ПРИКЛАДНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК,
БЕЗПЕКИ ТА МАТЕМАТИКИ, 2023, № 2

УДК 004.4-024.4:004.43

РЕАЛІЗАЦІЯ ДОДАТКУ СТВОРЕННЯ ГЕНЕАЛОГІЧНОГО ДЕРЕВА ДЛЯ ПРИСТРОЇВ З ОС ANDROID МОВОЮ ПРОГРАМУВАННЯ KOTLIN

Макута М.Ю.
Глинчук Л.Я. (ORCID: 0000-0002-8943-9604)
Гришанович Т.О. (ORCID: 0000-0002-3595-6964)
Волинський національний університет імені Лесі Українки

IMPLEMENTATION OF THE APPLICATION FOR CREATING A GENEALOGICAL TREE FOR DEVICES WITH THE ANDROID OS USING THE KOTLIN PROGRAMMING LANGUAGE

Makuta M. U., Hlynchuk L. Ya., Hryshanovych T. O.
Lesya Ukrainka Volyn National University

Abstract. The work describes the features of the Gedcom format and the implementation of the application that builds a genealogical tree. Since similar systems are used by Ukrainian scientists and ordinary people who are interested in their genealogy for many reasons, it was appropriate to implement such a free opportunity for devices with the Android operating system. It is also relevant because it is extremely convenient to work with genealogical data using your own phone, which is always with you. In order to implement the application, which will be used to build a genealogical tree, an analysis of the problems of such applications as Family Historian, MyHeritage, Gramps was carried out. And also, their advantages and disadvantages are revealed. As a result of the review of similar software, the idea and appearance of the future application, functional requirements, requirements for functional characteristics and reliability were set. To build a genealogical tree in the application, the Gedcom format was chosen for several important reasons: it provides a standardized way of presenting genealogical information; allows users to transfer their genealogical data between different programs without losing information or changing the data format and makes it easier for users; allows users to share their data and work together on different projects. The Android Studio integrated development environment and the Kotlin programming language were chosen as the main tool. Since this language is considered the main one in the development of software for the Android platform and uses the OOP paradigm, the code is written in this style. The entire interface is built on the basis of fragments, which are managed by the fragment manager. Every Android application must contain at least one Activity in order to run and perform its tasks. This development contains one Activity, which meets the minimum requirements for launching the application, and 4 fragments: MainFragment, GraphFragment, DetailsFragment, SettingsFragment. Activity plays the role of a navigator for these fragments. In the future, it will be advisable to add synchronization and backup, as well as authentication, encryption and access control mechanisms to ensure user privacy.

Keywords: addition; family tree; gedcom format; kotlin programming language; a piece of code.

Вступ. Розвиток технологій в сучасному світі сприяє створенню нових можливостей для зберігання і передачі інформації. Генеалогічне дерево – це важлива складова родинної історії, яка дозволяє зберігати інформацію про предків та нащадків. Результати досліджень часто відображаються у вигляді діаграм або генеалогічних схем. Генеалогія як історична дисципліна пропонує чотири варіанти (низхідне вертикальне древо, висхідне вертикальне древо, горизонтальне древо, кругова таблиця) оформлення інформації про родичів в одне ціле. [1] Вони дозволяють відобразити родинні зв'язки та родоводи. Дане дослідження пропонує огляд подібних додатків та реалізацію власного для створення генеалогічного дерева для пристроїв з операційною системою Android. Включає в себе опис функціональних вимог, технологій і архітектури, необхідних для реалізації такого додатку. Тематика дослідження є актуальною, оскільки створення родинного дерева актуальне ще з давніх часів. Актуальність розробки показує ще і той факт, що є цікаві аналоги з різними функціями. Використання мобільних пристроїв і

додатків з такою функціональністю стає надзвичайно зручним способом роботи з генеалогічними даними.

Оскільки, раніше це відбувалося вручну на папері, то з часом могло зіпсуватися. Тепер є можливість створити і зберігати генеалогічне дерево в електронному форматі, використовуючи зручний та простий додаток, або, при необхідності, роздрукувати. Програмне забезпечення, яке широко використовується для вирішення генеалогічних завдань, виконує різноманітні функції, такі як збір, аналіз та оприлюднення зібраних даних. На ринку існує низка сервісів, які дозволяють користувачам створювати та відображати свої родинні дерева. Проте багато з них обмежуються лише графічним відображенням і не надають можливості детального заповнення інформації для кожної окремої особи, збереженням інформації у відповідний формат та інше. В даній роботі проаналізовано процес розробки додатку, який, якраз таки, і дає можливість зберігати інформацію у відповідний зручний формат та має інші особливості.

Методологія дослідження. Перші дослідники, які займались українським родоводом були: Яків Маркевич, Олександр Лазаревський, Вадим Модзалевський, Григорій Милорадович. Зверталися в своїх дослідах до питань української генеалогії історики Осип Бодянський, Микола Костомаров, В'ячеслав Липинський, Михайло Грушевський, Дмитро Багалій. Започатковані були дослідження та культурологічні читання у Дрогобицькому державному педагогічному інституті 9 березня 1992 р. У період 1991-2007 рр. майже 3 тис. студентів працювали за програмою народознавчих досліджень. Українська традиція передбачає знання свого роду до сьомого коліна. Таких досліджень понад 25 %. Більшість досліджень охоплює 4-5 поколінь родини, трапляються дослідження, які охоплюють 8-14 поколінь. [2]

Серед дослідників нашого регіону, 2-є людей займаються пошуком та дослідженням генеалогічного дерева своєї сім'ї для того, аби лишити гарну згадку і пам'ять для майбутніх поколінь. Ці завзяті люди – це 2 рідні брати Янко Тимофій та Янко Микола. Обоє проживають у Волинській області, вони уже люди літнього віку, але не дивлячись на це, із захопленням виконують поставлену ціль. В результаті планується об'ємне видання.

Як бачимо, такий напрям не втратив актуальності, а сучасні інформаційні технології допомагають зробити цей процес набагато швидше та зберегти, знайдену інформацію про родину, надійніше.

Для аналізу проблем сучасних додатків створення родинних дерев варто розглянути існуючі аналоги та виявити їх переваги та недоліки. Для дослідження були обрані такі додатки як Family Historian, MyHeritage, Gramps.

Family Historian – програма для створення генеалогічних програм, яку випускає Calico Pie Limited лише для ОС Windows 8 і новіше. Мова програмування – C#. Архітектура – desktop application. Плюси: можна додати безліч осіб в дерево; є підтримка зображень; працює без підключення до інтернету. Мінуси: програма не безкоштовна; підтримка лише Windows. [3]

MyHeritage – додаток, який містить багато різних функцій, такий як: пошук предків по різним документам, докладна історія родича, пошук по ДНК, і багато чого іншого. Використовує Java, C++ і JavaScript в якості основних мов програмування. Архітектура – клієнт–сервер. Також наявний функціонал по створенню і редагуванню нових дерев. Підтримує всі мобільні і десктоп платформи і має веб версію. Плюси: багатий функціонал; збереження даних між пристроями; дерево можна експортувати в форматі pdf, або зображенням. Мінуси: більшість функцій платні; потрібна реєстрація на сайті або в додатку. [4]

Gramps – програма для створення генеалогічних дерев з можливістю створення звітів в будь якій формі (навіть у власній), написана на Python, як вільне програмне забеспечення. Архітектура – desktop application. Доступна на операційних системах Unix і Windows. Плюси: повністю безкоштовна; багато форматів для звітів; працює без доступу до інтернету; є багато доповнень. Мінуси: нема мобільної версії; свій формат для збереження даних, але можливий експорт в формат Gedcom.

Метою дослідження є дослідження формату Gedcom та реалізація додатку для створення генеалогічних дерев для пристроїв з ОС Android, з використанням мови програмування Kotlin.

Методи дослідження, які були використанні в роботі: аналіз літературних джерел, порівняння та дослідження плюсів та мінусів аналогічного програмного забезпечення, моделювання зовнішнього вигляду додатку, проектування структури додатку.

Результати дослідження та їхнє обговорення. В додатках для створення генеалогічних дерев використовують формат Gedcom. Причин для використання формату Gedcom є декілька, ось основні з них:

- Gedcom надає стандартизований спосіб представлення генеалогічної інформації. Це дозволяє різним програмам для генеалогії спілкуватися між собою, обмінюватися даними та інтегруватися.
- Даний формат дозволяє користувачам переносити свої генеалогічні дані між різними програмами без втрати інформації або зміни формату даних.
- Використання стандарту спрощує життя користувачам, оскільки вони можуть легко переміщати свої дані між різними сервісами та програмами без необхідності конвертації або ручного введення.
- Стандартний формат даних сприяє розвитку генеалогічної спільноти, оскільки він дозволяє користувачам обмінюватися своїми даними та спільно працювати над різними проектами.

Формат Gedcom (Genealogical Data Communication) став стандартом для обміну генеалогічною інформацією між різними програмами і службами. Хоча Gedcom має свої обмеження та недоліки, він залишається широко використовуваним у генеалогічній галузі. Даний формат підтримують більшість програм для генеалогічних досліджень. Розширення файлу може бути .ged або .gdz (з підтримкою медіа). Основні можливості формату: підтримка кодування UTF–8; невелика складність в читанні/записі; наявність розширень; підтримка медіа (фото). Цей формат не стандартизований у світі, тому він де-факто є стандартом, через його популярність і підтримку. Центральним елементом цього формату є особа. Вона може виступати будь-яким членом родини. Також це дозволяє переглядати всіх членів родини, формат розрізняє батька, матір і дітей, як основні елементи родини. Не існує ніяких обмежень на кількість партнерів у батьків, але тоді вони будуть в різних сім'ях. Також сім'я може бути без дітей. Якщо у дитини є партнер, тоді це вже буде нова сім'я. [6]

Отже, наступна задача – розробити програмний засіб, який буде працювати на ОС Android, з використанням формату Gedcom. Він повинен працювати на більшості пристроїв, і бути простим у використанні.

Основні функціональні вимоги до розробки, які були поставлені, після розгляду аналогічних додатків наступні: завантаження файлів у форматі Gedcom; малювання графу; можливість приближення або віддалення і навігації; експорт в PDF; можливість налаштувати малювання графу; безкоштовне використання.

Також були задані вимоги до функціональних характеристик та надійності. Тому, система повинна надавати наступні можливості: ефективність – працює з великою кількістю об'єктів; безпечність – програма не повинна розголошувати конфіденційні дані третім сторонам; продуктивність – однаково стабільна робота на різних пристроях з різними апаратними і програмними засобами. Надійність: у випадку коли програма завершує роботу з помилкою, має бути збережений останній результат; при повторному відкритті програми, програма має запитати про відкриття аварійного збереження для відновлення роботи.

Призначення розробки полягає в зручному і простому створенні генеалогічних дерев. Стиль інтерфейсу базується на стилі ОС Android – Material Design. Назва розробки – Family Tree (скорочено – fatr). В нашому випадку – в якості архітектури буде виступати Android-додаток. Проєкт буде використовуватись для перегляду і зміни генеалогічних дерев. В якості формату для зберігання, будемо використовувати формат Gedcom, так як він надає всі типи даних і є де-факто стандартом в цій сфері. UML діаграма цього формату зображена на рис. 1. Для зручного перегляду також буде додано: підтримку жестів; редагування осіб, які вже присутні в дереві.

Так як розроблялася програма для ОС Android, то відповідно були використані інструменти, які для цього призначені. Прикладом такого інструменту є IDE (Інтегроване середовище розробки) від компанії Google – Android Studio [7]. Воно містить всі необхідні інструменти для повноцінної роботи. Основна платформа для розробки і тестування – GNU/Linux (дистибутив Arch Linux). В якості мови програмування для розробки було обрано мову Kotlin, так як це головна мова в розробці ПЗ для платформи Android, вона має зручний і простий синтаксис, і містить багато корисних функцій. [8] Мова програмування Kotlin – компільована і може запускатись як на віртуальній машині Java, так і без неї, через використання LLVM, яка надає можливість компіляції в машинний код. Також вона може компілюватись в JavaScript, для фреймворку React. Kotlin сумісна з Java на рівні байт-коду для віртуальної машини Java, тому виконання коду Java у коді Kotlin або навпаки – можливе і не сповільнює розробку ПЗ.

Оскільки, використано мову програмування Kotlin, яка використовує парадигму ООП (Об'єктно-орієнтовного програмування), то і код написаний в цьому стилі. Весь інтерфейс побудовано на основі фрагментів, якими керує фрагментний менеджер [9]. Фрагмент – це такий модуль інтерфейсу, який можна додати до існуючого екрану і робити якісь дії, фрагмент обов'язково має предка – Activity або інший фрагмент [10]. Activity – це головний екран та спосіб запуску в додатку. Кожна Android програма має містити хоча б одне Activity, для того щоб запуститись і виконувати свої завдання.

Кожен фрагмент або Activity має свій макет (layout), в якому розміщені всі елементи керування, такі як: кнопка, текстове поле, поле для вводу текстових даних, зображення, випадаючий список, простий список та інші. Кожен елемент керування має свій унікальний ідентифікатор (id), і може задаватись в макеті, або такий елемент можна створити в вихідному коді програми без макету. Макет може бути написаний окремо у файлі макету або у вигляді коду разом з логікою програми. Макети пишуть мовою XML, які при збірці стискаються.

Всього в проекті є 1 Activity, яке задовольняє мінімум для запуску додатку, і 4 фрагменти. Activity тут відіграє роль навігатора по цих фрагментам. Головний фрагмент (MainFragment.kt) – це початковий фрагмент (рис. 2), який показується при вході в додаток, він має декілька кнопок, які мають перехід на інші фрагменти.

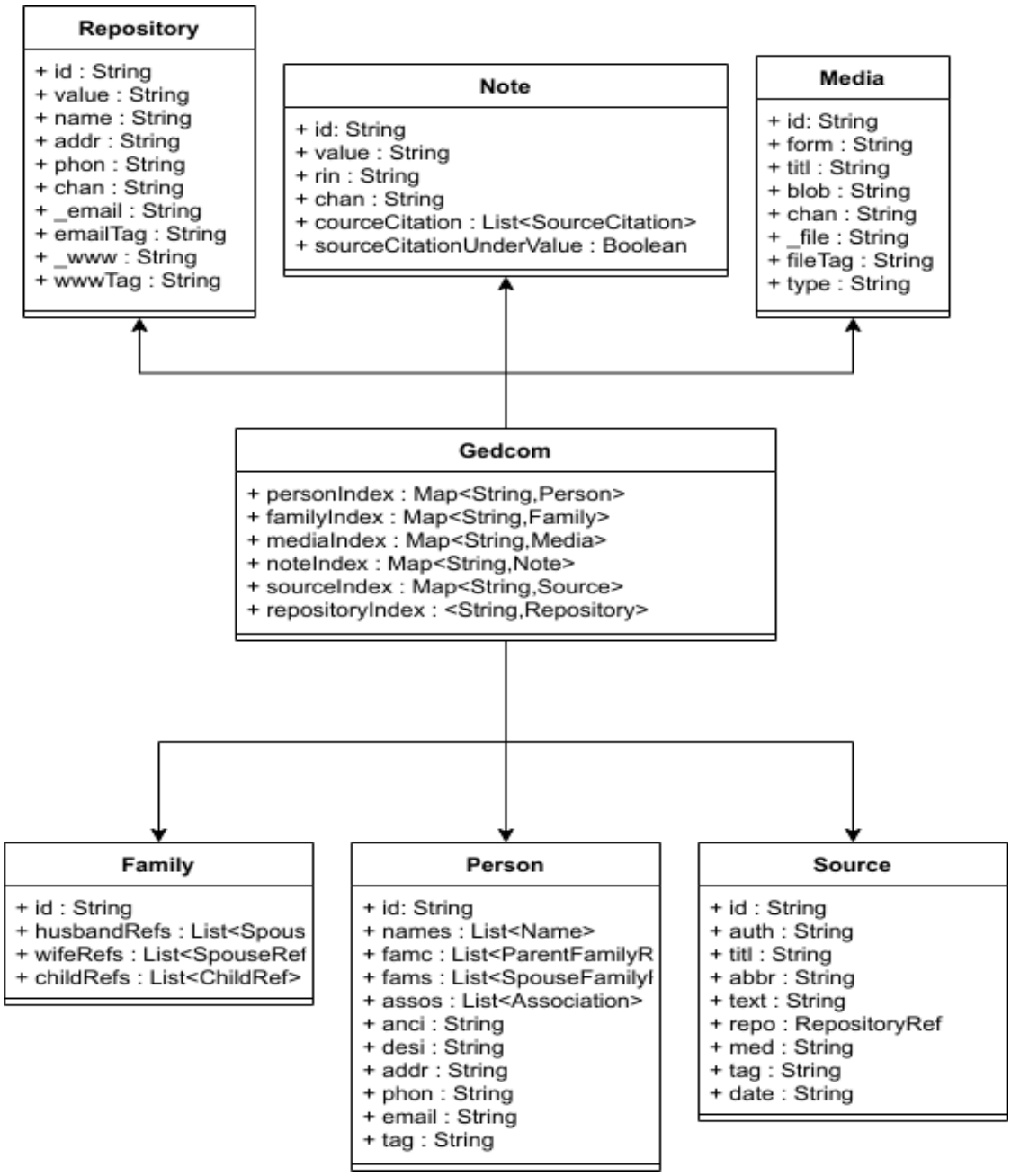

Рис. 1. UML-діаграма формату Gedcom

GraphFragment – основний фрагмент в додатку. Він дає можливість перегляду і редагування генеалогічних дерев. В ньому передбачено можливість приближення/віддалення, якщо дерево велике, і також можливо переміщуватись по дереву за допомогою жестів. Також на цьому фрагменті є кнопки для додавання особи, збереження, експорту в PDF і оновлення. На рисунках 3, 4, 5 зображено фрагменти коду для додавання особи, збереження, експорту дерева в файл PDF відповідно. Для компонування генеалогічного дерева було використано бібліотеку GedcomGraph, яка приймає дані з формату Gedcom і повертає дані, які містять координати для відображення [11]. Вона не займається малюванням, лише для компонування.

Також було створено фрагмент для створення і редагування нової особи – DetailsFragment. Для особи потрібно вказати ім'я і стать, а також її зв'язки з іншими особами в дереві, після цього її можна розмістити на дереві.

Останній фрагмент – фрагмент з налаштуваннями (SettingsFragment.kt), він дозволяє змінити налаштування для відображення дерева, такі як кількість предків, нащадків, не рідних братів і т.д. Ці налаштування застосовуються до всіх відкритих і нових файлів, з якими буде працювати користувач в майбутньому.

```
private lateinit var cardNew  : MaterialCardView
private lateinit var cardOpen : MaterialCardView
private lateinit var cardStng : MaterialCardView
private lateinit var tree     : ImageView

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    cardNew     = view.findViewById(R.id.btn_new)
    cardOpen    = view.findViewById(R.id.btn_open)
    cardStng    = view.findViewById(R.id.btn_settings)
    tree        = view.findViewById(R.id.i_tree)

    cardNew.setOnClickListener {
        findNavController().navigate(R.id.action_mainFragment_to_graphFragment,
            bundleOf(FILE_NEW to true, FILE_URI to "")
        )
    }
    cardOpen.setOnClickListener {
        fileLoader.launch("application/*")
    }
    cardStng.setOnClickListener {
        findNavController().navigate(R.id.action_mainFragment_to_settingsFragment)
    }

    tree.setImageResource(
        if(requireContext().isDarkThemeOn()){
            R.drawable.tree_night
        }else{
            R.drawable.tree_light
        }
    )

    val (a,b) = if(requireContext().isDarkThemeOn()){
        Pair(R.color.dark_secondary, R.color.dark_secondary_sec)
    }else{
        Pair(R.color.light_secondary, R.color.light_secondary_sec)
    }

    view.background = requireContext().gen(
        arrayListOf(R.drawable.leaf,R.drawable.tree_1,R.drawable.tree_2),
        requireContext().displaySize(), a,b
    )
}
```

Рис. 2. Ініціалізація фрагменту MainFragment

```
override fun onAdd(obj: Person) {
    if(gedcom.people.isEmpty()){
        obj.id = "I1"
        gedcom.addPerson(obj)
        gedcom.addFamily(Family().apply {
            id = "F1"
            if(Gender.isMale(obj))
                addHusband(SpouseRef().apply {
                    ref = obj.id
                })
            else if(Gender.isFemale(obj))
                addWife(SpouseRef().apply {
                    ref = obj.id
                })
        })
        completeSelect(obj,0)
        updateUI()
    }else {
        editMode = true
        obj.id = newID(gedcom.people.last().id)
        gedcom.addPerson(obj)
        frustum = obj
        snack(getString(R.string.choose_person))
    }
    gedcom.createIndexes()
    gedcom.updateReferences()
}
```

Рис. 3. Фрагмент коду, який відповідає за додавання нової особи

```
ged.setOnClickListener {
    val name = if(fName.isNotEmpty())
        "${fName.substring(fName.indexOf(':')+1,fName.indexOf('.'))}.ged"
    else
        "tree.ged"
    save.launch(name)
}

private val save =
registerForActivityResult(ActivityResultContracts.CreateDocument("*/*")){
if(it != null){
lifecycleScope.launch {
    val out: OutputStream? =
    requireContext().contentResolver.openOutputStream(it)
    if (out != null) {
        GedcomWriter().write(gedcom, out)
        out.close()
        Toast.makeText(context, R.string.saved_ok,
        Toast.LENGTH_LONG).show()
    } else {
        Toast.makeText(context, R.string.saved_bad,
        Toast.LENGTH_LONG).show()
    }
}
}else{
    Toast.makeText(requireContext(), R.string.no_directory,
    Toast.LENGTH_SHORT).show()
}
```

Рис. 4. Фрагмент коду для зберігання у файл

```
pdf.setOnClickListener {
    val name = if(fName.isNotEmpty())
        "${fName.substring(fName.indexOf(':')+1,fName.indexOf('.'))}.pdf"
    else
        "tree.pdf"
    export.launch(name)
}

 private val export =
 registerForActivityResult(ActivityResultContracts.CreateDocument("applica
 tion/pdf")){
if(it != null){
    for (i in 0 until graphView.childCount) {
        graphView.getChildAt(i).invalidate()
    }
    val document = PdfDocument()
    val pageInfo = PageInfo.Builder(graphView.width, graphView.height,
    1).create()
    val page = document.startPage(pageInfo)
    page.canvas.drawColor(if(requireContext().isDarkThemeOn())
    Color.BLACK else Color.WHITE)
    graphView.draw(page.canvas)
    document.finishPage(page)
    try {
        val out: OutputStream? =
        requireContext().contentResolver.openOutputStream(it)
        document.writeTo(out)
        out!!.flush()
        out.close()
    } catch (e: Exception) {
        Toast.makeText(context, e.localizedMessage,
        Toast.LENGTH_LONG).show()
    }
    Toast.makeText(context, R.string.pdf_exported_ok,
    Toast.LENGTH_LONG).show()
}else{
    Toast.makeText(requireContext(), R.string.pdf_exported_bad,
    Toast.LENGTH_SHORT).show()
}
```

Рис. 5. Код, який відповідає за експорт в PDF

Висновки. У роботі розглянуто процес розробки додатку мовою програмування Kotlin для пристроїв з ОС Android. Продемонстровано деякі частини коду. Додаток вміє: компонувати дерева; має можливість приближення або віддалення і навігації; має можливість налаштувати малювання графу; читати і зберігати файли Gedcom; експортувати дерева у файл PDF.

Дане ПЗ можна використовувати всім у кого є бажання зберегти пам'ять про родину та побачити усі зв'язки не витрачаючи на це багато часу. Основні корисні застосування генеалогічного дерева: в медицині, для діагностики та профілактики захворювань, які можуть бути успадковані; для знання сімейної історії; в психології як засіб розуміння сьогодення людей; в антропології дозволяє вивчати походження народів.

В перспективу було б доцільним додати синхронізацію та резервне копіювання: для забезпечення зручності та безпеки, додаток може містити можливість синхронізації даних з хмарними службами, такими як Google Drive або Dropbox, як його частину резервного копіювання. Це дозволить користувачам зберігати свої дані в безпечному місці та отримувати до них доступ з різних пристроїв. Також було б добре, щоб додаток мав механізми аутентифікації, шифрування і контролю доступу до забезпечення приватності користувача.

Бібліографія

1. Український генеалогічний портал: Дослідження дерев - Родовід. *Багатомовне генеалогічне дерево - Родовід*. URL: https://uk.rodovid.org/wk/ Український_генеалогічний_портал:Дослідження_дерев.
2. Філософія родознавства - С. Черепанова. *Головна*. URL: http://vysochanskiy-sas.com/pro-rodoznavstvo/289-filosofiya-rodoznavstva-s-cherepanova?showall=1.
3. Family Historian – genealogy and family tree software. *Family Historian – genealogy and family tree software*. URL: https://www.family-historian.co.uk/.
4. Безкоштовне родинне дерево, генеалогія, сімейна історія та ДНК-тестування. *MyHeritage*. URL: https://www.myheritage.com.ua/.
5. Gramps – Free Genealogy Software: Open Source Free Genealogy Software. URL: https://gramps-project.org/blog/.
6. Gedcom | Ancestris - Documentation. *Ancestris - Documentation*. URL: https://docs.ancestris.org/books/user-guide/page/gedcom.
7. New features in Android Studio Preview | Android Developers. *Android Developers*. URL: https://developer.android.com/studio/preview/features.
8. Kotlin language specification. *Kotlin Programming Language*. URL: https://kotlinlang.org/spec/introduction.html.
9. Fragments | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/fragments.
10. Introduction to activities | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/components/activities/intro-activities.
11. GitHub - michelesalvador/GedcomGraph: Java library to build genealogical trees on top of a FamilySearch GEDCOM. *GitHub*. URL: https://github.com/michelesalvador/GedcomGraph.

References

1. Ukrainian genealogical portal: Tree research - Genealogy. *Multilingual family tree - Genealogy*. URL: https://uk.rodovid.org/wk/Ukrainian_genealogical_portal:Doslyzhnezhnye_derev.
2. Philosophy of genealogy - S. Cherepanova. *Main*. URL: http://vysochanskiy-sas.com/pro-rodoznavstvo/289-filosofiya-rodoznavstva-s-cherepanova?showall=1.
3. Family Historian – genealogy and family tree software. *Family Historian – genealogy and family tree software*. URL: https://www.family-historian.co.uk/.
4. Free family tree, genealogy, family history and DNA testing. *MyHeritage*. URL: https://www.myheritage.com.ua/.
5. Gramps – Free Genealogy Software: Open Source Free Genealogy Software. URL: https://gramps-project.org/blog/.
6. Gedcom | Ancestris - Documentation. *Ancestris - Documentation*. URL: https://docs.ancestris.org/books/user-guide/page/gedcom.
7. New features in Android Studio Preview | Android Developers. *Android Developers*. URL: https://developer.android.com/studio/preview/features.
8. Kotlin language specification. *Kotlin Programming Language*. URL: https://kotlinlang.org/spec/introduction.html.
9. Fragments | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/fragments.
10. Introduction to activities | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/components/activities/intro-activities.
11. GitHub - michelesalvador/GedcomGraph: Java library to build genealogical trees on top of a FamilySearch GEDCOM. *GitHub*. URL: https://github.com/michelesalvador/GedcomGraph.