Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Лиськова Юрія Юрійовича*
(ПІБ)

академічної групи *121-19з-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка екранної клавіатури для операційної системи Android на мові Java*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст.викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19з-1* (група) *Лиськова Юрія Юрійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка екранної клавіатури для операційної системи Android на мові Java*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Лиськов Ю.Ю.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 85 с., 35 рис., 3 дод., 29 джерел.

Об'єкт розробки: екранна клавіатура, що дозволяє користувачеві вводити інформацію на різних пристроях.

Мета кваліфікаційної роботи: забезпечити зручний інструментарій для вводу інформації на телевізійних приставках, телевізорах зі смарт функціоналом, планшетах та смартфонах.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Актуальність даного програмного забезпечення визначається необхідністю користувачів мати інструмент для вводу інформації на різних пристроях.

Список ключових слів: OS, SDK, API, IME, UI, UML, PDA.

ABSTRACT

Explanatory note: 85 pages, 35 pics, 3 apps, 29 sources.

Object of development: an on-screen keyboard for for typing text, numbers, and symbols into application software on variety of devices.

The purpose of the diploma project: to provide convenient tool for for typing text, numbers, and symbols into application software on variety of devices such as set-top boxes, smartphones, tablets.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides the characteristics of the parameters of hardware, describes the call and application load, describes the program .

In the economic section, the complexity of the developed information subsystem is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The relevance of this software is determined by the need for users to have convenient tool for typing text, numbers, and symbols into application software on variety of devices.

List of keywords: OS, SDK, API, IME, UI, UML, PDA.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

OS — Operation System (операційна система).

SDK — Software Development Kit (комплект розробки програмного забезпечення).

API — Application Programming Interface (інтерфейс програмування додатків).

IME — Input Method Editor (редактор методів введення).

UI — User Interface (інтерфейс користувача).

PDA — Personal Digiatal Assistent (кишеньковий персональний комп'ютер).

UML — Unified Modeling Language (уніфікована мова моделювання).

JVM — Java Virtual Machine (віртуальна машина Java).

ART — Android Runtime (середовище виконання Android).

APK — Android Package Kit (формат файлів-застосунків для Android).

Google Play — крамниця застосунків від Google.

ПЗ — програмне забезпечення.

ЕОМ — електронно-обчислювальна машина.

Android TV — операційна система для телевізорів, яка була розроблена компанією Google на базі мобільної операційної системи Android.

ВСТУП

Екранна клавіатура, також відома як віртуальна клавіатура, — це екранне представлення фізичної клавіатури, яке дозволяє користувачам вводити текст або команди за допомогою пальців або стилуса на пристрої з сенсорним екраном, наприклад смартфоні, планшеті чи комп'ютері [1].

На відміну від фізичних клавіатур, які мають фізичні клавіші, екранна клавіатура складається з візуального макета клавіш на екрані, який відповідає символам на фізичній клавіатурі. Клавіші на екранній клавіатурі можуть мати різне розташування та розмір залежно від розміру екрана та орієнтації пристрою.

Екранні клавіатури зазвичай використовуються в пристроях із сенсорним екраном, де фізична клавіатура непрактична або неможлива. Вони також корисні в ситуаціях, коли фізична клавіатура недоступна, наприклад, на громадських комп'ютерах або кіосках. Екранні клавіатури також можуть підтримувати різні мови та методи введення, що робить їх універсальними для використання в різних налаштуваннях.

Екранні клавіатури часто мають такі функції, як автовиправлення та інтелектуальне введення тексту, які допомагають користувачам вводити текст ефективніше й точніше. Автовиправлення автоматично виправляє поширені помилки введення, тоді як інтелектуальний текст пропонує найбільш вірогідне слово чи фразу на основі контексту того, що було введено до цього часу.

Крім того, екранні клавіатури також можуть підтримувати жести, такі як проведення пальцем або торкання, для введення тексту та навігації. Гортання передбачає перетягування пальця по клавіатурі для створення слів, тоді як торкання передбачає натискання окремих клавіш для введення.

Екранні клавіатури стають дедалі важливішими, оскільки все більше людей використовують мобільні пристрої та покладаються на сенсорні екрани для доступу до інформації та спілкування. Вони забезпечують зручний і універсальний спосіб введення тексту та команд, дозволяючи використовувати

пристрої для широкого спектру завдань, від надсилання повідомлень і електронних листів до редагування документів і заповнення форм.

Однією з переваг екранних клавіатур є те, що їх можна налаштувати відповідно до конкретних потреб користувачів. Наприклад, користувачі можуть налаштувати розмір і розкладку клавіатури, щоб полегшити введення тексту на менших екранах, або вони можуть увімкнути такі функції, як тактильний зворотний зв'язок, щоб забезпечити тактильний зворотний зв'язок під час введення.

Екранні клавіатури також мають перевагу в тому, що вони здатні адаптуватися до різних методів введення. Наприклад, користувачі можуть перемикатися між набором тексту пальцями та використанням стилуса або використовувати програмне забезпечення для розпізнавання голосу для введення тексту.

Незважаючи на свої переваги, екранні клавіатури можуть бути менш точними та повільнішими, ніж фізичні клавіатури, особливо для користувачів, які звикли друкувати за допомогою фізичних клавіш. Вони також можуть займати цінний простір на екрані, що може бути проблемою на невеликих пристроях [1].

Загалом екранні клавіатури стали важливим інструментом для користувачів мобільних пристроїв, забезпечуючи зручний і гнучкий спосіб введення тексту та команд на різних пристроях. Оскільки технологія продовжує розвиватися, цілком імовірно, що програмні клавіатури стануть ще більш вдосконаленими та універсальними, забезпечуючи нові способи взаємодії з цифровими пристроями.

Основною метою розробки екранної клавіатури для Android є надання зручного та ефективного метода введення для користувачів пристроїв Android. Екранні клавіатури Android інтуїтивно зрозумілі, легко настроюються та пропонують низку функцій і опцій для покращення взаємодії з користувачем.

Беручи це все до уваги було сформовано тему кваліфікаційної роботи: «Розробка екранної клавіатури для операційної системи Android на мові Java»

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості с предметної галузі

Віртуальні клавіатури зазвичай використовуються як екранний метод введення в пристроях без фізичної клавіатури, де немає місця для неї, таких як кишеньковий комп’ютер, персональний цифровий помічник (PDA), планшет або мобільний телефон із сенсорним екраном. Текст зазвичай вводиться торканням віртуальної клавіатури або обведенням пальців. Віртуальні клавіатури також використовуються як функції програмного забезпечення емуляції для систем, які мають менше кнопок, ніж клавіатура комп’ютера [1].

Перші екранні клавіатури були розроблені наприкінці 1990-х років для пристроїв із сенсорним екраном. Ці клавіатури були простими та мали обмежену функціональність, але вони проклали шлях для майбутніх розробок.

На початку 2000-х років була представлена технологія прогнозування тексту, яка пропонувала слова або фрази під час введення користувачем. Ця технологія значно підвищила швидкість і точність набору тексту на екрані. Розкладка клавіатури QWERTY, яка вперше була розроблена для друкарських машинок у 19 столітті, стала стандартом для екранних клавіатур (рис. 1.1.).

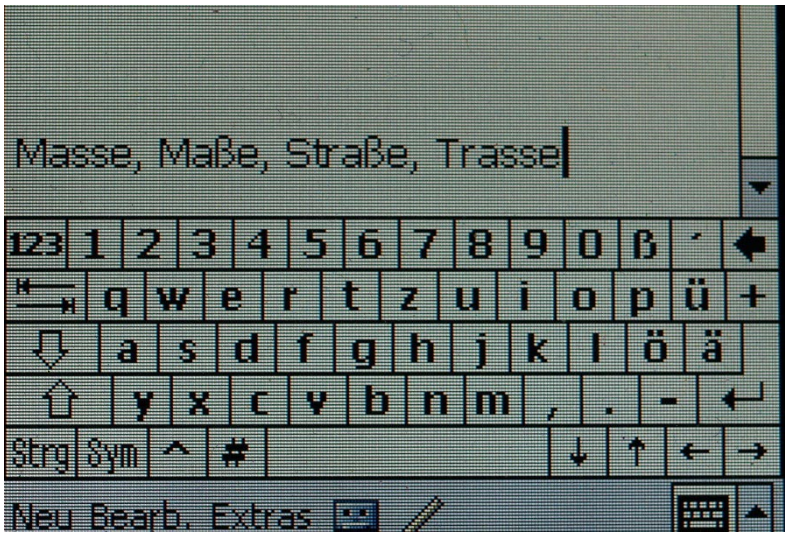

Рис. 1.1. Віртуальна клавіатура на Pocket PC.

Розкладка QWERTY базується на тому, як часто використовується та чи інша літера [18] (рис. 1.2.).

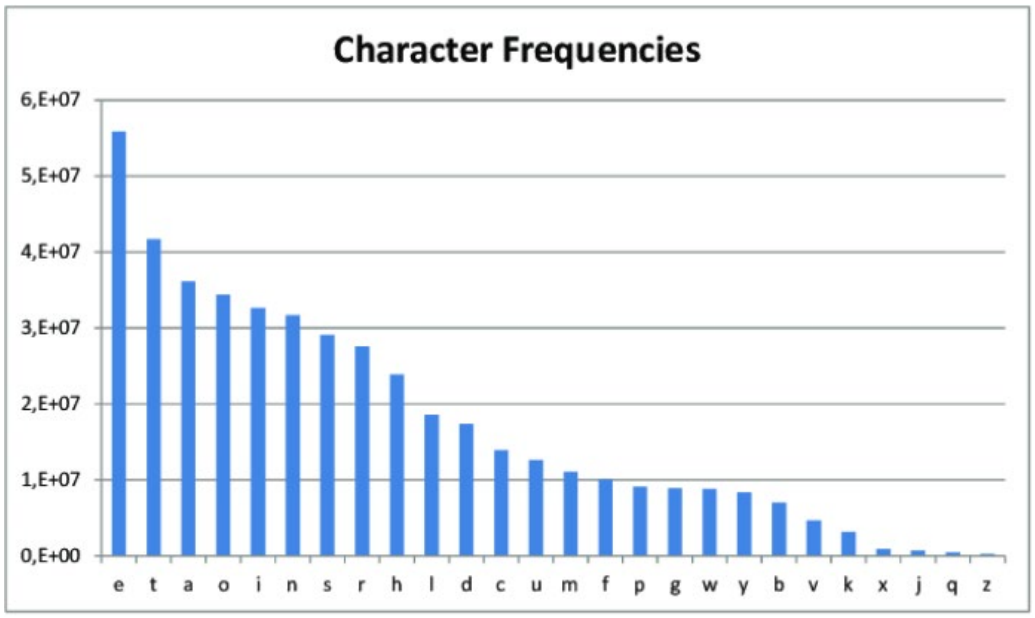

Рис. 1.2. Частоти символів англійських літер.

У 2010 році була представлена клавіатура Swype, яка дозволяла користувачам проводити пальцями по екранній клавіатурі, щоб вводити слова [3]. Пізніше ця технологія була інтегрована в стандартну клавіатуру багатьох пристроїв Android. Введення жестами, яке дозволяє користувачам малювати літери на екранній клавіатурі замість того, щоб вводити їх, було вперше представлено Google на клавіатурі Gboard у 2012 році.

Загальний вигляд клавіатури Swype (рис. 1.3.):

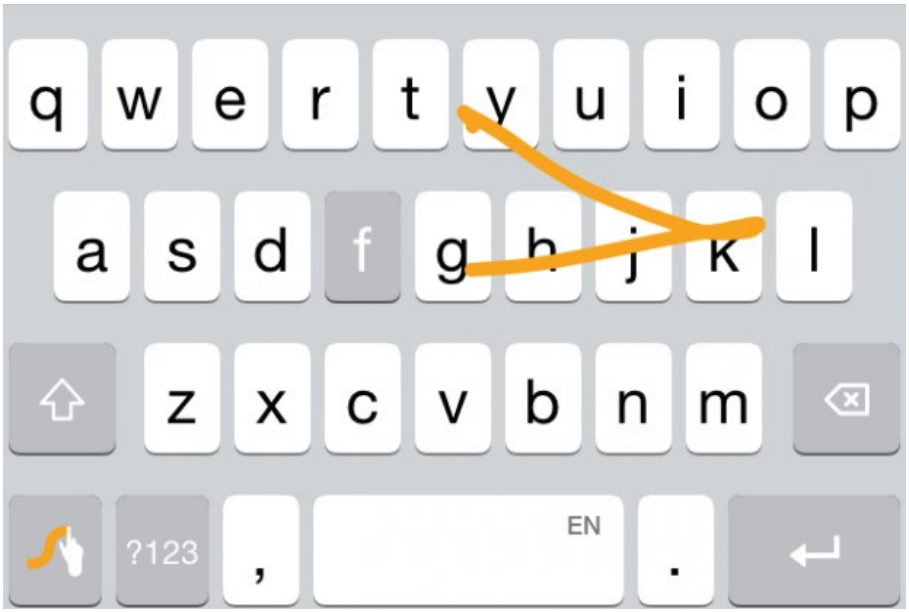

Рис. 1.3. Інтерфейс клавіатури Swype.

У середині 2010-х років був представлений тактильний зворотний зв’язок, який дозволяв користувачам відчувати вібрацію або тактильну реакцію, коли вони друкують на екранній клавіатурі. У цей час також були представлені клавіатури Emoji, які дозволяють користувачам легко отримувати доступ до емодзі та використовувати їх у своїх повідомленнях (рис. 1.4.).

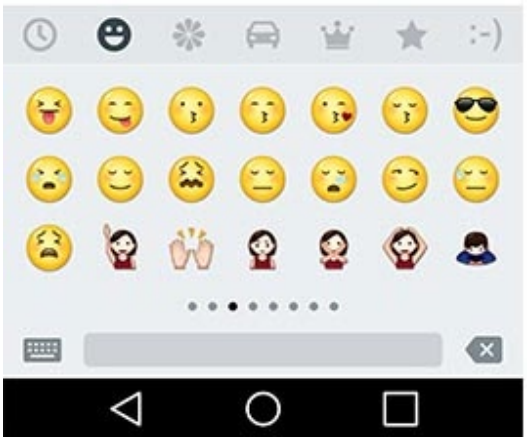

Рис. 1.4. Вигляд емодзі на клавіатурі Swype.

У 2016 році клавіатуру Google було оновлено до Gboard, яка інтегрувала багато служб Google безпосередньо в клавіатуру, зокрема пошук, карти та YouTube. Gboard також представив нові функції, такі як пошук емодзі та бібліотека GIF [4].

Останніми роками екранні клавіатури стали доступнішими для людей з обмеженими можливостями, наприклад, із вадами зору чи обмеженою рухливістю. Функції на основі штучного інтелекту, включаючи пропозиції фраз і речень, а також покращену функцію голосового введення, були представлені в 2018 році.

Головні властивості екранних клавіатур Android включають:

1. Інтелектуальний текст: екранні клавіатури Android використовують розширені алгоритми для передбачення слів, які вводять користувачі, що може допомогти підвищити швидкість і точність введення.
2. Персоналізоване автовиправлення: екранну клавіатуру для Android можна налаштувати, щоб вивчати звички користувача при наборі тексту та автоматично виправляти помилки на основі попередніх введених даних користувача.

3. Введення пальцем: ця функція дозволяє користувачам вводити текст, проводячи пальцем по екранній клавіатурі, а не торкаючись кожної літери окремо.
4. Розпізнавання голосу: екранні клавіатури Android також можуть містити технологію розпізнавання голосу, що дозволяє користувачам вводити текст і команди голосом.
5. Настроюванні розкладки: екранні клавіатури Android пропонують ряд настроюваних розкладок і тем, що дозволяє користувачам персоналізувати зовнішній вигляд клавіатури.
6. Функції доступності: багато екранних клавіатур для Android містять такі функції, як великі клавіші, настроювані макети та розпізнавання голосу, щоб покращити доступність для користувачів з обмеженими можливостями.
7. Кілька мов: екранні клавіатури Android також підтримують кілька мов, що дозволяє користувачам вводити текст улюбленою мовою.

Для Android доступно багато екранних клавіатур, кожна з яких має власний набір функцій і переваг. Ось порівняння деяких із найпопулярніших екранних клавіатур.

Gboard є офіційною клавіатурою Google для Android і iOS. Вона попередньо встановлена на багатьох пристроях Android, а також її можна безкоштовно завантажити з App Store для пристроїв iOS (рис. 1.5.).

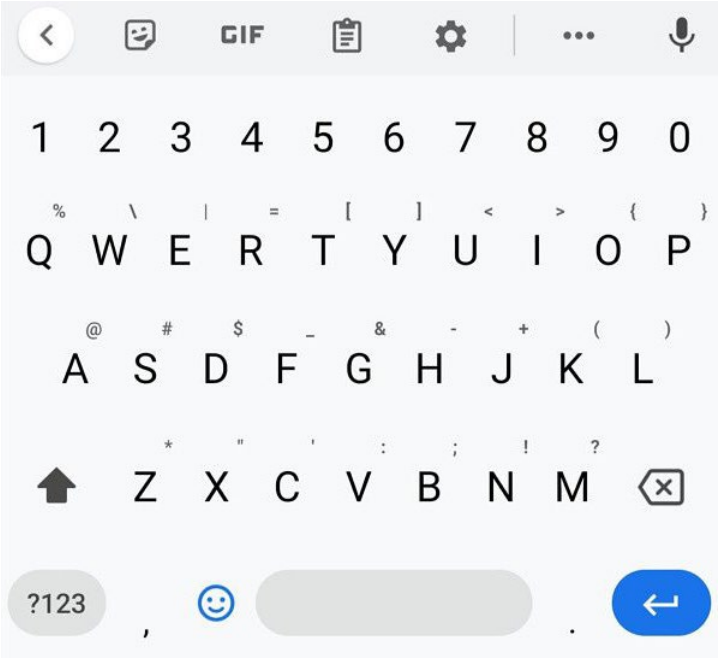

Рис. 1.5. Інтерфейс клавіатури Gboard.

Основними перевагами Gboard є:

- Багатомовна підтримка: Gboard підтримує понад 100 мов і дозволяє користувачам вводити текст кількома мовами одночасно. Він також містить пропозиції щодо часто використовуваних фраз і слів різними мовами.
- Введення жестами. Функція введення жестами в Gboard дозволяє користувачам проводити пальцями по літерах, створюючи слова, що робить введення тексту швидшим та більш інтуїтивним.
- Голосовий набір: функція голосового набору Gboard дозволяє користувачам диктувати текст за допомогою голосу, що може бути особливо корисним для людей з обмеженою мобільністю або тих, хто не хоче друкувати.
- Емодзі та GIF-файли: Gboard містить широкий вибір емодзі та GIF-файлів, що дозволяє легко виражати емоції під час введення тексту. Користувачі можуть шукати певні емодзі та GIF за допомогою ключових слів.

- Інтеграція пошуку Google: Gboard інтегрується з пошуком Google, дозволяючи користувачам шукати інформацію, зображення та навіть GIF-файли, не відходячи від клавіатури.
- Налаштування: Gboard пропонує низку параметрів налаштування, зокрема можливість змінити тему клавіатури, розмір клавіатури та налаштувати звук натискання клавіш.
- Особистий словник: Gboard дозволяє користувачам створювати особистий словник слів, фраз і імен, які часто вживаються або часто пишуться з помилками. Ця функція може заощадити час і зменшити кількість помилок при наборі.
- Конфіденційність і безпека: Gboard використовує наскрізне шифрування для захисту особистих даних користувачів. Він також надає користувачам можливість вимкнути обмін даними, якщо вони мають сумніви щодо конфіденційності.

Gboard — це потужна й універсальна екранна клавіатура, яка пропонує багато функцій і переваг. Вона допомагає зробити набір тексту на мобільному пристрої швидшим, ефективнішим і приємнішим [4].

SwiftKey — популярна екранна клавіатура, розроблена Microsoft для пристроїв Android і iOS. Він пропонує низку функцій, розроблених для покращення швидкості та точності введення тексту та персоналізації досвіду введення тексту для користувача [5] (рис. 1.6.).

Рис. 1.6. Інтерфейс клавіатури SwiftKey.

Основні переваги клавіатури SwiftKey:

- Інтелектуальний набір тексту: SwiftKey використовує машинне навчання, щоб пропонувати слова та фрази, коли користувач вводить текст, що робить набір тексту швидшим і ефективнішим.
- Багатомовна підтримка: SwiftKey підтримує понад 300 мов і дозволяє користувачам вводити текст кількома мовами одночасно.
- Налаштування: SwiftKey пропонує ряд параметрів налаштування, включаючи можливість змінити тему клавіатури, змінити розмір клавіатури та налаштувати звук натискання клавіш.
- Введення жестами: SwiftKey включає функцію введення жестами, яка дозволяє користувачам ковзати пальцем по літерах для створення слів, що робить введення тексту швидшим та більш інтуїтивним.
- Емодзі та GIF-файли: SwiftKey містить широкий вибір емодзі та GIF-файлів, що дозволяє легко виражати емоції під час набору тексту. Користувачі можуть шукати певні емодзі та GIF за допомогою ключових слів.
- Особистий словник: SwiftKey дозволяє користувачам створювати особистий словник слів, фраз і імен, які часто вживаються або часто пишуться з помилками. Ця функція може заощадити час і зменшити кількість помилок при наборі.
- Інтеграція буфера обміну: SwiftKey інтегрується з буфером обміну, дозволяючи користувачам швидко відкривати та вставляти раніше скопійований текст.
- Конфіденційність і безпека: SwiftKey використовує наскрізне шифрування для захисту особистих даних користувачів. Він також надає користувачам можливість вимкнути обмін даними, якщо вони мають сумніви щодо конфіденційності.

SwiftKey — це потужна та універсальна екранна клавіатура, яка пропонує низку функцій для підвищення ефективності введення та персоналізації. Його функція інтелектуального введення та багатомовна підтримка заслуговують

особливої уваги, що робить його популярним вибором для людей, які часто перемикаються між різними мовами [5].

AnySoftKeyboard — це екранна клавіатура з відкритим вихідним кодом для пристроїв Android, яка надає низку розширених функцій і параметрів налаштування. Вона доступна безкоштовно в магазині Google Play і може використовуватися з будь-яким пристроєм Android під керуванням Android 4.0.3 (Ice Cream Sandwich) або новіших версій [29] (рис. 1.7.).

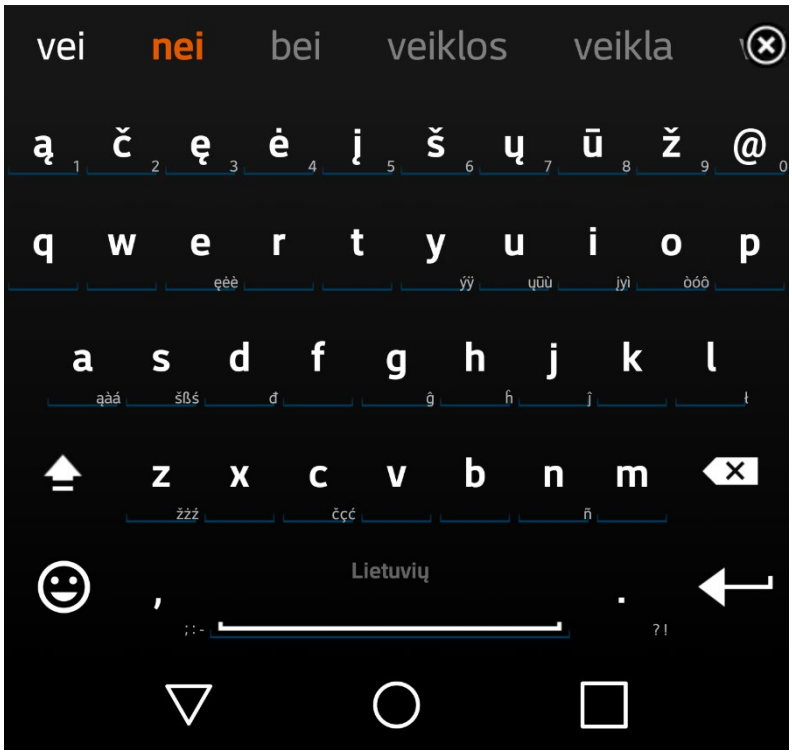

Рис. 1.7. Інтерфейс клавіатури AnySoftKeyboard.

Основні переваги клавіатури AnySoftKeyboard:

- Багатомовна підтримка: AnySoftKeyboard підтримує понад 200 мов і дозволяє користувачам вводити текст кількома мовами одночасно.
- Налаштування: AnySoftKeyboard пропонує низку параметрів налаштування, включаючи можливість змінити тему клавіатури, змінити розмір клавіатури та налаштувати звук натискання клавіш.
- Введення жестами: AnySoftKeyboard містить функцію введення жестами, яка дозволяє користувачам ковзати пальцем по літерах, щоб скласти слова, що робить введення тексту швидшим та більш інтуїтивним.

- Емодзі: AnySoftKeyboard містить широкий спектр емодзі, що дозволяє легко виражати емоції під час набору тексту.
- Словник: AnySoftKeyboard дозволяє користувачам створювати особистий словник слів, фраз і імен, які часто вживаються або часто пишуться з помилками. Ця функція може заощадити час і зменшити кількість помилок при наборі.
- Конфіденційність і безпека: AnySoftKeyboard не збирає дані користувача, а вся інформація, введена на клавіатурі, зберігається локально на пристрої.
- Легкість: AnySoftKeyboard створена як легка та швидка, що робить її гарним варіантом для пристроїв з обмеженими ресурсами.

AnySoftKeyboard — це багатофункціональна екранна клавіатура для Android-пристроїв, яка легко настроюється. Завдяки підтримці кількох мов, набору тексту пальцем і широким можливостям налаштування вона користується популярністю серед користувачів, які цінують гнучкість і універсальність клавіатури.

1.2. Призначення розробки та область застосування.

Темою бакалаврської кваліфікаційної роботи виступає: «Розробка екранної клавіатури для операційної системи Android на мові Java». В якості мови програмування обрана Java. Тобто головною метою роботи є створення екранної клавіатури, яка забезпечує зручне введення тексту з різноманітних пристроїв.

Головними критеріями розроблювальної клавіатури є:

- Зручність в використанні.
- Максимальна доступність для користувача.
- Легкість в освоєнні.

Клавіатура призначена для:

- Введення тексту з різноманітних пристроїв на базі Android.
- Додання до існуючих застосунків функціоналу введення тексту.

Клавіатура позиціонується як застосунок для операційної системи Android, який дає можливість вводити текст за допомогою сенсорного екрана, миші або звичайного пульта.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

- Освітня програма за спеціальністю 121 «Інженерія програмного забезпечення».
- Графік навчального процесу та навчальний план.
- Наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р.
- Завдання на кваліфікаційну роботу на тему «Розробка екранної клавіатури для операційної системи Android на мові Java».

1.4. Постановка завдання

Метою кваліфікаційної роботи є розробка екранної клавіатури для операційної системи Android. Клавіатура призначена для швидкого та зручного введення тексту з будь-якого пристрою на базі Android.

Клавіатура повинна реалізувати такі дії як:

- Ефективно та точно вводити текст у пристрій, який не має фізичної клавіатури, наприклад смартфон або планшет.
- Враховувати обмеження пристрою та переваги введення користувача.

Для виконання проекту необхідно:

- Проаналізувати існуючі екранні клавіатури.
- Спроектувати архітектуру клавіатури.
- Розробити дизайн та зручний інтерфейс клавіатури.
- Реалізувати спроектовану клавіатуру.

1.5. Вимоги до програми або програмного виробу.

1.5.1. Вимоги до функціональних характеристик.

Розроблена клавіатура, для того, щоб досягнути поставлених цілей, повинна підтримувати виконання таких дій:

- Підтримка широкого спектру пристроїв.
- Зручне встановлення на пристрої.
- Підтримка введення тесту як зі сенсорних екранів так і з звичайних кнопкових пультів.
- Зручне розташування екранних клавіш
- Багатомовна підтримка і зручне перемикання між мовами.
- Можливість налаштувати розкладку клавіатури, розмір, стиль та інші параметри.

- Клавіатура має бути швидкою, з низькою затримкою та мінімальним впливом на час автономної роботи.

Для підтримки вище перераховані функцій у застосунку має бути реалізовано:

- Сумісність з операційною системою Android 4.4 та вище.
- Клавіатура повинна бути доступна для встановлення з Play Store або зі звичайного apk-файлу.
- Надання початкової конфігурації для швидкого та зручного налаштування клавіатури.

1.5.2. Вимоги до інформаційної безпеки.

Для коректної роботи програми потрібно реалізувати:

- Дозволи користувача: програми для клавіатури Android мають запитувати лише ті дозволи, які необхідні для їх функціональності, і користувачів слід повідомляти про дозволи, які запитує програма. Це допомагає програмі запобігти доступу до даних, які їй не потрібні або до яких вона не має комерційного доступу.
- Збереження конфіденційності за рахунок мінімізації переданих даних і надання користувачам контролю над своїми даними.
- Контроль та обробка вхідних даних.
- Збереження цілісності даних у випадку збою.
- Локальне збереження даних для більшої захищеності.

Також екранна клавіатура повинна мати наступні характеристики:

- Регулярні оновлення, щоб гарантувати, що програма залишається актуальною з найновішими протоколами безпеки та технологіями.
- Ретельне тестування оновлень, щоб виявити та усунути будь-які вразливості системи безпеки.

1.5.3. Вимоги до складу та параметрів технічних засобів.

Для забезпечення надійного функціонування програмного забезпечення необхідно, щоб обчислювальна машина, на якій буде експлуатуватися клавіатура, мала такі характеристики:

- Сенсорний екран, маніпулятор «миша» або пульт керування.
- Пристрій на базі Android 4.4 або вище.
- 50 Мб вільного місця на жорсткому диску.
- Процесор ARM Cortex-A53 з тактовою частотою 1.4 ГГц.
- Не менше ніж 2 Гб оперативної пам’яті.
- Рідкокристалічний дисплей або монітор.

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

1.5.4. Вимоги до інформаційної та програмної сумісності.

Для коректного функціонування програми необхідно, щоб програмне забезпечення пристрою, на якому буде експлуатуватися застосунок, відповідало наступним вимогам:

- Операційна система Android 4.4 і вище.

Застосунок має бути реалізований на мові програмування Java з використанням бібліотек Android SDK.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Під час виконання кваліфікаційної роботи було розроблено програму, а саме, екранну клавіатуру, головна мета якого забезпечити швидке та зручне введення тексту з будь-якого пристрою на базі Android.

Призначення розробленого застосунку:

- Підтримка широкого спектру пристроїв.
- Зручне встановлення на пристрої.
- Підтримка введення тесту як зі сенсорних екранів так і з звичайних кнопкових пультів.
- Зручне розташування екранних клавіш
- Багатомовна підтримка і зручне перемикання між мовами.
- Можливість налаштувати розкладку клавіатури, розмір, стиль та інші параметри.
- Клавіатура має бути швидкою, з низькою затримкою та мінімальним впливом на час автономної роботи.

Для підтримки вище перераховані функцій у застосунку має бути реалізовано:

- Сумісність з операційною системою Android 4.4 та вище.
- Клавіатура повинна бути доступна для встановлення з Play Store або зі звичайного apk-файлу.
- Надання початкової конфігурації для швидкого та зручного налаштування клавіатури.

2.2. Опис застосованих математичних методів

Так як особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці екранної клавіатури математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Екранна клавіатура повинна бути реалізована на мові Java з використанням компонентів Android API для зберігання даних, рендерінгу та ін.

Опис мови програмування Java.

Java є основною мовою програмування, яка використовується для розробки застосунків Android. Переваги Java: простота, гнучкість та сумісності з різними платформами [6, 7].

Особливості Java.

Об’єктно-орієнтований підхід. Java є об’єктно-орієнтованою мовою програмування, що означає, що все в Java є об’єктом. Це полегшує написання модульного та багаторазового коду.

Незалежність від платформи. Програми Java можуть бути перенесені на будь-яку платформу, на якій встановлено віртуальну машину Android (Dalvik/ART) або Java (JVM). Це включає Android, Windows, Linux і macOS.

Збирання сміття. Java має автоматичний збирач сміття, який керує виділенням і звільненням пам’яті, що полегшує розробникам написання програм, не турбуючись про керування пам’яттю.

Багатопотоковість. Java підтримує багатопотоковість, що означає, що програми можуть мати кілька потоків виконання, що виконуються одночасно. Це дозволяє підвищити продуктивність і ефективніше використовувати системні ресурси.

Сувора типізація. Java — це строго типізована мова, що означає, що змінні мають певний тип, який перевіряється під час компіляції. Це допомагає запобігти помилкам і робить програми більш надійними.

Синтаксис. Синтаксис Java подібний до C і C++, що полегшує вивчення Java для розробників, які знайомі з цими мовами.

Стандартні бібліотеки. Java поставляється з великим набором стандартних бібліотек, які забезпечують функціональні можливості для роботи в мережі, файлового вводу/виводу, аналізу XML тощо.

Загалом Java є потужною та універсальною мовою, яка добре підходить для розробки Android. Його об’єктно-орієнтована природа, збирання сміття, підтримка багатопотоковості та великі бібліотеки роблять його популярним вибором для розробки мобільних додатків.

Опис компонентів Android API.

Android API (інтерфейс прикладного програмування) — це набір програмних інтерфейсів, протоколів і інструментів, які дозволяють розробникам створювати програми Android. API надає набір попередньо визначених класів, методів і функцій, які розробники можуть використовувати для взаємодії з операційною системою Android і створення своїх програм.

Android API надає розробникам доступ до різноманітних функцій операційної системи Android, таких як графіка, мережеве забезпечення, мультимедіа та обладнання. Він також містить низку готових компонентів, таких як віджети та макети, які розробники можуть використовувати для створення інтерфейсів користувача.

Android API розділений на кілька рівнів, кожен з яких надає різний рівень доступу до основної системи. Вищі рівні API надають розробникам більше функцій і контролю, але також вимагають більшого рівня досвіду для ефективного використання.

Детальна інформація про ключові компоненти Android API.

- Activities та Fragments для створення інтерфейсів користувача. Activities є будівельними блоками програм для Android. Вони представляють окремі екрани в програмі, і користувачі взаємодіють з ними, щоб виконувати різні задачі. Fragments — це менші компоненти Activities, які можна комбінувати для створення

складнішого інтерфейсу користувача. Activities та Fragments створюються за допомогою класів Java, і їх можна налаштувати за допомогою макетів XML [9].

- Служби (Services) для виконання фонових завдань і обробки сповіщень. Служби — це фонові компоненти, які виконують довгострокові завдання, наприклад завантаження файлів або відтворення музики. Вони працюють незалежно від інтерфейсу користувача програми, тому користувачі можуть продовжувати використовувати програму, поки служба працює. Служби створюються за допомогою класів Java і можуть запускатися та зупинятися за допомогою викликів API [16].
- Content Providers для керування зберіганням даних і обміну ними між програмами. Content Providers – це компоненти, які керують зберіганням даних і обміном ними між програмами. Вони надають програмам стандартизований спосіб доступу та зміни даних, наприклад контактів або файлів, без необхідності знати деталі того, як дані зберігаються. Постачальники вмісту створюються за допомогою класів Java, а доступ до них здійснюється за допомогою викликів API [21].
- Broadcast Receivers для реагування на системні події. Broadcast Receivers — це компоненти, які відстежують системні події, наприклад коли змінюється рівень заряду акумулятора або коли надходить нове SMS-повідомлення. Вони дозволяють програмам реагувати на ці події та вживати відповідних заходів. Broadcast Receivers створюються за допомогою класів Java і реєструються для отримання певних подій [22].
- Наміри (Intents) для спілкування між програмами. Intents — це механізм зв’язку між програмами. Вони дозволяють програмам запускати дії чи служби в інших програмах або транслювати

повідомлення іншим програмам. Intents створюються за допомогою класів Java і надсилаються за допомогою викликів API [10].

- Views та Layouts для створення інтерфейсів користувача. Views та Layouts — це компоненти, які використовуються для створення інтерфейсу користувача програми. Views — це окремі елементи інтерфейсу користувача, такі як кнопки або текстові поля, тоді як Layouts — це контейнери, які містять види та визначають їх розташування на екрані. Views та Layouts створюються за допомогою класів Java і макетів XML [11, 23].
- Анімація та графіка для створення візуальних ефектів. Android API включає підтримку для створення анімації та графіки, наприклад 2D та 3D графіки, векторної графіки та анімації. Ці функції надаються через класи Java і макети XML [27].
- Робота з медіа та камерою для обробки мультимедійного вмісту. Android API включає підтримку відтворення аудіо- та відеофайлів, запису аудіо та відео та роботи з камерою. Ці функції надаються через класи Java і виклики API [26].
- Підключення (Connectivity) для роботи з мережею і Bluetooth. Android API включає підтримку роботи з мережею, Bluetooth та інші функції підключення. Ці функції надаються через класи Java і виклики API [25].
- Розташування та карти для роботи з геолокаційними службами. Android API включає підтримку роботи зі службами на основі місцезнаходження, такими як GPS і карти. Ці функції надаються через класи Java і виклики API [24].

Для розробки Android також доступні численні бібліотеки та API сторонніх розробників, які можуть допомогти розробникам розширити функціональність своїх програм за межі можливостей основного API Android.

Крім цих основних компонентів, Android API містить багато інших функцій і бібліотек, які дозволяють розробникам створювати потужні та

багатофункціональні програми. API постійно розвивається, з кожною новою версією операційної системи Android додаються нові функції та вдосконалення [7].

2.4. Опис структури системи та алгоритмів її функціонування

Редактор методів введення (IME) — це елемент керування, який дозволяє користувачам вводити текст. Android надає розширювану структуру методів введення, яка дозволяє програмам надавати користувачам альтернативні методи введення, наприклад екранну клавіатуру або голосовий ввід. Після встановлення IME користувач може вибрати один із системних налаштувань і використовувати його в усій системі. Одночасно можна ввімкнути лише один IME [2].

Наприклад, ось інтерфейс латинського IME для введення тексту на платформі Android (рис. 2.1.):

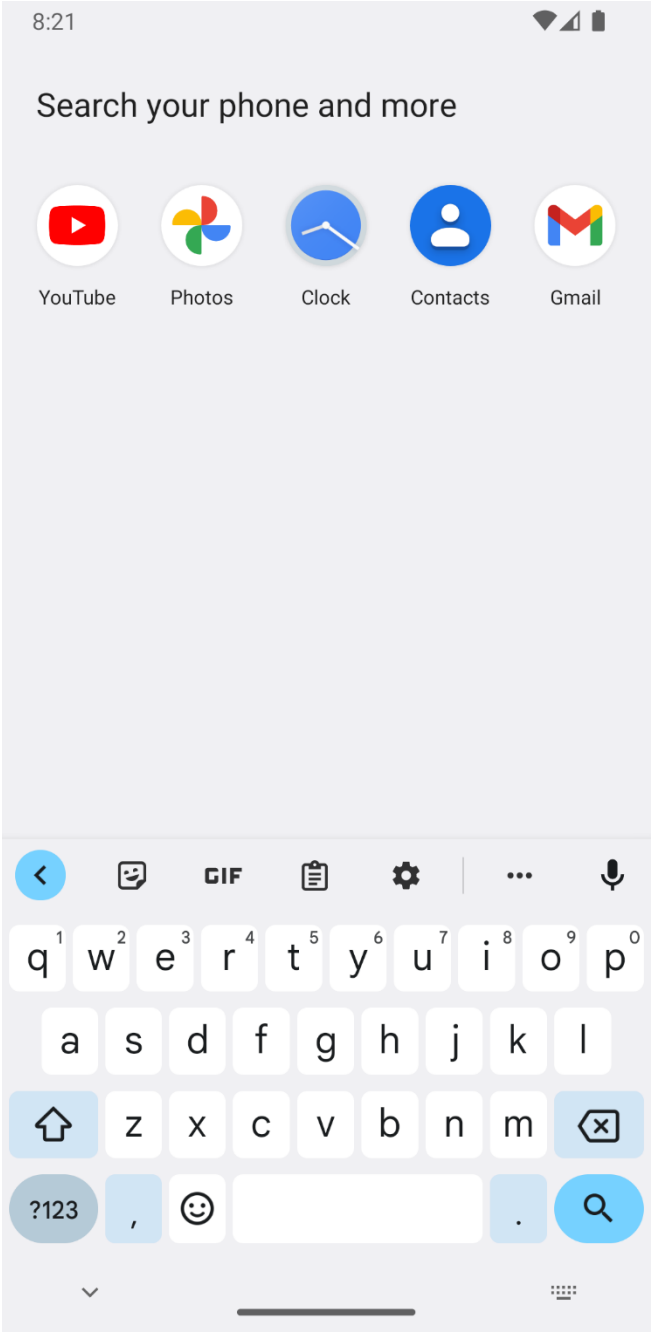

Рис. 2.1. Введення тексту латиницею.

Життєвий цикл IME.

На наступній діаграмі описано життєвий цикл IME [15] (рис. 2.2.):

Рис. 2.2. Життєвий цикл IME.

Життєвий цикл IME складається с наступних етапів:

1. Початок (InputMethodService starts).

2. Створення служби (onCreate()).
3. Створення вікна введення (onCreateInputView()).
4. Створення вікон кандидатів (onCreateCandidateViews()).
5. Отримання поточного підтипу метода введення (getCurrentInputMethodSubtype())
6. Початок введення тексту.
7. Зміна поточного метода введення (onCurrentInputMethodSubtypeChanged()).
8. Закінчення введення тексту (onFinishInput()).
9. Початок руйнування сервісу.
10. Зупинка сервісу.

Щоб додати IME до системи Android, треба створити програму Android, що містить клас, який розширює InputMethodService [17]. Крім того, ви зазвичай створюєте activity «параметри», яка передає параметри службі IME. Треба також визначити інтерфейс налаштувань, який відображатиметься як частина налаштувань системи.

InputMethodService надає стандартну реалізацію InputMethod, з якої остаточні реалізації можна отримувати та налаштовувати. Переглянемо базовий клас AbstractInputMethodService та інтерфейс InputMethod для отримання додаткової інформації про основи написання методів введення.

На додаток до звичайних методів життєвого циклу служби (Service), цей клас представляє деякі нові спеціальні зворотні виклики, які більшість підкласів можуть використовувати:

- onInitializeInterface() для ініціалізації інтерфейсу користувача, зокрема для обробки змін конфігурації під час роботи служби.
- onBindInput(), щоб дізнатися про перехід на новий клієнт.
- onStartInput(EditorInfo, boolean) для обробки сеансу введення, починаючи з клієнта.

- onCreateInputView(), onCreateCandidatesView() і onCreateExtractTextView() для створення інтерфейсу користувача без потреби.
- onStartInputView(android.view.inputmethod.EditorInfo, boolean) для роботи з введенням, починаючи з області введення IME.

Метод введення має значну свободу дій у своїй роботі: InputMethodService надає базову структуру для стандартних елементів інтерфейсу користувача (перегляд введення, перегляд кандидатів і запуск у повноекранному режимі), але конкретний розробник вирішує, як це зробити. використовувати їх. Наприклад, один метод введення може реалізовувати область введення за допомогою клавіатури, інший може дозволяти користувачеві малювати текст, тоді як третій може не мати області введення (і, отже, не бути видимим для користувача), але замість цього може слухати аудіо та виконувати текст до перетворення мовлення.

У представленій тут реалізації всі ці елементи розміщено разом в одному вікні, яким керує InputMethodService. Він виконуватиме зворотні виклики, коли йому буде потрібна інформація про них, і надаватиме API для програмного керування ними. Компонування цих елементів чітко визначено:

- Вікно програмного введення (soft input view), якщо доступно, розміщується в нижній частині екрана.
- Вікно кандидатів (candidates view), якщо наразі відображається, розміщується над вікном програмного введення.
- Якщо не працює в повноекранному режимі, програма переміщується або змінюється розмір, щоб бути над цими вікнами; якщо запущено в повноекранному режимі, вікно повністю закриватиме програму, а його верхня частина міститиме текст, що зараз редагується програмою.

Вікно програмного введення (Soft Input View).

Центральним для більшості методів введення є вікно програмного введення (soft input view). Саме тут відбувається більшість взаємодії з користувачем: натискання програмних клавіш, малювання символів або будь-який інший метод

введення, який хоче створити текст. Більшість реалізацій просто матимуть власне представлення, яке виконує всю цю роботу, і повертатимуть його новий екземпляр під час виклику onCreateInputView(). На цьому етапі, доки вікно введення є видимим, ми побачимо взаємодію користувача в цим вікном (view) та зможемо викликати службу InputMethodService для взаємодії з програмою належним чином.

У деяких ситуаціях нам треба вирішити, чи показувати користувачеві режим програмного введення. Це робиться шляхом реалізації onEvaluateInputViewShown() для повернення true або false залежно від того, чи потрібно це відображати в поточному середовищі. Якщо будь-який із ваших станів змінився, що може вплинути на це, викличте updateInputViewShown(), щоб повторно оцінити його. Реалізація за замовчуванням завжди показує вигляд введення, якщо немає доступної апаратної клавіатури, що є відповідною поведінкою для більшості методів введення.

Вікно кандидатів (Candidates View).

Часто, коли користувач вводить необроблений текст, метод введення може надати йому список можливих інтерпретацій цього тексту, який можна вибрати для використання. Це досягається за допомогою вікна кандидатів, і, подібно до вікна програмного введення, треба реалізувати onCreateCandidatesView(), щоб створити екземпляр власного вікна, реалізуючи власний інтерфейс кандидатів [17].

Керування вікном кандидатів дещо відрізняється від вікна введення, оскільки вікно кандидатів має тенденцію бути більш тимчасовим і відображається лише тоді, коли є можливі кандидати для поточного тексту, який вводить користувач. Щоб керувати вікном перегляду кандидатів, ви використовуєте setCandidatesViewShown(boolean). Зауважимо, що оскільки вікно кандидатів часто показується та приховується, воно не впливає на користувальницький інтерфейс програми так само, як вікно програмного введення: воно ніколи не призведе до зміни розміру вікон програми, лише

призведе до їх панорамування, якщо це необхідно для користувача, щоб побачити поточний фокус.

Повноекранний режим (Fullscreen Mode).

Іноді інтерфейс метода введення завеликий для інтеграції з інтерфейсом програми, тому є потреба у застосуванні всього екрана. Це досягається шляхом перемикання в повноекранний режим, у результаті чого вікно метода введення заповнює весь екран і додає власний редактор «вилученого тексту», який показує користувачеві текст, який вводиться. На відміну від інших елементів інтерфейсу користувача, для редактора «вилученого тексту» існує стандартна реалізація, яку вам не потрібно змінювати. Редактор розміщено у верхній частині IME, над вікнами введення та кандидатів.

Подібно до вікна вхідних даних, можна контролювати, чи IME працює в повноекранному режимі, реалізувавши onEvaluateFullscreenMode(), щоб повернути true або false залежно від того, чи має він бути повноекранним у поточному середовищі. Якщо будь-який із ваших станів змінився, що може вплинути на це, викличте updateFullscreenMode(), щоб повторно оцінити його (re-evaluate). Реалізація за замовчуванням вибирає повноекранний режим, коли екран знаходиться в альбомній орієнтації, що є прийнятною поведінкою для більшості методів введення, які мають значну область введення.

У повноекранному режимі є деякі особливі вимоги, оскільки користувач не може бачити інтерфейс програми. Зокрема, нам слід реалізувати onDisplayCompletions(android.view.inputmethod.CompletionInfo[]), щоб показати завершення (completions), згенеровані нашою програмою, як правило, у нашому вікні кандидатів, як ви зазвичай показуєте кандидатів.

Генерація тексту (Generating Text).

Ключовою частиною IME є, звичайно, генерація тексту для програми. Це робиться через виклики інтерфейсу InputConnection до програми, який можна отримати з getCurrentInputConnection(). Цей інтерфейс дозволяє генерувати необроблені ключові події або, якщо ціль (target) це підтримує, безпосередньо редагувати рядки кандидатів і зафіксований текст [17].

Інформацію про те, що ціль очікується та підтримує, можна знайти за допомогою класу EditorInfo, який отримується за допомогою методу getCurrentInputEditorInfo(). Найважливішою частиною цього є EditorInfo.inputType; зокрема, якщо це EditorInfo.TYPE_NULL, то ціль не підтримує складні редагування, і вам потрібно доставляти їй лише необроблені ключові події. Метод введення також захоче переглядати тут інші значення, наприклад, щоб виявити режим пароля, перегляд тексту з автоматичним заповненням, введення номера телефону тощо.

Коли користувач перемикається між цілями введення, ви отримуєте виклики onFinishInput() і onStartInput(android.view.inputmethod.EditorInfo, boolean). Ви можете використовувати їх для скидання та ініціалізації стану введення для поточної цілі. Наприклад, вам часто потрібно очистити будь-який стан введення та оновити програмну клавіатуру відповідно до нового типу введення.

Файлова структура екранної клавіатури (рис. 2.3.) має такий вигляд:

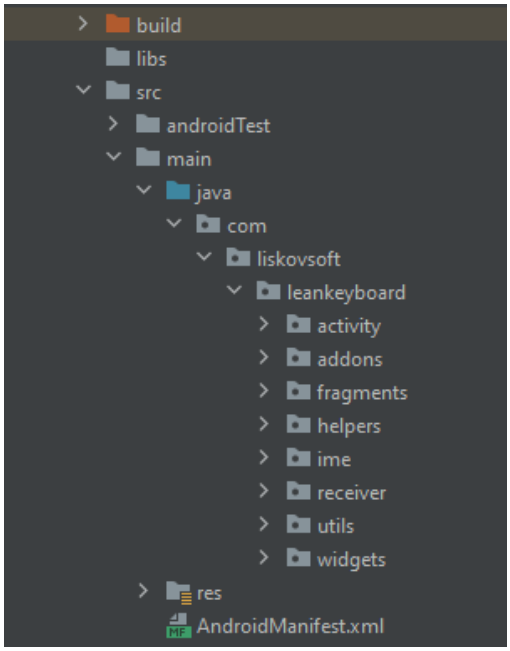

Рис. 2.3. Файлова структура екранної клавіатури.

У директорії build зберігаються тимчасові файли, які створюються у процесі компіляції. Папка libs зберігає додаткові бібліотекі (наразі вона порожня).

Сам застосунок розташований в папці src, яка розділена на такі директорії як: res, activity, addons, fragments, helpers, ime, receiver, utils, widgets [19]. Які функції виконують данні директорії:

- Res — містить ресурси застосунку, такі як зображення, файли перекладу, розкладки (layouts) для клавіатури.
- Activity — містить активності (activity) застосунку.
- Addons — містить класи, які розширюють функціонал застосунку.
- Fragments — містить фрагменти інтерфейсу, які розміщуються всередині різних активностей.
- Helpers — містить класи які допомагають у розробці застосунку.
- IME — ядро застосунку. Ця директорія містить фоновий сервіс, який потрібен для роботи екранної клавіатури.
- Receiver — містить класи, які відповідають за реагування на зовнішні події.
- Utils – містить різні утилітарні класи.
- Widgets — містить класи віджетів (примітивів графічного інтерфейсу).

Розглянемо більш детально структуру директорії res (рис. 2.4.).

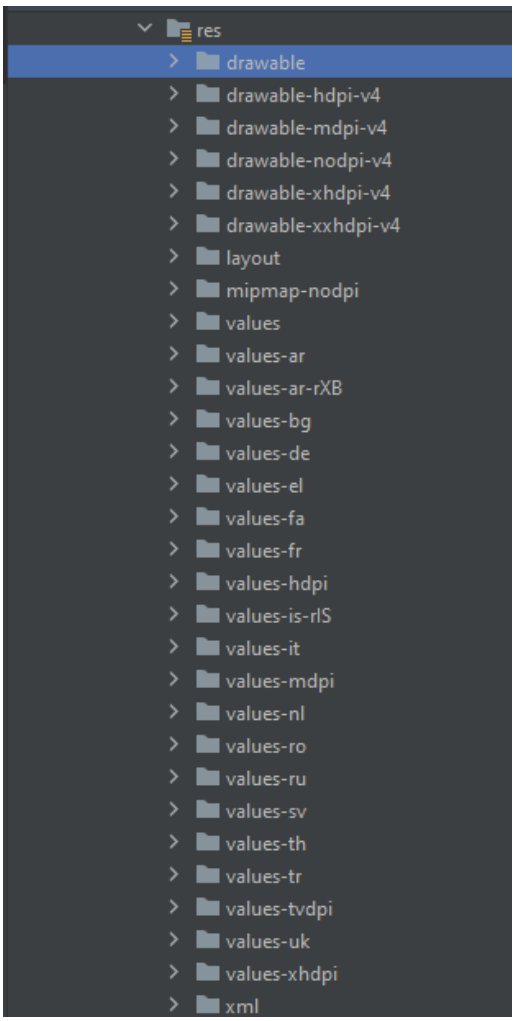

Рис. 2.4. Структура директорії res.

Директорія Res розділяється на такі піддиректорії як:

- drawable — зберігає ресурси такі як зображення.
- layout — зберігає файли для інтерфейсу користувача у програмі, наприклад у activity.
- values — зберігає переклади на різні мови та константи для всієї програми.
- xml — зберігає XML-описи клавіатури та атрибути клавіш для різних мов (клавіатура складається з рядів клавіш).

Розглянемо більш детально структуру директорії Activity (рис. 2.5.).

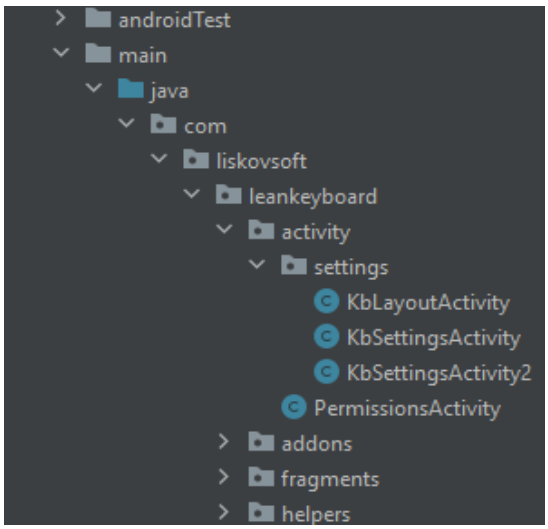

Рис. 2.5. Структура директорії Activity.

Директорія Activity містить такі файли як:

- KbLayoutActivity.java — вікно з вибором потрібних мов.
- KbSettingsActivity.java — вікно налаштувань клавіатури.
- KbSettingsActivity2.java — вікно налаштувань клавіатури, яке додається на робочий стіл, як лаунчер.
- PermissionsActivity.java — вікно надання дозволу для доступу до системних функцій.

Розглянемо більш детально структуру директорії Addons (рис. 2.6.).

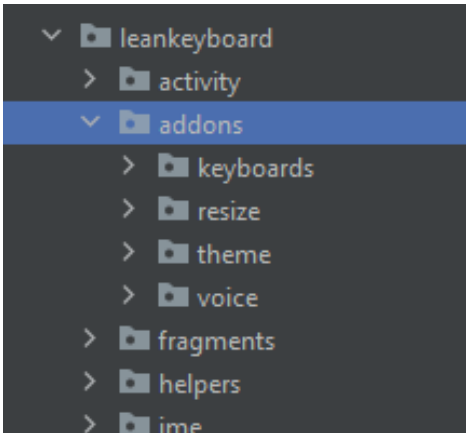

Рис. 2.6. Структура директорії Addons.

Директорія Addons містить такі піддиректорії як:

- Keyboards — містить класи, які допомагають будувати клавіатуру.
- Resize — містить класи, які допомагають змінювати розмір клавіатури.

- Theme — містить класи підтримки різних зовнішніх виглядів клавіатури.
- Voice — містить класи підтримки голосового пошуку.

Розглянемо більш детально структуру піддиректорії Keyboards (рис. 2.7.).

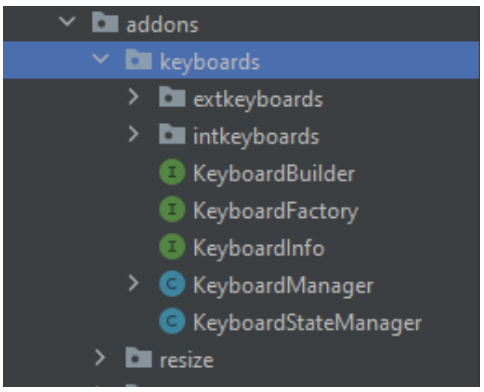

Рис. 2.7. Структура піддиректорії Keyboards.

Директорія Keyboards містить такі файли та директорії як:

- InitKeyboards — містить класи, які задають початковий стан клавіатури.
- KeyboardBuilder.java — інтерфейс, створює екземпляри клавіатури.
- KeyboardInfo.java — інтерфейс, надає інформацію про клавіатуру.
- KeyboardManager.java — допоміжний клас для керування клавіатурою.
- KeyboardStateManager.java — допоміжний клас для керування станом клавіатури.

Розглянемо більш детально структуру піддиректорії Resize (рис. 2.8.).

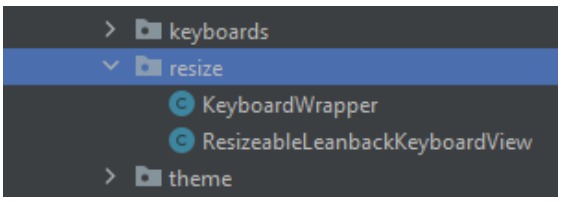

Рис. 2.8. Структура піддиректорії Resize.

Директорія Resize містить такі файли як:

- KeyboardWrapper.java — обгортка для клавіатури, додає можливість зміни розміру клавіатури (паттерн декоратор).

- ResizeableLeanbackKeyboardView.java — модифікована версія головного вікна клавіатури з функцією зміни розміру.

Розглянемо більш детально структуру піддиректорії Theme (рис. 2.9.).

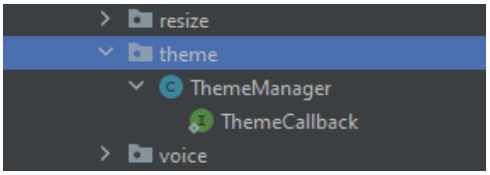

Рис. 2.9. Структура піддиректорії Theme.

Директорія Theme містить такі файли як:

- ThemeManager.java — допоміжний клас для підтримки різних зовнішніх виглядів клавіатури.

Розглянемо більш детально структуру піддиректорії Voice (рис. 2.10.).

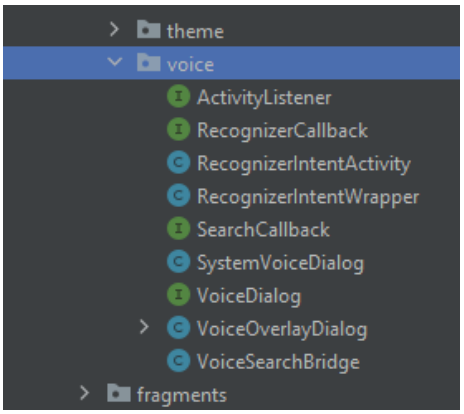

Рис. 2.10. Структура піддиректорії Voice.

Директорія Voice містить такі файли як:

- ActivityListener.java — інтерфейс загального обробника події Activity#onActivityResult.
- RecognizerIntentActivity.java — вікно, яке запускається при голосовому введенні.
- SystemVoiceDialog.java — діалог голосового введення, який вбудовано у RecognizerActivity.
- VoiceOverlayDialog.java — альтернативний діалог голосового введення, який вбудовано у RecognizerActivity.

Розглянемо більш детально структуру директорії Fragments (рис. 2.11.).

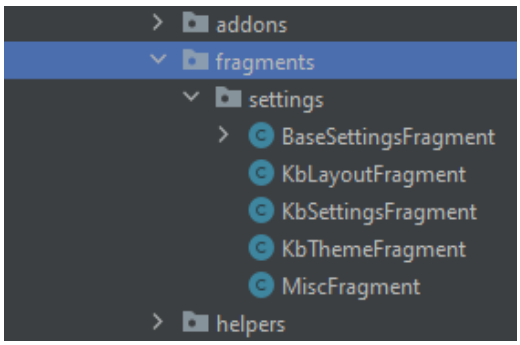

Рис. 2.11. Структура директорії Fragments.

Директорія Fragments містить такі файли як:

- BaseSettingsFragment.java — базовий клас для усіх фрагментів налаштувань застосунку.
- KbLayoutFragment.java — налаштування мов клавіатури.
- KbSettingsFragment.java — загальні налаштування клавіатури.
- KbThemeFragment.java — налаштування зовнішнього вигляду клавіатури.
- MiscFragment.java — додаткові налаштування клавіатури.

Розглянемо більш детально структуру директорії Helpers (рис. 2.12.).

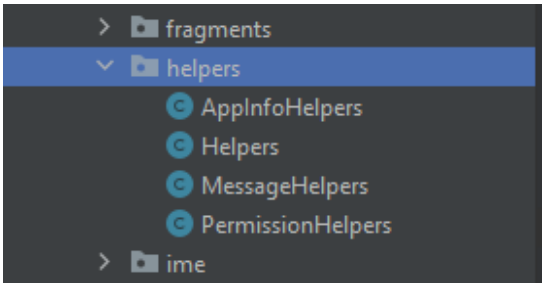

Рис. 2.12. Структура директорії Helpers.

Директорія Helpers містить такі файли як:

- AppInfoHelpers.java — інформація про додаток: версія, ідентифікатор та ін.
- Helpers.java — містить різні допоміжні функції.
- MessageHelpers.java — відповідає за випливаючи (toast) сповіщення.
- PermissionHelpers.java — відповідає за роботу з дозволами.

Розглянемо більш детально структуру директорії IME (рис. 2.13.).

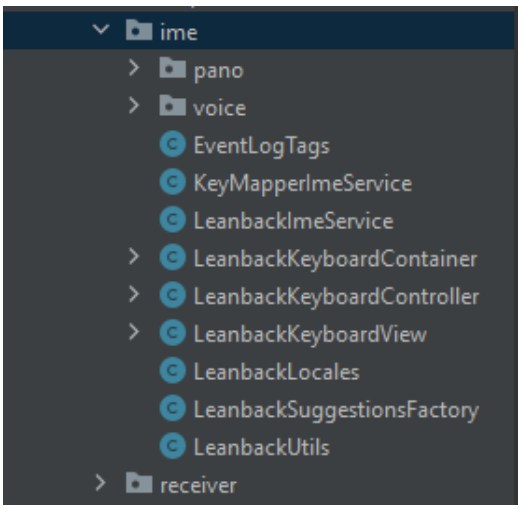

Рис. 2.13. Структура директорії IME.

Директорія IME містить такі файли як:

- KeyMapperImeService.java — базовий клас сервісу введення (IME). Містить функціонал обробки широкомоних повідомлень (broadcast).
- LeanbackImeService.java — похідний клас від KeyMapperImeService. Містить функціонал обробки подій введення.
- LeanbackKeyboardView.java — відповідає за відображення інтерфейса клавіатури.
- LeanbackSuggestionsFactory.java — відповідає за підказки під час введення тексту.

Розглянемо більш детально структуру директорії Receiver (рис. 2.14.).

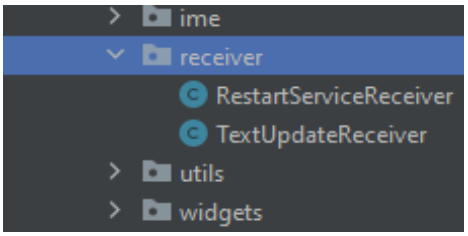

Рис. 2.14. Структура директорії Receiver.

Директорія Receiver містить такі файли як:

- RestartServiceReceiver.java — відповідає за запуск клавіатури після встановлення застосунку.

Розглянемо більш детально структуру директорії Utils (рис. 2.15.).

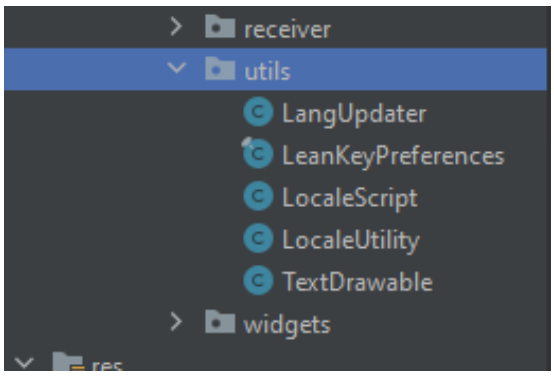

Рис. 2.15. Структура директорії Utils.

Директорія Utils містить такі файли як:

- LangUpdater.java — відповідає за локаль, під якою завантажується застосунок клавіатури.
- LeanKeyPreferences.java — відповідає за збереження налаштувань у приватній директорії застосунку.
- LocaleUtility.java — функції роботи з локалью.
- TextDrawable.java — перетворює текст у зображення.

Отже додаток розділений на різні пакети (packages) завдяки яким слідкувати за змінами в самій системі дуже зручно. Кожен файл відповідає за свою, окрему, функціональність і тому вносити будь-які зміни в проект не складає труднощів. Всі елементи логічно структуровані: є сервіси самої клавіатури, директорія addons в яких є додаткова функціональність, фрагменти, які відображують налаштування, класи голосового пошуку та керування зовнішнім виглядом клавіатури [14, 20].

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Екранна клавіатура отримує дані, коли користувач натискає на віртуальні клавіши за допомогою миші, тачпаду або екрану сенсорного дисплея.

Вихідні данні представлені в виді:

- Текстового рядка в консолі.
- Текстового рядка в окремому вікні програми.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для забезпечення надійного функціонування програмного забезпечення необхідно, щоб обчислювальна машина, на якій буде експлуатуватися клавіатура, мала такі характеристики:

- Сенсорний екран, маніпулятор «миша» або пульт керування.
- Пристрій на базі Android 4.4 або вище.
- 50 Мб вільного місця на жорсткому диску.
- Процесор ARM Cortex-A53 з тактовою частотою 1.4 ГГц.
- Не менше ніж 2 Гб оперативної пам’яті.
- Рідкокристалічний дисплей або монітор.

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

2.6.2. Використані програмні засоби

Застосунок реалізований на мові програмування Java з використанням стандартних бібліотек Android. Цих технологій достатньо для роботи з даними на стороні клієнта. Для візуалізації віртуальних клавіш та інших компонентів були використані віджети Android з пакету android.view [8, 13].

Необхідні програмні засоби на стороні клієнта:

- Операційна система Android 4.4 і вище.

2.6.3. Виклик та завантаження програми

Розроблений застосунок (екранна клавіатура) застосовується на таких пристроях, як смартфони, планшети, телевізійні приставки. Для нормальної

роботи потрібна операційна система Android 4.4 і вище. Робоча сборка завантажується та встановлюється на кінцевий пристрій у вигляді apk-файлу, або напряму з маркету застосунків Google Play. Після встановлення, клавіатуру потрібно додатково активувати у системних налаштуваннях (рис. 2.16.), а потім обрати у системному діалозі (рис. 2.17.). Після цього, нею можна користуватися як будь-якою іншою клавіатурою.

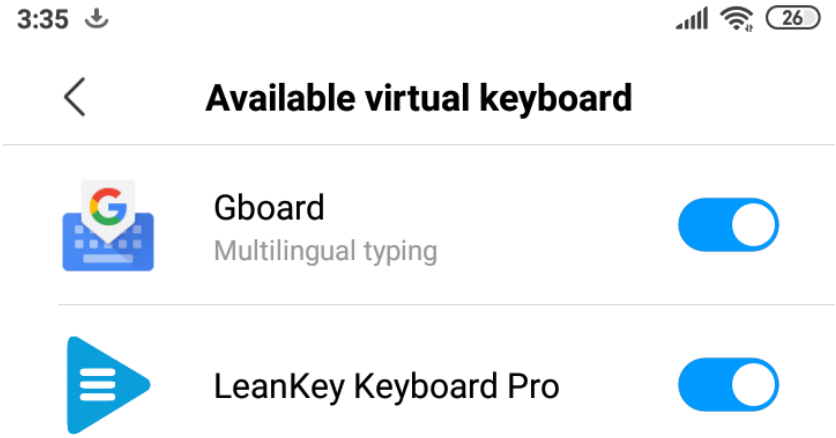

Рис. 2.16. Активація клавіатури у системних налаштуваннях.

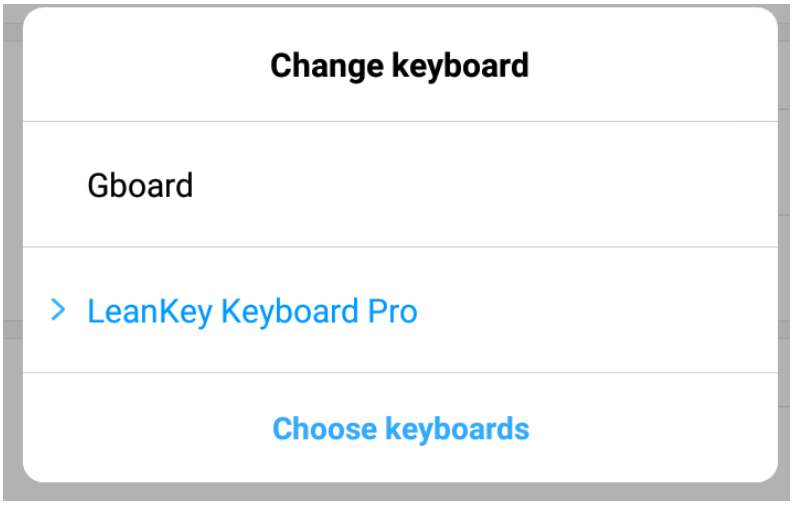

Рис. 2.17. Вибір клавіатури у системних налаштуваннях.

2.6.4. Опис інтерфейсу користувача

Для того щоб розпочати роботу з екранною клавіатурою, треба запустити будь-яку програму яка працює з текстом, або вимагає від користувача текстового

вводу. Наприклад, браузер. Клавіатура автоматично з’явиться на екрані (рис. 2.18.).

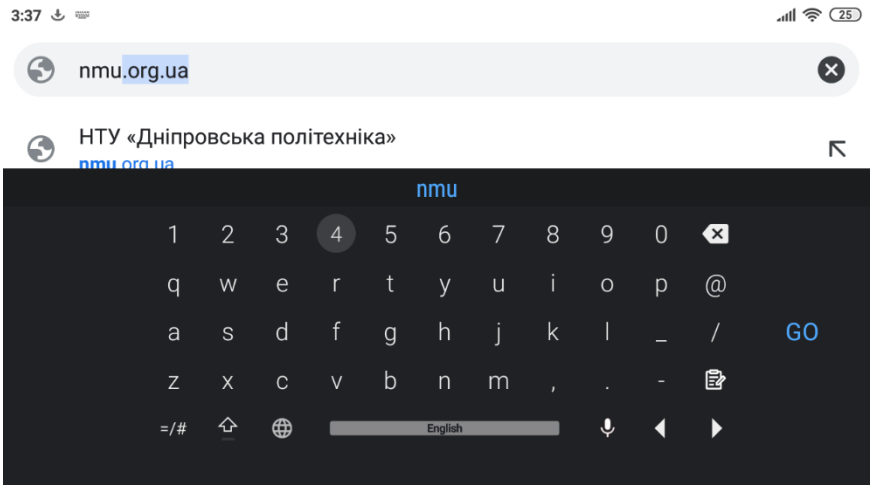

Рис. 2.18. Введення адреси сайту за допомогою екранної клавіатури у браузері.

Щоб ввести текст, просто натисніть або торкніться клавіш на екранній клавіатурі так само, як і на фізичній клавіатурі. Для пришвидшення введення, у верхній частині клавіатури можуть відображатися пропозиції. Наявність цих пропозицій залежить від поточного контексту. Щоб перейти на сторінку, адресу якої вводили у браузері, зазвичай треба натиснути клавішу «Перейти» (GO). Після цього клавіатура повинна автоматично закритися.

У випадку, якщо потрібна буква відсутня, або має декілька варіацій написання (зустрічається у іноземних мовах), треба відкрити список додаткових літер де можна обрати потрібну літеру. Список додаткових літер відкривається після довгого натискання на певну літеру (рис. 2.19.).

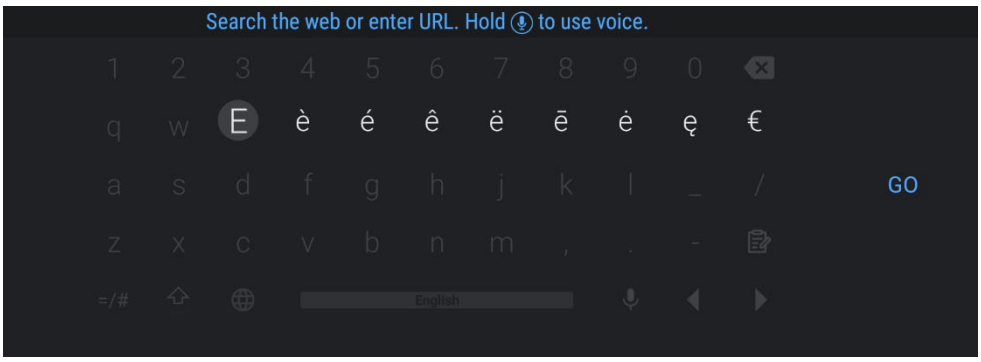

Рис. 2.19. Список додаткових літер для ‘e’.

Розкладка клавіатури, тобто мова введення, змінюється за допомогою клавіши з логотипом «земної кулі», яка розташована зліва від клавіші пробілу (рис. 2.20.). Зверніть увагу, що поточна мова введення відображається на клавіші пробілу. Одноразове натискання на клавішу з кулею циклічно перемикає активні розкладки клавіатури. Для налаштування активних розкладок треба зайти у налаштування клавіатури (це робиться, наприклад, з лаунчеру вашого пристрою), а потім у розділ вибору розкладок. В цьому розділі можна обрати потрібні розкладки (рис. 2.21.). Також, дістатися до розділу вибору розкладок можна за допомогою довгого натискання на клавішу з логотипом «земної кулі».

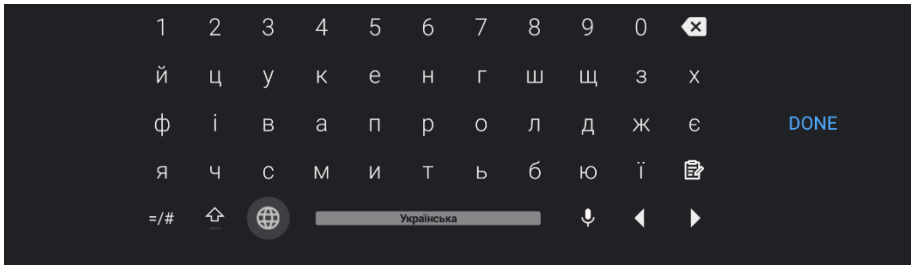

Рис. 2.20. Зміна активної розкладки ну українську.

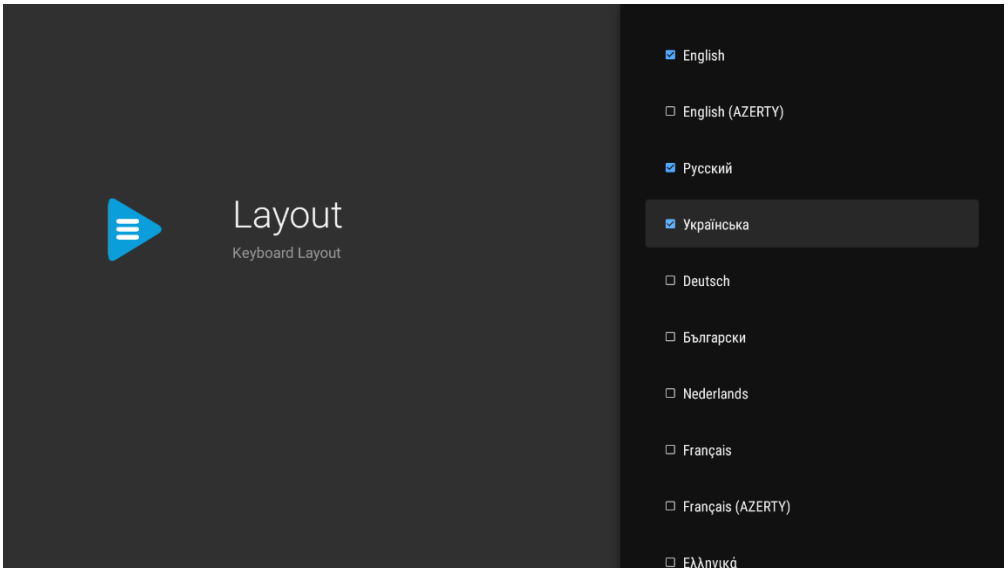

Рис. 2.21. Розділ налаштувань, де можна обрати активні розкладки.

Для набору спеціальних символів, треба натиснути на клавішу зі спеціальними позначками, яка знаходиться у нижньому лівому куті. Клавіатура перейде у режим введення спеціальних символів (рис. 2.22.). Тут можна побачити деякі розповсюджені символи, які часто використовуються при введенні. Наприклад: двокрапка, дефіс, коса риса.

Рис. 2.22. Вікно зі спеціальними символами.

Серед інших цікавих функцій клавіатури слід виділити клавішу шрифт (shift), яка змінює реєстр введених букв (рис. 2.23.), клавішу, яка дозволяє вставляти текст з буфера обміну, а також клавішу голосового пошуку (рис. 2.24.)

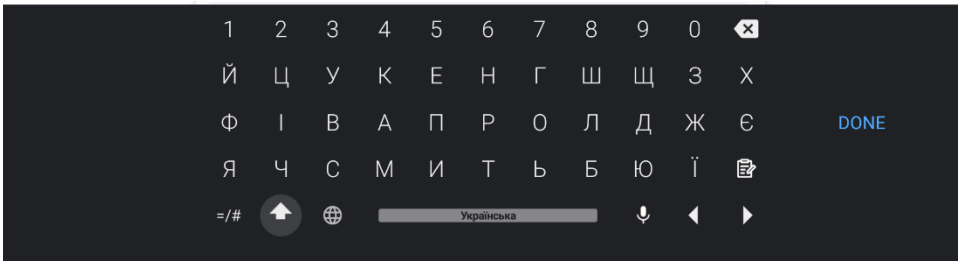

Рис. 2.23. Зміна реєстру літер за допомогою аналога клавіши Shift.

Рис. 2.24. Клавіша голосового пошуку.

За допомогою налаштувань можна гнучко змінювати зовнішній вигляд клавіатури, розміри, а також впливати на її поведінку. Давайте розберемо ці налаштування більш детально.

Для зміни зовнішнього вигляду потрібно зайти у відповідний розділ налаштувань клавіатури (рис. 2.25.).

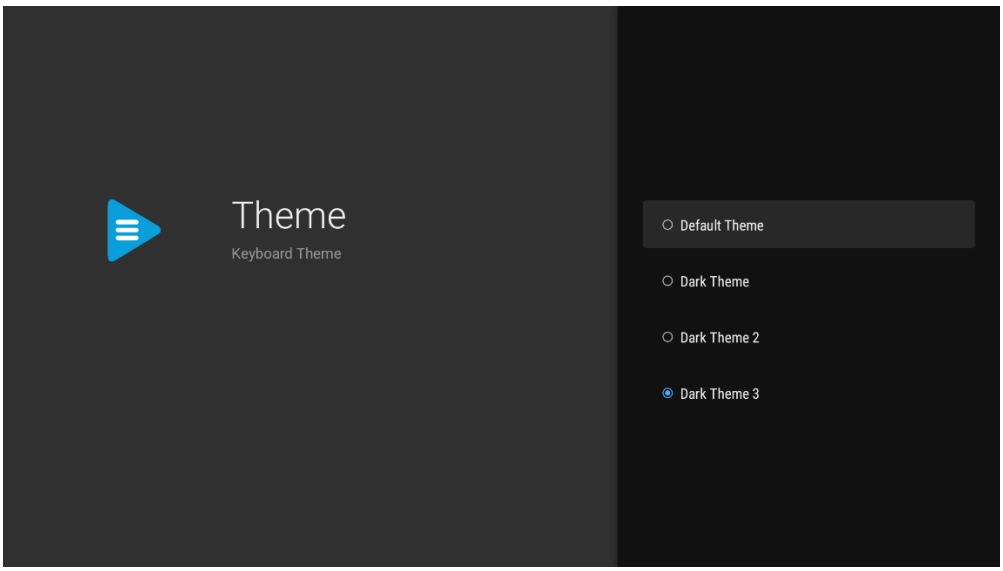

Рис. 2.25. Налаштування теми клавіатури.

У розділі налаштувань «Різне» (Misc) можна збільшити розмір клавіатури, вимкнути підказки при введені та інше (рис. 2.26.).

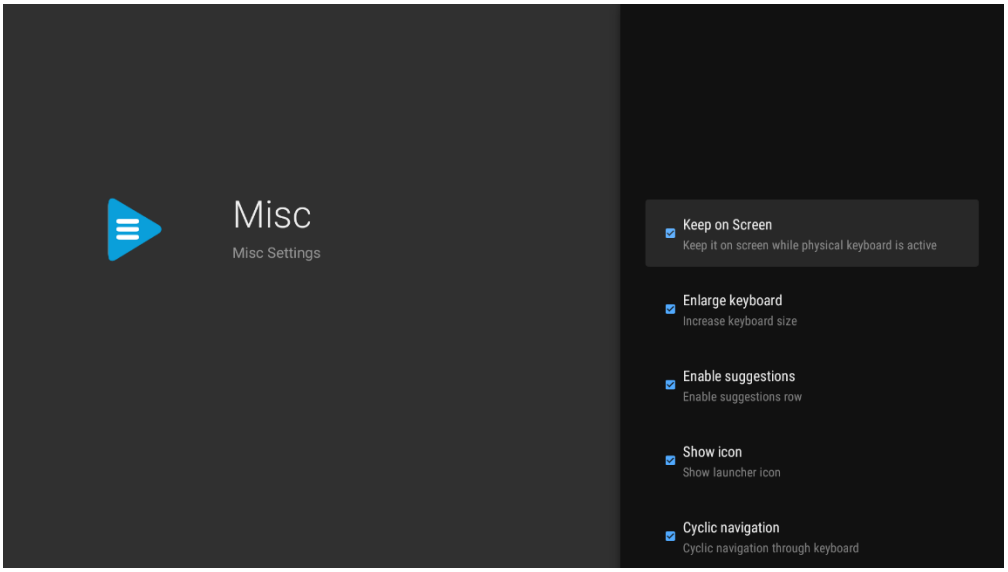

Рис. 2.26. Розділ налаштувань «Різне».

Загалом, за допомогою налаштувань можна гнучко змінювати різні аспекти клавіатури, від мови клавіатури до її розміру, що дозволяє клавіатурі пристосовуватися до різноманітних вимог користувача [23].

РОЗДІЛ 3
ЕКОНОМІЧНА ЧАСТИНА

Під час розробки програмного забезпечення важливими етапами є визначення трудомісткості розробки і розрахунок витрат на створення програмного продукту.

3.1. Визначення трудомісткості розробки програмного забезпечення

Задані дані:

1. передбачуване число операторів (підпрограм) – 1200;
2. коефіцієнт складності програми – 1,5;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста [28], грн/год – 150;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;
7. вартість машино-години ЕОМ, грн/год – 14 (розрахунок нижче).

Собівартість машино-години ЕОМ, грн/год:

$$M = \frac{S_1 + A + S_2 + S_3 + S_4 + S_5}{H}, \text{грн/год}, \quad (3.1)$$

де S_1 – річні витрати на заробітну плату обслуговуючого персоналу ЕОМ, грн;

A – річна сума амортизації, грн;

S_2 – річні витрати на електроенергію, грн;

S_3 – річні витрати на ремонтування та обслуговування обладнання, грн;

S_4 – річні витрати на матеріали, грн;

S_5 – річні накладні розходи, грн;

H – дійсний годовий фонд часу роботи, годин.

$$M = \frac{5000 + 20000 + 5000 + 5000 + 2000 + 1000}{2700} = 14, \text{грн/год},$$

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин}, \quad (3.2)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p), \quad (3.3)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт корекції програми в ході її розробки.

$$Q = 1200 * 1,5 * (1 + 0,2) = 2160.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q * B}{(75..85) * k}, \text{ людино-годин}. \quad (3.4)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{2160 * 1,3}{80 * 1} = \frac{2808}{80} = 35,1, \text{ людино-годин}.$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) * k}, \text{ людино-годин}, \quad (3.5)$$

$$t_a = \frac{2160}{20 * 1} = 108, \text{ людино-годин}.$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) * k}, \text{ людино-годин,} \quad (3.6)$$

$$t_n = \frac{2160}{25 * 1} = 86,4, \text{ людино-годин,}$$

Витрати праці на налагодження програми на ЕОМ:
- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) * k}, \text{ людино-годин,} \quad (3.7)$$

$$t_{отл} = \frac{2160}{4 * 1} = 540, \text{ людино-годин;}$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1,5 * t_{отл}, \text{ людино-годин,} \quad (3.8)$$

$$t^k_{отл} = 1,5 * 540 = 810, \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.9)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{15..20 * k}, \text{ людино-годин,} \quad (3.10)$$

$$t_{др} = \frac{2160}{20 * 1} = 108, \text{ людино-годин.}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 * t_{др}, \text{ людино-годин,} \quad (3.11)$$

$$t_{до} = 0,75 * 108 = 81, \text{ людино-година,}$$

$$t_д = 108 + 81 = 189 \text{ людино-годин.}$$

Тепер розрахуємо трудомісткість ПЗ:

$t = 81 + 35,1 + 108 + 86,4 + 540 + 189 = 1039,5$, людино-годин.

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{по} = З_{зп} + З_{мв}, \text{ грн.} \quad (3.12)$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t * С_{пр}, \text{ грн,} \quad (3.13)$$

де t - загальна трудомісткість, людино-годин;
$С_{пр}$ - середня годинна заробітна плата програміста, грн/година.

$$З_{зп} = 1039,5 * 150 = 155925, \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми:

$$З_{мв} = t_{отл} * С_{мч}, \text{ грн,} \quad (3.14)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год,
$С_{мч}$ - вартість машино-години ЕОМ, грн/год,
$С_{мч}$ = 14, грн/год.

$$З_{мв} = 540 * 14 = 7560, \text{ грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення ПЗ:

$$К_{по} = 155925 + 7560 = 163485, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{ міс.,} \quad (3.15)$$

де B_k - число виконавців (приймається 1),
F_p - місячний фонд робочого часу (40 годин на тиждень F_p = 176 годин).

$$T = \frac{1039,5}{1 * 176} = 5,9, міс.$$

Висновок:

Програмне забезпечення (екранна клавіатура) призначено для введення тексту за допомогою тачскріну, миші або пульта керування. Вартість даного продукту становить 163 тис. грн. і не вимагає додаткових витрат. Очікуваний час розробки становить 5,9 місяців. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В рамках кваліфікаційної роботи було розроблено застосунок екранної клавіатури.

Це програмне забезпечення призначено для того, щоб користувач міг вводити текст за допомогою тачскріну, миші або пульта керування. Застосунок має широкий обсяг налаштувань, які дозволяють змінювати розмір клавіатури, її зовнішній вигляд, обирати активні розкладки та багато іншого. Одна з важливих властивостей даної клавіатури – це можливість введення тексту за допомогою телевізійного пульта керування, що дозволяє використовувати клавіатуру на телевізорах, які працюють на операційній системі Android TV.

Під час виконання кваліфікаційної роботи були виконані наступні задачі:

- Проаналізовано предметну область поставленої задачі.
- Розроблено дизайн клавіатури.
- Спроектована внутрішня архітектура клавіатури.
- Визначено трудомісткість розробленої клавіатури.
- Підрахована вартість клавіатури.

Розроблена клавіатура дозволяє:

- Вводити текст на пристроях, які працюють на операційній системи Android.
- Використовувати налаштування для зміни різних аспектів застосунку.
- Зручне введення тексту за допомогою тачскріну, клавіатури, миші або телевізійного пульта.
- Введення тексту за допомогою голосу.

Застосунок реалізований за допомогою мови програмування Java, а саме за допомогою стандартних бібліотек доступних на операційній системі Android. Час створення застосунку становить 5,9 місяців, а його оціночна вартість становить 163 тис. грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поняття віртуальної клавіатури та її різновиди / URL: https://en.wikipedia.org/wiki/Virtual_keyboard (дата звернення 27.03.2023).

2. Поняття редактора метода введення / URL: https://en.wikipedia.org/wiki/Input_method (дата звернення 28.03.2023).

3. Опис клавіатури Swype / URL: https://en.wikipedia.org/wiki/Swype (дата звернення 30.03.2023).

4. Опис клавіатури Gboard / URL: https://en.wikipedia.org/wiki/Gboard (дата звернення 29.03.2023).

5. Опис клавіатури SwiftKey / URL: https://en.wikipedia.org/wiki/Microsoft_SwiftKey (дата звернення 30.03.2023).

6. Загальна інформація про мову програмування Java / URL: https://en.wikipedia.org/wiki/Java_(programming_language) (дата звернення 30.03.2023).

7. Загальний огляд операційної системи Android та її компонентів / URL: https://en.wikipedia.org/wiki/Android_(operating_system) (дата звернення 31.03.2023).

8. Документація по середовищу для розробки застосунків Android Studio / URL: https://developer.android.com/studio/intro (дата звернення 01.04.2023).

9. Введення у активності (activities) / URL: https://developer.android.com/guide/components/activities/intro-activities?hl=en (дата звернення 01.04.2023).

10. Наміри та фільтри намірів (intent filters) / URL: https://developer.android.com/guide/components/intents-filters?hl=en (дата звернення 02.04.2023).

11. Створення інтерфейсу за допомогою вікон / URL: https://developer.android.com/reference/android/view/View (дата звернення 02.04.2023).

12. Офіціальна документація розробника Android / URL: https://developer.android.com (дата звернення 03.04.2023).

13. Конфігурація збірки застосунка за допомогою Gradle / URL: https://developer.android.com/build (дата звернення 04.04.2023).

14. Інструкція з розробки екранної клавіатури / URL: https://developer.android.com/training/tv/start/onscreen-keyboard (дата звернення 05.04.2023).

15. Архітектура та розробка метода введення / URL: https://developer.android.com/develop/ui/views/touch-and-input/creating-input-method (дата звернення 07.04.2023).

16. Загальний огляд служб / URL: https://developer.android.com/guide/components/services (дата звернення 07.04.2023).

17. Документація по роботі служби метода введення (InputMethodService) / URL: https://developer.android.com/reference/android/inputmethodservice/InputMethodService (дата звернення 09.04.2023).

18. Дослідження проблем ефективності у дизайні віртуальних клавіатур / URL: https://www.researchgate.net/publication/261498006_On_the_efficiency_issues_of_virtual_keyboard_design (дата звернення 09.04.2023).

19. Набір документації по розробці застосунків для Android TV / URL: https://developer.android.com/training/tv (дата звернення 11.04.2023).

20. Рекомендації по швидкому створенню віртуальної клавіатури / URL: https://stackoverflow.com/questions/9577304/how-can-you-make-a-custom-keyboard-in-android/44939816#44939816 (дата звернення 14.04.2023).

21. Постачальники вмісту (Content Providers) / URL: https://developer.android.com/guide/topics/providers/content-provider-basics (дата звернення 20.04.2023).

22. Широкомовні сповіщення (Broadcasts) / URL: https://developer.android.com/guide/components/broadcasts (дата звернення 28.04.2023).

23. Вікна та макети (Views and Layouts) / URL: https://developer.android.com/develop/ui/views/layout/declaring-layout?hl=en (дата звернення 04.05.2023).

24. Створення застосунків, які визначають місцезнаходження / URL: https://developer.android.com/training/location (дата звернення 09.05.2023).

25. Мережевий стек Bluetooth. Загальна інформація / URL: https://developer.android.com/guide/topics/connectivity/bluetooth (дата звернення 14.05.2023).

26. Загальний огляд API по роботі з камерою / URL: https://developer.android.com/training/camerax/architecture (дата звернення 15.05.2023).

27. Створення анімованого інтерфейсу / URL: https://developer.android.com/develop/ui/views/animations/overview (дата звернення 17.05.2023).

28. Середня заробітна плата за видами економічної діяльності по місяцях / URL: https://ukrstat.gov.ua/operativ/operativ2005/gdn/Zarp_ek_m/Zp_ek_m_u/arh_zpm_u.html (дата звернення 25.05.2023).

29. Довідка з використання клавіатури AnySoftKeyboard / URL: https://github.com/AnySoftKeyboard/AnySoftKeyboard/wiki/Quick-Start-as-a-user (дата звернення 26.05.2023).

ДОДАТОК А

КОД ПРОГРАМИ

KbLayoutActivity.java // вікно з вибором потрібних мов

```
package com.liskovsoft.leankeyboard.activity.settings;

import android.content.Intent;
import android.os.Bundle;
import androidx.fragment.app.FragmentActivity;
import androidx.leanback.app.GuidedStepSupportFragment;
import com.liskovsoft.leankeyboard.fragments.settings.KbLayoutFragment;
import com.liskovsoft.leankeyboard.receiver.RestartServiceReceiver;

public class KbLayoutActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        GuidedStepSupportFragment.addAsRoot(this, new KbLayoutFragment(), android.R.id.content);
    }

    @Override
    protected void onStop() {
        super.onStop();

        // restart kbd service
        Intent intent = new Intent(this, RestartServiceReceiver.class);
        sendBroadcast(intent);
    }
}
```

KbSettingsActivity.java // вікно налаштувань клавіатури

```
package com.liskovsoft.leankeyboard.activity.settings;

import android.content.Intent;
import android.os.Bundle;
import androidx.fragment.app.FragmentActivity;
import androidx.leanback.app.GuidedStepSupportFragment;
import com.liskovsoft.leankeyboard.fragments.settings.KbSettingsFragment;
import com.liskovsoft.leankeyboard.receiver.RestartServiceReceiver;

public class KbSettingsActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        GuidedStepSupportFragment.addAsRoot(this, new KbSettingsFragment(), android.R.id.content);
    }

    @Override
    protected void onStop() {
        super.onStop();

        // restart kbd service
        Intent intent = new Intent(this, RestartServiceReceiver.class);
        sendBroadcast(intent);
    }
```

```
}
```

KbSettingsActivity2.java // вікно налаштувань клавіатури, яке додається на робочий стіл, як лаунчер

```
package com.liskovsoft.leankeyboard.activity.settings;

public class KbSettingsActivity2 extends KbSettingsActivity {
}
```

PermissionsActivity.java // вікно надання дозволу для доступу до системних функцій

```
package com.liskovsoft.leankeyboard.activity;

import android.content.Intent;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.fragment.app.FragmentActivity;
import com.liskovsoft.leankeyboard.helpers.PermissionHelpers;
import com.liskovsoft.leankeyboard.receiver.RestartServiceReceiver;

public class PermissionsActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        checkPermissions();
    }

    @Override
    protected void onStop() {
        super.onStop();

        // restart kbd service
        Intent intent = new Intent(this, RestartServiceReceiver.class);
        sendBroadcast(intent);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);

        checkPermissions();
    }

    private void checkPermissions() {
        if (!PermissionHelpers.hasMicPermissions(this)) {
            PermissionHelpers.verifyMicPermissions(this);
        } else if (!PermissionHelpers.hasStoragePermissions(this)) {
            PermissionHelpers.verifyStoragePermissions(this);
        } else {
            finish();
        }
    }
}
```

LeanbackImeService.java // містить функціонал обробки подій введення

```
package com.liskovsoft.leankeyboard.ime;
```

```
import android.annotation.SuppressLint;
import android.app.Service;
import android.content.Intent;
import android.inputmethodservice.InputMethodService;
import android.os.Build.VERSION;
import android.os.Handler;
import android.os.Message;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.InputDevice;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.inputmethod.CompletionInfo;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputConnection;
import androidx.core.text.BidiFormatter;
import com.liskovsoft.leankeyboard.ime.LeanbackKeyboardController.InputListener;
import com.liskovsoft.leankeyboard.utils.LeanKeyPreferences;

public class LeanbackImeService extends KeyMapperImeService {
    private static final String TAG = LeanbackImeService.class.getSimpleName();
    private static final boolean DEBUG = false;
    public static final String IME_CLOSE = "com.google.android.athome.action.IME_CLOSE";
    public static final String IME_OPEN = "com.google.android.athome.action.IME_OPEN";
    public static final int MAX_SUGGESTIONS = 10;
    static final int MODE_FREE_MOVEMENT = 1;
    static final int MODE_TRACKPAD_NAVIGATION = 0;
    private static final int MSG_SUGGESTIONS_CLEAR = 123;
    private static final int SUGGESTIONS_CLEAR_DELAY = 1000;
    private boolean mEnterSpaceBeforeCommitting;
    private View mInputView;
    private LeanbackKeyboardController mKeyboardController;
    private boolean mShouldClearSuggestions = true;
    private LeanbackSuggestionsFactory mSuggestionsFactory;
    public static final String COMMAND_RESTART = "restart";
    private boolean mForceShowKbd;

    @SuppressLint("HandlerLeak")
    private final Handler mHandler = new Handler() {
        public void handleMessage(Message msg) {
            if (msg.what == MSG_SUGGESTIONS_CLEAR && mShouldClearSuggestions) {
```

```
            mSuggestionsFactory.clearSuggestions();
            mKeyboardController.updateSuggestions(mSuggestionsFactory.getSuggestions());
            mShouldClearSuggestions = false;
        }

    }
};

private InputListener mInputListener = this::handleTextEntry;

@SuppressLint("NewApi")
@SuppressWarnings("deprecation")
public LeanbackImeService() {
    if (VERSION.SDK_INT < 21 && !enableHardwareAcceleration()) {
        Log.w("LbImeService", "Could not enable hardware acceleration");
    }
}

@Override
public void onCreate() {
    //setupDensity();
    super.onCreate();

    Log.d(TAG, "onCreate");

    initSettings();
}

private void setupDensity() {
    if (LeanKeyPreferences.instance(this).getEnlargeKeyboard()) {
        DisplayMetrics metrics = LeanbackUtils.createMetricsFrom(this, 1.3f);

        if (metrics != null) {
            getResources().getDisplayMetrics().setTo(metrics);
        }
    }
}

private void initSettings() {
    LeanKeyPreferences prefs = LeanKeyPreferences.instance(this);
    mForceShowKbd = prefs.getForceShowKeyboard();
```

```
    if (mKeyboardController != null) {
        mKeyboardController.setSuggestionsEnabled(prefs.getSuggestionsEnabled());
    }
}

private void clearSuggestionsDelayed() {
    if (!mSuggestionsFactory.shouldSuggestionsAmend()) {
        mHandler.removeMessages(MSG_SUGGESTIONS_CLEAR);
        mShouldClearSuggestions = true;
        mHandler.sendEmptyMessageDelayed(MSG_SUGGESTIONS_CLEAR, SUGGESTIONS_CLEAR_DELAY);
    }

}

private void handleTextEntry(final int type, final int keyCode, final CharSequence text) {
    final InputConnection connection = getCurrentInputConnection();
    if (connection != null) {
        boolean updateSuggestions;
        switch (type) {
            case InputListener.ENTRY_TYPE_STRING:
                clearSuggestionsDelayed();
                if (mEnterSpaceBeforeCommitting && mKeyboardController.enableAutoEnterSpace()) {
                    if (LeanbackUtils.isAlphabet(keyCode)) {
                        connection.commitText(" ", 1);
                    }

                    mEnterSpaceBeforeCommitting = false;
                }

                connection.commitText(text, 1);
                updateSuggestions = true;
                if (keyCode == LeanbackKeyboardView.ASCII_PERIOD) {
                    mEnterSpaceBeforeCommitting = true;
                }
                break;
            case InputListener.ENTRY_TYPE_BACKSPACE:
                clearSuggestionsDelayed();
                connection.deleteSurroundingText(1, 0);
                mEnterSpaceBeforeCommitting = false;
                updateSuggestions = true;
                break;
            case InputListener.ENTRY_TYPE_SUGGESTION:
```

```
            case InputListener.ENTRY_TYPE_VOICE:
                clearSuggestionsDelayed();
                if (!mSuggestionsFactory.shouldSuggestionsAmend()) {
                    connection.deleteSurroundingText(LeanbackUtils.getCharLengthBeforeCursor(connection),
LeanbackUtils.getCharLengthAfterCursor(connection));
                } else {
                    int location = LeanbackUtils.getAmpersandLocation(connection);
                    connection.setSelection(location, location);
                    connection.deleteSurroundingText(0, LeanbackUtils.getCharLengthAfterCursor(connection));
                }

                connection.commitText(text, 1);
                mEnterSpaceBeforeCommitting = true;
            case InputListener.ENTRY_TYPE_ACTION:  // User presses Go, Send, Search etc
                boolean result = sendDefaultEditorAction(true);

                if (result) {
                    hideWindow(); // SmartYouTubeTV: hide kbd on search page fix
                } else {
                    LeanbackUtils.sendEnterKey(connection);
                }

                updateSuggestions = false;
                break;
            case InputListener.ENTRY_TYPE_LEFT:
            case InputListener.ENTRY_TYPE_RIGHT:
                BidiFormatter formatter = BidiFormatter.getInstance();

                CharSequence textBeforeCursor = connection.getTextBeforeCursor(1000, 0);
                int lenBefore = 0;
                boolean isRtlBefore = false;
                //int rtlLenBefore = 0;
                if (textBeforeCursor != null) {
                    lenBefore = textBeforeCursor.length();
                    isRtlBefore = formatter.isRtl(textBeforeCursor);
                    //rtlLenBefore = LeanbackUtils.getRtlLenBeforeCursor(textBeforeCursor);
                }

                CharSequence textAfterCursor = connection.getTextAfterCursor(1000, 0);
                int lenAfter = 0;
                //int rtlLenAfter = 0;
                boolean isRtlAfter = false;
```

```
            if (textAfterCursor != null) {
                lenAfter = textAfterCursor.length();
                isRtlAfter = formatter.isRtl(textAfterCursor);
                //rtlLenAfter = LeanbackUtils.getRtlLenAfterCursor(textAfterCursor);
            }

            int index = lenBefore;
            if (type == InputListener.ENTRY_TYPE_LEFT) {
                if (lenBefore > 0) {
                    if (!isRtlBefore) {
                        index = lenBefore - 1;
                    } else {
                        if (lenAfter == 0) {
                            index = 1;
                        } else if (lenAfter == 1) {
                            index = 0;
                        } else {
                            index = lenBefore + 1;
                        }
                    }
                }

                //Log.d(TAG, String.format("direction key: before: lenBefore=%s, lenAfter=%s, rtlLenBefore=%s,
rtlLenAfter=%s", lenBefore, lenAfter, rtlLenBefore, rtlLenAfter));
                Log.d(TAG, String.format("direction key: before: lenBefore=%s, lenAfter=%s, isRtlBefore=%s",
lenBefore, lenAfter, isRtlBefore));
            } else {
                if (lenAfter > 0) {
                    if (!isRtlAfter) {
                        index = lenBefore + 1;
                    } else {
                        if (lenBefore == 0) {
                            index = lenAfter - 1;
                        } else if (lenBefore == 1) {
                            index = lenAfter + 1;
                        } else {
                            index = lenBefore - 1;
                        }
                    }
                }
```

```
                //Log.d(TAG, String.format("direction key: after: lenBefore=%s, lenAfter=%s, rtlLenBefore=%s,
rtlLenAfter=%s", lenBefore, lenAfter, rtlLenBefore, rtlLenAfter));
                Log.d(TAG, String.format("direction key: after: lenBefore=%s, lenAfter=%s, isRtlAfter=%s",
lenBefore, lenAfter, isRtlAfter));
            }

            Log.d(TAG, "direction key: index: " + index);

            connection.setSelection(index, index);
            updateSuggestions = true;
            break;
        case InputListener.ENTRY_TYPE_DISMISS:
            connection.performEditorAction(EditorInfo.IME_ACTION_NONE);
            updateSuggestions = false;
            break;
        case InputListener.ENTRY_TYPE_VOICE_DISMISS:
            connection.performEditorAction(EditorInfo.IME_ACTION_GO);
            updateSuggestions = false;
            break;
        default:
            updateSuggestions = true;
      }

      if (mKeyboardController.areSuggestionsEnabled() && updateSuggestions) {
        mKeyboardController.updateSuggestions(mSuggestionsFactory.getSuggestions());
      }
    }
  }

  @Override
  public View onCreateInputView() {
    mInputView = mKeyboardController.getView();
    mInputView.requestFocus();

    return mInputView;
  }

  @Override
  public void onDisplayCompletions(CompletionInfo[] infos) {
    if (mKeyboardController.areSuggestionsEnabled()) {
      mShouldClearSuggestions = false;
      mHandler.removeMessages(123);
```

```
            mSuggestionsFactory.onDisplayCompletions(infos);
            mKeyboardController.updateSuggestions(this.mSuggestionsFactory.getSuggestions());
        }

    }

    @Override
    public boolean onEvaluateFullscreenMode() {
        return false; // don't change it (true shows edit dialog above kbd)
    }

    /**
     * At this point, decision whether to show kbd taking place<br/>
     * <a href="https://stackoverflow.com/questions/7449283/is-it-possible-to-have-both-physical-keyboard-and-soft-keyboard-active-at-the-sa">More info</a>
     * @return whether to show kbd
     */
    @Override
    public boolean onEvaluateInputViewShown() {
        Log.d(TAG, "onEvaluateInputViewShown");
        return mForceShowKbd || super.onEvaluateInputViewShown();
    }

    // FireTV fix
    @Override
    public boolean onShowInputRequested(int flags, boolean configChange) {
        Log.d(TAG, "onShowInputRequested");
        return mForceShowKbd || super.onShowInputRequested(flags, configChange);
    }

    @Override
    public void onFinishInputView(boolean finishingInput) {
        super.onFinishInputView(finishingInput);
        sendBroadcast(new Intent(IME_CLOSE));
        mSuggestionsFactory.clearSuggestions();

        // NOTE: Trying to fix kbd without UI bug (telegram)
        reInitKeyboard();
    }

    @SuppressLint("NewApi")
    @Override
```

```
    public boolean onGenericMotionEvent(MotionEvent event) {
        return isInputViewShown() &&
                (event.getSource() & InputDevice.SOURCE_TOUCH_NAVIGATION) ==
InputDevice.SOURCE_TOUCH_NAVIGATION &&
                mKeyboardController.onGenericMotionEvent(event) || super.onGenericMotionEvent(event);
    }

    public void onHideIme() {
        requestHideSelf(InputMethodService.BACK_DISPOSITION_DEFAULT);
    }

    @Override
    public void onInitializeInterface() {
        mKeyboardController = new LeanbackKeyboardController(this, mInputListener);
        mKeyboardController.setHideWhenPhysicalKeyboardUsed(!mForceShowKbd);
        mEnterSpaceBeforeCommitting = false;
        mSuggestionsFactory = new LeanbackSuggestionsFactory(this, MAX_SUGGESTIONS);
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        // Hide keyboard on ESC key: https://github.com/yuliskov/SmartYouTubeTV/issues/142
        event = mapEscToBack(event);
        keyCode = mapEscToBack(keyCode);

        return isInputViewShown() && mKeyboardController.onKeyDown(keyCode, event) ||
super.onKeyDown(keyCode, event);
    }

    @Override
    public boolean onKeyUp(int keyCode, KeyEvent event) {
        // Hide keyboard on ESC key: https://github.com/yuliskov/SmartYouTubeTV/issues/142
        event = mapEscToBack(event);
        keyCode = mapEscToBack(keyCode);

        return isInputViewShown() && mKeyboardController.onKeyUp(keyCode, event) || super.onKeyUp(keyCode,
event);
    }

    private KeyEvent mapEscToBack(KeyEvent event) {
        if (event.getKeyCode() == KeyEvent.KEYCODE_ESCAPE) {
            // pay attention, you must pass the same action
```

```
            event = new KeyEvent(event.getAction(), KeyEvent.KEYCODE_BACK);
        }
        return event;
    }

    private int mapEscToBack(int keyCode) {
        if (keyCode == KeyEvent.KEYCODE_ESCAPE) {
            keyCode = KeyEvent.KEYCODE_BACK;
        }
        return keyCode;
    }

    @Override
    public int onStartCommand(final Intent intent, final int flags, final int startId) {
        if (intent != null) {
            Log.d(TAG, "onStartCommand: " + intent.toUri(0));

            if (intent.getBooleanExtra(COMMAND_RESTART, false)) {
                Log.d(TAG, "onStartCommand: trying to restart service");

                reInitKeyboard();

                return Service.START_REDELIVER_INTENT;
            }
        }

        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onStartInput(EditorInfo info, boolean restarting) {
        super.onStartInput(info, restarting);
        mEnterSpaceBeforeCommitting = false;
        mSuggestionsFactory.onStartInput(info);
        mKeyboardController.onStartInput(info);
    }

    @Override
    public void onStartInputView(EditorInfo info, boolean restarting) {
        super.onStartInputView(info, restarting);

        mKeyboardController.onStartInputView();
```

```
        sendBroadcast(new Intent(IME_OPEN));
        if (mKeyboardController.areSuggestionsEnabled()) {
            mSuggestionsFactory.createSuggestions();
            mKeyboardController.updateSuggestions(mSuggestionsFactory.getSuggestions());

            // NOTE: FileManager+ rename item fix: https://t.me/LeanKeyKeyboard/931
            // NOTE: Code below deletes text that has selection.
            //InputConnection connection = getCurrentInputConnection();
            //if (connection != null) {
            //    String text = LeanbackUtils.getEditorText(connection);
            //    connection.deleteSurroundingText(LeanbackUtils.getCharLengthBeforeCursor(connection),
LeanbackUtils.getCharLengthAfterCursor(connection));
            //    connection.commitText(text, 1);
            //}
        }
    }

    private void reInitKeyboard() {
        initSettings();

        if (mKeyboardController != null) {
            mKeyboardController.initKeyboards();
        }
    }
}
```

LeanbackSuggestionsFactory.java // відповідає за підказки під час введення тексту

```
package com.liskovsoft.leankeyboard.ime;

import android.inputmethodservice.InputMethodService;
import android.text.InputType;
import android.text.TextUtils;
import android.util.Log;
import android.view.inputmethod.CompletionInfo;
import android.view.inputmethod.EditorInfo;
import com.liskovsoft.leankeykeyboard.R;

import java.util.ArrayList;

public class LeanbackSuggestionsFactory {
    private static final String TAG = "LbSuggestionsFactory";
    private static final boolean DEBUG = Log.isLoggable(TAG, Log.DEBUG); // Use short text tag to fix "Log tag
exceeds limit of 23 characters"
    private static final int MODE_AUTO_COMPLETE = 2;
    private static final int MODE_DEFAULT = 0;
    private static final int MODE_DOMAIN = 1;
    private InputMethodService mContext;
    private int mMode;
```

```
    private int mNumSuggestions;
    private final ArrayList<String> mSuggestions = new ArrayList<>();

    public LeanbackSuggestionsFactory(InputMethodService context, int numSuggestions) {
        mContext = context;
        mNumSuggestions = numSuggestions;
    }

    public void clearSuggestions() {
        mSuggestions.clear();
        mSuggestions.add(null); // make room for user input, see
LeanbackKeyboardContainer.addUserInputToSuggestions
    }

    public void createSuggestions() {
        clearSuggestions();
        if (mMode == MODE_DOMAIN) {
            String[] domains = mContext.getResources().getStringArray(R.array.common_domains);
            int totalDomains = domains.length;

            for (int i = 0; i < totalDomains; ++i) {
                String domain = domains[i];
                mSuggestions.add(domain);
            }
        }

    }

    public ArrayList<String> getSuggestions() {
        return mSuggestions;
    }

    public void onDisplayCompletions(CompletionInfo[] infos) {
        createSuggestions();
        int len;
        if (infos == null) {
            len = 0;
        } else {
            len = infos.length;
        }

        for (int i = 0; i < len && mSuggestions.size() < mNumSuggestions && !TextUtils.isEmpty(infos[i].getText());
++i) {
            mSuggestions.add(i, infos[i].getText().toString());
        }

        if (DEBUG) {
            for (len = 0; len < mSuggestions.size(); ++len) {
                Log.d(TAG, "completion " + len + ": " + mSuggestions.get(len));
            }
        }

    }

    public void onStartInput(EditorInfo info) {
        mMode = MODE_DEFAULT;
        if ((info.inputType & InputType.TYPE_TEXT_FLAG_AUTO_COMPLETE) != 0) {
            mMode = MODE_AUTO_COMPLETE;
        }

        switch (LeanbackUtils.getInputTypeClass(info)) {
            case InputType.TYPE_CLASS_TEXT:
                switch (LeanbackUtils.getInputTypeVariation(info)) {
```

```
                case InputType.TYPE_DATETIME_VARIATION_TIME:
                case InputType.TYPE_TEXT_VARIATION_WEB_EMAIL_ADDRESS:
                    mMode = MODE_DOMAIN;
                    return;
                default:
                    return;
            }
        default:
    }
  }

  public boolean shouldSuggestionsAmend() {
    return mMode == MODE_DOMAIN;
  }
}
```

KeyboardBuilder.java // інтерфейс, створює екземляри клавіатури

```
package com.liskovsoft.leankeyboard.addons.keyboards;

import android.inputmethodservice.Keyboard;
import androidx.annotation.Nullable;

public interface KeyboardBuilder {
  Keyboard createAbcKeyboard();
  Keyboard createSymKeyboard();
  Keyboard createNumKeyboard();
}
```

KeyboardInfo.java // інтерфейс, надає інформацію про клавіатуру

```
package com.liskovsoft.leankeyboard.addons.keyboards;

public interface KeyboardInfo {
  String getLangCode();
  void setLangCode(String langCode);
  String getLangName();
  void setLangName(String langName);
  boolean isEnabled();
  void setEnabled(boolean enabled);
  boolean isAzerty();
  void setIsAzerty(boolean enabled);
}
```

ResizeableLeanbackKeyboardView.java // модифікована версія головного вікна клавіатури з функцією зміни розміру

```
package com.liskovsoft.leankeyboard.addons.resize;

import android.content.Context;
import android.inputmethodservice.Keyboard;
import android.inputmethodservice.Keyboard.Key;
import android.util.AttributeSet;
import com.liskovsoft.leankeyboard.ime.LeanbackKeyboardView;
import com.liskovsoft.leankeyboard.utils.LeanKeyPreferences;

import java.util.List;

public class ResizeableLeanbackKeyboardView extends LeanbackKeyboardView {
  private final LeanKeyPreferences mPrefs;
  private final int mKeyTextSizeOrigin;
  private final int mModeChangeTextSizeOrigin;
```

```
    private final float mSizeFactor = 1.3f;
    private int mKeyOriginWidth;

    public ResizeableLeanbackKeyboardView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mPrefs = LeanKeyPreferences.instance(getContext());
        mKeyTextSizeOrigin = mKeyTextSize;
        mModeChangeTextSizeOrigin = mModeChangeTextSize;
    }

    @Override
    public void setKeyboard(Keyboard keyboard) {
        if (mPrefs.getEnlargeKeyboard()) {
            mKeyTextSize = (int) (mKeyTextSizeOrigin * mSizeFactor);
            mModeChangeTextSize = (int) (mModeChangeTextSizeOrigin * mSizeFactor);

            keyboard = updateKeyboard(keyboard);
        } else {
            mKeyTextSize = mKeyTextSizeOrigin;
            mModeChangeTextSize = mModeChangeTextSizeOrigin;
        }

        mPaint.setTextSize(mKeyTextSize);

        super.setKeyboard(keyboard);
    }

    private Keyboard updateKeyboard(Keyboard keyboard) {
        List<Key> keys = keyboard.getKeys();

        if (isNotSizedYet(keys.get(0))) {
            for (Key key : keys) {
                key.width *= mSizeFactor;
                key.height *= mSizeFactor;
                key.gap *= mSizeFactor;
                key.x *= mSizeFactor;
                key.y *= mSizeFactor;
            }
        }

        KeyboardWrapper wrapper = KeyboardWrapper.from(keyboard, getContext());
        wrapper.setHeightFactor(mSizeFactor);
        wrapper.setWidthFactor(mSizeFactor);

        return wrapper;
    }

    private boolean isNotSizedYet(Key key) {
        boolean result = false;

        if (mKeyOriginWidth == 0) {
            mKeyOriginWidth = key.width;
        }

        if (mKeyOriginWidth == key.width) {
            result = true;
        }

        return result;
    }
}
```

ThemeManager.java // допоміжний клас для підтримки різних зовнішніх виглядів клавіатури

```
package com.liskovsoft.leankeyboard.addons.theme;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.drawable.Drawable;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import androidx.core.content.ContextCompat;
import com.liskovsoft.leankeyboard.ime.LeanbackKeyboardView;
import com.liskovsoft.leankeyboard.utils.LeanKeyPreferences;
import com.liskovsoft.leankeykeyboard.R;

public class ThemeManager {
    private static final String TAG = ThemeManager.class.getSimpleName();
    private final Context mContext;
    private final RelativeLayout mRootView;
    private final LeanKeyPreferences mPrefs;

    public ThemeManager(Context context, RelativeLayout rootView) {
        mContext = context;
        mRootView = rootView;
        mPrefs = LeanKeyPreferences.instance(mContext);
    }

    public void updateKeyboardTheme() {
        String currentThemeId = mPrefs.getCurrentTheme();

        if (LeanKeyPreferences.THEME_DEFAULT.equals(currentThemeId)) {
            applyKeyboardColors(
                    R.color.keyboard_background,
                    R.color.candidate_background,
                    R.color.enter_key_font_color,
                    R.color.key_text_default
            );
            applyShiftDrawable(-1);
        } else {
            applyForTheme((String themeId) -> {
                Resources resources = mContext.getResources();
                int keyboardBackgroundResId = resources.getIdentifier("keyboard_background_" + themeId.toLowerCase(),
"color", mContext.getPackageName());
                int candidateBackgroundResId = resources.getIdentifier("candidate_background_" + themeId.toLowerCase(),
"color", mContext.getPackageName());
                int enterFontColorResId = resources.getIdentifier("enter_key_font_color_" + themeId.toLowerCase(),
"color", mContext.getPackageName());
                int keyTextColorResId = resources.getIdentifier("key_text_default_" + themeId.toLowerCase(), "color",
mContext.getPackageName());

                applyKeyboardColors(
                        keyboardBackgroundResId,
                        candidateBackgroundResId,
                        enterFontColorResId,
                        keyTextColorResId
                );

                int shiftLockOnResId = resources.getIdentifier("ic_ime_shift_lock_on_" + themeId.toLowerCase(),
"drawable", mContext.getPackageName());

                applyShiftDrawable(shiftLockOnResId);
            });
```

```
        }
    }

    public void updateSuggestionsTheme() {
        String currentTheme = mPrefs.getCurrentTheme();

        if (LeanKeyPreferences.THEME_DEFAULT.equals(currentTheme)) {
            applySuggestionsColors(
                    R.color.candidate_font_color
            );
        } else {
            applyForTheme((String themeId) -> {
                Resources resources = mContext.getResources();
                int candidateFontColorResId = resources.getIdentifier("candidate_font_color_" + themeId.toLowerCase(),
"color", mContext.getPackageName());
                applySuggestionsColors(candidateFontColorResId);
            });
        }
    }

    private void applyKeyboardColors(
            int keyboardBackground,
            int candidateBackground,
            int enterFontColor,
            int keyTextColor) {

        RelativeLayout rootLayout = mRootView.findViewById(R.id.root_ime);

        if (rootLayout != null) {
            rootLayout.setBackgroundColor(ContextCompat.getColor(mContext, keyboardBackground));
        }

        View candidateLayout = mRootView.findViewById(R.id.candidate_background);

        if (candidateLayout != null) {
            candidateLayout.setBackgroundColor(ContextCompat.getColor(mContext, candidateBackground));
        }

        Button enterButton = mRootView.findViewById(R.id.enter);

        if (enterButton != null) {
            enterButton.setTextColor(ContextCompat.getColor(mContext, enterFontColor));
        }

        LeanbackKeyboardView keyboardView = mRootView.findViewById(R.id.main_keyboard);

        if (keyboardView != null) {
            keyboardView.setKeyTextColor(ContextCompat.getColor(mContext, keyTextColor));
        }
    }

    private void applySuggestionsColors(int candidateFontColor) {
        LinearLayout suggestions = mRootView.findViewById(R.id.suggestions);

        if (suggestions != null) {
            int childCount = suggestions.getChildCount();

            Log.d(TAG, "Number of suggestions: " + childCount);

            for (int i = 0; i < childCount; i++) {
                View child = suggestions.getChildAt(i);

                Button candidateButton = child.findViewById(R.id.text);
```

```
            if (candidateButton != null) {
                candidateButton.setTextColor(ContextCompat.getColor(mContext, candidateFontColor));
            }
        }
    }
}

private void applyShiftDrawable(int resId) {
    LeanbackKeyboardView keyboardView = mRootView.findViewById(R.id.main_keyboard);

    if (keyboardView != null && resId > 0) {
        Drawable drawable = ContextCompat.getDrawable(mContext, resId);

        keyboardView.setCapsLockDrawable(drawable);
    }
}

private void applyForTheme(ThemeCallback callback) {
    String currentThemeId = mPrefs.getCurrentTheme();
    Resources resources = mContext.getResources();
    String[] themes = resources.getStringArray(R.array.keyboard_themes);

    for (String theme : themes) {
        String[] split = theme.split("\\|");
        String themeName = split[0];
        String themeId = split[1];

        if (currentThemeId.equals(themeId)) {
            callback.onThemeFound(themeId);

            break;
        }
    }
}

private interface ThemeCallback {
    void onThemeFound(String themeId);
}
}
```

ActivityListener.java // інтерфейс загального обробника події Activity#onActivityResult

```
package com.liskovsoft.leankeyboard.addons.voice;

import android.content.Intent;

interface ActivityListener {
    void onActivityResult(int requestCode, int resultCode, Intent data);
}
```

RecognizerIntentActivity.java // вікно, яке запускається при голосовому введенні

```
package com.liskovsoft.leankeyboard.addons.voice;

import android.content.Intent;
import android.os.Bundle;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class RecognizerIntentActivity extends AppCompatActivity {
    public static RecognizerCallback sCallback;
    private VoiceSearchBridge mBridge;
```

```
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mBridge = new VoiceSearchBridge(this, searchText -> sCallback.openSearchPage(searchText));

        mBridge.displaySpeechRecognizers();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        mBridge.onActivityResult(requestCode, resultCode, data);

        finish();
    }
}
```

SystemVoiceDialog.java // діалог голосового введення, який вбудован у RecognizerActivity

```
package com.liskovsoft.leankeyboard.addons.voice;

import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.speech.RecognizerIntent;

import java.util.List;

class SystemVoiceDialog implements VoiceDialog, ActivityListener {
    private static final int SPEECH_REQUEST_CODE = 11;
    private final Activity mActivity;
    private final SearchCallback mCallback;

    public SystemVoiceDialog(Activity activity, SearchCallback callback) {
        mActivity = activity;
        mCallback = callback;
    }

    public boolean displaySpeechRecognizer() {
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        try {
            mActivity.startActivityForResult(intent, SPEECH_REQUEST_CODE);
        } catch (ActivityNotFoundException e) {
            e.printStackTrace();
            return false;
        }

        return true;
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        // got speech-to-text result, switch to the search page
        if (requestCode == SPEECH_REQUEST_CODE && resultCode == -1) {
            List<String> results = data.getStringArrayListExtra(
                    RecognizerIntent.EXTRA_RESULTS);
            if (results != null && results.size() > 0) {
                mCallback.openSearchPage(results.get(0));
```

```
        }
      }
    }
}
```

KbLayoutFragment.java // налаштування мов клавіатури

```
package com.liskovsoft.leankeyboard.fragments.settings;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.leanback.widget.GuidanceStylist.Guidance;
import com.liskovsoft.leankeyboard.addons.keyboards.intkeyboards.KeyboardInfoAdapter;
import com.liskovsoft.leankeyboard.addons.keyboards.intkeyboards.CheckedSource;
import com.liskovsoft.leankeyboard.addons.keyboards.intkeyboards.CheckedSource.CheckedItem;
import com.liskovsoft.leankeykeyboard.R;

public class KbLayoutFragment extends BaseSettingsFragment {
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);

        initCheckedItems();
    }

    @NonNull
    @Override
    public Guidance onCreateGuidance(Bundle savedInstanceState) {
        String title = getActivity().getResources().getString(R.string.kb_layout);
        String desc = getActivity().getResources().getString(R.string.kb_layout_desc);
        Drawable icon = ContextCompat.getDrawable(getActivity(), R.drawable.ic_launcher);

        return new Guidance(
                title,
                desc,
                "",
                icon
        );
    }

    private void initCheckedItems() {
        CheckedSource source = new KeyboardInfoAdapter(getActivity());
        for (CheckedItem item : source.getItems()) {
            addCheckedAction(item.getTitle(), item::getChecked, item::onClick);
        }
    }
}
```

KbSettingsFragment.java // загальні налаштування клавіатури

```
package com.liskovsoft.leankeyboard.fragments.settings;

import android.content.Context;
import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.leanback.app.GuidedStepSupportFragment;
import androidx.leanback.widget.GuidanceStylist.Guidance;
```

```
import com.liskovsoft.leankeyboard.activity.settings.KbActivationActivity;
import com.liskovsoft.leankeykeyboard.R;

public class KbSettingsFragment extends BaseSettingsFragment {
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);

        addNextAction(R.string.activate_keyboard, () -> {
            Intent intent = new Intent(getActivity(), KbActivationActivity.class);
            startActivity(intent);
        });

        addNextAction(R.string.change_layout, () -> startGuidedFragment(new KbLayoutFragment()));

        addNextAction(R.string.change_theme, () -> startGuidedFragment(new KbThemeFragment()));

        addNextAction(R.string.misc, () -> startGuidedFragment(new MiscFragment()));

        addNextAction(R.string.about_desc, () -> startGuidedFragment(new AboutFragment()));
    }

    @NonNull
    @Override
    public Guidance onCreateGuidance(Bundle savedInstanceState) {
        String title = getActivity().getResources().getString(R.string.kb_settings);
        String desc = getActivity().getResources().getString(R.string.kb_settings_desc);
        Drawable icon = ContextCompat.getDrawable(getActivity(), R.drawable.ic_launcher);

        return new Guidance(
                title,
                desc,
                "",
                icon
        );
    }

    private void startGuidedFragment(GuidedStepSupportFragment fragment) {
        if (getFragmentManager() != null) {
            GuidedStepSupportFragment.add(getFragmentManager(), fragment);
        }
    }
}
```

KbThemeFragment.java // налаштування зовнішнього вигляду калавіатури

```
package com.liskovsoft.leankeyboard.fragments.settings;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.leanback.widget.GuidanceStylist.Guidance;
import com.liskovsoft.leankeyboard.utils.LeanKeyPreferences;
import com.liskovsoft.leankeykeyboard.R;

public class KbThemeFragment extends BaseSettingsFragment {
    private Context mContext;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
```

```
        mContext = context;

        initRadioItems();
    }

    @NonNull
    @Override
    public Guidance onCreateGuidance(Bundle savedInstanceState) {
        String title = getActivity().getResources().getString(R.string.kb_theme);
        String desc = getActivity().getResources().getString(R.string.kb_theme_desc);
        Drawable icon = ContextCompat.getDrawable(getActivity(), R.drawable.ic_launcher);

        return new Guidance(
                title,
                desc,
                "",
                icon
        );
    }

    private void initRadioItems() {
        String[] themes = mContext.getResources().getStringArray(R.array.keyboard_themes);

        LeanKeyPreferences prefs = LeanKeyPreferences.instance(mContext);
        String currentTheme = prefs.getCurrentTheme();

        for (String theme : themes) {
            String[] split = theme.split("\\|");
            String themeName = split[0];
            String themeId = split[1];
            addRadioAction(themeName, () -> currentTheme.equals(themeId), (checked) ->
prefs.setCurrentTheme(themeId));
        }
    }
}
```

LeanKeyPreferences.java // відповідає за збереження налаштувань у приватній директорії застосунку

```
package com.liskovsoft.leankeyboard.utils;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

public final class LeanKeyPreferences {
    private static final String APP_RUN_ONCE = "appRunOnce";
    private static final String BOOTSTRAP_SELECTED_LANGUAGE = "bootstrapSelectedLanguage";
    private static final String APP_KEYBOARD_INDEX = "appKeyboardIndex";
    private static final String FORCE_SHOW_KEYBOARD = "forceShowKeyboard";
    private static final String ENLARGE_KEYBOARD = "enlargeKeyboard";
    private static final String KEYBOARD_THEME = "keyboardTheme";
    public static final String THEME_DEFAULT = "Default";
    public static final String THEME_DARK = "Dark";
    public static final String THEME_DARK2 = "Dark2";
    public static final String THEME_DARK3 = "Dark3";
    private static final String SUGGESTIONS_ENABLED = "suggestionsEnabled";
    private static final String CYCLIC_NAVIGATION_ENABLED = "cyclicNavigationEnabled";
    private static final String AUTODETECT_LAYOUT = "autodetectLayout";
    private static LeanKeyPreferences sInstance;
    private final Context mContext;
    private SharedPreferences mPrefs;

    public static LeanKeyPreferences instance(Context ctx) {
```

```
    if (sInstance == null)
        sInstance = new LeanKeyPreferences(ctx);
    return sInstance;
}

public LeanKeyPreferences(Context context) {
    mContext = context.getApplicationContext();
    mPrefs = PreferenceManager.getDefaultSharedPreferences(mContext);
}

public boolean isRunOnce() {
    return mPrefs.getBoolean(APP_RUN_ONCE, false);
}

public void setRunOnce(boolean runOnce) {
    mPrefs.edit()
            .putBoolean(APP_RUN_ONCE, runOnce)
            .apply();
}

public void setPreferredLanguage(String name) {
    mPrefs.edit()
            .putString(BOOTSTRAP_SELECTED_LANGUAGE, name)
            .apply();
}

public String getPreferredLanguage() {
    return mPrefs.getString(BOOTSTRAP_SELECTED_LANGUAGE, "");
}

public int getKeyboardIndex() {
    return mPrefs.getInt(APP_KEYBOARD_INDEX, 0);
}

public void setKeyboardIndex(int idx) {
    mPrefs.edit()
            .putInt(APP_KEYBOARD_INDEX, idx)
            .apply();
}

public boolean getForceShowKeyboard() {
    return mPrefs.getBoolean(FORCE_SHOW_KEYBOARD, true);
}

public void setForceShowKeyboard(boolean force) {
    mPrefs.edit()
            .putBoolean(FORCE_SHOW_KEYBOARD, force)
            .apply();
}

public boolean getEnlargeKeyboard() {
    return mPrefs.getBoolean(ENLARGE_KEYBOARD, false);
}

public void setEnlargeKeyboard(boolean enlarge) {
    mPrefs.edit()
            .putBoolean(ENLARGE_KEYBOARD, enlarge)
            .apply();
}

public void setCurrentTheme(String theme) {
    mPrefs.edit()
            .putString(KEYBOARD_THEME, theme)
```

```
            .apply();
    }

    public String getCurrentTheme() {
        return mPrefs.getString(KEYBOARD_THEME, THEME_DARK3);
    }

    public void setSuggestionsEnabled(boolean enabled) {
        mPrefs.edit()
                .putBoolean(SUGGESTIONS_ENABLED, enabled)
                .apply();
    }

    public boolean getSuggestionsEnabled() {
        return mPrefs.getBoolean(SUGGESTIONS_ENABLED, true);
    }

    public void setCyclicNavigationEnabled(boolean enabled) {
        mPrefs.edit()
                .putBoolean(CYCLIC_NAVIGATION_ENABLED, enabled)
                .apply();
    }

    public boolean getCyclicNavigationEnabled() {
        return mPrefs.getBoolean(CYCLIC_NAVIGATION_ENABLED, false);
    }

    public boolean getAutodetectLayout() {
        return mPrefs.getBoolean(AUTODETECT_LAYOUT, false);
    }
}
```

LocaleUtility.java // функції роботи з локалью

```
package com.liskovsoft.leankeyboard.utils;

import android.content.Context;
import android.content.res.Configuration;
import android.os.Build.VERSION;
import android.util.Log;

import java.util.Locale;

public class LocaleUtility extends LocaleScript {
    private static final String TAG = LocaleUtility.class.getSimpleName();

    public static Locale getSystemLocale(Context context) {
        return getSystemLocale(context.getResources().getConfiguration());
    }

    public static void setSystemLocale(Context context, Locale locale) {
        setSystemLocale(context.getResources().getConfiguration(), locale);
    }

    @SuppressWarnings("deprecation")
    public static void setSystemLocale(Configuration config, Locale locale) {
        if (VERSION.SDK_INT < 24) {
            config.locale = locale;
        } else {
            config.setLocale(locale);
        }
    }
```

```
    @SuppressWarnings("deprecation")
    public static Locale getSystemLocale(Configuration config) {
        if (VERSION.SDK_INT < 24) {
            return config.locale;
        } else {
            return config.getLocales().get(0);
        }
    }

    /**
     * <a href="https://stackoverflow.com/questions/40221711/android-context-getresources-updateconfiguration-
deprecated/40704077#40704077">Modern Solution</a>
     */
    @SuppressWarnings("deprecation")
    public static void forceLocaleOld(Context ctx, Locale locale) {
        Locale.setDefault(locale);
        Configuration config = ctx.getResources().getConfiguration();
        LocaleUtility.setSystemLocale(config, locale);
        ctx.getResources().updateConfiguration(config,
                ctx.getResources().getDisplayMetrics());
    }

    public static void switchRuLocale(Context ctx) {
        Log.d(TAG, "Trying to switch locale back and forward");
        Locale savedLocale = Locale.getDefault();
        LocaleUtility.forceLocaleOld(ctx, new Locale("ru"));
        LocaleUtility.forceLocaleOld(ctx, savedLocale);
    }
}
```

ДОДАТОК Б

Відзив керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Кваліфікаційна_робота.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна_робота.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Keyboard.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація кваліфікаційної роботи