Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Небесний А.В.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Небесному Андрію Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Небесний А.В.
(прізвище та ініціали)

АНОТАЦІЯ

Небесний А.В. Програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Метою розробки є програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Результат роботи – програмна реалізація системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на бездротових мережних пристроях під керуванням ОС Android.

Програму розроблено в середовищі Java, Eclipse, Android SDK.

Ключові слова: комп'ютерна інженерія, керування файловими операціями, ОС Android

ABSTRACT

Nebesnyi A.V. File management software based on wireless network devices running Android OS. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for a file operations management system based on wireless network devices running the Android OS.

The purpose of the development is the software of the file operations management system based on wireless network devices running the Android OS.

The result of the work is a software implementation of the file operations management system based on wireless network devices running the Android OS.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on wireless network devices running Android OS.

The program was developed in the environment of Java, Eclipse, Android SDK.

Keywords: computer engineering, file operations management, Android OS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

USB	–	Universal Serial Bus – універсальна послідовна шина;
URL	–	universal resource locator – локатор ресурсів Інтернет;
HTML	–	HyperText Murkup Language – мова розмітки гіпертекстових документів;
ОС	–	операційна система;
ПЕОМ	–	персональна електронно-обчислювальна машина;
КС	–	комп’ютерна система;
ПЗ	–	програмне забезпечення;
ПК	–	персональний комп’ютер;
ЦО	–	цивільна оборона

ВСТУП

Актуальність теми. Карти пам'яті в пристроях з операційної системи Android, розширюють її можливості в зберіганні інформації до рівня комп'ютерів 10-літньої давнини. Тоді стандартний вінчестер був обсягом 20-40 Гб, що практично відповідає обмеженню в 32 Гб на смартфонах.

На таку карту пам'яті може вміститися порядку 5 тисяч музичних композицій, 30 фільмів, а кількість фотографій і документів не піддається вирахуванню. Управляти цим багатством можна файловим менеджером з комп'ютера, підключивши мобільний пристрій через USB-порт або вставивши карту пам'яті в картрідер. Але більш практично мати файловий менеджер на смартфоні.

Крім стандартних дій з файлами (копіювання, переміщення, видалення) файлові менеджери для ОС Android допомагають підключатися до локальної мережі й працювати з нею, створювати архіви, переглядати зображення, мають убудовані утиліти для аналізу зайнятого місця й керування процесорами й багато чого іншого.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

– Дослідження системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

– Програмна реалізація системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система, яка розробляється у бакалаврському проекті призначена для керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android**.** Основне призначення системи це файловий менеджер для смартфону. На даний час кожна фірма розробник мобільних пристроїв розробляють та впроваджують своє ПЗ для роботи з мобільним пристроєм, що приводить до деякого хаосу, а саме: плутання стандартів; неможливості використання ПЗ з різними версіями; відторгнення деяких мобільних пристроїв. Інструментарій. Платформа легко пристосовується для використання VGA, 2D графічних бібліотек, 3D графічних бібліотек розроблених на основі OpenGL ES 1.0 специфікації, традиційних інструментаріїв для смартфонів. Бази даних: SQLite для структурованих даних. Технології зв'язку: Android підтримує багато технологій, що забезпечують зв'язок, включаючи: GSM, Bluetooth, EDGE, 3G/4G/5G та WiFi. Обмін повідомленнями: для обміну повідомленнями доступні як SMS, так і MMS сервіси, включаючи і потокові повідомлення. Веб браузери: на Android доступний веб-браузер розроблений на основі WebKit application framework. Java_Virtual_Machine: програми написані на Java можна скомпілювати в Dalvik байткод і виконувати на Dalvik virtual machine, яка являє собою розроблену спеціально для використання на мобільних пристроях VM, не зважаючи на те, що не є стандартною Java Virtual Machine. Підтримка медіа: Android підтримує такі формати для аудіо/відео даних та зображень: MPEG-4, H.264, MP3, та AAC, AMR, JPG, PNG, GIF. Підтримка нестандартного обладнання: Android підтримує відеокамери, фотоапарати, touchscreens, GPS, компаси, accelerometers, та прискорювачі 3D графіки. Середовище розробки: включає емулятор, засоби відлагодження, профілювання пам'яті та швидкодії.

1.2 Область застосування

Областю застосування системи є ОС Android. Мабуть, на сьогоднішній день це найбільш популярна операційна система. А все тому, що інші в основному використовуються на досить дорогих смартфонах, Android ж захопив і ринок відносно дешевих телефони. Багато моделей з сенсорним екраном працюють під цією системою. Можна сказати – Android заполонив магазини. Цю настільки популярну систему почала розробляти компанія Android Inc, але потім цю компанію купив Google, так що далі її розвиток йшов під його наглядом. Була створена ціла асоціація компаній, що займається Android-ом. Вона і зараз регулярно випускає нові версії та оновлення. У кожній версії поступово усувалися знайдені помилки і вносилися різні поліпшення, так що на сьогоднішній день ми маємо досить зручну і функціональну операційну систему.

Варто зауважити, що Android веде своє походження від системи Linux, відомої своєю надійністю і стабільною роботою. Але, також, відомо, що під Android вже існує не одна сотня програм – вірусів, втім, як і для інших систем.

Операційну систему Android можна встановити не тільки на телефон – її успішно застосовували на інтернет-планшетах, фоторамках і навіть справжніх комп'ютерах. Вміють люди можуть встановити її на різні пристрої – це називається, поміняти прошивку, тобто стерти стару програму і записати Android. Деякі виробники мобільних пристроїв стали забороняти таку можливість або хоча б ускладнювати.

Є співтовариство ентузіастів, які самостійно ведуть розробку системи Android або вносять зміни в офіційні версії. Так, наприклад, Google, як власник, може відстежувати положення будь-якого власника телефону на основі цієї системи за допомогою вбудованого GPS – приймача, а якщо його вимкнути – обчислювати по відношенню до вишок станцій. Одне це вже призводило до судових розглядів.

Але, з 2009 року, під впливом таких ось ентузіастів, що випускають альтернативні версії, які вільні від нав’язаних Google сервісів, відбулися зміни. Тепер користувач може сам вибирати, потрібні йому ці послуги чи ні. Якщо потрібні – він може легко встановити все бажане.

Так що система ця вже позбулася багатьох своїх недоліків, можна використовувати її сміливо і в задоволення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Проведемо огляд файлових менеджерів для планшетів і смартфонів на Android.

ES File Explorer (ES Провідник)

Мабуть, найвідомішим і популярним файловим менеджером для Android можливо назвати саме ES File Explorer. ES Провідник знайшов всенародну популярність завдяки тому, що це був найперший сторонній файловий менеджер, випущений для цієї ОС. Він з'явився в Google Play ще коли магазин додатків називався Android Market, а на всіх смартфонах стояв Android 1.6. Візуально ES File Explorer виконаний досить мінімалістично, але зі смаком. Споконвічно папки розташовані у вигляді сітки 4x4. Звичайно, можливо змінити сітку, наприклад, на 5x5, або ж включити режим списку з різним ступенем подробиць. Затиснення папки або файлу включає режим виділення, після цього можливо відзначити галочкою потрібні файли. Потім можливо вибрати одну з дій: копіювати, перемістити, видалити або перейменувати. Також можливо переміщати файли й папки втримуючи на них палець. У першому випадку з'явиться додатковий буфер обміну, а в другому – 4 опції по кутах екрана: відправити, видалити копіювати або поділитися. ES Провідник без проблем справляється з архівами. При натисканні на заархівірований об'єкт, він відкриється як звичайна папка з файлами, а далі ви вже самі зможете повністю або точково його розархівувати. Звичайно, можливо переглядати відео, слухати музику й любуватися фотографіями прямо з ES File Explorer. Якщо ж у вас встановлені root-права, то у вас з'являться додаткові можливості по видаленню системних папок і файлів.

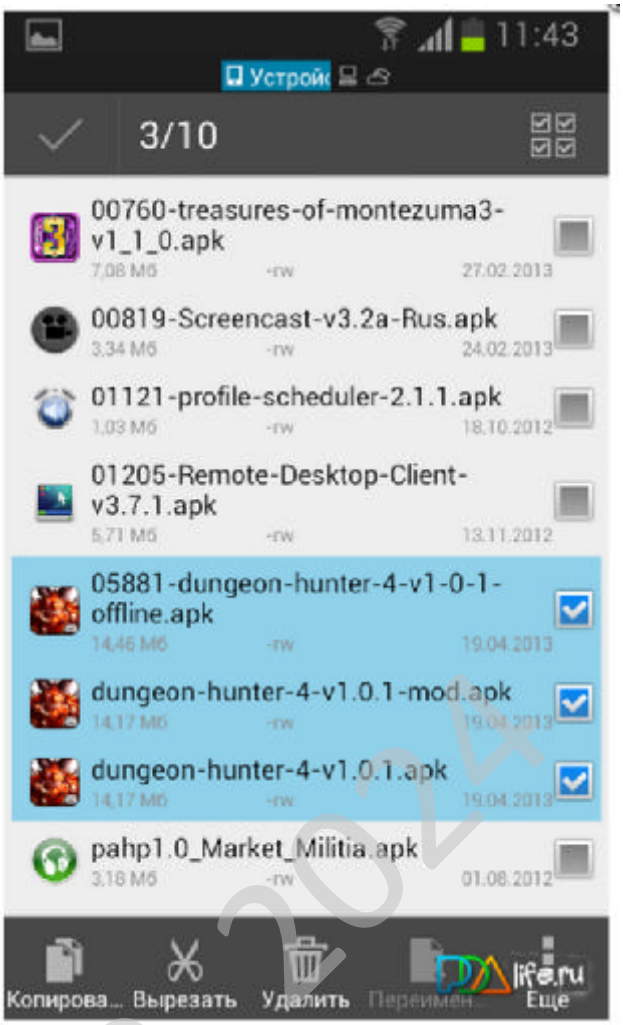

Рисунок 2.1 – Інтерфейс користувача ES File Explorer

FX File Explorer

Цей файловий менеджер зроблений дуже зручно – при відкритті кореневої папки SD-карти ви побачите ваші відеозаписи, музику, завантаження, документи й інші медіафайли. Це досить зручно – вам не потрібно зайвий раз блукати по папках у пошуках того або іншого файлу, всі відразу можливо знайти. Програма може працювати з багатьма видами архівів: ZIP, Tar, GZIP, BZIP2. Крім простого відкриття можливо й стискати файли в ці формати. Для RAR архівів підтримується тільки розпакування. Щоб виділити потрібний файл, потрібно просто свайпнути (провести пальцем по екрані) з право уліво. Після виділення потрібних файлів і затиснення кожного з них, відкриється додаткове меню з

вибором дій. Можливо також відкривати відразу кілька вікон, а перемикання між ними здійснюється за допомогою натискання вкладки Windows.

Total Commander

Популярний на ПК файловий менеджер знайшов любов користувачів Android-пристроїв, а все завдяки простоті й зручності програми. Так, зовнішній вигляд додатка залишає бажати кращого, але зате всі потрібні вкладки відразу ж винесені на головну сторінку, що дуже зручно. У горизонтальному положенні Total Commander ділиться на два стовпці, також такий зовнішній вигляд можливо включити й при портретній орієнтації.

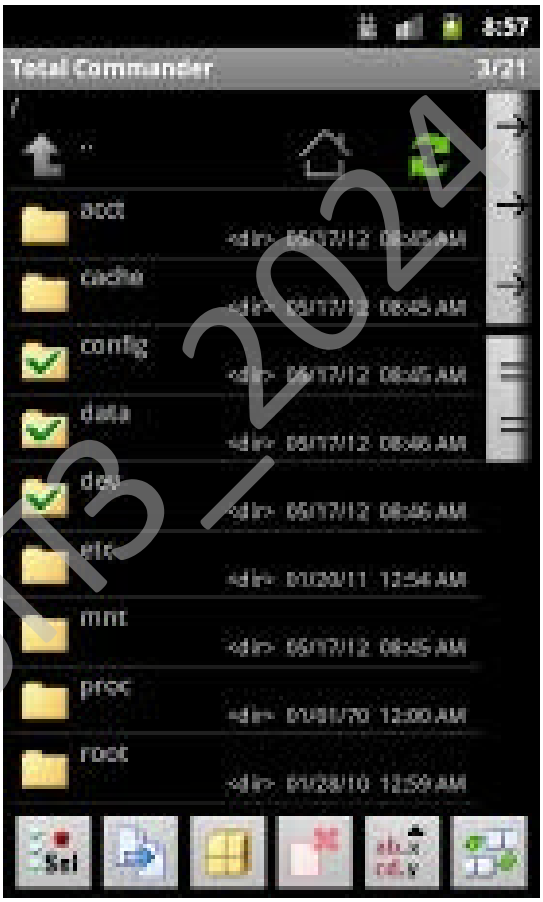

Рисунок 2.2 – Інтерфейс користувача Total Commander

Щоб виділити об'єкт, потрібно просто нажати на іконку файлу. Переміщати об'єкти не важко буде – можливо просто перетягати їхнім пальцем (Drag'n'Drop). У цьому випадку дуже зручним є режим із двома стовпцями.

Також є стандартний режим переміщення: затискаємо файл, вибираємо потрібну дію й вставляємо в будь-якому місці завдяки опції «Вставити тут».

Працює Total Commander тільки з ZIP-архівами.

Solid Explorer

Також досить непоганий файловий менеджер зі стильним, сучасним інтерфейсом і можливістю виділення файлів двома різними способами. Тут, як і в Total Commander можливо нажати на іконку (саме на неї) файлу, щоб виділити його, або скористатися класичним методом.

Також у ландшафтному режимі вікно Solid Explorer ділиться на два стовпці. У такому положенні дуже зручно переміщати потрібні об'єкти просто перетаскуючи їх.

Серед підтримуваних форматів архівів можливо виділити ZIP, RAR, TAR і RAR. Можливо як архівувати, так і розархівувати файли.

Astro File Meneger

Файловий менеджер ASTRO відрізняється не тільки якістю виконання, але й багатофункціональністю. З його допомогою можливо запускати, копіювати, переміщати й видаляти файли, він повноцінно працює з папками й з архівами форматів ZIP і TAR, є функція пошуку файлів і у власній пам'яті смартфону, і на карті пам'яті. Є власний менеджер процесів і засіб створення резервних копій установлених програм. Можливо також працювати з вилученими ресурсами й підключеними по Bluetooth і Wi-Fi пристроями (якщо встановити потрібні розширення). Найважливіші дії й операції винесені на верхню панель керування, що обертається за принципом каруселі й дозволяє швидко перейти до потрібної дії. Інтерфейс програми взагалі дуже зручний, але плюс до цього можливо встановити додаткові теми, які можуть краще гармоніювати з оболонкою Android.

Ще один інструмент керування – довгий тап, що традиційно викликає контекстне меню. Є убудована довідка, хоча навряд чи вона буде часто потрібна: всі основні дії зрозумілі інтуїтивно.

І при цьому ASTRO File Manager не вимагає рута. Це злегка урізує його можливості в порівнянні з деякими іншими менеджерами, але робить його більш безпечним і доступним для нерутованних пристроїв.

Крім всіх цих достоїнств, у програми є й мінуси – некритичні, але все-таки. Так, ASTRO не вміє переносити встановлені додатки на MicroSD, не індексує значки додатків, а об'ємні папки закриває довго. Плюс відсутність відновлень, через якого всі ці недоліки навряд чи будуть виправлені.

File Expert

File Expert, як видно з назви, позиціонується як один з наймогутніших файлових менеджерів для Android, у якому, крім роботи з файлами, безліч інших функцій. Головне для багатьох користувачів – те, що програма відмінно справляється зі своїми основними функціями. Копіювання й вставка, переміщення, копіювання, перейменування й видалення файлів і папок, запуск файлів і установка додатків – все це є в арсеналі «Експерта». Однак більше просунуті користувачі оцінять можливість розпаковувати й архіви, і настановні файли APK і змінювати або копіювати їхній уміст.

Ще одна приємна особливість цього файлового менеджера – просунута робота з мережними ресурсами. Можливо не тільки підключатися до зовнішніх мережних ресурсів (приміром, FTP), але й включати свій пристрій у локальні мережі (для цього передбачений убудований клієнт SMB), а також працювати із загальнодоступними хмарними сервісами, популярними останнім часом. Це, зокрема, Dropbox, Google Drive, Box.net, SkyDrive, а також менш популярні в нас азіатські SugarSync, Baidu Cloud і Sina Vdisk.

У програми простий інтерфейс, заплутатися тут практично ніде. Якщо ви любите копатися й освоювати програми, вас це може й віджахнути, а якщо додаток потрібен вам для реальної роботи, то це тільки плюс. Є убудований пошук файлів і папок, утиліта для перегляду текстових файлів, можливість упорядковувати файли за різними параметрами – типу, розміру, даті створення. Так, можливо завдяки функції впорядкування ввійти в пункт, присвячений

пакетам APK, і побачити всі настановні файли на вашім пристрої, де б вони не зберігалися. Цей підхід запозичений у програм-медіатек і зроблений більше універсальним.

Рисунок 2.3 – Інтерфейс користувача File Expert

Та ж функція дозволяє швидко перейти до файлів будь-якого типу – відеофайлам, аудіофайлам, зображенням, електронним книгам і текстовим документам. Настроювання програми дозволяють вам настроїти роботу з буфером обміну, включити або відключити показ схованих папок, показ мініатюр для зображень, настроїти режим (список або мозаїка). Що важливо для багатьох просунутих користувачів, програма вміє працювати з рутованими пристроями. Це дозволяє додатку працювати із системними папками, створювати бекап установлених додатків, проводити тиху установку й видалення додатків, редагувати файли конфігурації й робити багато чого іншого.

На жаль, як і багато інших файлових менеджерів, File Expert пригальмовує при роботі з об'ємними папками. Однак це єдиний серйозний мінус.

AndroZip File Manager

Ще один просунутий файловий менеджер, що вражає достатком доступних функцій. Зрозуміло, копіювання, переміщення, перейменування й видалення файлів і папок, а також пошук по файловій системі – першочергові функції для кожного файл-менеджера, і AndroZip справляється з ними відмінно. Інша можливість – відправляти файли на задану поштову адресу, створювати бекапи встановлених додатків у вигляді настановних пакетів, що працює навіть із платними додатками: купивши програму один раз, можливо відновити її навіть на іншому пристрої або аккаунті.

Убудований архіватор і диспетчер завдань ще більше розширюють функціональність пристрою. На відміну від інших додатків, алгоритми AndroZip дозволяють ефективно працювати навіть із файлами великого розміру, зокрема, з кешем ігор (часто такі файли мають розмір понад гігабайт і містять безліч дрібних файлів, що робить їхнє розпакування непростим завданням навіть для ПК). Є й підтримка захищених архівів.

Простий інтерфейс доступний повністю російською мовою, так що розібратися в ньому буде елементарно просто навіть для починаючих власників Android. Підтримується не тільки контекстне меню, але й переміщення й копіювання файлів за допомогою drag'n'drop (що для сенсорних екранів природно).

І що приємно, програма повністю безкоштовна в повнофункціональній версії. Якщо врахувати цей фактор, то говорити про якісь недоліки (крім недоліку розкручування) стосовно до AndroZip не доводиться.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного забезпечення мною було використане середовище Eclipse. Eclipse – вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C,C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP.

Eclipse являє собою фреймворк для розробки модульних кросплатформових застосунків із низкою особливостей:

– можливість розробки ПЗ на багатьох мовах програмування (рідною є Java);

– крос-платформова;

– модульна, призначена для подальшого розширення незалежним розробниками;

– з відкритим сирцевим кодом;

– розробляється і підтримується фондом Eclipse, куди входять такі постачальники ПЗ, як IBM, Oracle, Borland.

Спочатку проект розроблявся в IBM як корпоративний стандарт IDE, настановлений на розробки на багатьох мовах під платформи IBM. Потім проект було перейменовано на Eclipse і надано для подальшого розвитку спільноті.

Eclipse насамперед повноцінна Java IDE, націлена на групову розробку, має засоби роботи з системами контролю версій (підтримка CVS входить у поставку Eclipse, активно розвиваються кілька варіантів SVN модулів, існує

підтримкаVSS та інших). З огляду на безкоштовність, у багатьох організаціях Eclipse – корпоративний стандарт для розробки ПЗ на Java.

Друге призначення Eclipse – служити платформою для нових розширень. Такими стали C/C++ Development Tools (CDT), розроблювані інженерами QNX разом із IBM, засоби для підтримки інших мов різних розробників. Безліч розширень доповнює Eclipse менеджерами для роботи з базами даних, серверами застосунків та інших.

З версії 3.0 Eclipse став не монолітною IDE, яка підтримує розширення, а набором розширень. У основі лежать фреймворки OSGi, і SWT/JFace, на основі яких розроблений наступний шар – платформа і засоби розробки повноцінних клієнтських застосунків RCP (Rich Client Platform). Платформа RCP є базою для розробки різних RCP програм як торент-клієнт Azareus чи File Arranger. Наступний шар – платформа Eclipse, що є набором розширень RCP – редактори, панелі, перспективи, модуль CVS і модуль Java Development Tools (JDT).

Eclipse написана на Java, тому є платформо-незалежним продуктом, крім бібліотеки графічного інтерфейсу SWT, яка розробляється окремо для більшості поширених платформ. Бібліотека SWT використовує графічні засоби платформи (ОС), що забезпечує швидкість і звичний зовнішній вигляд інтерфейсу користувача.

Відповідно до IDC, із Eclipse працюють 2,3 мільйона розробників.

Основою Eclipse є платформа розширеного клієнта. Її складають такі компоненти:

– ядро платформи (завантаження Eclipse, запуск модулів);

– OSGi (стандартне середовище постачання комплектів);

– SWT (стандартний інструментарій віджетів);

– JFace (файлові буфери, робота з текстом, текстові редактори);

– робоче середовище Eclipse (панелі, редактори, проекції, майстри).

GUI в Eclipse написаний з використанням інструментарію SWT. Останній, на відміну від Swing (який лише емулює окремі графічні елементи

використовуваної платформи), дійсно використовує графічні компоненти даної системи. Призначений для користувача інтерфейс Eclipse також залежить від проміжного шару GUI, званого JFace, який спрощує побудову призначеного для користувача інтерфейсу, що базується на SWT.

Гнучкість Eclipse забезпечується за рахунок модулів, що підключаються, завдяки чому можлива розробка не тільки на Java, але і на інших мовах, таких як C/C++, Perl, Groovy, Ruby, Python, PHP, ErLang та інших.

Програми для Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik.

Google пропонує для вільного завантаження інструментарій для розробки (Software Development Kit), який призначений для x86-машин під операційними системами Linux, Mac OS X (10.4.8 або вище), Windows XP, Windows Vista таWindows 7. Для розробки потрібен Java Development Kit 5 або новіший.

Розробку додатків для Android можливо вести мовою Java (не нижче Java 1.5). Існує плагін для Eclipse – «Android Development Tools» (ADT), призначений для Eclipse версій 3.3-3.7. Для IntelliJ IDEA також існує плагін, який полегшує розробку Android-додатків. Для середовища розробки NetBeans розроблено плагін, який починаючи з версії Netbeans 7.0 перестав бути експериментальним, проте поки не є офіційним. Крім того існує Motodev Studio for Android, що являє собою комплексне середовище розробки, засноване на базі Eclipse і дозволяє працювати безпосередньо з Google SDK.

Крім того в 2009 році на додаток до ADT був опублікований Android Native Development Kit (NDK), пакет інструментаріїв і бібліотек дозволяє вести розробку додатків на мові C/C++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості.

Доступні бібліотеки:

– Bionic (Бібліотека стандартних функцій, несумісна з libc);

– libc (стандартна системна бібліотека мови Сі);

– мультимедійні бібліотеки (на базі PacketVideo OpenCORE; підтримують такі формати, як MPEG-4, H.264, MP3, AAC, AMR, JPEG та PNG);

– SGL (рушій двовимірної графіки);

– OpenGL ES 1.0 (рушій тривимірної графіки);

– Surface Manager (забезпечує для додатків доступ до 2D/3D);

– WebKit (готовий рушій для Web-браузера; обробляє HTML, JavaScript);

– FreeType (рушій обробки шрифтів);

– SQLite (проста система керування базами даних, доступна для всіх застосувань);

– SSL (протокол, що забезпечує безпечну передачу даних по мережі).

В порівнянні із звичайними додатками Linux, додатки Android підкоряються додатковим правилам:

– Content Providers – обмін даними між додатками.

– Resource Manager – доступ до таких ресурсів, як файли XML, PNG, JPEG.

– Notification Manager – доступ до рядка стану.

– Activity Manager – управління активними додатками.

Для Android був розроблений формат інсталяційних пакетів .apk.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для реалізації системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android використовується компонент Activity.

Activity – це компонент застосунку, який відповідає за екран, з яким користувачі можуть взаємодіяти для того, щоб виконати певні дії – набрати телефонний номер, прийняти фото, відправити email чи переглянути карту. Кожному activity відповідає вікно, в якому реалізовано його власний користувацький інтерфейс. Вікно, як правило, заповнює весь екран, проте воно може бути й меншим і знаходитись поверх інших вікон.

Застосунок зазвичай складається з багатьох activities, які вільно зв’язані між собою. Як правило, один activity в додатку визначається як «головний» – він відображається для користувача при першому запуску застосунку. Будь-який activity може запустити інший activity з метою виконання різних дій. Кожного разу при запуску нового activity, попередній activity зупиняється, проте система зберігає його в стек. Коли новий activity запускається, він поміщається в стек і приймає фокус користувача. Такий зворотній стек подібний до загального стекового механізму – «останнім увійшов – першим вийшов», тому, коли користувач закінчив роботу із поточним activity і натиснув кнопку «Назад», він видаляється зі стеку (і знищується), а робота попереднього activity поновлюється.

При зупинці activity через запуск нового activity, він повідомляє про цю зміну стану через методи зворотного виклику життєвого циклу activity. Існує кілька методів зворотного виклику, які activity може отримати, у зв'язку зі зміною його стану – чи то його створення системою, зупинка, відновлення, або знищення – і кожний зворотний виклик надає вам можливість виконувати певну

роботу, що відповідає цій зміни стану. Наприклад, при зупинці, ваш activity повинен звільнитися від всіх великих об'єктів, таких як мережеві з'єднання або з'єднання з базою даних. При відновленні activity можливо задіяти необхідні ресурси і відновити дії, які були перервані. Ці переходи станів є частинами життєвого циклу activity.

Далі в цьому розділі ми розглянемо основи створення і використання activity, а також обговоримо як працює життєвий цикл activity, так що ви зможете правильно організувати перехід між різними станами activity.

Створення Activity

Щоб створити activity, необхідно створити підклас Activity (або використати вже існуючий його підклас). У вашому підкласі вам необхідно реалізувати методи зворотного виклику, які система буде викликати при переході між різними станами життєвого циклу activity – при створенні, зупинці, відновленні чи знищенні. Двома найбільш відомими методами зворотного виклику є:

– onCreate() – необхідно реалізувати цей метод. Під час створення вашого activity система викликає цей метод. В межах вашої реалізації необхідно ініціалізувати необхідні компоненти вашого activity. Найголовніше, це саме той метод, де необхідно викликати setContentView() для визначення макета користувацького інтерфейсу в activity.

– onPause() – виклик цього методу системою – перша ознака того, що користувач залишає ваш activity (хоча це не завжди означає, що activity знищується). Це зазвичай необхідно, коли ви маєте зафіксувати деякі зміни, які повинні бути збережені поза поточної сесії користувача (оскільки користувач може не повернутися).

Існують і інші методи життєвого циклу зворотного виклику, які ви маєте використовувати для того, щоб забезпечити зручну роботу користувача під час переходів між activities і управлінням несподіваними неочікуваними перериваннями, які приводять до того, що ваш activity може бути зупиненим чи

навіть знищеним. Всі методи зворотного виклику життєвого циклу будуть розглянуті далі.

Реалізація користувацького інтерфейсу

Користувацький інтерфейс для activity реалізується за допомогою ієрархії подань – об'єктів, похідних від класу View. Кожне подання контролює окремий прямокутний простір у вікні activity і може реагувати на дії користувача. Наприклад, поданням може бути кнопка, яка ініціює дію, коли користувач торкається її.

Android надає ряд готових видів подання, які можна використовувати для проектування та організації відображення. Віджети – подання, які забезпечують візуальні (і інтерактивні) елементи для екрану, такі як кнопки, текстові поля, поля з вибором чи просто зображення. "Layouts" – подання, отримані з ViewGroup, які надають унікальний макет моделі для своїх дочірніх переглядів, таких як лінійні, сіткові чи відносі. Ви також можете успадковувати подання від класів View та ViewGroup у підкласах для створення власних віджетів та макетів і застосовувати їх для вашого макету activity.

Найпоширеніший спосіб визначити макет за допомогою подання – з макета XML-файлу, збереженого у ресурсах вашого застосунку. Таким чином, можливо зберегти дизайн інтерфейсу користувача окремо від вихідного коду, який визначає поведінку activity. Можливо встановити макет, як користувальницький інтерфейс для вашого activity з setContentView(), передавши ідентифікатор ресурсу макету. Крім цього, ви також можете створювати нові подання Views в коді вашого activity і будувати ієрархію подань шляхом додавання нових представлень Views в ViewGroup, а потім використовувати цей макет, передавши кореневий ViewGroup в setContentView().

Додаткові відомості про створення користувацького інтерфейсу див. в документації User Interface.

Оголошення activity в маніфест-файлі

Необхідно оголосити ваш activity в декларативному файлі-маніфесті для того, щоб він став доступним в системі. Для оголошення вашого activity відкрийте маніфест-файл і додайте елемент <activity> в якості дочірнього елементу <application>. Наприклад:

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

Існує також кілька інших атрибутів, які можливо включити в цей елемент для того, щоб визначити властивості, такі як мітка для activity, його іконка чи тема для стилю користувацького інтерфейсу. Атрибут android:name є єдиним обов'язковим атрибутом – ім'я класу activity. Після публікації вашого застосунку не потрібно змінювати це ім'я, тому що у випадку, якщо ви це зробите, можливо порушити деяку функціональність, а саме ярлики застосунку (більш детально можна ознайомитись в пості блогу Things That Cannot Change).

Для отримання більш детальної інформації про оголошення вашого activity в маніфест-файлі можливо переглянути посилання елементу <activity>.

Використання intent-фільтрів

Елемент <activity> також може містити різні intent-фільтри, використовуючи елемент < intent-filter>, в порядку оголошення того, як інші компоненти застосунку можуть активувати його.

При створенні нового застосунку, використовуючи інструменти Android SDK, «корінь» activity, який створюється автоматично, включає intent-фільтр, який вказує, що activity відповідає «головній» дії і має бути розміщений в категорії «запуску». intent-фільтр має наступний вигляд:

```
      <activity android:name=".ExampleActivity"
android:icon="@drawable/app_icon">
          < intent-filter>
              <action android:name="android. intent.action.MAIN" />
```

```
            <category android:name="android. intent.category.LAUNCHER" />
        </ intent-filter>
    </activity>
```

Елемент <action> вказує, що це є «головна» точка входу в застосунок. Елемент <category> вказує, що цей activity повинен бути вказаний в застосунку пускової системи (щоб дозволити користувачам запускати цей activity).

Якщо ви хочете, щоб ваш застосунок був замкненим і ви не хочете, щоб інші програми мали можливість активувати activity вашого застосунку, то вам не потрібні будь-які інші intent-фільтри. Тільки один activity повинний мати "головні" дій і категорії «запуску», як і в попередньому прикладі. Якщо ви не хочете, щоб ваші Activities були доступними для інших застосунків, вони повинні мати ніяких intent-фільтрів і можливо запустити їх самостійно, використовуючи явні способи (як це описано в наступному розділі).

Проте, якщо ви хочете, щоб ваш activity відповідав неявним intent-фільтрам, які передаються з інших застосунків (і з вашого власного), то необхідно визначити додаткові intent-фільтри для вашого activity.

Для кожного типу intent, якому вм хочете відповідати, потрібно включити < intent-filter> який містить елемент <action> і, за потреби, елемент <category> і/або елемент <data> . Ці елементи визначають тип intent, до якого ваш activity може відноситися.

Для отримання додаткових відомостей про те, як ваші activities можуть реагувати на intent, див. Документ intent s and intent Filters.

Запуск Activity

Можливо запустити інший activity, викликавши startActivity(), передавши його intent, що описує activity, який ви хочете запустити. intent конкретизує або точний activity, який ви хочете запустити, або, описує тип дії, яку ви хочете виконувати (і система вибирає відповідний activity для вас, який може навіть бути з іншого застосунку). intent може також нести невеликі обсяги даних, які будуть використовуватися в activity, що запускається.

Працюючи в межах вашого власного застосунку, вам часто буде необхідно просто запустити вже відомий activity. Ви це можете зробити, створюючи intent, який явно визначає activity, який ви хочете запустити, користуючись класовим ім'ям. Наприклад, тут – те, як один activity запускає інший activity, який має ім’я SignInActivity:

```
 intent intent = new intent (this, SignInActivity.class);
startActivity(intent );
```

Проте, вашому застосунку, можливо, також знадобилося б виконати деякі дії, такі, як відправити електронну пошту, текстові повідомлення, або оновлення статусу, використовуючи дані з вашого activity. В цьому випадку, ваш застосунок, можливо, не мав би свого власного activity, щоб виконувати такі дії, замість цього можливо посилити activity, забезпеченого іншими застосунками, на пристрій, який може виконувати дії для вас. Це є дійсно цінним при створенні вашого власного intent, який описує дії, які необхідно виконати, і система запустить відповідний activity з іншої програми. Якщо є кілька activities, які можуть впоратися з intent, то користувач може вибрати, які з них використовувати. Наприклад, якщо ви хочете дозволити користувачам відправляти повідомлення електронної пошти, можливо створити наступний intent :

```
 intent intent = new intent (intent.ACTION_SEND);
 intent.putExtra(intent.EXTRA_EMAIL, recipientArray);
startActivity(intent );
```

EXTRA_EMAIL, додатково доданий до intent – це масив рядків, який складається з адреси електронної пошти, до якої необхідно надсилати повідомлення електронної пошти. Коли програма електронної пошти реагує на цей intent, він зчитує рядки масиву, передбачених в extra, і поміщає їх у полі "Кому"("to") листа форми композиції. У цій ситуації activity програми електронної пошти стартує і по вказівці користувача ваша activity відновлюється.

Старт Activity для результату

Інколи, ви хотіли б отримати результат від activity, який ви запускаєте. У такому разі, запустіть activity, викликаючи startActivityForResult() (замість

startActivity()). Для отримання результату від такого activity реалізуйте метод зворотного виклику onActivityResult(). Коли activity відпрацює, він поверне результат в intent до вашого методу onActivityResult().

Наприклад, можливо ви хочете, щоб користувач вибрав один зі своїх контактів, так щоб ваш activity зміг щось зробити з інформацією, що знаходиться в контакті. Далі наведено приклад створення такого intent і обробки результату:

```
        private void pickContact() {
            // Створює intent «взяти» contact, як це визначено в контенті URI
постачальника
            intent intent = new intent (intent.ACTION_PICK, Contacts.CONTENT_URI);
            startActivityForResult(intent, PICK_CONTACT_REQUEST);
        }
        @Override
        protected void onActivityResult(int requestCode, int resultCode, intent
data) {
            // Якщо запит (OK) і запит був PICK_CONTACT_REQUEST
            if (resultCode == Activity.RESULT_OK && requestCode ==
PICK_CONTACT_REQUEST) {
                // Виконає запит контенту провайдера контакту для імені контакту
                Cursor cursor = getContentResolver().query(data.getData(),
                new String[] {Contacts.DISPLAY_NAME}, null, null, null);
                if (cursor.moveToFirst()) {
                    int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
                    String name = cursor.getString(columnIndex);
                    // Зробить що-небудь з ім'ям обраного контакту ...
                }
            }
        }
```

Цей приклад показує основну логіку, яку необхідно використовувати в вашому методі onActivityResult() для обробки результату activity. Перша умова перевіряє, чи був запит успішним – якщо так, то resultCode буде RESULT_OK – і чи запит, якому цей результат відповідає, відомий – в даному випадку requestCode відповідає другому параметру, надісланому з startActivityForResult(). Звідти код управляє результатом activity шляхом запиту даних, що повертаються в intent (параметр data).

Відбувається наступне: ContentResolver виконує запит до контент-провайдера, який повертає Cursor, який дозволяє запросити дані для читання. Для отримання додаткової інформації див документ Content Providers.

Завершення Activity

Для завершення Activity необхідно викликати його метод finish(). Ви також можете завершити окремі Activity, які ви створили раніше, шляхом виклику finishActivity().

Зауваження: У більшості випадків ви не повинні явно завершувати activity за допомогою використання цих методів. Як буде відмічено в наступному розділі про життєвий цикл, система Android обробляє життєвий цикл activity так, що вам не потрібно власноруч завершувати activity. Виклик цих методів може негативно вплинути на звичний досвід користувачів і його слід використовувати тільки у випадку, коли ви абсолютно не хочете, щоб користувач міг повернутися до цього зразку activity.

Керування життєвим циклом Activity

Для розробки сильного і гнучкого застосунку вирішальне значення має керування життєвим циклом вашого activity шляхом опрацювання методів зворотного виклику (тобто подій). На життєвий цикл activity безпосередньо впливає його зв'язок з іншими activity, його задача і зворотний стек activity.

Activity може існувати в основному в трьох станах:

– Resumed – Activity знаходиться на передньому плані екрану і приймає фокус користувача. (Цей стан також іноді називають "running").

– Paused – Інший activity знаходиться на передньому плані і має фокус, але перший activity все ще видимий. Тобто інший activity знаходиться над першим activity, при цьому він частково прозорий або не займає екран повністю. Activity, що був призупинений, повністю живий (об'єкт Activity зберігається в пам'яті, цей процес зберігає всі стани і елементи інформації, і залишається прикріпленим до віконного менеджера), але може бути знищений системою в ситуації нестачі пам'яті.

– Stopped – Activity повністю закритий іншим activity (activity зараз знаходиться в "фоні"). Зупинений activity все ще живий (об'єкт activity зберігається в пам'яті, він підтримує всі стани та інформацію, але вже не є прикріпленим до диспетчера вікон). Однак, він вже не відображається користувачу і може бути знищений системою, коли пам'ять буде необхідна в іншому місці.

Якщо activity був призупинений або зупинений, система може видалити його з пам'яті, роблячи запит на його завершення (викликаючи його метод finish()), або просто знищує його процес. Під час повторного відкриття activity (після завершення або знищення), його слід створити заново.

Обробка подій життєвого циклу activity

Коли activity переходить в і з різних станів, описаних вище, про це буде повідомлятися за допомогою різних методів зворотного виклику. Всі методи зворотного виклику – це обробники подій, їх можна перевизначити для виконання відповідної роботи, коли стан activity змінюється. Наступна схема activity включає в себе кожний з основних методів життєвого циклу:

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // activity створюється
    }
    @Override
    protected void onStart() {
        super.onStart();
        // activity стає видимою.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity стала видимою (в даний момент вона "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
```

```
        // Інша activity бере фокус (ця activity ось-ось буде "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        //activity більше не видима (в даний момент вона "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // activity зараз буде знищена.
    }
}
```

Перед виконанням будь-яких дій в цих методах необхідно завжди викликати відповідний метод суперкласу, як це показано в прикладах вище.

Взяті разом, ці методи визначають весь життєвий цикл activity. Реалізуючи ці методи, можливо контролювати три вкладені цикли в життєвому циклі activity:

Повна тривалість життя activity трапляється між викликом onCreate() і викликом onDestroy(). Ваш activity повинен виконувати установку "глобального" стану (як, наприклад, визначення макета) в onCreate(), і звільняти всі ресурси, що залишилась, в onDestroy(). Наприклад, якщо ваш activity працює у фоновому режимі, для завантаження даних з мережі, він може створити цей потік в onCreate(), а потім зупинити потік в onDestroy().

Видима тривалість життя activity відбувається між викликом onStart() і викликом onStop(). Впродовж цього часу користувач може бачити activity на екрані та взаємодіяти з ним. Наприклад, onStop() викликається, коли новий activity запущений и попередній більше не відображається. Між цими двома методами можливо зберігати ресурси, які потрібні для того, щоб показати activity користувачу. Наприклад, можливо зареєструвати BroadcastReceiver в onStart(), щоб контролювати зміни, що впливають на ваш користувацький інтерфейс, і скасувати його реєстрацію в onStop(), коли користувач більше не може бачити те, що ви показуєте. Система може викликати onStart() та onStop() кілька разів протягом всього життєвого циклу activity, оскільки activity почергово є видимим та прихованим для користувача.

Пріоритетна тривалість життя activity відбувається між зверненням до onResume() і викликом onPause(). Впродовж цього часу activity – поверх всіх інших activities на екрані і має фокус вводу користувача. Activity може часто переходити поза і на передній план, наприклад, onPause() викликається, коли пристрій засинає або коли з'являється діалог. Оскільки цей стан може переходити часто, код у цих двох методах повинен бути достатньо легким, щоб уникнути повільних переходів, які змушують користувача чекати.

Збереження стану Activity

Введення в управління життєвим циклом (Managing the Activity Lifecycle) activity відмічає, що коли activity призупинено або зупинено, стан activity зберігається. Це вірно, оскільки об'єкт activity, коли його призупинено або зупинено, ще тримається в пам'яті – вся інформація про його членів і поточний стан все ще жива. Таким чином, всі внесені користувачем зміни в рамках activity зберігаються так, що коли activity повертається на передній план (коли вона "resumes "), ці зміни все в силі.

Однак, коли система знищує activity з метою відновлення пам'яті, об'єкт activity знищено, тому система просто не може відновити його з його стану без змін. Замість цього, система повинні відновити об'єкт activity, якщо користувач переходить на нього. Тим не менш користувач не знає, чи система знищила activity і відтворила його і, таким чином, ймовірно очікує, що activity буде точно таким, як і був. У цій ситуації, можна гарантувати, що важлива інформація про стан activity зберігається шляхом впровадження додаткового методу зворотного виклику, що дозволяє зберігати інформацію про стан вашого activity: onSaveInstanceState().

Система викликає onSaveInstanceState() перед виконанням activity, вразливим до знищення. Система передає цей метод Bundle, в якому можливо зберегти інформацію про стан activity в якості пар ім'я-значення, використовуючи такі методи, як putString() і putInt(). Тоді, якщо система знищує ваші застосунки процесу і користувач переходить назад на ваш activity, система відновлює свою

діяльність і передає Bundle до onCreate() і до onRestoreInstanceState(). Використовуючи будь-який з цих методів можливо вилучити збережені стани від розшарування (Bundle) і відновлення стану activity. Якщо немає інформації для відновлення, то розшарування (Bundle) не відбулося (як у випадку, коли activity створюється вперше).

Немає гарантії, що onSaveInstanceState() буде викликатися перед знищенням вашого activity, тому що є випадки, коли не потрібно буде зберігати стан (наприклад, коли користувач залишає ваш activity, використовуючи кнопку Назад, (явне закриття activity)). Якщо система викликає onSaveInstanceState(), вона робить це до onStop() і, можливо, до onPause().

Однак, навіть якщо ви нічого не робите і не виконуєте onSaveInstanceState(), деякі стани activity відновлюються Activity класом реалізацією onSaveInstanceState() за замовчуванням. Зокрема, реалізація за замовчуванням викликає відповідний метод onSaveInstanceState() для кожного подання View в макет, який дозволяє кожному поданню надавати інформацію про себе, яка повинна бути збережена. Майже кожен віджет у рамках Android реалізує цей метод, у разі необхідності, так, що будь-які видимі зміни, внесені до інтерфейсу користувача, автоматично зберігаються і відновлюються, коли ваш activity перебудований. Наприклад, віджет EditText зберігає будь-який текст, введений користувачем, і EditText віджет зберігає відомості, чи цей текст перевіряється, чи ні. Єдина робота, яка від вас вимагається – надання унікального Ідентифікатора (android:id атрибуту) для кожного елемента керування, для якого ви хочете зберегти його стан. Якщо віджет не має Ідентифікатора, то системі не вдасться зберегти його стан.

Хоча реалізація onSaveInstanceState() за замовчуванням зберігає корисну інформацію про інтерфейси вашого activity, вам ще, можливо, буде потрібно змінити його, щоб зберегти додаткову інформацію. Наприклад, вам може знадобитися збереження значення елементів, які змінилися за час життя activity (який може корелювати зі значення відновлення в користувацькому інтерфейсі,

але члени, які містять значення цього користувальницького інтерфейсу не відновлюються, за замовчуванням).

Оскільки реалізація onSaveInstanceState() за замовчуванням допомагає зберегти стан користувацького інтерфейсу, якщо ви перевизначите метод для того, щоб зберегти додаткову інформацію про стан, необхідно завжди викликати реалізації суперкласу onSaveInstanceState(), перш ніж робити будь-яку роботу. Крім того, ви також повинні викликати реалізації суперкласу onRestoreInstanceState() якщо ви заміните його так, що реалізація за замовчуванням може відновити подання станів.

Оскільки не гарантується, що onSaveInstanceState() буде викликано, необхідно використовувати його тільки для запису перехідного стану активності (стан користувацького інтерфейсу) – ви ніколи не повинні використовувати його для збереження постійних даних. Замість цього необхідно використовувати onPause() для збереження постійних даних (наприклад, дані, які повинні бути збережені в базі даних), коли користувач залишає activity.

Хороший спосіб перевірки здібності вашої програми щодо відновлення свого стану – просто повернути пристрій так, щоб змінилася орієнтація екрану. При зміні орієнтації екрану система знищує і перебудовує activity в цілях застосування альтернативних ресурсів, які можуть бути доступні для нової конфігурації екрану. Тільки з цієї причини дуже важливо, щоб ваш activity повністю відновлював свій стан, коли він перебудовується, тому що користувачі регулярно повертають екран при використанні застосунків.

Обробка змін конфігурації

Деякі конфігурації пристроїв можуть змінюватися під час виконання (наприклад, орієнтації екрану, придатність клавіатури та мови). Коли відбувається така зміна, Android відтворює працюючу activity (система викликає onDestroy(), потім відразу ж викликає onCreate()). Така поведінка спланована для того, щоб допомогти вашому застосунку адаптуватися до нової конфігурації шляхом автоматичного перезавантаження застосуваня з альтернативними

ресурсами, які ви надали (наприклад, різні макети для різних орієнтацій екрану і розмірів). Якщо ви правильно спроектувати ваш activity, щоб впоратися з перезавантаженням через зміни орієнтації екрану і відновлення стану activity, як описано вище, ваш застосунок буде більш стійкими до інших несподіваних подій в життєвому циклі activity.

Кращий спосіб справитися з таким перезавантаженням – збереження і відновлення стану вашої activity з використанням onSaveInstanceState() і onRestoreInstanceState() (або onCreate()), як описано в попередньому розділі.

Для отримання додаткової інформації про конфігурацію змін, які відбуваються під час виконання і як можливо впоратися з ними, читайте керівництво Handling Runtime Changes.

Координація activities

Коли один activity заміщує інший, вони обидва мають життєвий цикл переходів. Перший activity призупиняється і припиняється (хоча, вон ще не зупинений, якщо він досі видимий у фоновому режимі), в той час як інший activity буде створений. У разі, якщо ці activities мають спільний доступ до даних, збережених на диску або в іншому місці, то важливо розуміти, що перший activity повністю не зупинився перед тим, як другий буде створюватися. Швидше за все, процес запуску другого activity збігається з процесом зупинки першого. Порядок життєвого циклу зворотного виклику визначено коректно, особливо, коли ці два activities в одному процесі і один починає інший. Ось порядок операцій, які відбуваються, коли Activity A запускає Acivity B:

onPause() метод Activity A у виконанні.

onCreate(), onStart(), і onResume() методи Activity B виконуються в послідовності. (Activity B тепер має фокус користувача).

Тоді, коли Activity A більше не видимий на екрані, його onStop() виконується.

Ця передбачувана послідовність життєвого циклу зворотного виклику дозволяє управляти переходом інформації з одного виду activity на інший.

Наприклад, якщо потрібно зробити зміни до бази даних, коли перший activity зупиняється так, що наступний activity може прочитати ці зміни, то необхідно записувати зміни в базу даних протягом onPause() замість того, щоб роботи це під час onStop().

3.2 Розробка структурної схеми

Структурна схема системи представлена на рисунку 3.1. Розроблена система структурно складається з наступних блоків:

1. Структурний блок розробленого програмного забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android:

– Інтерфейс.

– Пошук файлів та папок.

– Сортування файлів.

– Створення бекапу.

– Основний диск файлової підсистеми.

– Карта пам'яті формату SD.

– Перегляд архівних даних.

– Налаштування.

– Довідкова.

2. Структурний блок мультимедіа даних:

– Відео wmv, avi.

– Зображення jpg, bmp.

– Текст doc, rtf.

– Анімація gif.

– Аудіо mp3, wav.

3. Структурний блок підключення додаткових модулів.

4. Структурний блок оновлення розробленого ПЗ.

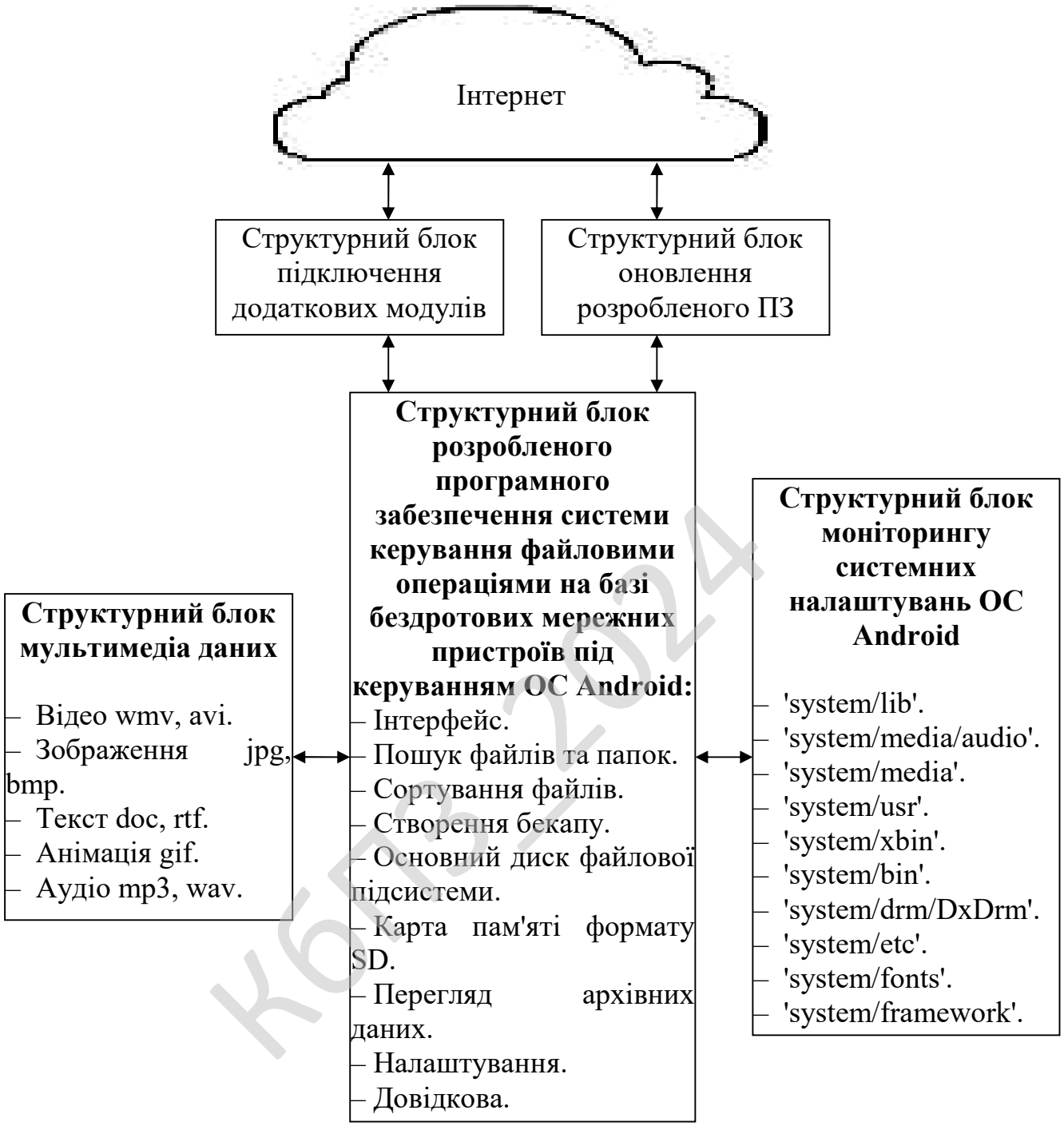

Рисунок 3.1 – Структурна схема системи

5. Глобальна мережа Інтернет.

6. Структурний блок моніторингу системних налаштувань ОС Android:

– 'system/lib'.

– 'system/media/audio'.

– 'system/media'.

– 'system/usr'.

– 'system/xbin'.

– 'system/bin'.

– 'system/drm/DxDrm'.

– 'system/etc'.

– 'system/fonts'.

– 'system/framework'.

Система реалізована з використанням компоненту Service.

Робота з Service

Service – це компонент застосунку, який може виконувати тривалі операції у фоновому режимі і не передбачає інтерфейсу користувача. Інший компонент застосунку може запустити service і він продовжить працювати у фоновому режимі, навіть якщо користувач переключиться на інший застосунок. Крім того, компонент можна прив’язати до service для взаємодії з ним і навіть для здійснення міжпроцесної взаємодії (IPC). Наприклад, service може обробляти мережеві операції, відтворювати музику, виконувати операції з файлами чи взаємодіяти з контент-провайдером – і все це в фоновому режимі.

Service, по суті, може мати дві форми:

– Started (запущений) – service є “started”, коли компонент застосунку (такий як activity) запускає його через виклик startService(). Після запуску service може працювати у фоновому режимі невизначено довго, навіть якщо компонент, який запустив його, був знищений. Зазвичай started service виконує одну операцію і не повертає результат виклику. Наприклад, він може завантажити чи вивантажити файл через мережу. Після завершення операції service повинен зупинити себе.

– Bound (прив’язаний) – service є “bound”, коли компонент застосунку прив’язується до нього через виклик bindService(). BoundService() пропонує клієнт-серверний інтерфейс, який дозволяє компонентам взаємодіяти з service,

відсилати запити, отримувати результати і навіть робити це через міжпроцесну взаємодію (IPC). BoundService() працює довго лише якщо інший компонент застосунку прив'язаний до нього. Декілька компонентів можуть бути привязані до service відразу, проте коли всі вони будуть відокремлені, service буде знищений.

Хоча в цій документації ці два типи service sобговорюються окремо, ваш service може працювати в обох напрямках – він може бути started (для запуску на невизначений час), а також дозволяти прив'язку. Це лише залежить від того, як ви виконали поєднання методів зворотного виклику, а саме onStartCommand() для дозволу компонентам запустити його та onBind() для дозволу прив'язки.

Незалежно від того яким є ваш застосунок – started, boundчи і тим і іншим – будь-який компонент застосунку може використовувати service (навіть з окремого застосунку) таким же чином, як і будь-який компонент може використовувати activity, – запускаючи його через intent. Однак у декларативному файлі можливо оголосити service в якості приватного і заблокувати доступ до нього з інших застосунків.

Основи

Для створення service необхідно створити підклас service (або використати один із уже існуючих його підкласів). У вашій реалізації вам потрібно перевизначити деякі методи зворотного виклику, які обробляють ключові аспекти життєвого циклу service та забезпечують механізм прив'язки компонентів до service у випадку такої необхідності. Найбільш важливими методами зворотного виклику, які слід перевизначити, є:

– onStartCommand() – система викликає цей метод, коли інший компонент, такий як, наприклад, activity, надсилає запит на запуск service через викликstartService(). Як тільки цей метод виконується, service запускається і може працювати у фоновому режимі невизначено довго.Якщо ви перевизначите цей метод, вашим обов'язком стане зупинення service, коли його робота буде

завершена, викликом stopSelf() чи stopService(). (Якщо ви хочете тільки забезпечити прив’язку, то вам не потрібно виконувати цей метод).

– onBind() – система викликає цей метод, коли ще один компонент необхідно прив’язати до service (наприклад, виконання RPC) через викликbindService(). Під час вашої реалізації цього методу необхідно забезпечити інтерфейс, який клієнти використовуватимуть для взаємодії /підтримки зв’язку/ з service через повернення IBinder. Ви завжди повинні використовувати цей метод, проте якщо ви не хочете дозволяти прив’язку, то можливо повернути /значення/ null.

– onCreate() – система викликає цей метод при першому створенні service для виконання одноразової установки процедур (перед тим, як будуть викликані onStartCommand() чи onBind()). Якщо service вже запущений, то цей метод не викликається.

– onDestroy() – система викликає цей метод, коли service більше не використовуєтьсяі має бути знищений. Ваш service повинен виконати цей метод для очищення будь-яких ресурсів, таких як потоки, зареєстровані слухачі, приймачі і т.і. Він є останнім, що отримує service.

Якщо компонент запускає service викликом startService() (який приводить до виклику onStartCommand()), тоді service залишається запущеним доти, доки він не зупинить себе через stopSelf() або інший компонент не зупинить його викликом stopService().

Якщо компонент викликає bindService() для створення service (і onStartCommand() не викликаний), тоді service працює лише до тих пір, доки компонент прив’язаний до нього. Як тільки service відокремитьсявід усіх клієнтів, система знищить його.

Android-система зупинить service тільки при малому об’ємі пам’яті /тоді, коли пам'ять буде вичерпана /при нестачі пам’яті/, і це повинно відновити системні ресурси для activity, на якому сфокусований користувач. Якщо service прив’язаний до activity, на якому сфокусована увага користувача, тоді він,

скоріше за все, буде знищений, і навпаки – якщо service заявлений runintheforeground (буде розглянуто /обговорено/ далі), тоді він майже ніколи не буде знищений. В іншому випадку, якщо service буде запущений тривалий час, тоді через деякий час система знизить його позицію у списку фонових завдань і service стане близьким до знищення – якщо ваш service запущений, тоді ви мусите сконструювати його для коректної обробки перезапусків системи Якщо система знищує ваш service, то вона запускає його як тільки ресурси стають знову доступними (хоча це також залежить від того, яке значення ви повертаєте з onStartCommand(), що буде розглянуто далі /обговорено пізніше/). Для отримання додаткової інформації про те, коли система може знищити service, перегляньте документ Processes and Threading.

У наступних розділах ви побачите як можна створити кожен тип service і як використовувати його з інших компонентів застосунку.

Оголошення service в маніфесті

Подібно activities(та іншим компонентам) ви мусите оголосити всі service у вашому manifest-файлі (декларативному файлі)застосунку.

Щоб оголосити ваш service додайте елемент < service > як дочірній /наслідуємий/ від елементу <application>. Наприклад:

```
<manifest ... >
  ...
  <application ... >
      < service android:name=".Example service " />
      ...
  </application>
</manifest>
```

Є й інші атрибути, які можливо включити до елементу < service > для визначення /зміни/ властивостей, таких як дозволи, необхідні для запуску service, і процесу, в якому service повинен запуститися. Атрибут android:name є єдиним обов’язковим атрибутом – він визначає ім’я класу service. Після того, як ви опублікували ваш застосунок, необхідно не змінювати це ім’я, тому що, якщо ви це зробите, то можливо порушити деякий функціонал /деяку функціональність/,

де явні /explicit/ intent s використовуються для посилання на ваш service (читайте пост блогу ThingsThatCannotChange).

Так само як і activity, service може визначати intent-фільтри, які дозволяють іншим компонентам викликати service, використовуючи неявні /implicit/ intent s. Оголосивши intent-фільтри, компоненти з інших застосунків, встановлених на пристрої користувача, потенційно можуть запустити ваш service, якщо він оголошує intent-фільтр, який співпадає з intent інших застосунків, що передаються до startService().

Якщо ви плануєте використовувати ваш service лише локально (інші застосунки не можуть використовувати його), тоді вам не потрібно (і ви не повинні)організовувати підтримку інших intent-фільтрів. Без інших intent-фільтрів ви мусите запускати service, використовуючи intent, який явно називає клас service. Більше інформації про startinga service можна буде дізнатися нижче.

Крім того можливо встановити, що ваш service є private/приватним/до вашого застосунку, але лише якщо ви включите атрибутandroid:exported і встановите його значення "false". Це ефективно навіть якщо ваш service підтримує /містить/ intent-фільтри.

Створення Started service

Started service – це перше, що інший компонент запускає викликом startService(), що є результатом виклику методу onStartCommand(), який належить service.

Коли service запущений, його життєвий цикл залежить від компонента, який запустив його, і service може бути запущений у фоновому режимі на невизначений час навіть якщо компонент, який викликав його, був знищений. Так, наприклад, service має зупинити себе викликом stopSelf(), коли його робота буде виконана, або інший компонент може зупинити його викликом stopService().

Компонент застосунку, такий як activity, може запустити service викликом startService() і передати intent, який обумовлює service і включає всі

дані для використання в service. service приймає цей /значення цього/ intent в методі onStartCommand().

Для прикладу припустимо, що activity потрібно зберегти деякі дані в онлайн-базі даних. Activityможе запустити companion service і передати дані для збереження передачею intent в startService(). service отримує intent в onStartCommand(), підключається до інтернету і виконує транзакції бази даних. Коли транзакція буде виконана, service зупинить себе і буде знищений.

Традиційно існує два класи, які можливо розширити для створення started service:

– Service – він є базовим класом для всіх service s. Коли ви розширите цей клас, важливо, щоб ви створили новий потік /thread/, в якому виконуватиметься вся робота service, тому що, за замовчуванням, service використовує головний потік вашого застосунку, а це може уповільнити виконання інших activity, запущених вашим застосунком.

– intent service – він є підкласом service, який використовує робочий потік для обробки всіх запущених запитів по одному. Він є найкращим вибором, якщо ви не потребуєте, щоб ваш service обробляв декілька запитів одночасно. Все, що вам потрібно зробити – це виконати onHandle intent (), який отримує intent для кожного запущеного запиту, то ж ви зможете організувати роботу в фоновому режимі.

У наступних розділах описується можливість реалізації вашого service, використовуючи один із цих класів.

Розширення класу intent service

Через те, що більшості запущених service s не потрібно обробляти декілька запитів одночасно (що насправді є небезпечним багатопотоковим сценарієм), найімовірніше краще всього буде, якщо ви виконаєте ваш service, використовуючи клас intent service.

Intent service виконує наступне:

– Створює робочий потік за замовчуванням, який виконує всі intent s, що передаються до onStartCommand() окремо від головного потоку вашого застосунку.

– Створює робочу чергу, яка періодично передає один intent для виконання вашого onHandle intent (), тому вам ніколи не потрібно турбуватись про багатопоточність.

– Зупиняє service після того, як всі запущені запити були оброблені, то ж вам ніколи не треба викликати stopSelf().

– За замовчуванням забезпечує виконання onBind(), що повертає /значення/ null.

– Забезпечує виконання за замовчуванням onStartCommand(), що посилає intent до робочої черги, а потім до implementation вашогоonHandle intent ().

Все це означає, що все, що вам потрібно зробити – це виконати onHandle intent () для виконання роботи, наданої клієнтом. (Хоча вам також потрібно передбачити невеликий конструктор для service).

Ось приклад реалізації intent service:

```
public class Hello intent service extends intent service {

public Hello intent Service() {
super("Hello intent service ");
 }
@Override
protected void onHandle intent (intent intent ) {
    long endTime = System.currentTimeMillis() + 5*1000;
    while (System.currentTimeMillis() < endTime) {
      synchronized (this) {
        try {
             wait(endTime - System.currentTimeMillis());
        } catch (Exception e) {
        }
      }
    }
  }
}
```

Це все, що вам потрібно: конструктор і реалізація onHandle intent ().

Якщо ви вирішили також перевизначити і інші методи зворотного виклику, такі як onCreate(), onStartCommand() чи onDestroy() впевніться у виклику super implementation, то ж intent service може належним чином обробити життя робочого потоку.

Наприклад, onStartCommand() повинен повернути implementation за замовчуванням (which is how the intent gets delivered to onHandle intent ()):

```
@Override
public int onStartCommand(intent intent, int flags, int startId) {
    Toast.makeText(this, " service starting", Toast.LENGTH_SHORT).show();
    return super.onStartCommand(intent,flags,startId);
}
```

Окрім onHandle intent (), єдиним методом, з яким вам не потрібно викликати super class є onBind() (проте вам потрібно реалізувати його лише, якщо ваш service дозволяє прив'язку).

У наступному розділі ви побачите як подібний вид service реалізується, коли розширяється базовий клас service, код якого є значно більшим, проте який може бути доречним, якщо вам потрібно обробляти одночасно запущені запити.

Розширення класу service

Як ви бачили у попередньому розділі, використання intent service робить вашу реалізацію started service дуже простою. Однак, якщо ви потребуєте, щоб ваш service підтримував багатопоточність (замість обробки запущених запитів через робочу чергу), тоді можливо розширити клас service для обробки кожного intent.

Для порівняння, наступний приклад коду є реалізацією класу service, який виконує точно таку ж роботу як і вищенаведений приклад використання intent service. Тобто для кожного запуску запиту він використовує робочий потік для виконання роботи і обробляє лише один запит одночасно.

```
public class Hello service extends service {
  private Looper m service Looper;
  private service Handler m service Handler;
  private final class service Handler extends Handler {
      public service Handler(Looper looper) {
          super(looper);
```

```
        }
        @Override
        public void handleMessage(Message msg) {
            long endTime = System.currentTimeMillis() + 5*1000;
            while (System.currentTimeMillis() < endTime) {
                synchronized (this) {
                    try {
                        wait(endTime - System.currentTimeMillis());
                    } catch (Exception e) {
                    }
                }
            }
            stopSelf(msg.arg1);
        }
    }
    @Override
    public void onCreate() {
      HandlerThread thread = new HandlerThread(" service StartArguments",
              Process.THREAD_PRIORITY_BACKGROUND);
      thread.start();
m service Looper = thread.getLooper();
m service Handler = new service Handler(m service Looper);
}
@Override
public int onStartCommand(intent intent, int flags, int startId) {
Toast.makeText(this, " service starting", Toast.LENGTH_SHORT).show();
Message msg = m service Handler.obtainMessage();
msg.arg1 = startId;
m service Handler.sendMessage(msg);
return START_STICKY;
}
@Override
public IBinder onBind(intent intent ) {
return null;
}
@Override
public void onDestroy() {
Toast.makeText(this, " service done", Toast.LENGTH_SHORT).show();
}
}
```

Як можливо бачити, тут набагато більше роботи, ніж при використанні intent service.

Однак, оскільки ви обробляєте кожен виклик onStartCommand() самостійно, можливо виконувати декілька запитів одночасно. Це не те, що виконує даний приклад, проте якщо це те, що ви хочете, тоді можливо створити новий потік для кожного запиту і запускати їх одразу (замість очікування завершення попереднього запиту).

Зверніть увагу, що метод onStartCommand() повинен повертати integer. Integer – це значення, яке описує як системі слід поновити service у випадку, коли система знищила його (як зазначалося вище, реалізація intent service за замовчуванням обробляє це для вас, хоча ви в змозі змінити це). Значення, що повертається з onStartCommand() мусить бути одним із наступних констант:

– START_NOT_STICKY – якщо система знищує service після відпрацювання onStartCommand(), не створюйте service повторно, хіба що є intent s, що очікують передачі. Це є найбезпечнішим варіантом для уникнення запуску вашого service, коли це не потрібно /не необхідно/ і коли ваш застосунок може просто повторно запустити всі незавершені роботи /завдання/.

– START_STICKY – якщо система знищує service після відпрацювання onStartCommand(), створіть service заново і викличіть onStartCommand(), але не повертайте останній intent. Тоді система викличе onStartCommand() з нульовим intent, якби тільки там були очікуючі intent s для запуску service, в цьому випадку ці intent sбудуть передані. Це підходить для медіа-плеєрів (чи подібних service s), які не виконують команди, проте запущені на невизначений час і чекають на завдання.

– START_REDELIVER_ INTENT – якщо система знищує service після того, як onStartCommand() повертається, створіть service заново і викличіть onStartCommand() з останнім intent, який було передано до service. Усі очікуючі intent будуть поставлені в чергу. Це підходить для service s, які активно

виконують роботу, яка негайно повинна бути відновлена, наприклад, завантаження файлу.

Для отримання більш детальної інформації стосовно цих значень, що повертаються, дивіться прикріплене посилання на документацію для кожної з констант.

Запуск service

Можливо запустити service з activityчи з іншого компонента застосунку через передачу intent (вказавши запуск service) до startService().Android-система викликає метод onStartCommand() в service і передає intent. (Ви ніколи не повинні безпосередньо викликатионStartCommand().) Наприклад, activity може запустити приклад service, що був розглянутий у попередній частині (HelloSevice), використовуючи явний /точний/ intent і з startService():

```
intent intent = new intent (this, Hello service.class);
start service (intent );
```

Метод startService() виконується безпосередньо, і Android-система викликає метод onStartCommand(), що належить service. Якщо service ще не запущений, система спочатку викликає onCreate(), а потім onStartCommand().

Якщо service також не забезпечує прив’язку, intent, що передається із startService() є єдиним способом зв’язку між компонентом застосунку і service. Однак, якщо ви хочете, щоб service відправляв результат назад, тоді клієнт, що запустив service, може створити Pending intent для broadcast (з getBroadcast()) і передавати його до service в intent, який запускає service. Потім service може використовувати broadcast для передання результату. Численні запити на запуск service приводять до численних відповідних викликів onStartCommand() в service. Однак, лише один запит зупиняє service при необхідності (разом із stopSelf() чи stopService()).

Зупинення service

Запущений service мусить керувати власним життєвим циклом. Тобто система не може зупинити чи знищити service, якщо не потрібно звільнити системну пам'ять, і service продовжує працювати після

виконання onStartCommand(). То ж service мусить зупинити себе викликом stopSelf(), чи інший компонент може зупинити його викликом stopService().

Як тільки отримано запит на зупинення разом із stopSelf() чи stopService() система знищує service як тільки це стає можливо.

Однак, якщо ваш service обробляє кілька запитів для onStartCommand() одночасно, тоді ви не можете зупинити service, коли ви запустили процес запиту, тому що після отримання можливо запустити новий запит (зупинка в кінці першого запиту призведе до другого). Для уникнення цієї проблеми можливо використати stopSelf(int) для того, щоб впевнитися в тому, що ваш запит на зупинку service завжди базується на самому останньому запущеному запиті. Тобто коли ви викликаєте stopSelf(int) ви передаєте ID запущеного запиту (startIdпередається до onStartCommand()), який відповідає вашому зупиненому запиту. Тоді, якщо service отримує новий запущений запит перед тим, як ви спроможні викликати stopSelf(int), тоді ID не будуть збігатися і service не буде зупинений.

Створення BoundService()

BoundService() є єдиним, що дозволяє компонентам застосунку прив’язатися до нього викликом bindService() для створення довготривалого з’єднання (і, як правило, не дозволяє компонентам запускати його викликом startService()).

Необхідно створювати boundService() , коли хочете взаємодіяти із service через activities та інщі компоненти у вашому застосунку або для розкриття деякого функціоналу вашого застосунку до інших застосунків через міжпроцесну взаємодію /зв'язок/ (IPC).

Для створення boundService() ви мусите реалізувати метод зворотного виклику onBind() для повернення IBinder, який визначає інтерфейс зв’язку з service. Інші компоненти застосунку можуть потім викликати bindService() для відновлення інтерфейсу і початку виклику методів у service. service існує лише для обслуговування компонентів застосунку, які зв’язані з ним, то ж коли немає

ніяких компонентів, зв’язаних з service, система знищує його (вам не потрібно зупиняти boundService() in the way – ви мусите зробити це, коли service запущений черезonStartCommand()).

Перше, що вам потрібно зробити для створення boundService() – це визначити інтерфейс, який вказує яким чином клієнт може зв’язатися з service. Цей інтерфейс між service та клієнтом мусить бути реалізацією IBinder і того, що ваш service мусить повертати з методу зворотного виклику onBind(). Як тільки клієнт отримує IBinder він може почати взаємодіяти з service через цей інтерфейс.

Декілька клієнтів можуть бути прив’язані до service відразу. Коли клієнт виконав взаємодію з service, він викликає unbindService() для від’єднання. Як тільки не стає клієнтів, прив’язаних до service, система його знищує.

Виведення оповіщень користувачу

Після запуску service може оповістити користувача про події, використовуючи ToastNotifications чи StatusBarNotifications.

Оповіщення – це повідомлення, яке з’являється на мить на поверхні поточного вікна, а потім зникає, тоді як оповіщення status bar’у передбачає іконку в status bar’і з повідомленням, яке користувач може вибрати для прийняття дії (наприклад, запуск activity).

Зазвичай оповіщення status bar’у є найкращим технічним прийомом, коли робота у фоновому режимі завершена (наприклад, завершення завантаження файлу) і користувач може now act on it. Коли користувач обирає оповіщення із розширеного view, оповіщення може запустити activity (наприклад, перегляд завантаженого файлу).

Запуск service на передньому плані

ForegroundService() – це service, про кий користувач активно обізнанийі отже не є кандидатом на знищення в системі при низькому об’ємі пам’яті. ForegroundService() мусить надавати оповіщення у status bar’і, який розташований під заголовком "Ongoing", що означає, що оповіщення не є проігнорованим, хіба що service є також зупиненим чи видаленим із foreground.

Наприклад, музичний плеєр, який програє музику із service має бути встановленим на роботу у foreground, тому що користувач явно обізнаний про цю операцію. Оповіщення в statusbar'і може показувати поточний трек і дозволяти користувачеві запускати activityдля взаємодії з музичним плеєром.

Щоб визначити чи ваш service запущений у foreground, викличіть startForeground(). Цей метод приймає два параметри: integer, який однозначно ідентифікує оповіщення і Notification для statusbar'а. Наприклад:

```
Notification notification = new Notification(R.drawable.icon,
getText(R.string.ticker_text),
        System.currentTimeMillis());
 intent notification intent = new intent (this, ExampleActivity.class);
Pending intent pending intent = Pending intent.getActivity(this, 0,
notification intent, 0);
notification.setLatestEventInfo(this, getText(R.string.notification_title),
        getText(R.string.notification_message), pending intent );
startForeground(ONGOING_NOTIFICATION, notification);
```

Для усунення service із foreground викличіть stopForeground(). Цей метод приймає boolean, вказуючи чи усувати оповіщення statusbar'у також. Цей метод не зупиняє service. Однак, якщо ви зупините service тоді як він все ще працює у foreground, тоді оповіщення також буде видалено.

Для отримання більш детальної інформації стосовно оповіщень перегляньте CreatingStatusBarNotifications.

Управління життєвим циклом service

Життєвий цикл service є дуже простим порівняно з activity. Однак він є навіть більш важливим, ніж оплата закриття уваги на те як ваш service був створений і знищений, тому що service може бути запущений у background'і без інформування користувача. Життєвий цикл service – з його створення до його знищення – може слідувати двома шляхами:

– Started service – service створюється, коли ще один /інший/ компонент викликає startService(). Тоді service працює невизначено довго /невизначений час/ і мусить зупинити себе через виклик stopSelf(). Інший компонент може також

зупинити service через виклик stopService(). Коли service зупинений, система знищує його.

– Bound Service – service створюється, коли ще один /інший/ компонент (клієнт) викликає bindService(). Тоді клієнт підтримує зв'язок з service через інтерфейс IBinder. Клієнт може закрити з'єднання через виклик unbindService(). Декілька клієнтів можуть прив'язатися таким же чином до service і коли всі з них будуть від'єданані, система знищить service.

Ці два шляхи не є цілком окремими. Тому можливо прив'язати до service, який вже був запущений з startService(). Наприклад, backgroundmusic service може бути запущений через виклик startService() з intent, який визначає музику для відтворення. Потім, можливо, коли користувач захоче здійснювати деякий контроль над плеєром чи отримувати інформацію про поточний трек, activity може прив'язатися до service через виклик bindService(). У подібних випадках stopService() чи stopSelf() не можуть дійсно зупинити service, доки всі клієнти відв'язані.

Реалізація життєвого циклу зворотних викликів

Подібно activity, service є життєвим циклом методів зворотного виклику, які можливо здійснювати для змін на екрані у стані service і виконувати роботу у відповідні часи. Наступна схема service демонструє кожен життєвий цикл методів:

```
public class Example service extends service {
    int mStartMode;
    IBinder mBinder;
    boolean mAllowRebind;
    @Override
    public void onCreate() {
    }
    @Override
    public int onStartCommand(intent intent, int flags, int startId) {
        return mStartMode;
    }
    @Override
    public IBinder onBind(intent intent ) {
```

```
        return mBinder;
    }
    @Override
    public boolean onUnbind(intent intent ) {
        return mAllowRebind;
    }
    @Override
    public void onRebind(intent intent ) {
    }
    @Override
    public void onDestroy() {
    }
}
```

Здійснюючи ці методи, можливо контролювати два вкладених цикли життєвого циклу service:

– Повна тривалість життя service відбувається в проміжку часу, коли onCreate() викликається і повертається onDestroy(). Як і activity, service виконує свої початкові налаштування під час onCreate() і звільняє всі ресурси, що залишилися, під час onDestroy(). Наприклад, service програвання музики може створити потік, в якому музика буде грати в onCreate(), а зупиняти потік – в onDestroy(). Методион Create() та onDestroy() викликаються для всіх service s, будь вони створені в startService() або bindService().

– Активна тривалість життя service починається з виклику або onStartCommand(), або onBind(). Кожному методу передається intent, який був переданий startService() або bindService() відповідно. Якщо service запущено, то активна тривалість життя закінчується одночасно з кінцем повної тривалості життя (service залишається активним навіть після повернення onStartCommand()). Якщо service прив'язаний, то активна тривалість життя завершується, коли onUnbind() повертається.

Незважаючи на те, що запущений service зупиняється викликом stopSelf() або stopService(),немає відповідного зворотного виклику для service (немає зворотного виклику onStop()). Тому поки service прив'язаний до клієнта, система знищує його, коли service зупинений – onDestroy() є єдиним отриманим зворотнім викликом.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Розроблене програмне забезпечення по керуванню файловими операціями для Android є додатком, що містить у собі файловий менеджер, менеджер додатків, завершення завдань, Dropbox клієнт, FTP-клієнт. Розроблене програмне забезпечення по керуванню файловими операціями дозволяє користувачам Android безкоштовно в будь-якій точці миру управляти своїми ресурсами, має простій у керуванні інтерфейс, працює за допомогою 3G, EDGE або Wi-Fi, дозволяє ділитися файлами із друзями, завантажувати фотографії й дивитися відео.

Список функцій розробленого програмного забезпечення по керуванню файловими операціями:

– Менеджер файлів (уміє копіювати, вставляти, вирізати/переміщати, створювати, видаляти, перейменовувати й відправляти файли) у телефоні й на комп'ютері.

– Множинний вибір файлів (Мультивибір).

– Диспетчер додатків (установка, видалення, резервне копіювання, сполучення клавіш, категорії).

– Стиск і розпакування файлів ZIP, RAR. Може створювати зашифровані AES (256 біт) ZIP-файли.

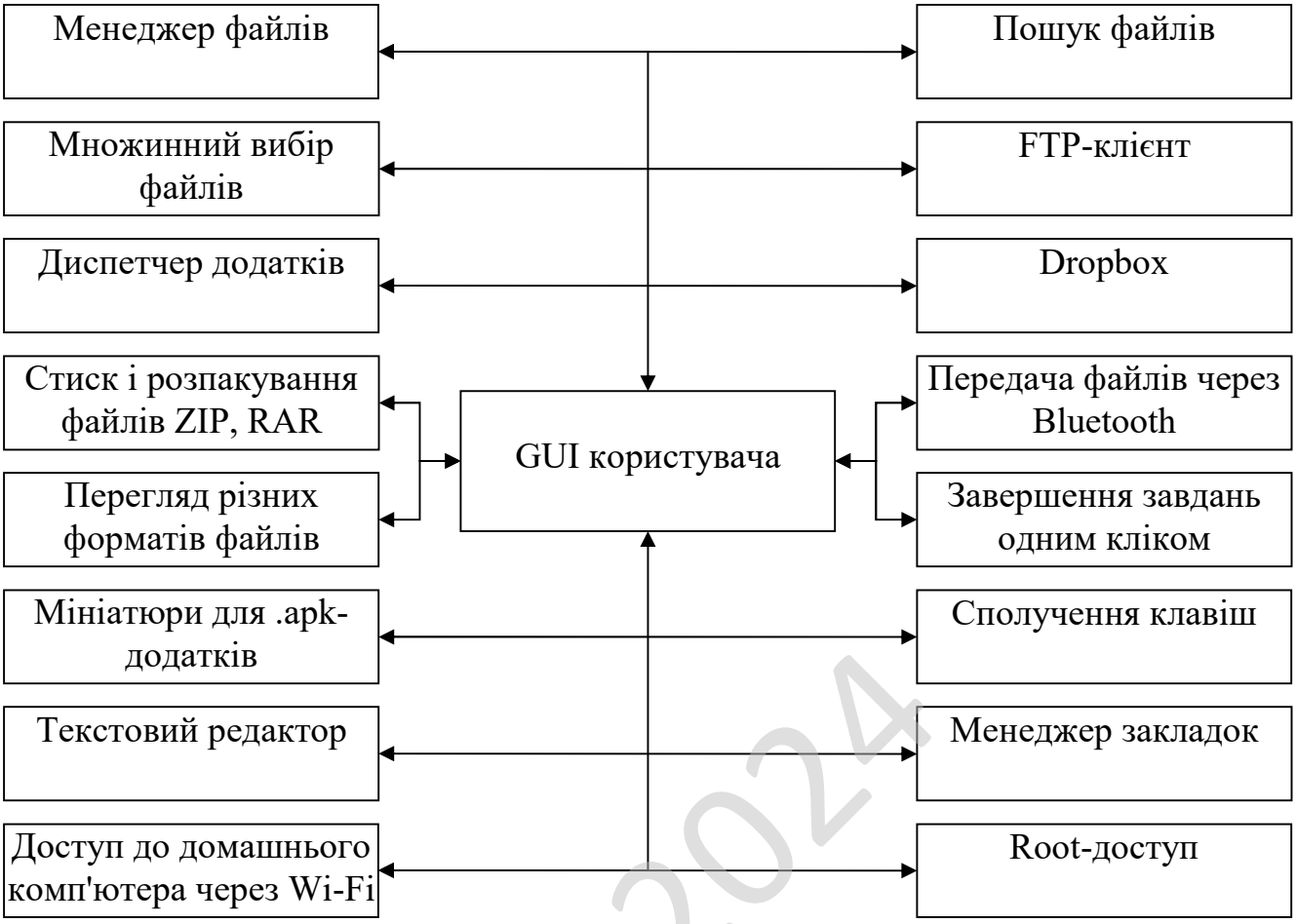

Рисунок 3.2 – Функціональна схема системи

– Перегляд різних форматів файлів, фотографій, документів, відео, підтримка сторонніх додатків, таких як Document To Go, для відкриття документів.

– Мініатюри для .apk-додатків, зображень і убудований "переглядач" зображень.

– Текстовий редактор.

– Пошук файлів.

– Доступ до домашнього комп'ютера через Wi-Fi з SMB (Server Message Block).

– FTP-клієнт дозволяє управляти файлами на FTP-сервері, так само, як на SD-карті.

– Dropbox – це розширення клієнта Dropbox до офіційної версії. Воно дозволяє зберігати фотографії й відео на Dropbox-аккаунте й обмінюватися фотографіями / документами, легко редагувати документи у вашій "коробці".

– Передача файлів через Bluetooth. Можливо копіювати й переміщати файли між сумісними Bluetooth-пристроями. Є підтримка OBEX FTP для перегляду й передачі файлів на Bluetooth-пристрої.

– Завершення завдань одним кліком. Є досить простий віджет для автоматичного завершення завдань, додайте важливі додатки в ігнор-список для того, щоб ці додатки ніколи не вбивалися. Це просунутий менеджер завдань, убивця запущених додатків і чистильник пам'яті.

– Сполучення клавіш.

– Менеджер закладок.

– Root-доступ. Це інструмент керування для просунутих користувачів. Можливо одержати повний доступ до всієї файлової системи й даних і міняти права доступу.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3.

Де після початку роботи проходе виведення головного вікна ПЗ звідки можна потрапити до налаштування ПЗ, моніторингу запитів ОС до файлової системи, перегляду статистики роботи ПЗ.

Крім цього через обробку запитів ОС та бібліотеку функцій ПЗ можливо провести створення бекапу, пошук файлів та папок, оновлення розробленого ПЗ, дії над файлами (видалення, редагування, копіювання, переміщення), моніторинг системних налаштувань ОС.

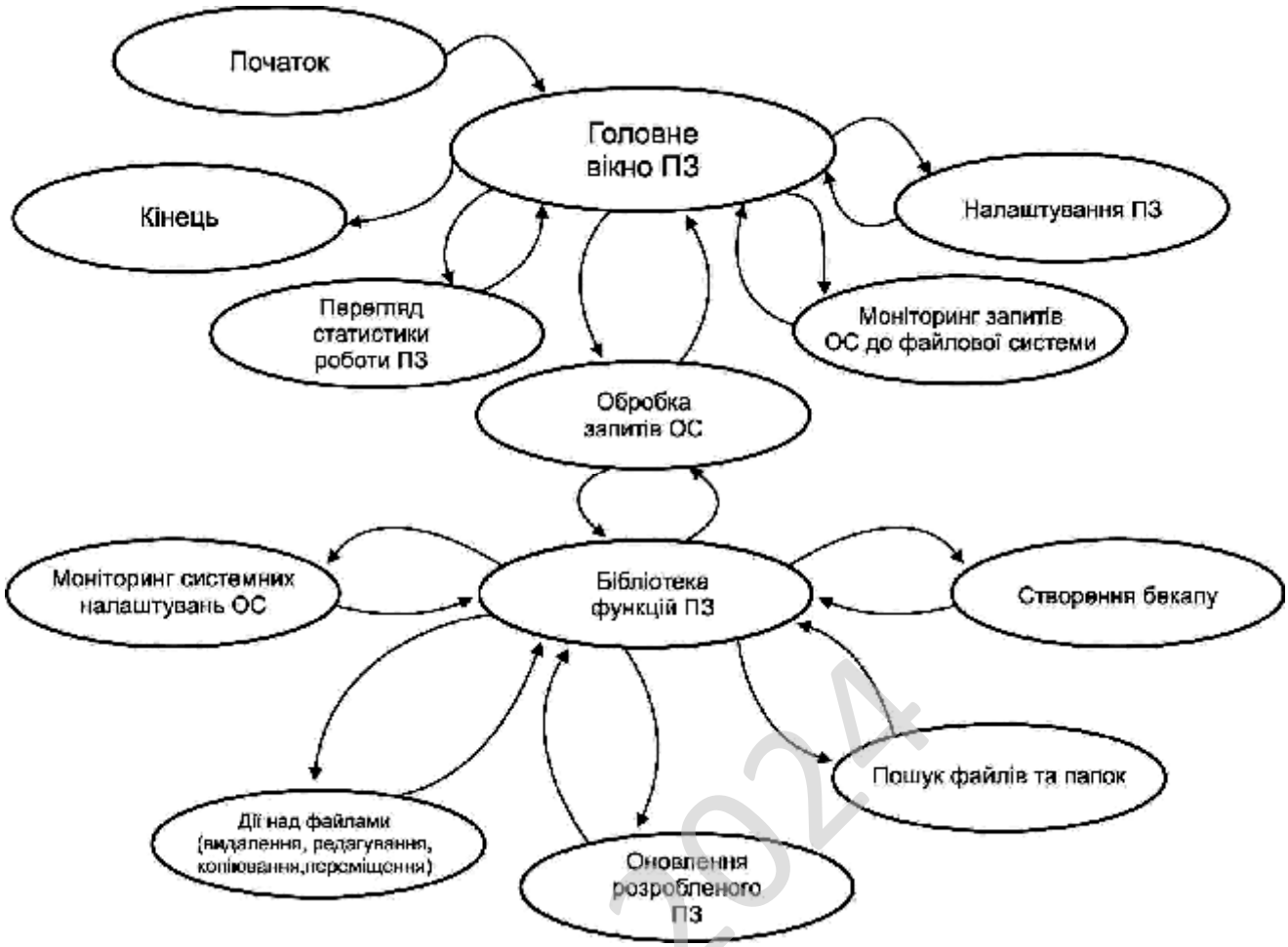

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Розглянемо детально блок-схему. Спершу відбувається створення стеку даних, налаштування обліку ПЗ відносно налаштувань ОС та перевірка наявності необхідної версії ОС. Якщо перевірка пройдена успішно проводиться виклик підпрограми моніторингу дискової підсистеми яка зображена на рисунку 4.2.

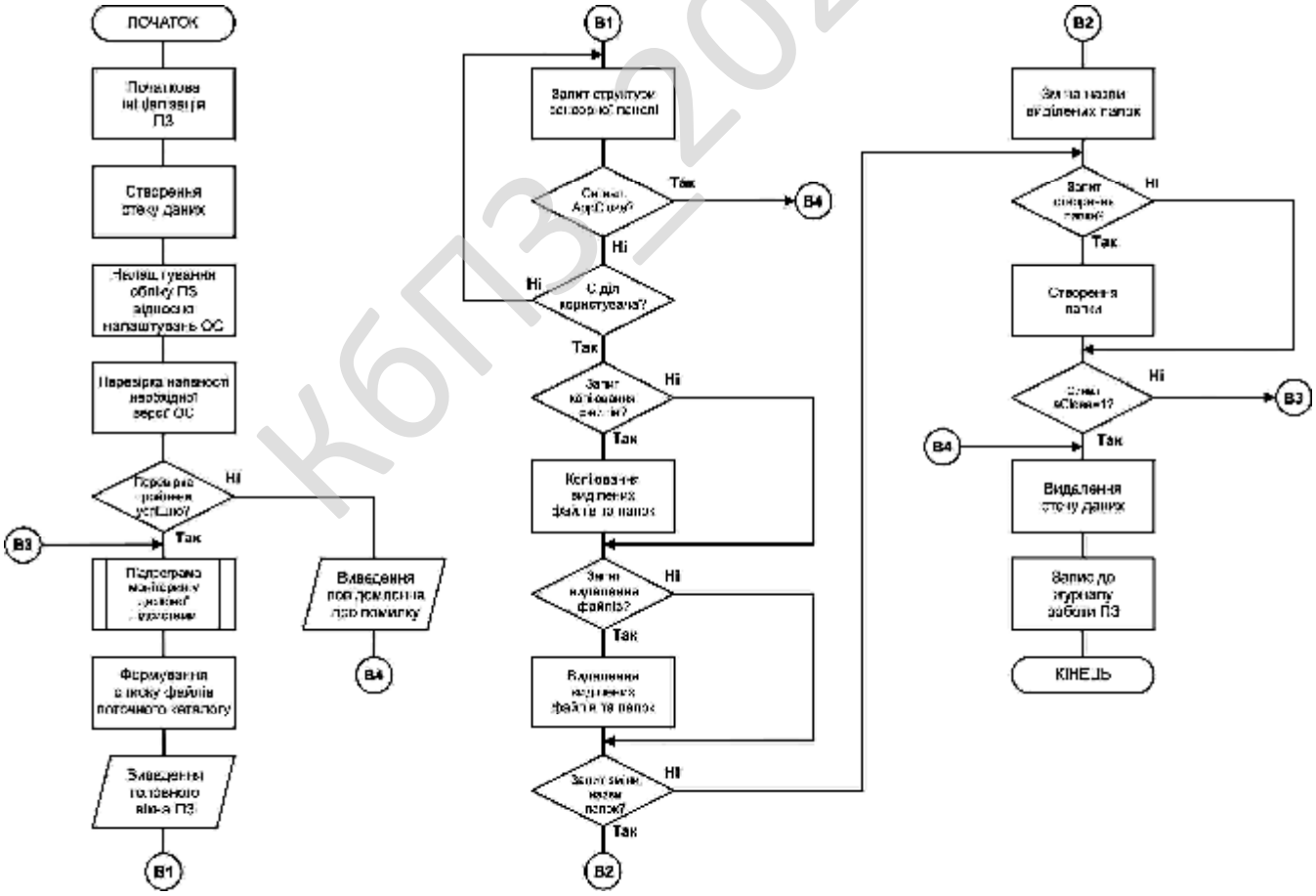

Рисунок 4.1 – Блок-схема основної програми

Після виклику підпрограми проводиться формування стеку списку папок, формування стеку списку файлів, запит структури поточного стану файлової підсистеми. Якщо є зміни у файловій підсистемі проводиться сканування основного диску, далі перевіряється наявність SD карти.

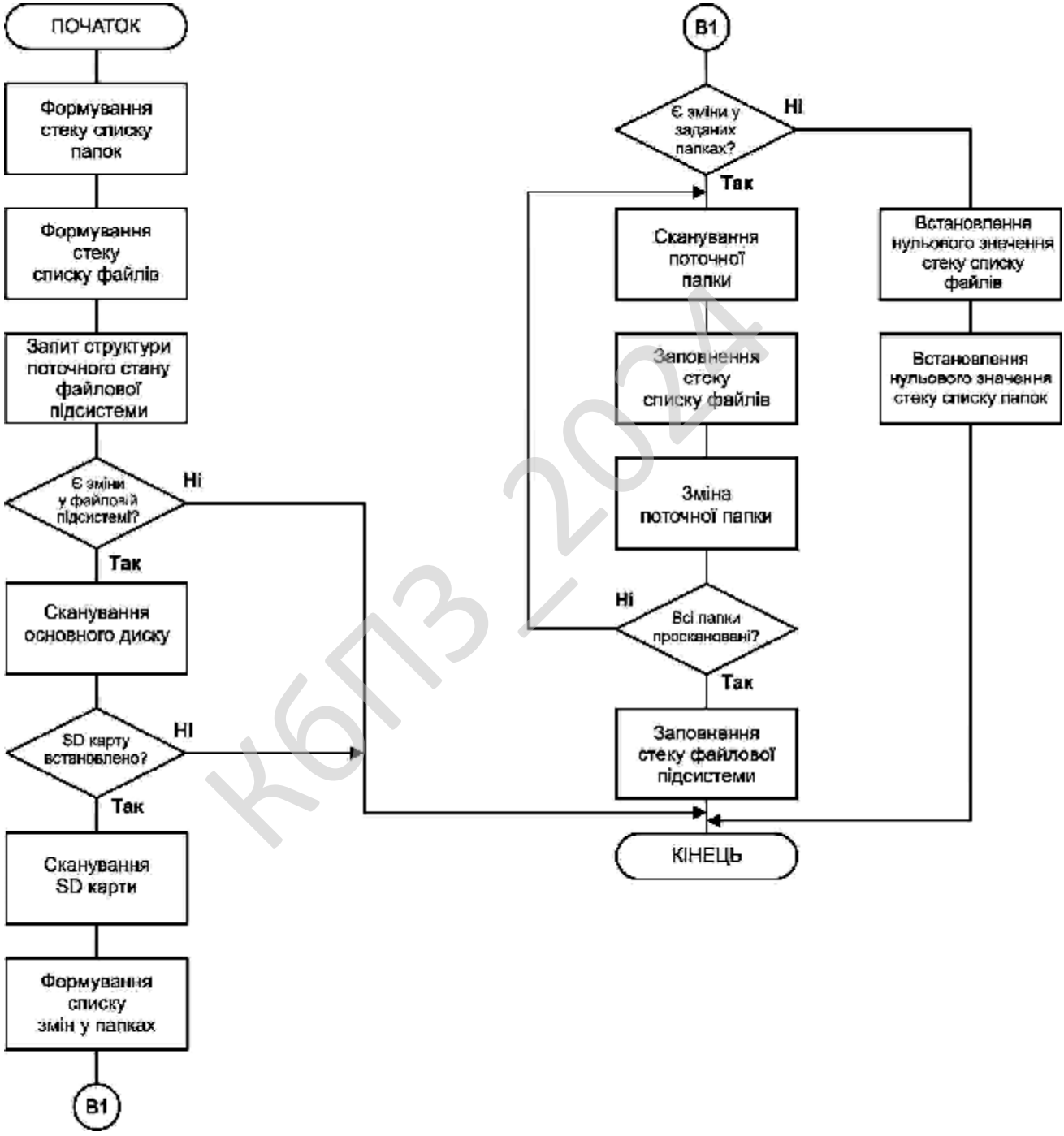

Рисунок 4.2 – Блок-схема підпрограми моніторингу дискової підсистеми

Якщо вона присутня теж проводиться сканування. Далі проходе формування списку змін у папках та якщо є зміни у заданих папках проводиться сканування поточної папки, заповнення стеку списку файлів та зміна поточної папки. Якщо всі папки просканово проводиться заповнення стеку файлової підсистеми та повернення до основної програми (рис. 4.1).

Після виконання підпрограми проходе формування списку файлів поточного каталогу, виведення головного вікна ПЗ та запит структури сенсорної панелі.

Далі якщо є дія з сенсорної панелі та немає сигналу appclose проводиться запит копіювання файлів з послідуючим копіюванням виділених файлів та папок. Якщо є запит видалення файлів проводиться видалення виділених файлів та папок. Якщо є запит зміни назви папок, проводиться зміна назви виділених папок. Та якщо є запит створення папки, проходе створення нової папки.

Якщо отримано сигнал aclose зі значенням 1 проводиться завершення роботи ПЗ – видалення стеку даних, запис до журналу роботи ПЗ.

Загалом, все що було вище – базові відомості, які кожен Android-програміст зобов'язаний знати. А тепер прийшла пора розбиратися з складнішими речами. Спробуємо створити простенький файловий менеджер

Файл main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns: android="http://schemas.android.com/apk/res/android"
  android: orientation="vertical"
  android: layout_width="fill_parent"
  android: layout_height="fill_parent">

  <TableRow>
    <TextView android: id="@+id/titleManager"
         android: layout_width="fill_parent"
         android: layout_height="fill_parent"
         android: padding="5dip"
          />
  </TableRow>
  <TableRow>
    <ListView android: id="@id/android: list"
```

```
            android: layout_width="fill_parent"
            android: layout_height="fill_parent"
            android: layout_weight="2"
            android: drawSelectorOnTop="false"/>
      </TableRow>
</TableLayout>
```

Тут у нас задається розмітка для нашого основного Layout 'а форми. TableLayout тут означає, що елементи у нас побудовані у вигляді таблиці. Далі у верхньому елементі таблиці розміщується елемент TextView(текстове поле), а в нижньому осередку – ListView(список). Обидва елементи мають id, використовуючи який, ми можемо змінювати вміст елементів. Наприклад, використовуючи R.id.titleManager для нашого текстового поля TextView.

Файл row.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns: android="http://schemas.android.com/apk/res/android"
    android: layout_width="fill_parent"
    android: layout_height="40sp"
    android: padding="5dip"
    android: gravity="center_vertical"
/>
```

Тут ми задаємо розмітку для кожного елементу нашого ListView, тобто безпосередньо для кожної окремо взятої теки або кожного файлу. У цьому коді у нас задається ширина кожного елементу, висота, відступ(padding) і вирівнювання center_vertical – тобто центрування по вертикалі.

Файл FileManager.java:

```
package ru.alwake.filemanager;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.content. intent ;
import android.content.DialogInterface.OnClickListener;
```

```
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
public class FileManager extends ListActivity {
  private List<String> directoryEntries = new ArrayList<String>();
  private File currentDirectory = new File("/");
  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    browseTo(new File("/"));
  }

  private void upOneLevel(){
    if(this.currentDirectory.getParent()!= null) {
      this.browseTo(this.currentDirectory.getParentFile());
    }
  }

  private void browseTo(final File aDirectory){
    if(aDirectory.isDirectory()){
      this.currentDirectory = aDirectory;
      fill(aDirectory.listFiles());
      TextView titleManager(TextView) findViewById(R.id.titleManager);
      titleManager.setText(aDirectory.getAbsolutePath());
    } else {
      OnClickListener okButtonListener = new OnClickListener(){
        public void onClick(DialogInterface arg0, int arg1) {
          intent i = new intent (android.content. intent.ACTION_VIEW,
Uri.parse("file://" + aDirectory.getAbsolutePath()));
          startActivity(i);
        }
      };
      OnClickListener cancelButtonListener = new OnClickListener(){
        public void onClick(DialogInterface arg0, int arg1) {
        }
      };
      new AlertDialog.Builder(this)
        . setTitle
```

```
                . setMessage
                . setPositiveButton
                . setNegativeButton
                . show();
            }
          }
          private void fill(File[] files) {
            this.directoryEntries.clear();
            if(this.currentDirectory.getParent()!= null)
              this.directoryEntries.add("".);
            for(File file: files) {
              this.directoryEntries.add(file.getAbsolutePath());
            }
            ArrayAdapter<String>  directoryList  =  new  ArrayAdapter<String>(this,
                R.layout.row, this.directoryEntries);
            this.setListAdapter(directoryList);
          }
          @Override
          protected void onListItemClick(ListView l, View v, int position, long id)
{
            int selectionRowID = position;
            String selectedFileString = this.directoryEntries.get(selectionRowID);
            if(selectedFileString.equals("".)){
              this.upOneLevel();
            } else {
              File clickedFile = null;
              clickedFile = new File(selectedFileString);
              if(clickedFile != null)
                this.browseTo(clickedFile);
            }
          }
        }
```

На початку вказується назва пакету(package name).

Рядки 2-18 відповідають за імпорт потрібних нам бібліотек. Важливо відмітити, що імпорт бібліотек Eclipse може проводити автоматично, як тільки зустріне що-небудь невідоме.

У цьому коді у нас всього 5 досить очевидних функцій, в яких нескладно розібратися. Вся робота базується на роботі з файлами в Java. І це – скелет застосування, що забезпечує основну навігацію по файловій структурі.

Проте у цього застосування існує одна проблема – при спробі зайти в директорію, доступ до якої дозволений тільки суперкористувачеві, ми отримуємо помилку і застосування завершує роботу.

Робота з сенсорним екраном

Займаючись програмуванням під Android необхідно знати, як взаємодіяти з сенсорним екраном – головним контролером для більшості Android пристроїв.

У Android є власний лог, куди можуть писати усі бажаючі застосування від Dvalvik VM до вашого застосування. Усі повідомлення у лозі розділяються по пріоритету(право вибирати пріоритет повідомлення, що посилається у лог залишається за вами)

Для запису у лог досить викликати один із статичних методів класу Log, що розрізняються тільки пріоритетом посиланих ними повідомлень.

Ці методи приведені нижче, за збільшенням пріоритету:

```
Log.v(String tag, String msg) – VERBOSE
Log.d(String tag, String msg) – DEBUG
Log.i(String tag, String msg) – INFO
Log.w(String tag, String msg) – WARNING
Log.e(String tag, String msg) – ERROR
Log.wtf(String tag, String msg) – What a Terrible Failure
```

Також є можливість відправляти у лог виключення. В цьому випадку додається ще один аргумент – Throwable tr. Наприклад:

```
Log.v(String tag, String msg, Throwable tr)
```

Вивести вікно Log 'a в Eclipse можна так:

Window -> Show View -> Other -> Android -> LogCat

Поняття про MotionEvent

Клас MotionEvent служить сховищем даних про дотик(touch event). Кожного разу, коли користувач проводить пальцем по екрану, або навіть просто тапає, створюється ціла послідовність екземплярів класу MotionEvent : послідовність починається, коли користувач доторкається до екрану, триває, поки користувач рухає палець по екрану, і кінчається, коли користувач піднімає палець. Таким чином виходить три основні типи дії, які описує MotionEvent : опускання пальця, його пересування і підняття.

Кожен MotionEvent зберігає в собі інформацію про тип дії, яку він описує. Для цього в класі MotionEvent передбачені константи:

int MotionEvent.ACTION_DOWN – користувач опускає палець на екран.

int MotionEvent.ACTION_MOVE – користувач пересуває пальцем по екрану.

int MotionEvent.ACTION_UP – користувач піднімає палець з екрану.

Упізнати яку дію описує цей MotionEvent можна за допомогою методу getAction().

Варто відмітити, що існує ще один тип дії – MotionEvent.ACTION_CANCEL, який по суті є позначенням того, що послідовність завершена некоректно, тобто без MotionEvent.ACTION_UP. Усе це буде пояснено детальніше на прикладі, але трохи пізніше.

Отримання і обробка MotionEvent 'ів

Тепер, коли ми отримали загальне уявлення про MotionEvent 'и, необхідно розібратися з тим, звідки нам їх «брати». Існує 2 способи, розберемо їх окремо.

Перший спосіб є оптимальним, коли ви використовуєте один із стандартних View. У класу View існує метод SetOnTouchListener(View.OnTouchListener l), аргументом якого є клас, що реалізовує інтерфейс OnTouchListener. Цей метод дозволяє призначити «слухача» MotionEvent 'ов. Інтерфейс OnTouchListener зобов'язує клас реалізовувати метод public boolean OnTouch(View v, MotionEvent event), другим аргументом якого і являється потрібний нам MotionEvent. Тепер кожного разу, коли користувач торкатиметься до екрану, Android викликатиме наш метод OnTouch, даючи йому в аргумент опис дії, що сталася, тобто MotionEvent. Розглянемо детальніше на прикладі:

Ось наш файл res/layout/main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout xmlns: android="schemas.android.com/apk/res/android"
       android: orientation="vertical"
       android: layout_width="fill_parent"
       android: layout_height="fill_parent"
```

```
              android: id="@+id/ll"
           </ LinearLayout
        А ось наш Activity клас:
        import android.app.Activity;
          import android.os.Bundle;
          import android.util.Log;
          import android.view.MotionEvent;
          import android.view.View;
          import android.view.View.OnTouchListener;
          import android.widget.LinearLayout;
          public class main extends Activity implements OnTouchListener{
           // Примушуємо наш Activity клас утілювати інтерфейс OnTouchListener
           ** Called when the activity is first created. */
             @Override
              public void onCreate(Bundle savedInstanceState) {
                   super.onCreate(savedInstanceState);
                   setContentView(R.layout.main);
                   LinearLayout ll(LinearLayout)
                   this.findViewById(R.id.ll);  //Дістаємо потрібний View
                   ll.setOnTouchListener(this);
// Встановлюємо цей клас в якості слухача MotionEvent'ів для нашого LinearLayout
              }
              @Override
              public boolean onTouch
                   // Ось, власне, метод, який і оброблятиме MotionEvent'и.
                  {
                   int Action=event.getAction();
// За допомогою методу getAction() отримуємо
// тип дії(ACTION_DOWN, ACTION_MOVE або ACTION_UP)
                   StringBuilder str=new StringBuilder();
                   str.append("type: ");
//Далі для кращого сприйняття(оскільки константи ACTION_DOWN,
// ACTION_MOVE і ACTION_UP числові)
// проводимо switch по змінній Action і додаємо в
// наш StringBuilder назву константи
                   switch(Action)
                      {
                       case MotionEvent.ACTION_DOWN :
str.append(«ACTION_DOWN»);break;
                       case MotionEvent.ACTION_MOVE :
str.append(«ACTION_MOVE»);break;
                       case MotionEvent.ACTION_UP : str.append(«ACTION_UP»);break;
```

```
                    }
        // За допомогою методів getX() і getY() отримуємо
        // координати по осі x і y відповідно
        // Слід зазначити, що точка 0 розташовується
        // в лівому верхньому кутку екрану.
                    // Вісь x спрямована управо
                    // Вісь y спрямована вниз(чим нижче, тим більше координати).
                    str.append(«Location: »).append(event.getX()).append(« x
»).append(event.getY()).append(«»);
        //Дізнаємося координати
                    str.append(«Edge flags:
»).append(event.getEdgeFlags()).append(«»);
        // Метод getEdgeFlags повертає інформацію про перетин країв екрану
                    str.append(«Pressure: »).append(event.getPressure()).append(«»);
        // Дізнаємося тиск
                    str.append(«Size: »).append(event.getSize()).append(«»);
        // Дізнаємося розмір покажчика(місця зіткнення пальця з екраном)
                    str.append(«Down time:
»).append(event.getDownTime()).append(«ms»);
        // Дізнаємося час, коли палець був опущений на екран в мілісекундах
                    str.append(«Event time:
»).append(event.getEventTime()).append(«ms»);
        //дізнаємося поточний час(що відповідає оброблюваному
        // MotionEvent 'y) в мілісекундах
                    str.append(" Elapsed: ").append(event.getEventTime() -
 event.getDownTime());
        //Дізнаємося скільки часу пройшло з моменту опускання пальця,
        // до поточного MotionEvent 'a
                    Log.v(«Mytag», str.toString());
        //Для того, щоб можна було відстежувати ці дії, записуємо
        // усю інформацію про них у лог.
                    return true;// Чому ми повертаємо true буде розглянуто потім
                    }
                }
```

Зрозуміло, що один клас може бути слухачем у декількох View. Щоб це враховувати метод onTouch приймає в аргумент View.

Другий спосіб є оптимальним, коли ви пишете свій View клас, який повинен реагувати на дотик. У View є метод onTouchEvent(MotionEvent event), який, власне, і дозволяє обробляти MotionEvent'и. Розглянемо на прикладі:

Напишемо свій клас SomeView:

```
package com.example.MotionEventExample1;
  import android.content.Context;
  import android.util.AttributeSet;
  import android.util.Log;
  import android.view.MotionEvent;
  import android.view.View;
  public class SomeView extends View
     {
  // Спочатку необхідно реалізувати конструктор.
  // Тут все просто – просто викликаємо конструктор суперкласу
      public SomeView(Context context, AttributeSet attrs)
     {
         super(context, attrs);
         }

        // Тепер приступаємо безпосередньо до обробки MotionEvent 'ів.
        // Для цього треба переписати метод onTouchEvent
         @Override
         public boolean onTouchEvent(MotionEvent event)
             {
             Float X(Float) event.getX();
             Float Y(Float) event.getY();
             int Action=event.getAction();
             String ActionString="";
             switch(Action)
                {
case MotionEvent.ACTION_DOWN : ActionString=«ACTION_DOWN»;break;
case MotionEvent.ACTION_MOVE : ActionString=«ACTION_MOVE»;break;
                  case MotionEvent.ACTION_UP : ActionString=«ACTION_UP»;break;
                }
             Log.v(«MyTag», «View OnTouchListener :»+«Coords: »+X.toString()+«
x »+Y.toString()+«type: »+ActionString);
             return true;
             }
        }

I наш /res/layout/main.xml:
<?xml version="1.0" encoding="utf-8"?>
  < LinearLayout xmlns: android="schemas.android.com/apk/res/android"
      android: orientation="vertical"
      android: layout_width="fill_parent"
      android: layout_height="fill_parent"
```

```
        >
       <com.example.MotionEventExample1.SomeView
          android: layout_width="fill_parent"
          android: layout_height="fill_parent"
          android: id="@+id/sv"
       />
  </ LinearLayout>
```

Activity клас:

```
package com.example.MotionEventExample1;
  import android.app.Activity;
  import android.os.Bundle;

  public class main extends Activity {
      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
      }
  }
```

Що повертають обробники MotionEvent'ів

Як ви могли помітити, обробники MotionEvent 'ів повинні повертати boolean 'ове значення. У прикладах до цього ми просто завжди повертали true. Так для чого ж цим методам взагалі треба повертати значення? Річ у тому, що у одного View може бути декілька слухачів(наприклад onTouchEvent в самому View і OnTouch у зовнішнього слухача) і вони мають деякий пріоритет: OnTouch викликається першим(якщо він взагалі є), а вже після нього може викликатися onTouchEvent. Виклик наступного по пріоритету обробника якраз залежить від повертаного поточним обробником значення (true – ніщо не викликається, false – викликається наступний по пріоритету, якщо такий є). Таким чином Android дозволяє нам розділяти обов'язки по обробці touch event 'ов на декілька слухачів.

Ось ми і розібралися з основами обробки MotionEvento'ів. Для закріплення навичок напишемо застосування, що реалізовує простенький Drag and Drop.

Ось наш /res/layout/main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns: android="schemas.android.com/apk/res/android"
       android: orientation="vertical"
```

```
        android: layout_width="fill_parent"
        android: layout_height="fill_parent"
        >
     <com.example.dragdrop.SomeView
        android: layout_width="fill_parent"
        android: layout_height="fill_parent"
        android: id="@+id/sv"
     />
  </LinearLayout>
```

Ось наш Activity клас:

```
package com.example.dragdrop;

   import android.app.Activity;
   import android.os.Bundle;

   public class main extends Activity {
       @Override
       public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
       }
  }
```

А ось наш основний клас SomeView :

```
package com.example.dragdrop;

   import android.content.Context;
   import android.graphics.Canvas;
   import android.graphics.Color;
   import android.graphics.Paint;
   import android.graphics.Paint.Style;
   import android.util.AttributeSet;
   import android.view.MotionEvent;
   import android.view.View;

  public class SomeView extends View
  {
      Paint paint;
      int X;
      int Y;
      final static int Radius=20;
      public SomeView(Context context, AttributeSet attrs)
```

```
        {
            super(context, attrs);
            paint = new Paint();
            paint.setColor(Color.YELLOW);
            paint.setStyle(Style.FILL);
            X=30;
            Y=30;
        }
        @Override
        public boolean onTouchEvent(MotionEvent event)
        {
            X(int) event.getX();
            Y(int) event.getY();
            return true;
        }

        @Override
        protected void onDraw
// метод OnDraw викликається Андроїдом тоді,
// коли треба відмалювати цей View
        {
            canvas.drawCircle(X, Y, Radius, paint);
            invalidate();
// invalidate() потрібний для того, щоб оповістити
// Android, що треба виконати метод OnDraw знову, без нього
// View не буде перемальовуватись.
        }

    }
```

Multitouch

У Android, скільки б ви пальців не використали, інформація про усіх них зберігається в одному MotionEvent'i. Перший метод, який треба упізнати, якщо ви збираєтеся використати мультитач – getPointerCount(), який повертає кількість пальців, зафіксованих на екрані(Не завжди співпадає з реальною кількістю із-за обмежень в залізі). Кожному пальцю, що знаходиться на екрані, надається індекс і id. Індекси завжди починаються з 0, id – не обов'язково. Для зрозумілості розглянемо на прикладі.

Перший палець опускається – йому привласнюється index=0 і id=0.

Другий палець опускається – йому привласнюється index=1 і id=1.

Перший палець піднімається – Другому пальцю привласнюється index=0(Індекси завжди розпочинаються з нуля), а id залишається тим же

Другий палець піднімається – MotionEvent 'ів більше немає, аж до наступного дотику.

Id завжди зберігається за покажчиком(пальцем), так що ми завжди можемо відстежити який завгодно покажчик.

Коли ми розібралися з id і індексами, розглянемо як набувати властивостей (координати, величину і т. п.) у конкретного покажчика, коли їх багато. Для цього в класі MotionEvent існують методи:

```
getX(int Index)
getY(int Index)
getSize(int Index)
```

і т. д., ми вже їх розглядали

У аргумент ці методи приймають індекс покажчика.

Але оскільки індекси покажчика можуть мінятися, а часто вимагається відстежити конкретний палець, то існує дуже важливий метод getPointerId(int index), який дозволяє по вказаному індексу дізнатися id покажчика.

Розглянемо приклад: перепишемо наше застосування(Drag and Drop) так, щоб воно сприймало мультитач.

Ось наш res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout xmlns: android="schemas.android.com/apk/res/android"
       android: orientation="vertical"
       android: layout_width="fill_parent"
       android: layout_height="fill_parent"
       >
   <com.example.multitouch.SomeView
       android: layout_width="fill_parent"
       android: layout_height="wrap_content"
      android: id="@+id/view"
      />
  </LinearLayout>
```

Наш Activity клас:

```
package com.example.multitouch;

   import android.app.Activity;
   import android.os.Bundle;

   public class main extends Activity{
       @Override
       public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
       }
  }
```

А ось основний клас – SomeView:

```
package com.example.multitouch;

   import android.content.Context;
   import android.graphics.Canvas;
   import android.graphics.Color;
   import android.graphics.Paint;
   import android.graphics.Paint.Style;
   import android.util.AttributeSet;
   import android.util.Log;
   import android.view.MotionEvent;
   import android.view.View;

  // Малюватимемо жовті круги під пальцями
   public class SomeView extends View
   {
      Paint paint;//потрібний щоб встановити колір.
      int[] X;
      int[] Y;
      final static int Radius=20;
      int PointerCount;
      public SomeView(Context context, AttributeSet attrs)
      {
          super(context, attrs);
          paint = new Paint();
          paint.setColor(Color.YELLOW);
          paint.setStyle(Style.FILL);
          PointerCount=0;
          X=new int[10];
```

```
        //Це будуть масиви координат(сприйматимемо до 10 пальців)
                Y=new int[10];
            }
            @Override
            public boolean onTouchEvent(MotionEvent event)
            {
                StringBuilder result=new StringBuilder(300);
                PointerCount=event.getPointerCount();
                for(int i=0;i<PointerCount;i++)
                {
                int ptrId=event.getPointerId(i);
                X[i]=(int) event.getX(i);// Запам'ятовуємо координати
                Y[i]=(int) event.getY(i);
                result.append(«Pointer Index : »).append(i);
                result.append("Pointer Id : ").append(ptrId).append("");
                result.append(«Location: »).append(event.getX(i));
                result.append(" x ").append(event.getX(i)).append("");
                result.append(«Pressure: »).append(event.getPressure(i));
                result.append(«Size: »).append(event.getSize(i)).append(«»);
                }
                Log.v(«Mytag», result.toString());//Виводимо усю інформацію у лог
                return true;
            }

            @Override
            protected void onDraw(Canvas canvas)
            {
                for//Дивимося скільки пальців було на екрані і рисуємо View
                {
                canvas.drawCircle(X[i], Y[i], Radius, paint);
                }
                invalidate();
// invalidate() потрібний для того, щоб оповістити Android, що
// треба виконати метод OnDraw знову, без нього View не буде перемальовувати
            }

        }
```

Інші способи взаємодії з тачскріном

Взаємодія з екраном через MotionEvent 'и є досить низькорівневим, треба це передусім для розробки ігор. Для вужчих, повсякденних ситуацій в Android

передбачений ряд інтерфейсів, які дозволяють розпізнавати найбільш «популярні» touch event 'и :

```
OnClickListener – onClick(View v)
OnFocusChangeListener – onFocusChange(View v, boolean hasFocus)
OnKeyListener – onKey(View v, int keyCode, KeyEvent event)
OnLongClickListener – onLongClick(View v)
```

і т. д.

Обробляються вони аналогічно, так що пояснення не потребують.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму UMAC (код автентифікації повідомлення на основі універсального гешування) – один з видів коду автентичності повідомлень (MAC).

Швидка «універсальна» функція використовується, для того, щоб гешувати вхідне повідомлення M у короткий рядок. До цього рядка потім застосовується функція XOR із псевдовипадковим значенням, у результаті чого ми одержуємо тег UMAC:

$$Tag=H_{K1}(M)\oplus F_{K2}(Nonce)$$

де K1 і K2 – секретні випадкові ключі, які мають одержувач і відправник.

Звідси видне, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну геш-функцію й псевдовипадкову послідовність. При цьому значення Nonce міняється кожний такт. Через використання Nonce, приймач і передавач повинні знати час відправлення повідомлення й принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторюване значення, наприклад порядковий номер повідомлення. При цьому даний номер не зобов'язано бути секретним, головне щоб він не повторювався.

UMAC розрахований на використання 32-х, 64-х, 92-х, і 128-бітових тегів, залежно від необхідного рівня безпеки. UMAC звичайно використовується разом з алгоритмом шифрування AES.

Функція створення ключа й псевдовипадкової послідовності

Створення псевдовипадкових байтів необхідно для роботи UHASH і при створенні тегів

Вибір блокового шифру

Для своєї роботи UMAC використовує блоковий шифр, вибір якого визначають наступні константи:

– BLOCLEN – довжина, у байтах, блоку з яким працює блоковий шифр.

– KEYLEN – довжина, у байтах, ключа блокового шифру.

При цьому використовується функція

– ENCIPHER(K,P) – зашифрувати рядок P з BLOCLEN байтів, використовуючи ключ K.

Приклад: якщо використовується AES з 16-байтним ключем, то BLOCLEN буде рівним 16(тому що AES працює з 16-байтними блоками).

KDF – функція створення ключа

Ця функція генерує послідовність псевдовипадкових байтів, використовуваних для ключових геш-функцій.

Вхід:

– K – рядок довжиною KEYLEN байт. / / Ключ блокового шифру.

– Index – ненегативне ціле число менше, чим 2 ^ 64.

– Numbytes – ненегативне ціле число менше, чим 2 ^ 64.

Вихід:

– Y – рядок довжини numbytes байт.

PDF: функція створення псевдовипадкового числа

Ця функція ухвалює ключ і даний час і повертає псевдовипадкове число для використання його в тегу покоління. За допомогою цієї функції можуть бути отримані числа довжиною 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– Nonce – рядок довжиною від 1 до BLOCKLEN байт.

– Taglen – ціле число 4, 8, 12 або 16.

Вихід:

– Y – послідовність байтів довжини taglen.

Генерація UMAC-тегів

Генерація UMAC-тегів відбувається за допомогою UHASH функції при використанні Nonce значенні й отриманої до цього рядка. Їхня довжина може бути 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– M – рядок довжиною менше 267 біт.

– Nonce – випадкове число від 1 до BLOCKLEN байт.

– Taglen – ціле 4, 8, 12 або 16.

Вихід:

– Тег, послідовність байтів довжиною taglen.

Алгоритм обчислення тегів:

Hashedmassage = UHASH(K, M, Taglen)

Pad = PDF(K, nonce, Taglen)

Tag = Pad xor Hashedmassage

UMAC-32 UMAC-64 UMAC-96 UMAC-128

Дані позначення містять у своїй назві певне значення довжини тегу:

– UMAC-32 (K, M, Nonce) = UMAC (K, M, Nonce, 4).

– UMAC-64 (K, M, Nonce) = UMAC (K, M, Nonce, 8).

– UMAC-96 (K, M, Nonce) = UMAC (K, M, Nonce, 12).

– UMAC-128 (K, M, Nonce) = UMAC (K, M, Nonce, 16).

Універсальна функція гешування(UHASH)

UHASH – універсальна функція гешування, серцевина алгоритму UMAC. UHASH – функція працює в три етапи. Спочатку до вхідного повідомлення застосовується L1-HASH, потім до цього результату застосовується L2-HASH і,

нарешті, до результату застосовується L3-HASH . Якщо при цьому довжина вхідного повідомлення не більш 1024 біт, то L2-HASH не використовується. Тому що функція L3-hash повертає тільки слово довжини 4 байта, те якщо потрібно одержати геш довжини більше 4 байт, здійснюється кілька ітерацій даної трирівневої схеми.

Універсальна функція

Нехай функція гешування вибирається із класу геш-функцій H, які відображають повідомлення в D, набір усіляких образів повідомлення. Цей клас називається універсальним, якщо для яких-небудь окремих пар повідомлень, існує на безлічі H/D функцій, функція, яка відображає їх в елемент D. Зміст цієї функції в тому, що якщо третя сторона прагне замінити одне повідомлення іншим, але при цьому вважає, що геш-функція була обрана абсолютно випадково, те ймовірність не виявлення підміни стороною, що ухвалює, прагне до 1/D.

L1-hash – перший етап

L1-hash розбиває повідомлення на шматки з 1024 байт і до кожного шматка застосовує алгоритм гешування називаний NH. Вихідний результат алгоритму NH в 128 раз менше вхідного.

L2-hash – другий етап

L2-hash працює з виходом L1-hash, використовує поліноміальний алгоритм POLY. Другий етап гешування використовується, тільки якщо довжина вхідного повідомлення більше 16 мегабайт. Використання алгоритму POLY потрібно для того, щоб уникнути тимчасову атаку. На виході з алгоритму POLY виходить 16 байтне число.

L3-hash – третій етап

Цей етап потрібно для того щоб з вихідних 16 байтів алгоритму L2-hash одержати 4-байтне значення.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми з різним функціоналом. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Навігаційне меню.

– Список файлів поточної директорії.

– Діалогові вікна функціоналу.

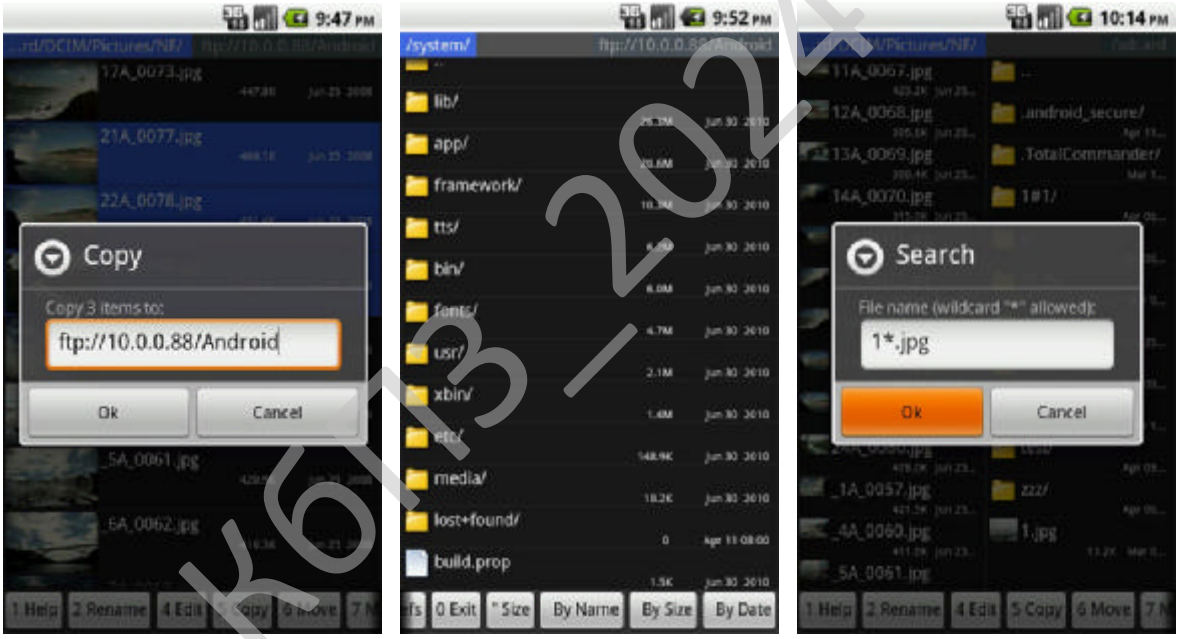

Рисунок 5.1 – Вікна розробленого ПЗ

Як зазначалося вище головне для багатьох користувачів програм – те, що програма відмінно справляється зі своїми основними функціями. Копіювання й вставка, переміщення, копіювання, перейменування й видалення файлів і папок, запуск файлів і встановлення стороннього ПЗ – все це було реалізовано в бакалаврській програмі.

У ПЗ можливо не тільки підключатися до зовнішніх мережних ресурсів (приміром, FTP), але й включати свій пристрій у локальні мережі (для цього передбачений убудований клієнт SMB), а також працювати із загальнодоступними хмарними сервісами, популярними останнім часом.

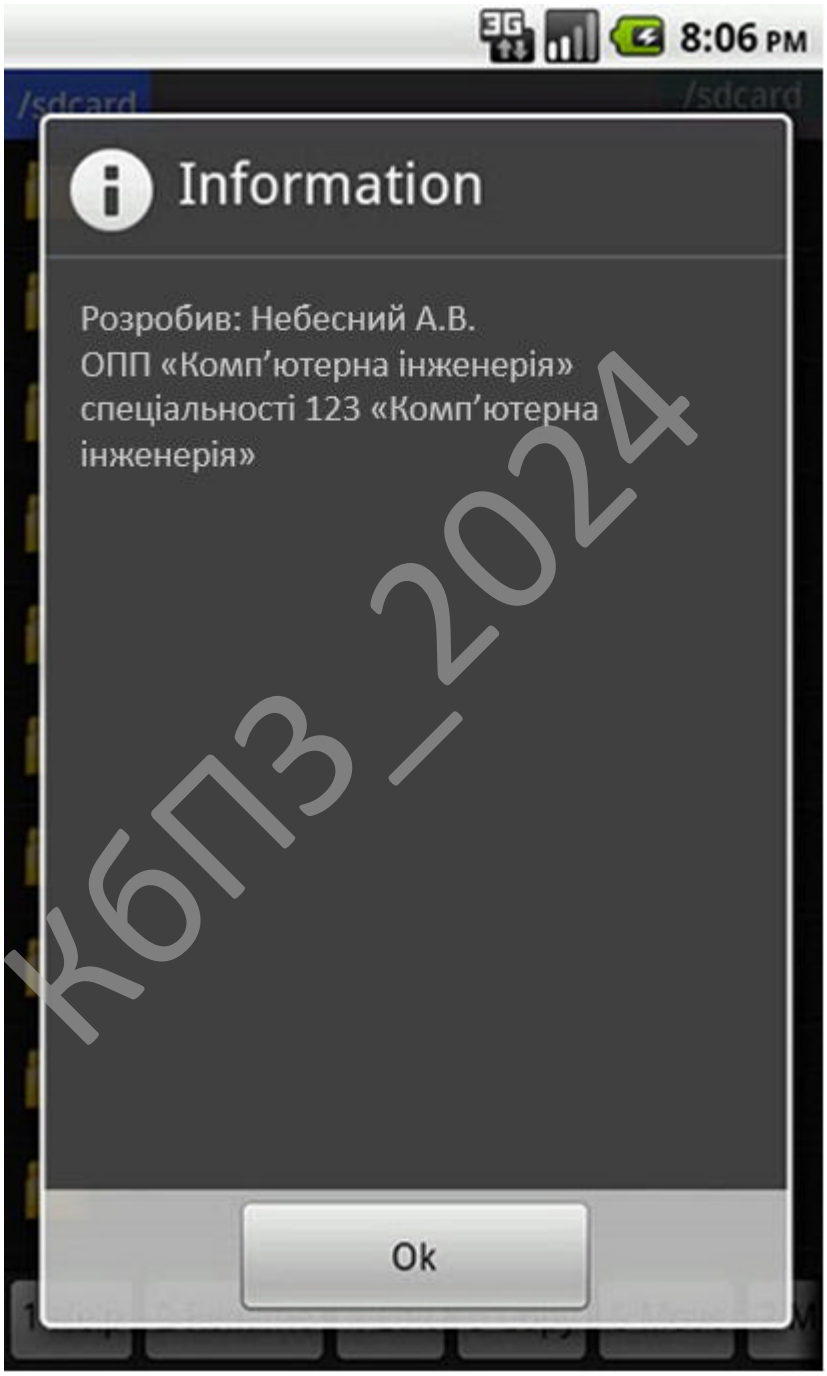

Рисунок 5.2 – Довідка розробника

Це, зокрема, Dropbox, Google Drive, Box.net, SkyDrive. У програми простий інтерфейс, заплутатися тут практично ніде. Є вбудований пошук файлів і папок, утиліта для перегляду текстових файлів, можливість упорядковувати файли за різними параметрами – типу, розміру, даті створення.

Так, можливо завдяки функції впорядкування ввійти в пункт, присвячений пакетам інсталяції (APK), і побачити всі настановні файли на вашім пристрої, де б вони не зберігалися. На рисунку 5.2 зображено форму авторського права.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

– Досліджена система керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

– На основі отриманих результатів досліджень створена програмна реалізація системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Java, Eclipse, Android SDK. Дана мова програмування дозволяє найбільш ефективно обробляти дані

призначені для системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання на бездротових мережних пристроях під керуванням ОС Android.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм UMAC.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adam Freeman. Pro Go The Complete Guide to Programming Reliable and Efficient Software Using Golang. Apress Media. 2022. 1078 p.

2. Fernando Doglio. Skills of a Successful Software Engineer. Manning. 2022. 182 c.

3. M. Holmes He. Creating Apps with React Native. Apress Media. 2022. 445 p.

4. Maurício Aniche. Effective Software Testing. Manning Publications. 2021. 372 p

5. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.

6. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.

7. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.

8. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.

9. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

10. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

11. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

12. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

13. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

14. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

15. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

16. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

17. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

18. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

19. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

20. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

21. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

22. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

23. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable

Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

24. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

25. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

26. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

27. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

28. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

29. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

30. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

31. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology»,

10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

32. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

33. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

34. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

35. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

36. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

37. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для

підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

38. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

39. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

40. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

41. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

42. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

43. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

44. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

45. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian

Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

46. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

47. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

48. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA - 2017.

49. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

50. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на бездротових мережних пристроях під керуванням ОС Android і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють на бездротових мережних пристроях під керуванням ОС Android.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Java, Eclipse, Android SDK.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 87 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 12.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О.М.

Програмне забезпечення системи керування файловими операціями на базі бездротових мережних пристроїв під керуванням ОС Android

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 57

Літера: РП

Кропивницький – 2024 року

TEXTVIEWER.JAVA ПЕРЕГЛЯД ДАНИХ

```
package com.mysq.diplom;
import com.mysq.diplom.adapters.CA;
import com.mysq.diplom.adapters.DiplomAdapter;
import com.mysq.diplom.utils.Credentials;
import com.mysq.diplom.utils.Utils;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.ContentResolver;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;
import java.io.InputStream;
/*
* Програмне забезпечення системи керування файловими операціями
* на базі бездротових мережних пристроїв під керуванням ОС Android
* Розробив:Небесний Андрій Вадимович
*/
public class TextViewer extends Activity {
    public  final static String TAG = "TextViewerActivity";
    private final static String SP_ENC = "encoding";
    public  final static String STRURI = "string:";
    public  final static String STRKEY = "string";
    private final static int VIEW_BOT = 595, VIEW_TOP = 590, VIEW_ENC = 363;
    private ScrollView scrollView;
    public  TextView    text_view;
    public  Uri uri;
    public  String encoding;
    @Override
public void onCreate( Bundle savedInstanceState ) {
        super.onCreate( savedInstanceState );
        try {
boolean ct_enabled = requestWindowFeature( Window.FEATURE_CUSTOM_TITLE );
            setContentView( R.layout.textvw );
```

```
            SharedPreferences shared_pref =
PreferenceManager.getDefaultSharedPreferences( this );
            int fs = Integer.parseInt( shared_pref != null ?
shared_pref.getString( "font_size", "12" ) : "12" );
            text_view = (TextView)findViewById( R.id.text_view );
            text_view.setTextSize( fs );
text_view.setTypeface( Typeface.create( "monospace", Typeface.NORMAL ) );
            ColorsKeeper ck = new ColorsKeeper( this );
            ck.restore();
            text_view.setBackgroundColor( ck.bgrColor );
            text_view.setTextColor( ck.fgrColor );

            if( ct_enabled ) {
                getWindow().setFeatureInt( Window.FEATURE_CUSTOM_TITLE,
R.layout.atitle );
                TextView act_name_tv = (TextView)findViewById( R.id.act_name );
                if( act_name_tv != null )
                    act_name_tv.setText( R.string.textvw_label );
            }
            scrollView = (ScrollView)findViewById( R.id.scroll_view );
        }
        catch( Exception e ) {
            e.printStackTrace();
        }
    }
    @Override
    protected void onStart() {
        super.onStart();
        SharedPreferences prefs = getPreferences( MODE_PRIVATE );
        if( prefs != null )
            encoding = prefs.getString( SP_ENC, "" );
        uri = getIntent().getData();
        if( !loadData() )
            finish();
        TextView file_name_tv = (TextView)findViewById( R.id.file_name );
        if( uri != null ) {
            String path = uri.getPath();
            if( file_name_tv != null && path != null && path.length() > 0 )
                file_name_tv.setText( " - " + path );
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
        SharedPreferences.Editor editor = getPreferences( MODE_PRIVATE ).edit();
        editor.putString( SP_ENC, encoding == null ? "" : encoding );
        editor.commit();
    }
```

```
    @Override
    protected void onSaveInstanceState( Bundle toSaveState ) {
        Log.i( TAG, "збереження: " + encoding );
        toSaveState.putString( SP_ENC, encoding == null ? "" : encoding );
        super.onSaveInstanceState( toSaveState );
    }
    @Override
    protected void onRestoreInstanceState( Bundle savedInstanceState ) {
        if( savedInstanceState != null )
            encoding = savedInstanceState.getString( SP_ENC );
        Log.i( TAG, "Відновлення статусу " + encoding );
        super.onRestoreInstanceState( savedInstanceState );
    }
    @Override
    public boolean onKeyDown( int keyCode, KeyEvent event ) {
        char c = (char)event.getUnicodeChar();
        switch( c ) {
        case 'q':
            finish();
            return true;
        case 'g':
            return dispatchCommand( VIEW_TOP );
        case 'G':
            return dispatchCommand( VIEW_BOT );
        }
        return super.onKeyDown( keyCode, event );
    }

    @Override
    public boolean onPrepareOptionsMenu( Menu menu ) {
        menu.clear();
menu.add(Menu.NONE,VIEW_TOP,Menu.NONE,getString(R.string.go_top)).setIcon(
android.R.drawable.ic_media_previous);
menu.add( Menu.NONE, VIEW_BOT, Menu.NONE, getString( R.string.go_end)).setIcon(
android.R.drawable.ic_media_next );
menu.add(Menu.NONE,VIEW_ENC,Menu.NONE,Utils.getEncodingDescr( this, encoding,
Utils.ENC_DESC_MODE_BRIEF ) ).setIcon(
android.R.drawable.ic_menu_sort_alphabetically );
        return true;
    }
    @Override
    public boolean onMenuItemSelected( int featureId, MenuItem item ) {
        if( dispatchCommand( item.getItemId() ) ) return true;
        return super.onMenuItemSelected( featureId, item );
    }
    public boolean dispatchCommand( int id ) {
        switch( id ) {
        case VIEW_BOT:
            scrollView.fullScroll( View.FOCUS_DOWN );
```

```
            return true;
        case VIEW_TOP:
            scrollView.fullScroll( View.FOCUS_UP );
            return true;
        case VIEW_ENC: {
                int cen = Integer.parseInt( Utils.getEncodingDescr( this,
encoding, Utils.ENC_DESC_MODE_NUMB ) );
                new AlertDialog.Builder( this )
                    .setTitle( R.string.encoding )
                    .setSingleChoiceItems( R.array.encoding, cen, new
DialogInterface.OnClickListener() {
                        public void onClick( DialogInterface dialog, int i ) {
encoding = getResources().getStringArray( R.array.encoding_vals )[i];
                            Log.i( TAG, "Обрання: " + encoding );
                            dialog.dismiss();
                            loadData();
                        }
                    }).show();
            }
            return true;
        case WRAP:
            try {
                EditText te = (EditText)findViewById( R.id.editor );
                horScroll = horScroll ? false : true;
                te.setHorizontallyScrolling( horScroll );
            }
            catch( Exception e ) {
                System.err.println("Виключення: " + e );
            }
        }
        return false;
    }
     private class DataLoadTask extends AsyncTask<Void, String, CharSequence> {
        @Override
        protected CharSequence doInBackground( Void... v ) {
            Uri uri = TextViewer.this.uri;
            try {
                final String   scheme = uri.getScheme();
                DiplomAdapter ca = null;
                InputStream is = null;
                if( ContentResolver.SCHEME_CONTENT.equals( scheme ) ) {
                    is = getContentResolver().openInputStream( uri );
                } else {
                    int type_id = CA.GetAdapterTypeId( scheme );
                    ca = CA.CreateAdapterInstance( type_id, TextViewer.this );
                    if( ca != null ) {
                        Credentials crd = null;
                        try {
```

```
crd = (Credentials)TextViewer.this.getIntent().getParcelableExtra(
      Credentials.KEY );
                        } catch( Exception e ) {
                        Log.e( TAG, "", e );
                        }
                        ca.setCredentials( crd );
                        is = ca.getContent( uri );
                    }
                }
                if( is != null ) {
                    CharSequence cs = Utils.readStreamToBuffer( is, encoding );
                    if( ca != null ) {
                        ca.closeStream( is );
                        ca.prepareToDestroy();
                    }
                    else
                        is.close();
                    return cs;
                }
            } catch( OutOfMemoryError e ) {
          Log.e( TAG, uri.toString(), e );
          publishProgress( getString( R.string.too_big_file, uri.getPath() ) );
            } catch( Throwable e ) {
Log.e( TAG, uri.toString(), e );
publishProgress( getString( R.string.failed ) + e.getLocalizedMessage() );
            }
            return null;
        }
        @Override
        protected void onProgressUpdate( String... err ) {
            Toast.makeText( TextViewer.this, err[0], Toast.LENGTH_LONG ).show();
        }
        @Override
        protected void onPostExecute( CharSequence cs ) {
            try {
                TextViewer.this.text_view.setText( cs );
            } catch( Throwable e ) {
onProgressUpdate( getString( R.string.failed ) + e.getLocalizedMessage() );
                e.printStackTrace();
            }
        }
     }

     private final boolean loadData() {
        if( uri != null ) {
            try {
                final String   scheme = uri.getScheme();
                if( STRKEY.equals( scheme ) ) {
                    Intent i = getIntent();
```

```
                    String str = i.getStringExtra( STRKEY );
                    if( str != null ) {
                        text_view.setText( str );
                        return true;
                    }
                    return false;
                }
                new DataLoadTask().execute();
                return true;
            } catch( OutOfMemoryError e ) {
                Log.e( TAG, uri.toString(), e );
Toast.makeText(this, getString( R.string.too_big_file, uri.getPath() ),
               Toast.LENGTH_LONG).show();
            } catch( Throwable e ) {
                Log.e( TAG, uri.toString(), e );
                Toast.makeText(this, getString( R.string.failed ) +
e.getLocalizedMessage(), Toast.LENGTH_LONG).show();
            }
        }
        return false;
    }
}
```

SERVERFORM.JAVA ОСНОВНА ФОРМА

```
package com.mysq.diplom;
import com.mysq.diplom.utils.Credentials;
import com.mysq.diplom.utils.Utils;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
/*
* Програмне забезпечення системи керування файловими операціями
* на базі бездротових мережних пристроїв під керуванням ОС Android
* Розробив:Небесний Андрій Вадимович
*/
public class ServerForm extends Activity implements View.OnClickListener {
    private static final String TAG = "ServerForm";
    private enum Type {
        FTP(   "ftp", "FTP" ),
        SFTP( "sftp", "SSH FTP" ),
        SMB(   "smb", "Windows PC" ),
        UNKNOWN( "?", "" );

        public String schema, title;

        private Type( String schema_, String title_ ) {
            schema = schema_;
            title  = title_;
        }
        public static Type getInstance( String s ) {
            if( s.equals(  FTP.schema ) ) return  FTP;
            if( s.equals( SFTP.schema ) ) return SFTP;
            if( s.equals(  SMB.schema ) ) return  SMB;
            return UNKNOWN;
        }
    };
    private Type type;

    private EditText server_edit;
    private EditText path_edit;
    private EditText domain_edit;
    private EditText name_edit;
```

```
    private CheckBox active_ftp_cb;
    private View     domain_block;
    @Override
    public void onCreate( Bundle savedInstanceState ) {
        try {
            super.onCreate( savedInstanceState );
            type = Type.getInstance( getIntent().getStringExtra( "schema" ) );
// Залиште поле порожнім, щоб побачити список вузлів
            setTitle( getString( R.string.connect ) + " " + type.title );
            requestWindowFeature( Window.FEATURE_LEFT_ICON );
            setContentView( R.layout.server );
// ширина
          getWindow().setLayout(LayoutParams.FILL_PARENT,
            LayoutParams.WRAP_CONTENT);
            getWindow().setFeatureDrawableResource( Window.FEATURE_LEFT_ICON,
            type == Type.SMB ? R.drawable.smb : R.drawable.server );
            server_edit = (EditText)findViewById( R.id.server_edit );
            path_edit = (EditText)findViewById( R.id.path_edit );
            domain_edit = (EditText)findViewById( R.id.domain_edit );
            domain_block = findViewById( R.id.domain_block );
            name_edit = (EditText)findViewById( R.id.username_edit );
            active_ftp_cb = (CheckBox)findViewById( R.id.active );
            Button connect_button = (Button)findViewById( R.id.connect );
            connect_button.setOnClickListener( this );
            Button browse_button = (Button)findViewById( R.id.browse );
            browse_button.setOnClickListener( this );
            Button cancel_button = (Button)findViewById( R.id.cancel );
            cancel_button.setOnClickListener( this );
active_ftp_cb.setVisibility( type == Type.FTP ? View.VISIBLE : View.GONE );
browse_button.setVisibility( type == Type.SMB ? View.VISIBLE : View.GONE );
domain_block.setVisibility( type == Type.SMB ? View.VISIBLE : View.GONE );
        }
        catch( Exception e ) {
            Log.e( TAG, "onCreate() Exception: ", e );
        }
    }
    @Override
    protected void onStart() {
        try {
            super.onStart();
            SharedPreferences prefs = getPreferences( MODE_PRIVATE );
            server_edit.setText( prefs.getString( "SERV", "" ) );
            path_edit.setText( prefs.getString( "PATH", "/" ) );
            domain_edit.setText( prefs.getString( "DOMAIN", "" ) );
            name_edit.setText( prefs.getString( "USER", "" ) );
            active_ftp_cb.setChecked( prefs.getBoolean( "ACTIVE", false ) );
        }
        catch( Exception e ) {
            Log.e( TAG, "onStart() Exception: ", e );
```

```
        }
    }
@Override
    protected void onPause() {
        try {
            super.onPause();
SharedPreferences.Editor editor = getPreferences( MODE_PRIVATE ).edit();
            editor.putString( "SERV", server_edit.getText().toString() );
            editor.putString( "PATH", path_edit.getText().toString() );
            editor.putString( "DOMAIN", domain_edit.getText().toString() );
            editor.putString( "USER", name_edit.getText().toString() );
            editor.putBoolean( "ACTIVE", active_ftp_cb.isChecked() );
            editor.commit();
        }
        catch( Exception e ) {
            Log.e( TAG, "onPause() Exception: ", e );
        }
    }

    @Override
    protected void onSaveInstanceState( Bundle outState ) {
        try {
            outState.putString( "SERV", server_edit.getText().toString() );
            outState.putString( "PATH", path_edit.getText().toString() );
            outState.putString( "USER", name_edit.getText().toString() );
            outState.putString( "DOMAIN", domain_edit.getText().toString() );
            outState.putBoolean( "ACTIVE", active_ftp_cb.isChecked() );
            super.onSaveInstanceState(outState);
        }
        catch( Exception e ) {
            Log.e( TAG, "onSaveInstanceState() Exception: ", e );
        }
    }
    @Override
    protected void onRestoreInstanceState( Bundle savedInstanceState ) {
        try {
            server_edit.setText( savedInstanceState.getString( "SERV" ) );
            path_edit.setText( savedInstanceState.getString( "PATH" ) );
            name_edit.setText( savedInstanceState.getString( "USER" ) );
            domain_edit.setText( savedInstanceState.getString( "DOMAIN" ) );
active_ftp_cb.setChecked( savedInstanceState.getBoolean( "ACTIVE", false ) );
            super.onRestoreInstanceState(savedInstanceState);
        }
        catch( Exception e ) {
            Log.e( TAG, "onRestoreInstanceState() Exception: ", e );
        }
    }
    @Override
    public void onClick( View v ) {
```

```
        try{
            if( v.getId() == R.id.browse ) {
if( type == Type.SMB )
setResult(RESULT_OK,new Intent(Diplom.NAVIGATE_ACTION,Uri.parse("smb://")));
            }
            else if( v.getId() == R.id.connect ) {
EditText pass_edit = (EditText)findViewById( R.id.password_edit );
                String user = name_edit.getText().toString().trim();
                String pass = pass_edit.getText().toString().trim();
                Credentials crd = null;
                if( user.length() > 0 ) {
                    if( type == Type.SMB ) {
                        EditText domain_edit = (EditText)findViewById(
R.id.domain_edit );
                        String domain = domain_edit.getText().toString().trim();
                        if( domain.length() > 0 )
                            user = domain + ";" + user;
                    }
                    crd = new Credentials( user, pass );
                }
                Uri.Builder uri_b = new Uri.Builder().scheme( type.schema )
.encodedAuthority( Utils.encodeToAuthority(
server_edit.getText().toString().trim() ) )
                    .path( path_edit.getText().toString().trim() );
                if( type == Type.FTP && active_ftp_cb.isChecked() )
                    uri_b.appendQueryParameter( "a", "true" );
Intent in = new Intent( Diplom.NAVIGATE_ACTION, uri_b.build() );
                if( crd != null )
                    in.putExtra( Credentials.KEY, crd );
                setResult( RESULT_OK, in );
            }
            else
                setResult( RESULT_CANCELED );
            finish();
        }
        catch( Exception e ) {
            Log.e( TAG, "onClick() Exception: ", e );
        }
    }
}
```

STREAMSERVER.JAVA ОСНОВНИЙ ПОТІК ПЗ

```
package com.mysq.diplom;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;
import com.mysq.diplom.adapters.CA;
import com.mysq.diplom.adapters.DiplomAdapter;
import com.mysq.diplom.adapters.DiplomAdapter.Item;
import com.mysq.diplom.utils.Credentials;
import com.mysq.diplom.utils.Utils;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.net.wifi.WifiManager;
import android.net.wifi.WifiManager.WifiLock;
import android.os.IBinder;
import android.util.Log;
/*
* Програмне забезпечення системи керування файловими операціями
* на базі бездротових мережних пристроїв під керуванням ОС Android
* Розробив:Небесний Андрій Вадимович
*/
public class StreamServer extends Service {
    private final static String TAG = "StreamServer";
    private final static String CRLF = "\r\n";
    private Context ctx;
    public  ListenThread  thread = null;
    public  WifiLock wifiLock = null;
    public  DiplomAdapter ca = null;
    public  String last_host = null;
// Передача повноважень
    public  static Credentials credentials = null;
    @Override
    public void onCreate() {
        super.onCreate();
        ctx = this;
        getApplicationContext();
        WifiManager manager = (WifiManager) getSystemService(
Context.WIFI_SERVICE );
        wifiLock = manager.createWifiLock( TAG );
        wifiLock.setReferenceCounted( false );
    }
```

```
    @Override
    public void onStart( Intent intent, int start_id ) {
        super.onStart( intent, start_id );
        Log.d( TAG, "onStart" );
        if( thread == null ) {
            Log.d( TAG, " Запуск сервера " );
            thread = new ListenThread();
            thread.start();
            getBaseContext();
        }
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d( TAG, "onDestroy" );
        if( thread != null && thread.isAlive() ) {
            thread.close();
            thread.interrupt();
            try {
                thread.join( 10000 );
            }
            catch( InterruptedException e ) {
                e.printStackTrace();
            }
            if( thread.isAlive() )
                Log.e( TAG, "переривання" );
        }
    }
    private class ListenThread extends Thread {
        private final static String TAG = "GCSS.ListenThread";
        private Thread stream_thread;
        public  ServerSocket ss = null;
        public  long lastUsed = System.currentTimeMillis();
        public void run() {
            try {
                Log.d( TAG, "старт" );
                setName( TAG );
                setPriority( Thread.MIN_PRIORITY );
                new Thread( new Runnable() {
                    @Override
                    public void run() {
                        while( true ) {
                            try {
synchronized( ListenThread.this ) {
final int max_idle = 100000;
ListenThread.this.wait( max_idle );
Log.d( TAG, " Перевірка часу: "
+ (System.currentTimeMillis()-lastUsed) + "" );
```

```
if( System.currentTimeMillis() - max_idle > lastUsed ) {
Log.d( TAG, "" );
ListenThread.this.close();
break;
                                    }
                                }
                            } catch( InterruptedException e ) {
                                e.printStackTrace();
                            }
                        }
                    }
                }, "Closer" ).start();

                StreamServer.this.wifiLock.acquire();
                Log.d( TAG, "" );
                synchronized( this ) {
                    ss = new ServerSocket( 5322 );
                }
                int count = 0;
                while( !isInterrupted() ) {
                    Log.d( TAG, "..." );
                    Socket data_socket = ss.accept();
                    if( data_socket != null && data_socket.isConnected() ) {
stream_thread = new StreamingThread( data_socket, count++ );
                        stream_thread.start();
                    }
                    touch();
                }
            }
            catch( Exception e ) {
                Log.w( TAG, "Виключення", e );
            }
            finally {
                StreamServer.this.wifiLock.release();
                Log.d( TAG, "" );
                this.close();
            }
            StreamServer.this.stopSelf();
        }
        public synchronized void touch() {
            lastUsed = System.currentTimeMillis();
        }

        public synchronized void close() {
            try {
                if( ss != null ) {
                    ss.close();
                    ss = null;
                }
```

```
            if( stream_thread != null && stream_thread.isAlive() ) {
                stream_thread.interrupt();
                stream_thread = null;
            }
        }
        catch( IOException e ) {
            e.printStackTrace();
        }
    }
};
private class StreamingThread extends Thread {
    private final static String TAG = "GCSS.StreamingThread";
    private Socket data_socket;
    private int num_id;
    public StreamingThread( Socket data_socket_, int num_id_ ) {
        data_socket = data_socket_;
        num_id = num_id_;
    }

    private void Log( String s ) {
        Log.d( TAG, "" + num_id + ": " + s );
    }

    public void run() {
        InputStream  is = null;
        OutputStream os = null;
        try {
            Log( "" );
            setName( TAG );
            if( data_socket == null || !data_socket.isConnected() ) {
                Log.e( TAG, "" );
                return;
            }
            os = data_socket.getOutputStream();
            if( os == null ) {
                return;
            }

            OutputStreamWriter osw = new OutputStreamWriter( os );
            yield();

            is = data_socket.getInputStream();
            if( is == null ) {
                SendStatus( osw, 500 );
                return;
            }

            InputStreamReader isr = new InputStreamReader( is );
            BufferedReader br = new BufferedReader( isr );
```

```
                String cmd = br.readLine();
                if( !Utils.str( cmd ) ) {
                    Log.e( TAG, "Invalid HTTP input" );
                    SendStatus( osw, 400 );
                    return;
                }

                String[] parts = cmd.split( " " );
                if( parts.length <= 1 ) {
                    Log.e( TAG, "Invalid HTTP input" );
                    SendStatus( osw, 400 );
                    return;
                }
                String passed_uri_s = parts[1].substring( 1 );
                if( !Utils.str( passed_uri_s ) ) {
                    SendStatus( osw, 404 );
                    return;
                }
                Uri uri = Uri.parse( Uri.decode( passed_uri_s ) );
                if( uri == null || !Utils.str( uri.getPath() ) ) {
                    SendStatus( osw, 404 );
                    return;
                }
                Log( "Requested URI: " + uri );

                long offset = 0;
                while( br.ready() ) {
                    String hl = br.readLine();
                    if( hl != null ) {
                        Log( hl );
                        if( hl.startsWith( "Range: bytes=" ) ) {
                            int end = hl.indexOf( '-', 13 );
                            String range_s = hl.substring( 13, end );
                            try {
                                offset = Long.parseLong( range_s );
                            } catch( NumberFormatException nfe ) {}
                        }
                    }
                }

                String scheme = uri.getScheme();
                if( scheme == null ) {
                    SendStatus( osw, 404 );
                    return;
                }
                int ca_type = CA.GetAdapterTypeId( scheme );
                String host = uri.getHost();
                if( StreamServer.this.ca == null ||
StreamServer.this.ca.getType() != ca_type ||
```

```
                        ( host != null && !host.equals( last_host ) ) )
                    ca = CA.CreateAdapterInstance( ca_type, ctx );
                if( ca == null ) {
                    SendStatus( osw, 500 );
                    return;
                }

                Log( "Adapter is created" );
                last_host = host;
                if( credentials != null )
                    ca.setCredentials( credentials );
                Item item = ca.getItem( uri );
                if( item == null ) {
                    Log.e( TAG, "Can't get the item for " + uri );
                    SendStatus( osw, 404 );
                    return;
                }
                InputStream cs = ca.getContent( uri, offset );
                if( cs == null ) {
                    Log.e( TAG, "Can't get the content for " + uri );
                    SendStatus( osw, 500 );
                    return;
                }
                if( offset > 0 && item != null ) {
                    SendStatus( osw, 206 );
                } else {
                    SendStatus( osw, 200 );
                }
String fn = "zip".equals( scheme ) ? uri.getFragment() :
            uri.getLastPathSegment();
                if( fn != null ) {
                    String ext = Utils.getFileExt( fn );
                    String mime = Utils.getMimeByExt( ext );
                    Log( "Content-Type: " + mime );
                    osw.write( "Content-Type: " + mime + CRLF );
                }
                else
osw.write( "Content-Type: application/octet-stream" + CRLF );
String content_range = null, content_len = "Content-Length: " + item.size;
if(offset==0) content_range="Content-Range: bytes 0-" +
              (item.size-1) + "/" + item.size;
                else
                content_range = "Content-Range: bytes " + offset + "-" +
                                (item.size-1) + "/" + item.size;
                osw.write( content_len + CRLF );
                osw.write( content_range + CRLF );
                Log( content_len );
                Log( content_range );
                Date date = new Date();
```

```
                osw.write( "Date: " + date + CRLF );
                Log( "Date: " + date + CRLF );
                osw.write( CRLF );
                osw.flush();
                ReaderThread rt = new ReaderThread( cs, num_id );
                rt.start();
                setPriority( Thread.MAX_PRIORITY );
                int count = 0;
                while( rt.isAlive() ) {
                    try {
                        if( br.ready() )
                            Log( "HTTP " +
                            br.readLine() );
                        thread.touch();
                        byte[] out_buf = rt.getOutputBuffer();
                        if( out_buf == null ) break;
                        int n = rt.GetDataSize();
                        if( n < 0 )
                            break;
                        os.write( out_buf, 0, n );
                        Log( "      ...W " + n + "/" + ( count += n ) );
                    }
                    catch( Exception e ) {
                        break;
                    }
                    finally {
                        rt.doneOutput();
                    }
                }
                ca.closeStream( cs );
                rt.interrupt();
                Log( "---------- done ----------" );
            }
            catch( Exception e ) {
                Log.e( TAG, "Exception", e );
            }
            finally {
                Log( "Thread exits" );
                try {
                    if( is != null ) is.close();
                    if( os != null ) os.close();
                    if( ca != null ) ca.prepareToDestroy();
                }
                catch( IOException e ) {
                    Log.e( TAG, "Exception on Closing", e );
                }
            }
        }
    private void SendStatus( OutputStreamWriter osw, int code ) throws IOException {
```

```
            final String http = "HTTP/1.1 ";
            String descr;
            switch( code ) {
            case 200: descr = "OK";                 break;
            case 206: descr = "Partial Content";    break;
            case 400: descr = "Invalid";            break;
            case 404: descr = "Not found";          break;
            case 500: descr = "Server error";       break;
            default:  descr = "";
            }
            osw.write( http + code + " " + descr + CRLF );
            Log( "Sending status: " + code );
        }
    };

    class ReaderThread extends Thread {
        private final static String TAG = "GCSS.RT";
        private InputStream is;
        private int      roller = 0, chunk = 4096;
        private final static int MAX = 32768;
        private byte[][] bufs = { new byte[MAX], new byte[MAX] };
        private byte[]   out_buf = null;
        private int      data_size = 0;
        private int      num_id;

        public ReaderThread( InputStream is_, int num_id_ ) {
            is = is_;
            setName( TAG );
            num_id = num_id_;
        }

        private void Log( String s ) {
            Log.d( TAG, "" + num_id + ": " + s );
        }
        public void run() {
            try {
                setPriority( Thread.MAX_PRIORITY );
                int count = 0;
                while( true ) {
                    byte[] inp_buf = bufs[roller++ % 2];
                    Log( "R..." );
                    int has_read = 0;
                    synchronized( TAG ) {
// Тестування
                        has_read = is.read( inp_buf, 0, chunk );
                    }
                    if( has_read < 0 )
                        break;
                    if( has_read == chunk && chunk < MAX )
```

```
                    chunk <<= 1;
                if( chunk > MAX )
                    chunk = MAX;
                Log( "...R " + has_read + "/" + ( count += has_read ) );
                synchronized( this ) {
                    Log( "?.." );
                    int wcount = 0;
                    while( out_buf != null ) {
                        wait( 10 );
                        wcount += 10;
                    }
                    Log( "...! (" + wcount + "ms)" );
                    out_buf = inp_buf;
                    data_size = has_read;
                    Log( "O=I ->" );
                    notify();
                }
            }
        } catch( Throwable e ) {
            Log.e( TAG, "" + num_id + ": ", e );
        }
    }
public synchronized byte[] getOutputBuffer() throws InterruptedException
{
        int wcount = 0;
        Log( "       ?.." );
        while( out_buf == null && this.isAlive() ) {
Log( " Очікування, коли вихідний буфер готовий " );
            wait( 10 );
            wcount += 10;
        }

        if( out_buf != null )
            Log( "      ..! (" + wcount + "ms)" );
        else
            Log( "X" );

        return out_buf;
    }
    public int GetDataSize() {
        int ds = data_size;
        data_size = 0;
        return ds;
    }
    public synchronized void doneOutput() {
        out_buf = null;
        notify();
    }
};
```

```
    @Override
    public IBinder onBind( Intent intent ) {
        return null;
    }
}
```

TOUCHIMAGEVIEW.JAVA МАСШТАБУВАННЯ І ПАНОРАМУВАННЯ ПЕРЕГЛЯДАЄМОГО ЗОБРАЖЕННЯ

```
package com.example.touch;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Matrix;
import android.graphics.PointF;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.ImageView;
/*
* Програмне забезпечення системи керування файловими операціями
* на базі бездротових мережних пристроїв під керуванням ОС Android
* Розробив:Небесний Андрій Вадимович
*/
public class TouchImageView extends ImageView {
    Matrix matrix = new Matrix();
// Ми можемо бути в одному з цих 3-х статусів
    static final int NONE = 0;
    static final int DRAG = 1;
    static final int ZOOM = 2;
    int mode = NONE;
// Збереження поточного положення
    PointF last = new PointF();
    PointF start = new PointF();
    float minScale = 1f;
    float maxScale = 3f;
    float[] m;

    float redundantXSpace, redundantYSpace;

    float width, height;
    static final int CLICK = 3;
    float saveScale = 1f;
    float right, bottom, origWidth, origHeight, bmWidth, bmHeight;

    ScaleGestureDetector mScaleDetector;

    Context context;
    public TouchImageView(Context context) {
        super(context);
        super.setClickable(true);
        this.context = context;
        mScaleDetector = new ScaleGestureDetector(context, new ScaleListener());
        matrix.setTranslate(1f, 1f);
        m = new float[9];
        setImageMatrix(matrix);
        setScaleType(ScaleType.MATRIX);
```

```
setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
          mScaleDetector.onTouchEvent(event);
          matrix.getValues(m);
          float x = m[Matrix.MTRANS_X];
          float y = m[Matrix.MTRANS_Y];
          PointF curr = new PointF(event.getX(), event.getY());

          switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                  last.set(event.getX(), event.getY());
                  start.set(last);
                  mode = DRAG;
                  break;
                case MotionEvent.ACTION_MOVE:
                      if (mode == DRAG) {
                            float deltaX = curr.x - last.x;
                            float deltaY = curr.y - last.y;
                float scaleWidth = Math.round(origWidth * saveScale);
                float scaleHeight = Math.round(origHeight * saveScale);
                            if (scaleWidth < width) {
                                  deltaX = 0;
                                  if (y + deltaY > 0)
                                        deltaY = -y;
                                  else if (y + deltaY < -bottom)
                                        deltaY = -(y + bottom);
                            } else if (scaleHeight < height) {
                                  deltaY = 0;
                                  if (x + deltaX > 0)
                                        deltaX = -x;
                                  else if (x + deltaX < -right)
                                        deltaX = -(x + right);
                            } else {
                                  if (x + deltaX > 0)
                                        deltaX = -x;
                                  else if (x + deltaX < -right)
                                        deltaX = -(x + right);

                                  if (y + deltaY > 0)
                                        deltaY = -y;
                                  else if (y + deltaY < -bottom)
                                        deltaY = -(y + bottom);
                            }
                      matrix.postTranslate(deltaX, deltaY);
                      last.set(curr.x, curr.y);
                  }
                      break;
```

```
                        case MotionEvent.ACTION_UP:
                              mode = NONE;
                              int xDiff = (int) Math.abs(curr.x - start.x);
                            int yDiff = (int) Math.abs(curr.y - start.y);
                            if (xDiff < CLICK && yDiff < CLICK)
                                performClick();
                              break;

                        case MotionEvent.ACTION_POINTER_UP:
                              mode = NONE;
                              break;
                    }
                setImageMatrix(matrix);
                invalidate();
                return true;
// вказує на останню подію яка була оброблена
            }
        });
    }
    @Override
    public void setImageBitmap(Bitmap bm) {
        super.setImageBitmap(bm);
        bmWidth = bm.getWidth();
        bmHeight = bm.getHeight();
    }

    public void setMaxZoom(float x)
    {
      maxScale = x;
    }

    private class ScaleListener extends
ScaleGestureDetector.SimpleOnScaleGestureListener {
      @Override
      public boolean onScaleBegin(ScaleGestureDetector detector) {
            mode = ZOOM;
            return true;
      }

            @Override
          public boolean onScale(ScaleGestureDetector detector) {
float mScaleFactor = (float)Math.min(Math.max(.95f,
                    detector.getScaleFactor()), 1.05);
                float origScale = saveScale;
            saveScale *= mScaleFactor;
            if (saveScale > maxScale) {
                saveScale = maxScale;
                mScaleFactor = maxScale / origScale;
            } else if (saveScale < minScale) {
```

```
                saveScale = minScale;
                mScaleFactor = minScale / origScale;
            }
right = width * saveScale - width - (2 * redundantXSpace * saveScale);
bottom = height * saveScale - height - (2 * redundantYSpace * saveScale);
if (origWidth * saveScale <= width || origHeight * saveScale <= height) {
matrix.postScale(mScaleFactor, mScaleFactor, width / 2, height / 2);
                if (mScaleFactor < 1) {
                      matrix.getValues(m);
                      float x = m[Matrix.MTRANS_X];
float y = m[Matrix.MTRANS_Y];
                if (mScaleFactor < 1) {
                      if (Math.round(origWidth * saveScale) < width) {
                            if (y < -bottom)
                                  matrix.postTranslate(0, -(y + bottom));
                            else if (y > 0)
                                  matrix.postTranslate(0, -y);
                      } else {
                            if (x < -right)
                                  matrix.postTranslate(-(x + right), 0);
                            else if (x > 0)
                                  matrix.postTranslate(-x, 0);
                      }
                }
                }
          } else {
matrix.postScale(mScaleFactor, mScaleFactor, detector.getFocusX(),
                detector.getFocusY());
                matrix.getValues(m);
                float x = m[Matrix.MTRANS_X];
                float y = m[Matrix.MTRANS_Y];
                if (mScaleFactor < 1) {
                if (x < -right)
                      matrix.postTranslate(-(x + right), 0);
                else if (x > 0)
                      matrix.postTranslate(-x, 0);
                if (y < -bottom)
                      matrix.postTranslate(0, -(y + bottom));
                else if (y > 0)
                      matrix.postTranslate(0, -y);
                }
          }
            return true;

        }
    }
    @Override
    protected void onMeasure (int widthMeasureSpec, int heightMeasureSpec)
    {
```

```
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        width = MeasureSpec.getSize(widthMeasureSpec);
        height = MeasureSpec.getSize(heightMeasureSpec);
// привести до розміру екрана
        float scale;
        float scaleX =  (float)width / (float)bmWidth;
        float scaleY = (float)height / (float)bmHeight;
        scale = Math.min(scaleX, scaleY);
        matrix.setScale(scale, scale);
        setImageMatrix(matrix);
        saveScale = 1f;
// Центрування зображення що переглядається
        redundantYSpace = (float)height - (scale * (float)bmHeight) ;
        redundantXSpace = (float)width - (scale * (float)bmWidth);
        redundantYSpace /= (float)2;
        redundantXSpace /= (float)2;
        matrix.postTranslate(redundantXSpace, redundantYSpace);

        origWidth = width - 2 * redundantXSpace;
        origHeight = height - 2 * redundantYSpace;
right = width * saveScale - width - (2 * redundantXSpace * saveScale);
bottom = height * saveScale - height - (2 * redundantYSpace * saveScale);
        setImageMatrix(matrix);
    }
}
```

TOUCHLISTVIEW.JAVA ОБРОБКА ЗАПИТІВ СЕНСОРА

```
package com.commonsware.cwac.tlv;
import com.mysq.diplom.R;
import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.PixelFormat;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewConfiguration;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.ListView;
/*
* Програмне забезпечення системи керування файловими операціями
* на базі бездротових мережних пристроїв під керуванням ОС Android
* Розробив:Небесний Андрій Вадимович
*/
public class TouchListView extends ListView {
      private ImageView mDragView;
      private WindowManager mWindowManager;
      private WindowManager.LayoutParams mWindowParams;
      private int mDragPos;
// який елемент тягнуть (перетаскивають). Сенсором
      private int mFirstDragPos;
// де був елемент спочатку
      private int mDragPoint;
// зміщення всередині елемента
      private int mCoordOffset;
// різниця між екранними координатами та координатами локальними
      private DragListener mDragListener;
      private DropListener mDropListener;
      private RemoveListener mRemoveListener;
      private int mUpperBound;
      private int mLowerBound;
      private int mHeight;
      private GestureDetector mGestureDetector;
      public static final int FLING = 0;
      public static final int SLIDE_RIGHT = 1;
      public static final int SLIDE_LEFT = 2;
      private int mRemoveMode = -1;
```

```
	private Rect mTempRect = new Rect();
	private Bitmap mDragBitmap;
	private final int mTouchSlop;
	private int mItemHeightNormal=-1;
	private int mItemHeightExpanded=-1;
	private int grabberId=-1;
	private int dragndropBackgroundColor=0x00000000;
	public TouchListView(Context context, AttributeSet attrs) {
		this(context, attrs, 0);
	}

public TouchListView(Context context, AttributeSet attrs, int defStyle) {
		super(context, attrs, defStyle);

		mTouchSlop = ViewConfiguration.get(context).getScaledTouchSlop();

if (attrs!=null)
{
TypedArray a=getContext().obtainStyledAttributes(attrs,
R.styleable.TouchListView, 0, 0);
mItemHeightNormal=a.getDimensionPixelSize(
                  R.styleable.TouchListView_normal_height, 0);
mItemHeightExpanded=a.getDimensionPixelSize(R.styleable.TouchListView_expanded_h
eight, mItemHeightNormal);
grabberId=a.getResourceId(R.styleable.TouchListView_grabber, -1);
dragndropBackgroundColor=a.getColor(R.styleable.TouchListView_dragndrop_backgrou
nd, 0x00000000);
mRemoveMode=a.getInt(R.styleable.TouchListView_remove_mode, -1);
a.recycle();
		}
  }

	@Override
final public void addHeaderView (View v, Object data, boolean isSelectable) {
throw new RuntimeException("Headers are not supported with TouchListView");
	}
	@Override
	final public void addHeaderView (View v) {
throw new RuntimeException("Headers are not supported with TouchListView");
	}

  @Override
  final public void addFooterView(View v, Object data, boolean isSelectable) {
if (mRemoveMode == SLIDE_LEFT || mRemoveMode == SLIDE_RIGHT)
{
throw new RuntimeException("Footers are not supported with TouchListView in
conjunction with remove_mode");
}
  }
```

```
  @Override
  final public void addFooterView(View v) {
    if (mRemoveMode == SLIDE_LEFT || mRemoveMode == SLIDE_RIGHT) {
      throw new RuntimeException("Footers are not supported with TouchListView
in conjunction with remove_mode");
    }
  }

      @Override
      public boolean onInterceptTouchEvent(MotionEvent ev) {
        if( mRemoveListener != null && mGestureDetector == null ) {
            if( mRemoveMode == FLING ) {
mGestureDetector = new GestureDetector( getContext(), new
                      SimpleOnGestureListener()
{
                    @Override
public boolean onFling( MotionEvent e1, MotionEvent e2,
float velocityX, float velocityY )
{
                        if( mDragView != null ) {
                            if( velocityX > 1000 ) {
                                Rect r = mTempRect;
                                mDragView.getDrawingRect( r );
                                if( e2.getX() > r.right * 2 / 3 ) {
// Правий край екрана
                                    stopDragging();
                                    mRemoveListener.remove( mFirstDragPos );
                                    unExpandViews( true );
                                }
                            }
// Кидаючи під час перетягування не повинно мати жодного ефекту. Сенсор
                            return true;
                        }
                        return false;
                    }
                } );
            }
        }
        if( mDragListener != null || mDropListener != null ) {
            switch( ev.getAction() ) {
            case MotionEvent.ACTION_DOWN:
                int x = (int)ev.getX();
                int y = (int)ev.getY();
                int itemnum = pointToPosition( x, y );
                if( itemnum == AdapterView.INVALID_POSITION ) {
                    break;
                }
View item = (View)getChildAt( itemnum - getFirstVisiblePosition() );
                if( isDraggableRow( item ) ) {
```

```
                    mDragPoint = y - item.getTop();
                    mCoordOffset = ( (int)ev.getRawY() ) - y;
                    View dragger = item.findViewById( grabberId );
                    Rect r = mTempRect;
dragger.getDrawingRect(r);
                    r.left = dragger.getLeft();
                    r.right = dragger.getRight();
                    r.top = dragger.getTop();
                    r.bottom = dragger.getBottom();
                    if( ( r.left < x ) && ( x < r.right ) ) {
                        item.setDrawingCacheEnabled( true );
// Створення копії кешу
Bitmap bitmap = Bitmap.createBitmap( item.getDrawingCache() );
                        Rect listBounds = new Rect();
                        getGlobalVisibleRect( listBounds, null );
                        startDragging( bitmap, listBounds.left, y );
                        mDragPos = itemnum;
                        mFirstDragPos = mDragPos;
                        mHeight = getHeight();
                        int touchSlop = mTouchSlop;
                      mUpperBound = Math.min( y - touchSlop, mHeight / 3 );
                      mLowerBound = Math.max( y + touchSlop, mHeight*2/3);
                        return false;
                    }
                    mDragView = null;
                }
                break;
            }
        }
        return super.onInterceptTouchEvent( ev );
    }

    protected boolean isDraggableRow(View view) {
        return(view.findViewById(grabberId)!=null);
    }

    private int myPointToPosition(int x, int y) {
      Rect frame = mTempRect;
      final int count = getChildCount();
      for( int i = count - 1; i >= 0; i-- ) {
          final View child = getChildAt( i );
          child.getHitRect( frame );
          if( frame.contains( x, y ) ) {
              return getFirstVisiblePosition() + i;
          }
      }
      return INVALID_POSITION;
    }
```

```
      private int getItemForPosition(int y) {
        int adjustedy = y - mDragPoint - 32;
        int pos = myPointToPosition( 0, adjustedy );
        if( pos >= 0 ) {
            if( pos <= mFirstDragPos ) {
                pos += 1;
            }
        } else if( adjustedy < 0 ) {
            pos = 0;
        }
        return pos;
      }

      private void adjustScrollBounds(int y) {
        if( y >= mHeight / 3 ) {
            mUpperBound = mHeight / 3;
        }
        if( y <= mHeight*2/3){
            mLowerBound = mHeight*2/3;
        }
      }
/*
* Відновлення розмірів і видимості для всіх ListItems
*/
      private void unExpandViews(boolean deletion) {
        for( int i = 0;; i++ ) {
            View v = getChildAt( i );
            if( v == null ) {
                if( deletion ) {
// Примусове оновлення mItemCount
                    int position = getFirstVisiblePosition();
                    int y = getChildAt( 0 ).getTop();
                    setAdapter( getAdapter() );
                    setSelectionFromTop( position, y );
//
                }
                layoutChildren();
// відтвореня layoutChildren() у разі необхідності
                v = getChildAt( i );
                if( v == null ) {
                    break;
                }
            }
            if( isDraggableRow( v ) ) {
                ViewGroup.LayoutParams params = v.getLayoutParams();
                params.height = mItemHeightNormal;
                v.setLayoutParams( params );
                v.setVisibility( View.VISIBLE );
            }
```

```
        }
      }

      private void doExpansion() {
        int childnum = mDragPos - getFirstVisiblePosition();
        if( mDragPos > mFirstDragPos ) {
            childnum++;
        }
        View first = getChildAt( mFirstDragPos - getFirstVisiblePosition() );
        for( int i = 0;; i++ ) {
            View vv = getChildAt( i );
            if( vv == null ) {
                break;
            }
            int height = mItemHeightNormal;
            int visibility = View.VISIBLE;
            if( vv.equals( first ) ) {
// обробка елемента, який тягнуть
                if( mDragPos == mFirstDragPos ) {
// колишнє місце
                    visibility = View.INVISIBLE;
                } else {
// Чи не над нею
                    height = 1;
                }
            } else if( i == childnum ) {
                if( mDragPos < getCount() - 1 ) {
                    height = mItemHeightExpanded;
                }
            }
            if( isDraggableRow( vv ) ) {
                ViewGroup.LayoutParams params = vv.getLayoutParams();
                params.height = height;
                vv.setLayoutParams( params );
                vv.setVisibility( visibility );
            }
        }
      }
@Override
      public boolean onTouchEvent(MotionEvent ev) {
        if( mGestureDetector != null ) {
            mGestureDetector.onTouchEvent( ev );
        }
if( ( mDragListener != null || mDropListener != null ) && mDragView != null ) {
            int action = ev.getAction();
            switch( action ) {
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                Rect r = mTempRect;
```

```
                mDragView.getDrawingRect( r );
                stopDragging();
if( mRemoveMode == SLIDE_RIGHT && ev.getX() > r.left + ( r.width()*3/4)){
                    if( mRemoveListener != null ) {
                        mRemoveListener.remove( mFirstDragPos );
                    }
                    unExpandViews( true );
} else if( mRemoveMode == SLIDE_LEFT && ev.getX() < r.left + ( r.width() / 4 ) )
{
                    if( mRemoveListener != null ) {
                        mRemoveListener.remove( mFirstDragPos );
                    }
                    unExpandViews( true );
                } else {
if( mDropListener != null && mDragPos >= 0 && mDragPos < getCount() ) {
                        mDropListener.drop( mFirstDragPos, mDragPos );
                    }
                    unExpandViews( false );
                }
                break;
            case MotionEvent.ACTION_DOWN:
            case MotionEvent.ACTION_MOVE:
                int x = (int)ev.getX();
                int y = (int)ev.getY();
                dragView( x, y );
                int itemnum = getItemForPosition( y );
                if( itemnum >= 0 ) {
if( action == MotionEvent.ACTION_DOWN || itemnum != mDragPos ) {
                        if( mDragListener != null ) {
                            mDragListener.drag( mDragPos, itemnum );
                        }
                        mDragPos = itemnum;
                        doExpansion();
                    }
                    int speed = 0;
                    adjustScrollBounds( y );
                    if( y > mLowerBound ) {
// прокрутки списку вгору трохи
                        speed = y > ( mHeight + mLowerBound ) / 2 ? 16 : 4;
                    } else if( y < mUpperBound ) {
// прокрутки списку вниз трохи
                        speed = y < mUpperBound / 2 ? -16 : -4;
                    }
                    if( speed != 0 ) {
                        int ref = pointToPosition( 0, mHeight / 2 );
                        if( ref == AdapterView.INVALID_POSITION ) {
ref = pointToPosition( 0, mHeight / 2 + getDividerHeight() + 64 );
                        }
                        View v = getChildAt( ref - getFirstVisiblePosition() );
```

```
                        if( v != null ) {
                            int pos = v.getTop();
                            setSelectionFromTop( ref, pos - speed );
                        }
                    }
                }
                break;
            }
            return true;
        }
        return super.onTouchEvent( ev );
    }

private void startDragging(Bitmap bm, int x, int y)
{
        stopDragging();
        mWindowParams = new WindowManager.LayoutParams();
        mWindowParams.gravity = Gravity.TOP | Gravity.LEFT;
        mWindowParams.x = x;
        mWindowParams.y = y - mDragPoint + mCoordOffset;
        mWindowParams.height = WindowManager.LayoutParams.WRAP_CONTENT;
        mWindowParams.width = WindowManager.LayoutParams.WRAP_CONTENT;
        mWindowParams.flags = WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE |
WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE
        | WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON |
WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN;
        mWindowParams.format = PixelFormat.TRANSLUCENT;
        mWindowParams.windowAnimations = 0;
        ImageView v = new ImageView( getContext() );
int backGroundColor =getContext().getResources().
                    getColor(R.color.dragndrop_background);
        v.setBackgroundColor( dragndropBackgroundColor );
        v.setImageBitmap( bm );
        mDragBitmap = bm;
        mWindowManager=(WindowManager)getContext().getSystemService( "window" );
        mWindowManager.addView( v, mWindowParams );
        mDragView = v;
    }

    private void dragView(int x, int y) {
        float alpha = 1.0f;
        int width = mDragView.getWidth();
        if( mRemoveMode == SLIDE_RIGHT ) {
            if( x > width / 2 ) {
                alpha = ( (float)( width - x ) ) / ( width / 2 );
            }
            mWindowParams.alpha = alpha;
        } else if( mRemoveMode == SLIDE_LEFT ) {
            if( x < width / 2 ) {
```

```
                alpha = ( (float)x ) / ( width / 2 );
            }
            mWindowParams.alpha = alpha;
        }
        mWindowParams.y = y - mDragPoint + mCoordOffset;
        mWindowManager.updateViewLayout( mDragView, mWindowParams );
    }

    private void stopDragging() {
        if( mDragView != null ) {
WindowManager wm = (WindowManager)getContext().getSystemService( "window" );
            wm.removeView( mDragView );
            mDragView.setImageDrawable( null );
            mDragView = null;
        }
        if( mDragBitmap != null ) {
            mDragBitmap.recycle();
            mDragBitmap = null;
        }
    }

    public void setDragListener(DragListener l) {
                mDragListener = l;
    }

    public void setDropListener(DropListener l)
    {
                mDropListener = l;
    }

    public void setRemoveListener(RemoveListener l) {
                mRemoveListener = l;
    }
    public interface DragListener {
                void drag(int from, int to);
    }
    public interface DropListener {
                void drop(int from, int to);
    }
    public interface RemoveListener
    {
                void remove(int which);
    }
}
```

FILEDIPLOM.JAVA РОБОТА З ФАЙЛАМИ

```
package com.mysq.diplom;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.ArrayList;
import com.mysq.diplom.adapters.DiplomAdapter;
import com.mysq.diplom.adapters.FSAdapter;
import com.mysq.diplom.adapters.FindAdapter;
import com.mysq.diplom.favorites.Favorite;
import com.mysq.diplom.root.MountAdapter;
import com.mysq.diplom.root.RootAdapter;
import com.mysq.diplom.utils.Credentials;
import com.mysq.diplom.utils.Utils;
import android.app.Activity;
import android.app.ActivityManager;
import android.app.Dialog;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.ProgressDialog;
import android.net.Uri;
import android.os.Bundle;
import android.os.Message;
import android.os.Parcelable;
import android.preference.PreferenceManager;
import android.content.ActivityNotFoundException;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.view.ContextMenu;
import android.view.Display;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MenuInflater;
import android.view.KeyEvent;
import android.view.Window;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.inputmethod.InputMethodManager;
import android.util.Log;
import android.widget.AdapterView;
import android.widget.RemoteViews;
import android.widget.Toast;
/*
```

```
* Програмне забезпечення системи керування файловими операціями
* на базі бездротових мережних пристроїв під керуванням ОС Android
* Розробив:Небесний Андрій Вадимович
*/
public class FileDiplom extends Activity implements Diplom, View.OnClickListener
{
    private final static String TAG = "MyDiplomActivity";
public  final static int REQUEST_CODE_PREFERENCES = 1, REQUEST_CODE_SRV_FORM =
2;
public  final static int FIND_ACT = 1017, DBOX_APP = 3592, SMB_ACT = 2751,
FTP_ACT = 4501, SFTP_ACT = 2450;

    private ArrayList<Dialogs> dialogs;
    private ProgressDialog     waitPopup;
    public  Panels  panels;
    private boolean on = false, exit = false, dont_restore = false,
viewActProcessed = false,
                    sxs_auto = true, show_confirm = true, back_exits = false;
    private String  lang = "";
// просто потрібно видати попередження про зміну
    private int     file_exist_resolution = Diplom.UNKNOWN;
    private NotificationManager notMan = null;
    private final static int NOTIF_PROGRESS = 1, NOTIF_DONE = 2;
    private final static String PARCEL = "parcel", MSG = "msg";
    private long    notLastTime = 0;

    public final void showMemory( String s ) {
        final ActivityManager sys =
(ActivityManager)getSystemService(Context.ACTIVITY_SERVICE);
        ActivityManager.MemoryInfo mem = new ActivityManager.MemoryInfo();
        sys.getMemoryInfo(mem);
showMessage(s + "\n Memory: " + mem.availMem + ( mem.lowMemory ? " !!!" : "" ));
    }

    public final void showMessage( String s ) {
        Toast.makeText(this, s, Toast.LENGTH_LONG).show();
    }

    public int getWidth() {
        return panels.getWidth();
    }

    protected final Dialogs getDialogsInstance( int id ) {
        for( int i = 0; i < dialogs.size(); i++ )
            if( dialogs.get(i).getId() == id )
                return dialogs.get(i);
        return null;
    }

    protected final Dialogs obtainDialogsInstance( int id ) {
```

```
        Dialogs dh = getDialogsInstance( id );
        if( dh == null ) {
            dh = new Dialogs( this, id );
            dialogs.add( dh );
        }
        return dh;
    }

    protected final void addDialogsInstance( Dialogs dh ) {
        dialogs.add(dh);
    }

    @Override
    public void onCreate( Bundle savedInstanceState ) {
        super.onCreate( savedInstanceState );

        requestWindowFeature( Window.FEATURE_NO_TITLE );
// показати прогрес, коли немає назви
        requestWindowFeature( Window.FEATURE_INDETERMINATE_PROGRESS );
        dialogs = new ArrayList<Dialogs>( Dialogs.numDialogTypes );
        SharedPreferences sharedPref = PreferenceManager.
                                    getDefaultSharedPreferences(this);
        back_exits = sharedPref.getBoolean( "exit_on_back", false );
        lang = sharedPref.getString( "language", "" );
        Utils.changeLanguage( this );
        String panels_mode = sharedPref.getString( "panels_sxs_mode", "a" );
        sxs_auto = panels_mode.equals( "a" );
        boolean sxs = sxs_auto ? getRotMode() : panels_mode.equals( "y" );
        panels = new Panels( this, sxs );
        setConfirmMode( sharedPref );
notMan = (NotificationManager)getSystemService( Context.NOTIFICATION_SERVICE );
    }
    @Override
    protected void onStart() {
        Log.v( TAG, "Старт\n" );
        super.onStart();
        on = true;
        if( dont_restore )
            dont_restore = false;
        else {
            Utils.changeLanguage( this );
            Intent intent = getIntent();

            SharedPreferences prefs = getPreferences( MODE_PRIVATE );
            Panels.State s = panels.new State();
            s.restore( prefs );

            String action = intent.getAction();
            Log.i( TAG, "дія: " + action );
```

```
            int use_panel = -1;
            Uri uri = intent.getData();
            if( uri != null && !viewActProcessed && Intent.ACTION_VIEW.equals(
action ) ) {
                Log.v( TAG, "Intent URI: " + uri );
                Credentials crd = null;
                try {
crd = (Credentials)intent.getParcelableExtra( Credentials.KEY );
                } catch( Throwable e ) {
                    Log.e( TAG, "on extracting credentials from an intent", e );
                }

                String file_name = null;
                String type = intent.getType();
                if( "application/x-zip-compressed".equals( type ) ||
                                "application/zip".equals( type ) )
                    uri = uri.buildUpon().scheme( "zip" ).build();
else if( ContentResolver.SCHEME_CONTENT.equals( uri.getScheme() ) ) {
                     unknown content is being passed. Let's just save it
                    try {
                   InputStream is = getContentResolver().openInputStream( uri );
                        if( is != null ) {
                          File dwf = new File( Panels.DEFAULT_LOC, "download" );
                            if( !dwf.exists() )
                                dwf.mkdirs();

                            String fn = uri.getLastPathSegment();
                            File f = null;
                            for( int i = 0; i < 99; i++ ) {
                                file_name = i == 0 ? fn : fn + "_" + i;
                                f = new File( dwf, file_name );
                                if( f.exists() ) continue;
                                if( f.createNewFile() )
                                    break;
                                f = null;
                            }
                            if( f!= null ) {
                                BufferedOutputStream bos = new
BufferedOutputStream( new FileOutputStream( f ), 8192 );
                                byte[] buf = new byte[4096];
                                int n;
                                while( ( n = is.read( buf ) ) != -1 )
                                    bos.write( buf, 0, n );
                                bos.close( );
                                is.close( );
                                uri = Uri.fromFile( dwf );
                    showMessage( getString( R.string.copied_f, f.toString() ) );
                            }
                            else
```

```
                                showError( getString( R.string.not_accs, fn ) );
                            }
                        } catch( Exception e ) {
                            showError( getString( R.string.not_accs, "" ) );
// докладніше
                        }
                    }
                    use_panel = Panels.LEFT;
                    panels.Navigate( use_panel, uri, crd, file_name );
                    viewActProcessed = true;
                }
                panels.setState( s, use_panel );
                final String FT = "first_time";
                if( !viewActProcessed && prefs.getBoolean( FT, true ) ) {
                    SharedPreferences.Editor editor = prefs.edit();
                    editor.putBoolean( FT, false );
                    editor.commit();
                    showInfo( getString( R.string.keys_text) );
                }
                if( use_panel >= 0 && viewActProcessed )
                    panels.setPanelCurrent( use_panel );
                if( Intent.ACTION_SEARCH_LONG_PRESS.equals( action ) ) {
                    showSearchDialog();
                    return;
                }
            }
        }

        @Override
        protected void onPause() {
            Log.v( TAG, "Pausing\n");
            super.onPause();
            on = false;
            SharedPreferences.Editor editor = getPreferences( MODE_PRIVATE ).edit();
            Panels.State s = panels.getState();
            s.store( editor );
            editor.commit();
        }

        @Override
        protected void onResume() {
            Log.v( TAG, "Resuming\n");
            super.onResume();
            on = true;
        }

        @Override
        protected void onStop() {
            Log.v( TAG, "Stopping\n");
```

```
        super.onStop();
        on = false;
    }

    @Override
    protected void onDestroy() {
        Log.v( TAG, "Destroying\n");
        on = false;
        super.onDestroy();
        if( notMan != null ) notMan.cancelAll();
        panels.Destroy();
        if( isFinishing() && exit ) {
            Log.i( TAG, "Good bye cruel world...");
            System.exit( 0 );
        }
    }
    @Override
    protected void onSaveInstanceState( Bundle outState ) {
        Log.i( TAG, "Saving Instance State");
        Panels.State s = panels.getState();
        s.store( outState );
        super.onSaveInstanceState( outState );
    }

    @Override
    protected void onRestoreInstanceState( Bundle savedInstanceState ) {
        Log.i( TAG, "Restoring Instance State");
        if( savedInstanceState != null ) {
            Panels.State s = panels.new State();
            s.restore( savedInstanceState );
            panels.setState( s, -1 );
        }
        super.onRestoreInstanceState( savedInstanceState );
    }

    @Override
    public void onConfigurationChanged( Configuration newConfig ) {
        Utils.changeLanguage( this );
        super.onConfigurationChanged( newConfig );
        panels.setLayoutMode( sxs_auto ? newConfig.orientation ==
Configuration.ORIENTATION_LANDSCAPE : panels.getLayoutMode() );
// Перевіряє наявність апаратної клавіатури
     if (newConfig.hardKeyboardHidden == Configuration.HARDKEYBOARDHIDDEN_NO) {
            Toast.makeText(this, "keyboard visible", Toast.LENGTH_SHORT).show();
} else if (newConfig.hardKeyboardHidden == Configuration.HARDKEYBOARDHIDDEN_YES)
{
            Toast.makeText(this, "keyboard hidden", Toast.LENGTH_SHORT).show();
        }
    }
```

```
    @Override
    public void onCreateContextMenu( ContextMenu menu, View v, ContextMenuInfo
menuInfo ) {
        try {
            Utils.changeLanguage( this );
            AdapterView.AdapterContextMenuInfo acmi =
(AdapterView.AdapterContextMenuInfo)menuInfo;
            menu.setHeaderTitle( getString( R.string.operation ) );
            DiplomAdapter ca = panels.getListAdapter( true );
            ca.populateContextMenu( menu, acmi, panels.getNumItemsChecked() );
        }
        catch( Exception e ) {
            Log.e( TAG, "onCreateContextMenu()", e );
        }
    }

    @Override
    public boolean onContextItemSelected( MenuItem item ) {
        try {
            panels.resetQuickSearch();
            AdapterView.AdapterContextMenuInfo info;
            info = (AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
            if( info == null ) return false;
            panels.setSelection( info.position );
            int item_id = item.getItemId();
            if( OPEN == item_id )
                panels.openItem( info.position );
            else
                dispatchCommand( item_id );
            return true;
        } catch( Exception e ) {
            Log.e(TAG, "onContextItemSelected()", e);
            return false;
        }
    }

    @Override
    protected Dialog onCreateDialog( int id ) {
        if( !on ) {
            Log.e( TAG, "onCreateDialog() is called when the activity is down"
);
        }
        Dialogs dh = obtainDialogsInstance( id );
        Dialog d = dh.createDialog( id );
        return d != null ? d : super.onCreateDialog( id );
    }

    @Override
```

```
    protected void onPrepareDialog( int id, Dialog dialog ) {
        Dialogs dh = getDialogsInstance( id );
        if( dh != null )
            dh.prepareDialog( id, dialog );
        super.onPrepareDialog( id, dialog );
    }

    @Override
    public boolean onCreateOptionsMenu( Menu menu ) {
        try {
            Utils.changeLanguage( this );
// обраного до пункту меню, XML ресурсу
            MenuInflater inflater = getMenuInflater();
            inflater.inflate( R.menu.menu, menu );
            return true;
        } catch( Error e ) {
            e.printStackTrace();
        }
        return false;
    }

    @Override
    public boolean onMenuItemSelected( int featureId, MenuItem item ) {
        panels.resetQuickSearch();
        boolean processed = super.onMenuItemSelected( featureId, item );
        if( !processed )
            dispatchCommand( item.getItemId() );
        return true;
    }

    @Override
    public void issue( Intent in, int ret ) {
        if( in == null ) return;
        try {
            if( ret == 0 )
                startActivity( in );
            else
                startActivityForResult( in, ret );
        } catch( Exception e ) {
            Log.e( TAG, in.getDataString(), e );
        }
    }

    @Override
    protected void onActivityResult( int requestCode, int resultCode, Intent
data ) {
        super.onActivityResult( requestCode, resultCode, data );
        switch( requestCode ) {
        case REQUEST_CODE_PREFERENCES: {
```

```
                SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(this);
                back_exits = sharedPref.getBoolean( "exit_on_back", false );
                String lang_ = sharedPref.getString( "language", "" );
                if( !lang.equalsIgnoreCase( lang_ ) ) {
                    lang = lang_;
                    Utils.changeLanguage( this );
                    showMessage( getString( R.string.restart_to_apply_lang ) );
                    exit = true;
                }
                panels.applySettings( sharedPref, false );
                String panels_mode = sharedPref.getString( "panels_sxs_mode",
"a" );
                sxs_auto = panels_mode.equals( "a" );
                boolean sxs = sxs_auto ? getRotMode() : panels_mode.equals( "y"
);
                panels.setLayoutMode( sxs );
             panels.showToolbar( sharedPref.getBoolean("show_toolbar", true ) );
                setConfirmMode( sharedPref );
            }
            break;
        case REQUEST_CODE_SRV_FORM: {
                if( resultCode == RESULT_OK ) {
                    dont_restore = true;
                    Uri uri = data.getData();
                    if( uri != null ) {
                        Credentials crd = null;
                        try {
                            crd = (Credentials)data.getParcelableExtra(
Credentials.KEY );
                        } catch( Exception e ) {
                          Log.e( TAG, "on taking credentials from parcel", e );
                        }
                        panels.Navigate( panels.getCurrent(), uri, crd, null );
                    }
                }
            }
            break;
        case R.id.create_shortcut:
            if( data != null ) {
              data.setAction( "com.android.launcher.action.INSTALL_SHORTCUT" );
                sendBroadcast( data );
            }
            break;
        case OPERATION_COMPLETED_REFRESH_REQUIRED:
            Log.i( TAG, "An activity ends. Refresh required." );
            panels.refreshLists( null );
            break;
        }
```

```
    }

    @Override
    public boolean onKeyDown( int keyCode, KeyEvent event ) {
Log.v( TAG, "global key:" + keyCode + ", number:" + event.getNumber() + ",
uchar:" + event.getUnicodeChar() );
        char c = (char)event.getUnicodeChar();
        panels.resetQuickSearch();
        switch( c ) {
        case '=':
            panels.makeOtherAsCurrent();
            return true;
        case '&':
            openPrefs();
            return true;
        case '/':
            showSearchDialog();
            return true;
        case '1':
            showInfo(getString(R.string.keys_text));
            return true;
        case '9':
            openPrefs();
            return true;
        case '0':
            exit = true;
            finish();
            return true;
        }
        switch( keyCode ) {
        case KeyEvent.KEYCODE_TAB:
            panels.togglePanels( false );
            return true;
        case KeyEvent.KEYCODE_VOLUME_DOWN:
            if( panels.volumeLegacy ) {
                panels.togglePanels( false );
                return true;
            }
            break;
        case KeyEvent.KEYCODE_SEARCH:
            showSearchDialog();
            return false;
        }
        return super.onKeyDown( keyCode, event );
    }

    @Override
    public boolean onKeyUp( int keyCode, KeyEvent event ) {
```

```
        if( keyCode == KeyEvent.KEYCODE_VOLUME_UP || keyCode ==
KeyEvent.KEYCODE_VOLUME_DOWN )
            return true;

        return super.onKeyUp(keyCode, event);
    }

    @Override
    public void onClick( View button ) {
        panels.resetQuickSearch();
        if( button == null )
            return;
        dispatchCommand( button.getId() );
    }

    public final boolean backExit() {
        if( back_exits ) {
            finish();
            return true;
        }
        return false;
    }

    public void dispatchCommand( int id ) {
        try {
            Utils.changeLanguage( this );
            switch( id ) {
            case R.id.keys:
            case R.id.F1:
                showInfo( getString( R.string.keys_text ) );
                break;
            case R.id.F3:
                panels.openForView();
                break;
            case R.id.F4:
                panels.openForEdit(null);
                break;
            case R.id.F2:
            case R.id.new_zip:
            case R.id.F5:
            case R.id.F6:
            case R.id.F8:
                if( panels.getNumItemsSelectedOrChecked() > 0 )
                    showDialog( id );
                else
                    showMessage( getString( R.string.no_items ) );
                break;
            case R.id.new_file:
            case R.id.SF4:
```

```
                case R.id.F7:
                case R.id.about:
                case R.id.donate:
                    showDialog( id );
                    break;
                case R.id.prefs:
                case R.id.F9:
                    openPrefs();
                    break;
                case R.id.exit:
                case R.id.F10:
                    exit = true;
                    finish();
                    break;
                case R.id.oth_sh_this:
                case R.id.eq:
                    panels.makeOtherAsCurrent();
                    break;
                case R.id.toggle_panels_mode:
                    panels.togglePanelsMode();
                    break;
                case R.id.tgl:
                    panels.togglePanels(true);
                    break;
                case R.id.sz:
                    panels.showSizes();
                    break;
                case R.id.home:
                    Navigate( Uri.parse( "home:" ), null, null );
                    break;
                case R.id.favs:
                    Navigate( Uri.parse( "favs:" ), null, null );
                    break;
                case  R.id.sdcard:
                    Navigate( Uri.parse( Panels.DEFAULT_LOC ), null, null );
                    break;
                case  R.id.root: {
                        Uri cu = panels.getFolderUriWithAuth( true );
                        Navigate( Uri.parse( RootAdapter.DEFAULT_LOC +
                          ( cu == null || cu.getScheme() != null &&
cu.getScheme().length() > 0 ? "" : cu.getPath() ) ), null, null );
                    }
                    break;
                case  R.id.mount:
                    Navigate( Uri.parse( MountAdapter.DEFAULT_LOC ), null, null );
                    break;

                case FTP_ACT: {
                        Intent i = new Intent( this, ServerForm.class );
```

```
            i.putExtra( "schema", "ftp" );
            startActivityForResult( i, REQUEST_CODE_SRV_FORM );
        }
        break;
    case SFTP_ACT: {
            Intent i = new Intent( this, ServerForm.class );
            i.putExtra( "schema", "sftp" );
            startActivityForResult( i, REQUEST_CODE_SRV_FORM );
        }
        break;
    case SMB_ACT: {
            Intent i = new Intent( this, ServerForm.class );
            i.putExtra( "schema", "smb" );
            startActivityForResult( i, REQUEST_CODE_SRV_FORM );
        }
        break;
    case R.id.search:
        showSearchDialog();
        break;
    case R.id.enter:
        panels.openGoPanel();
        break;
    case R.id.add_fav:
        panels.addCurrentToFavorites();
        break;
    case R.id.by_name:
        panels.changeSorting( DiplomAdapter.SORT_NAME );
        break;
    case R.id.by_ext:
        panels.changeSorting( DiplomAdapter.SORT_EXT );
        break;
    case R.id.by_size:
        panels.changeSorting( DiplomAdapter.SORT_SIZE );
        break;
    case R.id.by_date:
        panels.changeSorting( DiplomAdapter.SORT_DATE );
        break;
    case R.id.refresh:
        panels.refreshLists( null );
        break;
    case R.id.sel_all:
        showDialog( Dialogs.SELECT_DIALOG );
        break;
    case R.id.uns_all:
        showDialog( Dialogs.UNSELECT_DIALOG );
        break;
    case R.id.online: {
            Intent intent = new Intent( Intent.ACTION_VIEW );
          intent.setData( Uri.parse( getString( R.string.help_uri ) ) );
```

```
                    startActivity( intent );
                }
                break;
            case SEND_TO:
                panels.tryToSend();
                break;
            case OPEN_WITH:
                panels.tryToOpen();
                break;
            case COPY_NAME:
                panels.copyName();
                break;
            case FAV_FLD:
                panels.faveSelectedFolder();
                break;
            case R.id.softkbd:
                InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
                imm.toggleSoftInput(0, 0);
                break;
            case  R.id.hidden:
                panels.toggleHidden();
                break;
            case  R.id.rescan:
                sendBroadcast( new Intent( Intent.ACTION_MEDIA_MOUNTED,
Uri.parse( "file://" + Panels.DEFAULT_LOC ) ) );
                break;
            default:
                DiplomAdapter ca = panels.getListAdapter( true );
                if( ca != null )
                    ca.doIt( id, panels.getSelectedOrChecked() );
            }
        }
        catch( Throwable e ) {
            e.printStackTrace();
        }
    }
    private final void openPrefs() {
        try {
Intent launchPreferencesIntent = new Intent().setClass( this, Prefs.class );
startActivityForResult( launchPreferencesIntent, REQUEST_CODE_PREFERENCES );
        } catch( Error e ) {
            Log.e( TAG, "Preferences can't open", e );
        }
    }

    private final void showSearchDialog() {
        DiplomAdapter ca = panels.getListAdapter( true );
        if( ca instanceof FSAdapter || ca instanceof FindAdapter ) {
```

```
            String cur_s = ca.toString();
            if( cur_s != null ) {
                Uri cur_uri = Uri.parse( cur_s );
                if( cur_uri != null ) {
                    String cur_path = cur_uri.getPath();
                    if( cur_path != null ) {
                        Dialogs dh = obtainDialogsInstance( FIND_ACT );
                        dh.setCookie( cur_path );
                        showDialog( FIND_ACT );
                        return;
                    }
                }
            }
            showMessage( "Error" );
        }
        else
            showError( getString( R.string.find_on_fs_only ) );
    }
     /*
     * Дипломна реалізація інтерфейсу
     */
    @Override
    public void Navigate( Uri uri, Credentials crd, String posTo ) {
         panels.Navigate( panels.getCurrent(), uri, crd, posTo );
    }

    @Override
    public void Open( Uri uri, Credentials crd ) {
        try {
            if( uri == null ) return;
            String scheme = uri.getScheme();
            String path = uri.getPath();
            String ext = Utils.getFileExt( "zip".equals( scheme ) ?
uri.getFragment() : path );
            String mime = Utils.getMimeByExt( ext );
            if( scheme == null || scheme.length() == 0 ) {
                Intent i = new Intent( Intent.ACTION_VIEW );
                Intent op_intent = getIntent();
                if( op_intent != null ) {
                    String action = op_intent.getAction();
                    if( Intent.ACTION_PICK.equals( action ) ) {
                        i.setData( uri );
                        setResult( RESULT_OK, i );
                        finish();
                        return;
                    }
                    if( Intent.ACTION_GET_CONTENT.equals( action ) ) {
                       i.setData( Uri.parse( FileProvider.URI_PREFIX + path ) );
                        setResult( RESULT_OK, i );
```

```
                        finish();
                        return;
                    }
                }
                if( ext != null && ext.compareToIgnoreCase( ".zip" ) == 0 ) {
Navigate( uri.buildUpon().scheme( "zip" ).build(), null, null );
                    return;
                }
i.setDataAndType( uri.buildUpon().scheme( "file" ).authority( "" ).build(), mime
);
i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.
           FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET  );
           | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED
                startActivity( i );
            }
            else
if( mime != null && ( mime.startsWith( "audio" ) || mime.startsWith( "video" ) )
) {
                startService( new Intent( this, StreamServer.class ) );
                Intent i = new Intent( Intent.ACTION_VIEW );
String http_url = "http://127.0.0.1:5322/" + Uri.encode( uri.toString() );
                if( crd != null )
                    StreamServer.credentials = crd;
                Log.d( TAG, "Stream " + mime + " from: " + http_url );
                i.setDataAndType( Uri.parse( http_url ), mime );
i.setFlags( Intent.FLAG_ACTIVITY_NEW_TASK |
            Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET  );
                startActivity( i );
                return;
            }

        } catch( ActivityNotFoundException e ) {
showMessage("Application for open '" + uri.toString() + "' is not available, ");
        } catch( Exception e ) {
            Log.e( TAG, uri.toString(), e );
        }
    }
    @Override
    public Context getContext() {
        return this;
    }

    @Override
    protected void onNewIntent( Intent intent ) {
        on = true;
        super.onNewIntent( intent );
        try {
            Message msg = intent.getParcelableExtra( PARCEL );
            if( msg != null ) {
```

```
                notifyMe( msg );
                return;
            }
            Log.v( TAG, "No parcel found in the intent." );
        } catch( Exception e ) {
            Log.e( TAG, "Can't extract a parcel from intent", e );
        }
    }

    @Override
    public boolean notifyMe( Message progress ) {
        final boolean TERMINATE = true, CONTINUE = false;
        String string = null;
        try {
            if( progress.obj != null ) {
                if( progress.obj instanceof String )
                    string = (String)progress.obj;
                else if( progress.obj instanceof Bundle ) {
                    Bundle b = (Bundle)progress.obj;
                    string = b.getString( MSG );
                }
            }
            Bundle b = progress.getData();
            String cookie = b != null ? b.getString( NOTIFY_COOKIE ) : null;
            if( progress.what == Diplom.OPERATION_STARTED ) {
                setProgressBarIndeterminateVisibility( true );
                if( string != null && string.length() > 0 )
                    showMessage( string );
                return CONTINUE;
            }
            Dialogs dh = null;
            if( progress.what == OPERATION_IN_PROGRESS ) {
                if( progress.arg1 >= 0 ) {
                    if( on ) {
                        dh = obtainDialogsInstance( Dialogs.PROGRESS_DIALOG );
                        if( dh != null ) {
                            dh.showDialog();
dh.setProgress( string, progress.arg1, progress.arg2, b != null ? b.getInt(
                NOTIFY_SPEED):0);
                        }
                    }
                    else {
                        if( string != null && string.length() > 0 )
                          setSystemOngoingNotification( string, progress.arg1 );
                    }
                } else {
                    if( waitPopup == null )
waitPopup=ProgressDialog.show(this,"",getString(R.string.wait),true,true);
                }
```

```
                return CONTINUE;
            }
            dh = getDialogsInstance( Dialogs.PROGRESS_DIALOG );
            if( dh != null )
                dh.cancelDialog();
            if( notMan != null ) notMan.cancel( NOTIF_PROGRESS );
            if( !on ) {
                setSystemNotification( progress );
                return progress.what != OPERATION_SUSPENDED_FILE_EXIST;
            }
            if( waitPopup != null ) {
                waitPopup.cancel();
                waitPopup = null;
            }
            setProgressBarIndeterminateVisibility( false );
            panels.operationFinished();
            switch( progress.what ) {
            case OPERATION_SUSPENDED_FILE_EXIST: {
                    dh = obtainDialogsInstance( Dialogs.FILE_EXIST_DIALOG );
                    dh.setMessageToBeShown( string, null );
                    dh.showDialog();
                }
                return CONTINUE;
            case OPERATION_FAILED:
                if( Utils.str( cookie ) ) {
                    int which_panel = cookie.charAt( 0 ) == '1' ? 1 : 0;
panels.setPanelTitle( getString( R.string.fail ), which_panel );
                }
                if( Utils.str( string ) )
                    showError( string );
                panels.redrawLists();
                return TERMINATE;
            case OPERATION_FAILED_LOGIN_REQUIRED:
                if( string != null ) {
                    dh = obtainDialogsInstance( Dialogs.LOGIN_DIALOG );
                    if( b != null ) {
                        Parcelable crd_p = b.getParcelable( NOTIFY_CRD );
                        if( crd_p != null && crd_p instanceof Credentials )
                            dh.setCredentials( (Credentials)crd_p, Utils.str(
cookie ) ? cookie.charAt( 0 ) == '1' ? 1 : 0 : -1 );
                    }
                    dh.setMessageToBeShown( string, cookie );
                    showDialog( Dialogs.LOGIN_DIALOG );
                }
                return TERMINATE;
            case OPERATION_COMPLETED_REFRESH_REQUIRED:
                String posto = b != null ? b.getString( NOTIFY_POSTO ) : null;
                panels.refreshLists( posto );
                break;
```

```
            case OPERATION_COMPLETED:
                if( Utils.str( cookie ) ) {
                    int which_panel = cookie.charAt( 0 ) == '1' ? 1 : 0;
                    String item_name = cookie.substring( 1 );
                    panels.recoverAfterRefresh( item_name, which_panel );
                }
                else
                    panels.recoverAfterRefresh( null, -1 );
                break;
            }
if( ( show_confirm || progress.arg1 == OPERATION_REPORT_IMPORTANT ) &&
     Utils.str( string ) )
                showInfo( string );
        } catch( Exception e ) {
            Log.e( TAG, string, e );
        }
        return TERMINATE;
    }

private PendingIntent getPendingIntent( Parcelable parcel ) {
        Intent intent = new Intent( this, FileDiplom.class );
        intent.setFlags( Intent.FLAG_ACTIVITY_CLEAR_TOP |
        Intent.FLAG_ACTIVITY_SINGLE_TOP );
        intent.setAction( Intent.ACTION_MAIN );
        if( parcel != null )
            intent.putExtra( PARCEL, parcel );
        return PendingIntent.getActivity( this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT );
    }

private void setSystemOngoingNotification( String str, int p ) {
        if( notMan == null || str == null ) return;
        long cur_time = System.currentTimeMillis();
        if( notLastTime + 1000 > cur_time ) return;
        notLastTime = cur_time;
Notification notification = new Notification( R.drawable.icon, getString(
                            R.string.inprogress ), cur_time );
        notification.contentIntent = getPendingIntent( null );
        notification.flags |= Notification.FLAG_ONGOING_EVENT;
RemoteViews not_view = new RemoteViews( getPackageName(), R.layout.progress );
        not_view.setTextColor( R.id.text, 0xFF000000 );
        not_view.setTextViewText( R.id.text, str.replace( "\n", " " ) );
        not_view.setProgressBar( R.id.progress_bar, 100, p, false );
        not_view.setTextColor( R.id.percent, 0xFF000000 );
        not_view.setTextViewText( R.id.percent, "" );
        notification.contentView = not_view;
        notMan.notify( NOTIF_PROGRESS, notification );
    }
```

```
    private void setSystemNotification( Message msg ) {
        if( notMan == null || msg == null ) return;
        String str;
        try {
            str = (String)msg.obj;
        } catch( Exception e ) {
            str = "Unknown Event";
        }
Notification notification = new Notification( R.drawable.icon, str,
                            System.currentTimeMillis() );
        Bundle b = new Bundle( 1 );
        b.putString( MSG, str );
        msg.obj = b;
// повідомлення для виведення parcelable
notification.setLatestEventInfo(this, getString( R.string.app_name ), str,
                                getPendingIntent( msg ) );
notification.flags |= Notification.FLAG_ONLY_ALERT_ONCE |
                      Notification.FLAG_AUTO_CANCEL;
        notMan.notify( NOTIF_DONE, notification );
    }

    public void setResolution( int r ) {
        synchronized( this ) {
            file_exist_resolution = r;
            notify();
        }
    }

    @Override
    public int getResolution() {
        int r = file_exist_resolution;
        file_exist_resolution = Diplom.UNKNOWN;
        return r;
    }

    @Override
    public void showError( String errMsg ) {
        try {
            if( !on ) return;
            Dialogs dh = obtainDialogsInstance( Dialogs.ALERT_DIALOG );
            dh.setMessageToBeShown( errMsg, null );
            dh.showDialog();
            return;
        } catch( Exception e ) {
            e.printStackTrace();
        }
        showMessage( errMsg );
    }
```

```
    @Override
    public void showInfo( String msg ) {
        if( !on ) return;
        if( msg.length() < 32 )
            showMessage( msg );
        else {
            Dialogs dh = obtainDialogsInstance( Dialogs.INFO_DIALOG );
            dh.setMessageToBeShown( msg, null );
            dh.showDialog();
        }
    }

    public final void startViewURIActivity( int res_id ) {
        Intent i = new Intent( Intent.ACTION_VIEW );
        i.setData( Uri.parse( getString( res_id ) ) );
        startActivity( i );
    }

    private final boolean getRotMode() {
        boolean sideXside = false;
        try {
            Display disp = getWindowManager().getDefaultDisplay();
            sideXside = disp.getWidth() > disp.getHeight();
        } catch( Exception e ) {
            Log.e( TAG, "", e );
        }
        return sideXside;
    }

    private final void setConfirmMode( SharedPreferences sharedPref ) {
        show_confirm = sharedPref.getBoolean("show_confirm", true );
    }
}
```

DIPLOM.PROJECT ПРОЕКТ

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>DIPLOM</name>
	<comment></comment>
	<projects>
	</projects>
	<buildSpec>
		<buildCommand>
	<name>com.android.ide.eclipse.adt.ResourceManagerBuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
		<buildCommand>
			<name>com.android.ide.eclipse.adt.PreCompilerBuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
		<buildCommand>
			<name>org.eclipse.jdt.core.javabuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
		<buildCommand>
			<name>com.android.ide.eclipse.adt.ApkBuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
	</buildSpec>
	<natures>
		<nature>com.android.ide.eclipse.adt.AndroidNature</nature>
		<nature>org.eclipse.jdt.core.javanature</nature>
	</natures>
</projectDescription>
```