створимо окрему таблицю для тегів з своїм ідентифікатором і винесемо туди інформацію, про запис який містить цей тег. Результат – таблиці 2.2 та 2.3.

Таблиця 2.2 – Відношення «Записи»

Ідентифікатор запису	Дата за яку відповідає запис	Головні задачі дня, текст що відображається на сторінці тижня	Додатковий текст дня
1	12.04.2021	Купити продукти; Випрати речі; Спекти торт.	Купити молоко, яйця, борошно, хліб, свічки
2	13.04.2021	Зайти в зоомагазин Покупати кота; Обробити кота від паразитів Привітати брата.	Купити коту корм, краплі, таблетку та шампунь
3	14.04.2021	Купити продукти; Зашити рюкзак.	Купити сир, молоко, печиво, м’ясо.

Таблиця 2.3 – Відношення в «Теги»

Ідентифікатор тегу	Ідентифікатор запису	Тег
1	1	Покупки
2	2	День народження брата
3	2	Обробка кота від паразитів
4	3	Покупки

Знову бачимо, що записи в таблиці «Теги» надлишкові, тому проводимо наступний етап нормалізації і приводимо нашу БД до 3НФ.

Відношення знаходиться у 3НФ тоді і тільки тоді, коли воно знаходиться у 2НФ і всі не ключові атрибути взаємонезалежні, тобто жоден з них не є функціонально залежним від іншого. Тому винесемо проміжну таблицю «теги в записах», яка зробить всі не ключові атрибути взаємонезалежними. Структура БД у 3НФ подана у таблицях 2.4-2.6.

Таблиця 2.4 – Відношення «Записи»

Ідентифікатор запису	Дата за яку відповідає запис	Головні задачі дня, текст що відображається на сторінці тижня	Додатковий текст дня
1	12.04.2021	Купити продукти; Випрати речі; Спекти торт.	Купити молоко, яйця, борошно, хліб, свічки
2	13.04.2021	Зайти в зоомагазин Покупати кота; Обробити кота від паразитів Привітати брата.	Купити коту корм, краплі, таблетку та шампунь
3	14.04.2021	Купити продукти; Зашити рюкзак.	Купити сир, молоко, печиво, м’ясо.

Таблиця 2.5 – Відношення в «Теги»

Ідентифікатор тегу	Тег
1	Покупки
2	День народження брата
3	Обробка кота від паразитів

Таблиця 2.6 – Відношення в «Теги в записах»

Ідентифікатор тегу	Ідентифікатор запису
1	1
2	2
3	2
1	3

Тепер наша база даних знаходиться в 3НФ, адже задовольняє всі вищеперераховані умови. Сама архітектура БД є не складною та перевантаженою. В ній міститься всього три таблиці, на основі яких було розроблено діаграму сутність зв'язок (рисунок 2.5), яка ілюструє архітектуру розроблюваної бази даних

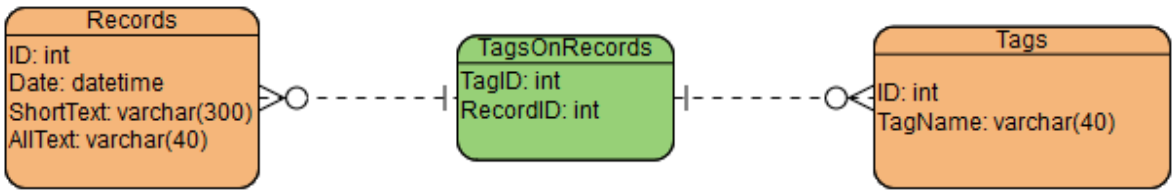


Рисунок 2.5 – Діаграма сутність зв'язок

Саме така архітектура БД дозволяє реалізувати в додатку швидкий пошук по тегам та по ключовим словам які містяться в замітках.

2.3 Проектування інтерфейсу користувача

Розробка інтерфейсу додатку неможлива без розуміння базових правил UX/UI-дизайну. Важливо розуміти, що не все що зручно використовувати при роботі з комп'ютером чи ноутбуком можна буде так же використовувати на смартфону, і не всі рішення які гарно виглядають на моніторі будуть такими ж на екрані мобільного телефону.

UX (user experience) – термін який дослівно перекладається як «досвід користувача», і розуміє під собою весь досвід який отримує користувач при взаємодії з продуктом. В свою чергу UX-дизайн – це дизайн и проектування інтерфейсів основане на вивченні користувацького досвіду та поведінки.

UI (user interface) – перекладається як «користувацький інтерфейс», і означає поняття яке охоплює не тільки графічний інтерфейс, а також і тактильний, звуковий та голосовий. UI-дизайн – це втілення матеріальне втілення

досвіду користувача, яке включає в себе роботу над візуальною і графічною частиною інтерфейсу.

При розробці дизайну я вирішила притримуватись мінімалістичного підходу, адже про організації свого часу та думок нічого не повинно відволікати. За праобраз дизайну я взяла оформлення першої сторінки з паперового щоденника на рисунку 2.6.

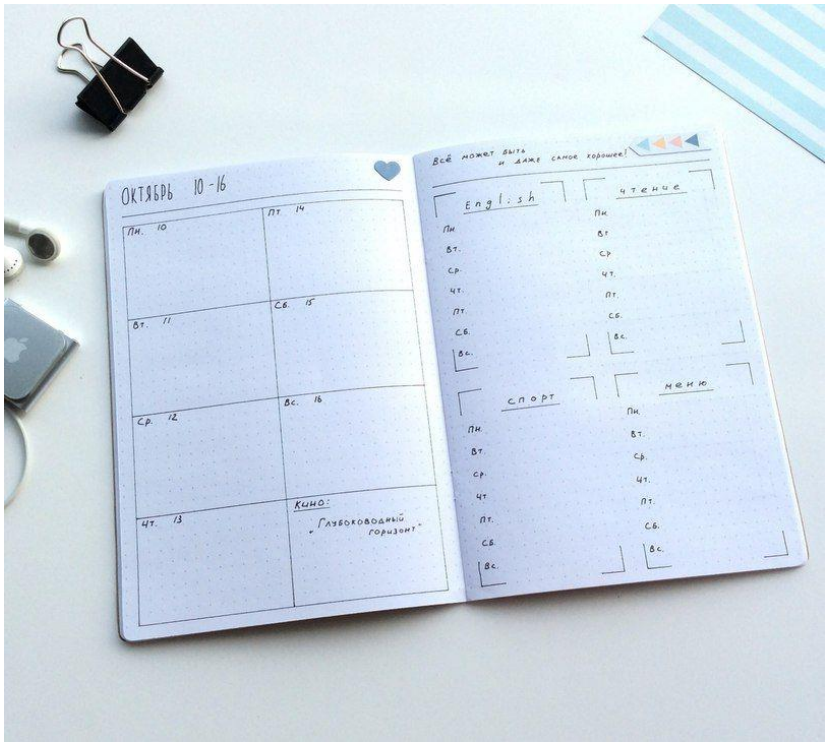


Рисунок 2.6 – Паперовий щоденник з саморобною розміткою

Кольорову гаму для додатку було обрано нейтрально-сіру (рисунок 2.7), щоб в майбутньому була можливість доробити інструменти для персоналізації. Також перевагою такого кольорового оформлення є те, що сірий є найелегантнішим рішенням при використанні монохроматичної схеми кольорової палітри.

Перший колір схеми на рисунку 2.7 буде використовуватись як головний колір додатку, тобто в верхній функціональній лінії. Другий колір – для тексту який написано на світлому тлі. Третій та четвертий кольори будуть слугувати тлом для записів. А останній буде використаний для тексту на темному тлі.

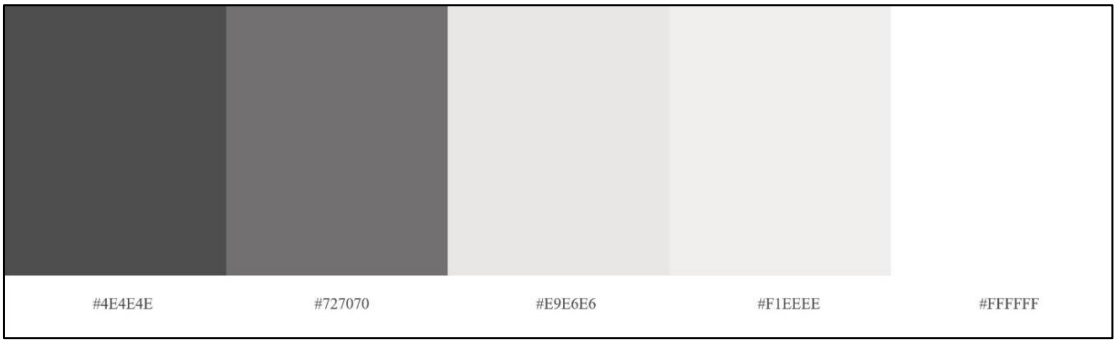


Рисунок 2.7 – Кольорова схема додатку (монохроматична)

Притримуючись такого розподілення кольорів, а також беручи до уваги розмежування днів тижня як на рисунку 2.6 було розроблено прототип головної сторінки додатку (рисунок 2.8), на якій будуть виводитись головні завдання кожного з днів тижня.

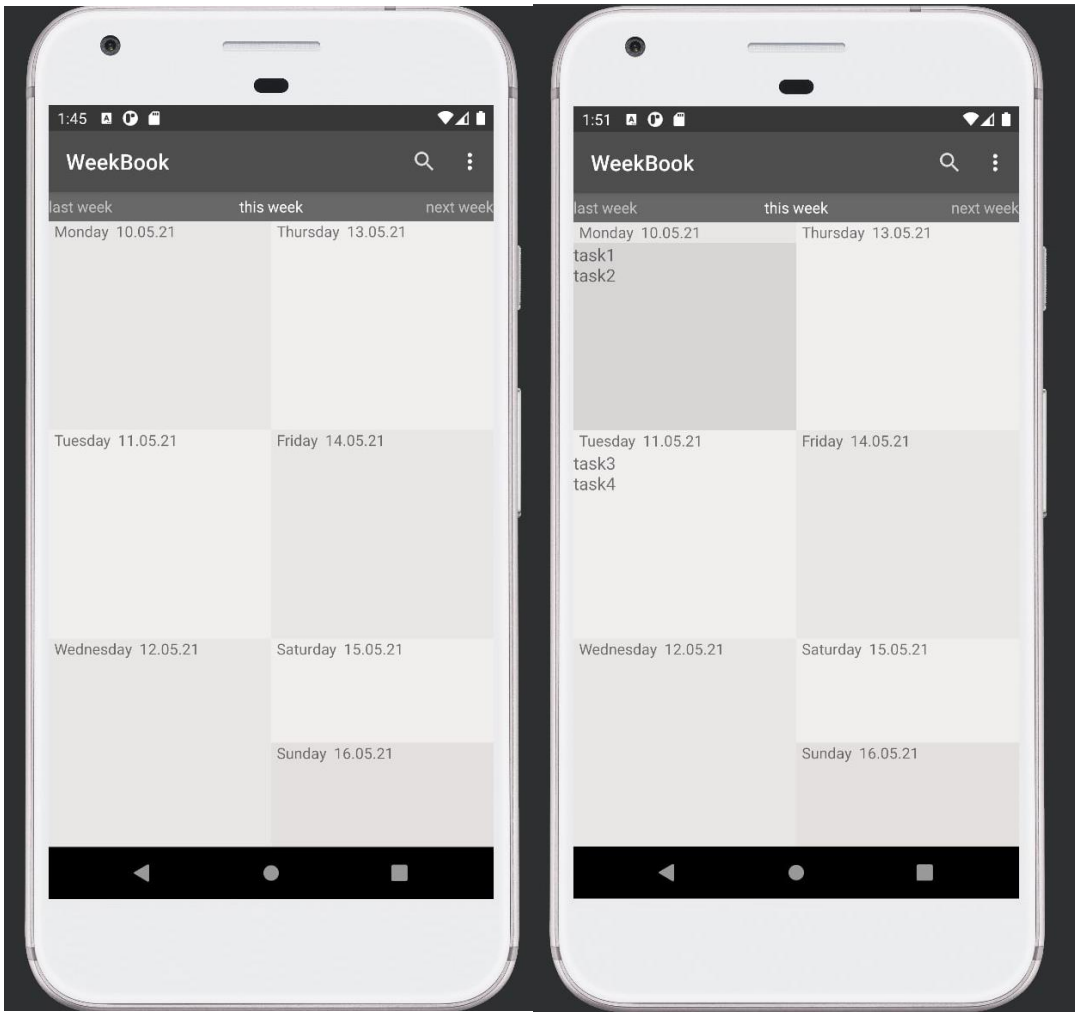


Рисунок 2.8 – Головний екран додатку

В верхньому правому куті розміщено значок меню, при натисканні на який буде відкриватися меню з пунктами: зробити бекап, налаштування та про програму. Поруч розміщений значок пошуку, при натисканні на який користувач зможе здійснити пошук по даті, тегам, чи просто тексту.

Нижче, для зручності користування, розміщена лінія, на якій буде виведено який тиждень зараз відображається (поточний, минулий, наступний).

Для того щоб створити запис користувачу необхідно натиснути а плитку з днем, до якого цей запис відноситься. Після чого відкриється сторінка дня (рисунок 2.9), де можна буде ввести основний текст, який буде виводитись на головній сторінці, а також додатковий текст дня, який буде видно тільки на сторінці дня. В самій верхній частині розміщений підпис дати і дня тижня. Для того щоб зробити запис необхідно натиснути у відповідному полі.

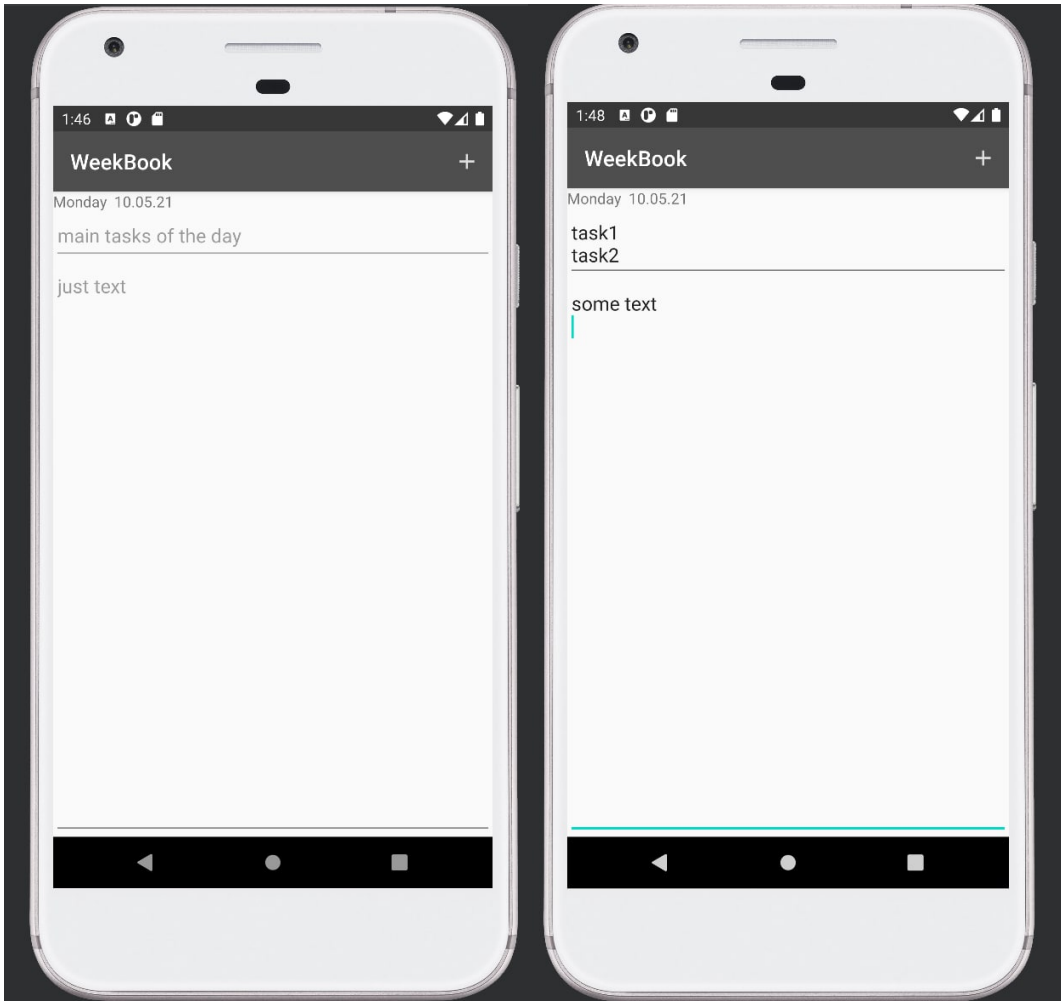


Рисунок 2.9 – Сторінка дня, для редагування заміток

В правому верхньому куті сторінки на рисунку 2.9 розміщений знак плюса, при натисканні на який відкривається сторінка додавання тегів зображена на рисунку 2.10, яка дозволяє додавати та видаляти теги поточного запису.

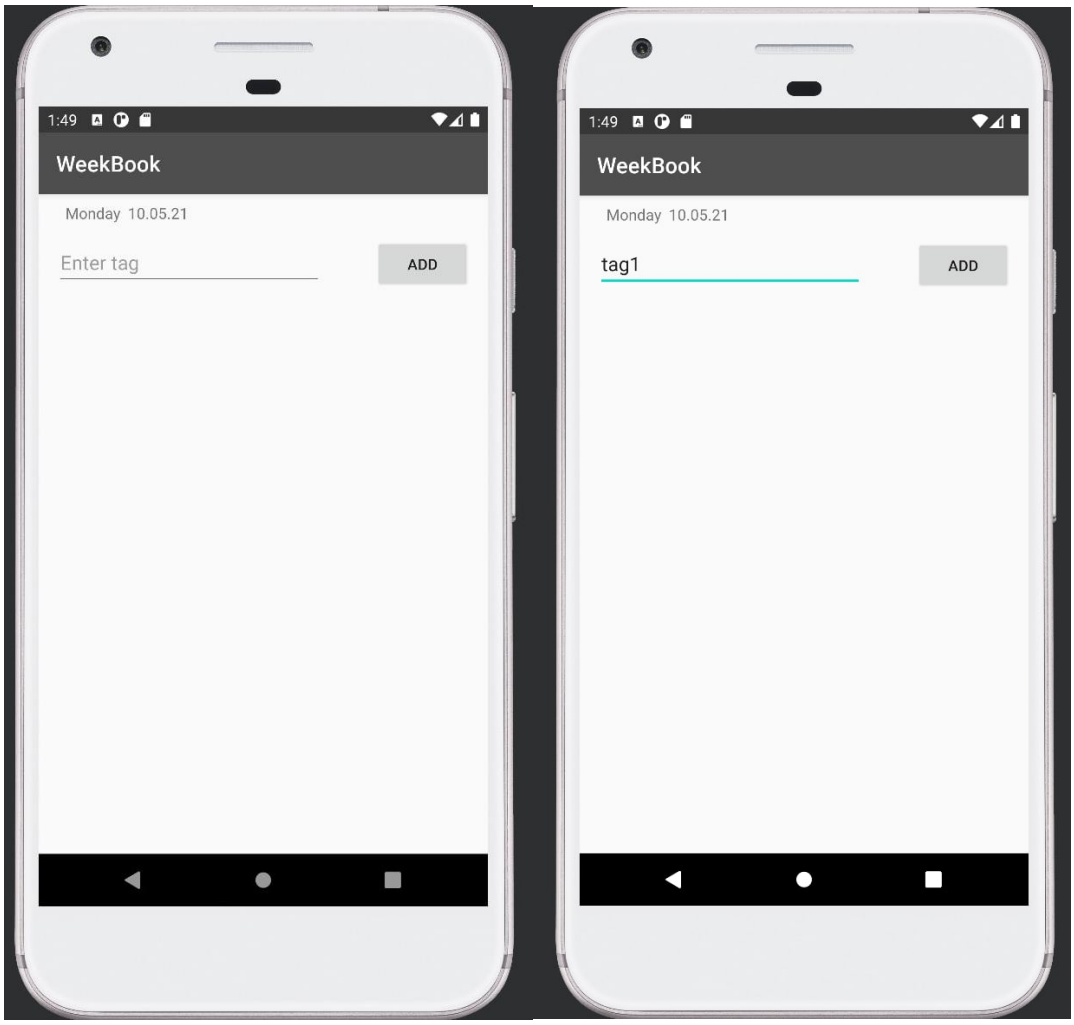


Рисунок 2.10 – Сторінка для додавання тегів

На сторінці розміщене поле для вводу назви тегу, та кнопка яка додає тег до запису з сторінки якого було відкрито додавання тегів. Нижче розміщене поле, в якому відображаються вже додані до цього запису теги.

На рисунку 2.11 подано повну схему всіх переходів між екранами додатку, а також всіх полів для введення, пунктів меню та кнопок, які містить в собі кожен з цих екранів та переходів, які вони ініціюють. Стартовим екраном додатку є сторінка «задачі тижня»

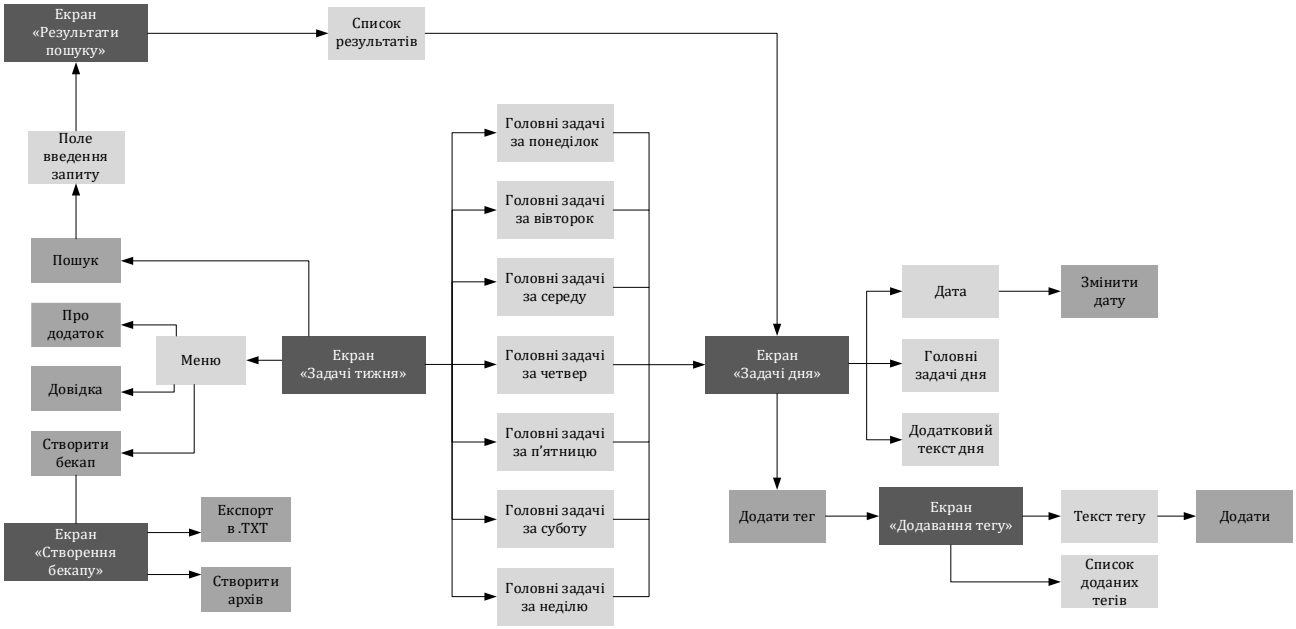


Рисунок 2.11 – Схема переходів екранів додатку

Останнім, проте дуже важливим етапом створення дизайну додатку є створення його іконки. Саме вона слугує обличчям додатку та репрезентує продукт користувачу. Головною задачею іконки додатку є наглядна демонстрація суті застосунку та його особливих відмінностей від інших програм. Саме тому було вирішено обрати як базу для створення іконки зображення записника, і створити на ньому характерне розмежування сторінки, яке і є одною з головних особливостей додатку і вирізняє його з поміж інших продуктів. В результаті було розроблено та намальовано іконку, зображену на рисунку 2.12, яка містить в собі всі вищеописані особливості.

Рисунок 2.12 – Дизайн іконки програми

2.4 Розробка алгоритму роботи мобільного додатка

Після того, як було обрано архітектуру додатку та спроектовано його інтерфейс та архітектуру БД, можна приступити до створення алгоритму роботи програми. В першу чергу необхідно виділити основні процеси які відбуваються при роботі з додатком. Для розроблюваного додатку такими процесами є:

- створення та редагування заміток та їх запис в БД;
- додавання тегів до заміток;
- пошук заміток;
- створення бекапу.

Відповідно до цих процесів було створено діаграми переходів станів, що зображені на рисунках 2.13- 2.16.

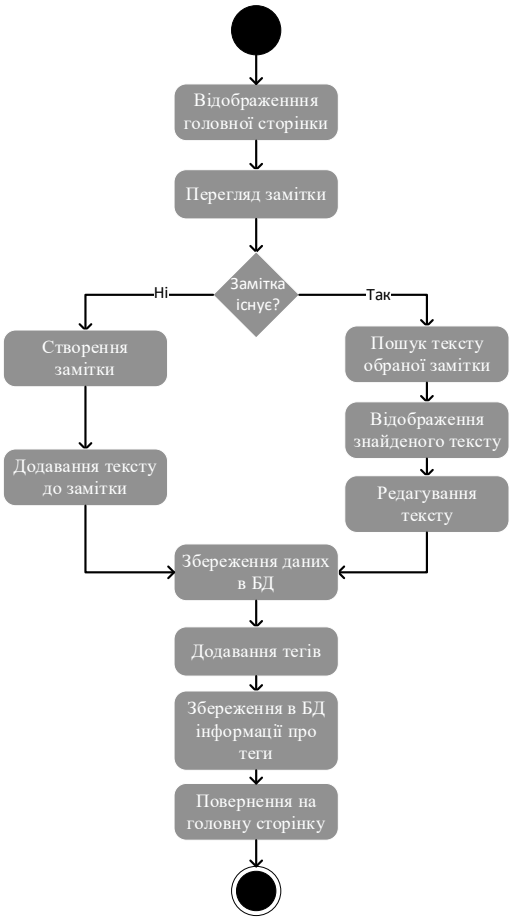


Рисунок 2.13 – Діаграма переходів станів створення і редагування замітки

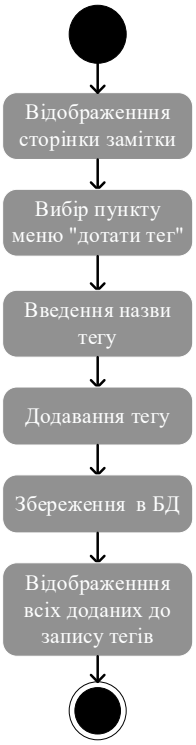


Рисунок 2.14 – Діаграма переходів станів створення і додавання тегів

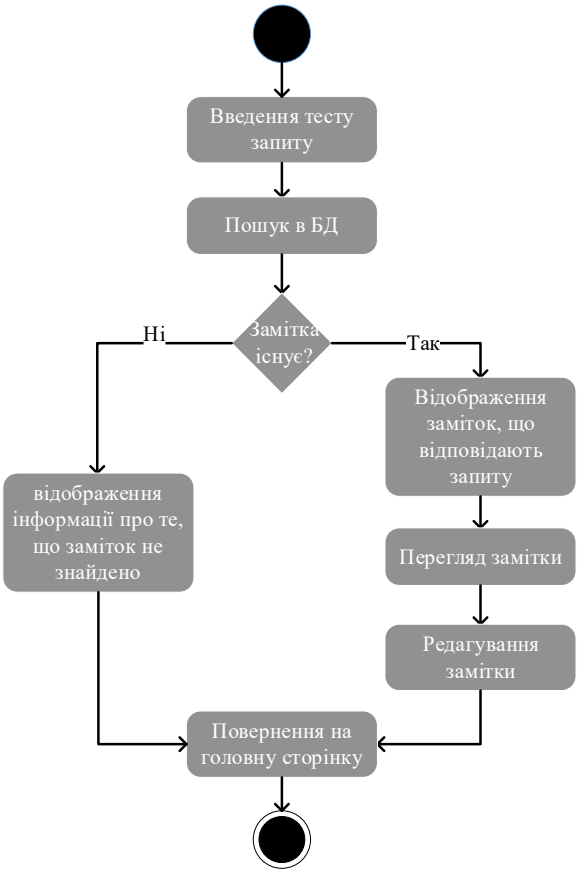


Рисунок 2.15 – Діаграма переходів станів пошуку записів

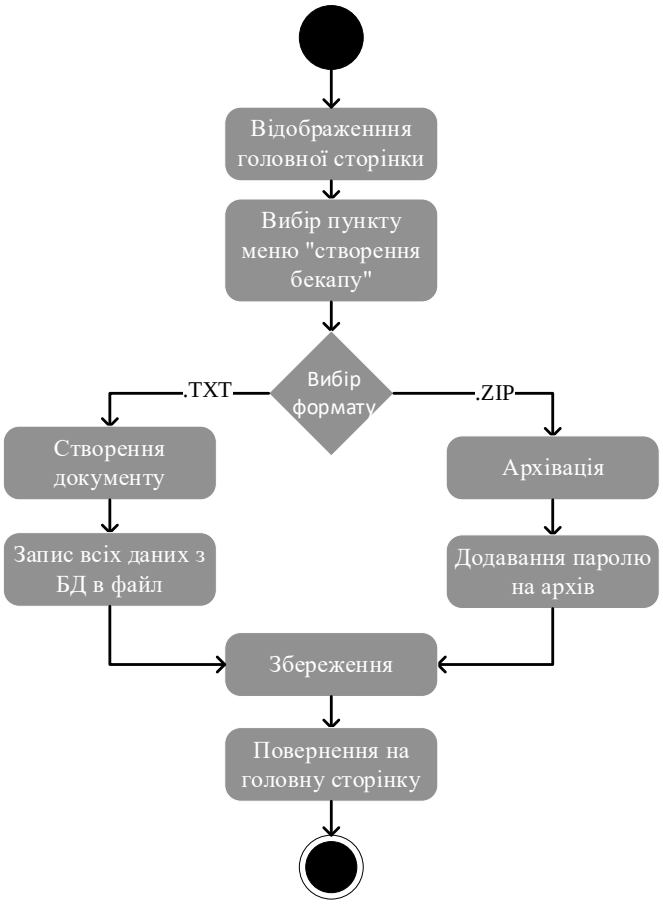


Рисунок 2.16 – Діаграма переходів станів створення бекапу

При проектуванні інтерфейсу розроблюваного додатку було виділено п’ять екранів які будуть відображати різні функції. Кожен з цих екранів стане відповідною активністю додатку.

Всі активності різним чином взаємодіють з базою даних додатку. Специфікації процесів взаємодії наведено в DFD діаграмі на рисунку 2.17. Коли користувач запускає додаток, то програма запитує дані з локальної БД, і якщо замітки на поточний тиждень створені – вводить їх. Якщо користувач створює замітку – вона зберігає короткий і довгий текст замітки в БД а також, записує дату, якій відповідає ця замітка. При редагуванні відбуваються аналогічні дії.

Якщо користувач вирішив добавити тегів до замітки, то в таблиці БД з тегами буде створено новий запис, а в таблиці-відповідності буде створено запис з ключем замітки та ключем тегу.

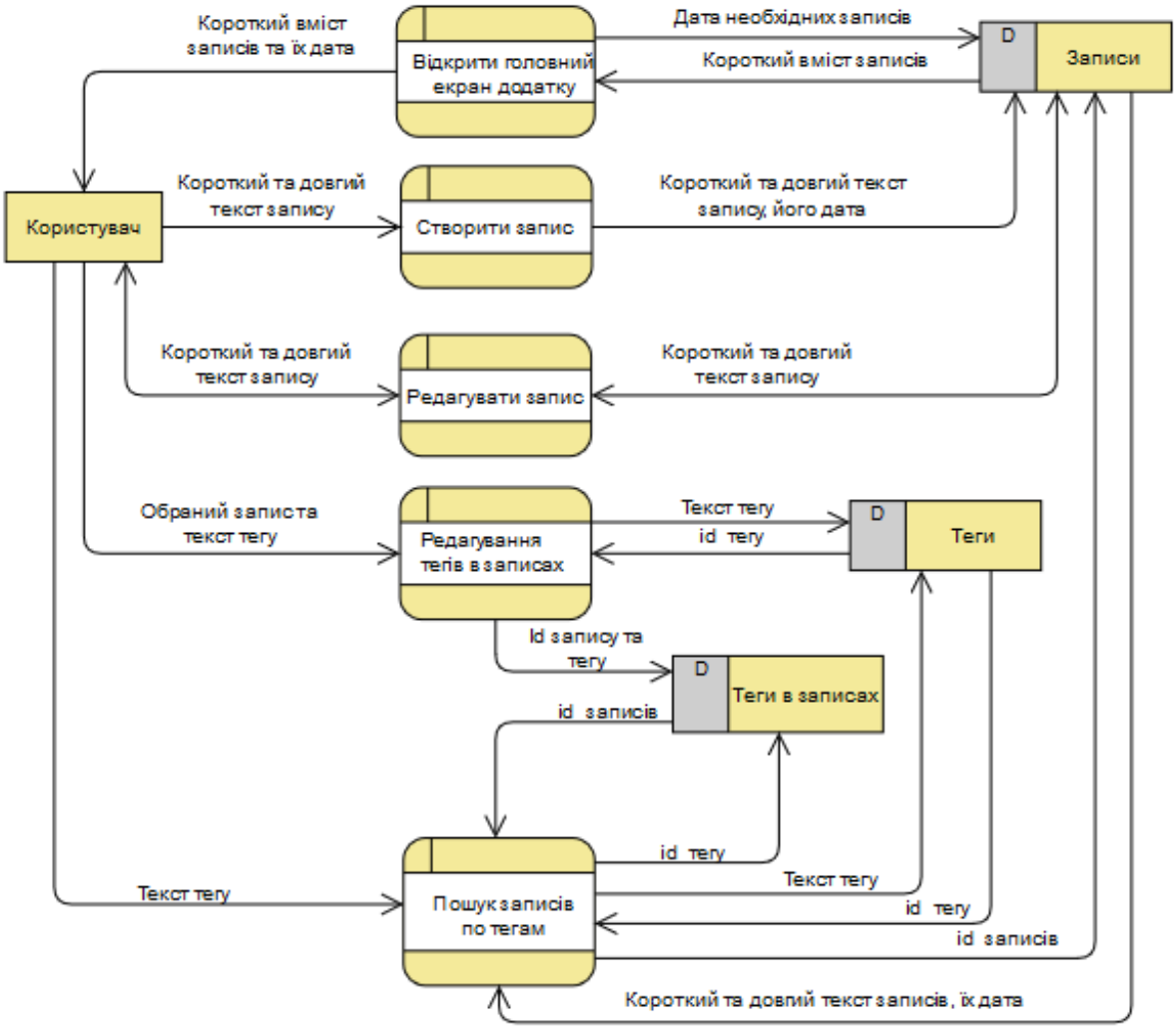


Рисунок 2.17 – Діаграма потоків даних

2.5 Аналіз та вибір технологій і методів реалізації додатка

На теперішній час ринок мобільних застосунків є дуже великим, прибутковим та затребуваним. Через це створюється величезна кількість інструментів, методів та технологій для розробки додатків. Частина з них допомагають розробнику швидко та якісно створювати продукт. Якісь дозволяють програмувати тільки під певну платформу. Інші ж існують тільки для вирішення якихось вузьких задач.

Так як у розділі 1.1 було проведено аналіз предметної області та порівняно різні платформи та типи додатків і було обрано розроблювати додаток нативного

типу для платформи Android, необхідно проаналізувати та обрати технології для реалізації додатку.

Переважно виділяють вісім мов програмування, за допомогою яких можна створити Android-додаток [18]:

– Java;
– Kotlin;
– Python;
– C/C++;
– JavaScript;
– Dart;
– C#.

Проаналізуємо кожну з них та на основі їх сильних та слабких сторін оберемо ту, яка підходить найкраще.

Java – строго типізована, об’єктно орієнтована мова програмування, офіційна мова для розробки Android-додатків. Головними перевагами цієї мов є велика спільнота розробників, завдяки якій існує багато готових рішень, модулів та різноманітних бібліотек. Також значною перевагою цієї мови програмування є те, що одне з найпопулярніших та найзручніших середовищ розробки Android Studio від самого початку було орієнтовано для роботи з Java. З недоліків цієї мови можна відмітити тільки громіздкий синтаксис.

Kotlin – статично типізована, об’єктно орієнтована мова програмування, що працює поверх віртуальної машини Java. Головними відмінностями Kotlin та Java є лаконічність коду та реалізація перевірки виключень. І якщо лаконічність і правда є дуже вагомою перевагою, адже робить код менш громіздким та легшим для розуміння, то відсутність умов для виключень, що перевіряються (checked exception) є недоліком для одних розробників і перевагою для інших. Також з недоліків цієї мови відмічають малу спільноту розробників, причиною цього є те, що мова є достатньо молодою, а наслідком – менша кількість готових рішень, модулів та бібліотек.

На відміну від розглянутих двох мов, інші рекомендується розглядати як альтернативу, а не базу для Android-розробки.

Python – динамічно типізована, об’єктно орієнтована мова програмування. Сама мова не передбачала забезпечення засобів для мобільної розробки, проте ентузіастам вдалось це змінити, та пристосувати її для Android-розробки. Реалізована така можливість за допомогою open source бібліотеки Kivy та набору UI-інструментів BeeWare. Зазвичай Python використовують для розробки якихось специфічних проектів, де потребується вирішення тих задач, з якими ця мова справляється краще інших. В звичайній мобільній розробці його майже ніколи не використовують.

C – слабо типізована, імперативна, процедурно орієнтована мова програмування з можливостями низькорівневого доступу до пам’яті.

C++ – високорівнева мова програмування з підтримкою трьох падигм: процедурної, об'єктно-орієнтованої та узагальненої. Ця мова програмування базується на основі C, та розроблялася як її наслідниця з можливістю використання класів.

Ці дві мови програмування зазвичай позиціонуються як мови з хорошою швидкодією, тому найдоцільніше використовувати їх при розробці програм з складними обчисленнями, прикладом такого додатку може бути мобільна 3D-гра. Суттєвим недоліком роботи з цими мовами є те, що вони не забезпечують прямих інструментів для роботи з Android. Тому на цих мовах можна буде написати тільки частину застосунку, до прикладу бібліотеку, а для всього іншого доведеться використати Java.

JavaScript – динамічно типізована, прототипна, об’єктно орієнтована мова програмування. Переважно застосовується при розробці скриптів для веб-сторінок. Для роботи з Android зазвичай використовується фреймворк React Native, який дозволяє створювати додатки з багатофункціональним UI. При тому ці застосунки є повністю нативними і не є мобільними веб-додатками, тому що React Native використовує і ж компоненти, що і звичайні додатки для Android.

Перевагами написання мобільної програми а цій мові є: швидка збірка проекту, наявність flexbox для створення UI та проста передача даних через мережу з використанням API. З недоліків можна відмітити те, що ядро React Native є досить нестабільним, а також не підходить для складних проектів які потребують більше ніж відправлення запитів та отримання відповіді.

Dart – це мова структурованого програмування, розроблена компанією Google, для використання в веб-розробці. Вона створювалась як заміна JavaScript, проте на сьогоднішній день компанія змінила вектор розвитку мови в користь серверних та мобільних застосунків. Для мобільної розробки на Dart використовується SDK Flutter.

Перевагами Dart вважають хорошу швидкодію програм та технологію Hot Reload в Flutter – швидке перезавантаження з збереженням стану. З недоліків можна назвати те що мова є дуже молодою і технологія Flutter є достатньо новою, а отже ще нема достатньої кількості цікавих рішень та інструментів які допомагають під час розробки.

C# – це строго статично типізована, об’єктно орієнтована мова програмування, створена для платформи .NET. Для розробки мобільних додатків на C# необхідно використовувати платформу Xamarin, за допомогою якої можна створювати кросплатформні додатки, при цьому користувацький інтерфейс необхідно створювати для кожної платформи окремо. Зазвичай для написання Android-додатків, цю мову використовують ті, хто її знає і хоче спробувати себе в мобільній розробці.

Після розгляду наявних технологій для створення Android-додатків, було прийнято рішення розробляти застосунок на мові Java, тому що вона є класичним рішенням, яке має багато інструментів які допоможуть в розробці програми, а також ідеально підходить для обраного типу додатку.

Для середовище для написання програмного коду було обрано Android Studio, адже саме була вона визначена Google як офіційна IDE для розробки програм для Android. Android Studio є зручною та функціональною IDE, адже

містить всі необхідні інструменти для розробки мобільних додатків. Також вона була розроблена спеціально для створення додатків на мові Java і повністю підтримує весь її функціонал, а отже є вдалим вибором, при написанні додатку на цій мові програмування.

У цьому розділі було розроблено архітектуру додатку, та його бази даних, спроектовано зручний сучасний інтерфейс, який відповідає поточним тенденціям UI/UX розробки, а також обрано технології та інструменти для розробки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Реалізація логіки мобільного додатку

У попередньому розділі було виділено та спроектовано п’ять активностей додатку. Так як кожна з них виконує одну конкретну функцію, то таких функцій також є п’ять, а саме:

- відображення задач на тиждень;
- редагування заміток;
- додавання тегів до заміток;
- пошук заміток;
- створення бекапу.

Так як основною особливістю додатку є можливість бачити всі свої завдання на тиждень, то це і буде головною активністю MainActivity, яка буде відображатись при загрузці додатку. Так як нам необхідно перегортати тижні, то доцільно буде використати для цієї активності елемент пейджер адаптер, клас якого відповідно називається WeekPagerAdapter, та самі завдання днів тижня розмістити на окремому фрагменті WeekFragment.

Фрагмент – це багаторазова частина інтерфейсу додатка, яка визначає і управляє власним макетом, має власний життєвий цикл і може обробляти власні події введення. Фрагменти не існують самі по собі – їх повинна розміщувати якась активність чи інший фрагмент. [19]

В коді класу головної активності MainActivity в методах onCreateOptionsMenu та onOptionsItemSelected відбувається генерація та обробка роботи пунктів меню, тих що викликають до прикладу активності для пошуку записів та створення бекапу; також в цій активності в методі onCreate відбувається створення та налаштування елементу WeekPagerAdapter

```
@Override
 protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        weekPagerAdapter = new WeekPagerAdapter(getSupportFragmentManager(),
        1, this );
        ViewPager viewPager = findViewById(R.id.viewpager);
        viewPager.setAdapter(weekPagerAdapter);
        viewPager.setCurrentItem(4);

    }

@Override
 public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        MenuItem searchItem = menu.findItem(R.id.action_search);
        SearchView searchView =(SearchView) searchItem.getActionView();
        return true;
    }

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    switch(id){
        case R.id.settings:
            return true;
        case R.id.about:
            showAbout();
            return true;
        case R.id.action_search:
            Intent intent = new Intent(getApplicationContext(),
SearchActivity.class);
            startActivity(intent);
            return true;
        case R.id.action_backup:
            Intent intent2 = new Intent(getApplicationContext(),
BackupActivity.class);
            startActivity(intent2);
            return true;
    }
    return super.onOptionsItemSelected(item);
}
```

В коді класу пейджер адаптера WeekPagerAdapter відбувається генерація та керування фрагментом WeekFragment в методі getItem, та розраховуються дати – обчислюється те якому дню тижня відповідає яка дата в методі getToday

```
@NonNull
@Override
public Fragment getItem(int position) {
        Bundle arguments = new Bundle();
        WeekFragment weekFragment=new WeekFragment();
        getToday(position,arguments);
        weekFragment.setArguments(arguments);
        return weekFragment;
}


public void getToday(int position,Bundle arguments)
{
        //находимо сьогоднішню дату
```

```
        SimpleDateFormat sdf = new SimpleDateFormat(" dd.MM.yy");
        Calendar calendar = Calendar.getInstance();
        //знаходимо дату понеділка поточного тижня
        calendar.add(Calendar.DAY_OF_MONTH, -(calendar.get(Calendar.DA
Y_OF_WEEK)== Calendar.SUNDAY? 6:calendar.get(Calendar.DAY_OF_WEEK)-2));
        //здвигаємо на кількість пролистаних фрагментів
        calendar.add(Calendar.DAY_OF_MONTH,(position-4)*7);
        //присвоюємо дату кожноднятижня і далі здвигаємо на 1 день
        arguments.putString(WeekFragment.MONDAY, "Monday "+
sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH,  1);
        arguments.putString(WeekFragment.TUESDAY, "Tuesday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.WEDNESDAY, "Wednesday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.THURSDAY, "Thursday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.FRIDAY, "Friday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.SATURDAY, "Saturday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.SUNDAY, "Sunday
"+sdf.format(calendar.getTime()));
}
```

В класі WeekFragment розміщені елементи які відображають головний текст кожного дня тижня. При натисканні на елемент дня тижня відкриється активність дня DayActivity. Клас фрагменту в методі displayDatabaseInfo взаємодіє з спеціальними класами DBHelper та WeekbookContract, що працюють з БД для завантаження заміток поточного дня.

Спочаткуу методі onCreateView ми отримуємо дати, які згенерував WeekPagerAdapter, та записуємо їх у відповідні змінні

```
@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.week_fragment, container, false);
        Bundle arguments = getArguments();
        if (arguments != null) {
             text_Monday = arguments.getString(MONDAY);
             text_Tuesday = arguments.getString(TUESDAY);
             text_Wednesday = arguments.getString(WEDNESDAY);
             text_Thursday = arguments.getString(THURSDAY);
             text_Friday = arguments.getString(FRIDAY);
             text_Saturday = arguments.getString(SATURDAY);
             text_Sunday = arguments.getString(SUNDAY);
```

```
            displayValues(view);
        }
     return view;
}
```

Для відображення інформації з БД створюємо метод displayDatabaseInfo який буде працювати з спеціальними класами DBHelper та WeekbookContract, щоб зробити запит та завантажити запис з бази даних та відобразити його у певному елементі

```
private void displayDatabaseInfo(String date, TextView textView) {
        // Створюємо і відкриваємо для читання БД
        SQLiteDatabase db = myDBHelper.getReadableDatabase();

        // Задаємо умову для вибірки - список стовбчиків
        String[] projection = {
                WeekbookContract.RecordsEntry._ID,
                WeekbookContract.RecordsEntry.COLUMN_SHORT_TEXT,
                WeekbookContract.RecordsEntry.COLUMN_ALL_TEXT};
        String selection = WeekbookContract.RecordsEntry.COLUMN_DATE + "=?";
        String[] selectionArgs = {date};
        // Робимо запит
        Cursor cursor = db.query(
                WeekbookContract.RecordsEntry.TABLE_NAME,   // таблиця
                projection,            // стовбчики
                selection,                  // стовбчики для умови WHERE
                selectionArgs,                  // значення для умови WHERE
                null,                  // Don't group the rows
                null,                  // Don't filter by row groups
                null);                   // порядок сортування


        TextView displayTextView = textView;
        if (cursor.getCount()==0) {
            insert( date);
        }
        try {
            // Дізнаємось індекс стовбчиків
            int idColumnIndex
=cursor.getColumnIndex(WeekbookContract.RecordsEntry._ID);
            int Short_Text_ColumnIndex =
cursor.getColumnIndex(WeekbookContract.RecordsEntry.COLUMN_SHORT_TEXT);

            // Проходимо через всі записи
            while (cursor.moveToNext()) {
                //Використовуємо індекс для отримання рядків чи чисел
                int currentID = cursor.getInt(idColumnIndex);
                //зберігамо індекс запиту
                recId = (long)currentID;
                String currentShortText =
cursor.getString(Short_Text_ColumnIndex);

                // Виводимо значення кожного стовпчика
                displayTextView.setText(currentShortText);
            }
        } finally {
```

```
            // Закриваємо курсор
            cursor.close();
        }
    }
```

Наступним створюємо метод getTextForAll в якому звертаємось до методу displayDatabaseInfo і передаємо інформацію для кожного дня тижня, та отримуємо записи які відповідають цим датам

```
public void getTextForAll()
    {
        //відображаємо дані з БД
        displayDatabaseInfo(text_Monday,mondayEditText);
        displayDatabaseInfo(text_Tuesday,tuesdayEditText);
        displayDatabaseInfo(text_Wednesday,wednesdayEditText);
        displayDatabaseInfo(text_Thursday,thursdayEditText);
        displayDatabaseInfo(text_Friday,fridayEditText);
        displayDatabaseInfo(text_Saturday, saturdayEditText);
        displayDatabaseInfo(text_Sunday,sundayEditText);
    }
```

В методі displayValues ініціалізуємо елементи для відображення тексту та створюємо лістенер, який прослуховує чи натискалось на елемент

```
private void displayValues(View v) {
        mondayEditText=(TextView) v.findViewById(R.id.mondayEditText);
        tuesdayEditText=(TextView) v.findViewById(R.id.tuesdayEditText);
        wednesdayEditText=(TextView) v.findViewById(R.id.wednesdayEditText);
        thursdayEditText=(TextView) v.findViewById(R.id.thursdayEditText);
        fridayEditText=(TextView) v.findViewById(R.id.fridayEditText);
        saturdayEditText=(TextView) v.findViewById(R.id.saturdayEditText);
        sundayEditText=(TextView) v.findViewById(R.id.sundayEditText);


        myDBHelper= new DBHelper(v.getContext());
        getToday(v);
        getTextForAll();
        //створюємо лістенер, який слухає чи натискалось на елемент
        View.OnClickListener listener= new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                goNewView(v);
            }
        };
        //назначаємо цей лістенер елементам
        mondayEditText.setOnClickListener(listener);
        tuesdayEditText.setOnClickListener(listener);
        wednesdayEditText.setOnClickListener(listener);
        thursdayEditText.setOnClickListener(listener);
        fridayEditText.setOnClickListener(listener);
```

```
        saturdayEditText.setOnClickListener(listener);
        sundayEditText.setOnClickListener(listener);
 }
```

Завдяки лістенеру створеному в displayValues реалізуємо метод goNewView, який після натискання на елемент, знаходить до якого дня належав цей елемент, та створює нову активність DayActivity, яка буде відповідати обраній даті, та передає в неї необхідну інформацію

```
public void goNewView(View v){
        Intent intent = new Intent(v.getContext(), DayActivity.class);
        switch (v.getId()) {
            case R.id.mondayEditText:
                // Вказуємо між якими Activity буде зв'язок
                intent.putExtra("weekDay", text_Monday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            case R.id.tuesdayEditText:

                intent.putExtra("weekDay", text_Tuesday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            case R.id.wednesdayEditText:

                intent.putExtra("weekDay",text_Wednesday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            case R.id.thursdayEditText:

                intent.putExtra("weekDay", text_Thursday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            case R.id.fridayEditText:

                intent.putExtra("weekDay",text_Friday);
                // показуємо нове Activity
                startActivity(intent);

                break;
            case R.id.saturdayEditText:

                intent.putExtra("weekDay", text_Saturday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            case R.id.sundayEditText:

                intent.putExtra("weekDay", text_Sunday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            default:
```

```
                break;
        }
 }
```

Наступною розглянемо активність DayActivity, її функціональна задача полягає в тому, щоб відображати всю інформацію про замітку, її головний текст та додатковий текст, а також надавати інструменти для редагування цієї замітки.

Щоб відобразити інформацію з бази даних, подібно як у класі фрагменту WeekFragment створюємо метод displayDatabaseInfo, який також буде працювати з спеціальними класами DBHelper та WeekbookContract, щоб зробити запит та завантажити записи з бази даних та відобразити їх у відповідних елементах, і має подібний код, до однойменного методу з класу WeekFragment.

Далі в методі onCreate класу DayActivity ініціалізуємо елементи інтерфейсу та об’єкт класу DBHelper, після чого викликаємо вищеописаний метод displayDatabaseInfo, щоб відобразити інформацію в проініціалізованих полях

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_day);
        myDBHelper= new DBHelper(this);
        weekDayTextView = (TextView) findViewById(R.id.weekDayTextView);
        weekDayTextView.setText(getIntent().getStringExtra("weekDay"));
        weekDay = getIntent().getStringExtra("weekDay");
        editText = (EditText)findViewById(R.id.editText);
        editTextMainTasks=(EditText)findViewById(R.id.editTextMainTask);

        displayDatabaseInfo();
}
```

Для повноцінної роботи з даними в цій активності реалізовано CRUD-функціонал. Read-функцію виконує вищезгаданий метод displayDatabaseInfo, а інші функції реалізовані за допомогою однойменних методів

```
 private void insert( ) {
        // Gets the database in write mode
        SQLiteDatabase db = myDBHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(RecordsEntry.COLUMN_DATE,
        getIntent().getStringExtra("weekDay") );
        recId = db.insert(RecordsEntry.TABLE_NAME, null, values);
```

```
        Log.d("myLog", "inserted rows id = " + recId);
}


private void update( String shortText, String allText) {
        // Gets the database in write mode
        SQLiteDatabase db = myDBHelper.getWritableDatabase();
        String selection = RecordsEntry.COLUMN_ID + "=?";
        String[] selectionArgs = new String[] {recId.toString()};
        ContentValues values = new ContentValues();
        values.put(RecordsEntry.COLUMN_SHORT_TEXT, shortText);
        values.put(RecordsEntry.COLUMN_ALL_TEXT,allText);
        long updCount= db.update(RecordsEntry.TABLE_NAME,values,
        selection,selectionArgs);
        Log.d("myLog", "updated rows count = " + updCount);
}
```

Для реалізації функції автоматичного збереження записів, при закритті чи згортанні програми використаємо властивість тану програми onPause та перевизначимо відповідний метод таким чином, щоб він зберігав текст запису.

```
@Override
protected void onPause(){
        super.onPause();

update(editTextMainTasks.getText().toString(),editText.getText().toString());

}
```

Також у цієї активності наявний один пункт меню, який відповідає за додавання тегів, тобто після його натискання відкривається активність TagActivity, яка дозволяє редагувати теги. Створення цього пункту меню відбувається в методі onCreateOptionsMenu, а обробка в onOptionsItemSelected.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.day_menu, menu);

        return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {

            case R.id.action_add:
                Intent intent = new Intent(getApplicationContext(),
TagActivity.class);
```

```
                        intent.putExtra("Day", weekDay);
                        // показуємо нове Activity
                        startActivity(intent);
                    return true;

                default:
                    // If we got here, the user's action was not recognized.
                    // Invoke the superclass to handle it.
                    return super.onOptionsItemSelected(item);

        }
}
```

Активність для створення тегів TagActivity, подібно до попередньої DayActivity реалізує CRUD-функціонал і має подібний програмний код. Для виведення вже доданих до запису тегів. Після натискання на кнопку додати виконується процедура insert, а після видалення відповідно delete. Взаємодія активності з БД відбувається аналогічно тому, як і в інших активностях – за допомогою класів DBHelper та WeekbookContract.

Активність для пошуку записів SearchActivity після ініціації пошуку робить запит до бази даних, якщо знайдено записи, які відповідають запиту, то ініціалізуються елементи для відображення цих записів і за допомогою методу getTextForAll в кожен із них відображається дата та текст який їй відповідає викликаючи функцію displayDatabaseInfo, яка працює подібно до попередньо описаних подібних функцій.

```
public void getTextForAll()
    {
        for (EditText findedEditText : EditTextList) {
          displayDatabaseInfo(text, findedEditText);
            }
    }
```

Останньою активністю є BackupActivity. Вона працює використовуючи вбудовані пакети java.io.File та java.util.zip. Перший використовується для імпорту всіх існуючих заміток в звичайний текстовий файл формату .TXT, а другий для створення бекапу всієї бази даних з замітками – тобто її архівації у формат .ZIP.

Вся структура проекту складається з 10 класів та приведена на рисунку 3.1. Всі вище описані класи являються класами представниками, відповідно до моделі MVP. Класи DBHelper та WeekbookContract є частиною моделі, і будуть описані у розділі 3.3. А код представлення буде описано в розділі 3.2.

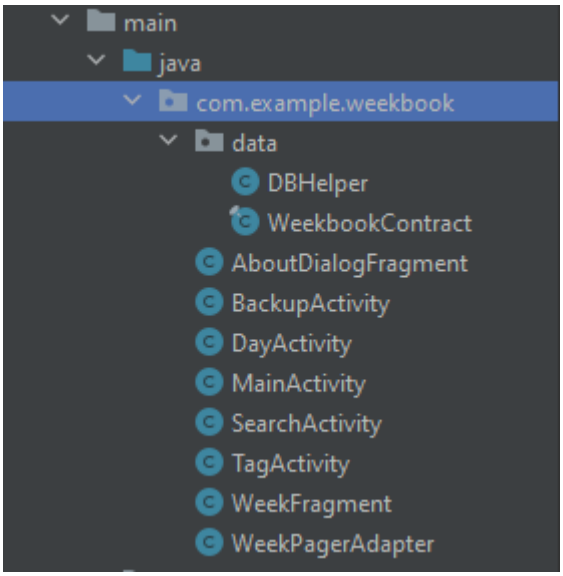


Рисунок 3.1 – Структура проекту

3.2 Реалізація розмітки мобільного додатка

Опис реалізації розмітки варто починати з головної активності додатку. Її розмітка складається з 3 частин. Перша частина це сама активність (рисунок 3.2), на якій розміщено елемент PagerTitleStrip. Саме завдяки цьому елементу можливо реалізувати перегортання тижнів на екрані додатку. І в цей елемент буде вміщуватись фрагмент.

Кодова реалізація виглядає наступним чином:

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

<androidx.viewpager.widget.ViewPager
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <androidx.viewpager.widget.PagerTitleStrip
        android:id="@+id/pager_title_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#686868"
        android:paddingTop="4dp"
        android:paddingBottom="4dp"
        android:textColor="#fff" />
</androidx.viewpager.widget.ViewPager>
```

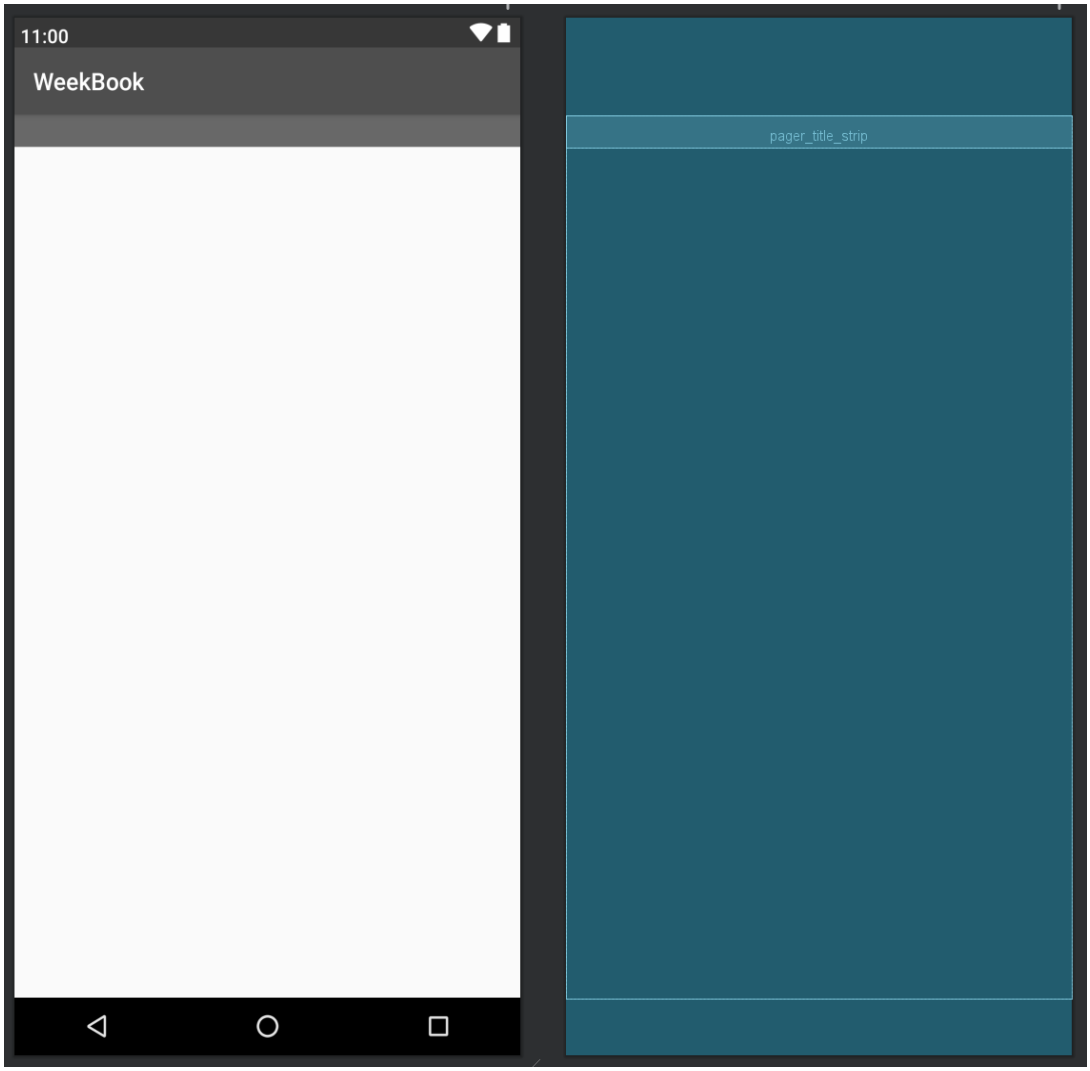


Рисунок 3.2 – Головна активність в конструкторі

Друга частина це фрагмент (рисунок 3.3), який вставляється в елемент PagerTitleStrip головної активності. В цьому фрагменті розміщено 14 елементів TextView , половина з яких використовується для відображення дня тижня та дати, а інша половина – для відображення головних задач днів тижня.

Скомпоновані ці елементи між собою за допомогою 10 контейнерів LinearLayout, які допомагають розміщати елементи пропорційно до розмірів екрану.

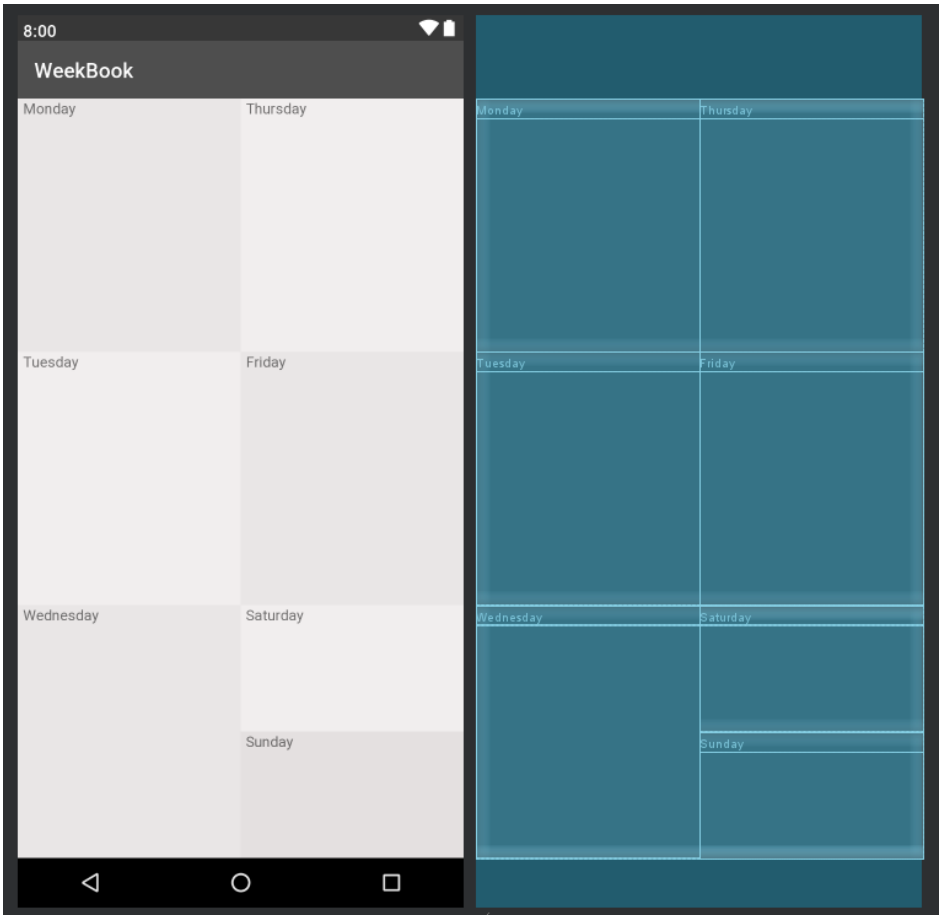


Рисунок 3.3 – Фрагмент головної активності в конструкторі

Повне кодове представлення цього фрагменту наведено в додатку. Нижче приведено структура одного з контейнерів LinearLayout.

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="horizontal">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#32C5C5C5"
            android:orientation="vertical">

            <TextView
                android:id="@+id/textViewWednesday"
```

```
                android:layout_width="match_parent"
                android:layout_height="38dp"
                android:layout_weight="1"
                android:paddingStart="5dp"
                android:text="Wednesday" />

            <TextView
                android:id="@+id/wednesdayEditText"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:baselineAligned="false"
                android:ems="10"
                android:gravity="start|top"
                android:inputType="none"
                android:onClick="goNewView"
                android:textAlignment="textStart"
                android:textSize="16sp" />
</LinearLayout>
```

Останньою частиною головної активності є меню (рисунок 3.4). Воно складається з трьох звичайних пунктів та одного винесеного на головну панель представленого у вигляді іконки

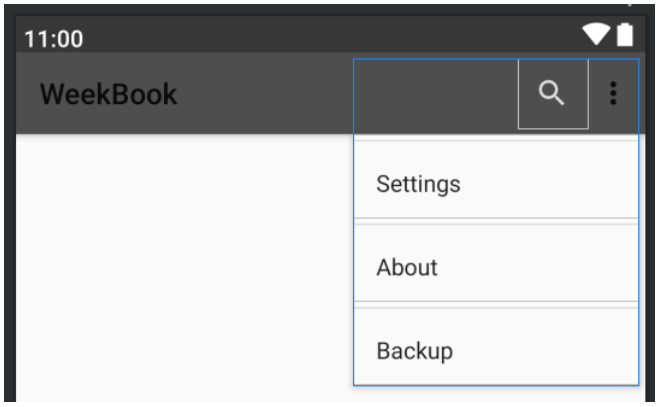


Рисунок 3.4 – Меню головної активності в конструкторі

Код розмітки цього меню виглядає наступним чином:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/settings"
        android:title="Settings" />
    <item
        android:id="@+id/about"
        android:title="About" />
    <item
```

```
        android:id="@+id/action_backup"
        android:title="Backup" />
    <item android:id="@+id/action_search"
        android:title="search"
        android:icon="@drawable/ic_search_white_24dp"
        app:showAsAction="ifRoom|collapseActionView"
        app:actionViewClass="android.widget.SearchView" />

</menu>
```

Наступною розглянемо структуру активності дня, в якій відбувається редагування . Вона складається з двох частин: самої активності та меню.

Активність вміщує в собі три елементи (рисунок 3.5): TextView для відображення дати та два EditText в яких відбувається редагування замітки. Розміри TextView є сталими, а EditText динамічно змінюють розмір в залежності від тексту, який вміщують.

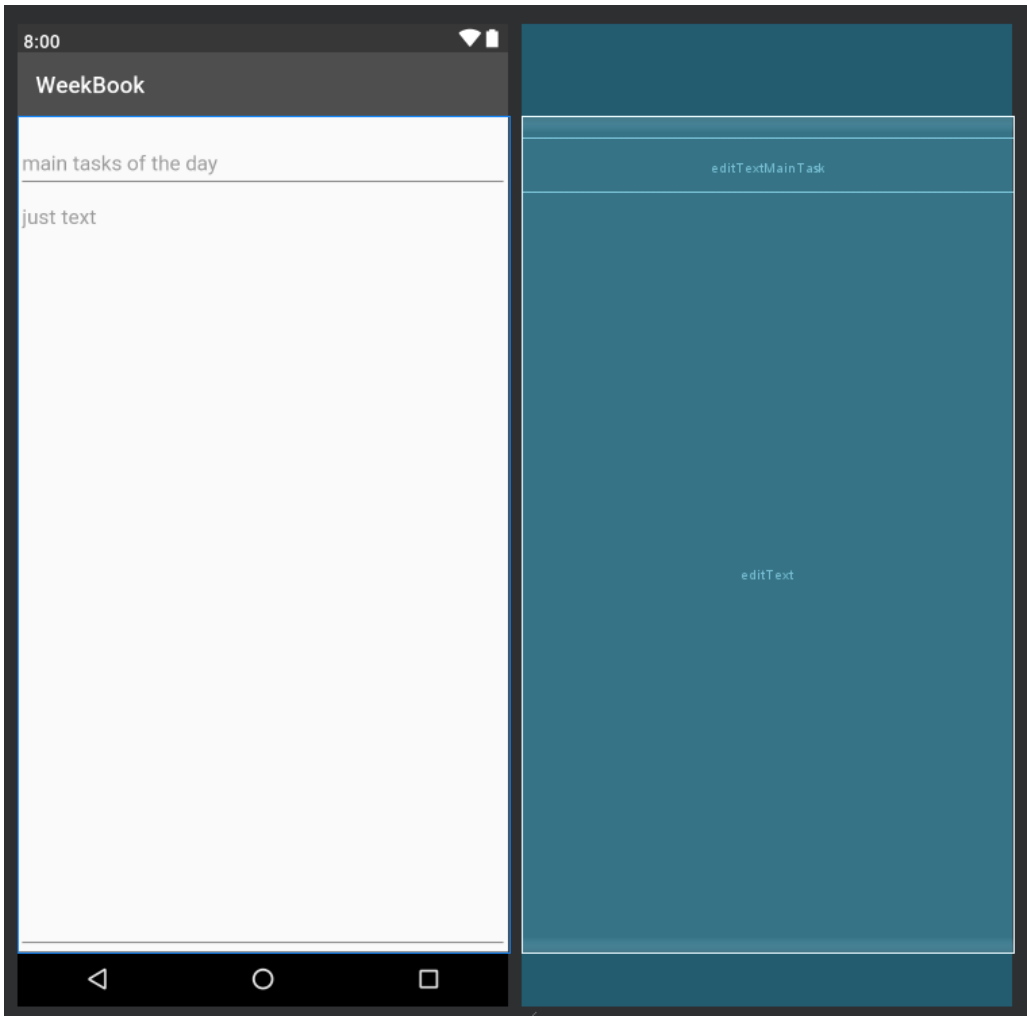


Рисунок 3.5 – Активність дня в конструкторі

Код активності дня:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DayActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/weekDayTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textMultiLine" />

        <EditText
            android:id="@+id/editTextMainTask"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:gravity="start|top"
            android:hint="main tasks of the day"
            android:inputType="textMultiLine"
            android:maxHeight="400dp"
            android:minHeight="47dp" />

        <EditText
            android:id="@+id/editText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:gravity="start|top"
            android:hint="just text"
            android:inputType="textMultiLine" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Меню активності дня (рисунок 3.6) складається з одного пункту, який виведений на головну панель у вигляді іконки плюса.

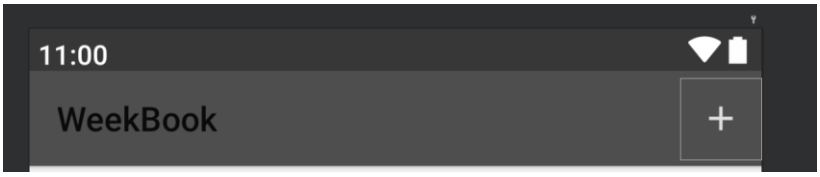


Рисунок 3.6 – Меню активності дня в конструкторі

Код меню активності дня:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_add"
        android:icon="@drawable/ic_add_black_24dp"
        android:title="add"
        app:showAsAction="ifRoom"/>
</menu>
```

Наступною розглянемо активність для додавання тегів. Вона містить 4 різних елементи: TextView для відображення дати, MultiAutoCompleteTextView для введення назви тегу, кнопку додати та ListView для відображення списку доданих тегів.

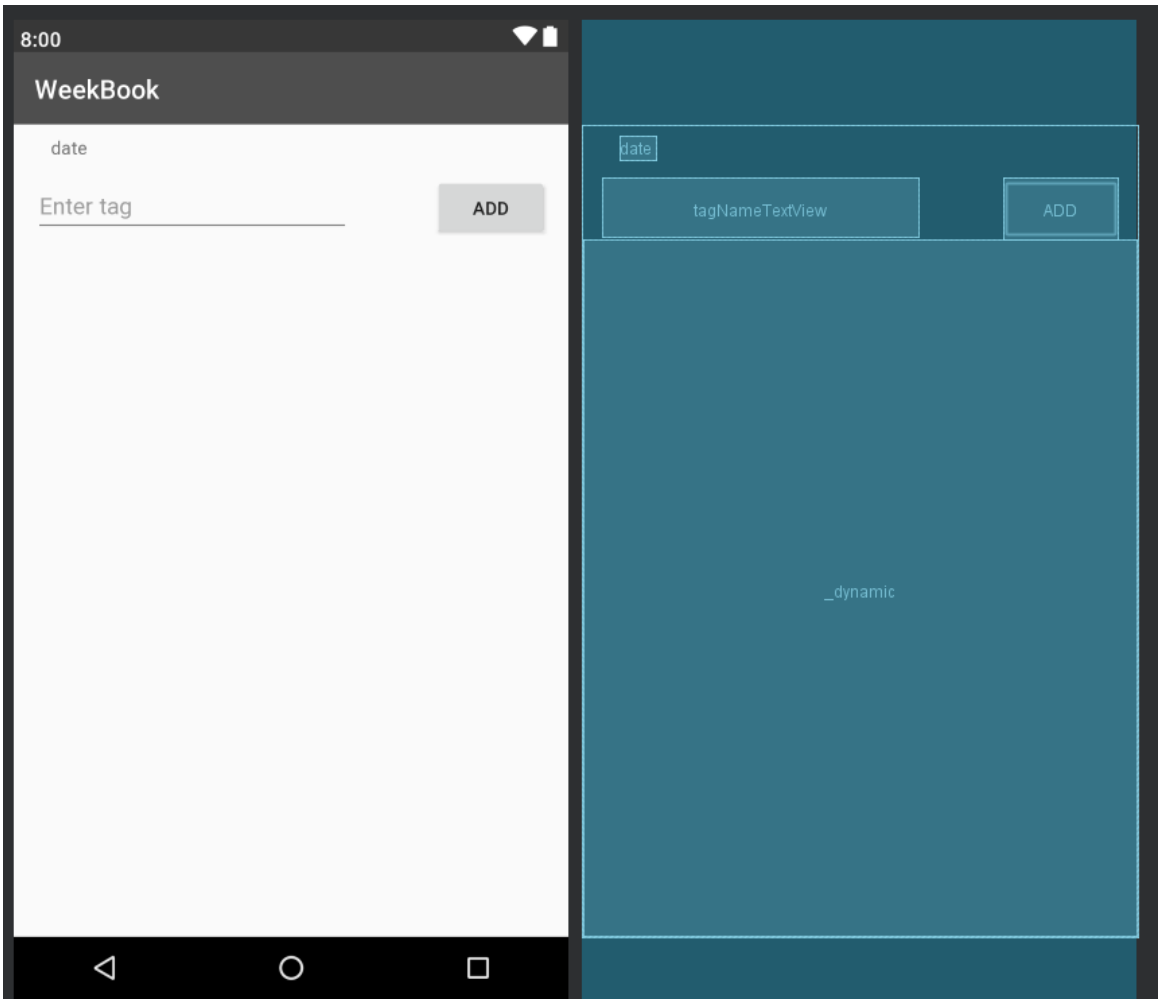


Рисунок 3.7 – Активність для додавання тегів в конструкторі

Код розмітки активності для додавання тегів:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TagActivity">
    <TextView
        android:id="@+id/dayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="14dp"
        android:layout_marginTop="8dp"
        android:text="date"
        app:layout_constraintBottom_toTopOf="@+id/tagNameTextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.037"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <Button
        android:id="@+id/buttonAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:layout_marginEnd="16dp"
        android:text="Add"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toEndOf="@+id/tagNameTextView"
        app:layout_constraintTop_toTopOf="parent" />

    <MultiAutoCompleteTextView
        android:id="@+id/tagNameTextView"
        android:layout_width="244dp"
        android:layout_height="46dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="40dp"
        android:hint="Enter tag"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ListView
        android:layout_width="427dp"
        android:layout_height="537dp"
        android:layout_marginStart="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Активність для пошуку заміток (рисунок 3.8) містить тільки один елемент – спеціальний елемент для пошуку SearchView.

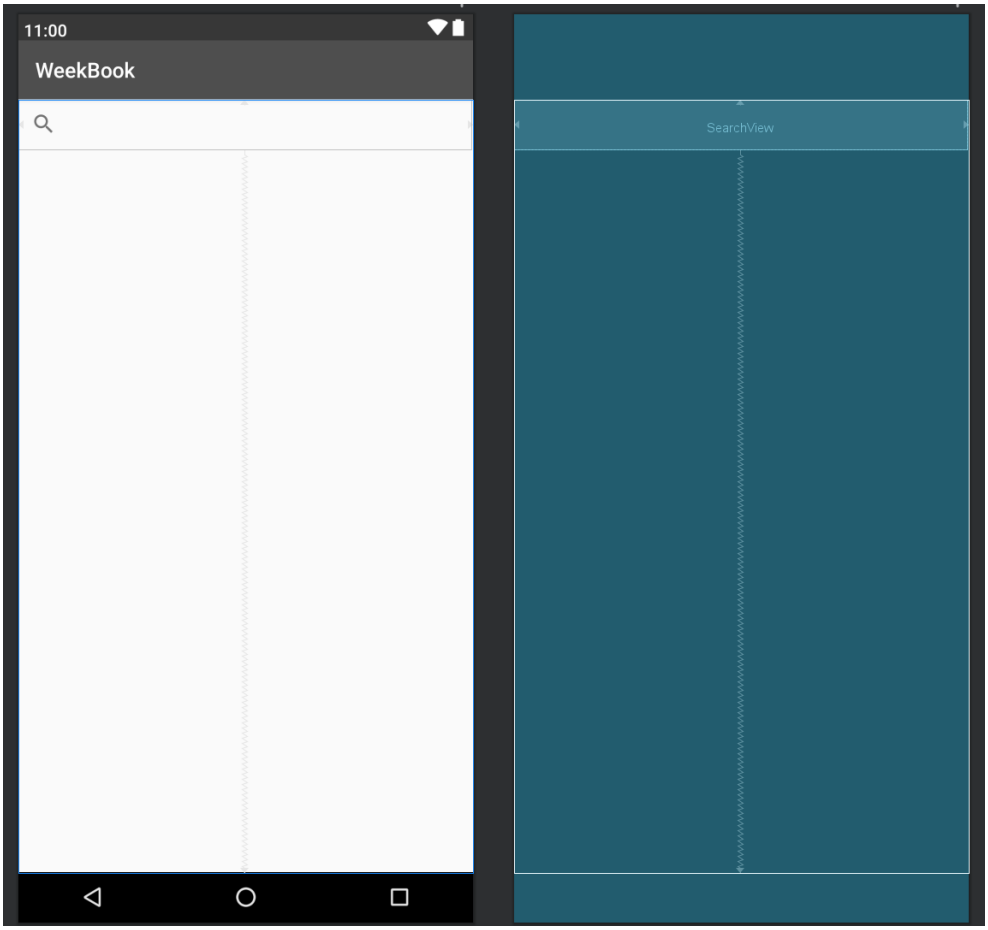


Рисунок 3.8 – Активність для пошуку записів в конструкторі

Код розмітки активності для пошуку:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SearchActivity">

    <SearchView
        android:layout_width="430dp"
        android:layout_height="47dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Останньою є активність для створення бекапу (рисунок 3.9). Вона містить три елементи – три кнопки однакового розміру розміщені одна під одною.

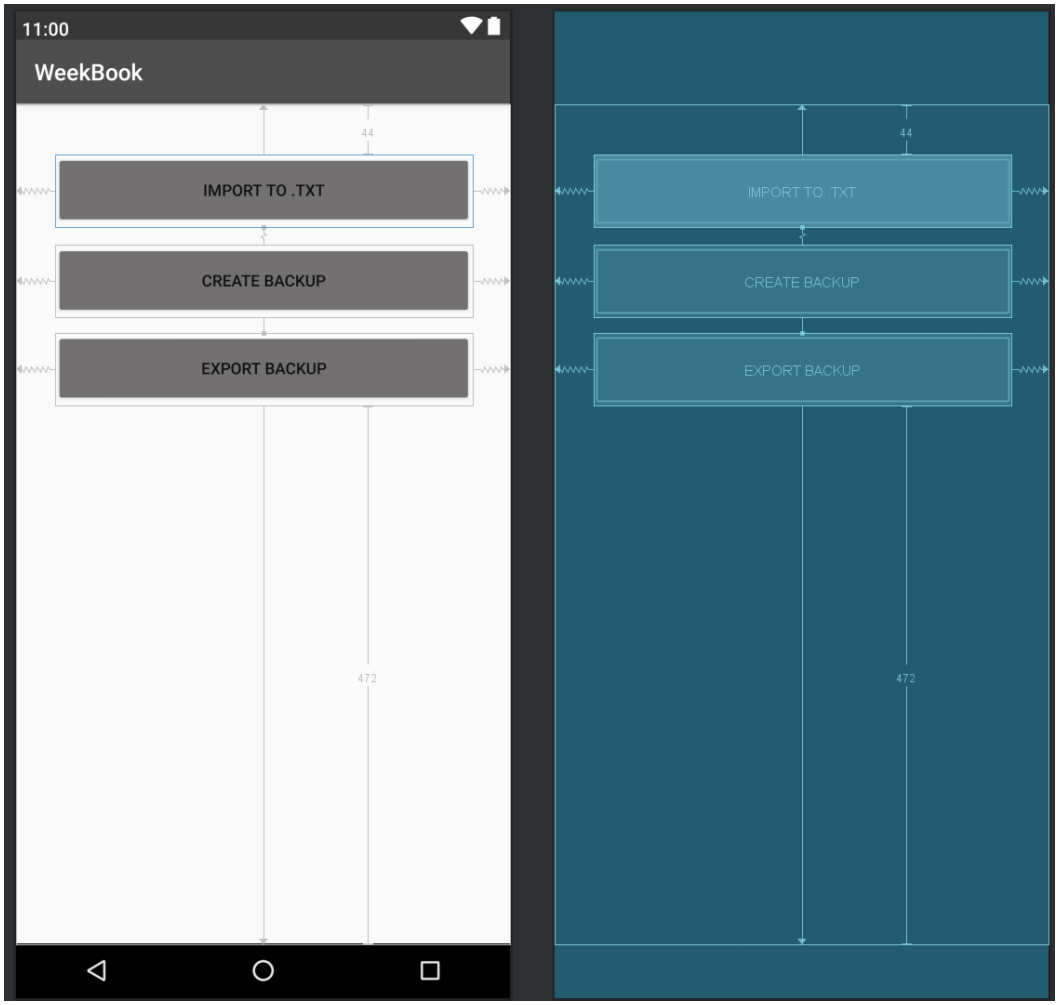


Рисунок 3.9 – Активність для створення бекапу в конструкторі

Код розмітки активності для створення бекапу:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".BackupActivity">
    <Button
        android:id="@+id/buttonTXT"
        android:layout_width="365dp"
        android:layout_height="63dp"
        android:backgroundTint="#727070"
        android:text="Import to .TXT"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.507"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.04" />
    <Button
        android:id="@+id/buttonCreateBackup"
        android:layout_width="365dp"
        android:layout_height="63dp"
        android:layout_marginTop="16dp"
        android:backgroundTint="#727070"
        android:text="Create backup"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.507"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.134" />
    <Button
        android:id="@+id/buttonExportBackup"
        android:layout_width="365dp"
        android:layout_height="63dp"
        android:backgroundTint="#727070"
        android:text="Export backup"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.507"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.278" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.3 Розробка бази даних

Для реалізації було обрано роботу з СКБД SQLite, адже вона є ідеальною для роботи на локальних пристроях, а також в Android існує її вбудована підтримка та надається API-оболонка з сумісним інтерфейсом, що є вагомою перевагою для використання SQLite.

Створюємо структуру бази даних, яка буде відповідати архітектурі, що була спроектована у розділі 2.2. Формальний опис розроблюваної бази даних представлений нижче.

В таблиці БД Записи (табл. 3.1) зберігається окремо текст головних завдань дня, саме цей текст виводиться на головну активність, окремо додатковий текст, що виводиться тільки на активності дня, а також за яку відповідає запис і його унікальний ідентифікатор.

Таблиця 3.1 – Організація таблиці БД Записи(Records)

Назва поля	Тип поля	Обмеження	Коментар
RecordId	INTEGER	NOT NULL PRIMARY KEY AUTOINCREMENT	Ідентифікатор запису
Date	DATETIME	NOT NULL	Дата за яку відповідає запис
ShortText	VARCHAR2(300)	Без обмежень	Головні задачі дня, текст що відображається на сторінці тижня
AllText	VARCHAR2(40)	Без обмежень	Додатковий текст дня

В таблиці БД Теги (табл. 3.2) зберігається назва тегу та його ідентифікатор.

Таблиця 3.2 – Організація таблиці БД Теги(Tags)

Назва поля	Тип поля	Обмеження	Коментар
TagId	INTEGER	NOT NULL PRIMARY KEY AUTOINCREMENT	Ідентифікатор тегу
TagName	VARCHAR2(40)	NOT NULL	Назва тегу

Кожен запис може вміщувати багато тегів, в той же час кожен тег може використовуватись в багатьох записах. Тобто тут використовується відношення багато до багатьох. Для реалізації такого відношення ми використовуємо таблицю БД Теги в Записах (таблиця 3.3), в якій зберігається інформація про те який тег використовується в якому записі.

Таблиця 3.3 – Організація таблиці БД Теги в записах(TagsOnRecords)

Назва поля	Тип поля	Обмеження	Коментар
TagId	INTEGER	NOT NULL Посилання на тег	Ідентифікатор тегу який використовується в записі
RecordId	INTEGER	NOT NULL Посилання на запис	Ідентифікатор запису в якому використовується тег

Взаємодія активностей з БД відбувається через спеціальний клас-контракт та клас-помічник, так як це рекомендує робити Google, в офіційній документації по Android-розробці [19].

В класі-помічнику ми створюємо метод onCreate який викликається при створенні БД. Саме в ньому і відбувається реалізація розробленої структури БД. В методі onUpgrade розміщуємо код, який має виконатись у випадку оновлення версії БД, а саме – видалити існуючі таблиці та створити нові.

Лістинг коду класу-помічника для роботи з БД DBHelper виглядає наступним чином

```
package com.example.weekbook.data;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import  com.example.weekbook.data.WeekbookContract.RecordsEntry;
import  com.example.weekbook.data.WeekbookContract.TagsEntry;
import  com.example.weekbook.data.WeekbookContract.TagsOnRecordsEntry;

public class DBHelper extends SQLiteOpenHelper {
    public static final String LOG_TAG = DBHelper.class.getSimpleName();
    private static final String DATABASE_NAME = "weekbook.db";
    private static final int DATABASE_VERSION = 1;

    /**
     * конструктор
     * @param context
     */
    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    /**
     *Викликається при створенні БД
     */
    @Override
    public void onCreate(SQLiteDatabase db) {
        // Запит на створення
        String SQL_CREATE_RECORDS_TABLE = "CREATE TABLE "
+RecordsEntry.TABLE_NAME+ " ("
                + RecordsEntry.COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,
"
                + RecordsEntry.COLUMN_SHORT_TEXT + " TEXT, "
                + RecordsEntry.COLUMN_ALL_TEXT + " TEXT, "
                + RecordsEntry.COLUMN_DATE + " TEXT NOT NULL );";
        //Запускаємо створення таблиці з записами
        db.execSQL(SQL_CREATE_RECORDS_TABLE);
```

```
        // Запит на створення
        String SQL_CREATE_TAGS_TABLE = "CREATE TABLE " +TagsEntry.TABLE_NAME+ "
("
                + TagsEntry.COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + TagsEntry.COLUMN_TAG_NAME + " TEXT NOT NULL );";
        //Запускаємо створення таблиці з тегами
        db.execSQL(SQL_CREATE_TAGS_TABLE);


        // Запит на створення
        String SQL_CREATE_TAGS_ON_RECORDS_TABLE = "CREATE TABLE "
+TagsOnRecordsEntry.TABLE_NAME+ " ("
                + TagsOnRecordsEntry.COLUMN_RECORD_ID + " INTEGER NOT NULL, "
                + TagsOnRecordsEntry.COLUMN_TAG_ID + " INTEGER NOT NULL, "
                +"FOREIGN KEY ("+TagsOnRecordsEntry.COLUMN_TAG_ID+") REFERENCES
"+TagsEntry.TABLE_NAME
                +"("+TagsEntry.COLUMN_ID+"), "
                +"FOREIGN KEY ("+TagsOnRecordsEntry.COLUMN_RECORD_ID+")
REFERENCES "+RecordsEntry.TABLE_NAME
                +"("+RecordsEntry.COLUMN_ID+" ));";
        //Запускаємо створення зведеної таблиці
        db.execSQL(SQL_CREATE_TAGS_ON_RECORDS_TABLE);
    }

    /**
     *Викликається при оновленні БД
     */
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Запис в журнал
        Log.w("SQLite", "Upgrading from version " + oldVersion + " to version "
+ newVersion);

        // Видаляємо старі таблиці
        db.execSQL("DROP TABLE IF  EXISTS " + RecordsEntry.TABLE_NAME);
        db.execSQL("DROP TABLE IF  EXISTS " + TagsEntry.TABLE_NAME);
        db.execSQL("DROP TABLE IF  EXISTS " + TagsOnRecordsEntry.TABLE_NAME);
        // Створюємо нові
        onCreate(db);
    }
}
```

В класі-контракті WeekbookContract розміщується сама структура БД, де таблиці записуються в вигляді статичних класів з статичними полями, які відповідають кожному полю цієї таблиці

```
package com.example.weekbook.data;

import android.provider.BaseColumns;


public final class WeekbookContract {
    private WeekbookContract(){}
    //таблиця з записами
    public static final class RecordsEntry implements BaseColumns {
        public static final String TABLE_NAME = "records"; // назва таблиці в бд
        // назва стовбців
```

```
        public static final String COLUMN_ID = "_id";
        public static final String COLUMN_SHORT_TEXT = "shortText";
        public static final String COLUMN_ALL_TEXT = "allText";
        public static final String COLUMN_DATE = "date";
    }

    //таблиця з тегами
    public static final class TagsEntry implements BaseColumns {
        static final String TABLE_NAME = "tags"; // назва таблиці в бд
        // назва стовбців
        public static final String COLUMN_ID = "_id";
        public static final String COLUMN_TAG_NAME = "tagName";
    }

    //зведена таблиця(допоміжна) для створення зв'язку багато до багатьох
    public static final class TagsOnRecordsEntry implements BaseColumns {
        static final String TABLE_NAME = "TagsOnRecords"; // назва таблиці в бд
        // назва стовбців
        public static final String COLUMN_TAG_ID = "TagId";
        public static final String COLUMN_RECORD_ID = "RecordId";
    }
}
```

3.4 Керівництво користувача

Для початку роботи з додатком необхідно завантажити його на пристрій, після чого потрібно запустити програму, натиснувши на її іконку на робочому столі смартфону. Після цього відкриється головна сторінка застосунку, на якій буде відображено інформацію про поточний тиждень.

В верхній частині екрану розміщено панель, яка містить назву додатку та пункт пошуку та меню. Нижче розміщено лінію з вказанням тижня який зараз відображається: поточний, наступний, минулий, тощо. Під лінією розміщено сім панелей, на яких відображаються дати з вказанням дня тижня, якому відповідає кожна з цих панелей.

При натисканні на одну з панелей відкриється нова сторінка, яка буде відповідати запису за день, панель якого було натиснута.

На цій сторінці розмішається головна панель з меню, під нею дата дня за який відповідає запис, а нижче два текстових поля, натиснувши на які можна додавати та редагувати текст. Верхнє з цих полів відповідає за головні задачі для, і буде відображатись на головній сторінці додатку у полі відповідного дня. Те

поле, що розташоване під ним відповідає за додатковий текст дня, і буде відображатись тільки на сторінці цього дня і не буде переноситись на головну сторінку тижня.

В верхньому правому куті головної панелі розміщена іконка у вигляді плюса, після натискання на яку відкриється сторінка додавання тегів. На цій сторінці є поле для введення тексту та кнопка. В поле необхідно вводити тег, який відповідає дню з кого була відкрита сторінка додавання тегів. Після введення необхідно натиснути кнопку додати, після чого тег відобразиться нижче, у списку доданих тегів, там будуть з'являтись всі теги, що належать цьому запису. Якщо натиснути на вже доданий тег, то з'явиться вікно, З запитом підтвердження видалення цього тегу.

Для пошуку по записам необхідно на головній сторінці додатку натиснути на іконку з зображенням збільшуваної лупи, після чого з'явиться поле, куди необхідно вводити шуканий текст та натиснути кнопку шукати. Після цього відкриється сторінка з результатами пошуку, у випадку, якщо нічого не знайдено відобразиться відповідний напис. Якщо записи, які відповідають пошуковому запиту існують вони відображуються у вигляді списку плиток з датою запису та головними задачами на цей день. Щоб перейти до самого запису необхідно натиснути на його плитку, і тоді відкриється сторінка дня.

Для створення бекапу необхідно на головній сторінці додатку натиснути на три крапки, тоді з'явиться меню, в якому необхідно обрати пункт «створити бекап». Після цього відкриється сторінка створення резервної копії, експорту файлів та імпорту резервної копії. з трьома кнопками. При натисканні першої кнопки створиться текстовий файл куди будуть перенесені всі створені записи додатку з всіма доданими до них тегами та датами записів. Після натискання другої, буде створено архів з резервною копією всіх записів. Після натискання третьої відкриється сторінка вибору файлу, на якій потрібно знайти архів з бекапом та обрати його. Після чого всі записи які були збережені в архіві відобразяться в додатку.

3.5 Технічні характеристики мобільного додатка

В додатку використовуються прості рішення і нема складних обчислень, тому він не є вибагливим до технічних характеристик пристрою.

Мінімальна конфігурація:

- Android 4.4 та вище;
- частота процесора 500 МГц;
- 500 МБ ОЗП;
- 40 МБ вільної пам'яті.

Конфігурація, що рекомендується:

- Android 7.0;
- частота процесора 500 МГц;
- 1 ГБ ОЗП;
- 100 МБ вільної пам'яті.

Розроблений додаток має весь необхідний функціонал, з робочими активностями та приємним інтерфейсом, який є простим, зручним та інтуїтивно зрозумілим у використанні.

У цьому розділі була виконана повна програмна реалізація додатку, розроблено базу даних та інтерфейс програми, які відповідають спроектованим у попередньому розділі, а також складено керівництво користувача та технічні вимоги додатку. Повий код програми наведений у додатку Б.

4 ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКА

4.1 Аналіз методів тестування мобільного додатка

Тестування ПЗ – це процес дослідження та випробовування програмного продукту, метою якого є перевірка відповідності реальної поведінки програми і її очікуваної поведінки.

На сьогоднішній день існує досить велика кількість методів тестування мобільних додатків. Основні з них :

- функціональне тестування;
- модульне тестування;
- тестування навантаження;
- конфігураційне тестування;
- тестування безпеки додатку;
- юзабіліті-тестування.

Всі ці види тестування є важливими та необхідними три розробці ПЗ. Проте важливо розуміти, з якою метою проводиться те чи інше тестування, адже якщо тестування проводиться тільки для того щоб провестись, то краще його взагалі не проводити. Тому розглянемо кожен з видів тестування детальніше, і дослідимо з якою метою проводиться кожне з них.

Функціональне тестування додатку проводиться для того, щоб перевірити чи правильно функціонує застосунок, тобто чи так як задумано в записано в ТЗ. Також цей метод тестування часто називають методом «чорного ящика». Він допомагає перевірити чи правильно розробник зрозумів побажання замовника. Всі функціональні вимоги беруться з технічного завдання, та по ним проводиться тестування та створюються тест-кейси.

Тест-кейси – це сценарії поведінки користувачів в додатку. Весь процес тестування будується на перевірці кейсів. Для правильного створення тест-кейсів потрібно розуміти бізнес-функціональність додатку та те, хто є цільовою аудиторією додатку.

Сценаріїв функціонального тестування досить багато. Наприклад:

– перевірка коректності роботи полів;

– перевірка логіки переходів по екранах;

– перевірка коректності роботи всіх кнопок;

– перевірка коректності взаємодії з соцмережами;

– перевірка підтримки транзакцій через системи онлайн-оплати;

– перевірка коректності установки програми та його роботи на всіх передбачених пристроях.

Модульне тестування – метод тестування окремих частин продукту, модулів – зазвичай окремих функцій та методів. Зазвичай таке тестування є першим етапом тестування системи, на якому кожен модуль програми перевіряється окремо. Такий метод тестування зазвичай називають методом «білої скриньки», адже відповідальна особа знає внутрішню структуру програми та її очікувані поведінку.

Тестування навантаження – метод тестування, який використовується для перевірки коректності функціонування додатку при різному навантаженні на систему, наприклад при різній кількості користувачів, при різному типі підключення до інтернету. Такий тип тестування дозволяє знайти ділянки застосунку, які можуть гальмувати його роботу та переконатися, що програма не з'їдає всю батарею смартфона. Важливість тестування навантаження неможливо переоцінити, адже якщо додаток не впорається з навантаженням і почне гальмувати та глючити – користувачі будуть дуже невдоволені, і можливо буде втратити їх значну частину.

Зазвичай цей тип тестування проводиться в 3 етапи:

– збір інформації про систему;

– створення моделей навантаження;

– запуск тестів.

Конфігураційне тестування – це процес перевірки чи коректно працює мобільний додаток на різних пристроях. Зазвичай перед конфігураційним

тестуванням готується матриця покриття, куди заносять всі необхідні конфігурації. Далі їх сортують за пріоритетністю і перевіряють в першу чергу важливі варіанти. Так як перевірити функціонування і відображення на всіх пристроях і при будь-яких умовах практично неможливо – потрібно зрозуміти, чим можна пожертвувати, а краще, як знайти оптимальний компроміс.

Пріоритети виставляються в залежності від тих пристроїв, користувачі яких є цільовою аудиторією додатку. Після того як перевірено пріоритетні конфігурації, поступово перевіряються і всі інші. Додаток тестують відповідно до технічного завдання:

– на різних гаджетах: планшети, смартфони, десктопи тощо;
– в різних конфігураціях: типи процесора, роздільність екрана, ОЗП;
– на різних версіях операційних систем iOS, Android;
– в різних типах мережі: GSM, Wi-Fi.

Тестування безпеки допомагає зрозуміти, чи зроблено все можливе для того, щоб захистити дані від загроз. Тобто по-суті, перевіряється стійкість додатки до різним загрозам безпеки: DoS-атак, вірусів, крадіжки даних. Процес тестування безпеки додатку складний і багатогранний. Для різних цілей використовуються різні методи: від експертних аудитів до імітації дій зловмисників автоматизованим шляхом.

Варіанти сценаріїв тестування:

– перевірка захисту даних користувача від мережевих атак;
– перевірка обов'язкової аутентифікації при доступі до контенту;
– захист додатку від хакерів і атак;
– пошук і усунення некерованого коду;
– контроль криптографічних кодів;
– перевірка захисту бізнес-логіки додатку і т.д.

Юзабіліті тестування допомагає зрозуміти як користувачі взаємодіють з інтерфейсом. Юзабіліті – це властивість інтерфейсу, яке або допомагає взаємодії користувачів з додатком, або ускладнює його.

Існують такі методи юзабіліті тестування:

– експертний аналіз;

– тестування за допомогою веб-аналітики;

– тестування за участі живих користувачів зі спеціальних сервісів;

– Живе тестування, з запрошенням користувачів, що належать до цільової аудиторії додатку.

Для перевірки додатку доцільно використати 3 методи: функціональний, модульний та конфігураційний. Адже саме ці типи тестування дозволять виконати повноцінну перевірку додатку, а саме перевірити відповідність ТЗ, роботу окремих функцій та роботу додатку на різних пристроях.

4.2 Тестування додатка за допомогою емулятора

Зазвичай тестування програм починають з модульного тестування. Тому визначимо потенційні сценарії роботи з додатком та згрупуємо їх у таблиці 4.1.

Таблиця 4.1 – Тестові сценарії модульних тестів

Іденти-фікатор	Активність	Модуль	Вхідні дані	Очікуваний результат
1	2	3	4	5
М-М-1	MainActivity	Елемент дня	Натискання на елемент дня, Дата	Відкриття відповідної активності дня
М-М-2	MainActivity	Дата	Сьогоднішня дата	Дата понеділка поточного тижня
М-М-3	MainActivity	Меню	Вибір пункту меню пошук	Відкриття активності для пошуку
М-М-4	MainActivity	Дата	Вибір пункту меню бекап	Відкриття активності для створення бекапу
М-М-5	MainActivity	Фрагмент	Перегортання фрагменту на 1 позицію вправо	Зміна записів поточного тижня на записи минулого тижня

Кінець таблиці 4.1

1	2	3	4	5
D-M-1	DayActivity	Поле для введення головних задач	Текст	Занесення тексту в БД, відображення тексту на головній активності в елементі цього дня
D-M-2	DayActivity	Поле для введення додаткового тексту	Текст	Занесення тексту в БД
D-M-3	DayActivity	Меню	Вибір пункту меню додати тег	Відкриття активності для додання тегів
T-M-1	TagActivity	Додавання тегу	Назва тегу	Занесення тегу в відповідну таблицю БД та створення зв’язку з записом
T-M-2	TagActivity	Видалення тегу	Назва тегу	Видалення тегу в відповідну таблицю БД та його зв’язку з записом
S-M-1	SearchActivity	Пошук запису	Текст, що потрібно знайти	Записи, що містять шуканий текст
B-M-1	BackupActivity	Імпорт записів в .TXT	Всі записи з БД	Текстовий файл з всіма записами та їх датами
B-M-2	BackupActivity	Створення бекапу	БД	Архів БД
B-M-3	BackupActivity	Імпорт бекапу	Архів БД	Записи з архіву перенесені в БД

По створеному сценарію були розроблені unit-тести. Нижче приведений фрагмент коду для тесту M-M-1:

```
@Test
public void testCheckDayActivityShowed() {
        onView(withId(R.id.MondayTextView))
        .perform(click())
        .check(DayActivity(isDisplayed()));
}
```

Для проведення конфігураційних тестів було використано емулятори: Samsung Galaxy Core Prime, Nexus 5, Xiaomi Redmi Note 4, Samsung M30, Nexus7, POCO X3PRO, Samsung A50.

Далі проведемо функціональне тестування, для цього створимо сценарії тестування опираючись на функціональні вимоги описані в ТЗ програми, та опишемо їх у таблиці 4.2.

Таблиця 4.2 – Тестові сценарії функціональних тестів

Іденти-фікатор	Функціональна вимога	Вихідні дані	Очікуваний результат
1	2	3	4
FT-1	Перегляд головних задач тижня	Користувач відкриває додаток на головній сторінці тижня	Йому відображаються головні задачі кожного з днів тижня
FT-2	Відкриття задачі кожного дня окремо	Користувач на головній сторінці натискає на плитку дня	Відкривається активність дня де відображаються головні задачі на день та додатковий текст цього дня
FT-3	Можливість написати для кожного дня текст, який не буде відображатись на сторінці тижня	1.Користувач на головній сторінці натискає на плитку дня 2.Користувач вводить текст в нижнє поле	1.Відкривається активність дня де відображаються головні задачі на день та додатковий текст цього дня. 2.Текст зберігається та заноситься в БД, і відображається тільки на сторінці дня.
FT-4	Перегляд старих записів	Користувач на головній сторінці робить свайп вправо	На плитках замінюються записи поточного тижня на записи минулого тижня
FT-5	Створення записів на перед	1.Користувач на головній сторінці робить свайп вліво 2.Користувач натискає на плитку понеділка.	1.На плитках замінюються записи поточного тижня на записи наступного тижня 2.Відкривається активність дня де відображаються записи наступного понеділка

Кінець таблиці 4.2

1	2	3	4
FT-6	Пошук записів за ключовими словами та тегами	1.Користувач на головній сторінці натискає на меню пошуку 2.Користувач вводить пошуковий запит з тегом	1.Відкривається активність для пошуку записів. 2.Користувачу відображаються запити з введеним тегом
FT-7	При випадковому згортанні записані дані зберігаються	1.Користувач на головній сторінці натискає на плитку дня. 2.Користувач вводить текст в нижнє поле. 3.Користувач згортає додаток	1.Відкривається активність дня де відображаються головні задачі на день та додатковий текст цього дня. 2,3.Текст зберігається та заноситься в БД
FT-8	при виході з програми записані дані автоматично зберігаються	Користувач закриває додаток	Нічого не відбувається, адже записи були збережені раніше
FT-9	створення бекапу в пам’яті телефону	1.Користувач натискає на пункт меню створити бекап 2.Користувач натискає на кнопку створити бекап	1. Відкривається активність для створення бекапу. 2.БД додатку зберігається на пристрій у вигляді архіву

Зазвичай функціональне тестування проводять на емуляторі чи реальному пристрої. Я вирішила створити декілька емуляторів, що імітують різні пристрої з різними версіями Android та різними діагоналями екрану та сумістити функціональне і конфігураційне тестування. Тому провела на емуляторах, створених для проведення конфігураційних тестів, всі функціональні тести.

Результати проведеного тестування та їх аналіз описано у наступному підрозділі.

4.3 Аналіз результатів тестування мобільного додатка

Під час тестування додатка за допомогою unit-тестів та емулятора було знайдено і виправлено наступні баги:

– несправне відображення днів тижня;

– створення дублікатів в БД;

– при перегортанні фрагментів і відкритті дня якогось тижня відкривався не обраний день, а день з фрагменту який завантажувався останнім.

Після виправлення вищеперерахованих багів всі етапи модульного тестування пройшли успішно та було отримано наступні результати тестування, що занесені в таблицю 4.3.

Таблиця 4.3 – Результати модульного тестування

Іденти-фікатор	Модуль	Вхідні значення	Отриманий результат	Результат
1	2	3	4	5
M-M-1	Елемент дня	Натискання на елемент дня, понеділок 10.05.21	Відкриття активності дня, яка відповідає за понеділок 10.05.21	Правильно
M-M-2	Дата	12.05.21	10.05.21	Правильно
M-M-3	Меню	Вибір пункту меню пошук	Відкриття активності для пошуку	Правильно
M-M-4	Дата	Вибір пункту меню бекап	Відкриття активності для створення бекапу	Правильно
M-M-5	Фрагмент	Перегортання фрагменту на 1 позицію вправо	Зміна записів поточного тижня на записи минулого тижня	Правильно
D-M-1	Поле для введення головних задач	«купити корм коту»	В БД занесено запис, відображення тексту на головній активності в елементі цього дня	Правильно
D-M-2	Поле для введення додаткового тексту	«рецепт кексів»	Занесення тексту в БД	Правильно

Кінець таблиці 4.3

D-M-3	Меню	Вибір пункту меню додати тег	Відкриття активності для додання тегів	Правильно
T-M-1	Додавання тегу	«покупки»	Тег занесено в відповідну таблицю	Правильно
T-M-2	Видалення тегу	«покупки»	Тег видалено в відповідну таблицю БД	Правильно
S-M-1	Пошук запису	«корм»	Виведено запис «купити корм коту»	Правильно
B-M-1	Імпорт записів в .TXT	Всі записи з БД	Текстовий файл з всіма записами та їх датами	Правильно
B-M-2	Створення бекапу	БД	Архів БД	Правильно
B-M-3	Імпорт бекапу	Архів БД	Записи з архіву перенесені в БД	Правильно

Далі було проведено функціональні тести на створених емуляторах пристроїв. Отримані результати було занесено до таблиці 4.4.

Таблиця 4.4 – Результати функціонального та конфігураційного тестування, проведеного на емуляторах

Назва пристрою, емулятор якого використовувався	Ідентифікатор тесту								
	FT-1	FT-2	FT-3	FT-4	FT-5	FT-6	FT-7	FT-8	FT-9
Samsung Galaxy Core Prime	+	+	+	+	+	+	+	+	+
Nexus 5	+	+	+	+	+	+	+	+	+
Xiaomi Redmi Note 4	+	+	+	+	+	+	+	+	+
Samsung M30	+	+	+	+	+	+	+	+	+
Nexus7	+	+	+	+	+	+	+	+	+
POCO X3PRO	+	+	+	+	+	+	+	+	+
Samsung A50	+	+	+	+	+	+	+	+	+

Проаналізувавши результати проведених тестів за допомогою різних методів тестування виявлено, що функціонал додатку реалізовано відповідно до вимог складених у технічному завданні, і програма є повністю працездатною та оптимізованою до роботи на різних пристроях.

В цьому розділі описано процес тестування додатку, а саме – проведення функціонального, модульного та конфігураційного тестування. Отримані результати було проаналізовано, та зроблено висновок про те, що розроблений додаток є повнофункціональним та готовим до роботи на різних пристроях.

ВИСНОВКИ

При виконанні дипломного проекту було проведено аналіз предметної області, її функціональних та структурних особливостей, визначено особливості та специфіку процесу організації часу та створення списків задач. Як результат було прийняте рішення про необхідність розробки додатку який задовольнить потреби користувачів і зможе повністю замінити паперові блокноти.

Після цього було проведено детальний аналіз існуючих на ринку рішень, виконано їх порівняння між собою та зроблено висновок про те, що всі вони однотипні, переповнені рекламою, мають недостатній функціонал, та незручний спосіб відображення заміток.

Далі було визначено функціональні задачі додатку, на основі яких було створено функціональні та нефункціональні вимоги. Для якісного виконання поставлених задач було розроблено технічне завдання.

Наступним кроком було проведення аналіз інструментів та технологій, що використовуються для створення Android додатків та порівняння різних методів для запису та збереження інформації. Опираючись на результати проведеного аналізу було розроблено архітектуру додатка та спроектовано сучасний та функціональний інтерфейс.

На основі спроектованої архітектури та інтерфейсу, виконано програмну реалізацію мобільного застосунку. Після розробки було проведено модульне, функціональне та конфігураційне тестування готового програмного продукту.

В результаті цього в розробленому додатку присутні всі функції, які були визначені в технічному завдання, та він готовий для використання.

При виконанні дипломного проекту було розроблено додаток для мобільних пристроїв з операційною системою Android, який допомагає вирішити проблему тайм-менеджменту, правильної організації задач та записів, а також зручного пошуку та зберігання попередніх записів, завдяки чому було закріплено теоретичні та практичні знання і навички розробки життєвого циклу ПЗ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Л. П. Бедратюк. Дипломний проект : методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун, О. М. Яшина. – Хмельницький: ХНУ, 2020. – 77 с.
2. В. А. Шеховцов. Операційні системи. – К.: Видавнича група BHV, 2005. – 576с.
3. Carroll R. Bullet Journal [Електронний ресурс] / Ryder Carroll // Офіційний сайт Bullet Journal. – Режим доступу до ресурсу: https://bulletjournal.com. (дата звернення – 10.03.2021). – Назва з екрану.
4. Carroll, Ryder. The Bullet Journal method: track your past, order your present, plan your future. – New York: Portfolio/Penguin, 2018. – 310c.
5. My Diary [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=mydiary.journal.diary.diary withlock.diaryjournal.secretdiary. (дата звернення – 17.03.2021). – Назва з екрану.
6. Universum [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=ru.schustovd.diary. (дата звернення – 18.03.2021). – Назва з екрану.
7. Pi Journal [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=diary.journal.app. (дата звернення – 18.03.2021). – Назва з екрану.
8. SplenDO [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.splendapps.splendo. (дата звернення – 19.03.2021). – Назва з екрану.
9. MyNotes [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=net.kreosoft.android.mynotes. (дата звернення – 19.03.2021). – Назва з екрану.

10. Diary [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=journal.notebook.memoir.write.diary. (дата звернення – 20.03.2021). – Назва з екрану.

11. Notes [Електронний ресурс] / Play Market – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.gcteam.tonote. (дата звернення – 20.03.2021). – Назва з екрану.

12. Systems and software engineering — Architecture description: ISO/IEC/IEEE 42010

13. Potel M. MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java / Mike Potel., 1996.

14. Введение в VIPER (перевод) [Електронний ресурс]. / Habr. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/post/273061/. (дата звернення – 05.04.2021). – Назва з екрану.

15. Information technology — Vocabulary — Part 1: Terms and definitions: ISO/IEC 2382:2015

16. Мейер,Д. Теория реляционных баз данных / Д.Мейер. – Москва: Мир, 1987. – 608 с

17. Codd, E.F. "Further Normalisation of the Data Base Relational Model", 34 p.

18. 8 языков программирования для Android-разработчика [Електронний ресурс] / Tproger. – Режим доступу до ресурсу: https://tproger.ru/articles/8-jazykov-programmirovanija-dlja-android-razrabotchika/.(дата звернення – 29.04.2021). – Назва з екрану.

19. Офіційна документація для android-розробки [Електронний ресурс] / Developers. – Режим доступу до ресурсу: https://developer.android.com/ docs. (дата звернення – 29.04.2021) – Назва з екрану.

.

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки мобільного додатку для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії

Умовне позначення системи: електронний щоденник «WeekBook

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії.

2 Призначення розробки

Додаток призначений для використання на смартфонах з операційною системою Android з метою надання зручного інструменту для організації часу, буденних справ та думок.

Користувачами ПЗ є звичайні користувачі телефонів на платформі Android.

Експлуатаційне призначення, система може використовуватися на будь-якому гаджеті з ОС Android версії вище 4.4, попередньо встановивши додаток, додаткових налаштувань для початку роботи застосунок не потребує.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Перелік функціональних можливостей, які має надавати програма:

- перегляд головних задач тижня ;
- відкриття задачі кожного дня окремо;
- можливість написати для кожного дня текст, який не буде відображатись на сторінці тижня;
- перегляд старих записів;
- створення записів на перед;
- пошук записів за ключовими словами та тегами;
- при випадковому згортанні записані дані зберігаються;
- при виході з програми записані дані автоматично зберігаються;

– імпорт резервної копії;

– експорт даних в файл;

– створення резервної копії в пам’яті телефону.

Вимоги до інтерфейсу:

– стримані або базові кольори (сірий, білий, блакитний тощо);

– мінімалістичні іконки

– при відкритті додатку має відображатись сторінка з замітками на дні поточного тижня;

– дні поточного тижня мають розміщуватись на головному екрані ергономічно та інтуїтивно зрозуміло (як приклад: розміщення днів тижня в шкільному щоденнику);

– при натисканні на плитку дня тижня має відкриватись сторінка обраного дня, на якій відображається головні задачі, та додаткове поле для тексту, який не виводиться на головну сторінку;

– на головній сторінці має бути перехід до сторінки для пошуку заміток по тегам та часині тексту замітки;

– при перегортанні сторінки вправо-вліво на головній сторінці поточний тиждень має змінюватись на попередній та наступний відповідно.

3.2 Вимоги до надійності

Електронний щоденник «WeekBook» повинен забезпечити наступні вимоги до надійності:

– збереження всіх записів у випадку ненавмисного закриття додатку;

– збереження інформації у випадку згортання додатку;

– передбачити блокування некоректних дій користувача при роботі з додатком.

Для забезпечення якості додатку, він обов’язково має пройти модульне, конфігураційне, функціональне та інтеграційне тестування.

3.3 Умови експлуатації

Умови експлуатації повинні відповідати санітарним і технічним нормам експлуатації мобільного пристрою, при температурі та відносній вологості навколишнього середовища, визначених розробником гаджету.

Додаток може використовуватись на смартфонах та планшетах.

Кожен додаток буде використовуватись тільки одним користувачем на його особистому пристрої.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна коректно працювати на пристроях з версією Android вище 4.4

Мінімальна конфігурація:

- Android 4.4 та вище;
- частота процесора 500 МГц;
- 500 МБ ОЗП;
- 40 МБ вільної пам'яті.
- Конфігурація, що рекомендується:
- Android 7.0;
- частота процесора 500 МГц;
- 1 ГБ ОЗП;
- 100 МБ вільної пам'яті.

3.5 Вимоги до інформаційної та програмної сумісності

Програма повинна працювати під управлінням операційної системи андроїд версії вище 4.4, а також похідних від неї системах від виробників смартфонів. У разі використання СКБД - використовувати SQLite, або сумісну систему

3.6 Вимоги до програмної документації

Попередній склад програмної документації встановлений відповідно до ДСТУ 3008–95 та Єдиній системі програмної документації. Нижче перерахований список програмних документів і їх зміст:

- структурна схема системи;
- код програми – запис коду програми з всіма необхідними поясненнями і відповідними коментарями;

– опис програми – відомості про логічну і фізичну модель, відомості про функціонування програми;

– програма і методика випробувань – вимоги, що підлягають перевірці при випробуванні програми, також порядок і методи контролю;

– технічне завдання – цей документ;

– загальний опис алгоритму або функціонування програми, а також обґрунтування ухвалених технічних і техніко-економічних рішень;

– інструкція користувача.

3.7 Стадії та етапи розробки

Стадії та етапи розробки проекту «мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії» наведено в таблиці .

Таблиця А.1 – Стадії та етапи розробки проекту

Стадії розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.21 – 31.01.21	Обґрунтування необхідності розробки програми	Коротка характеристика додатку; підстава і призначення розробки; функціональні вимоги і вимоги до інтерфейсу документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.21 – 14.02.21	Розробка ескізного проекту	Розробка структури бази даних; вибір середовища розробки; розробка і опис загальної алгоритмічної структури системи, що буде розроблюватися

Кінець таблиці А.1

Технічний проект 15.02.21 – 28.02.21.	Розробка технічного проекту	Нормалізація структури БД, визначення проекту інтерфейсу; розробка докладного алгоритму; розробка структури програми; остаточне визначення конфігурації технічних засобів; розробка
Робочий проект 01.03.21 – 10.05.21	Розробка програми	Реалізація додатку; розробка методики випробувань; проведення попередніх випробувань (тестування); коректування додатку; розробка документації
Розробка програмної документації 11.05.21 – 20.05.21	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 21.05.21 – 30.05.21	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних конфігураційних, модульних та функціональних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми.	Випуск додатку на платформі PlayMarket; внесення коректувань в програмне забезпечення і документацію

3.8 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування системи.

Після закінчення розробки системи повинні бути проведене тестування на захист від некоректного введення.

ДОДАТОК Б

(обов'язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Клас MainActivity

```
package com.example.weekbook;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.viewpager.widget.ViewPager;

import android.app.SearchManager;
import android.content.Intent;
import android.os.Bundle;
import android.widget.SearchView;
import android.view.Menu;
import android.view.MenuItem;


public class MainActivity extends AppCompatActivity {

private  WeekPagerAdapter weekPagerAdapter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         weekPagerAdapter = new WeekPagerAdapter(getSupportFragmentManager(),  1,
this );

        ViewPager viewPager = findViewById(R.id.viewpager);
        viewPager.setAdapter(weekPagerAdapter);
        viewPager.setCurrentItem(4);

    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.menu, menu);
        MenuItem searchItem = menu.findItem(R.id.action_search);
        /*SearchView searchView =
                (SearchView) searchItem.getActionView();
        SearchManager searchManager = (SearchManager)
getSystemService(SEARCH_SERVICE);

searchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentName()));*/
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        switch(id){
```

```
            case R.id.settings:
                return true;
            case R.id.about:
                showAbout();
                return true;
            case R.id.action_search:
                Intent intent = new Intent(getApplicationContext(),
SearchActivity.class);
                startActivity(intent);
                return true;
            case R.id.action_backup:
                Intent intent2 = new Intent(getApplicationContext(),
BackupActivity.class);
                startActivity(intent2);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
    public void showAbout() {
        FragmentManager manager = getSupportFragmentManager();
        AboutDialogFragment aboutDialogFragment = new AboutDialogFragment();
        aboutDialogFragment.show(manager, "myDialog");
    }

    }
```

Клас DayActivity

```
package com.example.weekbook;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;

import com.example.weekbook.data.DBHelper;

import  com.example.weekbook.data.WeekbookContract.RecordsEntry;

public class DayActivity extends AppCompatActivity {

   private TextView weekDayTextView;
   private EditText editText;
   private EditText editTextMainTasks;
   private String weekDay;
   private DBHelper myDBHelper;
   private Long  recId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_day);
```

```
        myDBHelper= new DBHelper(this);
        weekDayTextView = (TextView) findViewById(R.id.weekDayTextView);
        weekDayTextView.setText(getIntent().getStringExtra("weekDay"));
        weekDay = getIntent().getStringExtra("weekDay");
        editText = (EditText)findViewById(R.id.editText);
        editTextMainTasks=(EditText)findViewById(R.id.editTextMainTask);

        displayDatabaseInfo();

    }

    @Override
    protected void onPause(){
        super.onPause();
        update(editTextMainTasks.getText().toString(),editText.getText().toString());

    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        getMenuInflater().inflate(R.menu.day_menu, menu);

        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {

            case R.id.action_add:
                Intent intent = new Intent(getApplicationContext(),
TagActivity.class);

                        intent.putExtra("Day", weekDay);
                        intent.putExtra("recId", recId);
                        // показуємо нове Activity
                        startActivity(intent);
                return true;

            default:
                // If we got here, the user's action was not recognized.
                // Invoke the superclass to handle it.
                return super.onOptionsItemSelected(item);

        }
    }



    private void displayDatabaseInfo() {
        // Создадим и откроем для чтения базу данных
        SQLiteDatabase db = myDBHelper.getReadableDatabase();

        // Зададим условие для выборки - список столбцов
        String[] projection = {
                RecordsEntry._ID,
                RecordsEntry.COLUMN_SHORT_TEXT,
                RecordsEntry.COLUMN_ALL_TEXT};
        String selection = RecordsEntry.COLUMN_DATE + "=?";
        String[] selectionArgs = {getIntent().getStringExtra("weekDay")};
        // Делаем запрос
        Cursor cursor = db.query(
                RecordsEntry.TABLE_NAME,   // таблица
```

```
                projection,            // столбцы
                selection,                  // столбцы для условия WHERE
                selectionArgs,                  // значения для условия WHERE
                null,                  // Don't group the rows
                null,                  // Don't filter by row groups
                null);                   // порядок сортировки

        TextView displayTextView = editText;
        TextView displayTextView2 = editTextMainTasks;
        if (cursor.getCount()==0) {
            insert();


        }
        try {

            // Узнаем индекс каждого столбца
            int idColumnIndex =cursor.getColumnIndex(RecordsEntry._ID);
            int All_Text_ColumnIndex =
cursor.getColumnIndex(RecordsEntry.COLUMN_ALL_TEXT);
            int Short_Text_ColumnIndex =
cursor.getColumnIndex(RecordsEntry.COLUMN_SHORT_TEXT);

            // Проходим через все ряды
            while (cursor.moveToNext()) {
                // Используем индекс для получения строки или числа
                int currentID = cursor.getInt(idColumnIndex);
                recId = (long)currentID;
                String currentAllText = cursor.getString(All_Text_ColumnIndex);
                String currentShortText = cursor.getString(Short_Text_ColumnIndex);

                // Выводим значения каждого столбца
                displayTextView.setText(currentAllText);
                displayTextView2.setText(currentShortText);
            }
        } finally {
            // Всегда закрываем курсор после чтения
            cursor.close();
        }
    }
    private void insert( ) {

        // Gets the database in write mode
        SQLiteDatabase db = myDBHelper.getWritableDatabase();
        // Создаем объект ContentValues, где имена столбцов ключи,
        // а информация о госте является значениями ключей
        ContentValues values = new ContentValues();
        values.put(RecordsEntry.COLUMN_DATE, getIntent().getStringExtra("weekDay") );

        recId = db.insert(RecordsEntry.TABLE_NAME, null, values);
        Log.d("myLog", "inserted rows id = " + recId);
    }
    private void update( String shortText, String allText) {

        // Gets the database in write mode
        SQLiteDatabase db = myDBHelper.getWritableDatabase();
        String selection = RecordsEntry.COLUMN_ID + "=?";
        String[] selectionArgs = new String[] {recId.toString()};
        ContentValues values = new ContentValues();
        values.put(RecordsEntry.COLUMN_SHORT_TEXT, shortText);
        values.put(RecordsEntry.COLUMN_ALL_TEXT,allText);
        long updCount= db.update(RecordsEntry.TABLE_NAME,values,
selection,selectionArgs);
```

```
        Log.d("myLog", "updated rows count = " + updCount);
    }

}
```

Клас TagActivity

```
package com.example.weekbook;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.example.weekbook.data.DBHelper;
import com.example.weekbook.data.WeekbookContract;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class TagActivity extends AppCompatActivity {
    private TextView weekDayTextView;
    private TextView tagNameTextView;
   private RecyclerView tagsRecyclerView;
   private TagAdapter adapter;
    private DBHelper myDBHelper;
    private Long  recId;
    TagAdapter.OnTagClickListener tagClickListener;
    ArrayList<Tag> tags = new ArrayList<Tag>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tag);
        myDBHelper= new DBHelper(this);
        weekDayTextView = (TextView) findViewById(R.id.dayTextView);
        tagNameTextView= (TextView) findViewById(R.id.tagNameTextView);
        weekDayTextView.setText(getIntent().getStringExtra("Day"));
        recId = getIntent().getExtras().getLong("recId");
       /* if (recId==15) {
            tags.add(new Tag("tag1", recId, 1));
            tags.add(new Tag("tag2", recId, 1));
        }
        if (recId==16) {
            tags.add(new Tag("tag3", recId, 1));
            tags.add(new Tag("ardf", recId, 1));
        }*/
        displayDatabaseInfo();
         tagsRecyclerView = (RecyclerView)findViewById(R.id.list);
        // определяем слушателя нажатия элемента в списке
         tagClickListener = new TagAdapter.OnTagClickListener() {
            @Override
            public void onTagClick(Tag tag, int position) {
```

```
                removeTag(tag);
                Toast.makeText(getApplicationContext(), "tag \"" + tag.getName()+"\"
was deleted",
                        Toast.LENGTH_SHORT).show();
            }
        };
       // создаем адаптер
        createTagAdapter();

    }

    public void removeTag(Tag tag){
        tags.remove(tag);
        createTagAdapter();
    }

    public void createTagAdapter(){
        // создаем адаптер
        adapter = new TagAdapter(this, tags,tagClickListener);
        // устанавливаем для списка адаптер
        tagsRecyclerView.setAdapter(adapter);
    }

    public void addTag(View v){
        String tagName=tagNameTextView.getText().toString();
        long tagId=insert(tagName);
        if (tagId!=-100) {
            tags.add(new Tag(tagName, recId, tagId));
            createTagAdapter();
        }
        tagNameTextView.setText("");

    }


    private void displayDatabaseInfo() {
        // Создадим и откроем для чтения базу данных
        SQLiteDatabase db = myDBHelper.getReadableDatabase();

        // Зададим условие для выборки - список столбцов
      /*   String[] projection = {
                WeekbookContract.TagsEntry._ID,
                WeekbookContract.TagsEntry.COLUMN_TAG_NAME};
        String selection = WeekbookContract.TagsOnRecordsEntry.COLUMN_RECORD_ID +
"=?";
        String[] selectionArgs = {recId.toString()};
        // Делаем запрос
       Cursor cursor = db.query(
                WeekbookContract.TagsEntry.TABLE_NAME +" left join "+
WeekbookContract.TagsOnRecordsEntry.TABLE_NAME,   // таблица
                projection,            // столбцы
                selection,                  // столбцы для условия WHERE
                selectionArgs,                  // значения для условия WHERE
                null,                  // Don't group the rows
                null,                  // Don't filter by row groups
                null);                   // порядок сортировки

*/
        final Cursor cursor = db.rawQuery("SELECT _id, tagName FROM tags left join
TagsOnRecords tor on tags._id = tor.TagId where tor.RecordId = ?;", new
String[]{recId.toString()});
```

```
        try {

            // Узнаем индекс каждого столбца
            int idColumnIndex =cursor.getColumnIndex(WeekbookContract.TagsEntry._ID);
            int columnTagName =
cursor.getColumnIndex(WeekbookContract.TagsEntry.COLUMN_TAG_NAME);

            // Проходим через все ряды
            while (cursor.moveToNext()) {
                // Используем индекс для получения строки или числа
                int currentID = cursor.getInt(idColumnIndex);
                String currentTagName = cursor.getString(columnTagName);


                // Выводим значения каждого столбца
                Tag currentTag = new Tag(currentTagName,recId,currentID);
                tags.add(currentTag);
            }
        } finally {
            // Всегда закрываем курсор после чтения
            cursor.close();
        }
    }

    private long insert( String tagName) {

        // Gets the database in write mode
        SQLiteDatabase db = myDBHelper.getWritableDatabase();
        long tagId=-10;
        final Cursor cursor = db.rawQuery("SELECT _id FROM tags  where tags.tagName =
?;", new String[]{tagName});
        try {

            // Узнаем индекс каждого столбца
            int idColumnIndex =cursor.getColumnIndex(WeekbookContract.TagsEntry._ID);

            // Проходим через все ряды
            while (cursor.moveToNext()) {
                // Используем индекс для получения строки или числа
                tagId = cursor.getInt(idColumnIndex);

            }
        } finally {
            // Всегда закрываем курсор после чтения
            cursor.close();
        }
        // Создаем объект ContentValues, где имена столбцов ключи,
        // а информация о госте является значениями ключей
        if (tagId==-10) {
            ContentValues values = new ContentValues();
            values.put(WeekbookContract.TagsEntry.COLUMN_TAG_NAME, tagName);
            tagId = db.insert(WeekbookContract.TagsEntry.TABLE_NAME, null, values);

            Log.d("myLog", "inserted rows id = " + tagId);

           /* List<Tag> result = tags.stream().filter(tag -> {
                tag.getName() == tagName
            }).collect(Collectors.toList());*/
        }else if (tags.contains(new Tag (tagName, recId, tagId))){
            Toast.makeText(getApplicationContext(), "tag \"" + tagName+"\" actually
exist",
                    Toast.LENGTH_SHORT).show();
            return -100;
```

```
        }

        ContentValues values2 = new ContentValues();
        values2.put(WeekbookContract.TagsOnRecordsEntry.COLUMN_RECORD_ID, recId);
        values2.put(WeekbookContract.TagsOnRecordsEntry.COLUMN_TAG_ID, tagId);
        db.insert(WeekbookContract.TagsOnRecordsEntry.TABLE_NAME, null, values2);
        Log.d("myLog", "inserted rows id = " + tagId);
        return tagId;
    }
}
```

Клас Tag

```
package com.example.weekbook;

import androidx.annotation.Nullable;

public class Tag {
    private String name;
    private long idDay;
    private long idTag;

    public Tag(String name, long idDay, long idTag){

        this.name=name;
        this.idDay=idDay;
        this.idTag=idTag;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getIdDay() {
        return this.idDay;
    }

    public void setIdDay(long idDay) {
        this.idDay = idDay;
    }

    public long getIdTag() {
        return this.idTag;
    }

    public void setIdTag(long idTag) {
        this.idTag = idTag;
    }

    @Override
    public boolean equals(@Nullable Object obj) {
        if(name==obj.getName())
        return super.equals(obj);
    }
}
```

Клас TagAdapter

```
package com.example.weekbook;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;
public class TagAdapter extends RecyclerView.Adapter<TagAdapter.ViewHolder>{
    interface OnTagClickListener{
        void onTagClick(Tag tag, int position);
    }

    private final OnTagClickListener onClickListener;

    private final LayoutInflater inflater;
    private final List<Tag> tags;

    TagAdapter(Context context, List<Tag> tags, OnTagClickListener onClickListener) {
        this.onClickListener = onClickListener;
        this.tags = tags;
        this.inflater = LayoutInflater.from(context);
    }
    @Override
    public TagAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

        View view = inflater.inflate(R.layout.list_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(TagAdapter.ViewHolder holder, int position) {
        Tag tag = tags.get(position);
        holder.nameView.setText(tag.getName());
        holder.itemView.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v)
            {
                onClickListener.onTagClick(tag, position);
            }
        });
    }

    @Override
    public int getItemCount() {
        return tags.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        final TextView nameView;
        ViewHolder(View view){
            super(view);
            nameView = (TextView) view.findViewById(R.id.tagTextView);
        }
    }
}
```

Клас Week Fragment

```
package com.example.weekbook;
import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.example.weekbook.data.DBHelper;
import com.example.weekbook.data.WeekbookContract;


public class WeekFragment extends Fragment {
    final static String MONDAY = "text_Monday";
    final static String TUESDAY = "text_Tuesday";
    final static String WEDNESDAY = "text_Wednesday";
    final static String THURSDAY = "text_Thursday";
    final static String FRIDAY = "text_Friday";
    final static String SATURDAY = "text_Saturday";
    final static String SUNDAY = "text_Sunday";

    private TextView mondayEditText;
    private TextView tuesdayEditText;
    private TextView wednesdayEditText;
    private TextView thursdayEditText;
    private TextView fridayEditText;
    private TextView saturdayEditText;
    private TextView sundayEditText;

    private DBHelper myDBHelper;
    private Long  recId;

    private String text_Monday ;
    private String text_Tuesday;
    private String text_Wednesday;
    private String text_Thursday;
    private String text_Friday;
    private String text_Saturday;
    private String text_Sunday;

    public WeekFragment(){}


    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.week_fragment, container, false);

        Bundle arguments = getArguments();
        if (arguments != null) {
```

```
            text_Monday = arguments.getString(MONDAY);
            text_Tuesday = arguments.getString(TUESDAY);
            text_Wednesday = arguments.getString(WEDNESDAY);
            text_Thursday = arguments.getString(THURSDAY);
            text_Friday = arguments.getString(FRIDAY);
            text_Saturday = arguments.getString(SATURDAY);
            text_Sunday = arguments.getString(SUNDAY);

           displayValues(view);

        }

        return view;
    }

    @Override
    public void onResume() {
        super.onResume();
        getTextForAll();
    }


    private void displayValues(View v) {

        mondayEditText=(TextView) v.findViewById(R.id.mondayEditText);
        tuesdayEditText=(TextView) v.findViewById(R.id.tuesdayEditText);
        wednesdayEditText=(TextView) v.findViewById(R.id.wednesdayEditText);
        thursdayEditText=(TextView) v.findViewById(R.id.thursdayEditText);
        fridayEditText=(TextView) v.findViewById(R.id.fridayEditText);
        saturdayEditText=(TextView) v.findViewById(R.id.saturdayEditText);
        sundayEditText=(TextView) v.findViewById(R.id.sundayEditText);


        myDBHelper= new DBHelper(v.getContext());
        getToday(v);
        getTextForAll();
        //створюємо лістенер, який слухає чи натискалось на елемент
        View.OnClickListener listener= new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                goNewView(v);
            }
        };
        //назначаємо цей лістенер елементам
        mondayEditText.setOnClickListener(listener);
        tuesdayEditText.setOnClickListener(listener);
        wednesdayEditText.setOnClickListener(listener);
        thursdayEditText.setOnClickListener(listener);
        fridayEditText.setOnClickListener(listener);
        saturdayEditText.setOnClickListener(listener);
        sundayEditText.setOnClickListener(listener);
    }
    public void getToday(View v)
    {
        TextView textViewMonday = (TextView) v.findViewById(R.id.textViewMonday);
        TextView textViewTuesday = (TextView) v.findViewById(R.id.textViewTuesday);
        TextView textViewWednesday = (TextView)
v.findViewById(R.id.textViewWednesday);
        TextView textViewThursday = (TextView) v.findViewById(R.id.textViewThursday);
        TextView textViewFriday = (TextView) v.findViewById(R.id.textViewFriday);
        TextView textViewSaturday = (TextView) v.findViewById(R.id.textViewSaturday);
```

```
    TextView textViewSunday = (TextView) v.findViewById(R.id.textViewSunday);
    //встановлюємо текст з датою
    textViewMonday.setText(text_Monday);
    textViewTuesday.setText(text_Tuesday);
    textViewWednesday.setText(text_Wednesday);
    textViewThursday.setText(text_Thursday);
    textViewFriday.setText(text_Friday);
    textViewSaturday.setText(text_Saturday);
    textViewSunday.setText(text_Sunday);
}
public void getTextForAll()
{
    //відображаємо дані з БД
    displayDatabaseInfo(text_Monday,mondayEditText);
    displayDatabaseInfo(text_Tuesday,tuesdayEditText);
    displayDatabaseInfo(text_Wednesday,wednesdayEditText);
    displayDatabaseInfo(text_Thursday,thursdayEditText);
    displayDatabaseInfo(text_Friday,fridayEditText);
    displayDatabaseInfo(text_Saturday, saturdayEditText);
    displayDatabaseInfo(text_Sunday,sundayEditText);

}
public void goNewView(View v){
    Intent intent = new Intent(v.getContext(), DayActivity.class);
    switch (v.getId()) {
        case R.id.mondayEditText:
            // Вказуємо між якими Activity буде зв'язок
            intent.putExtra("weekDay", text_Monday);
            // показуємо нове Activity
            startActivity(intent);
            break;
        case R.id.tuesdayEditText:

            intent.putExtra("weekDay", text_Tuesday);
            // показуємо нове Activity
            startActivity(intent);
            break;
        case R.id.wednesdayEditText:

            intent.putExtra("weekDay",text_Wednesday);
            // показуємо нове Activity
            startActivity(intent);
            break;
        case R.id.thursdayEditText:

            intent.putExtra("weekDay", text_Thursday);
            // показуємо нове Activity
            startActivity(intent);
            break;
        case R.id.fridayEditText:

            intent.putExtra("weekDay",text_Friday);
            // показуємо нове Activity
            startActivity(intent);

            break;
        case R.id.saturdayEditText:

            intent.putExtra("weekDay", text_Saturday);
            // показуємо нове Activity
            startActivity(intent);
            break;
        case R.id.sundayEditText:
```

```
                intent.putExtra("weekDay", text_Sunday);
                // показуємо нове Activity
                startActivity(intent);
                break;
            default:
                break;

        }
    }

    private void displayDatabaseInfo(String date, TextView textView) {
        // Створюємо і відкриваємо для читання БД
        SQLiteDatabase db = myDBHelper.getReadableDatabase();

        // Задаємо умову для вибірки - список стовбчиків
        String[] projection = {
                WeekbookContract.RecordsEntry._ID,
                WeekbookContract.RecordsEntry.COLUMN_SHORT_TEXT,
                WeekbookContract.RecordsEntry.COLUMN_ALL_TEXT};
        String selection = WeekbookContract.RecordsEntry.COLUMN_DATE + "=?";
        String[] selectionArgs = {date};
        // Робимо запит
        Cursor cursor = db.query(
                WeekbookContract.RecordsEntry.TABLE_NAME,   // таблиця
                projection,            // стовбчики
                selection,                  // стовбчики для умови WHERE
                selectionArgs,                  // значення для умови WHERE
                null,                  // Don't group the rows
                null,                  // Don't filter by row groups
                null);                   // порядок сортування


        TextView displayTextView = textView;
        if (cursor.getCount()==0) {
            insert( date);


        }
        try {

            // Дізнаємось індекс стовбчиків
            int idColumnIndex
=cursor.getColumnIndex(WeekbookContract.RecordsEntry._ID);
            int Short_Text_ColumnIndex =
cursor.getColumnIndex(WeekbookContract.RecordsEntry.COLUMN_SHORT_TEXT);


            // Проходимо через всі записи
            while (cursor.moveToNext()) {
                //Використовуємо індекс для отримання рядків чи чисел
                int currentID = cursor.getInt(idColumnIndex);
                //зберігамо індекс запиту
                recId = (long)currentID;
                String currentShortText = cursor.getString(Short_Text_ColumnIndex);

                // Виводимо значення кожного стовпчика
                displayTextView.setText(currentShortText);
            }
        } finally {
            // Закриваємо курсор
            cursor.close();
```

```
        }
    }
    private void insert(String date ) {

        SQLiteDatabase db = myDBHelper.getWritableDatabase();
        // створюємо об'єкт ContentValues, де імена стовбчиків ключі,
        // а дата - значення ключів
        ContentValues values = new ContentValues();
        values.put(WeekbookContract.RecordsEntry.COLUMN_DATE, date );

        recId = db.insert(WeekbookContract.RecordsEntry.TABLE_NAME, null, values);
        Log.d("myLog", "inserted rows id = " + recId);
    }

}
```

Клас WeekPagerAdapter

```
package com.example.weekbook;

        import android.content.Context;
        import android.content.res.Resources;
        import android.os.Bundle;

        import androidx.annotation.NonNull;
        import androidx.annotation.Nullable;
        import androidx.fragment.app.Fragment;
        import androidx.fragment.app.FragmentManager;
        import androidx.fragment.app.FragmentPagerAdapter;

        import java.text.SimpleDateFormat;
        import java.util.Calendar;
        import java.util.GregorianCalendar;

public class WeekPagerAdapter extends FragmentPagerAdapter {
    private String[] weeksTitles;

    public WeekPagerAdapter(@NonNull FragmentManager fm, int behavior, Context
context) {
        super(fm, behavior);
        Resources resources = context.getResources();
        weeksTitles = resources.getStringArray(R.array.weeksTitles);
    }

    @NonNull
    @Override
    public Fragment getItem(int position) {
        Bundle arguments = new Bundle();
        WeekFragment weekFragment=new WeekFragment();
        getToday(position,arguments);
        weekFragment.setArguments(arguments);
        return weekFragment;
    }

    @Override
    public int getCount() {
        return 13;
    }
    @Nullable
    @Override
```

```
    public CharSequence getPageTitle(int position) {
        return weeksTitles[position];
    }


    public void getToday(int position,Bundle arguments)
    {
        //находимо сьогоднішню дату
        SimpleDateFormat sdf = new SimpleDateFormat(" dd.MM.yy");
        Calendar calendar =Calendar.getInstance();
        // Calendar calendar = new GregorianCalendar(2021,4,13,13,24,56);
        //знаходимо дату понеділка поточного тижня
        calendar.add(Calendar.DAY_OF_MONTH, -
(calendar.get(Calendar.DAY_OF_WEEK)==Calendar.SUNDAY?
6:calendar.get(Calendar.DAY_OF_WEEK)-2));
        //здвигаємо на кількість пролистаних фрагментів
        calendar.add(Calendar.DAY_OF_MONTH,(position-4)*7);
        //присвоюємо дату кожногоднятижня і далі здвигаємо на 1 день
        arguments.putString(WeekFragment.MONDAY, "Monday "+
sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH,  1);
        arguments.putString(WeekFragment.TUESDAY, "Tuesday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.WEDNESDAY, "Wednesday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.THURSDAY, "Thursday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.FRIDAY, "Friday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.SATURDAY, "Saturday
"+sdf.format(calendar.getTime()));
        calendar.add(Calendar.DAY_OF_MONTH, 1);
        arguments.putString(WeekFragment.SUNDAY, "Sunday
"+sdf.format(calendar.getTime()));
    }
}
```

Клас WeekBookContract

```
package com.example.weekbook.data;

import android.provider.BaseColumns;

public final class WeekbookContract {
    private WeekbookContract(){}
    //таблиця з записами
    public static final class RecordsEntry implements BaseColumns {
        public static final String TABLE_NAME = "records"; // назва таблиці в бд
        // назва стовбців
        public static final String COLUMN_ID = "_id";
        public static final String COLUMN_SHORT_TEXT = "shortText";
        public static final String COLUMN_ALL_TEXT = "allText";
        public static final String COLUMN_DATE = "date";
    }
    //таблиця з тегами
    public static final class TagsEntry implements BaseColumns {
```

```
        public static final String TABLE_NAME = "tags"; // назва таблиці в бд
        // назва стовбців
        public static final String COLUMN_ID = "_id";
        public static final String COLUMN_TAG_NAME = "tagName";
    }
    //зведена таблиця(допоміжна) для створення зв'язку багато до багатьох
    public static final class TagsOnRecordsEntry implements BaseColumns {
        public static final String TABLE_NAME = "TagsOnRecords"; // назва таблиці в
бд
        // назва стовбців
        public static final String COLUMN_TAG_ID = "TagId";
        public static final String COLUMN_RECORD_ID = "RecordId";
    }
}
```

Клас DBHelper

```
package com.example.weekbook.data;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import  com.example.weekbook.data.WeekbookContract.RecordsEntry;
import  com.example.weekbook.data.WeekbookContract.TagsEntry;
import  com.example.weekbook.data.WeekbookContract.TagsOnRecordsEntry;

public class DBHelper extends SQLiteOpenHelper {
    public static final String LOG_TAG = DBHelper.class.getSimpleName();
    private static final String DATABASE_NAME = "weekbook.db";
    private static final int DATABASE_VERSION = 1;

    /**
     * конструктор
     * @param context
     */
    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    /**
     *Викликається при створенні БД
     */
    @Override
    public void onCreate(SQLiteDatabase db) {
        // Запит на створення
        String SQL_CREATE_RECORDS_TABLE = "CREATE TABLE " +RecordsEntry.TABLE_NAME+ "
("
                + RecordsEntry.COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + RecordsEntry.COLUMN_SHORT_TEXT + " TEXT, "
                + RecordsEntry.COLUMN_ALL_TEXT + " TEXT, "
                + RecordsEntry.COLUMN_DATE + " TEXT NOT NULL );";
        //Запускаємо створення таблиці з записами
        db.execSQL(SQL_CREATE_RECORDS_TABLE);


        // Запит на створення
        String SQL_CREATE_TAGS_TABLE = "CREATE TABLE " +TagsEntry.TABLE_NAME+ " ("
                + TagsEntry.COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
```

```
                + TagsEntry.COLUMN_TAG_NAME + " TEXT NOT NULL );";
        //Запускаємо створення таблиці з тегами
        db.execSQL(SQL_CREATE_TAGS_TABLE);


        // Запит на створення
        String SQL_CREATE_TAGS_ON_RECORDS_TABLE = "CREATE TABLE "
+TagsOnRecordsEntry.TABLE_NAME+ " ("
                + TagsOnRecordsEntry.COLUMN_RECORD_ID + " INTEGER NOT NULL, "
                + TagsOnRecordsEntry.COLUMN_TAG_ID + " INTEGER NOT NULL, "
                +"FOREIGN KEY ("+TagsOnRecordsEntry.COLUMN_TAG_ID+") REFERENCES
"+TagsEntry.TABLE_NAME
                +"("+TagsEntry.COLUMN_ID+"), "
                +"FOREIGN KEY ("+TagsOnRecordsEntry.COLUMN_RECORD_ID+") REFERENCES
"+RecordsEntry.TABLE_NAME
                +"("+RecordsEntry.COLUMN_ID+" ));";
        //Запускаємо створення зведеної таблиці
        db.execSQL(SQL_CREATE_TAGS_ON_RECORDS_TABLE);
    }

    /**
     *Викликається при оновленні БД
     */
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Запис в журнал
        Log.w("SQLite", "Upgrading from version " + oldVersion + " to version " +
newVersion);

        // Видаляємо старі таблиці
        db.execSQL("DROP TABLE IF  EXISTS " + RecordsEntry.TABLE_NAME);
        db.execSQL("DROP TABLE IF  EXISTS " + TagsEntry.TABLE_NAME);
        db.execSQL("DROP TABLE IF  EXISTS " + TagsOnRecordsEntry.TABLE_NAME);
        // Створюємо нові
        onCreate(db);
    }
}
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький Національний Університет
Факультет програмування та комп’ютерних
і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

Дипломний проект, на тему:

«Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії»

Студент: Дьоміна Анастасія Іванівна

Керівник: Гурман І.В. кандидат технічних наук, доцент

Вступ

Проблеми тайм-менеджменту, планування своїх справ та впорядкування своїх думок є одними з найактуальніших та головних проблем сьогодення. , що потрібно.

Здавалося б, ми живемо в вік технологій, коли вся інформація світу завжди під рукою - у нашому смартфоні, і для вирішення проблем тайм-менеджменту створено понад сотню додатків в тому числі і встановлених за замовчуванням на смартфон, але всі вони однотипні, і вирішують проблему тільки частково, адже головне вікно програми відображає або просто списки задач які існують, або задачі на один конкретний день.

Саме через це багато людей вимушені використовувати паперові щоденники та записники з надрукованою або саморобною розміткою, для того, щоб мати цілісну картину своїх поточних планів. Проте паперові носії мають здатність псуватись та закінчуватись в самий неочікуваний момент. Також їх недовговічність та скінченність створює ще одну проблему, а саме - неможливість завжди мати при собі повну картину своїх планів чи думок. Тому я вирішила створити додаток, який буде поєднувати переваги паперового щоденника та можливості смартфону.

Актуальність та мета дипломного проекту

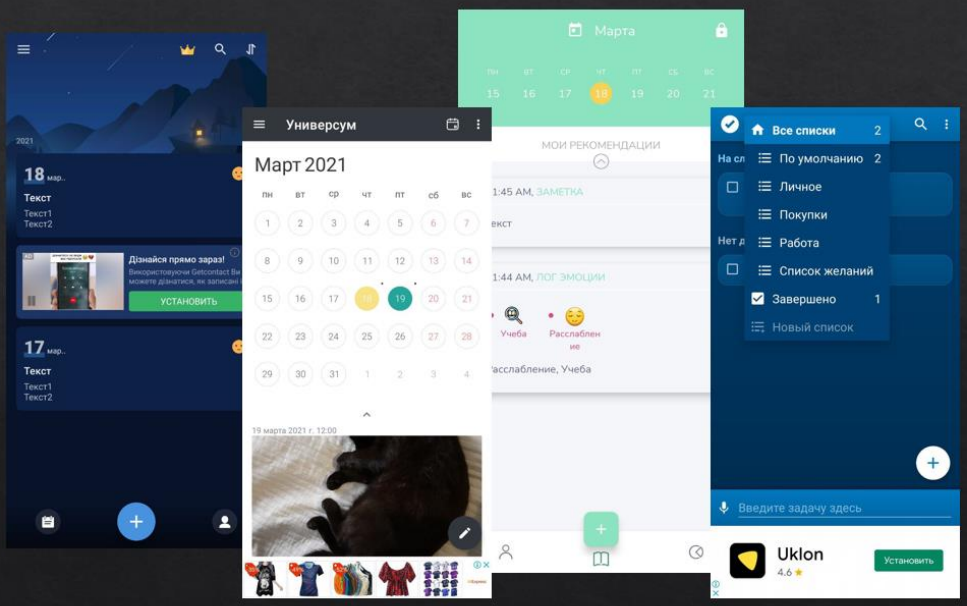
- Актуальність даної теми полягає у тому, що існує безліч додатків з однотипним функціоналом, які не мають зручного та ефективного вирішення вищезазначених проблем, а отже існує необхідність у створенні мобільного застосунку, який зможе запропонувати новий функціонал, що задовольнить потреби користувачів як в організації свого часу так і своїх думок.
- Мета проекту – розробити додаток для мобільних пристроїв з операційною системою Android, який дозволить вирішити проблему тайм-менеджменту, правильної організації задач та записів, а також зручного пошуку та зберігання попередніх записів.

Завдання дипломного проекту

- провести дослідження предметної області, визначити особливості та специфіку процесу організації часу та створення списків задач;
- виконати детальний аналіз існуючих рішень;
- визначити функціональні задачі;
- розробити технічне завдання;
- провести аналіз інструментів та технологій, що використовуються для створення Android додатків;
- проаналізувати та порівняти різні методи для запису та збереження інформації;
- розробити архітектуру додатку;
- розробити сучасний та функціональний інтерфейс;
- виконати програмну реалізацію мобільного застосунку;
- провести тестування готового програмного продукту.

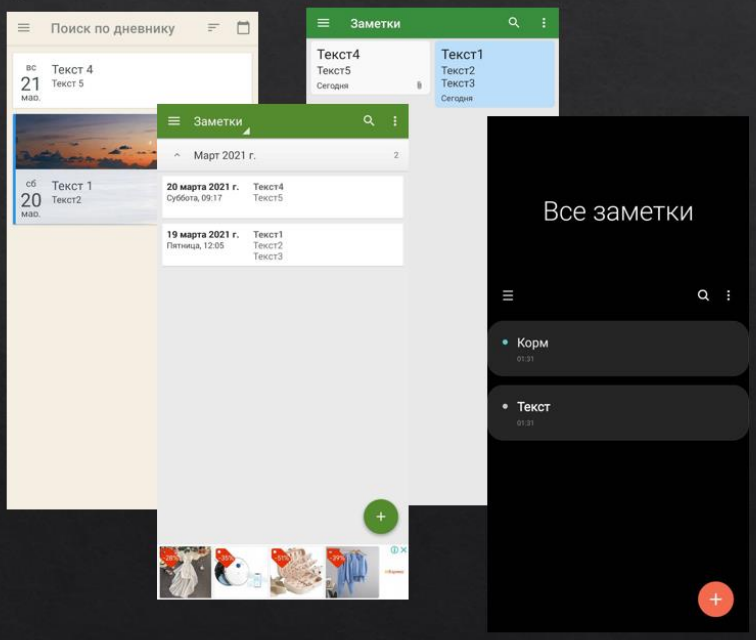
Наявне програмне забезпечення

- My Diary;
- Universum;
- Pi Journal;
- SplenDO;
- MyNotes;
- Diary;
- Notes.



Наявне програмне забезпечення

Переважна більшість продуктів, які існують на ринку мають функціонал направлений на створення звичайних заміток, або створення заміток з простою прив'язкою до дати. Також було виявлено, що майже всі додатки дозволяють створювати бекапи тільки після придбання платної версії, і при тому абсолютна більшість з них робить бекап в хмару, що є не самим безпечним методом для збереження таких інтимних речей, як особистий щоденник.



Проектування мобільного додатку

Виконавши аналіз існуючих архітектурних шаблонів, було обрано найоптимальнішу архітектуру для додатка-щоденника, а саме – MVP, адже вона вирішує проблеми зв'язності, які є в MVC та не ускладнює код програми, а також підходить для невеликих проектів і не впливає на їх швидкодію негативним чином.

MVP – це архітектурний шаблон, похідний від MVC, що відділяє поведінку роботи інтерфейсу, тобто події та його візуальне відображення у різні класи: Представник (Presenter) та Представлення (View) відповідно

Проектування структури бази даних

В таблиці БД Записи зберігається окремо текст головних завдань дня, саме цей текст виводиться на головну активність, окремо додатковий текст, що виводиться тільки на активності дня, а також за яку відповідає запис і його унікальний ідентифікатор.

В таблиці БД Теги зберігається назва тегу та його ідентифікатор.

Кожен запис може вміщувати багато тегів, в той же час кожен тег може використовуватись в багатьох записах. Тобто тут використовується відношення багато до багатьох. Для реалізації такого відношення ми використовуємо таблицю БД Теги в Записах, в якій зберігається інформація про те який тег використовується в якому записі.

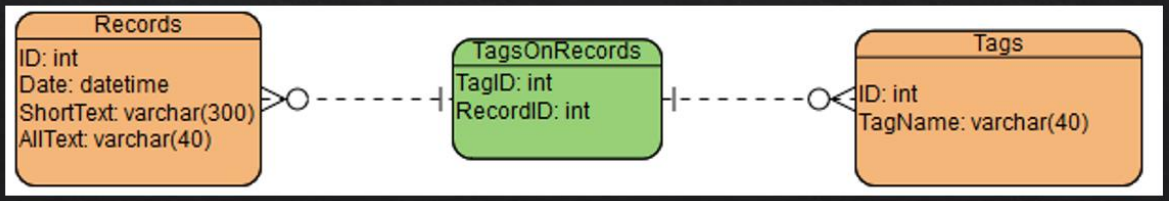
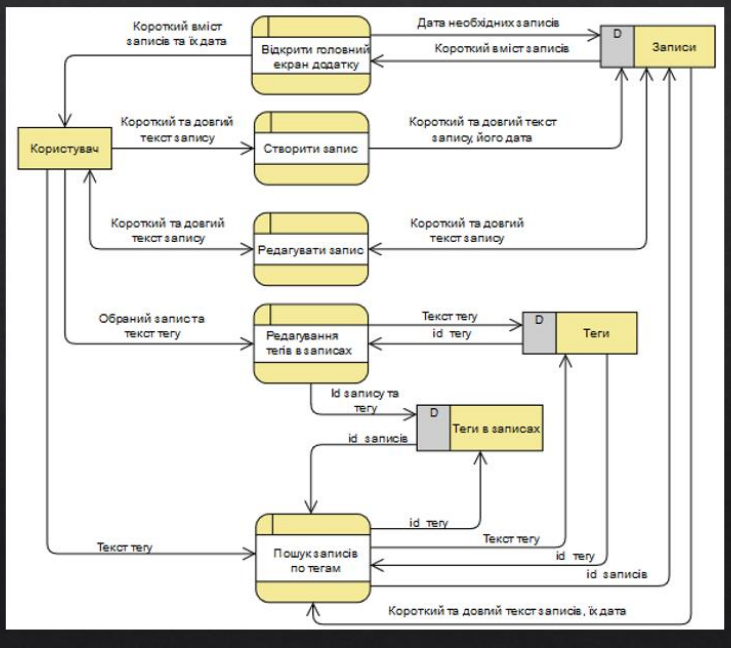


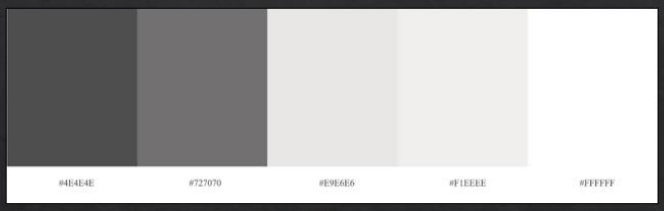

Схема переходів між екранами
Екран «Результати пошуку»
Список результатів
Поле введення запиту
Пошук
Про додаток
Меню
Екран «Задачі тижня»
Довідка
Створити бекап
Екран «Створення бекапу»
Експорт в .TXT
Створити архів
Головні задачі за понеділок
Головні задачі за вівторок
Головні задачі за середу
Головні задачі за четвер
Головні задачі за п'ятницю
Головні задачі за суботу
Головні задачі за неділю
Екран «Задачі дня»
Дата
Змінити дату
Головні задачі дня
Додатковий текст дня
Додати тег
Екран «Додавання тегу»
Текст тегу
Додати
Список доданих тегів

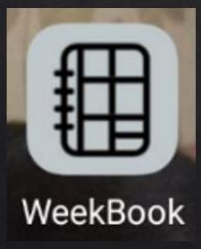

Діаграми станів
Відображення головної сторінки
Перегляд замітки
Замітка існує?
Так
Ні
Створення замітки
Пошук тексту обраної замітки
Відображення знайденого тексту
Редагування тексту
Додавання тексту до замітки
Збереження даних в БД
Додавання тегів
Збереження в БД інформації про теги
Повернення на головну сторінку
Відображення сторінки замітки
Вибір пункту меню "дотати тег"
Введення назви тегу
Додавання тегу
Збереження в БД
Відображення всіх доданих до запису тегів
Введення тексту запиту
Пошук в БД
Замітка існує?
Так
Ні
Відображення замітки, що відповідають запиту
Перегляд замітки
Редагування замітки
відображення інформації про те, що замітки не знайдено
Повернення на головну сторінку
Відображення головної сторінки
Вибір пункту меню "створення бекапу"
Вибір формату
TXT
ZIP
Створення документу
Запис всіх даних з БД в файл
Архівація
Додавання паролю на архів
Збереження
Повернення на головну сторінку

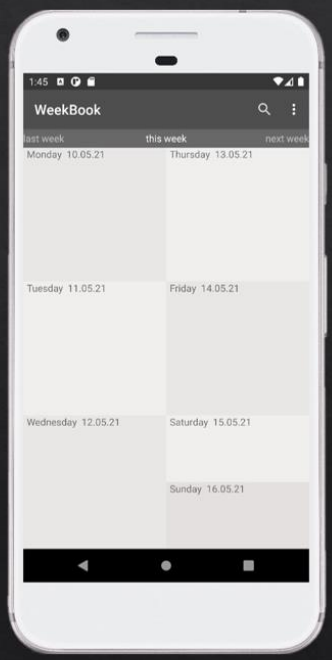
Діаграма потоків даних



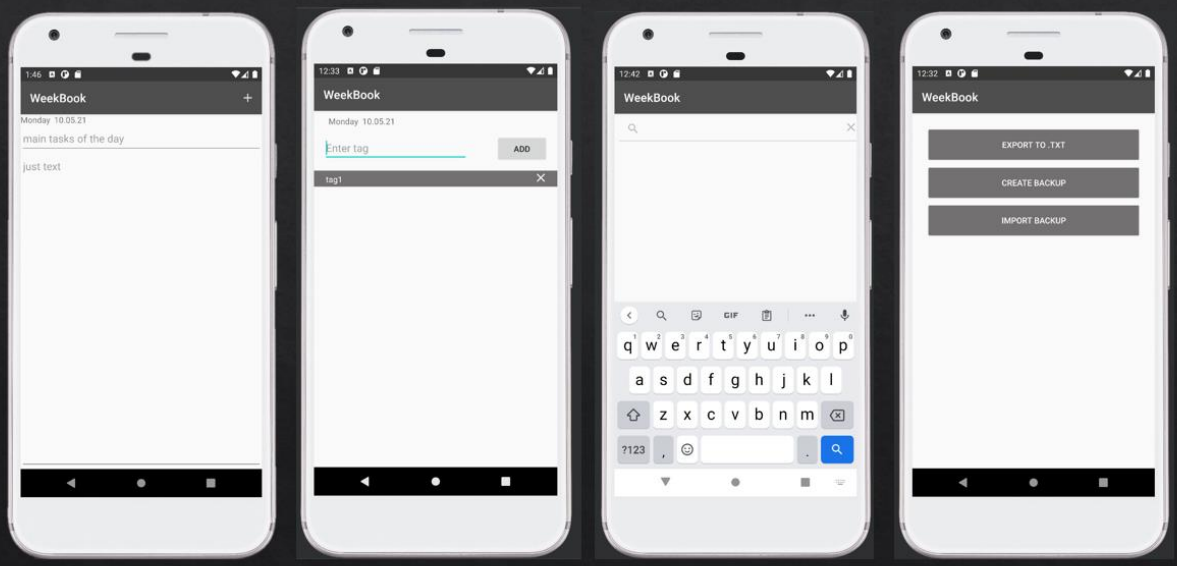
Інтерфейс додатку







Сторінки інтерфейсу



Висновки

При виконанні дипломного проекту було проведено аналіз предметної області, її функціональних та структурних особливостей, визначено особливості та специфіку процесу організації часу та створення списків задач. Проведено детальний аналіз існуючих на ринку рішень, виконано їх порівняння між собою та зроблено висновок про те, що всі вони однотипні, переповнені рекламою, мають недостатній функціонал, та незручний спосіб відображення заміток.

Далі було визначено функціональні задачі додатку, на основі яких було створено функціональні та нефункціональні вимоги. Для якісного виконання поставлених задач розроблено технічне завдання.

Проведено аналіз інструментів та технологій, для створення Android додатків та порівняння різних методів для запису та збереження інформації. Опираючись на результати аналізу розроблено архітектуру додатка та спроектовано сучасний та функціональний інтерфейс.

На основі спроектованої архітектури та інтерфейсу, реалізовано мобільний застосунок. Після розробки проведено модульне, функціональне та конфігураційне тестування, в результаті якого в розробленому додатку присутні всі функції, які були визначені в технічному завдання, і він готовий для використання.

При виконанні дипломного проекту було розроблено додаток для мобільних пристроїв з операційною системою Android, який допомагає вирішити проблему тайм-менеджменту, правильної організації задач та записів, а також зручного пошуку та зберігання попередніх записів, завдяки чому було закріплено теоретичні та практичні знання і навички розробки життєвого циклу програмного забезпечення.

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Дьоміної А. І.
Прізвище, ініціали

факультет ПКТС, 4 курс, група ІПЗ-17-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

01.06.2021р.
дата

підпис

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1008188300

Дата перевірки:
05.06.2021 10:53:49 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
05.06.2021 11:08:13 EEST

ID користувача:
100005589

Назва документа: **Записка_з_рамками_і_додатками_Дьоміна**

Кількість сторінок: **127** Кількість слів: **21542** Кількість символів: **170562** Розмір файлу: **4.73 MB** ID файлу: **1008265198**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

15.2%
Схожість

Найбільша схожість: **3.09%** з Інтернет-джерелом (https://www.CyberForum.ru/android-dev/thread2629313.html)

11.3% Джерела з Інтернету	927	Сторінка 129
4.84% Джерела з Бібліотеки	73	Сторінка 142

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	30 сторінок

Fri Jun 04 21:12:09 EEST 2021, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 14%**

ID: 92363 Назва: Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії Додано в БД: 2021-06-04 Автора: А. І. Дьоміна Керівники: І. В. Гурман Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	106493	1451	12401 (12%)	136 (9%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Дьоміна Анастасія Іванівна

Тема Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки 125

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті виконано аналіз різних методів ведення обліку щоденних та щотижневих задач та особистих записів, а також існуючих рішень на ринку мобільних додатків. Визначено функціональні задачі та розроблено ТЗ. Проведено аналіз існуючих інструментів та технологій розробки Android-додатків та спроектовано архітектуру застосунку та унікальний функціональний інтерфейс. На основі яких розроблено додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії. Після розробки було проведено модульне, функціональне та конфігураційне тестування готового програмного продукту.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект освітнього ступеня "бакалавр" у повній мірі відповідає поставленому завданню як у теоретичній, так і в практичній її частині.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі описано проблему предметної області, обґрунтовано актуальність теми, сформульовано мету і завдання дипломного проекту. У першому розділі: проведено аналіз функціональних і структурних особливостей та специфіки предметної області; виконано порівняння існуючих рішень; визначено функціональні задачі додатку, на основі яких було створено функціональні та нефункціональні вимоги. У другому розділі спроектована архітектура основана на актуальних архітектурних патернах та розроблено інтерфейс опираючись на передові тенденції UX/UI-розробки. У третьому розділі виконана програмна реалізація додатку за допомогою сучасних та надійних інструментів. В останньому розділі проведено модульне, функціональне та конфігураційне тестування готового програмного продукту.

4. Позитивні сторони проекту Дипломний проект містить інноваційні рішення, зокрема зі сторони підходу до розробки UX/UI рішень. Розроблений додаток має унікальний дизайн та підхід до методу збереження та відображення щоденних задач. Також перевагою застосунку є метод резервного копіювання даних, а саме збереження даних на локальному пристрої з використанням паролю, а не в хмарі, чи сервері, де зловмисники можуть отримати доступ до особистих записів користувача.

5. Негативні сторони проекту При виконанні аналізу предметної області було розглянуто існуючі типи мобільних додатків та різні методи ведення записів – краще було детальніше дослідити методи ведення та обліку задач та порівняти їх, а розгляд типів мобільних додатків описати в іншому розділі. Недолік не є суттєвим, та не зменшує позитивне враження від роботи

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконано відповідно до теми дипломного проекту з дотриманням всіх вимог та стандартів. У загальному графічне оформлення виконане на хорошому рівні. Пояснювальна записка відповідає усім вимогам стандартів до її оформлення.

7. Відгук про дипломний проект в цілому В цілому дипломний проект заслуговує позитивної оцінки. Матеріал подано структуровано, чітко та послідовно. Описи процесів аналізу, проектування, реалізації і тестування подано відповідно до структури записки. Графічний матеріал ілюструє актуальність інноваційність та практичну цінність рішень, які були прийняті для вирішення поставленої задачі.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект виконаний в повному обсязі. Беручи до уваги всі зазначені переваги і недоліки, можна зробити висновок, що робота заслуговує оцінки «відмінно»(4.75/A).

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Нічепорук Андрій Олександрович, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та системного програмування (КІСП) ХНУ

"____" _____ 2021 р. _____

(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії»

Автор: Дьоміна Анастасія Іванівна

Спеціальність: 121 – Інженерія програмного забезпечення

Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»

Науковий керівник: Гурман Іван Васильович, канд. тех. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломного проекту системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень в бланках (титульного аркушу, бланку завдання, в структурі підрозділів, ВСТУПУ);
2) В якості запозичень системою було зафіксовано стандартні конструкції та послідовності коду, а також посилання на використання бібліотек які є спільними для великої кількості мобільних додатків та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
3) усі запозичення фрагментарні, або мають належним чином оформленні посилання.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності / схожості, складає 15,2% і адресується до 1000 першоджерел, що з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ І.В. Гурман

Гарант ОП _____ Л.П. Бедратюк

Завідувач кафедри _____ Л.П. Бедратюк