Лабораторні роботи 1-5

Створено системою Doxygen 1.9.1

1 Звіт з лабораторних робіт 1-5

за дисципліною "Проектування систем під мобільні платформи"
студента групи ПА-17-2
Панасенка Єгора Сергійовича
Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2017-2018 навч.р.

Звіт доступний за посиланням
https://gaurapanasenko.github.io/unilab_opt/DSfMP/html/index.html.
Вихідний код доступний за посиланням
https://github.com/gaurapanasenko/unilab/tree/master/08/DSfMP

1.0.1 Постановка задачі лабораторної роботи №1

Тема: Вступ у розробку Android - додатків

Мета роботи: знайомство з інструментами розробки Android- додатків; отримання навичок складання та налагодження програм Android-додатків.

1.0.1.1 Методичні вказівки Android - операційна система для мобільних пристроїв: смартфонів, планшетних комп'ютерів, КПК. В даний час саме Android є найбільш широко використовуваною операційною системою для мобільних пристроїв. Це безкоштовна операційна система, заснована на Linux з інтерфейсом програмування Java.

Платформа Android об'єднує операційну систему, побудовану на основі ядра ОС Linux, проміжне програмне забезпечення і вбудовані мобільні додатки. Розробка і розвиток мобільної платформи Android виконується в рамках проекту AOSP (Android Open Source Project) під керуванням OHA (Open Handset Alliance), керує всім процесом Google.

Android підтримує фонове виконання завдань; надає розвинену бібліотеку елементів інтерфейсу користувача; підтримує 2D- і BD-графіку, використовуючи OpenGL стандарт; підтримує доступ до файлової системи і вбудованої бази даних SQLite.

Додаток для Android пишеться на мові Java, а середовище розробки можна вибрати, наприклад, з популярних засобів розробки, таких як Android Studio або Eclipse.

Android Studio - середовище розробки під Android, заснована на IntelliJ IDEA. Вона надає інтегровані інструменти для розробки і налагодження. Для налагодження додатків використовується емулятор телефону - віртуальна машина, на якій буде запускатися створене додаток.

Щоб створити емулятор телефону, вибираємо в меню Android Studio пункти SDK Manager | Tools | AVD Manager | Device Definitions.

Далі потрібно вибрати потрібну версію віртуального пристрою і при необхідності відредагувати його параметри.

Далі розглянемо етапи створення нового проекту. Після запуску Android Studio вибираємо New Project, з'явиться діалогове вікно майстра.

- Поле Application name - ім'я, яке буде відображатися в заголовку програми.
- Поле Company Domain служить для вказівки адреси вашого сайту.
- Поле Package name формує спеціальний Java-пакет на основі вашого імені з попереднього поля.
- Поле Project location дозволяє вибрати місце на диску для створюваного проекту.
- Після натискання на кнопку Next переходимо до наступного вікна. Тут вибираємо типи пристроїв, під які будемо розробляти додаток. У більшості випадків ми буде писати для смартфонів і планшетів, тому залишаємо прапорець у першого пункту. Також можна створювати додатки для Android TV, Android Wear і Glass.
- Далі необхідно вибрати зовнішній вигляд екрану програми. Виберемо, наприклад, варіант Blank Activity. Після натискання кнопки Finish студія формує проект і створює необхідну структуру з різних файлів і папок.

У лівій частині середовища розробки на вкладці Android з'явиться ієрархічний список з папок, які відносяться до проекту. Вкладка Android містить дві основні папки: app і Gradle Scripts. Перша папка app є окремим модулем для програми та містить всі необхідні файли програми - код, ресурси картинок і т.п. Друга папка служить для різних налаштувань для керування проектом. Розкриваємо папку app. У ній знаходяться три папки: manifest, java, res.

Папка manifest містить єдиний файл маніфесту AndroidManifest.xml. У цьому файлі повинні бути оголошені всі активності, служби, приймачі і контент-провайдери додатки. Також він повинен містити необхідні додатком дозволу. «AndroidManifest.xml» можна розглядати, як опис для розгортання Android-додатки.

Папка java містить три папки - робочу і для тестів. Робоча папка має назву пакета і містить файли класів. Зараз там один клас MainActivity.

Папка res містить файли ресурсів, розбитих на окремі папки.

Запуск програми здійснюється вибором команди Run в пункті меню Run. Після цього в емуляторі завантажиться розроблена програма, на екрані з'явиться вікно програми з написом «Hello World!» і заголовком програми.

1.0.1.2 Порядок виконання роботи

1. Вивчити методичні вказівки до виконання лабораторної роботи №1.
2. Запустити на виконання Android Studio.
3. Якщо буде використовуватися зовнішній емулятор для запуску Android-додатків (наприклад, BlueStacks або Genymotion), завантажити його.
4. Створити в середовищі Android Studio проект.
5. Вивчити вміст файлу AndroidManifest.xml з папки manifest і папку res, що містить файли ресурсів проекту.
6. Запустити створене додаток в емуляторі Android і спостерігати за появою цього додатка і результатів його роботи у вікні додатків емулятора.
7. Вивчити вміст файлів проекту activity_main.xml і MainActivity.java. Відредагувати їх вміст для отримання нового тексту на екрані і запустити додаток в емуляторі Android.

1.0.2 Постановка задачі лабораторної роботи №2

Тема: Створення призначених для користувача інтерфейсів і використання елементів управління в додатках під Android

Мета роботи: навчитися створювати додатки з елементами управління; отримати навички використання елементів управління в Android-додатках.

1.0.2.1 Методичні вказівки Часто при роботі додатка на екрані дисплея повинні знаходитися елементи управління. Елементи управління - це доступні для маніпулювання екранні об'єкти. Їх можна розділити на чотири основні категорії:

- командні елементи управління, які застосовуються для виконання функцій;
- елементи вибору, що дозволяють вибирати дані або налаштування;
- елементи введення, що застосовуються для введення даних;
- елементи відображення, використовувані для виведення.

Командні елементи управління виконують деякі дії. Головним командним елементом є кнопка, яка має безліч варіантів відображення. Дія виконується відразу після натискання на кнопку. Часто особливим чином виділяється кнопка за замовчуванням, відповідна найбільш часто використовуваному дії. Кнопки, поміщені на панель інструментів, зазвичай стають квадратними, втрачають текстовий напис і обзаводяться піктограмою - поясненням у вигляді графічного значка.

Елементи вибору дозволяють користувачеві вибрати з групи допустимих об'єктів той, з яким буде вчинено дію.

Елементи вибору застосовуються також для дій з налаштування. Поширеними елементами вибору є прапорці і списки. Клацнувши по прапорці, користувач негайно побачить що з'явилась галочка. Елементи управління типу «список» дозволяють здійснювати вибір з кінцевого безлічі текстових рядків, кожна з яких представляє команду, об'єкт або ознака. Користувач може вибрати єдиний рядок тексту, натиснувши на неї.

Елементи введення дають користувачеві можливість не тільки вибирати існуючі відомості, але і вводити нову інформацію. Найпростіший елемент - поле редагування тексту (поле введення). В цю категорію потрапляють також такі елементи управління, як лічильники і повзунки.

Елементи управління відображенням використовуються для управління візуальним представленням інформації на екрані. Типовими прикладами елементів відображення є роздільники і смуги прокрутки. Сюди ж входять роздільники сторінок, лінійки, направляючі, сітки і рамки.

Кожному об'єкту потрібно задати розміри, координати, колір, текст і т.д. Android підтримує спосіб, заснований на XML- розмітці, який нагадує розмітку веб-сторінки. Можна використовувати і візуальний спосіб перетягування об'єктів за допомогою миші.

Файли XML-розмітки знаходяться в папці res / layout проекту. Папка містить ресурси, які не пов'язані з кодом. Крім розмітки, там же містяться зображення, звуки, рядки для локалізації і т.д.

Відкриємо створений на практичному занятті №2 проект. Коли розмітка відкрита в графічному поданні, то зліва від основної частини редактора коду можна побачити панель інструментів, в якій згруповані різні елементи за категоріями: Widgets, Texts, Layouts і т.д.

У групі Images знайдіть елемент ImageButton, перетягніть його на форму і відпустіть. Далі необхідно вибрати зображення для кнопки.

У вкладці Component Tree виберемо елемент Constraint Layout (екран). В панелі властивостей Properties відобразяться найуживаніші властивості обраного компонента. До них відносяться ідентифікатор, ширина і висота. Вибираємо View all properties (дві стрілочки), щоб відкрити всі властивості компонента. Знайдіть властивість background. Клацніть поруч з цим словом у другій колонці. З'явиться текстове поле, в яке можна ввести значення вручну і кнопка з трьома крапками, яка запустить діалогове вікно для створення ресурсу.Переходимо на вкладку Color і вибираємо потрібний колір.

Далі можна поміняти картинку для графічної кнопки. Знаходимо відповідне зображення і копіюємо його в папку res / drawable проекту. Потім виділяємо елемент ImageButton на формі і в панелі властивостей вибираємо властивість srcCompat. Знову клацаємо на кнопці з трьома крапками і вибираємо ресурс в категорії Drawable - там має з'явитися ресурс з ім'ям доданого раніше файлу. Там же в вікні властивостей знаходимо властивість onClick і вручну прописуємо onClick - це буде ім'ям методу для обробки натискання на кнопку.

Далі встановіть курсор миші всередині тексту "onClick" у кнопки (переключившись в режим Text) і натисніть комбінацію Alt + Enter. У спливаючому вікні вибрати варіант Create 'onClick (View)' in 'MainActivity'. У коді класу MainActivity з'явиться заготовка для обробки клацання кнопки. Необхідно набрати наступний текст:

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    private TextView mHelloTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mHelloTextView=(TextView)findViewById(R.id.textView);
    }
    public void onClick(View view) {
        mHelloTextView.setText("Hello!");
    }
}
```

Далі необхідно запустити програму в емуляторі Android, натиснути на створену кнопку і спостерігати за зміною тексту на екрані.

1.0.2.2 Порядок виконання роботи

1. Вивчити методичні вказівки до виконання лабораторної роботи №2.
2. Запустити на виконання Android Studio.
3. Якщо буде використовуватися зовнішній емулятор для запуску Android-додатків (наприклад, BlueStacks або Genymotion), завантажити його.
4. Відкрити в середовищі Android Studio проект, розроблений в ході виконання лабораторної роботи №2 або створити новий проект.
5. Розмістити на формі елементи управління ImageButton і TextView, налаштувати їх параметри.
6. У вікні java-коду проекту додати рядки обробки натискання на кнопку.
7. Запустити створений додаток в емуляторі Android і спостерігати за появою цього додатка і результатів його роботи у вікні емулятора.
8. Додати в проект інші елементи управління, налаштувати їх властивості та перевірити роботу програми в емуляторі Android.

1.0.3 Постановка задачі лабораторної роботи №2

Тема: 2Б-анімація, створення і використання служб в додатках під Android

Мета роботи: знайомство з можливостями створення Android- додатків з використанням анімації; отримати навички створення Android-додатків з використанням ID-анімації.

1.0.3.1 Методичні вказівки

Android надає потужні API для анімації елементів призначених для користувацького інтерфейсу і побудови 2D і 3D зображень. Платформа Android надає дві системи анімації: анімація властивостей, що з'явилася в Android 3.0, і анімація компонентів для користувача інтерфейсу (спадкоємців класу View).

Анімація властивостей (Property Animation) дозволяє визначити анімацію для зміни будь-якого властивості об'єкта, незалежно від того зображується воно на екрані чи ні.

Анімація компонентів для користувача інтерфейсу використовується для реалізації анімації перетворень над спадкоємцями класу View. Для розрахунку анімації перетворень використовується наступна інформація: початкова точка, кінцева точка, розмір, поворот і інші загальні аспекти анімації. Анімація перетворень може виконувати серії простих змін вмісту екземпляра класу View. Наприклад, для текстового поля можна переміщати, обертати, розтягувати, стискати текст, якщо визначено фонове зображення, воно повинно змінюватися разом з текстом. Пакет android.view.↩ animation надає всі класи, необхідні для реалізації анімації перетворень.

Для завдання послідовності інструкцій анімації перетворень використовується XML код. Такі файли з XML кодом повинні розташовуватися в папці res / anim / проекту.

Розглянемо послідовність розробки програми, що використовує анімацію.

Створимо новий проект в Android Studio. Далі підготуємо кілька додаткових файлів, які будуть використовуватися при анімації. В папці drawable проекту створимо файл sun.xml з таким вмістом:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android" android:dither="true" android:shape="oval" >
<gradient android:endColor="#ffff6600" android:gradientRadius="150" android:startColor="#ffffcc00" android:type="radial"
android:useLevel="false" />
<size
android:height="150dp" android:width="150dp" />
</shape>
```

Тут для зображення сонця використовується овал, а також для кольору застосовується градієнт - плавну зміну кольору від темно-жовтого (startColor) до світло-жовтого (endColor).

У тій же папці drawable створимо новий файл sky.xml такого змісту:
```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android" android:dither="true" android:shape="rectangle" >
<gradient android:angle="90"
android:endColor="#ff000033 "
android:startColor="#ff0000ff"
/>
</shape>
```

Тут задана фігура в вигляді прямокутника (rectangle) з блакитним градієнтом від нижнього краю до верхнього.

В тієї ж папці drawable створимо новий файл grass.xml такого змісту:
```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android" android:dither="true" android:shape="rectangle" >
<gradient android:angle="90"
android:endColor="#ff003300" android:startColor="#ff009900" />
</shape>
```

Тут заданий зелений прямокутник з градієнтом.

Далі в файл strings.xml в папці res/values додамо наступні рядки:
```
<string name="sun">Солнце</string>
<string name="grass">Трава</string>
<string name="sky">Небо</string>
```

Потім відкриємо розмітку головною активіті activity_main.xml і додамо в неї кілька елементів ImageView. Має вийти наступний вміст файлу:
```
<RelativeLayout xmlns:android= "http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
        android:layout_height="match_parent" tools:context=".MainActivity" > <ImageView
android:id="@+id/sky" android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:contentDescription="@string/sk y" android:src="@drawable/sky" />
<ImageView android:id="@+id/sun"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_centerHorizontal="true"
        android:contentDescription="@string/sun" android:scaleType="fitCenter" android:src="@drawable/sun" />
<ImageView android:id="@+id/grass" android:layout_width="fill_parent"
android:layout_height="150dp" android:layout_alignParentBottom="true" android:contentDescription="@string/grass"
        android:src="@drawable/grass" />
</RelativeLayout>
```

У всіх елементів ImageView в атрибуті android: src прописані створені фігури, які тепер можна бачити на екрані. Далі створимо нову папку res/anim, в якій будуть знаходитися файли анімації. У цій папці створимо новий файл sun_rise.xml наступного змісту:
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:duration="5000"
android:fillAfter="true" android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        android:shareInterpolator="false" >
<scale
android:fromXScale="1.0" android:fromYScale="1.0" android:pivotX="50%" android:pivotY="50%" android:toXScale="1.5"
        android:toYScale="1.5" />
<translate android:fromYDelta="80%p" android:toYDelta="10%p"
/>
<alpha
android:fromAlpha="0.3" android:toAlpha="1.0" />
</set>
```

У блоці set встановлені параметри анімації. Наприклад, параметр android: duration показує, що анімація повинна відбутися протягом 5 секунд. Параметр fillAfter управляє станом анімації - вона не повинна стрибати в початок. Параметр android: interpolator використовує системну константу для невеликого прискорення від початку до середини анімації і гальмування від середини до кінця анімації.

Усередині блоку set встановлюються спеціальні блоки, що відповідають за характер анімації: зміна розмірів, позиції і прозорості. Наприклад, фігура сонця буде збільшуватися від свого початкового розміру в півтора рази рівномірно від своєї середини (scale). Елемент translate рухає сонце по екрану вертикально вгору. Ми відштовхуємося щодо батьківського елементу, використовуючи суфікс "p".

Сонце починає рух в позиції 80% від батьківського елемента по осі Y і закінчує рух в позиції 10%. При русі також змінюється прозорість сонця від повної прозорості до повної непрозорості (alpha). Далі в файлі MainActivity.java записуємо програмний код програми:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity{
    @Override
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate (savedInstanceState);
        setContentView (R.layout.activity_main);
        // Отримуємо посилання на зображення сонця
        ImageView sunImageView = (ImageView) findViewById (R.id.sun);
        // Анімація для сходу сонця Animation sunRiseAnimation =
        AnimationUtils.loadAnimation (this, R.anim.sun_rise);
        // Підключаємо анімацію до фотографії View sunImageView.startAnimation (sunRiseAnimation)
    }
}
```

1.0.3.2 Порядок виконання роботи

1. Вивчити методичні вказівки до виконання лабораторної роботи №3.
2. Запустити на виконання Android Studio.
3. Якщо буде використовуватися зовнішній емулятор для запуску Android-додатків (наприклад, BlueStacks або Genymotion), завантажити його.
4. Створити в середовищі Android Studio проект, розроблений на практичному занятті №4.
5. Створити в відповідних директоріях проекту файли grass.xml, sky.xml, sun.xml.
6. Додати в проект графічні зображення нерухомих і рухомих зображень.
7. Створити нову папку res/anim і додати в неї файл sun_rise.xml.
8. У вікні java-коду проекту додати рядки для отримання анімованого зображення.
9. Запустити створене додаток в емуляторі Android і спостерігати за появою цього додатка і результатів його роботи у вікні додатків емулятора.
10. Створити новий додаток із застосуванням анімованого зображення.

1.0.4 Постановка задачі лабораторної роботи №4

Тема: Створення та використання Меню

Мета роботи: Знайомство з контекстним меню в Android, створення меню за допомогою ресурсів

1.0.4.1 Методичні вказівки

1.0.4.1.1 Меню в Android Використання меню в додатках дозволяє зберегти цінне екранний простір, яке в іншому випадку було б зайнято відносно рідко використовуваними елементами призначеного для користувача інтерфейсу.

Кожна Активність може мати меню, що реалізують специфічні тільки для неї функції. Можна використовувати також контекстні меню, індивідуальні для кожного Уявлення на екрані.

1.0.4.1.2 Основи використання меню В Android реалізована підтримка триступеневої системи меню, оптимізовану, в першу чергу, для невеликих екранів:

- Основне меню виникає внизу на екрані при натисканні на кнопку «menu» пристрою. Воно може відображати текст і іконки для обмеженого (за замовчуванням, не більше шести) числа пунктів. Для цього меню рекомендується використовувати іконки з колірною гамою в вигляді відтінків сірого з елементами рельєфності. Це меню не може містити радиокнопки і чекбокси. Якщо число пунктів такого меню перевищує максимально допустиме значення, в меню автоматично з'являється пункт з написом «ще» («more»). При натисканні на нього відобразиться Розширене меню.
- Розширене меню відображає прокручуваний список, елементами якого є пункти, які не ввійшли до основного меню. У цьому списку не можуть відображатися іконки, але є можливість відображення радіокнопок і чекбоксів. Оскільки не існує способу відобразити розширене меню замість основного, про зміну стану якихось компонентів програми або системи рекомендується повідомляти користувача за допомогою зміни іконок або тексту пунктів меню.
- Дочірнє меню (меню третього рівня) може бути викликано з основного або розширеного меню і відображається у спливаючому вікні. Вкладеність не підтримується, і спроба викликати з дочірнього ще одне меню призведе до викиду винятку. створення меню

При зверненні до меню викликається метод onCreateOptionsMenu Активності і для появи меню на екрані його потрібно перевизначити. Даний метод отримує в якості параметра об'єкт класу Menu, який в подальшому використовується для маніпуляцій з пунктами меню.

Для додавання нових пунктів в меню використовується метод add об'єкта Menu з наступними параметрами:

- Група: для об'єднання пунктів меню для групової обробки
- Ідентифікатор: унікальний ідентифікатор пункту меню. Цей ідентифікатор передається обробнику натиснення на пункт меню - методу onOptionsItemSelected.
- Порядок: значення, яке вказує порядок, в якому пункти меню будуть виводитися.
- Текст: напис на даному пункті меню.

Після успішного створення меню метод onCreateOptionsMenu повинен повернути значення true.

Приклад показує створення меню з трьох пунктів з використанням строкових ресурсів:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, Menu.FIRST, Menu.NONE, R.string.menu_item1);
    menu.add(0, Menu.FIRST+1, Menu.NONE, R.string.menu_item2);
    menu.add(0, Menu.FIRST+2, Menu.NONE, R.string.menu_item3);
    return true;
}
```

1.0.4.1.3 Створення контекстних меню Найбільш часто використовуваним способом створення контекстного меню в Android є перевизначення методу onCreateContextMenu Активності:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.setHeaderTitle("Конекстное меню");
    menu.add(0, Menu.FIRST, Menu.NONE, "Пункт 1");
    menu.add(0, Menu.FIRST+1, Menu.NONE, "Пункт 2");
    menu.add(0, Menu.FIRST+2, Menu.NONE, "Пункт 3");
}
```

Реєстрація обробника контекстного меню для потрібних уявлень здійснюється за допомогою методу Активності registerForContextMenu:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    tv = (TextView) findViewById(R.id.text_view);
    registerForContextMenu(tv);
}
```

Приклад:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case DELETE_ID:
        AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
        db.deleteItem(info.id);
        populate();
        return true;
    }
    return super.onContextItemSelected(item);
}
```

1.0.4.2 Порядок виконання роботи

Модифікуйте проект MetroPicker наступним чином:

1. Додайте головне меню в Активність, яка буде показувати список станцій метро. В меню повинен бути один пункт: «повернутися». Меню створіть динамічно в коді, без використання строкових ресурсів.
2. Динамічно створіть контекстне меню для Уявлення TextView, що відображає обрану станцію метро головною Активності. Вибір пункту меню повинен скидати обрану станцію.
3. Для головної Активності створіть основне меню з двох пунктів: «скинути» і «вийти». Реалізуйте потрібні функції при виборі цих пунктів.
4. Повторіть реалізацію п.п. 1, 2 і 3 за допомогою ресурсів, що описують меню.

1.0.5 Постановка задачі лабораторної роботи №5

Тема: Робота з SQLite без класу-адаптера

Мета роботи: Освоїти взаємодію з СКБД SQLite без застосування спеціальних класів-адаптерів в Android

1.0.5.1 Методичні вказівки

Механізм роботи з базами даних в Android дозволяє зберігати і обробляти структуровану інформацію. Будь-який додаток може створювати свої власні бази даних, над якими воно буде мати повний контроль.

В Android використовується бібліотека SQLite, що представляє із себе реляционную СКБД, що володіє наступними характерними особливостями: вільно поширювана (open source), що підтримує стандартна мова запитів і транзакції, легка, однорівнева (вбудована), відмовостійка.

1.0.5.1.1 Робота з СКБД без адаптера

При небажанні використовувати клас SQLiteOpenHelper (і взагалі адаптер БД) можна скористатися методом openOrCreateDatabase контексту програми:

```
private static final String DATABASE_NAME = "myDatabase.db";
private static final String DATABASE_TABLE = "mainTable";
private static final String DATABASE_CREATE = "create table "
    + DATABASE_TABLE + " ( _id integer primary key autoincrement,"
    + "column_one text not null);";
SQLiteDatabase myDatabase;
private void createDatabase() {
    myDatabase = openOrCreateDatabase(DATABASE_NAME, MODE_PRIVATE, null);
    myDatabase.execSQL(DATABASE_CREATE);
}
```

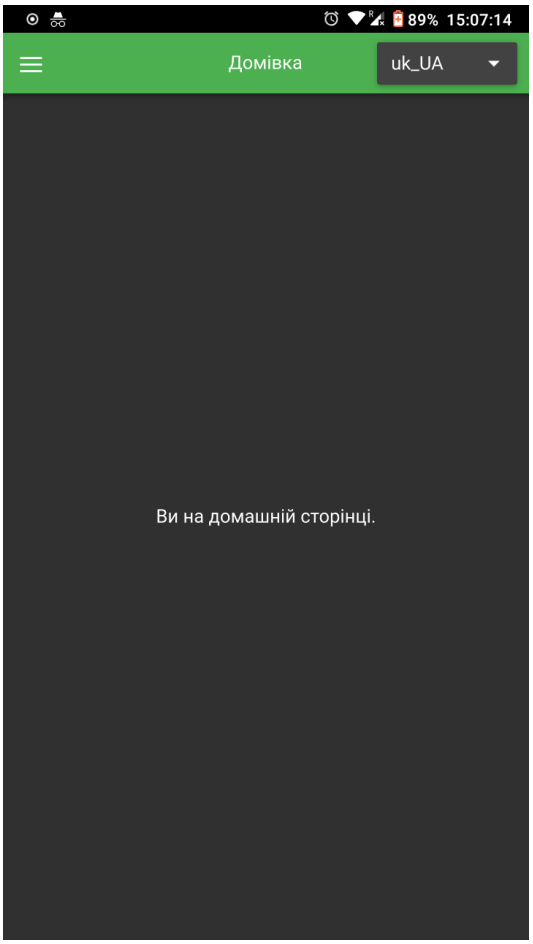
В цьому випадку можна працювати з базою даних за допомогою, наприклад, методу execSQL примірника БД, як показано на прикладі вище.

1.0.5.1.2 Особливості роботи з БД в Android При роботі з базами даних в Android слід уникати зберігання BLOB'ов з таблицях через різкого падіння ефективності роботи. Як показано в прикладі адаптера БД, для кожної таблиці рекомендується створювати автоікрементное поле _id, яке буде унікальним індексом для рядків. Якщо ж планується делегувати доступ до БД за допомогою контент-провайдерів, таке поле є обов'язковим.

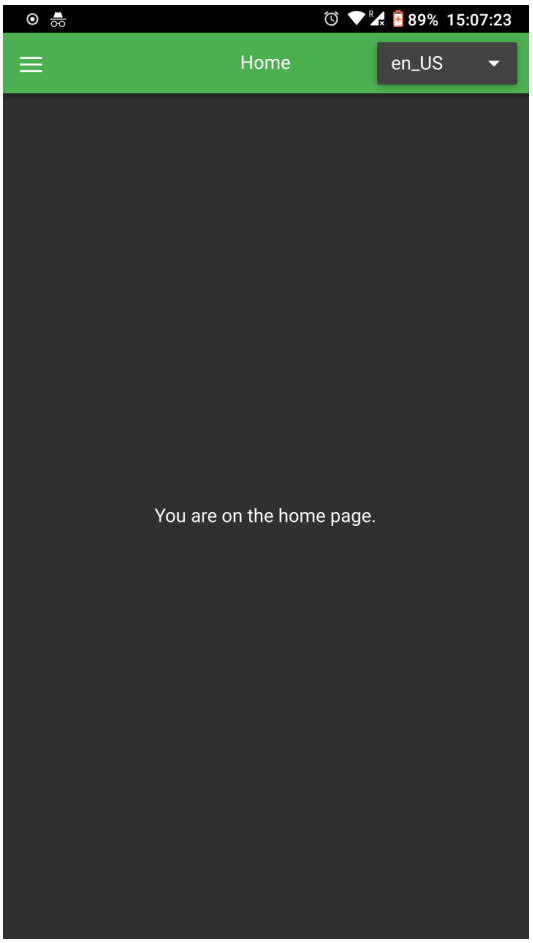
1.0.5.2 Порядок виконання роботи Мета даної лабораторної роботи - освоїти взаємодію з СУБД SQLite без застосування спеціальних класів-адаптерів.

1. Створіть новий проект SQLTest, головна Активність якого буде розширювати клас ListActivity.
2. У методі onCreate головною Активності зробіть відкриття або створення БД, використовуючи в якості основи інформацію з пункту "Робота з СУБД без адаптера" і методу onUpgrade допоміжного класу з попереднього пункту. Після створення БД створіть таблицю з будь-яким полем (плюс індекс), в яку додайте 3-5 унікальних записів зі строковим полем. Індекс повинен інкрементуватися автоматично.
3. У цьому ж методі зробіть запит до таблиці, який одержує всі рядки і за допомогою наявного курсора запишіть дані в масив рядків.
4. За допомогою додаткової розмітки і адаптера ArrayAdapter покажіть отримані з СУБД дані на екрані Активності.
5. Додайте обробник кліка на елемент списку, щоб при ньому запитували з БД індекс елемента з цим рядком і відображався за допомогою Toast.
6. Додайте контекстне меню до елементів списку, що містить пункт "видалити" і реалізуйте видалення. Після видалення повинні проводиться дії з п.п. 3 і 4 для зміни складу відображуваних даних.

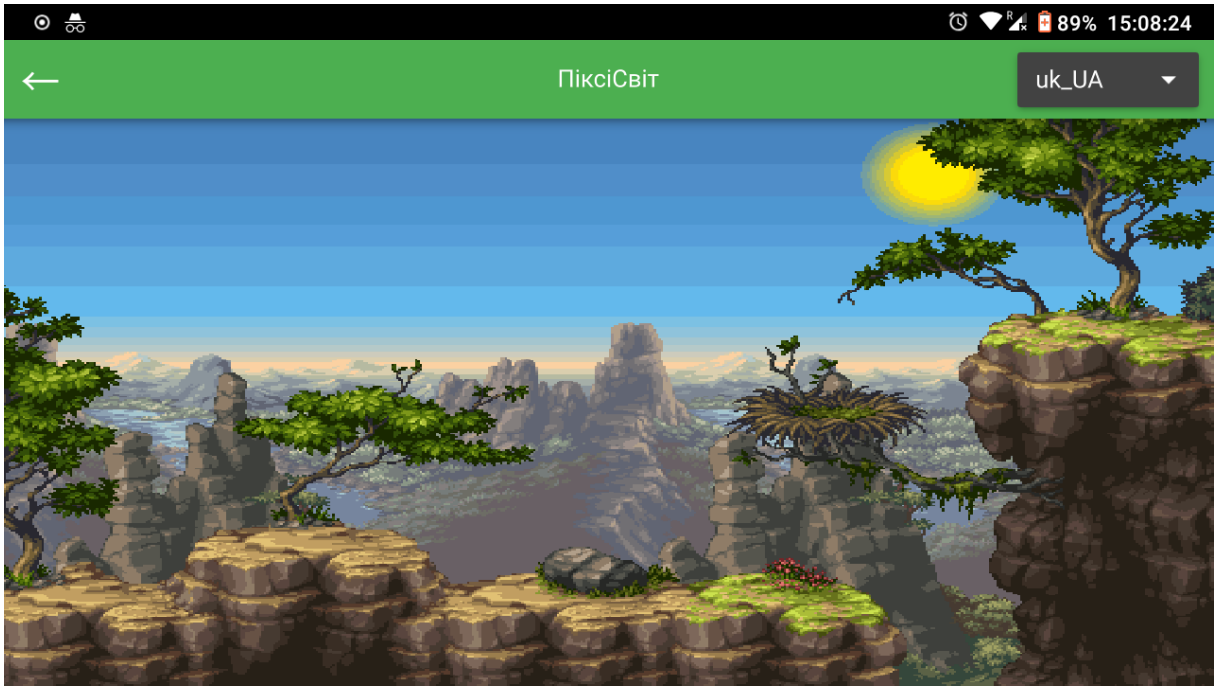
1.0.6 Приклад роботи програми



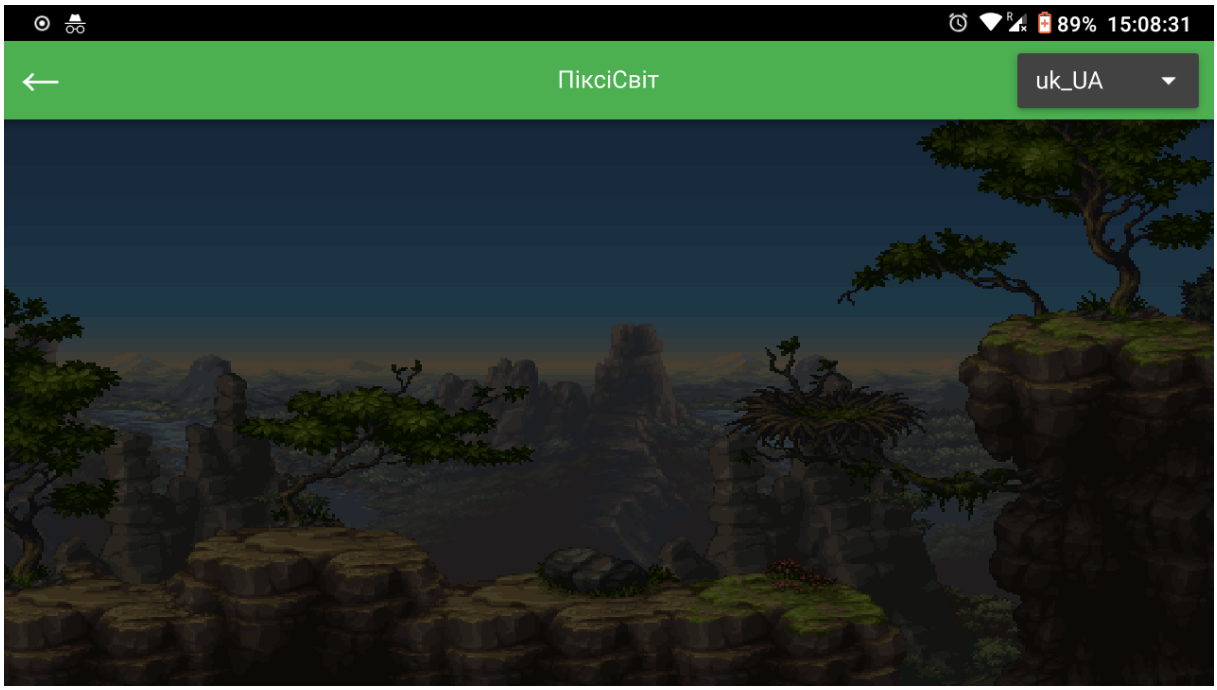
Домашня сторінка програми на українській мові.



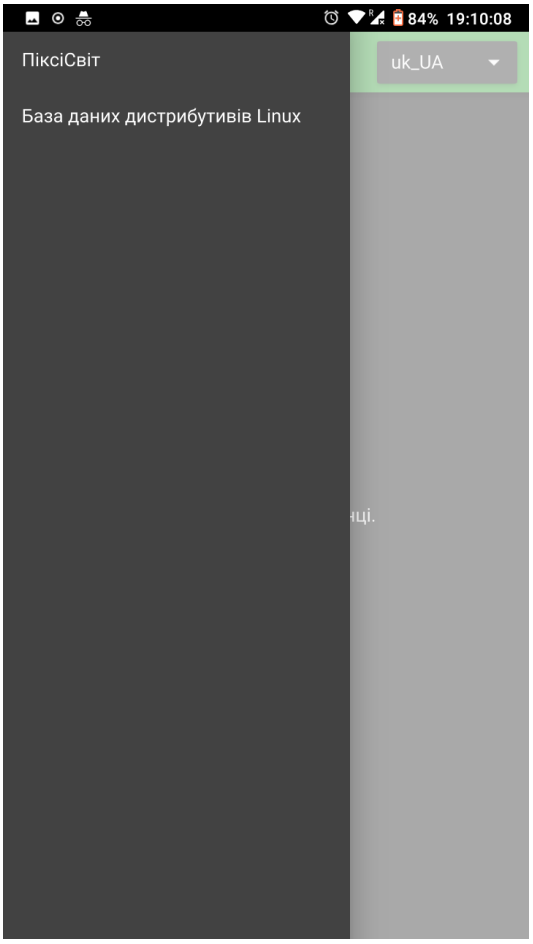
Домашня сторінка програми на англійскій мові.



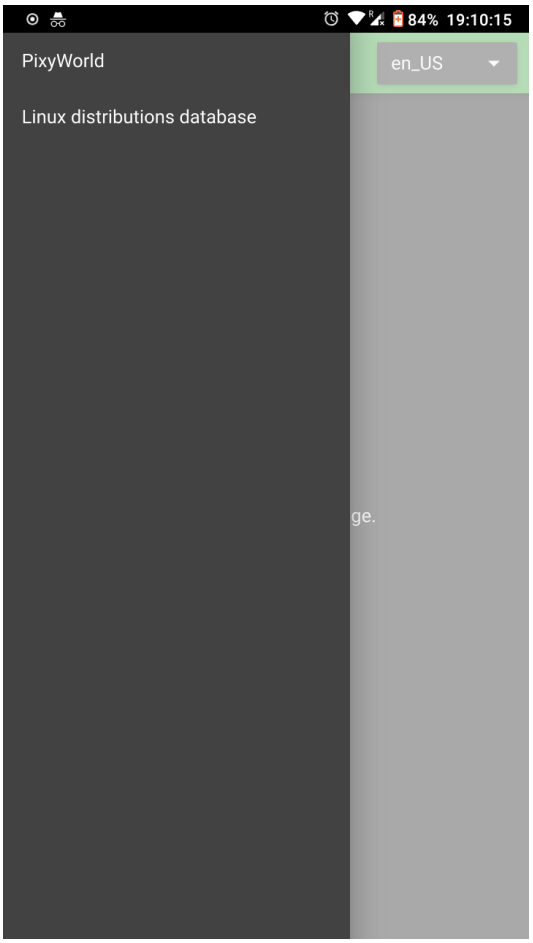
Анімований экран з картинками, у данному випадку день.



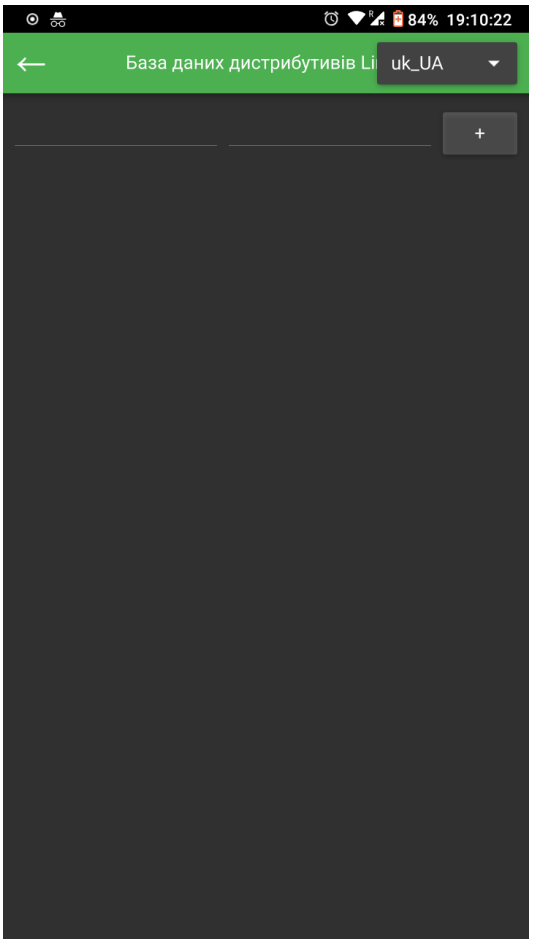
Анімований екран з картинками, у данному випадку ніч.



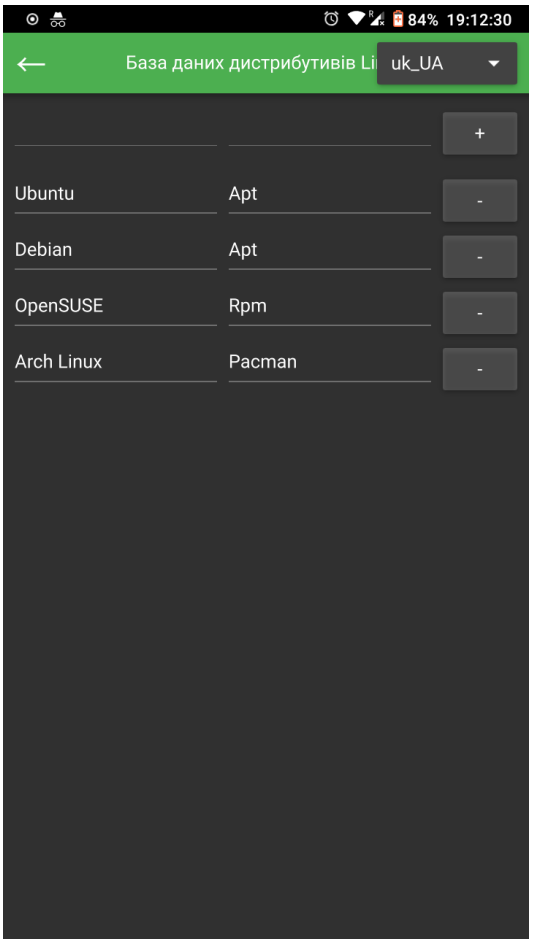
Головне меню на українській мові.



Головне меню на англійскій мові.



Пуста база даних дастрибутивів Linux.



Заповнена база даних дастрибутивів Linux.

1.0.7 Висновки

1. Було застосовано фрамеворк QT та програмний застосунок для розробнки QtCreator для розробки програми під мобільні системи.
2. Створено інтерфейс за допомогою мови QML та теми Material за допомогою графічного редактору QtCreator у дезайнері.
3. У дезайнері використаний таймлайн для створення 2D анімації.
4. Створено меню програми у якому можна перейти на іншу сторінку програми, також реалізовано вибір мови при роботі програми.
5. Використано технологію SQLite як базу даних та розроблено інтерфейс для керування базою даних.

2 Ієрархічний покажчик класів

2.1 Ієрархія класів

Список успадкувань впорядковано наближено до алфавіту

3 Алфавітний покажчик класів

3.1 Класи

Класи, структури, об'єднання та інтерфейси з коротким описом.

4 Покажчик файлв

4.1 Файли

Повний список файлів.

5 Класи

5.1 Клас Linux

Клас який позначає деякий дистрибутив Linux.

#include <linux.h>

Загальнодоступні елементи

- Linux (const QString &name, const QString &family)
- QString name () const

  Забирає назву дистрибутиву.
- void setName (const QString &name)

  Задає нову назву дистрибутиву.
- QString family () const
- void setFamily (const QString &family)

Приватні дані

- QString m_name

  Сигнал який говорить що назва дистрибутиву була змінена.
- QString m_family

  Сімейство дистрибутиву.

5.1.1 Детальний опис

Клас який позначає деякий дистрибутив Linux.

Див. визначення в файлі linux.h, рядок 9

5.1.2 Конструктор(и)

5.1.2.1 Linux() Linux::Linux (const QString & name, const QString & family)

Див. визначення в файлі linux.cpp, рядок 3

```
00004     : m_name(name), m_family(family)
00005 {
00006
00007 }
```

5.1.3 Опис методів компонент

5.1.3.1 family() QString Linux::family ( ) const

Див. визначення в файлі linux.cpp, рядок 20

```
00021 {
00022     return m_family;
00023 }
```

5.1.3.2 name() QString Linux::name ( ) const

Забирає назву дистрибутиву.

Повертає

Назва дистрибутиву.

Див. визначення в файлі linux.cpp, рядок 9

```
00010 {
00011     return m_name;
00012 }
```

5.1.3.3 setFamily() void Linux::setFamily (const QString & family)

Див. визначення в файлі linux.cpp, рядок 25

```
00026 {
00027     m_family = family;
00028     //emit familyChanged();
00029 }
```

5.1.3.4 setName() void Linux::setName (const QString & name)

Задає нову назву дистрибутиву.

Аргументи

name	Нова назва дистрибутиву.

Див. визначення в файлі linux.cpp, рядок 14

```
00015 {
00016     m_name = name;
00017     //emit nameChanged();
00018 }
```

5.1.4 Компонентні дані

5.1.4.1 m_family QString Linux::m_family [private]

Сімейство дистрибутиву.

Див. визначення в файлі linux.h, рядок 43

5.1.4.2 m_name QString Linux::m_name [private]

Сигнал який говорить що назва дистрибутиву була змінена.

Ім'я дистрибутиву Linux.

Див. визначення в файлі linux.h, рядок 39

Документація цих класів була створена з файлів:

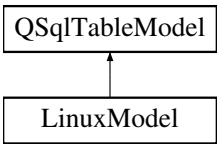
- linux.h
- linux.cpp

5.2 Клас LinuxModel

Модель бази даних дистрибутивів Linux.

#include <linuxmodel.h>

Схема успадкувань для LinuxModel



Загальнодоступні елементи

- LinuxModel (QSqlDatabase &sdb, QObject *parent=nullptr)
  LinuxModel конструктор
- void refresh ()
  Загрузити оновлені данні з бази даних.
- QVariant data (const QModelIndex &index, int role) const
  Перевантажений метод, який дозволяє QT отримати значення даних.
- QHash< int, QByteArray > roleNames () const
  Виводить прив'язку номеру стовбця до її назви.
- Q_INVOKABLE void remove (int i)
  Видаляє елемент по id у базі данних.
- Q_INVOKABLE void update (int i, QString col, QString value)
  Оновити значення у базі даних.
- Q_INVOKABLE void append (QJsonObject obj)
  Додає новий об'єкт до бази даних. Дані необхідно передавати у вигляді json. Наприклад {"name": "Ubuntu", "family": "apt"}

Приватні дані

- QHash< int, QByteArray > m_roleNames

  Прив'язку номеру стовбця до її назви.
- QSqlDatabase & m_sdb

  Посилання на об'єкт, який підключений до бази даних.

5.2.1 Детальний опис

Модель бази даних дистрибутивів Linux.

Див. визначення в файлі linuxmodel.h, рядок 12

5.2.2 Конструктор(и)

5.2.2.1 LinuxModel() LinuxModel::LinuxModel (QSqlDatabase & sdb, QObject * parent = nullptr) [explicit]

LinuxModel конструктор

Аргументи

sdb	Підключена база даних.
parent	Батьтівскій об'єкт

Див. визначення в файлі linuxmodel.cpp, рядок 16

```
00017     : QSqlTableModel(parent), m_sdb(sdb)
00018 {
00019     setTable("linux");
00020     setEditStrategy(QSqlTableModel::QSqlTableModel::OnFieldChange);
00021     select();
00022     setHeaderData(0, Qt::Horizontal, tr("Name"));
00023     setHeaderData(1, Qt::Horizontal, tr("Family"));
00024
00025     int idx = 0;
00026     while(COLUMN_NAMES[idx]) {
00027         m_roleNames[Qt::UserRole + idx + 1] = COLUMN_NAMES[idx];
00028         idx++;
00029     }
00030     refresh();
00031 }
```

5.2.3 Опис методів компонент

5.2.3.1 append() void LinuxModel::append (QJsonObject obj)

Додає новий об'єкт до бази даних. Дані необхідно передавати у вигляді json. Наприклад {"name": "Ubuntu", "family": "apt"}

Аргументи

obj	дані об'єкту у вигляді json.

Див. визначення в файлі linuxmodel.cpp, рядок 76

```
00077 {
00078     QSqlQuery query(m_sdb);
00079     query.prepare("INSERT INTO linux (name, family) VALUES (:name, :family)");
00080     query.bindValue(0, obj.value("name").toString());
00081     query.bindValue(1, obj.value("family").toString());
00082     bool success = query.exec();
00083     if (!success) {
00084         qDebug() « query.lastError().text();
00085     }
00086     refresh();
00087 }
```

5.2.3.2 data() QVariant LinuxModel::data (const QModelIndex & index, int role) const

Перевантажений метод, який дозволяє QT отримати значення даних.

Аргументи

index	Номер рядка.
role	Номер стовбця.

Повертає

Значення комірки у базі даних.

Див. визначення в файлі linuxmodel.cpp, рядок 33

```
00033                                       {
00034     QVariant value = QSqlQueryModel::data(index, role);
00035     if(role < Qt::UserRole)
00036     {
00037         value = QSqlQueryModel::data(index, role);
00038     }
00039     else
00040     {
00041         int columnIdx = role - Qt::UserRole - 1;
00042         QModelIndex modelIndex = this->index(index.row(), columnIdx);
00043         value = QSqlQueryModel::data(modelIndex, Qt::DisplayRole);
00044     }
00045     return value;
00046 }
```

5.2.3.3 refresh() void LinuxModel::refresh ( )

Загрузити оновлені данні з бази даних.

Див. визначення в файлі linuxmodel.cpp, рядок 89

```
00089                       {
00090     QSqlQuery query(SQL_SELECT, m_sdb);
00091     this->setQuery(query);
00092 }
```

5.2.3.4 remove() void LinuxModel::remove (int i)

Видаляє елемент по id у базі данних.

Аргументи

i	id об'єкту.

Див. визначення в файлі linuxmodel.cpp, рядок 53

```
00054 {
00055     QSqlQuery query(m_sdb);
00056     query.prepare("DELETE FROM linux WHERE id = :id;");
00057     query.bindValue(0, i);
00058     query.exec();
00059     refresh();
00060 }
```

5.2.3.5 roleNames() QHash< int, QByteArray > LinuxModel::roleNames ( ) const

Виводить прив'язку номеру стовбця до її назви.

Повертає

Хеш таблицю.

Див. визначення в файлі linuxmodel.cpp, рядок 48

```
00049 {
00050     return m_roleNames;
00051 }
```

5.2.3.6 update() void LinuxModel::update (int i, QString col, QString value)

Оновити значення у базі даних.

Аргументи

i	id об'єкту.
col	Назва колонки.
value	Нове значення.

Див. визначення в файлі linuxmodel.cpp, рядок 62

```
00063 {
00064     QSqlQuery query(m_sdb);
00065     QString q = "UPDATE linux SET %1 = '%2' WHERE id = :id;";
00066     query.prepare(q.arg(col).arg(value));
00067     query.bindValue(":id", i);
00068     bool success = query.exec();
00069     if (!success) {
00070         qDebug() << query.lastError().text() << value;
```

```
00071         qDebug() « query.lastQuery();
00072     }
00073     refresh();
00074 }
```

5.2.4 Компонентні дані

5.2.4.1 m_roleNames QHash<int, QByteArray> LinuxModel::m_roleNames [private]

Прив'язку номеру стовбця до її назви.

Див. визначення в файлі linuxmodel.h, рядок 62

5.2.4.2 m_sdb QSqlDatabase& LinuxModel::m_sdb [private]

Посилання на об'єкт, який підключений до бази даних.

Див. визначення в файлі linuxmodel.h, рядок 66

Документація цих класів була створена з файлів:

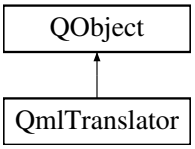
- linuxmodel.h
- linuxmodel.cpp

5.3 Клас QmlTranslator

Дозволяє змінювати мову при роботі програми з QML.

#include <qmltranslator.h>

Схема успадкувань для QmlTranslator



Сигнали

- void languageChanged ()

  Сигнал який говорить що мова була змінена.

Загальнодоступні елементи

- QmlTranslator (QObject *parent=nullptr)
  QmlTranslator конструктор
- Q_INVOKABLE void setTranslation (QString translation)
  Задає мову яка повинна відображатися

Приватні дані

- QTranslator m_translator
  Перекладач, який зберігає поточний переклад програми.

5.3.1 Детальний опис

Дозволяє змінювати мову при роботі програми з QML.

Див. визначення в файлі qmltranslator.h, рядок 11

5.3.2 Конструктор(и)

5.3.2.1 QmlTranslator() QmlTranslator::QmlTranslator (QObject * parent = nullptr) [explicit]

QmlTranslator конструктор

Аргументи

parent	Посилання на батьківскій об'єкт

Див. визначення в файлі qmltranslator.cpp, рядок 4

```
00004                                   : QObject(parent)
00005 {
00006
00007 }
```

5.3.3 Опис методів компонент

5.3.3.1 languageChanged void QmlTranslator::languageChanged ( ) [signal]

Сигнал який говорить що мова була змінена.

5.3.3.2 setTranslation() void QmlTranslator::setTranslation (QString translation)

Задає мову яка повинна відображатися

Аргументи

translation	Код мови, наприклад "en_US".

Див. визначення в файлі qmltranslator.cpp, рядок 9

```
00010 {
00011     qApp->removeTranslator(&m_translator);
00012     if (m_translator.load("DSfMP_" + translation, ":/")) {
00013         qApp->installTranslator(&m_translator);
00014     } else {
00015         qDebug() « "Failed to load translation file: "
00016             « "DSfMP_" + translation;
00017     }
00018     emit languageChanged();
00019 }
```

5.3.4 Компонентні дані

5.3.4.1 m_translator QTranslator QmlTranslator::m_translator [private]

Перекладач, який зберігає поточний переклад програми.

Див. визначення в файлі qmltranslator.h, рядок 38

Документація цих класів була створена з файлів:

- qmltranslator.h
- qmltranslator.cpp

6 Файли

6.1 Файл HomeForm.ui.qml

6.2 HomeForm.ui.qml

```
00001 import QtQuick 2.9
00002 import QtQuick.Controls 2.2
00003
00004 Page {
00005     width: 600
00006     height: 400
00007     anchors.fill: parent
00008
00009     title: qsTr("Home")
00010
00011     Label {
00012         text: qsTr("You are on the home page.")
00013         anchors.centerIn: parent
00014     }
00015 }
```

6.3 Файл linux.cpp

#include "linux.h"

6.4 linux.cpp

```
00001 #include "linux.h"
00002
00003 Linux::Linux(const QString &name, const QString &family)
00004     : m_name(name), m_family(family)
00005 {
00006
00007 }
00008
00009 QString Linux::name() const
00010 {
00011     return m_name;
00012 }
00013
00014 void Linux::setName(const QString &name)
00015 {
00016     m_name = name;
00017     //emit nameChanged();
00018 }
00019
00020 QString Linux::family() const
00021 {
00022     return m_family;
00023 }
00024
00025 void Linux::setFamily(const QString &family)
00026 {
00027     m_family = family;
00028     //emit familyChanged();
00029 }
```

6.5 Файл linux.h

#include <QString>

Класи

- class Linux

  Клас який позначає деякий дистрибутив Linux.

6.6 linux.h

```
00001 #ifndef LINUX_H
00002 #define LINUX_H
00003
00004 #include <QString>
00005
00009 class Linux
00010 {
00011 public:
00012     Linux(const QString &name, const QString &family);
00013
00018     QString name() const;
00023     void setName(const QString &name);
00024
00025     QString family() const;
00026     void setFamily(const QString &family);
00027
00028 //signals:
00032     //void nameChanged();
00033     //void familyChanged();
00034
00035 private:
00039     QString m_name;
00043     QString m_family;
00044 };
00045
00046 #endif // LINUX_H
```

6.7 Файл Linux.qml

6.8 Linux.qml

```
00001 import QtQuick 2.14
00002
00003 LinuxForm {
00004     Connections {
00005         target: addButton
00006         onClicked: {
00007             linuxModel.append({"name": nameEdit.text, "family": familyEdit.text})
00008             nameEdit.text = ""
00009             familyEdit.text = ""
00010         }
00011     }
00012 }
00013
00014 /*##^##
00015 Designer {
00016     D{i:0;autoSize:true;height:480;width:640}
00017 }
00018 ##^##*/
```

6.9 Файл LinuxForm.ui.qml

6.10 LinuxForm.ui.qml

```
00001 import QtQuick 2.14
00002 import QtQuick.Controls 2.2
00003 import Qt.labs.qmlmodels 1.0
00004
00005 Page {
00006     id: page
00007     width: 600
00008     height: 400
00009     anchors.fill: parent
00010     property alias addButton: addButton
00011     property alias familyEdit: familyEdit
00012     property alias nameEdit: nameEdit
00013
00014     title: qsTr("Linux distributions database")
00015
00016     Item {
00017         id: addRow
00018         height: addButton.height
00019         anchors.left: parent.left
00020         anchors.right: parent.right
00021         anchors.top: parent.top
00022         anchors.rightMargin: 10
00023         anchors.leftMargin: 10
00024         anchors.topMargin: 10
00025         Button {
00026             id: addButton
00027             text: "+"
00028             anchors.right: parent.right
00029             anchors.top: parent.top
00030             anchors.topMargin: 0
00031             anchors.rightMargin: 0
00032         }
00033         Item {
00034             id: addFields
00035             anchors.verticalCenter: addButton.verticalCenter
00036             anchors.right: addButton.left
00037             anchors.top: parent.top
00038             anchors.topMargin: 0
00039             anchors.left: parent.left
00040             anchors.rightMargin: 10
00041             anchors.leftMargin: 0
00042
00043             TextField {
00044                 id: nameEdit
00045                 text: ""
00046                 anchors.left: parent.left
00047                 anchors.right: parent.horizontalCenter
00048                 anchors.rightMargin: 5
00049                 anchors.leftMargin: 0
00050                 color: "white"
00051             }
00052             TextField {
00053                 id: familyEdit
00054                 text: ""
```

```
00055                 anchors.left: parent.horizontalCenter
00056                 anchors.right: parent.right
00057                 anchors.top: parent.top
00058                 anchors.topMargin: 0
00059                 anchors.leftMargin: 5
00060                 anchors.rightMargin: 0
00061                 color: "white"
00062             }
00063         }
00064     }
00065
00066     LinuxView {
00067         id: linuxView
00068         anchors.left: parent.left
00069         anchors.right: parent.right
00070         anchors.top: addRow.bottom
00071         anchors.bottom: parent.bottom
00072         anchors.rightMargin: 0
00073         anchors.leftMargin: 0
00074         anchors.bottomMargin: 10
00075         anchors.topMargin: 10
00076         model: linuxModel
00077     }
00078 }
```

6.11 Файл linuxmodel.cpp

```
#include "linuxmodel.h"
#include <QSqlQuery>
#include <QSqlError>
```

Змінні

- const char * COLUMN_NAMES []
- const char * SQL_SELECT

6.11.1 Опис змінних

6.11.1.1 COLUMN_NAMES const char* COLUMN_NAMES[]

Початкові значення
```
= {
  "linux_id",
  "name",
  "family",
  NULL
}
```

Див. визначення в файлі linuxmodel.cpp, рядок 5

6.11.1.2 SQL_SELECT const char* SQL_SELECT

Початкові значення
```
=
  "SELECT linux.id, linux.name, linux.family FROM linux"
```

Див. визначення в файлі linuxmodel.cpp, рядок 12

6.12 linuxmodel.cpp

```
00001 #include "linuxmodel.h"
00002 #include <QSqlQuery>
00003 #include <QSqlError>
00004
00005 const char* COLUMN_NAMES[] = {
00006   "linux_id",
00007   "name",
00008   "family",
00009   NULL
00010 };
00011
00012 const char* SQL_SELECT =
00013   "SELECT linux.id, linux.name, linux.family FROM linux";
00014
00015
00016 LinuxModel::LinuxModel(QSqlDatabase &sdb, QObject *parent)
00017     : QSqlTableModel(parent), m_sdb(sdb)
00018 {
00019     setTable("linux");
00020     setEditStrategy(QSqlTableModel::QSqlTableModel::OnFieldChange);
00021     select();
00022     setHeaderData(0, Qt::Horizontal, tr("Name"));
00023     setHeaderData(1, Qt::Horizontal, tr("Family"));
00024
00025     int idx = 0;
00026     while(COLUMN_NAMES[idx]) {
00027         m_roleNames[Qt::UserRole + idx + 1] = COLUMN_NAMES[idx];
00028         idx++;
00029     }
00030     refresh();
00031 }
00032
00033 QVariant LinuxModel::data(const QModelIndex &index, int role) const {
00034     QVariant value = QSqlQueryModel::data(index, role);
00035     if(role < Qt::UserRole)
00036     {
00037         value = QSqlQueryModel::data(index, role);
00038     }
00039     else
00040     {
00041         int columnIdx = role - Qt::UserRole - 1;
00042         QModelIndex modelIndex = this->index(index.row(), columnIdx);
00043         value = QSqlQueryModel::data(modelIndex, Qt::DisplayRole);
00044     }
00045     return value;
00046 }
00047
00048 QHash<int, QByteArray> LinuxModel::roleNames() const
00049 {
00050     return m_roleNames;
00051 }
00052
00053 void LinuxModel::remove(int i)
00054 {
00055     QSqlQuery query(m_sdb);
00056     query.prepare("DELETE FROM linux WHERE id = :id;");
00057     query.bindValue(0, i);
00058     query.exec();
00059     refresh();
00060 }
00061
00062 void LinuxModel::update(int i, QString col, QString value)
00063 {
00064     QSqlQuery query(m_sdb);
00065     QString q = "UPDATE linux SET %1 = '%2' WHERE id = :id;";
00066     query.prepare(q.arg(col).arg(value));
00067     query.bindValue(":id", i);
00068     bool success = query.exec();
00069     if (!success) {
00070         qDebug() << query.lastError().text() << value;
00071         qDebug() << query.lastQuery();
00072     }
00073     refresh();
00074 }
00075
00076 void LinuxModel::append(QJsonObject obj)
00077 {
00078     QSqlQuery query(m_sdb);
00079     query.prepare("INSERT INTO linux (name, family) VALUES (:name, :family)");
00080     query.bindValue(0, obj.value("name").toString());
00081     query.bindValue(1, obj.value("family").toString());
00082     bool success = query.exec();
00083     if (!success) {
00084         qDebug() << query.lastError().text();
00085     }
```

```
00086     refresh();
00087 }
00088
00089 void LinuxModel::refresh() {
00090     QSqlQuery query(SQL_SELECT, m_sdb);
00091     this->setQuery(query);
00092 }
```

6.13 Файл linuxmodel.h

```
#include <QObject>
#include <QSqlTableModel>
#include <QSqlDatabase>
#include <QJsonObject>
```

Класи

- class LinuxModel

  Модель бази даних дистрибутивів Linux.

6.14 linuxmodel.h

```
00001 #ifndef LINUXMODEL_H
00002 #define LINUXMODEL_H
00003
00004 #include <QObject>
00005 #include <QSqlTableModel>
00006 #include <QSqlDatabase>
00007 #include <QJsonObject>
00008
00012 class LinuxModel : public QSqlTableModel
00013 {
00014     Q_OBJECT
00015 public:
00021     explicit LinuxModel(QSqlDatabase &sdb, QObject *parent = nullptr);
00025     void refresh();
00032     QVariant data(const QModelIndex &index, int role) const;
00037     QHash<int, QByteArray> roleNames() const;
00042     Q_INVOKABLE void remove(int i);
00049     Q_INVOKABLE void update(int i, QString col, QString value);
00056     Q_INVOKABLE void append(QJsonObject obj);
00057
00058 private:
00062     QHash<int, QByteArray> m_roleNames;
00066     QSqlDatabase &m_sdb;
00067 };
00068
00069 #endif // LINUXMODEL_H
```

6.15 Файл LinuxView.qml

6.16 LinuxView.qml

```
00001 import QtQuick 2.14
00002 import QtQuick.Controls 2.2
00003 import Qt.labs.qmlmodels 1.0
00004
00005 ListView {
00006     id: view
00007     height: 300
00008     delegate: Item {
00009         id: row
00010         height: button.height
00011         x: 10
00012         width: view.width - 20
00013
00014         Button {
00015             id: button
```

```
00016             text: "-"
00017             onClicked: view.model.remove(linux_id)
00018             anchors.right: parent.right
00019             anchors.top: parent.top
00020             anchors.topMargin: 0
00021             anchors.rightMargin: 0
00022         }
00023         Item {
00024             anchors.verticalCenter: button.verticalCenter
00025             anchors.right: button.left
00026             anchors.top: parent.top
00027             anchors.topMargin: 0
00028             anchors.left: parent.left
00029             anchors.rightMargin: 10
00030             anchors.leftMargin: 0
00031
00032             TextField {
00033                 id: nameEdit
00034                 text: name
00035                 anchors.left: parent.left
00036                 anchors.right: parent.horizontalCenter
00037                 anchors.rightMargin: 5
00038                 anchors.leftMargin: 0
00039                 color: "white"
00040                 onEditingFinished: linuxModel.update(linux_id, "name", text)
00041             }
00042             TextField {
00043                 id: familyEdit
00044                 text: family
00045                 anchors.left: parent.horizontalCenter
00046                 anchors.right: parent.right
00047                 anchors.top: parent.top
00048                 anchors.topMargin: 0
00049                 anchors.leftMargin: 5
00050                 anchors.rightMargin: 0
00051                 color: "white"
00052                 onEditingFinished: linuxModel.update(linux_id, "family", text)
00053             }
00054         }
00055     }
00056 }
```

6.17 Файл main.cpp

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include <QFile>
#include "qmltranslator.h"
#include "linuxmodel.h"
```

Макровизначення

- #define DATABASE "db_name.sqlite"

Функції

- int main (int argc, char *argv[])

Змінні

- const char * CREATE_DB
- const char * FILL_DB

6.17.1 Опис макровизначень

6.17.1.1 DATABASE #define DATABASE "db_name.sqlite"

Див. визначення в файлі main.cpp, рядок 11

6.17.2 Опис функцій

6.17.2.1 main() int main (int argc, char * argv[])

Див. визначення в файлі main.cpp, рядок 25

```
00026 {
00027 #if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
00028     QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
00029 #endif
00030
00031     QGuiApplication app(argc, argv);
00032     QmlTranslator qmlTranslator;
00033
00034     bool existsDb = QFile::exists(DATABASE);
00035
00036     QSqlDatabase sdb = QSqlDatabase::addDatabase("QSQLITE");
00037     QSqlQuery query(sdb);
00038     sdb.setDatabaseName(DATABASE);
00039     if (!sdb.open()) {
00040         qDebug() << sdb.lastError().text();
00041         qDebug() << "Failed to open database.";
00042         return -1;
00043     }
00044     if (!existsDb) {
00045         {
00046             bool success = query.exec(CREATE_DB);
00047             if (!success) {
00048                 qDebug() << query.lastError().text();
00049                 qDebug() << "Failed to create database.";
00050                 return -1;
00051             }
00052         }
00053         {
00054             bool success = query.exec(FILL_DB);
00055             if (!success) {
00056                 qDebug() << query.lastError().text();
00057                 qDebug() << "Failed to fill database.";
00058                 return -1;
00059             }
00060         }
00061     }
00062
00063     LinuxModel *linuxModel = new LinuxModel(sdb, &app);
00064
00065     QQmlApplicationEngine engine;
00066     const QUrl url(QStringLiteral("qrc:/main.qml"));
00067     engine.rootContext()->setContextProperty("qmlTranslator", &qmlTranslator);
00068     engine.rootContext()->setContextProperty("linuxModel", linuxModel);
00069     QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,
00070                      &app, [url](QObject *obj, const QUrl &objUrl) {
00071         if (!obj && url == objUrl)
00072             QCoreApplication::exit(-1);
00073     }, Qt::QueuedConnection);
00074     QObject::connect(&qmlTranslator, &QmlTranslator::languageChanged,
00075                      &engine, &QQmlApplicationEngine::retranslate, Qt::QueuedConnection);
00076     engine.load(url);
00077
00078     return app.exec();
00079 }
```

6.17.3 Опис змінних

6.17.3.1 CREATE_DB const char* CREATE_DB

Початкові значення
```
= "CREATE TABLE 'linux' ("
"    'id'   INTEGER,"
"    'name' TEXT,"
"    'family'   TEXT,"
"    PRIMARY KEY('id' AUTOINCREMENT)"
");"
```

Див. визначення в файлі main.cpp, рядок 13

6.17.3.2 FILL_DB const char* FILL_DB

Початкові значення
```
= "INSERT INTO linux (name, family) VALUES"
"('Debian', 'apt'),"
"('Ubuntu', 'apt'),"
"('OpenSUSE', 'rpm'),"
"('Arch Linux', 'pacman');"
```

Див. визначення в файлі main.cpp, рядок 19

6.18 main.cpp

```
00001 #include <QGuiApplication>
00002 #include <QQmlApplicationEngine>
00003 #include <QQmlContext>
00004 #include <QSqlDatabase>
00005 #include <QSqlQuery>
00006 #include <QSqlError>
00007 #include <QFile>
00008 #include "qmltranslator.h"
00009 #include "linuxmodel.h"
00010
00011 #define DATABASE "db_name.sqlite"
00012
00013 const char * CREATE_DB = "CREATE TABLE 'linux' ("
00014 "    'id'   INTEGER,"
00015 "    'name' TEXT,"
00016 "    'family'   TEXT,"
00017 "    PRIMARY KEY('id' AUTOINCREMENT)"
00018 ");";
00019 const char * FILL_DB = "INSERT INTO linux (name, family) VALUES"
00020 "('Debian', 'apt'),"
00021 "('Ubuntu', 'apt'),"
00022 "('OpenSUSE', 'rpm'),"
00023 "('Arch Linux', 'pacman');";
00024
00025 int main(int argc, char *argv[])
00026 {
00027 #if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
00028     QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
00029 #endif
00030
00031     QGuiApplication app(argc, argv);
00032     QmlTranslator qmlTranslator;
00033
00034     bool existsDb = QFile::exists(DATABASE);
00035
00036     QSqlDatabase sdb = QSqlDatabase::addDatabase("QSQLITE");
00037     QSqlQuery query(sdb);
00038     sdb.setDatabaseName(DATABASE);
00039     if (!sdb.open()) {
00040         qDebug() << sdb.lastError().text();
00041         qDebug() << "Failed to open database.";
00042         return -1;
```

```
00043     }
00044     if (!existsDb) {
00045         {
00046             bool success = query.exec(CREATE_DB);
00047             if (!success) {
00048                 qDebug() « query.lastError().text();
00049                 qDebug() « "Failed to create database.";
00050                 return -1;
00051             }
00052         }
00053         {
00054             bool success = query.exec(FILL_DB);
00055             if (!success) {
00056                 qDebug() « query.lastError().text();
00057                 qDebug() « "Failed to fill database.";
00058                 return -1;
00059             }
00060         }
00061     }
00062
00063     LinuxModel *linuxModel = new LinuxModel(sdb, &app);
00064
00065     QQmlApplicationEngine engine;
00066     const QUrl url(QStringLiteral("qrc:/main.qml"));
00067     engine.rootContext()->setContextProperty("qmlTranslator", &qmlTranslator);
00068     engine.rootContext()->setContextProperty("linuxModel", linuxModel);
00069     QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,
00070                      &app, [url](QObject *obj, const QUrl &objUrl) {
00071         if (!obj && url == objUrl)
00072             QCoreApplication::exit(-1);
00073     }, Qt::QueuedConnection);
00074     QObject::connect(&qmlTranslator, &QmlTranslator::languageChanged,
00075                      &engine, &QQmlApplicationEngine::retranslate, Qt::QueuedConnection);
00076     engine.load(url);
00077
00078     return app.exec();
00079 }
```

6.19 Файл main.qml

6.20 main.qml

```
00001 import QtQuick 2.9
00002 import QtQuick.Controls 2.2
00003
00004 ApplicationWindow {
00005     id: window
00006     width: 640
00007     height: 480
00008     visible: true
00009     //title: qsTr("Stack")
00010
00011     header: ToolBar {
00012         id: toolbar
00013         contentHeight: toolButton.implicitHeight
00014
00015         ToolButton {
00016             id: toolButton
00017             text: stackView.depth > 1 ? "\u2190" : "\u2261"
00018             font.pixelSize: Qt.application.font.pixelSize * 1.6
00019             onClicked: {
00020                 if (stackView.depth > 1) {
00021                     stackView.pop()
00022                 } else {
00023                     drawer.open()
00024                 }
00025             }
00026         }
00027
00028         Label {
00029             text: stackView.currentItem.title
00030             anchors.centerIn: parent
00031         }
00032
00033         ComboBox {
00034             id: comboBox
00035             anchors.verticalCenter: parent.verticalCenter
00036             anchors.right: parent.right
00037             anchors.rightMargin: 10
00038
00039             model: ["uk_UA", "en_US"]
00040
00041             onCurrentTextChanged: {
```

```
00042                 qmlTranslator.setTranslation(comboBox.currentText)
00043             }
00044         }
00045     }
00046
00047     MouseArea {
00048         anchors.fill: parent
00049         acceptedButtons: Qt.RightButton
00050         onClicked: drawer.open()
00051     }
00052
00053     Drawer {
00054         id: drawer
00055         width: window.width * 0.66
00056         height: window.height
00057
00058         Column {
00059             anchors.fill: parent
00060
00061             ItemDelegate {
00062                 text: qsTr("PixyWorld")
00063                 width: parent.width
00064                 onClicked: {
00065                     stackView.push("PixyWorld.ui.qml")
00066                     drawer.close()
00067                 }
00068             }
00069             ItemDelegate {
00070                 text: qsTr("Linux distributions database")
00071                 width: parent.width
00072                 onClicked: {
00073                     stackView.push("Linux.qml", {"linuxModel": linuxModel})
00074                     drawer.close()
00075                 }
00076             }
00077         }
00078     }
00079
00080     StackView {
00081         id: stackView
00082         initialItem: "HomeForm.ui.qml"
00083         anchors.fill: parent
00084     }
00085 }
00086
00087 /*##^##
00088 Designer {
00089     D{i:0;formeditorZoom:0.5}D{i:4}
00090 }
00091 ##^##*/
```

6.21 Файл mainpage.dox

6.22 Файл PixyWorld.ui.qml

6.23 PixyWorld.ui.qml

```
00001 import QtQuick 2.9
00002 import QtQuick.Controls 2.2
00003 import QtQuick.Timeline 1.0
00004 import QtGraphicalEffects 1.0
00005
00006 Page {
00007     id: page
00008     width: 600
00009     height: 400
00010     anchors.fill: parent
00011
00012     title: qsTr("PixyWorld")
00013
00014     Item {
00015         id: scene
00016         width: 512
00017         height: 338
00018         transform: Scale {
00019             xScale: page.width / scene.width
00020             yScale: page.height / scene.height
00021         }
00022
00023         Image {
```

```
00024           id: skyImage
00025           source: "sky.png"
00026           anchors.fill: parent
00027           smooth: false
00028           visible: false
00029       }
00030
00031       BrightnessContrast {
00032           id: sky
00033           anchors.fill: skyImage
00034           source: skyImage
00035           brightness: -0.7
00036           contrast: 0
00037       }
00038
00039       Image {
00040           id: sun
00041           source: "sun.png"
00042           smooth: false
00043           x: 224
00044           y: 0
00045       }
00046
00047       Image {
00048           id: groundImage
00049           source: "ground.png"
00050           anchors.fill: parent
00051           smooth: false
00052           visible: false
00053       }
00054
00055       BrightnessContrast {
00056           id: ground
00057           anchors.fill: groundImage
00058           source: groundImage
00059           brightness: -0.7
00060           contrast: 0
00061       }
00062   }
00063
00064   Timeline {
00065       id: timeline
00066       animations: [
00067           TimelineAnimation {
00068               id: timelineAnimation
00069               duration: 5000
00070               loops: -1
00071               running: true
00072               to: 1000
00073               from: 0
00074           }
00075       ]
00076       startFrame: 0
00077       enabled: true
00078       endFrame: 1000
00079
00080       KeyframeGroup {
00081           target: sun
00082           property: "x"
00083           Keyframe {
00084               value: 518
00085               frame: 251
00086           }
00087
00088           Keyframe {
00089               value: 224
00090               frame: 499
00091           }
00092
00093           Keyframe {
00094               value: -70
00095               frame: 750
00096           }
00097
00098           Keyframe {
00099               value: 518
00100              frame: 0
00101          }
00102      }
00103
00104      KeyframeGroup {
00105          target: sun
00106          property: "y"
00107          Keyframe {
00108              value: 70
00109              frame: 250
00110          }
```

```
00111
00112             Keyframe {
00113                 easing.bezierCurve: [0.11, 0.989, 0.222, 0.994, 1, 1]
00114                 value: 0
00115                 frame: 500
00116             }
00117
00118             Keyframe {
00119                 easing.bezierCurve: [0.779, 0.000643, 0.889, 0.00527, 1, 1]
00120                 value: 60
00121                 frame: 750
00122             }
00123
00124             Keyframe {
00125                 value: 80
00126                 frame: 0
00127             }
00128         }
00129
00130         KeyframeGroup {
00131             target: ground
00132             property: "brightness"
00133             Keyframe {
00134                 value: -0.7
00135                 frame: 0
00136             }
00137
00138             Keyframe {
00139                 value: -0.7
00140                 frame: 250
00141             }
00142
00143             Keyframe {
00144                 value: 0
00145                 frame: 350
00146             }
00147
00148             Keyframe {
00149                 value: 0
00150                 frame: 650
00151             }
00152
00153             Keyframe {
00154                 value: -0.7
00155                 frame: 750
00156             }
00157         }
00158
00159         KeyframeGroup {
00160             target: sky
00161             property: "brightness"
00162             Keyframe {
00163                 value: -0.7
00164                 frame: 0
00165             }
00166
00167             Keyframe {
00168                 value: -0.7
00169                 frame: 250
00170             }
00171
00172             Keyframe {
00173                 value: 0
00174                 frame: 350
00175             }
00176
00177             Keyframe {
00178                 value: 0
00179                 frame: 650
00180             }
00181
00182             Keyframe {
00183                 value: -0.7
00184                 frame: 750
00185             }
00186         }
00187     }
00188 }
00189
00190 /*##^##
00191 Designer {
00192     D{i:0;formeditorZoom:0.75}D{i:5}D{i:7}D{i:8}
00193 }
00194 ##^##*/
00195
```

6.24 Файл qmltranslator.cpp

#include <QtGui>
#include "qmltranslator.h"

6.25 qmltranslator.cpp

```
00001 #include <QtGui>
00002 #include "qmltranslator.h"
00003
00004 QmlTranslator::QmlTranslator(QObject *parent) : QObject(parent)
00005 {
00006
00007 }
00008
00009 void QmlTranslator::setTranslation(QString translation)
00010 {
00011     qApp->removeTranslator(&m_translator);
00012     if (m_translator.load("DSfMP_" + translation, ":/")) {
00013         qApp->installTranslator(&m_translator);
00014     } else {
00015         qDebug() « "Failed to load translation file: "
00016             « "DSfMP_" + translation;
00017     }
00018     emit languageChanged();
00019 }
```

6.26 Файл qmltranslator.h

#include <QObject>
#include <QTranslator>
#include <QGuiApplication>

Класи

- class QmlTranslator

  Дозволяє змінювати мову при роботі програми з QML.

6.27 qmltranslator.h

```
00001 #ifndef QMLTRANSLATOR_H
00002 #define QMLTRANSLATOR_H
00003
00004 #include <QObject>
00005 #include <QTranslator>
00006 #include <QGuiApplication>
00007
00011 class QmlTranslator : public QObject
00012 {
00013     Q_OBJECT
00014 public:
00019     explicit QmlTranslator(QObject *parent = nullptr);
00020
00021 signals:
00025     void languageChanged();
00026
00027 public:
00032     Q_INVOKABLE void setTranslation(QString translation);
00033
00034 private:
00038     QTranslator m_translator;
00039 };
00040
00041 #endif // QMLTRANSLATOR_H
```