Tian Karaoke List - app showcase

The Tian Karaoke List app returns a track ID for a given entered name. This article describes the main elements of the app's design.

Introduction

You do love singing karaoke? But you know, you will waste time to look for the ID of a song although you do know the song name. This application will help you to find the song information quickly by typing the song name or whatever you remember about that song. This application is used for looking for the Vietnamese and English songs on the Ariang devices.

Maybe my application is not an international use application, just useful for Vietnamese, but this is a Qt Quick Components, and the Nokia 808 can't let me turn my back on it, so I have decided to port its functions to Qt Quick App (previous versions is powered by Qt Widget and also available on Nokia Store).

Design

The layout is quite simple. The root is based on a PageStackWindow element. It use the a shared ToolBar element(contains a ToolBarLayout to manage five ToolButton elements) to navigate between four views.

ToolBar

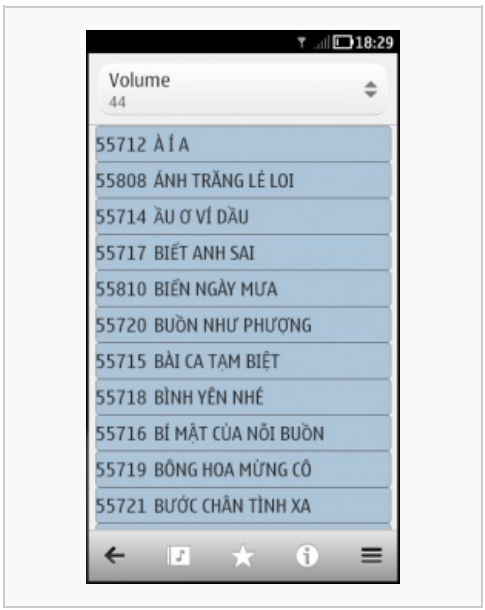
ToolBar with buttons has function: Back/Exit, push the second view, push the third view, push the fourth view, show main menu.

There are four views. The first is searching view with a SearchBox, a ListView with a ScrollerBar, a DetailView, whenever you type in the SearchBox, the ListView will show the result. The DetailView is used to show more information about the song, it's also available in the second and the third view and has the same function, but I will tell you later about its design and its reason since it's a custom component combined from some basic Qt Quick Component. This view is initiated by PageStackWindow, so there is no ToolButton to navigate to it, but the Back ToolButton.

The first view

The second is the view that shows the list of songs by volume. It include a SelectionListItem, a SelectionDialog for choosing volumes, a ListView with a ScrollerBar and a DetailView. When you choose a volume, the ListView will show the songs of that volume.

The second view

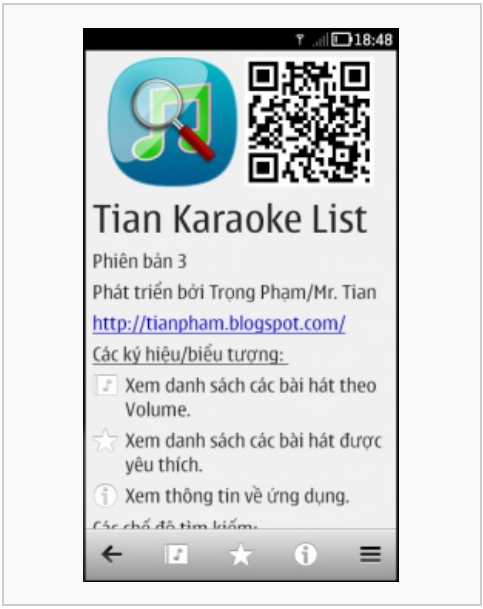


The third view shows the list of favorite songs. It include a ListView with a ScrollerBar, a DetailView.

The third view

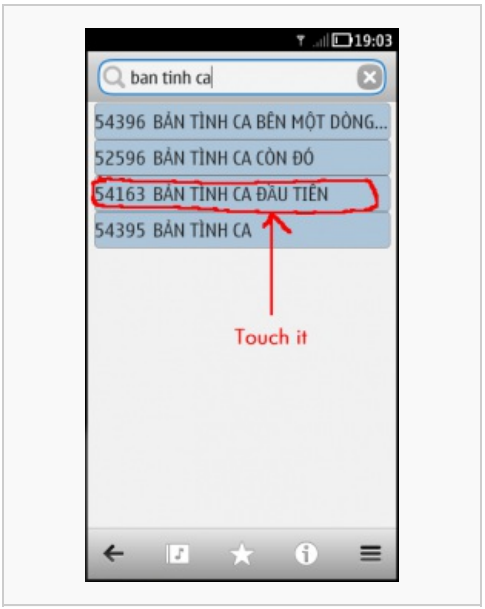
The last view with a Flickable, two Images and some Label - is used to provide the information about the App and some use guides.

The fourth view

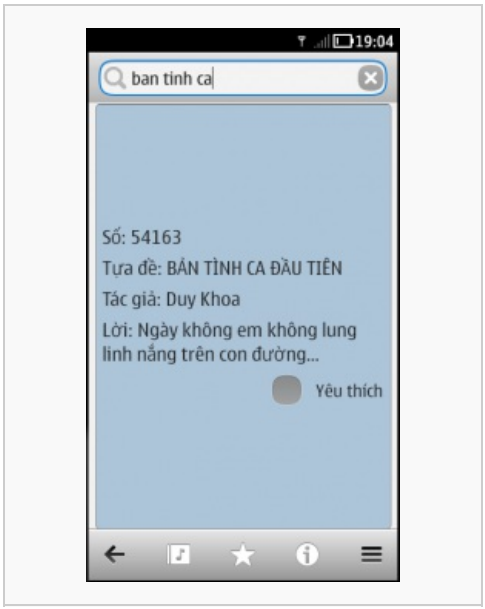


For the ListView delegate, it is quite simple, so on this site, people would be easy to know what is it. When touching the delegate, it will expand the height and hide the Labels, call the suitable DetailView show up to provide more information about the song (author, lyric, is favorite or not?)

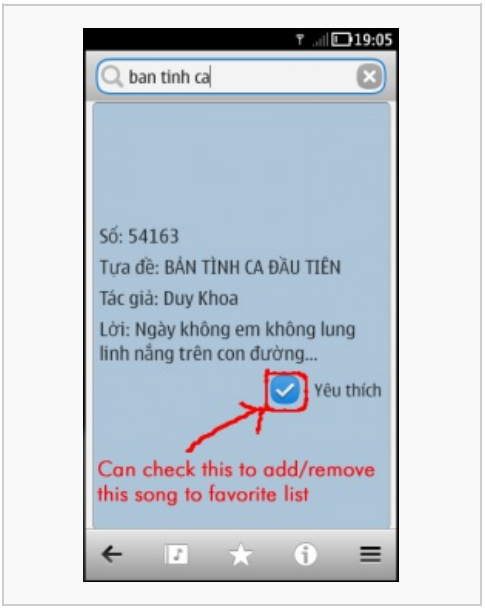
ListView delegate and DetailView



ListView delegate



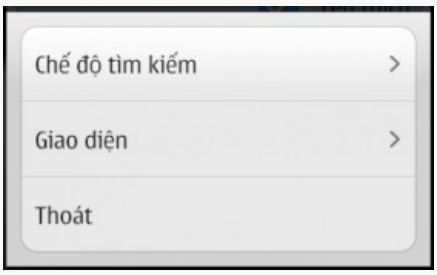
DetailView



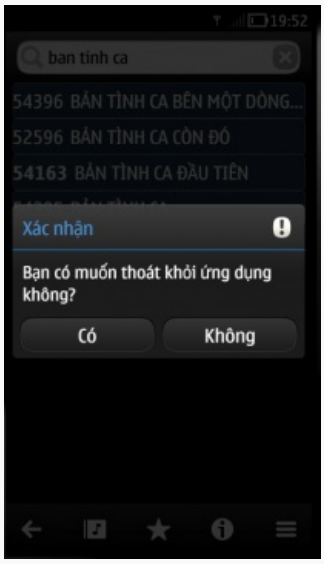
DetailView

The DetailView include five Labels and one CheckBox. In the beginning, I decided to make a Flipable, the ListView at front, and the DetailView at back, when delegate is clicked, it rotates 180 degrees to show the DetailView, but with that animation, I feel like my App is disjointed, not connected, so I change the design, bring the DetailView into the delegate to make my work is easier than separate it from delegate, but the result is lagging when scroll fast with a low specs of my N8, cause the delegate has too much component. And the final result is a custom component separated from delegate but still keep the smoothness in transition.

Main menu



Main menu, from bottom up, is exit (you also can do this by tap the back ToolButton if there is no view remains in pageStack), style (use the property platformInverted to make it), search mode (the way the App search for song)

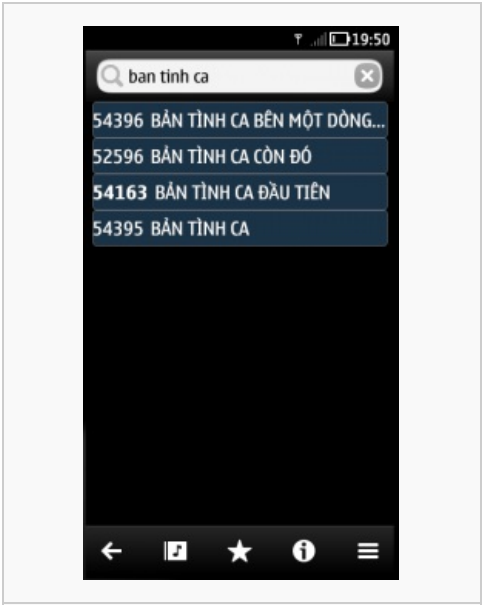


The confirm dialog when you try to quit the App. At the beginning, this is a QueryDialog, but I realize that, it can't show fully a message with two row without ScrollerBar, and when the device's orientation is right up, it display nothing. So I used CommonDialog to let me freely define the content.

Submenu and Style

Submenu - Search mode, there are some others when scroll down.

Submenu - Style, have two option, dark and light, in this image, the light option is chosen.



Dark style was chosen.

The App icon, I combined some parts from three free icon (the blue round shape, the music symbol, the magnifier), change color, added some layer.

Video Demo

The media player is loading...

Summary

It's quite easy to port a Qt Widget App to Qt Quick App using Qt Quick Component, Almost I can get whatever I imagine about the UI my App should be. With Qt Quick, my App bring different/better touch experiences compared to Qt Widget. But I think the Qt Quick Components still have a lot of thing to improve. They still have some limits but bring a good consistent look with Nokia Belle. As my opinion, Qt Quick has a very bright future if it can be brought to more other platforms.

Thank for reading my bad-English article.

Download

Now available on Nokia Store.

Note: This is an entry in the Symbian Qt Quick Components Competition 2012Q1