МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система Decider для запобігання прийняття імпульсивних рішень»

Завідувач

випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Берест. О. Б.**

Студента групи ІН – 73 **Лічний Я.Я.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-73 спеціальності "Комп'ютерні науки" денної форми навчання Лічного Ярослава Ярославовича.

Тема: "Інформаційна система Decider для запобігання прийняття імпульсивних рішень"

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів розпізнавання образів; 2) постановка завдання й формування завдань дослідження; 3) опис основних положень, математичних моделей і критеріїв, що використовуються інформаційно-екстремальною інтелектуальною технологією; 5) розробка інформаційного й програмного забезпечення інтелектуальної системи; 6) аналіз результатів моделювання.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Берест О.Б.

Завдання приняв до виконання _____ Лічний Я.Я.

РЕФЕРАТ

Записка: 53 стор., 28 рис., 1 табл., 1 додаток, 13 джерел.

Об’єкт дослідження — інформаційні процеси аналізу прийняття імпульсивних дій.

Мета роботи — розробка мобільного додатку для запобігання скоєння імпульсивних дій. Додаток має дати людині певний час, за який людина вносить позитивні та негативні сторони певної ді

Методи дослідження — метод статистичних випробувань.

Результати — розроблено додаток для боротьби зі скоєнням імпульсивних дій. Додаток створений для використання на платформі Андроїд. Для його створення використовувались актуальні технології, архітектура, комплекси рішень. Зокрема, додаток написаний на мові програмування джава та зберігає свої данні у базі даних Рум.

МОБІЛЬНИЙ ДОДАТОК, ОПЕРАЦІЙНА СИСТЕМА АНДРОЇД, БАЗА ДАНИХ, ІМПУЛЬСИВНІ ДІЇ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, АРХІТЕКТУРА КОМПОНЕНТІВ

Зміст

1.Актуальність 5
1.1.Проблема 5
1.2.Вибір платформи для використання 6
2.Причини та пошук рішення імпульсивних придбань 7
2.1.Причини 7
2.2.Пошук рішення 8
2.3.Необхідний функціонал 9
3.Аналіз літератури 10
3.1.Огляд існуючих рішень 10
3.2.Огляд технологій 13
3.2.1.Актівіті 13
3.2.2.Фрагменти 15
4.Розробка дизайну 18
4.1.Створення UI дизайну 18
4.2.Створення UX дизайну 20
4.3.Створення візуальної частини додатку 29
5.Розробка функціоналу 31
5.1.Вибір архітектури додатку 31
5.2.Розробка бізнес логіки 34
6.Тестування 48
Висновки 51

Вступ

Робота присвячена темі розробки мобільного додатку. Під час виконання цієї роботи буде знайдено пошук рішення існуючої проблеми та розроблено дизайн для мобільного додатку під платформу Андроїд. Буде чітко сформована проблема, яку має вирішувати додаток и визначити яким функціоналом воно має володіти, щоб вирішувати існуючі проблеми. Етап створення прототипу надзвичайно важливий, бо інтерфейс додатку – це саме те, з чим має справу користувач. Якщо додаток має незручний інтерфейс, який є важким для користування, то це дуже негативно відобразиться на бажанні користуватися ним. Буде обрано архітектуру, яка підходить для додатку та враховує подальше масштабування. Буде розроблено візуальну частину додатку та бізнес логіку. Для зберігання даних у додатку буде налаштована його взаємодія з базою даних.

1. Актуальність

1.1. Проблема

Імпульсивність - риса характеру, що виражається в схильності діяти без достатнього свідомого контролю, під впливом зовнішніх обставин або в силу емоційних переживань.

Імпульсивні покупці - люди, що приносять інтернет-магазину відмінну прибуток.[4]

Причин поведінки декілька:

• низька ціна, наприклад, під час акції;

• привабливий дизайн сайту;

• надмірна емоційність;

• невміння контролювати свої бажання;

• реакція на рекламу;

• дуже вигідні пропозиції, від яких неможливо відмовитися;

• правильна робота консультанта, стимулюючого покласти в кошик максимальну кількість позицій.

Умовно всі спонтанні замовлення потрібно розділити на кілька груп:

• повністю незаплановані. Людина приходить в онлайн-магазин за чаєм, а замовляє пачку кави, набір чашок, фруктовий смузі із сухофруктів, піддавшись рекламі або азартній шопінгу;

• незаплановані частково. Замовник планував щось придбати, але зробив остаточний вибір безпосередньо під час перегляду каталогу;

• повна заміна. Клієнт відвідав онлайн-магазин заради улюбленого сорту чаю, але в підсумку віддав перевагу іншому сорту.

1.2. Вибір платформи для використання

Оскільки імпульсивні дії є не спланованими, тому боротьби з ними потрібен інструмент, котрий завжди знаходиться поряд з людиною. Таким інструментом є телефон. Згідно з дослідженнями, 55 відсотків українців мають смартфони. Для молоді цей показник дорівнює 92 відсотків. [5]

На першу половину 2021 року, дві мобільні операційні системи займають абсолютну більшість по кількості девайсів, що їх використовують. Це операційні системи Android та IOS.

Операційна система андроїд є більш розповсюдженою в Україні, тому саме для неї буде створено додаток.

2. Причини та пошук рішення імпульсивних придбань

2.1. Причини

Імпульсивний покупець – це людина, яка відрізняється своєю щедрістю. Вона не звикла планувати. Він може робити замовлення навіть в тому випадку, якщо знаходиться в складній фінансовій ситуації. Нерідко вирішальним фактором стають емоції:

• смуток;

• нудьга;

• піднесений настрій;

• депресія;

• бажання себе побалувати;

• стан, викликаний вживанням тонізуючих напоїв.[4]

2.2. Пошук рішення

Дослідивши варіанти боротьби з імпульсивними рішеннями з різних ресурсів, я зробив наступні висновки. Тож, наступні допоможуть протидіяти імпульсивним рішенням:

Проаналізуйте дійсну ціну речі та зрозумійте, скільки річ коштує насправді. Просто зрозумійте, скільки годин ви працювали, щоб купити його. Для цього розділіть розмір вашої зарплати на кількість робочих годин у місяці. Так ви отримаєте вартість години. А тепер порахуйте, скільки робочих годин «закладено» в вартість речі, яку вам захотілося купити. Якщо на рюкзак ви працювали пару днів, можна брати. А якщо пару тижнів, то він того не варто. Втім, можливі нюанси. Річ дійсно хорошої якості і прослужить вам роки?

Відволіктися та дати собі час на роздум. Іноді під час шопінгу нас дійсно дуже сильні емоції і непереборне бажання купувати. Саме на це націлені маркетингові виверти: спеціальна викладка товарів, розслаблююча музика, приємне світло і навіть шрифт на цінниках. В інтернеті теж не уникнути небезпек: красиві фотографії, спеціальні акції, вигідні пропозиції і підказки на кшталт «Також зверніть увагу на ці товари». В умовах, коли буквально все налаштовує на те, щоб ви що-небудь купили, тримати себе в руках дуже складно. Але можна - досить дати собі трохи часу. Фахівці з особистих фінансів рекомендують брати на роздуми 24 години. Ноги вже самі несуть вас до каси, а рука тягнеться до гаманця? Сфотографуйте на смартфон то, за що ви готові віддати останні гроші, і їдьте додому. А через добу відкрийте фотографію і уважно на неї подивіться: ви точно хочете це купити? Якщо рішучості поменшало, значить ви піддалися емоціям. А якщо ні, значить, не така вже ця покупка і імпульсивна.

Дізнатися думку близьких, друзів і знайомих. Це допоможе вам отримати погляд на те, що ви хочете зробити під іншим кутом. Можливо, ви не помічали, що є інші сторони вашої ідеї.

Збивати імпульс диханням. Ця методика полягає у тому, щоб відійти від емоцій та щоб вони менш впливали на ваш вибір.

Не купуйте в поганому настрої. Так, шопінг здатний знімати стрес. Однак якщо ви дійсно не в кращому стані, краще сходити в спортзал або на прогулянку. Витрачати гроші варто тоді, коли у вас нормальний настрій і не дуже великий рівень стресу - так ймовірність того, що ви нахапався чогось абсолютно непотрібного, нижче. Якщо ж намагатися розвеселити себе саме покупками, розплата наздожене дуже скоро - стрес від бездарно витрачених грошей стане ще сильніше.

Планувати заздалегідь. Планування заздалегідь допоможе вам не будучи під емоціями дійсно зрозуміти, що вам потрібно, а що ні.

Тож, виходячи з пунктів, що були перераховані вище, можна зробити висновок, що для того, щоб зробити якусь дію обдумано потрібен час та побачити як плюси, так і мінуси того, що Ви хочете зробити.

2.3. Необхідний функціонал

Функціонал додатку повинен бути таким, щоб реалізувати шляхи вирішення проблем, які були описані у пункті “2.2. Пошук рішення”. Виходячи з цього зробимо опис загальної ідеї додатку. Нам потрібно мати сутність дії, яку користувач хоче зробити. Надалі будемо це називати дією.

Серед необхідного фунціоналу я би виділив наступні пункти:

1. Додаток має мати список з дій.
2. Користувач може додавати дії.
3. Користувач має можливість вказувати:
    - Назву дії
    - Плюси дії
    - Мінуси дії
    - Час, за який потрібно прийняти рішення, чи здійснювати дію, чи ні.
4. Користувач може редагувати дії.
5. Користувач може видаляти дії.

3. Аналіз літератури

3.1. Огляд існуючих рішень

Для пошуку існуючих додатків для усунення поставленої проблеми будем використовувати Google Play. Google Play - це послуга цифрового розповсюдження, якою керує та розробляє Google. Він служить офіційним магазином додатків для пристроїв, що працюють на операційній системі Android, сертифікованій Google, що дозволяє користувачам переглядати та завантажувати програми, розроблені за допомогою набору для розробки програмного забезпечення Android (SDK) та опубліковані через Google.[7]

Тож, мені вдалося знайти наступні додатки, які мають систематизувати дії користувача.

Універсум. Цей додаток був створений для систематизації планів. У нього можна заносити дані о діях, які користувач хоче здійснити. Чи підходить цей додаток для рішення описаною раніше проблеми?

Як бачимо, у цьому додатку немає можливості додавати опис дії, тому він не дуже підходить. Ми маємо лише змогу додати назву дії та час, коли її треба виконати.

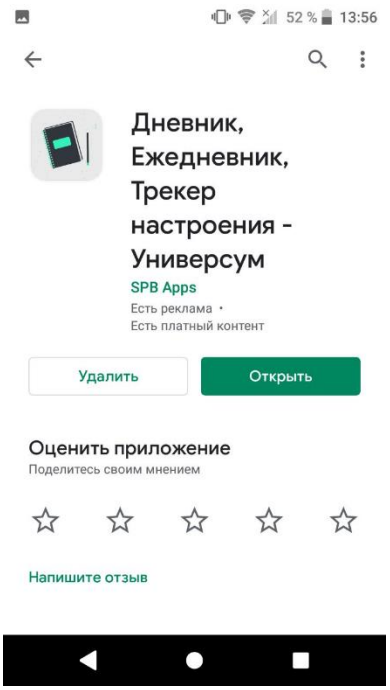

Рисунок 1 - Сторінка додатку на Play Market

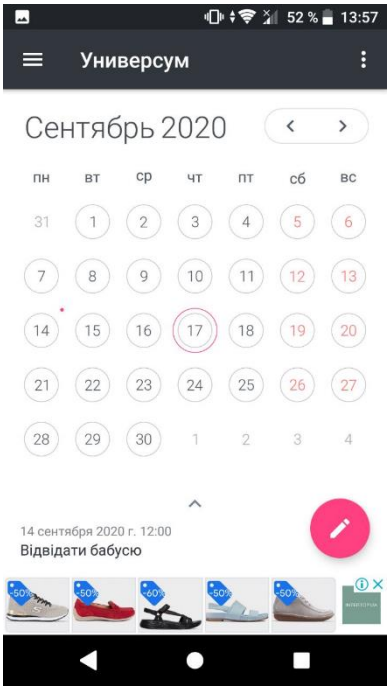

Рисунок 2 - Головне меню додатку

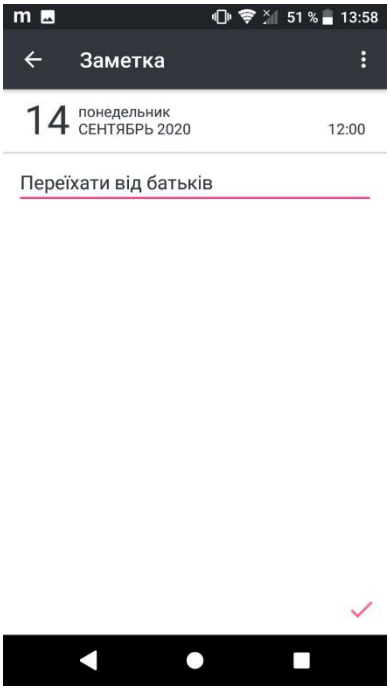

Рисунок 3 - Створення запису

Наступним додатком став додаток «Мої справи». У ньому можна створити и зберігати дії, які хоче створити користувач. Як ми бачимо, цей додаток має такі ж проблеми, як і минулий. Ми не можемо додавати опис до дії, де могли би зберігати детальну інформацію о діях

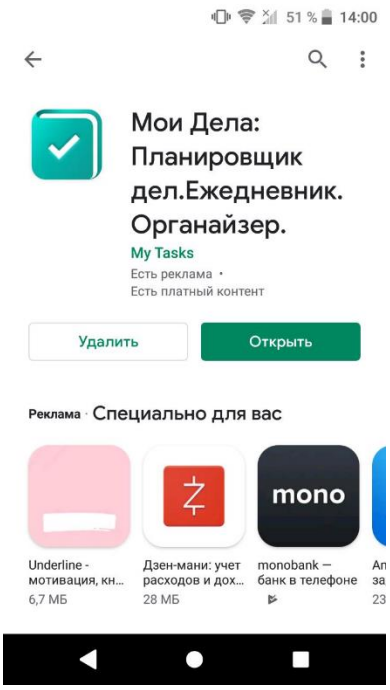

Рисунок 4 – Сторінка додатку на Play Market

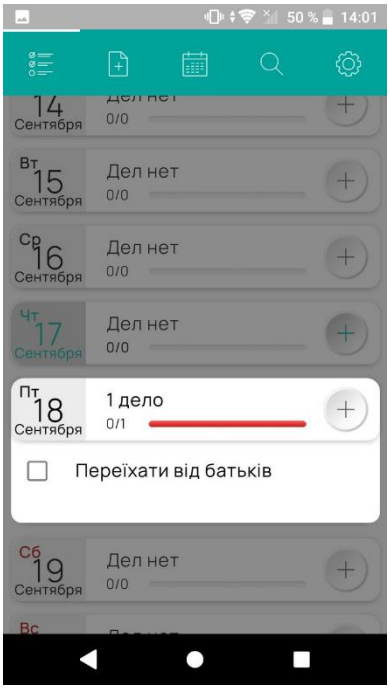

Рисунок 5 – Головне меню

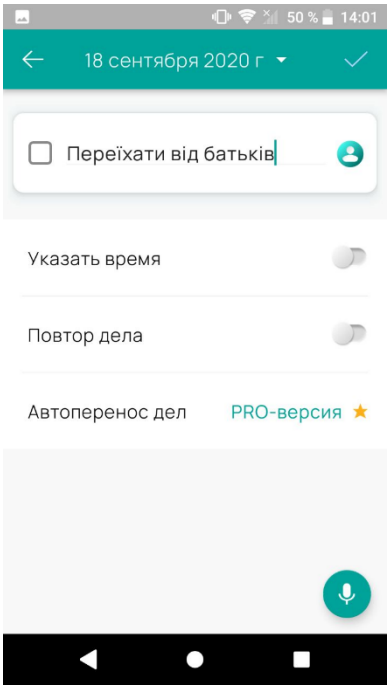

Рисунок 6 – Створення запису

Також, можна використовувати один з блокнотів, яких безліч у Google Play. У моєму випадку це був додаток «Швидкий блокнот». Це дуже простий додаток, але він не є зручним для того, щоб вирішити поставлену проблему.

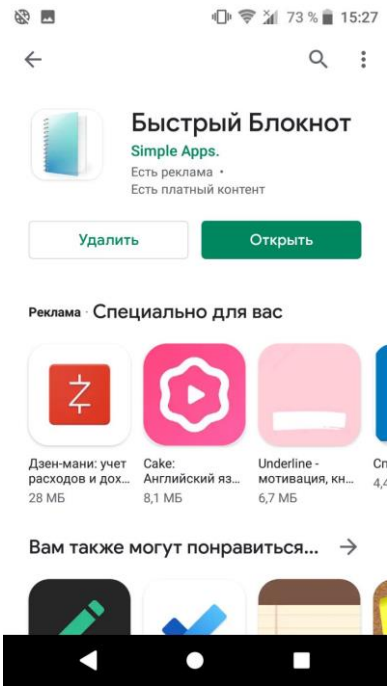

Рисунок 7 - Сторінка додатку на Play Market

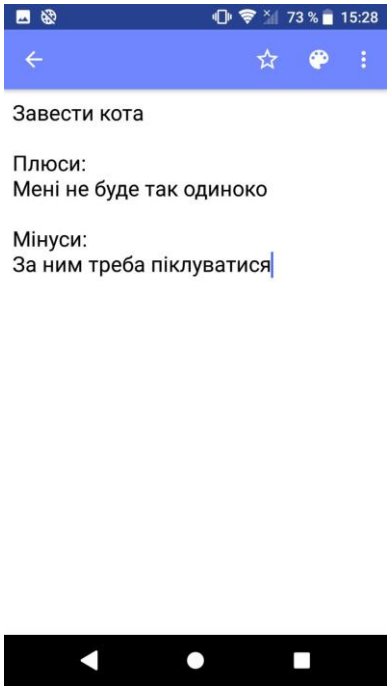

Рисунок 8 - Створення запису

Як ми можемо бачити, цей додаток дуже простий, але і має не обширний функціонал.

3.2. Огляд технологій

3.2.1. Актівіті

Activity представляє окремий екран в Android. Його можна порівняти з вікном в додатку для робочого столу чи фрейм у Java програмі. Activity дозволяє розташувати усі компоненти користувальницького інтерфейсу або віджети на екрані.

Важливо розуміти, що у Activity є життєвий цикл, простіше кажучи, це означає, що вона може бути в одному з різних стадій, в залежності від того, що відбувається з додатком при діях користувача. Актівіті має свій життєвий цикл

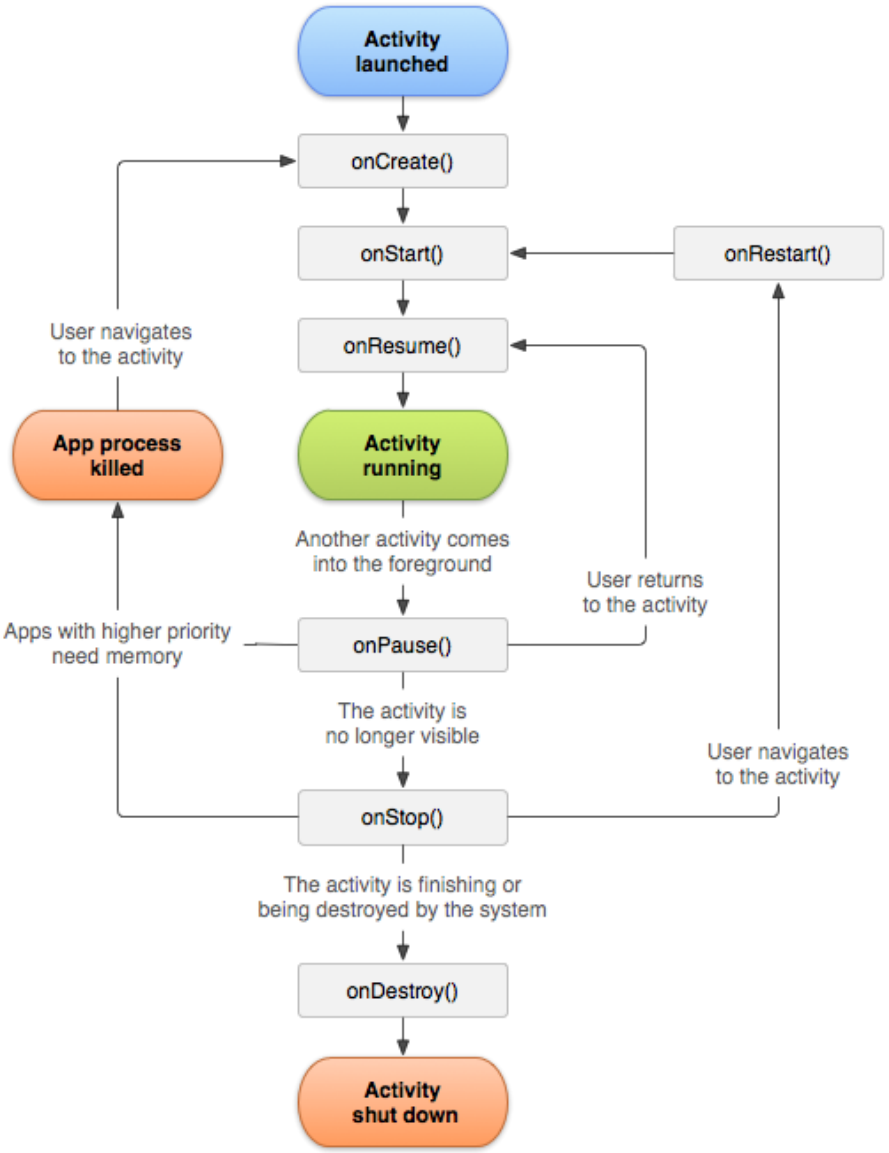

Рисунок 9 - Життєвий цикл актівіті

Для навігації переходами між етапами життєвого циклу діяльності клас Activity надає базовий набір із шести зворотних викликів: onCreate(), onStart(), onResume(), onPause(), onStop() та onDestroy(). Система викликає кожен із цих зворотних викликів, коли діяльність переходить у новий стан.

onCreate (): викликається при першій ініціалізації Activity. Вам потрібно виконати цей метод для виконання будь-якої ініціалізації у вашій Activity.

onStart (): икликається, коли Activity готова і відображається користувачеві в перший раз, так як Activity готується стати інтерактивної і вийти на передній план.

onResume (): коли Activity переходить в цей стан, воно починає взаємодіяти з користувачем. Activity продовжує працювати в цьому стані, поки що-небудь не відведе фокус від додатка або Activity (наприклад, вхідний дзвінок). Якщо це станеться, буде викликаний метод onPause ().

onPause (): цей метод використовується для припинення дій, які не повинні відбуватися, поки Activity в стадії паузи. Виклик цього методу вказує на те, що користувач покинув додаток. Наприклад, вхідний дзвінок може перевести, програвач музики в стан паузи. Це повинно заглушити або зупинити відтворюється музику.

onStop (): цей метод викликається, якщо Activity більше не видно в додатку. Таке може статися, якщо довантажуючи інша Activity і вона займає весь екран пристрою. Коли викликає цей метод, Activity повідомляється перейти в стан припинення роботи. У цьому стані, система або викликає onRestart () для повернення взаємодії з Activity, або викликає метод onDestroy (), щоб прибрати Activity.

onDestroy (): цей метод викликається перед тим, як Activity буде завершена. Система викликає цей метод, коли користувач завершує Activity, або якщо система тимчасово прибирає процес, що містить Activity, для вивільнення місця. З цього методом, обов'язково звільніть будь-які ресурси, створені вашої Activity, інакше ваш додаток буде мати витік пам'яті.

onRestart (): це викликається, якщо Activity перезапускается, після того, як було зупинено.

3.2.2. Фрагменти

Активіті є досить тяжкими ресурсномісткими та система Анднроїд витрачає багато ресурсів для їх створення.

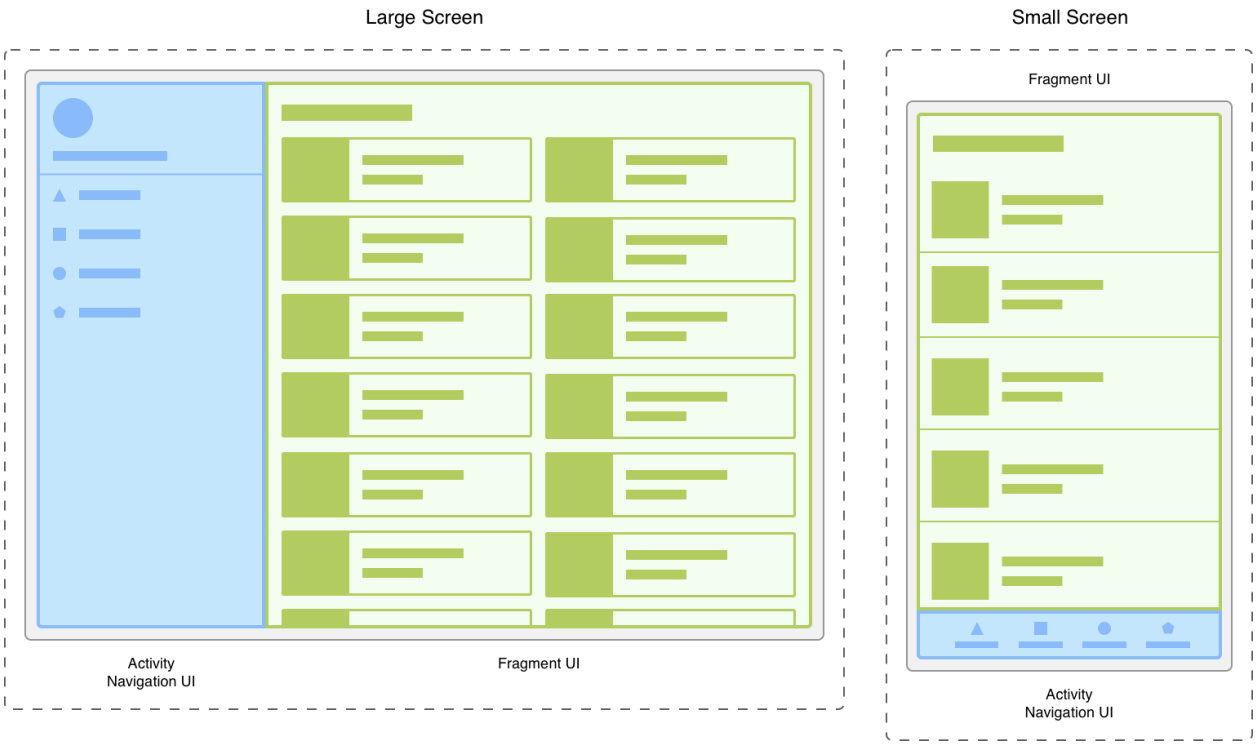

Рисунок 10 - Приклад фрагменту на планшеті та телефоні

У фрагментів, як і к активіті є життєвий цикл. Вони дещо схожі та детально життєвий цикл фрагменту можна побачити на рисунку 3.

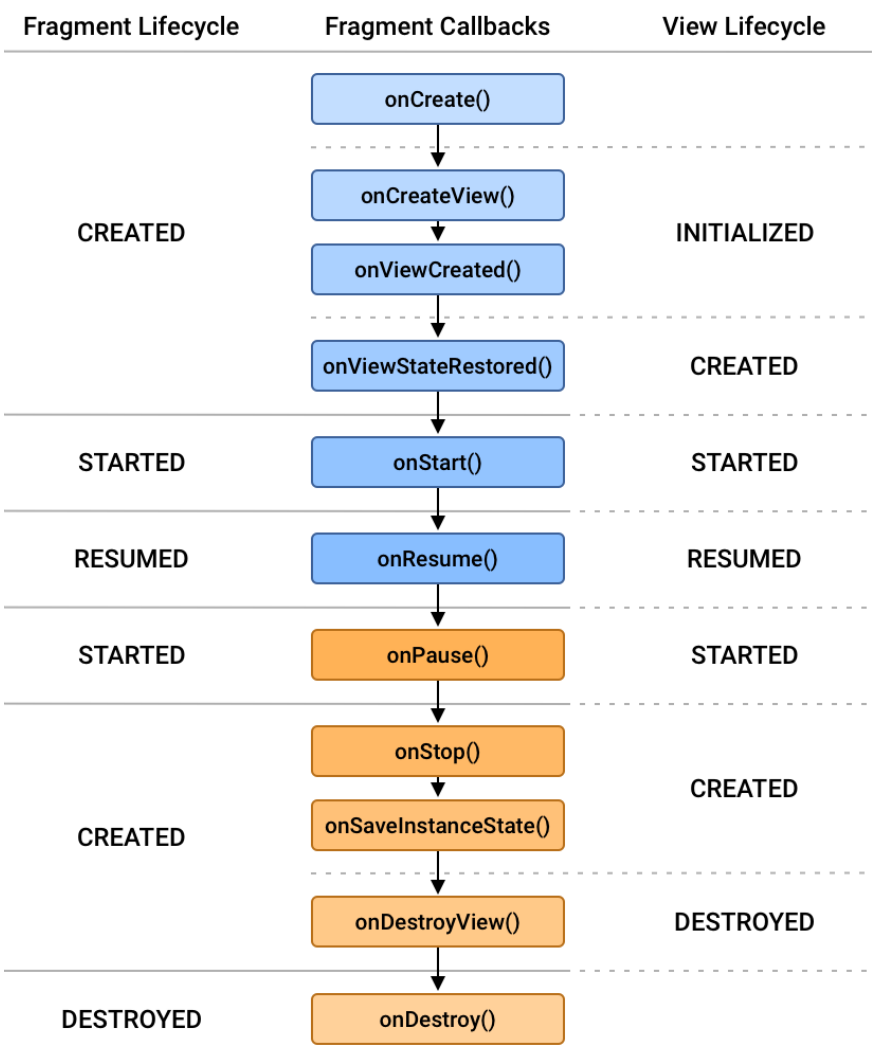

Рисунок 11 – Життєвий цикл фрагмента

4. Розробка дизайну

4.1. Створення UI дизайну

UX-дизайн відповідає за функції, адаптивність продукту і те, які емоції він викликає у користувачів. Чим зрозуміліше інтерфейс, тим легше користувачеві отримати результат і зробити цільове дію. UX-дизайн - це проектування інтерфейсу на основі досліджень користувацького досвіду і поведінки.

Рисунок 12 – UX-дизайн головного меню додатку

Перед нами стоїть задача створити інтуїтивно простий інтерфейс. Розмістимо на головній сторінці у шапці назву додатку. Додамо список з дій, який буде займати увесь вільний простір. Для додавання дій передбачимо

кнопку снизу екрану у його правій частині, щоб користувачу не було потрібно тягнутися до неї.

Рисунок 13 – UX-дизайн екрану для детального перегляду дії

Натискаючи на один з айтемів списку дій відкривається детальна інформація о дії. Користувач бачить назву дії, її мінуси та плюси. Також, користувач бачить скільки залишилось часу, щоб вирішити, чи потрібно робити дію чи ні. Щоб уникнути випадкового редагування інформації, вона на цій сторінці лише відображується. Для її редагування потрібно натиснуту

спеціальну кнопку. Вікно для редагування дії відрізняється від вікна для відображення.

Рисунок 14 – UX-дизайн екрану для редагування дії

У вікні для редагування поля з текстом займаюсь увесь вільний простір.

Наразі ми вже маємо створений UX-дизайн, тому переходимо до створення UI частини.

4.2. Створення UX дизайну

Для створення дизайну будемо використовувати методологію створення дизайну – Material Design. Останнім часом «Матеріал дизайну» став дуже популярним, оскільки багато дизайнерів почали інтегрувати його не лише у

свої Android-проекти, а й у інші проекти, пов’язані з Інтернетом. Концепція вперше була представлена компанією Google в основній програмі Google I / O 2014. На сьогоднішній день цю презентацію переглядали понад 1,5 мільйона разів, і вона все ще розглядається як основний огляд того, що таке "матеріальний дизайн" і як ми повинні про це думати.

Тож, у чому полягає “Material Design”?

1. Ознайомтесь з основним ресурсом

Щоб навчитися дизайну матеріалів, найкраща можлива відправна точка - офіційний ресурс, оприлюднений Google. Він постійно оновлюється, і він пояснює всі дрібні деталі, що входять до Material design. Тут чудово те, що він не охоплює лише специфічні для Android аспекти дизайну матеріалів, а обговорює дизайн матеріалу в цілому стосовно будь-якого проекту програми чи навіть веб-дизайну.

2. Використовувати тіні для передачі ієрархії

Поверхня, краї, реалістичні тіні та освітлення - основні інструменти дизайну матеріалів. Додавання глибини у конструкції має вирішальне значення, але потрібно бути обережним лише для використання мінімальної ефективної дози. Наприклад, тіні - це головний інструмент для передачі ієрархії різних елементів, що поєднуються в повноцінний дизайн. Вирішуючи, що кидає малу реалістичну тінь на що, ви представляєте візуальну ієрархію елементів та шарів, на яких вони розміщені. Тут важлива загальна структура дизайну і якщо тіньова структура в цілому має сенс для людського ока - якщо вона зображує концепцію реальних матеріалів.

3. Використовувати яскраві кольори

Матеріал дизайну, безумовно, є мінімалістичним типом дизайну. Іншими словами, не вдається використовувати багато інструментів дизайну та стилістичних смаків. Тому дизайнерам доводиться обійти це обмеження та

знайти інший спосіб створити сенс та створити правильний фокус. Одне з небагатьох речей, що залишилося - це колір. Точніше, сміливий колір і часто кричущий. Сміливі кольори відіграють дійсно важливу роль у дизайні матеріалів (і плоский дизайн теж з цього приводу), вони роблять речі цікавими та роблять взаємодію з дизайном приємним для користувача.

4. Використовуйте основний колір та колір акценту

Офіційні документи Google кажуть: «Обмежте свій вибір кольорів, вибравши три відтінки з первинної палітри та один акцентний колір із вторинної палітри.»

Способом адаптувати це до будь-якого типу дизайну може бути: виберіть три відтінки, які становлять вашу основну палітру, плюс один колір, який буде виступати як акцент. Основні кольори можна використовувати для таких речей, як фони, поля, поля, шрифти та інші ключові елементи інтерфейсу. Колір акценту - це саме те, що підказує назва - це дає додаткові можливості, коли ви хочете відобразити головний елемент на заданому екрані або сторінці. Зайве говорити, що акцентний колір повинен бути сильно контрастним з первинними відтінками.

5. Усе є в деталях

Один з головних елементів, який робить дизайн матеріалу настільки важким для виконання бездоганно, це те, що він настільки спрощений.

6. Використовувати пробіли

Матеріал дизайну отримує багато ідей від традиційного дизайну друку та принципів, які він нам приніс. Наприклад, пробіл є важливим елементом у будь-якому дизайні матеріалів, і він може значно покращити типографіку та макет тексту. Насправді пробіли - це найефективніший інструмент для створення фокусу, привернення уваги користувача та приведення його до певного елемента.

Отже, коротко кажучи, треба використовувати велику типографіку для заголовків, додавати багато пробілів і не боятися мати загальну кількість порожніх пробілів у дизайні.

7. Використовувати зображення від краю до краю

Дизайн матеріалів справді сприймає зображення. Зображення в Material Design розміщені "від краю до краю" - з повним кровотоком. Це означає, що між краєм зображення та краєм екрана немає меж. Якщо все зроблено правильно, це створює захоплюючий досвід для користувача, а також дає нам деякі додаткові інструменти дизайну серед досить невеликого набору попередньо затверджених тіней, кольорової палітри та шарів.

8. Для дизайну на основі зображень витягуйте кольори із зображень

Говорячи про зображення, Google закликає нас витягувати кольори із зображень, які ми використовуємо в нашому дизайні, а потім робити їх частиною кольорової палітри. Важко сперечатися з цим міркуванням. Роблячи таку річ, обов’язково створіть послідовний досвід для користувача, створіть враження, що все стає на свої місця красиво і просто підходить загалом.

9. Включити рух

Рух - одна з ключових складових хорошого дизайну матеріалів. Зрештою, ми звикли відчувати рух у реальному світі. Це допомагає нам зрозуміти, як все працює і куди нам слід спрямувати свою увагу. Матеріал дизайну використовує той самий принцип і використовує рух для взаємодії з користувачем, ефективно даючи їм знати, як використовувати дизайн.

Що ввести в рух? Просто треба додати користувачеві відгук про дію, яку вони щойно виконували. Наприклад, чи натиснули кнопку? Анімуйте це, щоб підтвердити, що вхід був отриманий.

10. Зробіть рух автентичним

Тут є ключовим словом "Автентичний". Дні фальшивого руху - речі, які просто рухаються навколо екрану - давно минули. Google присвячує цілий розділ в керівництві щодо дизайну матеріалів концепції автентичного руху. У цих рекомендаціях вони пояснюють, як вводити масу і вагу, прискорення і уповільнення, і як працює ослаблення (спосіб зробити анімацію здається більш природним, змінюючи швидкість руху в різні моменти часу). У матеріальному дизайні рух так само гарний, як і його здатність імітувати рух предметів реального життя. Це єдиний спосіб, коли рух збагатить інтерфейс і зробить його більш зрозумілим для користувача.

11. Робити все чуйним

Один з головних принципів дизайну матеріалів - зробити отриману роботу доступною та зручною для використання на будь-якому пристрої та будь-якого розміру екрана. Перш за все, мета - зробити досвід послідовним. Таким чином, користувач не заплутається, якщо переключається з одного пристрою на інший, оскільки він не отримує абсолютно новий інтерфейс для них. Завдяки хорошому дизайну вони можуть переходити без будь-яких перешкод і просто продовжувати користуватися додатком прямо з того моменту, коли вони закінчилися. Природно, це означає, що дизайн повинен бути чуйним. На щастя, за допомогою сучасних рамок ви отримуєте велику частину вже побудованих будівельних лісів, тож зробити вашу роботу чутливою не повинно бути великим завданням.

На офіцфйному сайті Material Design обираємо кольори, які будут використовуватись у додатку.

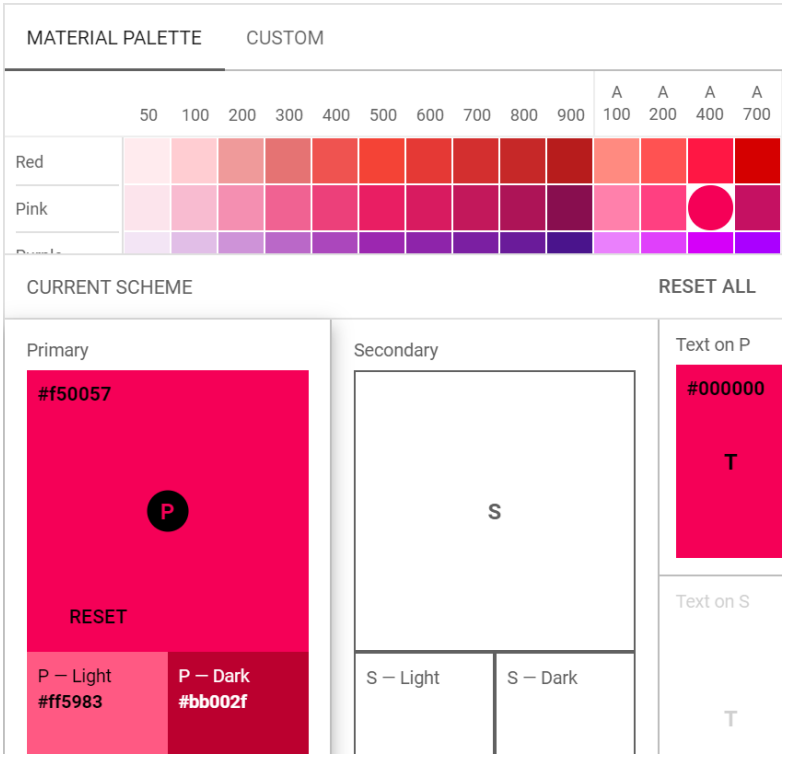

Рисунок 15 – Основний колір для додатку

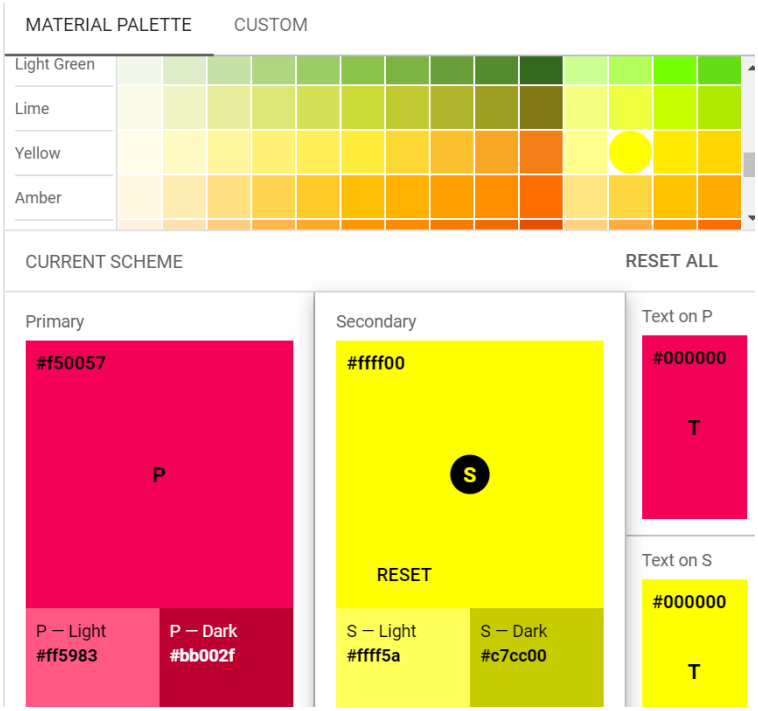

Рисунок 16 - Другорядний колір для додатку

Тож, мною було обрано кольори. Тепер, переходимо до створення UI-дизайну. перекладається як «призначений для користувача інтерфейс». І необов'язково тільки графічний: тактильний, голосовий або звуковий. UI-дизайн - процес візуалізації прототипу, який розробили на підставі

призначеного для користувача досвіду і дослідження цільової аудиторії. UI-дизайн включає в себе роботу над графічною частиною інтерфейсу: анімацією, ілюстраціями, кнопками, меню, слайдерами, фотографіями та шрифтами. У результаті розробки UI-дизайну, я отримав наступні прототипи: На головному екрані айтеми знаходяться на жовтому фоні.

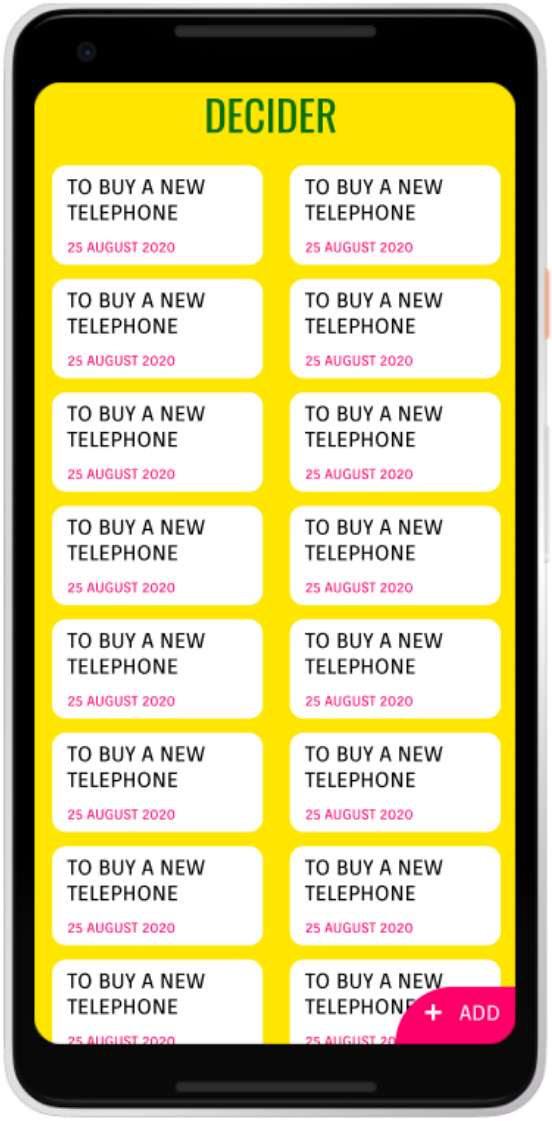

Рисунок 17 - UI-дизайн головного меню додатку

Також, мною було вирішено додати кнопку для додавання дій у правий нижній кут, щоб користувачу не потрібно було тягнутися до неї. Вона була виділена одним з основних кольорів, щоб не зливатися з фоном.

Рисунок 18 - UI-дизайн екрану для детального перегляду дії

Як можемо бачити на екрані (Рисунок 18), мінуси та плюси здійснення дії знаходяться поряд, щоб користувачу було легше робити висновок, чи потрібно її здійснювати, чи ні.

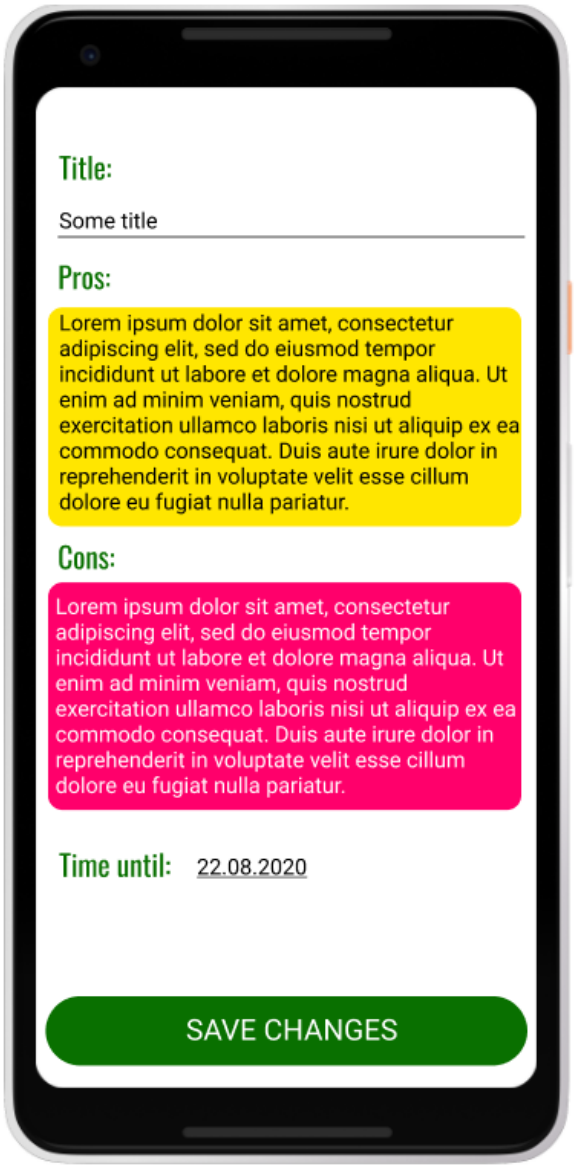

Рисунок 19 - UI-дизайн екрану для редагування дії

Як бачимо на Рисунку 19 поля з інформацією о дії займають усю довжину екрану та щоб користувачу було легше орієнтуватися у інтерфейсі ми кольорами відказуємо, де знаходяться так звані «плюси», а де «мінуси» дії.

4.3. Створення візуальної частини додатку

Макет визначає структуру для користувальницького інтерфейсу у вашому додатку, наприклад, в діяльності. Усі елементи в макеті побудовані з використанням ієрархії об'єктів View та ViewGroup. Вид зазвичай малює те, що користувач може бачити та взаємодіяти. Тоді як ViewGroup - це невидимий контейнер, який визначає структуру макета для View та інших об’єктів ViewGroup, як показано на малюнку 1.

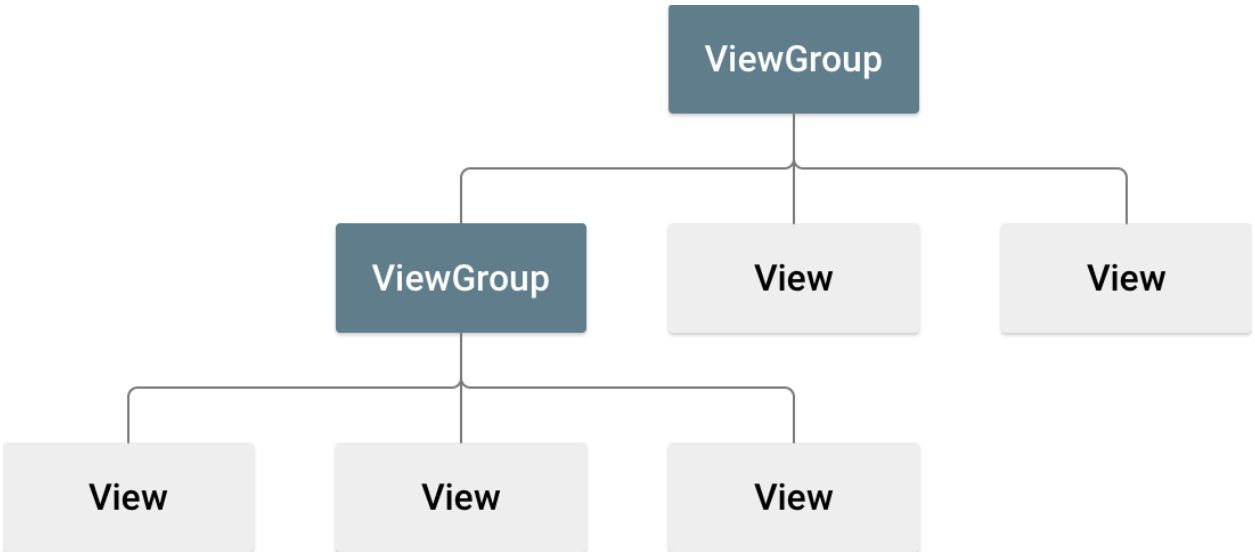

Рисунок 18 - Ієрархія комонентів інтерфейсу

Створення візуальної частини додатку здійснюється за допомогою xml-коду.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World" />
<Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="It is a Button" />
</LinearLayout>
```

Рисунок 9 – Приклад фрагменту коду для створення візуальної частини з підсвічуванням певних його частин

5. Розробка функціоналу

5.1. Вибір архітектури додатку

У якості архітектури додатку обираємо архітектура компонентів. Вона ж Architecture Components – це набір рішень від Google, які допоможуть у створенні додатку. Те, який вигляд має архітектура можна побачити на рисунку 4.

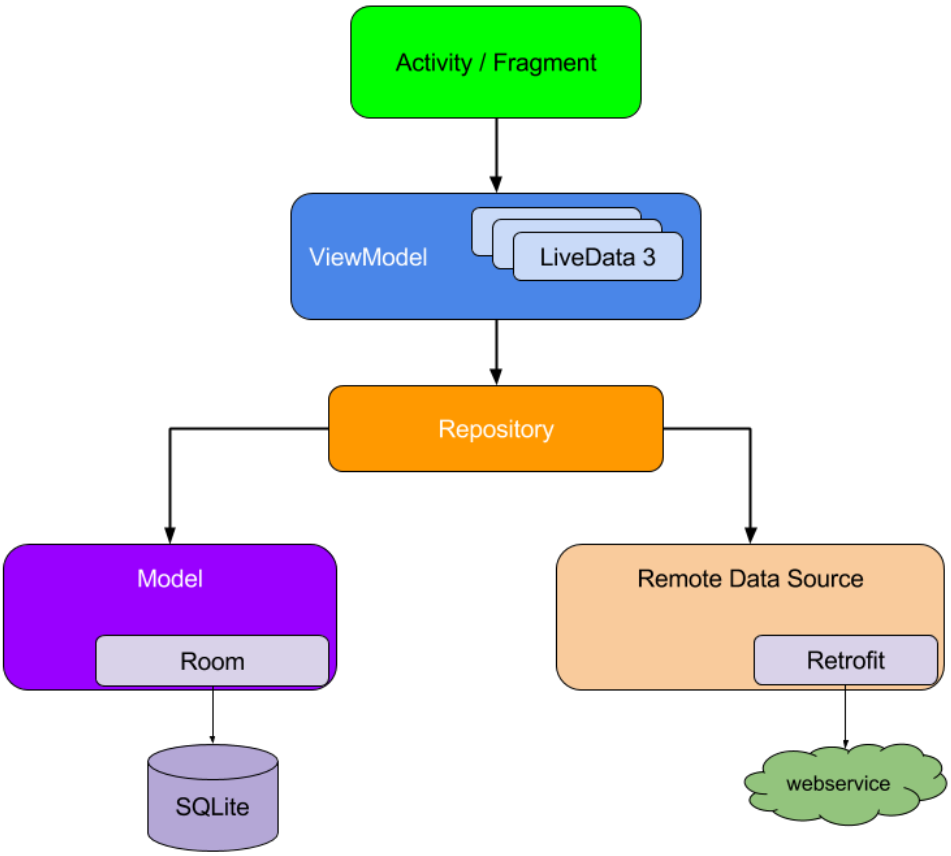

Рисунок 19 – Архітектура, взята за основу для додатку

В цілому Android Architecture Components можна розділити на чотири блоки: Lifecycles\UI (Activity, Fragment), ViewModel, Repository та джерела даних. Перейдемо до розбору кожного із них.

UI

UI складається з таких компонентів, як Activity та Fragment. Вони відповідають за візуальну частину додатку. Кожен із компонентів данної архітектури має відповідати тільки за вузько направлену дію. Як було зазначено вище, у випадку UI – це відображення інформації, візуальна частина додатку. Саме тому компоненти UI мають виконуватись у головному потоці, окремо від бізнес логіки, взаємодії з базами даних чи іншими ресурсами даних. Це допоможе зробити роботу UI швидкою та стабільною.

ViewModel

ViewModel - клас, що дозволяє активність та фрагменти зберігають необхідні їм об'єкти живими при повороті екрану. Клас ViewModel призначений для зберігання та управління даними, пов’язаними з користувацьким інтерфейсом, у свідомому життєвому циклі. Клас ViewModel дозволяє даним переживати зміни конфігурації, такі як обертання екрана.

Фреймворк Android управляє життєвими циклами контролерів інтерфейсу користувача, такими як дії та фрагменти. Фреймворк може вирішити знищити або заново створити контролер інтерфейсу користувача у відповідь на певні дії користувача або події пристрою, які повністю поза вашим контролем.

Якщо система знищує або відтворює контролер інтерфейсу користувача, будь-які перехідні дані, пов’язані з інтерфейсом користувача, які ви зберігаєте в них, втрачаються. Наприклад, ваш додаток може включати список користувачів в одній зі своїх дій. Коли дія відтворюється повторно для зміни конфігурації, нова дія повинна повторно отримати список користувачів. Для простих даних діяльність може використовувати метод onSaveInstanceState () і відновити його дані з набору в onCreate (), але цей підхід підходить лише для невеликих обсягів даних, які можна серіалізувати, а потім десеріалізувати, а не для потенційно великих обсягів даних як список користувачів або растрові зображення.

Компонент Lifecycle - покликаний спростити роботу з життєвим циклом. Виділено основні поняття такі як LifecycleOwner і LifecycleObserver.

LifecycleOwner - це інтерфейс з одним методом getLifecycle (), який повертає стан життєвого циклу. Являє собою абстракцію власника життєвого циклу (Activity, Fragment). Для спрощення додані класи LifecycleActivity і LifecycleFragment.

LifecycleObserver - інтерфейс, позначає слухача життєвого циклу owner-а. Не має методів, зав'язаний на OnLifecycleEvent, який в свою чергу дозволяє відстежувати життєвий цикл.

Repository

Репозиторії є абстракцією над джерелами даних, яку додаток може використовувати для отримання даних та їх кешування. Ця абстракція корисна з двох основних причин:

- код не залежить від конкретної реалізації сховища даних, а також
- в наслідок попередньої причини, ми можемо легко змінювати конкретні реалізації сховища даних, наприклад, для тестування

5.2. Розробка бізнес логіки

Для розробки додатку буде використовуватись середа розробки Android Studio від JetBrains. За її допомогою створюємо візуальну частину додатку. Треба підібрати потрібний layout, бо від цього залежить те, як будуть розміщуватись елементи. Для нашого додатку будемо використосувати ConstraintLayout.

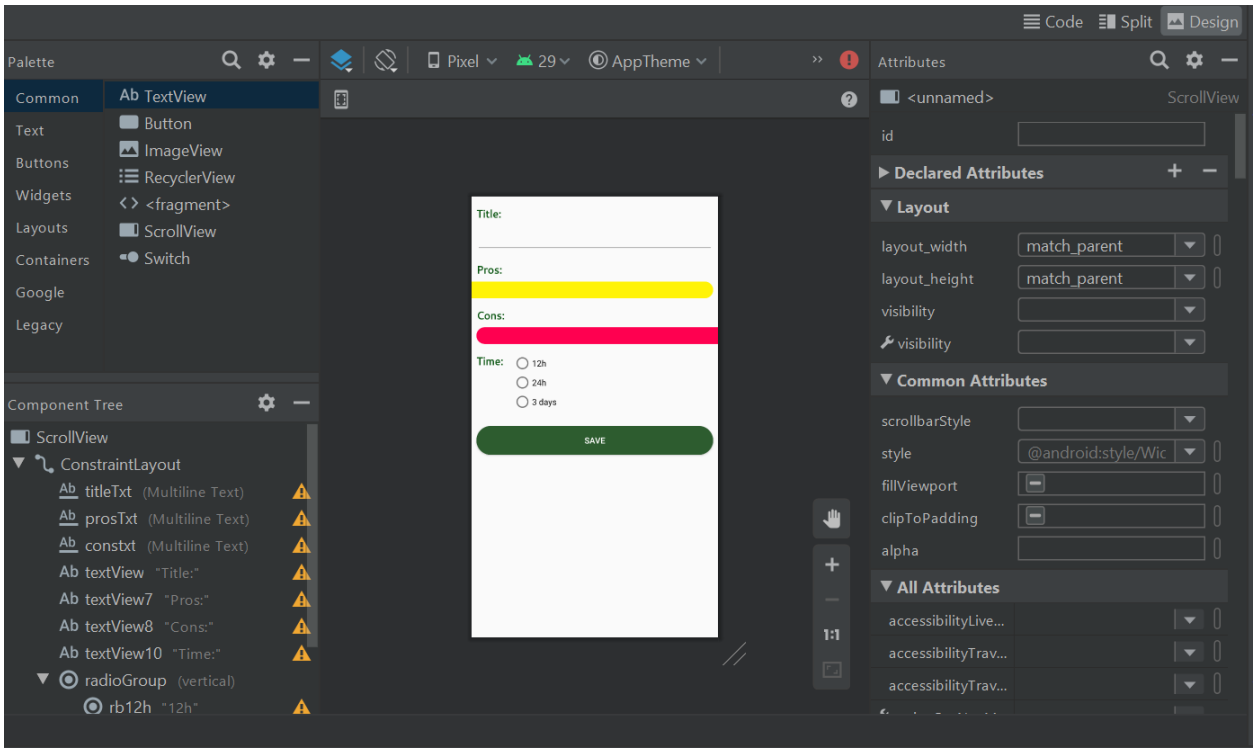

Рисунок 20 - Створення візуальної частини

Середа для розробки Android Studio трансформує макет, який ми створюємо у xml – формат. Для найкращого результату не використовуємо лише xml–код чи інструменти для дизайну Android Studio, а комбінуємо їх.

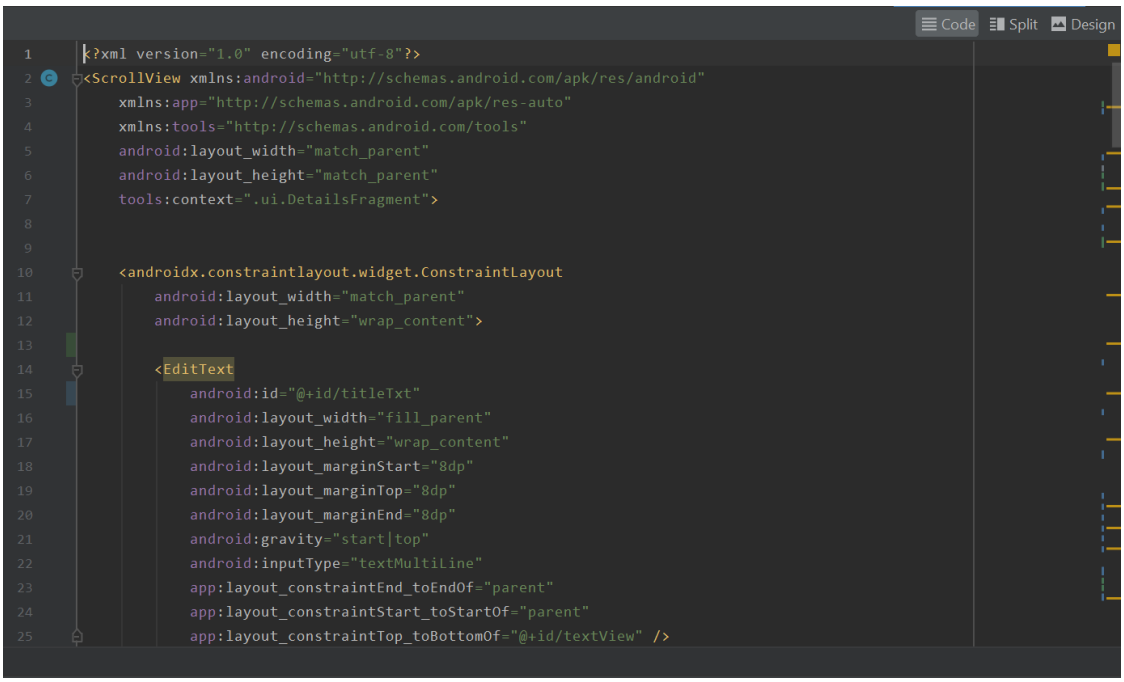

Рисунок 21 - Створення візуальної частини за допомогою коду

Розміщуємо елементи користувацького інтерфейсу вказуючи їх місцезнаходження залежно від інших елементів. Як нам відомо, платформа андроїд встановлена на девайсах з самими різними розмірами екрану, тому такий підхід у розміщенні елементів допоможе додатку коректно відображатись на екранах більшості форматів телефонів.

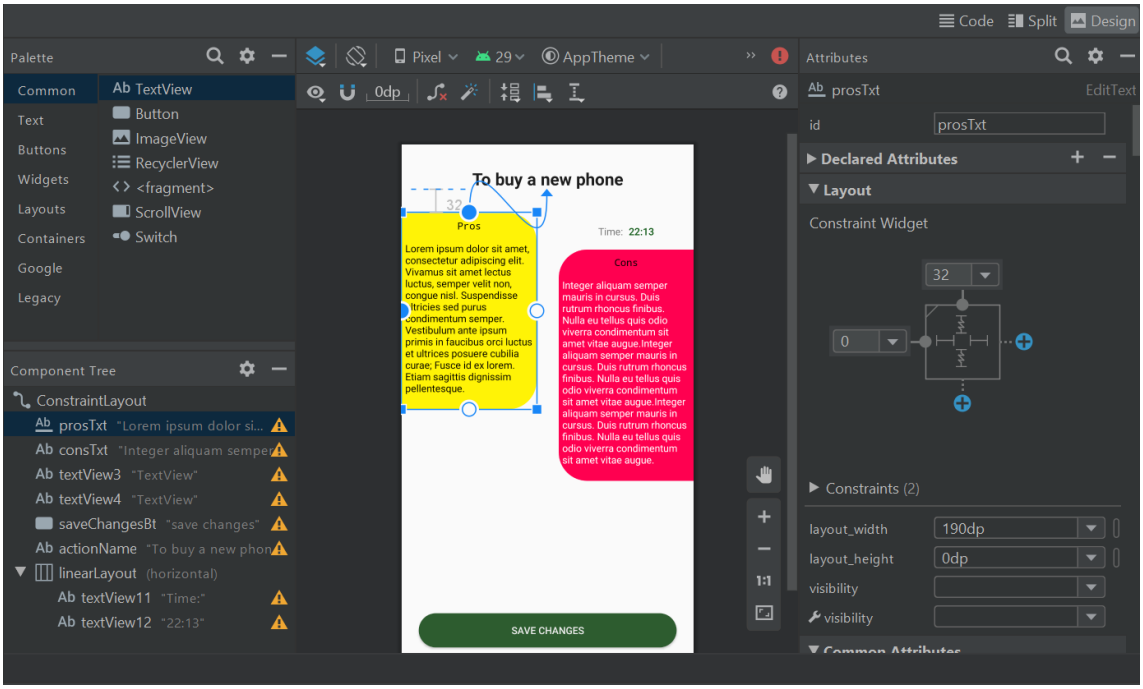

Рисунок 22 - Створення екрану для просмотру та редагування дії

Використовуємо фрагменти для відображення інформації для користувача. Для переключення між фрагментами використовуємо Navigation Components. Для цього будуємо граф у папці, яку помічаємо як папку з ресурсами для навігації

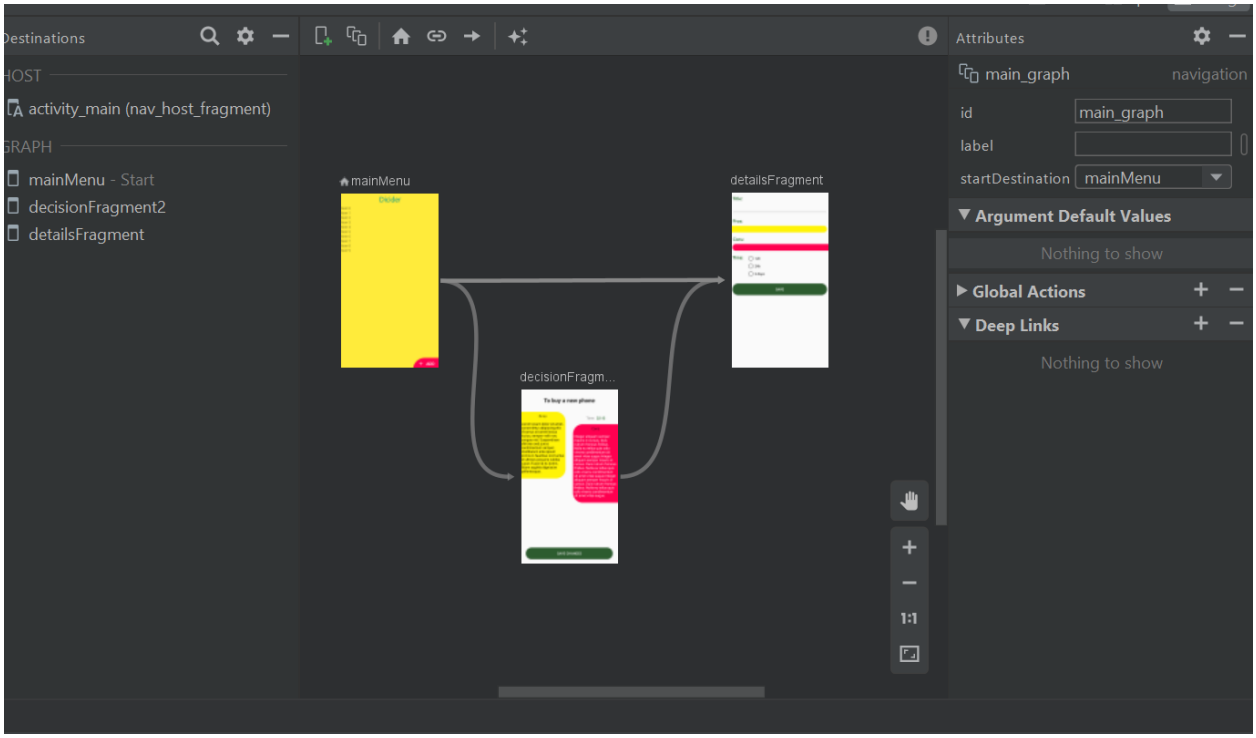

Рисунок 23 - Навігація між екранами

Також, ми можемо побудувати граф за допомогою коду.

```
<?xml version="1.0" encoding="utf-8"?>
<navigation 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:id="@+id/main_graph"
    app:startDestination="@id/mainMenu">

    <fragment
        android:id="@+id/mainMenu"
        android:name="com.example.decision.ui.MainMenu"
        android:label="fragment_main_menu"
        tools:layout="@layout/fragment_main_menu" >
```

```
            <action
                android:id="@+id/action_mainMenu_to_decisionFragment2"
                app:destination="@id/decisionFragment2" />
            <action
                android:id="@+id/action_mainMenu_to_detailsFragment"
                app:destination="@id/detailsFragment" />
        </fragment>
        <fragment
            android:id="@+id/decisionFragment2"
            android:name="com.example.decision.ui.DecisionFragment"
            android:label="fragment_decision"
            tools:layout="@layout/fragment_decision" >
            <action

    android:id="@+id/action_decisionFragment2_to_detailsFragment"
                app:destination="@id/detailsFragment" />
        </fragment>
        <fragment
            android:id="@+id/detailsFragment"
            android:name="com.example.decision.ui.DetailsFragment"
            android:label="fragment_details"
            tools:layout="@layout/fragment_details" />
    </navigation>
```

Переходимо до бізнес логіки. Загальна структура проекту буде виглядати наступним чином.

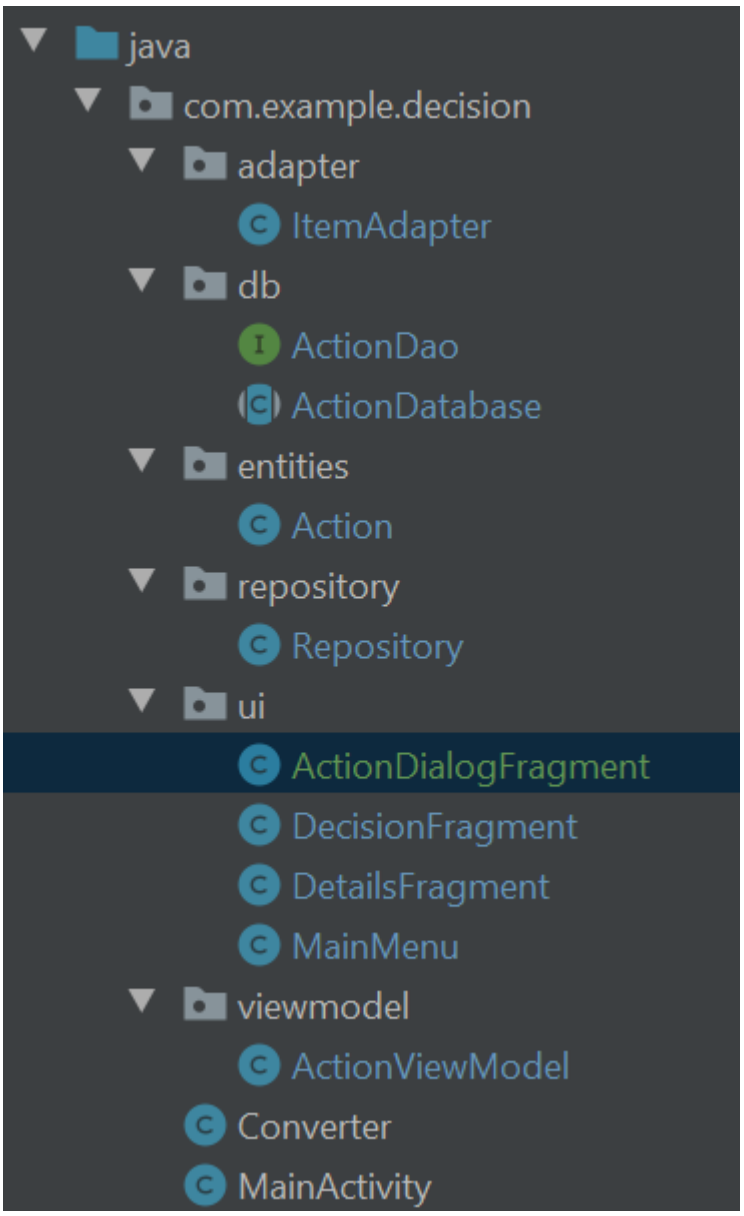

Рисунок 24 - Структура проекту

Entities

Головною сутністю у нашому додатку буде дія – Action. Для того, щоб перетворит объект на запис у таблиці використовуємо анотації.

Entity (tableName = "actions")	Назва таблиці у базі даних.
@NonNull	Означає, що поле не може бути не ініціалізованим
@PrimaryKey	Означає, что з цого поля буде складатися ключ для запичу у базі даних.

Назви усіх полів є інтуїтивно простими та дають зрозуміти, що вони означають.

```
      @Entity (tableName = "actions")
public class Action implements Serializable {
    @NonNull
    @PrimaryKey
    private String title;
    private String pros;
    private String cons;
    private Calendar deadline;

    public Action(String title) {
        this.title = title;
        deadline = new GregorianCalendar();
    }

     }
```

db

У цій папці ми маємо два класи.

Клас ActionDatabase наслідує клас RoomDatabase та саме за його допомогою ми будемо створювати об’єкт бази данних. Для цього класу ми використовуэмо статичний метод getDatabase, щоб можна було використовувати класс у якості синглтону.

```
     @Database(entities = {Action.class}, version = 1)
@TypeConverters(Converter.class)
public abstract class ActionDatabase extends RoomDatabase {
    private static ActionDatabase INSTANCE;
    private static final int NUMBER_OF_THREADS = 4;

    public abstract ActionDao actionDao() throws
SQLiteConstraintException;

    public static final ExecutorService databaseWriteExecutor =
            Executors.newFixedThreadPool(NUMBER_OF_THREADS);

    public static ActionDatabase getDatabase(final Context context){
        if (INSTANCE == null) {
            synchronized (ActionDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                            ActionDatabase.class, "action-database")
                            .fallbackToDestructiveMigration()
                            .build();
                }
            }
        }
        return INSTANCE;
    }
}
```

Інший інтерфейс допомагає робити запити до бази даних

```
     @Dao
public interface ActionDao {
    @Query("SELECT * FROM actions")
    Flowable<List<Action>> getAllActions();

    @Query("SELECT * FROM actions WHERE title = :title")
    Action getActionByTitle(String title);

    @Query("SELECT * FROM actions")
    List<Action> getActionList();

    @Query("DELETE FROM actions")
    void deleteAllActions();

    @Delete
    void deleteAction(Action action);

    @Insert
    void insertAction(Action action) throws SQLiteConstraintException;

}
```

Adapter

Для роботи з Recycler View нам потрібек клас з адаптером. Перивизначаємо матоди для адаптеру.

```
@NonNull
@Override
public ItemAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
    return new
ViewHolder(ItemXmlBinding.inflate(LayoutInflater.from(parent.getContex
t()), parent, false));
```

```
}

@Override
public void onBindViewHolder(@NonNull ItemAdapter.ViewHolder holder,
int position) {
    if (list != null) {
        if(list.get(position) != null){

holder.itemXmlBinding.title.setText(list.get(position).getTitle());

holder.itemXmlBinding.time.setText(list.get(position).getDeadline().ge
t(Calendar.DATE));
        } else{
            Log.e("Object is null", "onBindViewHolder: item is null");
        }
    } else {
        Log.e("List of objects is null", "onBindViewHolder: item list
is null");
    }
}

@Override
public int getItemCount() {
    return list != null  && list.size() > 0 ? list.size() : 0;
}
```

Також, сворюємо внутрєшній клас для зв'язування айтемів та листа айтемів.

```
     public class ViewHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
    private ItemXmlBinding itemXmlBinding;
    public ViewHolder(ItemXmlBinding itemXmlBinding) {
        super(itemXmlBinding.getRoot());
        this.itemXmlBinding = itemXmlBinding;
```

```
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if (getAdapterPosition() != RecyclerView.NO_POSITION){
            Bundle bundle = new Bundle();
            bundle.putString(ACTION_ID,
list.get(getAdapterPosition()).getTitle());

Navigation.findNavController(v).navigate(R.id.action_mainMenu_to_decis
ionFragment2, bundle);
        }
    }
}
```

ViewModel

Створюємо вьюмодел для нашого додатку. ViewModel містить об’єкт репозіторию, саме тому ми передаємо об’єкт класу Application у якості параметру у конструкторі.

```
 public class ActionViewModel extends AndroidViewModel {
    private Repository repository;
    private MutableLiveData<List<Action>> actionsData;
    private Disposable disposable;

    public ActionViewModel(@NonNull Application application) {
        super(application);
        repository = Repository.getInstance(application);
        actionsData = new MutableLiveData<>();
        subscribeOnDbUpdates();
    }

    public Action getActionByTitle(String title){
```

```
        Action result;
        ActionDatabase.databaseWriteExecutor.execute(() -> {
            result =
repository.getDatabase().actionDao().getActionByTitle(title);
        });
        return result;
    }

    public void loadActions(){
        ActionDatabase.databaseWriteExecutor.execute(() -> {

actionsData.postValue(repository.getDatabase().actionDao().getActionLi
st());
        });
    }

    public void insertAction(Action action) throws
SQLiteConstraintException {
        ActionDatabase.databaseWriteExecutor.execute(() -> {
            repository.getDatabase().actionDao().insertAction(action);
        });
    }

    public MutableLiveData<List<Action>> getActionLiveData() {
        return actionsData;
    }

    @SuppressLint("CheckResult")
    public void subscribeOnDbUpdates(){
        if(disposable != null && !disposable.isDisposed()){
            disposable.dispose();
        }
        disposable = repository
                .getDatabase()
```

```
                .actionDao()
                .getAllActions()
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(actions -> actionsData.postValue(actions));
    }
}
```

UI

Для роботи з візуальною частиною додатку маэмо 3 класи, що наслідуються від класу Fragment: DecisionFragment, DetailsFragment, MainMenu.

В усіх трьох класах перевиначаємо метод onCreate.

```
     @Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    viewModel =
ViewModelProviders.of(this).get(ActionViewModel.class);
}
```

Для класу MainMenu використовуємо методи для ініціалізації списку з даними та підписки на дані з бази даних.

```
     private void initRecyclerView(){
    adapter = new ItemAdapter();
    binding.recyclerView.setAdapter(adapter);
}

private void subscribeOnLiveData(){
    viewModel
            .getActionLiveData()
            .observe(getViewLifecycleOwner(), actions -> {
                setActionList(actions);
```

```
                insertFilmsIntoRecyclerView(actions);
            });
}
```

Для класу DecisionFragment створюємо метод, який буде виконувати певну дію при натиску на кнопку зберегти. Від перевіряє, чи була введена назва дії та у разі вводу даних у це поле, зчитує дані з інших полей та створює дії з відповідними даними.

```
     @Override
public void onClick(View v) {
    String text = binding.titleTxt.getText().toString();
    if(text.length() < 1){
        Toast.makeText(
                getContext(),
                "The title must consist of at least one symbol",
                Toast.LENGTH_LONG ).show();
    } else {
        Action newAction = new
Action(binding.titleTxt.getText().toString());
        newAction.setCons(binding.constxt.getText().toString());
        newAction.setPros(binding.prosTxt.getText().toString());
        GregorianCalendar calendar = new GregorianCalendar();
        if(binding.rb12h.isSelected()){
            calendar.add(GregorianCalendar.HOUR, 12);
        } else if (binding.rb24h.isSelected()){
            calendar.add(GregorianCalendar.HOUR, 24);
        } else {
            calendar.add(GregorianCalendar.HOUR, 24*3);
        }
        newAction.setDeadline(calendar);
        try {
            viewModel.insertAction(newAction);
```

```
Navigation.findNavController(binding.getRoot()).popBackStack();
        } catch (Exception e){
            new AlertDialog.Builder(getActivity())
                    .setMessage("An action with this name is already
exist. Please input another name")
                    .setTitle("Error")
                    .create()
                    .show();
        }
    }
```

Для класу DecisionFragment перевизначаємо метод, щоб занести дані до відповідних полей.

```
     @Override
public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    View.OnClickListener textViewListener = v -> {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
        builder
                .setMessage(((TextView)v).getText())
                .setTitle(((TextView)v).getContentDescription());
        AlertDialog dialog = builder.create();
        dialog.show();
    };
    binding.actionName.setText(action.getTitle());
    binding.prosTxt.setText(action.getPros());
    binding.prosTxt.setOnClickListener(textViewListener);
    binding.consTxt.setText(action.getCons());
    binding.consTxt.setOnClickListener(textViewListener);
}
```

6. Тестування

Запустимо додаток та створимо дію у ньому. На головному екрані (Рисунок 25) натискаємо на кнопка «+ Add», щоб додати дію. Вводимо дані та натискаємо на «Save», щоб зберегти дію (Рисуок 26).

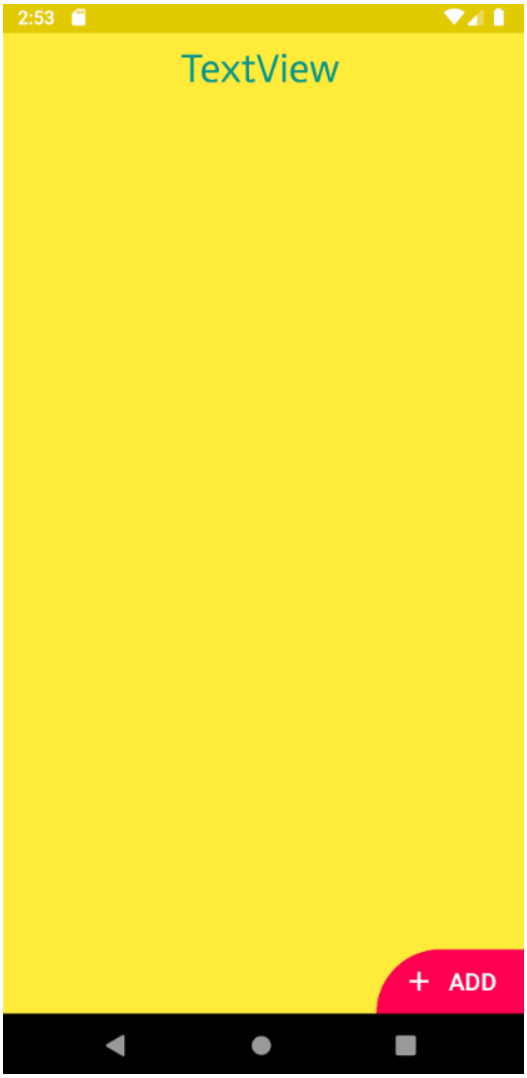

Рисунок 25 – Головний екран без створених дій

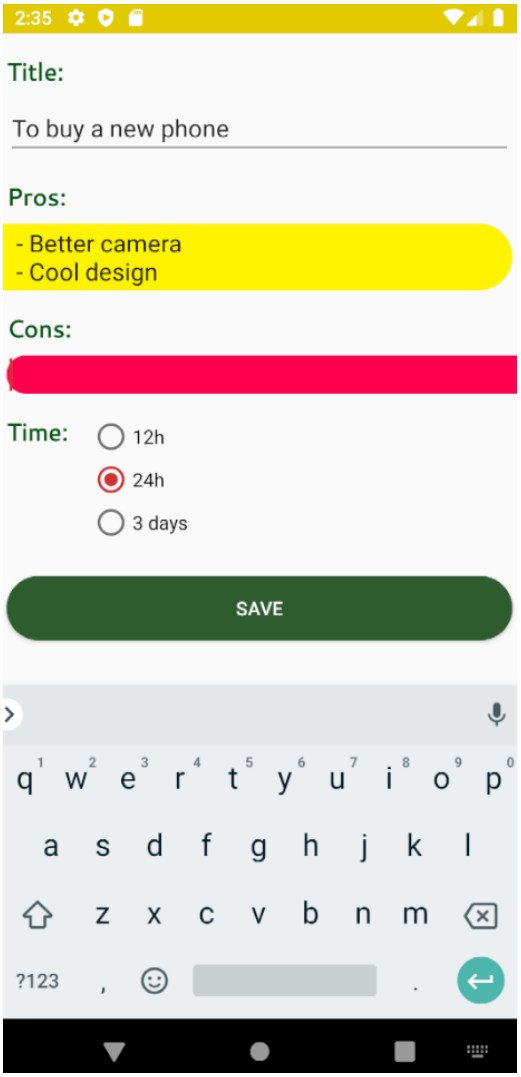

Рисунок 26 – Створення дії

Бачимо, що дія була створена. Вона відображається коректно.

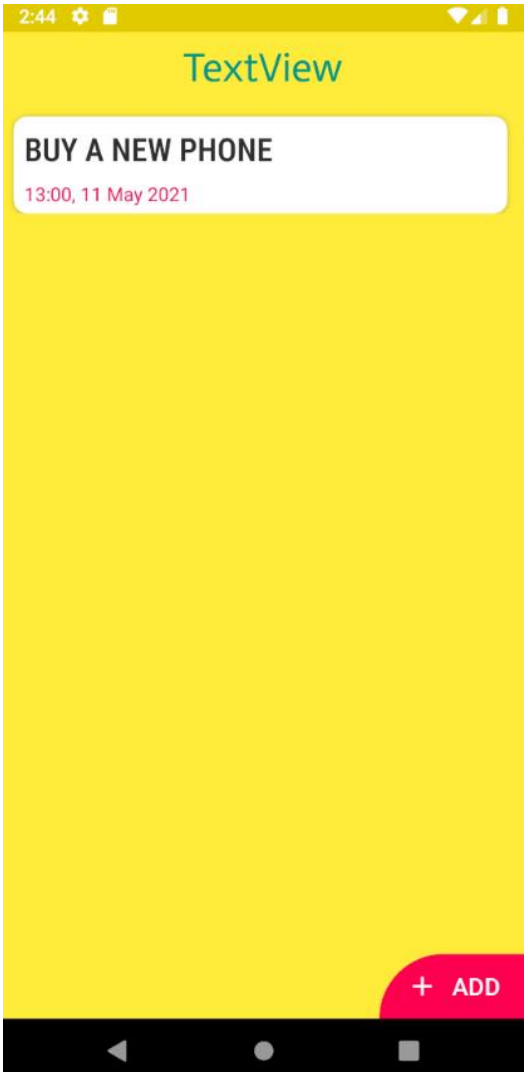

Рисунок 27 - Головний екран із створеною дією

Також, додато ще дій, щоб впевнитись, що вони відображуються коректно.

Рисунок 28 - Головний екран з декількома діями

Висновки

В результаті написання дипломної роботи, було досліджено існування проблеми з імпульсивними діями. Було розглянуто, які є методи запобіганню імпульсивним діям. Виходячи з методів запобіганню імпульсивним діям було вирішено, яким функціоналом повинен володіти додаток. UX-дизайн створювався опираючись на потрібний фунціонал додатку. Було приділено достатньо уваги як UX- дизайну, так і UI-дизайну. UI-дизайн створювався спираючись на методологію створення дизайну Material Design. Ця методологія є актуальною и рекомендована компанією Google, яка і займається розвитком операції системи Android. Були обрані фірмові кольори для додатку. Це допомогло зробити мобільний додаток приємним для користувача. Була розроблена візуальна частина додатку, гідно з ескізами UI-дизайну. Було зроблено огляд технологій, які використовуються для створення додатків на андроїд. Також, була створена бізнес логіка додатку. Для зберігання даних використовувалась база даних Room. Розробка проводилась мовою програмування Java. Беручи до уваги усі деталі, що були описані раніше, додаток є клієнт-орієнтованим та його дизайн та функціонал було розроблено таким чином, щоб він мав змогу допомогти користувачу вирішити проблему з імпульсивними діями.

Використана література

1. Грегорі Дж. Медден, доктор філософії, та Уоррен К. Бікель, доктор філософії. Імпульсивність: поведінкова та неврологічна наука про дисконтування, 2010. -453 с.

2. Як навчитися контролювати імпульсивні покупки: п'ять простих способів - https://knife.media/impulse-purchase/

3. Вебстер, К. Д., Джексон, М. А. Імпульсивність: теорія, оцінка та лікування, 1997. -462 с.

4. Як стимулювати клієнта здійснювати імпульсивні покупки: ефективні маркетингові прийоми - https://www.insales.com.ua/blogs/blog/impulsivnye-pokupki

5. https://www.unian.net/economics/telecom/10500207-smartfony-est-u-55-ukraincev-sredi-molodezhi-92-infografika.html

6. 9 способів протистояння імпульсивної поведінки, або як зберегти самоконтроль? - https://psy-zoom.ru/2018/01/01/9impulse.html

7. Material Design. Introduction - https://material.io/design/introduction

8. Wikipedia. Google play - https://en.wikipedia.org/wiki/Google_Play

9. Material Design. The Color system. Система Кольорів - https://material.io/design/color/the-color-system.html

10. Figma. Get Started . - https://help.figma.com/hc/en-us/categories/360002051613-Getting-Started#Get-Started

11. ІМПУЛЬСИВНІ ДІЇ - https://leksika.com.ua/16520205/legal/impulsivni_diyi.

12. Мимовільні й довільні дії - https://pidru4niki.com/12300824/psihologiya/mimovilni_dovilni_diyi

13. Зображення ієрархії компонентів відображення https://developer.android.com/images/viewgroup_2x.png