Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Курсова робота

освітній ступінь – бакалавр

на тему: «**Розробка рекомендаційного мобільного додатку під Android для вибору гри**»

Виконала: студентка 3-го року навчання

Спеціальності

121 «Інженерія програмного забезпечення»

Матвієнко Ірина Валентинівна

Керівник Вовк Наталя Євгенівна,

старший викладач

«____» _____ 20____ р.

Київ - 2022

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Освітній ступінь бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма бакалавр

ЗАТВЕРДЖУЮ

Завідувач кафедри мультимедійних систем

доцент О. П. Жежерун

“____” _____20___року

ЗАВДАННЯ

ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Матвієнко Ірині Валентинівні

1. Тема роботи «**Розробка рекомендаційного мобільного додатку під Android для вибору гри**», керівник роботи Вовк Наталя Євгенівна, старший викладач
2. Строк подання студентом роботи: 6 червня 2022
3. План роботи

    Анотація

    Вступ

    Розділ 1. Аналіз та підготовка до написання додатку

    Розділ 2. Функціонал, UI, алгоритм рекомендацій

    Розділ 3. Аналіз створеного додатку

    Висновки

    Список використаних джерел

ГРАФІК ПІДГОТОВКИ КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	15 жовтня 2021			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	15 жовтня – 15 листопада 2021			
3.	Складання плану кваліф. роботи та узгодження з науковим керівником	15 листопада 2021			
4.	Написання розділів роботи	15 листопада 2021 – 15 березня 2022			
5.	Проміжний контроль виконання роботи	1 березня 2022			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	1 січня – 1 травня 2022			
	Розділ 1 (постановка проблеми, теоретичні основи)	1 лютого 2022			
	Розділ 2 (аналітично-дослідницька частина)	1 березня 2022			
	Розділ 3 (проектно-рекомендаційна частина)	1 квітня 2022			
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	1 травня – 5 червня 2022			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності	6 червня 2022			
9.	Здача курсової роботи	16 червня 2022			

Графік узгоджено 15 жовтня 2021 р.

Науковий керівник Вовк Наталя Євгенівна

Виконавець курсової роботи Матвієнко Ірина Валентинівна

Зміст

Анотація

Дана робота присвячена любителям настільних ігор, адже вона скоротить час тяжких та суперечливих роздумів, яку би гру обрати та надасть більше часу для самого процесу гри з друзями. У результаті роботі розроблено Android застосунок для рекомендації та систематизації настільних ігор. Описано функціонал та структуру застосунку, алгоритм рекомендацій, код, способи збереження та структуру даних.

Перелік термінів та умовних позначень

Активіті (англ. – activity) – Java-клас з графічним представленням в файлі XML; забезпечує вікно (один екран), в якому додаток відмальовує свій UI [1].

Фрагмент (англ. – fragment) – Java-клас з графічним представленням в файлі XML; репрезентує частину UI додатку, призначену для повторного використання. Визначає своє графічне представлення та керує ним, має свій життєвий цикл і може обробляти свої події введення [2].

Інтент (англ. – intent) – структура даних, яка містить в собі абстрактний опис дії, яка має бути виконана; найважливіше його призначення – запуск активіті за допомогою методу startActivity() [3].

Шерд преференсес (англ. – shared preferences) – інтерфейс для доступу та редагування вибраних даних, які є спільними для всього додатку [4].

Вступ

Мені з дитинства дуже подобалося грати в настільні ігри, яких в мене вдома накопичилося вже доволі багато. Достатньо для того, щоб забувати про деякі, коли приходить час вибору гри для проведення часу в компанії. Також я люблю упорядковувати різні однорідні речі та пункти інформації. Виявивши у себе ці дві риси, я вирішила об'єднати їх в ідею для моєї курсової роботи і написати Android додаток для рекомендацій та упорядкування моїх настільних ігор. Над такою темою мені захочеться працювати і потім я з радістю буду користуватися своїм додатком.

Гра в настільні ігри – дуже розповсюджений в світі, цікавий і розвиваючий спосіб проведення вільного часу, який стає все більш популярним і в Україні. Збільшується кількість людей, які захоплюються настільними іграми і колекціонують їх. А коли ігри вже не вміщаються на дві полички в шафі, з'являється така проблема, як довгі роздуми і вагання, в яку б гру пограти із родиною, а в яку – із компанією друзів. Про деякі ігри вже не перший раз забуваєш, навіть щоб просто запропонувати їх друзям на вибір, і ці ігри припадають пилом десь на антресолях, а від деяких вже давно загубилися правила. А іноді просто ліньки робити вибір, бо ігор так багато і майже всі улюблені. Окрім цього, коли вперше зіграв у дуже цікаву нову гру, так і хочеться розповісти скоріше людям, які поділяють твоє захоплення, про це та похвалитися своїм придбанням.

На ринку не існує комплексного рішення всіх цих проблем. Люди зберігають дані про свої настільні ігри в таблицях Excel, а деякі навіть використовують для цього Word. Тому створення зручного та багатофункціонального додатку, де можна не тільки зберігати інформацію про ігри, а й отримувати рекомендації про них, є гарною ідеєю.

Об'єктом дослідження є розробка архітектури додатку під операційну систему Android із різноманітними активіті, фрагментами, меню, пошуком та фільтрами. Метою даної роботи є розробка Android додатку для

систематизації настільних ігор, отримання рекомендацій для визначення, у що б пограти, та надсилання рекомендацій настільних ігор іншим користувачам.

Розділ 1. Аналіз та підготовка до написання додатку

1.1. Аналіз існуючих рішень

Повноцінних аналогів створюваного додатку знайдено не було. Можна стверджувати, що це рішення є унікальним. Однак, звичайно, існують аналоги окремих частин функціоналу додатку, які послугували натхненням при роботі. По-перше, ідеї для зовнішнього вигляду списку пунктів, що прокручується, активіті перегляду гри, способу переходу до списку категорій та їх редагування було взято з додатку для планування «Список завдань», де схожим чином організована робота з планами та категоріями планів. По-друге, фільтри та сортування якихось пунктів у вигляді, як у створюваному додатку, наявні в багатьох додатках та веб-сайтах, хоча б в тому ж інтернет-магазині Rozetka. По-третє, існують інтернет-магазини настільних ігор, з яких в тому числі було взято назви категорій ігор за замовчуванням. Також розташування картинки та тексту на пункті гри було скомбіновано з пункту страви в додатку McDonald’s та пункту новини в додатку «Київ Цифровий».

1.2. Обґрунтування вибору інструментів для розробки

Мобільні додатки під операційну систему Android можна писати за допомогою таких засобів та мов програмування: Java + XML, Kotlin + XML, C++ + Android Native Development Kit, C# + Xamarin, Python + Kivy, HTML + CSS + JavaScript та інших. Для створення додатку було обрано мову програмування Java та мову розмітки XML, тому що це найбільш розповсюджене поєднання засобів для написання Android додатків, більшість додатків в Google Play написано на Java і ця мова ще довго не втрачатиме своєї популярності [5]. Окрім цього, Java було обрано через достатній досвід програмування на цій мові (включно зі створенням Android додатків) та велику кількість доступних навчальних матеріалів в Інтернеті.

Є багато середовищ розробки програмного забезпечення, які надають можливість писати додатки під Android на Java, як-от: Android Studio, Eclipse,

IntelligJ IDEA, Net Beans, Komodo та інші. Для роботи над додатком обрано Android Studio, як варіант, який обирається найчастіше. Це середовище розробки рекомендоване Google, надає зручний інтерфейс для роботи, включно з можливістю переглядати графічні представлення під час їх створення, дозволяє автоматично генерувати активіті та фрагменти за різноманітними шаблонами, має потужний емулятор та надає багато інших корисних можливостей.

1.3. Цільова аудиторія

Додаток розраховується на два типи користувачів:

1) Люди будь-якого віку старше 5 років з будь-якого населеного пункту України незалежно від рівня забезпеченості. У них є пристрій з операційною системою Android та можливість встановити додаток із Google Play. Найголовніше, це мають бути любителі, в яких вдома багато настільних ігор та є велика різноманітна компанія для гри. Ці користувачі можуть систематизувати свої настільні ігри, користуватися додатком для вибору у що б пограти з друзями та родичами та рекомендувати таким самим любителям, як вони самі, свої улюблені настільні ігри, щоб вони також могли їх собі придбати.

2) Працівники та відвідувачі антикафе та клубів настільних ігор. Працівники можуть пропонувати планшет з встановленим додатком відвідувачам для вибору у що б пограти. Там відвідувачі зможуть побачити, як оцінили гру попередні відвідувачі та оцінити її для наступників, виставивши грі бали. Також, погравши в якусь гру і уподобавши її, відвідувач зможе за допомогою додатку порекомендувати гру другу або собі (якщо в нього також є встановлений додаток), щоб не забути її назву та згодом купити. Окрім цього, кількість обирань та оцінка ігор відвідувачами допоможуть працівникам цих закладів отримувати статистику, які ігри подобаються відвідувачам.

На основі цих даних вони зможуть робити висновки, які ігри, або доповнення до них, варто докупити.

Розділ 2. Функціонал, UI, алгоритм рекомендацій

2.1. Назва, кольорова тема, логотип

Виникало декілька ідей, як назвати додаток: «MyRecommendations», «BoardGamesRecommendations», «MyBoardGames», «BoardGames», «WhatToPlay», «BoardGamesStore». Деякі назви виявилися занадто довгими, деякі описують лише частину функціоналу, а з інших взагалі не зрозуміло, яка суть додатку. Після довгих роздумів, назвою додатку було обрано «MyBoardGames» (Мої настільні ігри). По-перше, ця назва прийнятна по довжині. По-друге, вона описує те, що додаток про настільні ігри – про настільні ігри користувача, з інформацією про які він може взаємодіяти. По-третє, в назві є трохи вихваляння, і це розкриває суть рекомендацій ігор, які в тебе є, друзям.

Першим основним кольором кольорової теми додатку обрано бузковий з номером #C171CF. Також використовується трохи світліша його версія з номером #D2A8DA. Другим основним кольором обрано яскравий жовто-помаранчевий з номером #FFC107. Давно хотілося написати додаток в улюбленому поєднанні кольорів, і це вдалося реалізувати в додатку для курсової роботи.

Логотип додатку – це два гральні кубики основного кольору на світло-жовто-помаранчевому фоні, як зображено на рис. 2.1. На логотипі зображено кубики, тому що це один з головних символів настільних ігор та вони є майже в кожній настільній грі. Також ці кубики, але жовто-помаранчевого кольору, це зображення, яке в додатку встановлюється за замовчуванням при додаванні нової настільної гри.

Рис. 2.1 Логотип додатку

2.2. Функціонал додатку

В додатку реалізовано такий функціонал: перегляд, додавання, редагування, видалення категорій ігор; перегляд, додавання, редагування (з можливістю додати зображення), видалення, фільтрація, пошук, сортування настільних ігор, отримання рекомендацій у що б пограти, обрання ігор для того, щоб в них пограти, визначення гри улюбленою (натисканням на сердечко), виставлення іграм балів-зірочок. Про ігри зберігається така інформація: назва, опис, фото, правила, місцезнаходження, мінімальний та максимальний вік гравців (максимальний на випадок якщо гра зовсім дитяча), мінімальна та максимальна кількість гравців, приблизний час гри, категорії, кількість балів-зірочок по п’ятибальній шкалі, чи є улюбленою, дата додавання, кількість разів обрання, дата останнього обрання гри. Ігри можна фільтрувати за віком, кількістю гравців, часом гри, категоріями, балами та улюбленістю. Сортувати можна за назвою, датою останнього обрання та кількістю балів – в прямому та зворотному порядку. Додаток може працювати як із підключенням до Інтернету, так і без нього. Всі вище зазначені функції доступні без доступу до Інтернету. При першому після завантаження вході в додаток користувачу автоматично генерується його унікальний id. Доступ до Інтернету потрібен для створення/зміни псевдоніму користувача, для того, щоб хвалитися перед друзями своїми іграми (надсилати їм рекомендації про свої ігри) та для того, щоб переглядати рекомендації ігор від друзів. Рекомендувати другу гру можна ввівши в спеціальне поле його id або унікальний псевдонім.

2.3. Навігація по додатку

При вході в додаток користувач потрапляє на головну вкладку Ігри зі списком своїх настільних ігор, які можна фільтрувати, шукати, сортувати та по яким отримувати рекомендації, у щоб б пограти (рис. 2.2).

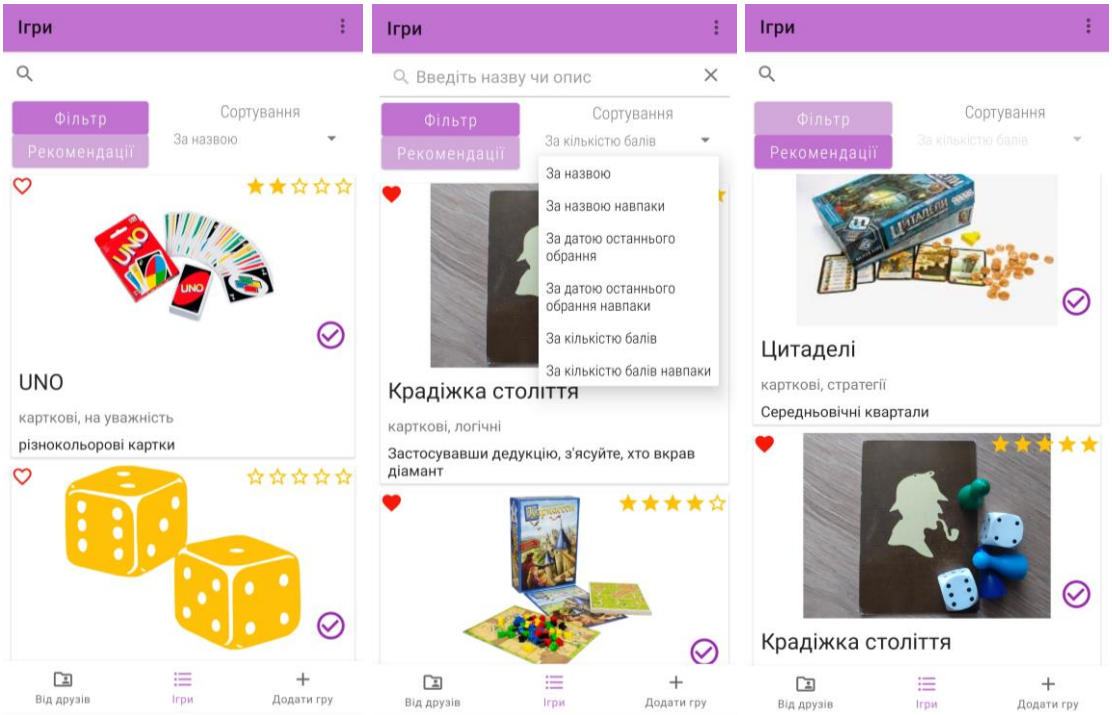

Рис. 2.2 Головна вкладка

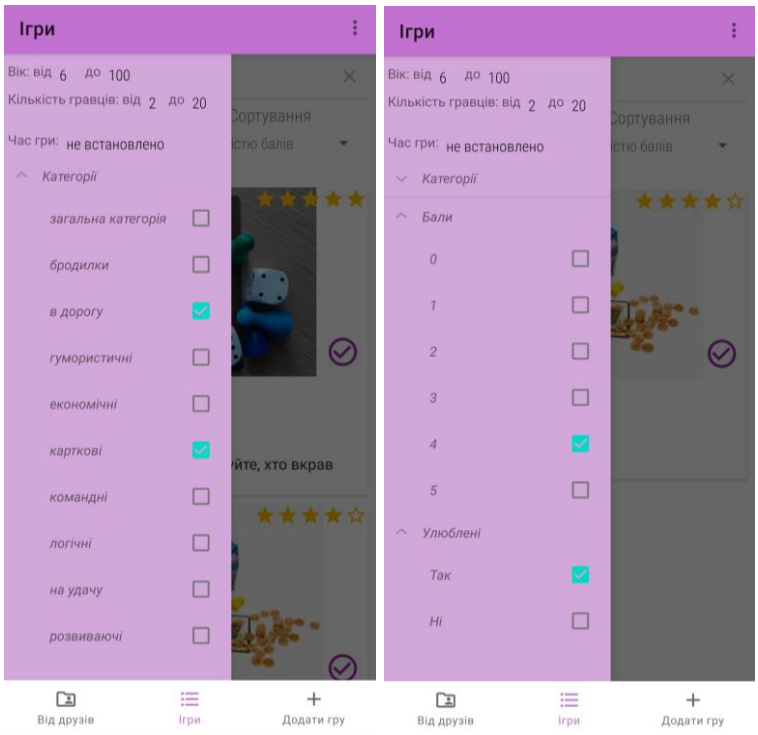

Рис. 2.3 Фільтри

При натисканні на кнопки Фільтр або Рекомендації відкривається бокова панель, де можна встановити фільтри, які застосовуються одразу. Фільтри потрібні для звичайної фільтрації або отримання рекомендацій, залежно від поточного режиму (рис. 2.3). Рекомендації відрізняються від фільтрації тим, що встановлені фільтри впливають на результат не так строго, а за певним алгоритмом, описаним у підрозділі 2.4.

При кліку на пункт гри відкривається вікно з детальною інформацією про гру. Знизу є кнопки для оновлення інформації та видалення гри, а праворуч від дати останнього обрання – кнопка для рекомендування гри іншому користувачу (рис. 2.4).

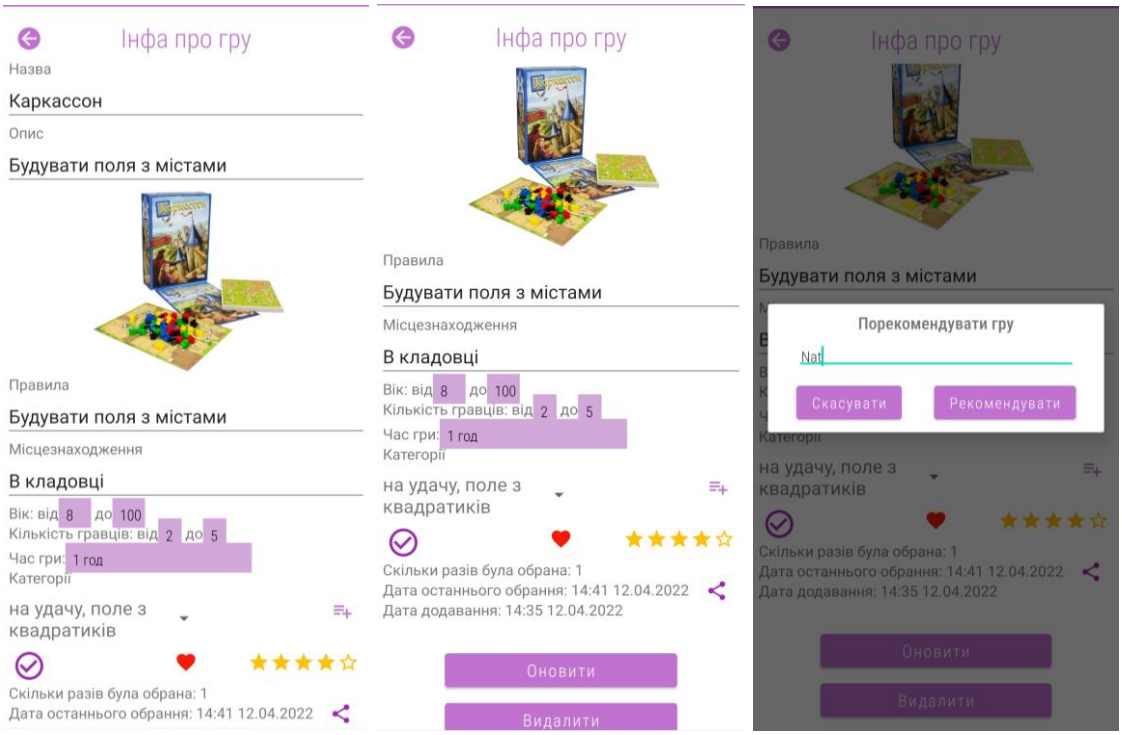

Рис. 2.4 Вікно редагування гри

Праворуч від головної вкладки розташована вкладка Додати гру, призначена для додавання нової гри за допомогою кнопки знизу. При кліку на зображення за замовчуванням з кубиками відкривається діалогове вікно для вибору іншого додатку, за допомогою якого можна обрати або створити зображення для гри. При кліку на кнопку праворуч від списку категорій відкривається діалогове вікно для додавання нової категорії (рис. 2.5).

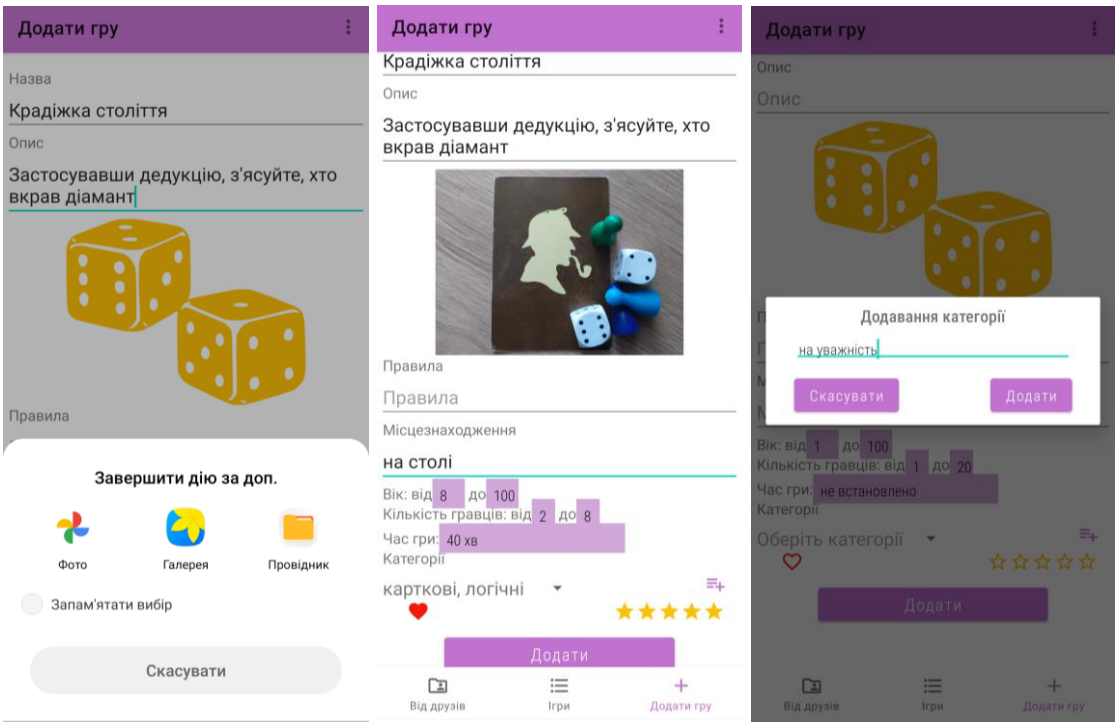

Рис. 2.5 Вкладка додавання гри

Ліворуч від головної вкладки є вкладка Від друзів, де можна переглянути ігри, які порекомендували тобі друзі. При кліку на пункт гри відкривається вікно з детальною інформацією про гру (рис. 2.6).

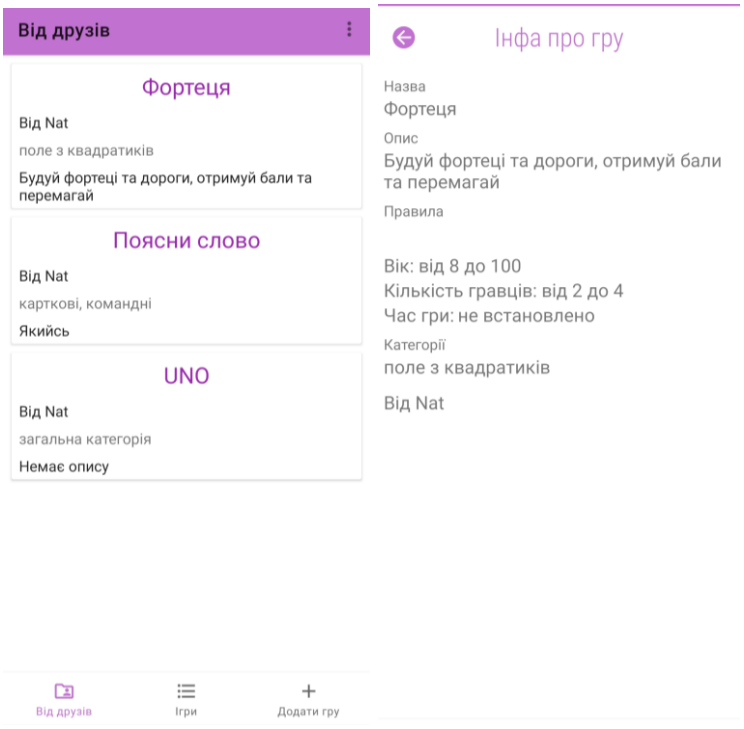

Рис. 2.6 Рекомендації від друзів

Зверху праворуч в додатку є три крапки, які відкривають меню з двома пунктами – Категорії та Профіль (рис. 2.7).

Рис. 2.7 Верхнє меню

При кліку на перший пункт меню відкривається вікно зі списком категорій, біля кожної з яких є кнопки для редагування та видалення (рис. 2.8).

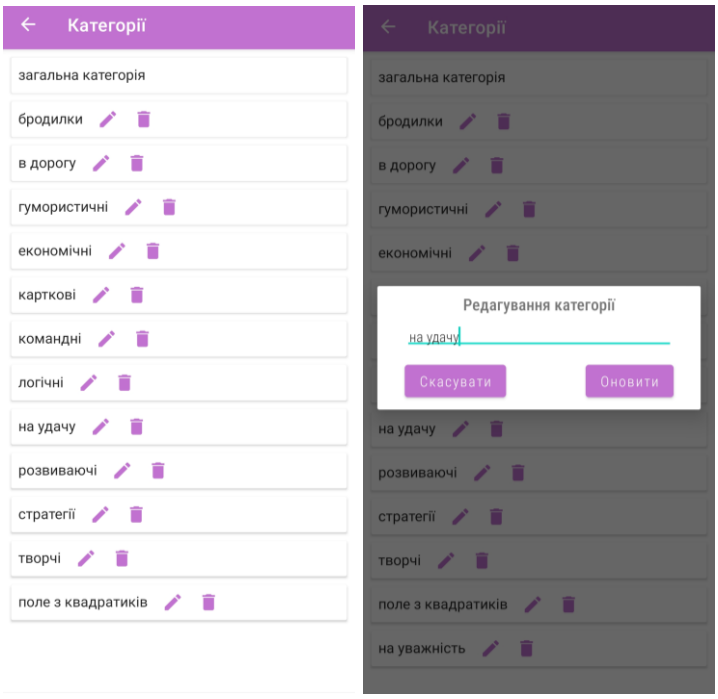

Рис. 2.8 Категорії

При кліку на другий пункт меню відкривається вікно з профілем користувача, де можна скопіювати свій id та змінити псевдонім (рис. 2.9).

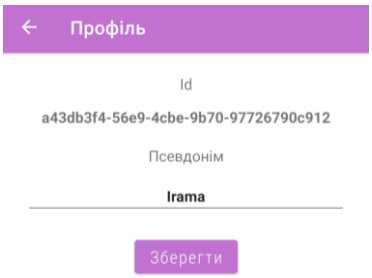

Рис. 2.9 Профіль користувача

2.4. Алгоритм рекомендацій

В режимі рекомендацій для кожної гри обраховується рейтинговий бал, заснований на її характеристиках та виставлених фільтрах. Початково гра має 0 балів. Якщо в фільтрах обрано категорії, гра отримує +7 балів за кожну з обраних категорій, якій вона належить. Якщо в фільтрах обрано бали: якщо гра має точно таку кількість балів, яка є в обраних значеннях фільтру, гра отримує +10 балів; якщо гра має на 1 бал менше чи більше, ніж якесь обране значення з фільтру, гра отримує +5 балів. Якщо обрано фільтрацію за тим, чи є гра улюбленою, і ця характеристика гри відповідає обраному значенню з фільтру, гра отримує +15 балів. Якщо вік гравців є між меншим та більшим встановленими значеннями в фільтрах, гра отримує +10 балів; інакше якщо мінімальний вік гравців менший за встановлений на 1, гра отримує +8 балів, менший на 2 – гра отримує +6 балів, менший на 3 – гра отримує +4 бали; інакше якщо максимальний вік більше за встановлений на 1 чи 2, гра отримує +8 балів, більший на 3 чи 4 – гра отримує +6 балів, а інакше гра отримує +4

бали. Якщо кількість гравців є між меншим та більшим встановленими значеннями в фільтрах, гра отримує +10 балів; інакше якщо мінімальна кількість гравців менша за встановлену на 1, гра отримує +5 балів, менша на 2 – гра отримує +2 бали; інакше якщо максимальна кількість гравців більша за встановлену на 1, гра отримує +7 балів, більша на 2 – гра отримує +5 балів. Час гри в фільтрах впливає на рейтинг, якщо його значення не дорівнює «не встановлено». Якщо час гри «не встановлено» або він дорівнює встановленому значенню, гра отримує +10 балів; якщо час гри відрізняється на 1 позицію від встановленого, гра отримує +7 балів; якщо відрізняється на дві позиції, гра отримує +5 балів; інакше бали за час гри не додаються.

Ігри сортуються за рейтинговими балами в порядку спадання і користувачу показуються перші 6 ігор – від найбільш рекомендованої з найкращих для того, щоб пограти, до найменш рекомендованої з них.

Розділ 3. Аналіз створеного додатку

3.1. Опис БД

Відносно схеми бази даних додаток можна поділити на дві частини за місцем зберігання даних: в JSON файлах, щоб надавати доступ в будь-який час (рис. 3.1), і в Firebase, щоб забезпечити зв’язок між користувачами (рис. 3.2).

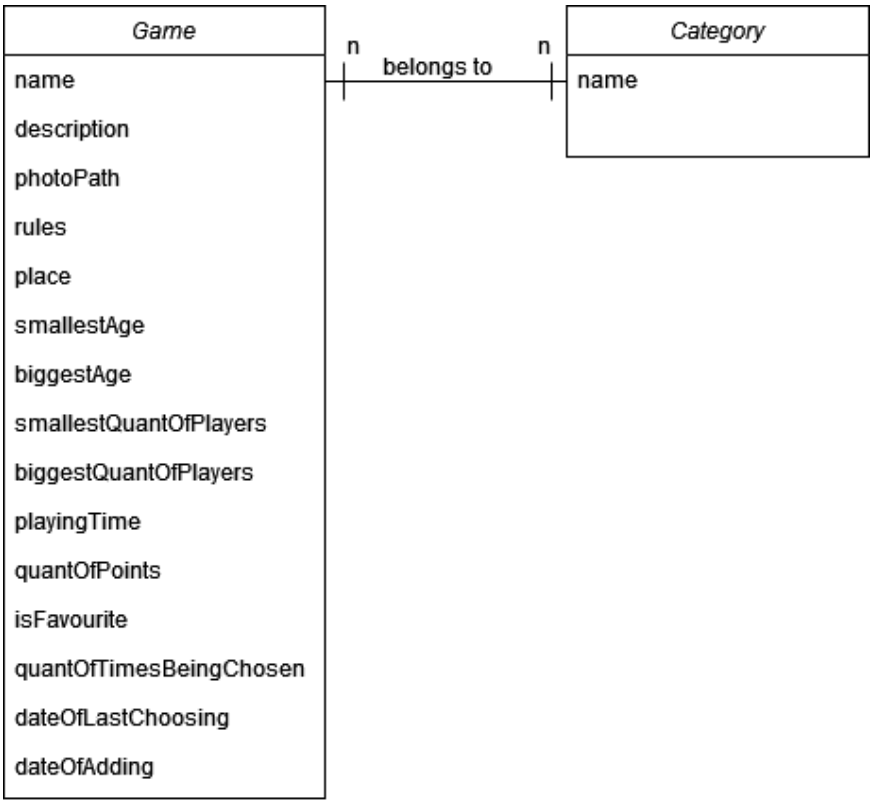

Рис. 3.1 ER-модель частини даних, що зберігається офлайн

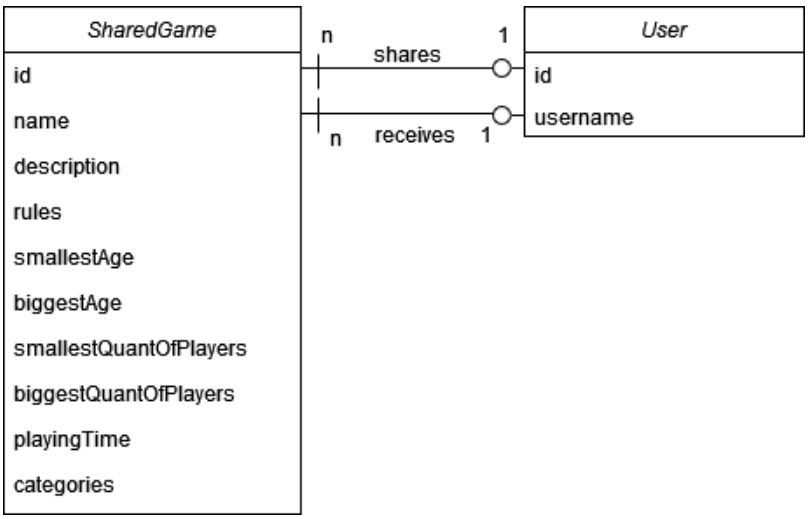

Рис. 3.2 ER-модель частини даних, що зберігається онлайн

У сутності гра Game є такі поля: назва, опис, фото, правила, місцезнаходження, мінімальний та максимальний вік гравців, мінімальна та максимальна кількість гравців, приблизний час гри, кількість балів, чи є улюбленою, дата додавання, кількість разів обрання, дата останнього обрання гри. Гра належить декільком категоріям Category (мінімум одній). Із одного екземпляра сутності Game створюється багато екземплярів сутності SharedGame. Сутність рекомендована гра SharedGame має унікальний id та позичає з сутності Game тільки ті поля, які будуть важливими для користувача, якому її рекомендують: назва, опис, правила, мінімальний та максимальний вік гравців, мінімальна та максимальна кількість гравців, приблизний час гри. Рекомендовану гру рекомендує один користувач одному іншому користувачеві. Сутність користувач User містить поля id та псевдонім.

Необхідність відділити сутність SharedGame від Game викликана декількома чинниками. По-перше, екземляр SharedGame створюється кожен раз, коли користувач хоче порекомендувати гру. Відповідно, додається унікальний ідентифікатор, id користувача, який рекомендує, та id користувача, якому рекомендують. Також прибирається некорисна для отримувача інформація. По-друге, користувач може редагувати та видаляти свої ігри, а редагування чи видалення рекомендацій не є бажаним. Тим паче, якби була лише одна сутність Game, дані про неї довелося б оновлювати одразу в двох сховищах даних і з’явилася б додаткова небажана залежність від підключення до Інтернету.

3.2. Опис структури проєкту

Проєкт містить: папку myboardgames із файлами класів Java, зображену на рис. 3.3, папку res з файлами ресурсів xml, AndroidManifest та файли налаштувань Android проєкту.

Файли xml виконують такі функції: графічне представлення активіті та фрагментів, навігація між активіті та фрагментами, векторні зображення для кнопок та логотипу, кольори, теми та стрічки для текстових полів.

В AndroidManifest встановлюються назва, логотип та тема додатку та декларуються всі активіті (із вказуванням, які активіті є батьківськими для інших).

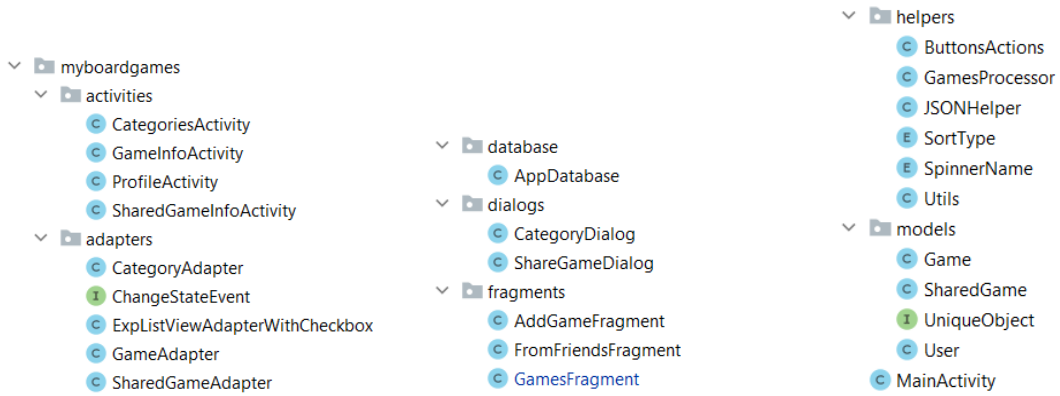

Рис. 3.3 Структура папки з Java-кодом myboardgames

3.3. Опис використаних бібліотек

Підключено бібліотеку DateTimeUtils для дати і часу. Використано її для зручного порівняння двох об’єктів дат для сортування ігор за датою додавання [6].

Підключено бібліотеку gson для серіалізації ігор та категорій в JSON формат та десеріалізації з нього.

Підключено бібліотеку firebase:firebase-database для зберігання даних в нереляційній базі даних Realtime Database Firebase [7].

3.4. Опис Java-класів

3.4.1. MainActivity

Клас MainActivity – клас, що наслідується від AppCompatActivity; це основне активіті додатку. В методі onCreate() встановлюється графічне представлення для активіті, налаштовуються пункти нижнього меню та

підключається верхнє приховане меню. Метод onCreateOptionsMenu() встановлює графічне представлення для верхнього меню, в якому є 2 пункти: Категорії та Профіль. В методі onOptionsItemSelected() при обранні пункту верхнього меню відбувається перенаправлення на відповідне до цього пункту активіті за допомогою інтентів. Метод onResume() викликається, коли активіті стає видимим користувачу. Він надає користувачу унікальний id при першому вході в додаток, зберігає цей id в шерд преференсес та додає нового користувача з цим id та порожнім псевдонімом в Firebase. Метод onActivityResult() опрацьовує результат вибору зображення для нової гри за допомогою іншого додатку (Галереї, Камери тощо) та встановлює це зображення для перегляду у фрагмент додавання нової гри [8].

3.4.2. Пакет models

Клас Game реалізує сутність настільної гри користувача. Клас містить всі необхідні поля, зазначені на ER-моделі на рис. 3.1 (включно зі списком List<String> категорій), їх селектори та модифікатори, порожній конструктор, конструктор з усіма полями, метод toString() для стрічкового представлення, метод equals() для порівняння об’єктів та метод getCategoriesToString(), який повертає стрічкове представлення списку категорій.

Інтерфейс UniqueObject є допоміжним для роботи з базою даних та має сигнатури методів getId() та whichGroup().

Клас SharedGame імплементує інтерфейс UniqueObject та є реалізацією сутності рекомендована гра. Клас містить всі необхідні поля, зазначені на ER-моделі на рис. 3.2 (тут категорії – це стрічка для зручності роботи, бо вони фіксуються в певний момент часу із екземпляра класу Game та не можуть бути відредагованими), їх селектори та модифікатори, порожній конструктор, конструктор з усіма полями, конструктор, куди передається екземпляр класу Game, id користувача, який рекомендує гру та id користувача, якому рекомендують гру. Метод з інтерфейсу UniqueObject whichGroup() повертає назву класу – «SharedGame».

Клас User імплементує інтерфейс UniqueObject та є реалізацією сутності користувач. Клас містить всі поля id та username, їх селектори та модифікатори, порожній конструктор, конструктор з усіма полями, метод toString() для стрічкового представлення. Метод з інтерфейсу UniqueObject whichGroup() повертає назву класу – «User».

3.4.3. Клас AppDatabase в пакеті database

Клас AppDatabase служить для взаємодії з базою даних Realtime database з Firebase. Клас містить об’єкт доступу до бази даних database та його селектор, посилання на певну колекцію в базі даних databaseReference та конструктор без параметрів з ініціалізацією поля database. Метод addSharedGameToDatabase() відповідає за додавання рекомендованої гри в базу даних, а метод addUserToDatabase() – за додавання туди користувача. Метод updateObjectInDatabase() потрібен для оновлення об’єктів в базі даних, в додатку використовується для оновлення псевдоніму користувача. Методи usernameExists() та userIdExists() перевіряють, чи існує вже в базі даних, відповідно, такий псевдонім чи id користувача. Метод getReferenceToGroup() повертає посилання на певну колекцію в базі даних.

3.4.4. Пакет helpers

Клас JSONHelper [9] – це клас зі статичними методами для збереження ігор та категорій в файлах в форматі JSON та відновлення їх звідти. Клас містить дві константи – назви файлів та два статичних внутрішніх класи – GameItems та CategoryItems. GameItems містить список ігор, селектор і модифікатор, а CategoryItems – список категорій, селектор і модифікатор. JSONHelper також містить декілька важливих методів. Метод inFinaly() забезпечує закриття потоків після закінчення роботи з файлом. В методі exportGamesToJSON() відбувається серіалізація списку ігор в JSON файл за допомогою внутрішнього класу GameItems, файлового вихідного потоку та бібліотеки gson. В методі exportCategoriesToJSON() відбувається серіалізація списку категорій аналогічним чином. В методі importGamesFromJSON() відбувається

десеріалізація списку ігор з JSON файлу за допомогою потоку читання, файлового вхідного потоку, внутрішнього класу GameItems та бібліотеки gson. В методі importCategoriesFromJSON() відбувається десеріалізація списку категорій з JSON файлу аналогічним чином.

Перелік (англ. – enum) SortType – це перелік 6-ти типів сортування ігор: за назвою, за датою останнього обрання, за кількістю балів – в прямому та зворотному порядку.

Перелік SpinnerName – це перелік 5-ти типів фільтрів ігор, значення яких встановлюються за допомогою випадаючих списків: найменший вік гравців, найбільший вік гравців, найменша кількість гравців, найбільша кількість гравців, приблизний час гри.

Клас GamesProcessor – це клас-сервіс зі статичними методами для роботи з колекцією ігор games та колекцією категорій categories. Клас містить статичний список екземплярів класу Game та статичний список стрічок-категорій, які завдяки селекторам доступні з будь-якої частини додатку. Є два методи перевірок, що ці списки були ініціалізовані: gamesAreLoaded() та categoriesAreLoaded(). У методі loadGames() відбувається ініціалізація списку games за допомогою методу importGamesFromJSON() класу JSONHelper. У методі loadCategories() відбувається ініціалізація списку categories за допомогою методу importCategoriesFromJSON() класу JSONHelper. Якщо із JSON файлу отримано порожній список (тобто користувач ще не додав жодної категорії), categories ініціалізується списком категорій за замовчуванням для зручності користування додатком. Методи saveGames() та saveCategories() зберігають оновлені списки, відповідно, ігор та категорій (за допомогою методів класу JSONHelper) та показують користувачу короткі повідомлення про успіх чи невдачу цих дій. В методі sortGames(), відповідно до переданого параметру – екземпляру класу SortType, відбувається сортування списку ігор за певним критерієм за допомогою екземплярів класу Comparator та різних методів compare(). Метод getGameByName() повертає

гру, якщо гра з переданим параметром name існує в списку games; інакше повертає null. Метод gameAlreadyExists() перевіряє, чи гра з переданим параметром name вже існує в списку ігор (перевірка відбувається нечутливо до регістру). Метод categoryNameAlreadyExists() перевіряє, чи така сама категорія, як переданий параметр categoryName вже існує в списку категорій (перевірка також відбувається нечутливо до регістру). Метод getCopyOfGames() повертає копію списку games. Метод addCategory() додає до списку категорій нову категорію, а метод addGame() додає до списку ігор нову гру. Метод chooseGame() забезпечує збільшення кількості обирань гри та встановлює поточні дату та час в якості дати останнього обрання гри. Метод deleteGame() видаляє гру зі списку ігор. Метод deleteCategory() видаляє категорію зі списку категорій та з усіх ігор, які належать до цієї категорії; якщо в якоїсь гри не залишається категорій, до неї додається категорія за замовчуванням «загальна категорія». Метод editCategory() відповідає за редагування категорії.

Клас ButtonsActions – це клас зі статичними методами для подій, які відбуваються при натиску на кнопки. Методи подій винесені в окремий клас, бо використовуються більше, ніж в одному активіті чи фрагменті (наприклад, виставлення грі балів при натиску на зірочки присутнє в активіті редагування гри та в пункті гри в списку ігор). У методі makeFavorite() кнопці встановлюється зображення зафарбованого сердечка (означає, що гра стала улюбленою), а в методі makeNotFavorite() – зображення контуру сердечка (означає, що гра перестала бути улюбленою). У методі favoriteAction() кнопці встановлюється початкове зображення та додається слухач подій, який забезпечує зміну зображення на кнопці за допомогою вище описаних методів та змінює значення, чи є гра улюбленою. У методі chooseAction() кнопці додається слухач подій, який викликає метод chooseGame() класу GameProcessor. У методі addStar() кнопці встановлюється зображення зафарбованої зірочки (означає, що кількість балів для гри збільшилась на 1), а в методі removeStar() – зображення контуру зірочки (означає, що кількість

балів для гри зменшилась на 1). Методи processSmallerQuantOfPoints(), processTheSameQuantOfPoints(), processBiggerQuantOfPoints() відповідають за графічне встановлення потрібної кількості зірочок з п’яти за допомогою двох попередніх методів. У методі pointsAction() відбувається встановлення кнопкам-зірочкам початкових зображень та слухачів подій, які забезпечують зміну кількості балів для гри, за допомогою 5-и попередніх методів. Метод setDefaultStars() встановлює стан зірочок за замовчуванням (всі контури). Метод hideKeyboard() відповідає за приховання клавіатури. В методі setCategoriesListener() відбувається додавання слухача подій до текстового поля категорій гри. При натиску на це текстове поле з’являється діалогове вікно для вибору однієї або декількох категорій, до яких належить гра. Якщо при закритті діалогового вікна не обрано жодної категорії, грі встановлюється категорія за замовчуванням – «загальна категорія».

Клас Utils – клас зі статичними допоміжними методами-утилітами, які використовуються в інших класах. Метод getCurrentDate() повертає поточну дату. Метод convertToLocalDateViaInstant() слугує для конвертації екземпляра класу Date в екземпляр класу LocalDateTime. Метод convertDateToLocalString() повертає стрічкове представлення дати та часу в звичному для українців форматі та з урахуванням часового поясу. Метод getPositionOfStr() повертає позицію стрічки в масиві, або -1, якщо її там немає. Метод validateName() перевіряє, чи передане ім’я непорожнє. Метод isNetworkAvailable() використовується для перевірки, чи доступна зараз на пристрої мережа Інтернет.

3.4.5. Пакет dialogs

Клас CategoryDialog – клас, що наслідується від Dialog, та реалізує діалогове вікно для додавання або редагування категорії. В конструктор передається заголовок діалогового вікна, підказка для текстового поля для нової назви категорії, напис для правої кнопки та назва категорії (потрібна в випадку редагування), бо ці стрічки мають бути різні для додавання та редагування

категорії; також передається слухач подій для правої кнопки. В конструкторі відбувається встановлення переданих стрічок на відповідні елементи графічного представлення, встановлюються слухачі подій на ліву кнопку скасування та праву кнопку додавання/редагування. Метод getCategoryName() повертає введену назву категорії без зайвих пробілів.

Клас ShareGameDialog – клас, що наслідується від Dialog, та реалізує діалогове вікно для рекомендування іншому користувачу гри. В конструктор передається слухач подій для кнопки «Рекомендувати». В конструкторі відбувається встановлення слухачів подій на кнопку скасування та кнопку рекомендування. Крім цих двох кнопок, в діалоговому вікні є текстове поле, в яке потрібно вводити id або псевдонім друга, якому хочеш порекомендувати гру. Метод getFriendName() повертає введену інформацію про користувача без зайвих пробілів.

3.4.6. Пакет adapters

Інтерфейс ChangeStateEvent є допоміжним для класу ExpListViewAdapterWithCheckbox та має сигнатуру методу callback().

Клас ExpListViewAdapterWithCheckbox – клас, що наслідується від BaseExpandableListAdapter та реалізує адаптер для фільтрів – списків, що розгортаються і дозволяють галочками обирати декілька значень (відмітити декілька чекбоксів). В класі є два внутрішніх final класи: GroupViewHolder з текстовим полем – назвою групи чекбоксів та ChildViewHolder з екземпляром класу CheckBox та текстовим полем – назвою чекбоксу. В конструкторі ініціалізується список назв груп фільтрів mListDataGroup, мапа назв та значень чекбоксів mListDataChild, контекст і мапа позицій в фільтрі та обраних значень mChildCheckStates. Перевизначений метод getGroupCount() повертає розмір списку mListDataGroup. Перевизначений метод getChildrenCount() повертає розмір певної групи чекбоксів. Перевизначений метод getGroup() повертає назву групи чекбоксів на певній позиції. Перевизначений метод getChild() повертає назву чекбокса певної

групи на певній позиції. Перевизначений метод getGroupId() повертає позицію групи чекбоксів. Перевизначений метод getChild() повертає позицію чекбокса в його групі. Перевизначені методи hasStableIds() та isChildSelectable() повертають false. В перевизначеному методі getGroupView() з використанням екземпляру класу GroupViewHolder групі чекбоксів встановлюється графічне представлення, яке ж і повертається. В перевизначеному методі getChildView(), з використанням екземпляру класу ChildViewHolder, чекбоксу встановлюється графічне представлення, яке ж і повертається; також, якщо раніше було збережено стан чекбоксу (обраний чи не обраний), цей стан відновлюється; чекбоксу встановлюється слухач подій. Метод setOnChangeStateListener() потрібен для встановлення слухача подій на будь-яку зміну стану хоч одного з фільтрів. В методі doEventCallback() відбувається виклик методу callback() у екземплярів класу ChangeStateEvent зі списку observers. Метод getCheckedItems() повертає мапу з назвами груп фільтрів на назвами обраних чекбоксів в цих групах фільтрів.

Клас CategoryAdapter [10] – клас, що наслідується від RecyclerView.Adapter та реалізує адаптер для списку категорій. Клас має внутрішній статичний клас CategoryViewHolder, який наслідується від RecyclerView.ViewHolder та містить всі графічні елементи пункту категорії (текстове поле з назвою, кнопку редагування та кнопку видалення) та діалогове вікно для редагування. В конструкторі класу CategoryAdapter ініціалізується список категорій categories та boolean масив isHidden. Перевизначений метод onCreateViewHolder() використовується для встановлення графічного представлення пункту категорії, повертає новий екземпляр класу CategoryViewHolder. В методі onBindViewHolder пункту категорії встановлюються: назва категорії, слухач подій для кнопки видалення, який слугує для видалення категорії за допомогою класу GamesProcessor, та слухач подій для кнопки редагування, який слугує для відкривання CategoryDialog (де кнопці «Оновити» встановлюється слухач подій для збереження оновленої інформації про категорію). Метод getItemCount()

повертає розмір списку categories. Метод getDefaultLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams із розмірами та відступами в графічному представленні за замовчуванням. Метод getZeroLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams із нульовими значеннями висоти та ширини.

Клас GameAdapter [10] – клас, що наслідується від RecyclerView.Adapter та реалізує адаптер для списку ігор. Клас має внутрішній інтерфейс OnGameClickListener із сигнатурою методу onGameClicked() для визначення дії при натиску на пункт гри. Також клас має внутрішній статичний клас GameViewHolder, який наслідується від RecyclerView.ViewHolder та містить всі графічні елементи пункту гри (картинку, текстове поле з назвою, текстове поле з описом, текстове поле зі списком категорій, кнопки-зірочки, кнопку-сердечко, кнопку обрання). В конструкторі класу GameAdapter ініціалізується список ігор games, boolean масив isHidden та onGameClickListener. Метод search() реалізує пошук ігор, в назві або описі яких міститься задана стрічка. Перевизначений метод onCreateViewHolder() використовується для встановлення графічного представлення пункту гри, повертає новий екземпляр класу GameViewHolder. В методі onBindViewHolder пункту гри встановлюються: зображення, назва, опис, стрічка категорій, слухачі подій для кнопок-зірочок, кнопки-сердечка та кнопки обрання за допомогою методів класу ButtonsActions, слухач подій на натиску на пункт гри. Метод getItemCount() повертає розмір списку games. Методи getDefaultLayoutParams() та getZeroLayoutParams() є аналогічними до однойменних методів в класі CategoryAdapter.

Клас SharedGameAdapter [10] – клас, що наслідується від RecyclerView.Adapter та реалізує адаптер для списку рекомендованих ігор. Клас має внутрішній інтерфейс OnSharedGameClickListener із сигнатурою методу onSaredGameClicked() для визначення дії при натиску на пункт рекомендованої гри. Також клас має внутрішній статичний клас SharedGameViewHolder, який наслідується від RecyclerView.ViewHolder та

містить всі графічні елементи пункту рекомендованої гри (текстове поле з назвою, текстове поле з описом, текстове поле зі списком категорій, текстове поле з id чи псевдонімом користувача, який порекомендував гру. В конструкторі класу SharedGameAdapter ініціалізується список рекомендованих ігор sharedGames, boolean масив isHidden, onSharedGameClickListener та екземпляр класу AppDatabase – appDatabase. Перевизначений метод onCreateViewHolder() використовується для встановлення графічного представлення пункту рекомендованої гри, повертає новий екземпляр класу SharedGameViewHolder. В методі onBindViewHolder пункту рекомендованої гри встановлюються: назва, опис, категорії, id користувача, який порекомендував гру, слухач подій при натиску на пункт. В цьому ж методі встановлюється слухач подій на appDatabase, який викликає свій метод onDataChanged() при завантаженні з бази даних інформації про користувача, який порекомендував гру. Якщо інформація успішно завантажилася і у користувача є псевдонім, він записується в призначене текстове поле замість id. Метод getItemCount() повертає розмір списку sharedGames. Методи getDefaultLayoutParams() та getZeroLayoutParams() є аналогічними до однойменних методів в класі GameAdapter.

3.4.7. Пакет activities

Всі класи, які є в пакеті activities, наслідуються від AppCompatActivity, тобто є актівіті.

Клас ProfileActivity – актівіті профілю користувача. В методі onCreate() встановлюється графічне представлення для актівіті, id та псевдонім користувача дістаються з шерд преференсес та записуються у відповідні текстові поля, слухач подій встановлюється для кнопки оновлення псевдоніму. При натиску на кнопку, якщо мережа Інтернет доступна та такого псевдоніму ще не існує, оновлений псевдонім зберігається в шерд преференсес та Firebase. Метод onCreateContextMenu() потрібен для того,

щоб дати можливість копіювати значення id користувача за допомогою пункту «Скопіювати» в контекстному меню, яке з’являється при довгому натисканні на текстовому полі.

Клас CategoriesActivity – активіті для перегляду, редагування та видалення категорій ігор. В методі onCreate() встановлюється графічне представлення для активіті, ініціалізується список категорій за допомогою методу getCategories() класу GamesProcessor та встановлюється адаптер – екземпляр класу CategoryAdapter.

Клас GameInfoActivity – активіті для перегляду детальної інформації, редагування та видалення гри game. В методі initSpinnerAndSetSelection() відбувається ініціалізація випадаючого списку та встановлення обраного значення зі значення, переданого в метод параметром. Метод initMultiSpinner() потрібен для ініціалізації списку і масиву всіх категорій, масиву обраних категорій гри. В методі recommendGame() відбувається перевірка, чи цей користувач вже рекомендував гру game користувачу з id, що дорівнює переданому параметру; якщо не рекомендував, створюється новий екземпляр класу SharedGame зі змінної game та додається в базу даних за допомогою методу addSharedGameToDatabase() об’єкту appDatabase, щоб зразу стати доступною для перегляду у користувача, якому рекомендують гру. В методі updateGame(), якщо в редагованому текстовому полі з назвою гри непорожнє значення і іншої гри з такою назвою не існує, грі встановлюються оновлені характеристики з редагованих текстових полів та випадаючих списків, за неправильного обрання найбільше та найменше значення віку та кількості гравців змінюються місцями, якщо не обрано жодну категорію, грі встановлюється категорія за замовчуванням «загальна категорія». Метод removeGame() забезпечує видалення гри за допомогою методу deleteGame() класу GamesProcessor. В методі onCreate() встановлюється графічне представлення для активіті, з класу GamesProcessor отримується гра за переданою в інтенті назвою та ініціалізує глобальну змінну game, кнопці btnBack (стрілка в лівому верхньому куті екрану)

встановлюється слухач подій для повернення до фрагменту зі списком ігор, для категорій викликається метод initMultiSpinner() та встановлюється слухач подій за допомогою методу setCategoriesListener() класу ButtonsActions, редагованим текстовим полям встановлюються відповідні описові значення зі змінної game (назва, правила і т.д.), вік, кількість гравців та час гри – це випадаючі списки, потрібне значення з гри в яких встановлюються за допомогою методу initSpinnerAndSetSelection(), дати додавання та останнього обрання форматуються за допомогою методу convertDateToLocalString класу Utils, встановлюється слухач подій для кнопки addCategoryButtonI, який слугує для відкривання CategoryDialog (де кнопці «Додати» встановлюється слухач подій для додавання нової категорії), встановлюються слухачі подій для кнопок-зірочок, кнопки-сердечка та кнопки обрання за допомогою методів класу ButtonsActions, встановлюється слухач подій для кнопки shareButton, який слугує для відкривання ShareGameDialog (де кнопці «Рекомендувати» встановлюється слухач подій для рекомендування іншому користувачу гри game за його id або псевдонімом, якщо вони існують, тобто виклику методу recommendGame()), кнопці updateButton встановлюється слухач подій, який забезпечує виклик методу updateGame() та збереження оновленої гри, а кнопці removeButton – слухач подій, який забезпечує виклик методу removeGame() та збереження списку ігор без гри game; елементу-утримувачу зображень встановлюється зображення гри, а якщо його немає – картинка за замовчуванням з кубиками та слухач подій, який при натиску на зображення створює інтент типу «image/*», який запустить активіті одного з додатків, призначених для вибору/створення зображень [8].

Клас SharedGameInfoActivity – активіті для перегляду детальної інформації по рекомендованій грі sharedGame. В методі onCreate() встановлюється графічне представлення для активіті, з інтенту отримується рекомендована гра та ініціалізує глобальну змінну sharedGame, кнопці btnBack (стрілка в лівому верхньому куті екрану) встановлюється слухач подій для повернення до

фрагменту зі списком ігор, текстовим полям встановлюються відповідні описові значення зі змінної sharedGame (назва, опис, правила, вік, кількість гравців, час гри, категорії та id користувача, який порекомендував гру). В цьому ж методі встановлюється слухач подій на appDatabase, який викликає свій метод onDataChanged() при завантаженні з бази даних інформації про користувача, який порекомендував гру. Якщо інформація успішно завантажилася і у користувача є псевдонім, він записується в призначене текстове поле замість id.

3.4.8. Пакет fragments

Всі класи, які є в пакеті fragments, наслідуються від Fragment, тобто є фрагментами.

Клас FromFriendsFragment – фрагмент для перегляду рекомендованих користувачу ігор. Метод onCreateView() встановлює графічне представлення для фрагменту. Метод onPause() викликається, коли фрагмент перестає бути видимим користувачу; в ньому видаляється слухач подій для databaseReference. В методі initRecyclerView() ініціалізується список рекомендованих ігор sharedGames іграми, отриманими з бази даних, які були порекомендовані саме цьому користувачу, встановлюється адаптер – екземпляр класу SharedGameAdapter, якому передається слухач подій, який відповідає за створення інтенту, що відкриває активіті SharedGameInfoActivity. В методі onViewCreated() викликається метод initRecyclerView() та за необхідності користувачу видається повідомлення, що мережа Інтернет відсутня та рекомендовані йому ігри завантажити неможливо.

Клас AddGameFragment – фрагмент для додавання нової гри. Метод onCreateView() встановлює графічне представлення для фрагменту. В методі initSpinner() відбувається ініціалізація випадаючого списку та встановлення обраного значення за замовчуванням. Метод initMultiSpinner() потрібен для ініціалізації списку і масиву всіх категорій, масиву обраних категорій гри та

встановлення слухача подій за допомогою методу setCategoriesListener() класу ButtonsActions. В методі addGame(), якщо в редагованому текстовому полі з назвою гри непорожнє значення і іншої гри з такою назвою не існує, створюється новий екземпляр класу Game з характеристик із редагованих текстових полів і випадаючих списків та викликається метод addGame() класу GamesProcessor (за неправильного обрання найбільше та найменше значення віку та кількості гравців змінюються місцями, якщо не обрано жодну категорію, грі встановлюється категорія за замовчуванням «загальна категорія»). В методі clearFields() всі редаговані текстові поля та випадаючі списки очищуються, встановлюється зображення з кубиками за замовчуванням. В методі onViewCreated() ініціалізуються редаговані текстові поля, випадаючі списки за допомогою методу initSpinner(), елемент-утримувач зображень та кнопки, для категорій викликається метод initMultiSpinner() та встановлюється слухач подій за допомогою методу setCategoriesListener() класу ButtonsActions, встановлюється слухач подій для кнопки addCategoryButtonI, який слугує для відкривання CategoryDialog (де кнопці «Додати» встановлюється слухач подій для додавання нової категорії), встановлюються слухачі подій для кнопок-зірочок та кнопки-сердечка за допомогою методів класу ButtonsActions, кнопці addButton встановлюється слухач подій, який забезпечує виклик методу addGame(), збереження нової гри та очищення полів за допомогою методу clearFields(); елементу-утримувачу зображень встановлюється картинка за замовчуванням з кубиками, та слухач подій, який при натиску на зображення створює інтент типу «image/*», який запустить активіті одного з додатків, призначених для вибору/створення зображень [8].

Клас GamesFragment – фрагмент для перегляду, сортування, фільтрації, пошуку, обрання ігор, отримання рекомендацій у що б пограти, виставлення іграм балів та визначення, які ігри є улюбленими. В класі є змінна games, яка є копією списку ігор games з класу GamesProcessor, та змінна filteredGames, ігри в якій відповідають виставленим фільтрам, містять стрічку пошуку та

списком відображаються для користувача. Метод onCreateView() встановлює графічне представлення для фрагменту та панелі фільтрації. В методі onViewCreated() викликаються методи ініціалізації (які починаються на «init»). В методі onSaveInstanceState() відбувається збереження стану обраного сортування перед тим, як фрагмент стане невидимим, щоб відновити його, коли фрагмент знову стане видимим за допомогою метода onViewStateRestored(). В методі onResume(), який також викликається, коли фрагмент знову стає видимим, перевіряється, чи якась гра не була видалена в GameInfoActivity; якщо так – ця гра видаляється зі списку ігор filterdGames. В методі initSpinner() відбувається ініціалізація випадаючого списку та встановлення обраного значення за замовчуванням. В методі init() ініціалізується графічне представлення для панелі фільтрів, випадаючі списки фільтрів за допомогою метода initSpinner(), випадаючий список сортування, кнопки застосування фільтрів та рекомендацій та текстове поле пошуку. В методі sortWithCurrentSorter() визначається змінна type типу SortType в залежності від обраного значення з випадаючого списку та викликається метод sortGames() для сортування списку ігор filteredGames за типом type. Метод showAll() потрібен для очищення списку filterGames, заповнення його всіма іграми з games та застосування поточного типу сортування за допомогою методу sortWithCurrentSorter(). В методі filterGames(), який відповідає за фільтрацію ігор відповідно до встановлених критеріїв, очищується список filteredGames; якщо в фільтрах не обрана жодна категорія, туди додаються всі ігри зі списку games, а інакше – лише ігри, в яких є хоч одна обрана категорія (з допомогою методу disjoint() класу Collections); якщо в фільтрах обрано бали, то зі списку filteredGames видаляються всі ігри, які мають іншу кількість балів; якщо обрана фільтрація за тим, чи є гра улюбленою, зі списку filteredGames видаляються всі ігри, які не підпадають під цей фільтр; в кінці зі списку filteredGames видаляються всі ігри, мінімальний вік яких менший за встановлений, або максимальний вік більший за встановлений, або мінімальна кількість гравців менша за

встановлену, або максимальна кількість гравців більша за встановлену, або час не дорівнює встановленому. В методі makeRecommendations() отримуються виставлені значення фільтрів, ігри сортуються за рейтинговим балом в порядку його зменшення, обрахованим за алгоритмом рекомендацій, описаним раніше, та з відсортованого списку обираються 6 (або всі, якщо їх менше, ніж 6) перших ігор в тому ж порядку, які після цього і додаються в список filteredGames. В методі filterOrRecommend() якщо жодних фільтрів і стрічку пошуку не встановлено, викликається метод showAll(), інакше – якщо обрана фільтрація, викликаються методи filterGames() та sortWithCurrentSorter(), а якщо рекомендації – метод makeRecommendation(). В методі initDrawer() кнопкам фільтрації та рекомендацій встановлюються слухачі подій, які потрібні для зміни, яка кнопка з цих двох є активною, та для виклику методу filterOrRecommend() (якщо активною стає кнопка рекомендацій, то кнопка сортування стає неактивною, тобто забороняється сортувати ігри), кнопці сортування встановлюється слухач подій, який забезпечує виклик методу sortWithCurrentSorter(), полю пошуку встановлюється слухач подій, який при кожній зміні введених символів викликає метод search() класу GameAdapter. В методі initFilter() відбувається ініціалізація всіх фільтрів, встановлення випадаючим спискам мінімального та максимального віку гравців, мінімальної та максимальної кількості гравців і приблизного часу гри слухачів подій, які забезпечують виклик методу filterOrRecommend() при виборі якогось значення, встановлення слухача подій екземпляру класу ExpListViewAdapterWithCheckbox, який також забезпечує виклик методу filterOrRecommend(). В методі initRecyclerView() ініціалізується список ігор games за допомогою методу getGames() класу GamesProcessor та список filteredGames за допомогою методу getCopyOfGames() класу GamesProcessor, викликається метод sortWithCurrentSorter() та встановлюється адаптер – екземпляр класу GameAdapter, якому передається слухач подій

GameAdapter.OnGameClickListener(), який відповідає за створення інтенту, що відкриває активіті GameInfoActivity.

Висновки

Отже, під час роботи було:

а) проаналізовано існуючі рішення;

б) визначено цільову аудиторію;

в) вигадано назву та логотип додатку;

г) детально розписано функціонал додатку;

д) розроблено алгоритм рекомендацій у що б пограти;

е) створено дві ER-моделі;

ж) реалізовано Android додаток із повним запланованим функціоналом;

з) описано роботу всіх класів та методів додатку.

Список використаних джерел

1. Документація по Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/components/activities/intro-activities
2. Документація по Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/fragments
3. Документація по Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/content/Intent
4. Документація по Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/content/SharedPreferences?hl=en
5. Топ мов програмування для розробки Android додатків [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/
6. Бібліотеки для роботи з датою та часом [Електронний ресурс] – Режим доступу до ресурсу: https://camposha.info/android-examples/android-datetime-libraries/#gsc.tab=0
7. Встановлення та налаштування Firebase в Android проєкті [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/database/android/start?authuser=1&hl=en
8. Вибір зображення з галереї та інших додатків [Електронний ресурс] – Режим доступу до ресурсу: http://learn-android.ru/news/vybiraem_izobrazhenie_iz_galerei_s_pomoshhju_intent_action_pick/2015-04-19-95.html
9. Як написати клас JSONHelper [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/java/android/13.3.php
10. Використання RecyclerView.Adapter [Електронний ресурс] – Режим доступу до ресурсу: https://guides.codepath.com/android/using-the-recyclerview