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 Вовк Наталя Євгенівна, старший викладач
2. Строк подання студентом роботи: 6 червня 2022
3. План роботи

Анотація

Вступ

Розділ 1.

Розділ 2.

Розділ 3.

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 каліф. 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, огляд літературних джерел | 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 5](#_Toc104375153)

[Перелік термінів та умовних позначень 6](#_Toc104375154)

[Вступ 7](#_Toc104375155)

[Розділ 1. 8](#_Toc104375156)

[1.1 Аналіз існуючих рішень 8](#_Toc104375157)

[1.2 Обгрунтування вибору інструментів??? 8](#_Toc104375158)

[1.3 І щось ще 8](#_Toc104375159)

[Розділ 2. 9](#_Toc104375160)

[2.1 Функціонал додатку 9](#_Toc104375161)

[2.2 Цільова аудиторія 9](#_Toc104375162)

[2.3 І ще щось 10](#_Toc104375163)

[Розділ 3. Аналіз створеного додатку 11](#_Toc104375164)

[3.1 Опис БД 11](#_Toc104375165)

[3.2 Опис структури проєкту 12](#_Toc104375166)

[3.3 Опис використаних бібліотек 13](#_Toc104375167)

[3.4 Опис Java-класів 13](#_Toc104375168)

[3.4.1 MainActivity 13](#_Toc104375169)

[3.4.2 Пакет models 13](#_Toc104375170)

[3.4.3 Клас AppDatabase в пакеті database 14](#_Toc104375171)

[3.4.4 Пакет helpers 14](#_Toc104375172)

[Висновки 16](#_Toc104375173)

[Список використаних джерел 17](#_Toc104375174)

Анотація

Дана робота присвячується 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 систематизації та рекомендації настільних ігор. Детально описано функціонал та структуру застосунку, структуру даних та способи їх збереження.

Перелік термінів та умовних позначень

Актівіті (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 актівіті за допомогою методу [startActivity](https://developer.android.com/reference/android/content/Context#startActivity(android.content.Intent))(). [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 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 упорядкування та рекомендацій моїх настільних ігор. Над такою темою мені захочеться працювати і потім я з радістю буду користуватися своїм додатком.

Розділ 1.

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, актівіті 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 айтемів 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 айтемі гри було скомбіновано з айтему страви в додатку McDonalds та айтему новини в додату «Київ Цифровий».

1.2 Обгрунтування вибору інструментів???

івсці

1.3 І щось ще

Олджп

Розділ 2.

2.1 Функціонал додатку

В додатку реалізовано такі функції: додавання, редагування, видалення категорій ігор; додавання, редагування (з можливістю додати зображення), видалення, фільтрація, рекомендації, пошук, сортування, обрання для гри ігор, визначення гри улюбленою (натисканням на сердечко). 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 з 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.

2.2 Цільова аудиторія

Додаток розраховується на два типи користувачів:

1. Люди будь-якого віку старше 5 років з будь-якого населеного пункту України та будь-яким рівнем забезпеченості. У них є пристрій з операційною системою Android та можливість встановити додаток з Play Market. 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 та користуватися додатком для вибору у що б пограти з друзями та родичами.
2. Працівники та відвідувачі антикафе/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.

2.3 І ще щось

Фвс

Розділ 3. Аналіз створеного додатку

3.1 Опис БД

З точки зору схеми бази даних, додаток можна поділити на дві частини за місцем зберігання: в JSON файлах, щоб надавати доступ в будь-який час (рис. 1), і в Firebase, щоб забезпечити зв’язок між різними користувачами (рис. 2).

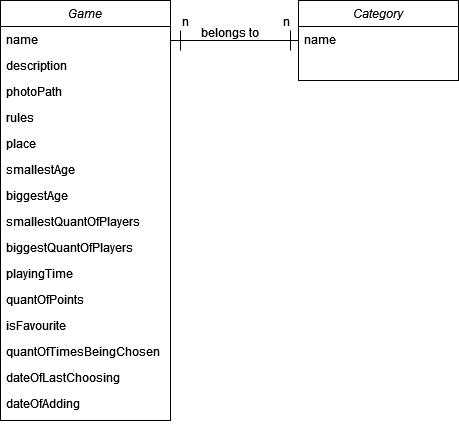


Рисунок 1 ER-модель першої частини даних

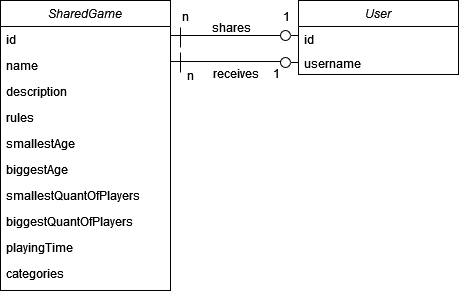


Рисунок 2 ER-модель другої частини даних

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). Сутність рекомендована гра SharedGame має унікальний id і позичає з сутності Game поля, які є 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 користувачу. Сутність користувач User містить поля id та псевдонім.

Необхідність відділити сутність SharedGame від Game викликана декількома чинниками. По-перше, екземляр SharedGame створюється кожен раз, як користувач хоче порекомендувати гру. Відповідно, додається унікальний ідентифікатор, id того, хто рекомендує, і id того, кому рекомендують. Також забирається 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.

Можна додати детальніше про Firebase та JSON.

3.2 Опис структури проєкту

Проєкт містить: папку myboardgames з файлами класів java, папку res з файлами ресурсів xml, AndroidManifest та файли налаштувань Android-проєкту.

Файли xml виконують такі функції: графічне представлення актівіті та фрагментів, навігація між актівіті 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 актівіті (із вказуванням, які актівіті є батьківськими для інших).

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.

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

Підключено біліотеку firebase:firebase-database для зберігання даних в нереляційній базі даних Realtime Database Firebase.

3.4 Опис Java-класів

3.4.1 MainActivity

MainActivity – це основне актівіті додатку. В методі onCreate() встановлюється графічне представлення для актівіті, налаштовуються пункти нижнього меню та підключається верхнє приховане меню. Метод onCreateOptionsMenu() встановлює графічне представлення для верхнього меню. В методі onOptionsItemSelected() при обранні пункту верхнього меню відбувається перенаправлення на відповідне до цього пункту актівіті за допомогою інтентів. Метод onResume() викликається, коли актівіті 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.

3.4.2 Пакет models

Клас Game реалізує сутність настільної гри користувача. Клас містить всі необхідні поля, зазначені на ER-моделі на рис. 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. 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 уже 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 [5] – 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.

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 класу JSONHelper importGamesFromJSON(). У методі loadCategories() відбувається ініціалізація списку categories за допомогою методу класу JSONHelper importCategoriesFromJSON(). 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 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 актівіті чи фрагменті (наприклад, виставлення грі балів при натиску на зірочки присутнє в актівіті редагування гри та в айтемі 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 п’яти 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 кнопку скасування та 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 другу гри. В конструктор передається слухач подій для правої кнопки. 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 друга 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 finsl 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. Перевизначениий метод getChild()повертає назву чекбокса певної групи на певній позиції. Перевизначений метод getGroupId() повертає позицію групи чекбоксів. Перевизначений метод getChiId() 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 – 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 айтема 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 айтему категорії, повертає новий екземпляр класу CategoryViewHolder. В методі onBindViewHolder айтему 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 з розмірами та відступами в графічному представленні за замовчуванням. Метод getZeroLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams з нульовими значеннями висоти та ширини.

Клас GameAdapter – 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 айтем гри. Також клас має внутрішній статичний клас GameViewHolder, який наслідується від RecyclerView.ViewHolder та містить всі графічні елементи айтема 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 айтему гри, повертає новий екземпляр класу GameViewHolder. В методі onBindViewHolder айтему 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 айтем. Метод getItemCount() повертає розмір списку games. Метод getDefaultLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams з розмірами та відступами в графічному представленні за замовчуванням. Метод getZeroLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams з нульовими значеннями висоти та ширини.

Клас SharedGameAdapter – 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 айтем рекомендованої гри. Також клас має внутрішній статичний клас SharedGameViewHolder, який наслідується від RecyclerView.ViewHolder та містить всі графічні елементи айтема 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 айтему рекомендованої гри, повертає новий екземпляр класу SharedGameViewHolder. В методі onBindViewHolder айтему рекомендованої гри встановлюються: назва, опис, стрічка категорій, id користувача, який порекомендував гру, слухач подій на натиску на айтем. 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. Метод getDefaultLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams з розмірами та відступами в графічному представленні за замовчуванням. Метод getZeroLayoutParams() повертає екземпляр класу RecyclerView.LayoutParams з нульовими значеннями висоти та ширини.

Висновки

Вся

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. Як написати клас JSONHelper [Електронний ресурс] – Режим доступу до ресурсу: <https://stackoverflow.com/questions/45580324/cant-create-a-file-from-a-android-helper-class>