

KIRIGAMI

Marco Martin

Design phase, HIG work in progress

- Based upon Design guidelines of visual Design Group
- https://community.kde.org/KDE_Visual_Design_Group/KirigamiHIG
- The fastest way to have consistent apps with the HIG is to have pre-made controls
- A lot of those controls in kirigami.git
- Plan: make it a tier-1 framework
- Multiplatform: has to work on Desktop Linux, Plasma Mobile, Android, Ubuntu Phone, Windows...



Primitive controls

- What Kirigami is **NOT** about:
 - Buttons, checkboxes etc:
 - Not in topic, use QQC (soon QQC2)
 - Future migration to QQuickControls2 must be as painless as possible



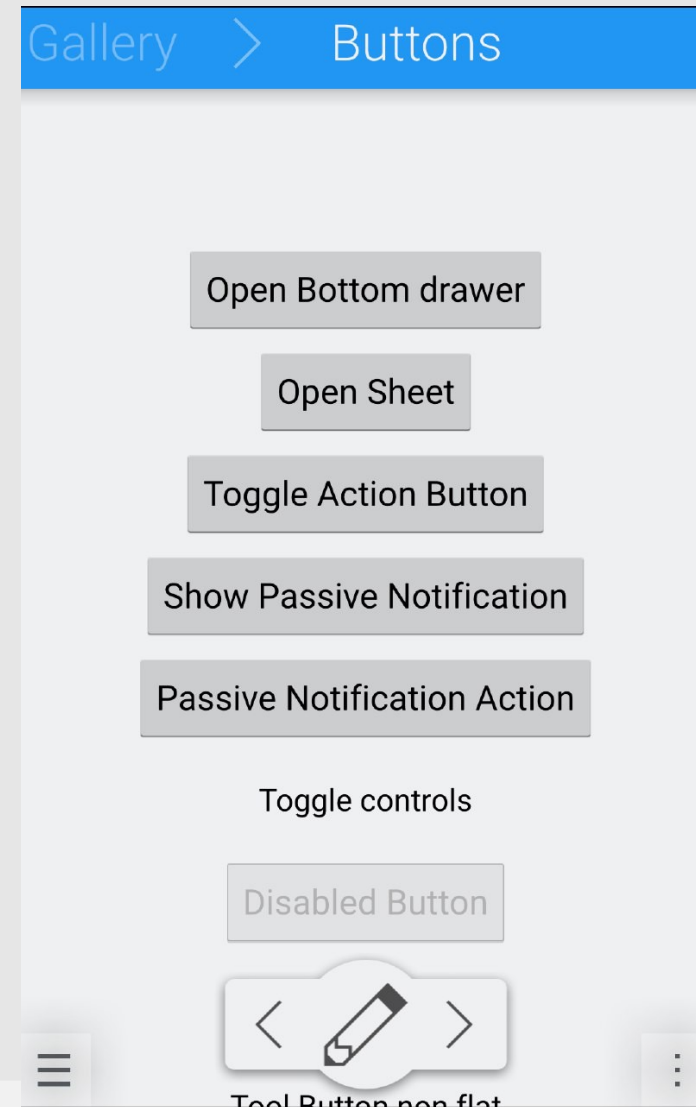
ApplicationWindow

- Base class for applications
- Derives from QQC ApplicationWindow
- Implements some of the central UI elements
- Properties for Drawers
- Passive notifications
- The page navigation looking like a scrollable row (our main way to go back between pages is through gestures, very effective except corner cases)



Primary Action Button

- Not a class that can be directly instantiated
- Triggers actions and acts as an handle for the side drawers
- The Page provides the actions as a model, Kirigami decides how to visually represent them

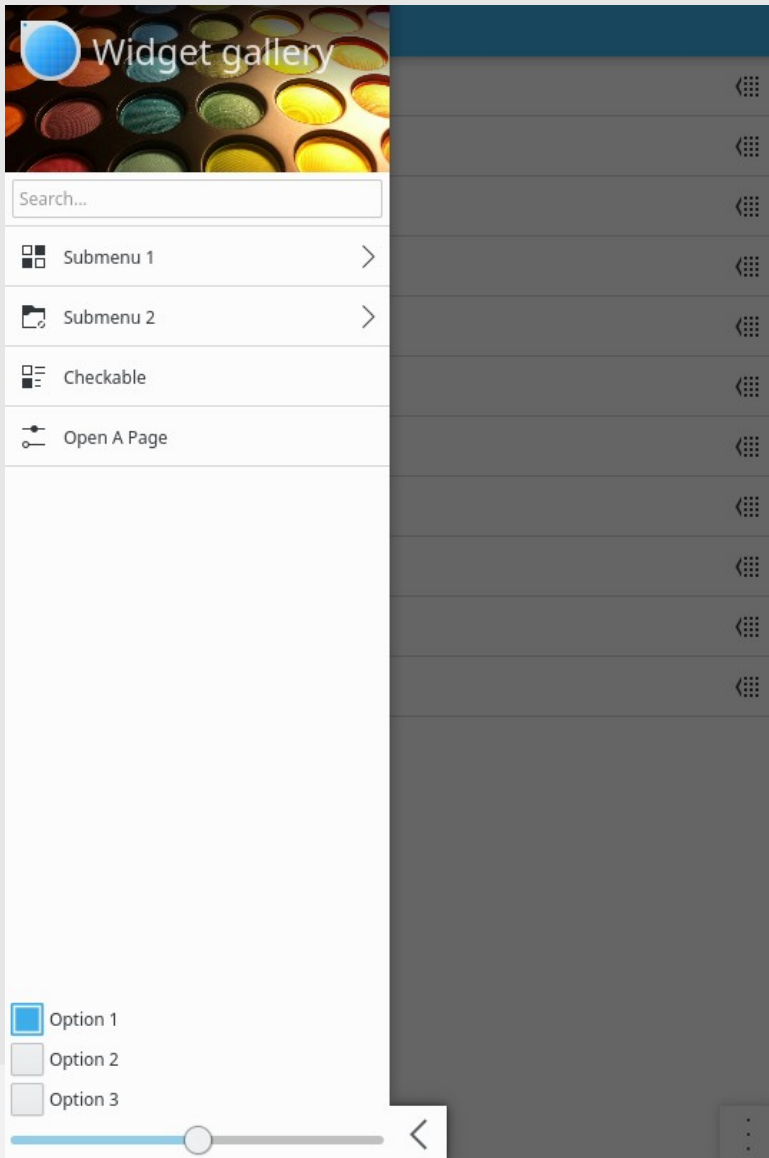


Drawers

- ApplicationWindow supports two drawers that will be overlaid on top of the application contents
- Base class: OverlayDrawer
- It's empty, can be used everywhere
- It can have 4 orientations: left, right, top, bottom
- API compatible with QQuickControls2 Drawer
- Edge slide from left and right, **and** from bottom on platforms that have sides reserved for the system (Ubuntu, Windows 8-10)



GlobalDrawer

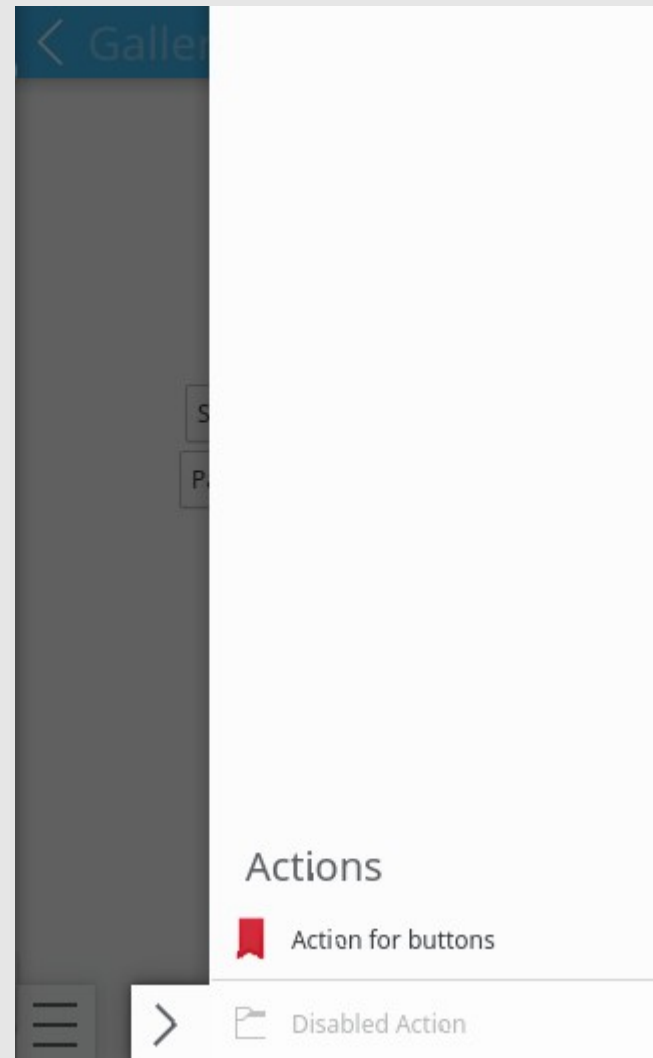


- Edge slide from left (screenedge or Primary Action button)
- Big pretty title
- Navigable menu, like a menubar
- Bottom and top areas to put arbitrary controls

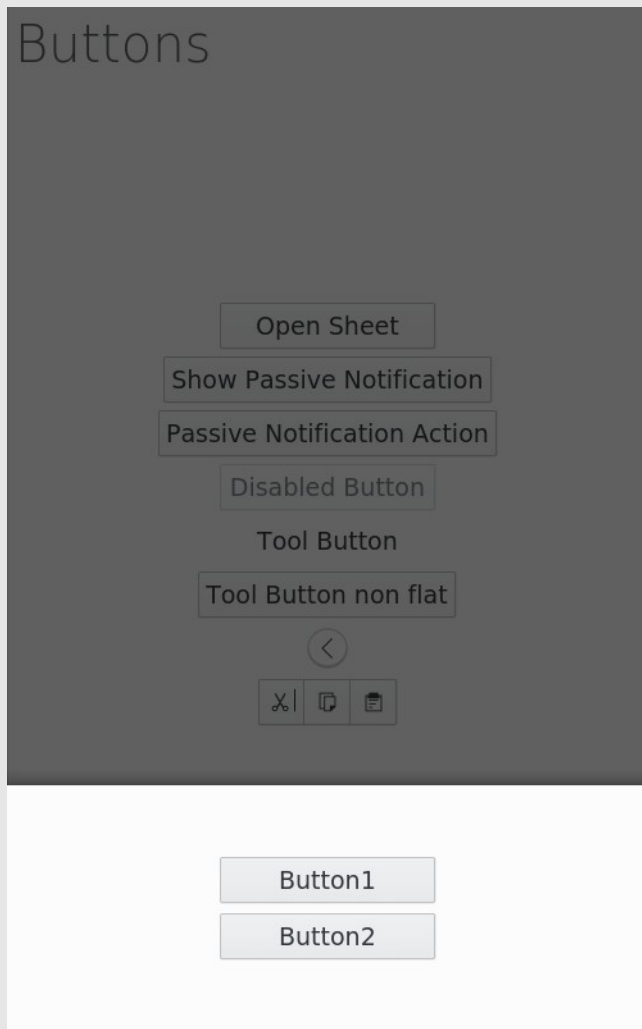


ContextDrawer

- Edge slide from right (screen edge, Action Button or bottom-right gesture)
- Actions that depend from the current app page: context-dependent
- Bottom-aligned to be thumb-friendly



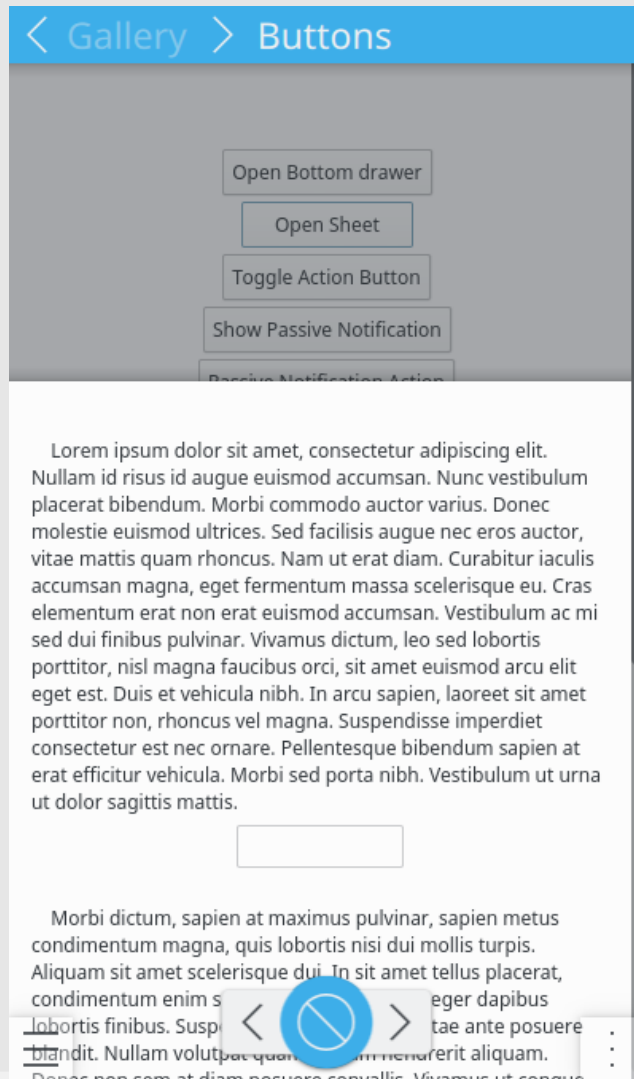
Custom drawers



- The developer can use an arbitrary number of custom drawers
- Drawers from bottom useful to be used as a kind of “dialog”



OverlaySheet

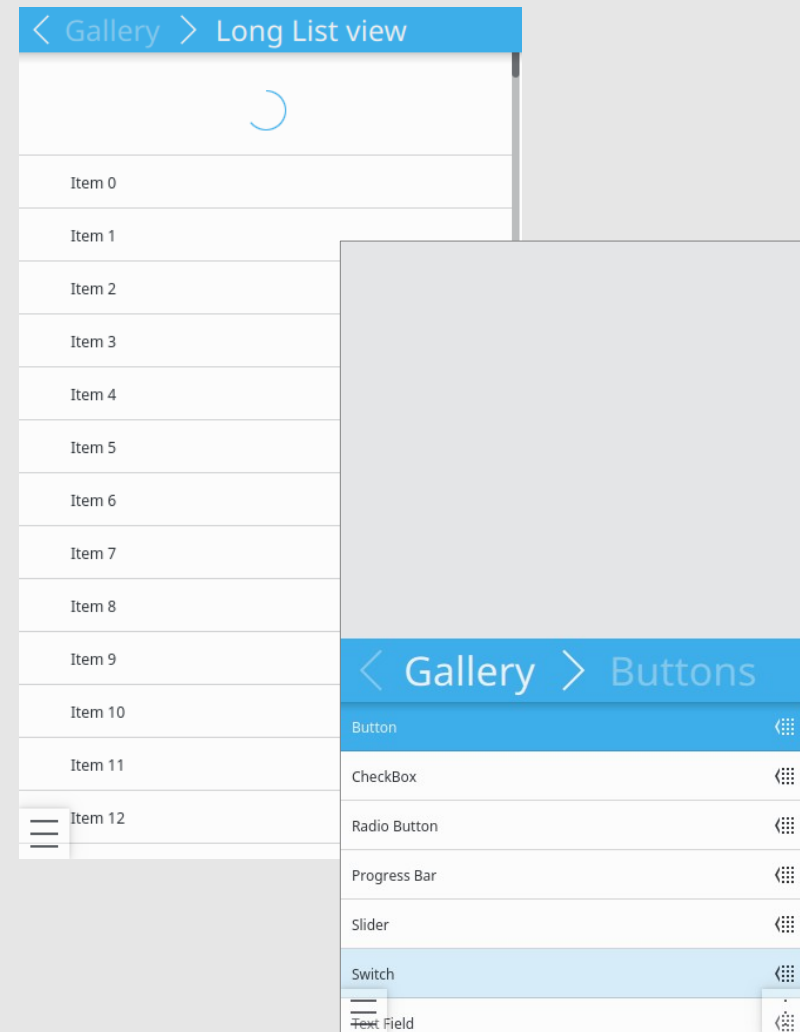


Better than drawers, to have big, scrolling dialog-like pages that can be dismissed by gesture, use OverlaySheet

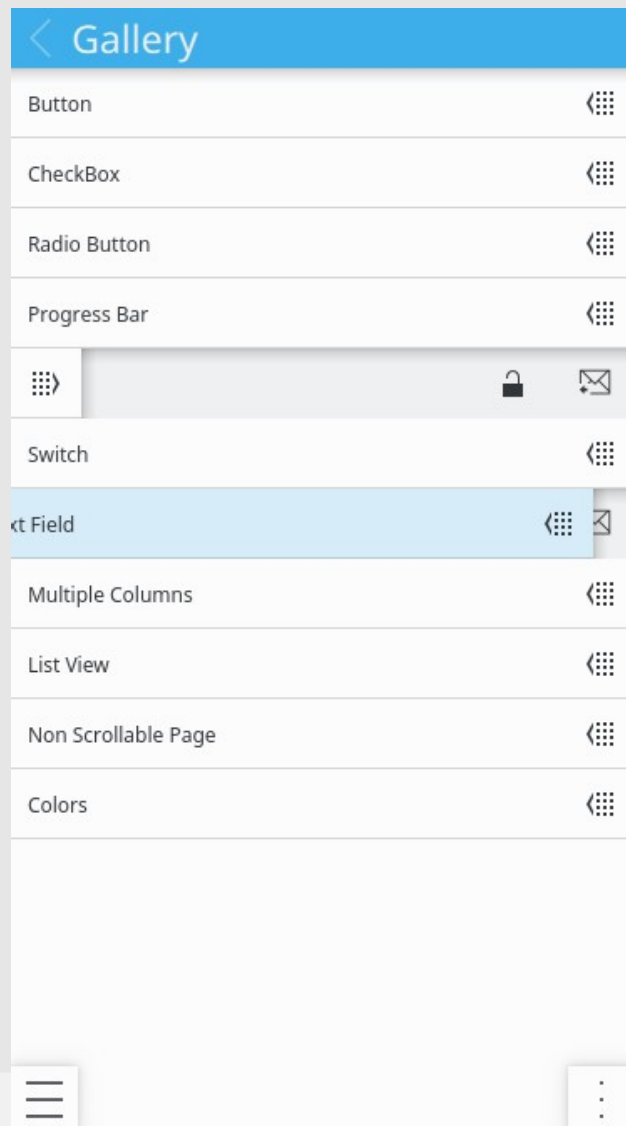


ScrollingPage

- Many Pages on mobile devices are scrollable controls, either lists of items or just complicated layouts that don't fit in the screen
- ScrollingPage manages that without the need of explicitly including Flickables
- Supports the popular “pull down to refresh” gesture present in many mobile apps
- By overpulling, the whole UI gets dragged down making it reachable by thumb



ListItemWithActions



- A list item with a standard look, plus a draggable handle to reveal actions

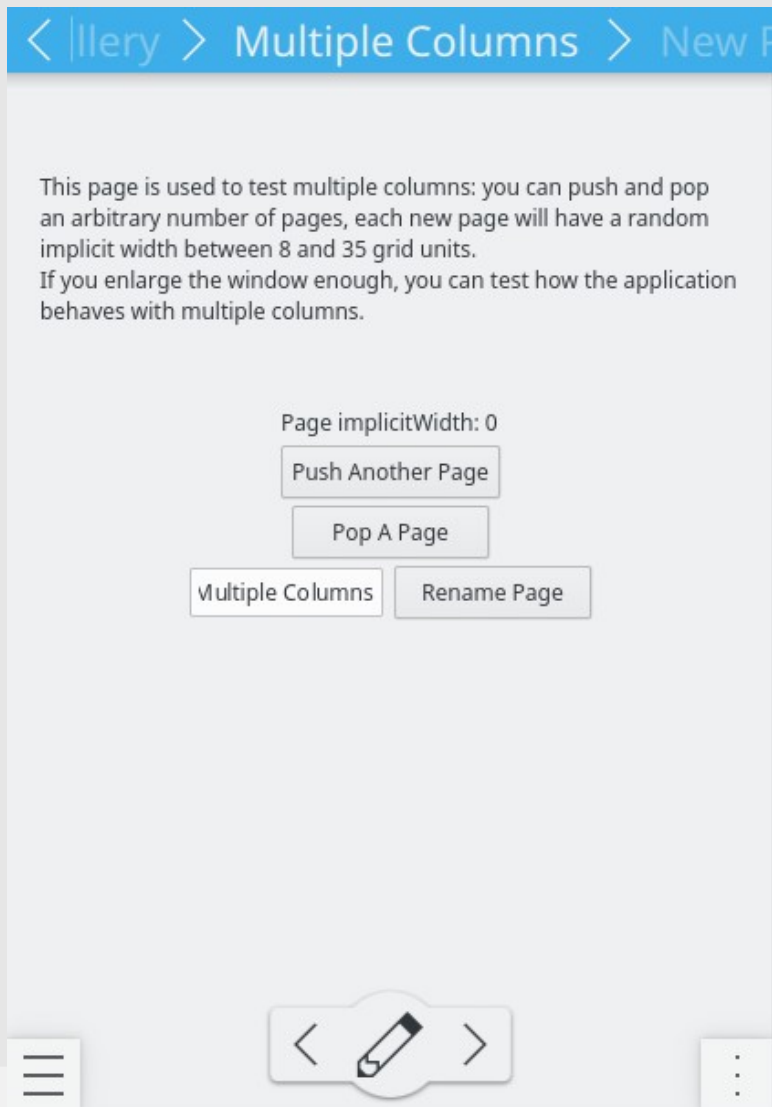


Page

- The application is divided in “pages”
- The class sets a standard behavior
- Exposes “actions” properties
 - main: main Primary Action Button action
 - left: optional smaller button at left of Primary Action Button
 - Right: optional smaller button at right of Primary Action Button
 - contextualActions: list that will go in the contextual drawer



ApplicationHeader



- The top of the application is reserved to a “title” area that acts as a breadcrumb in the currently open page hierarchy
- Content can be customized using `AbstractApplicationHeader` instead



Other classes

- Action: “model” representation of an action
- ListItem/BasicListItem
- Heading
- IconGrid
- Label
- SplitDrawer

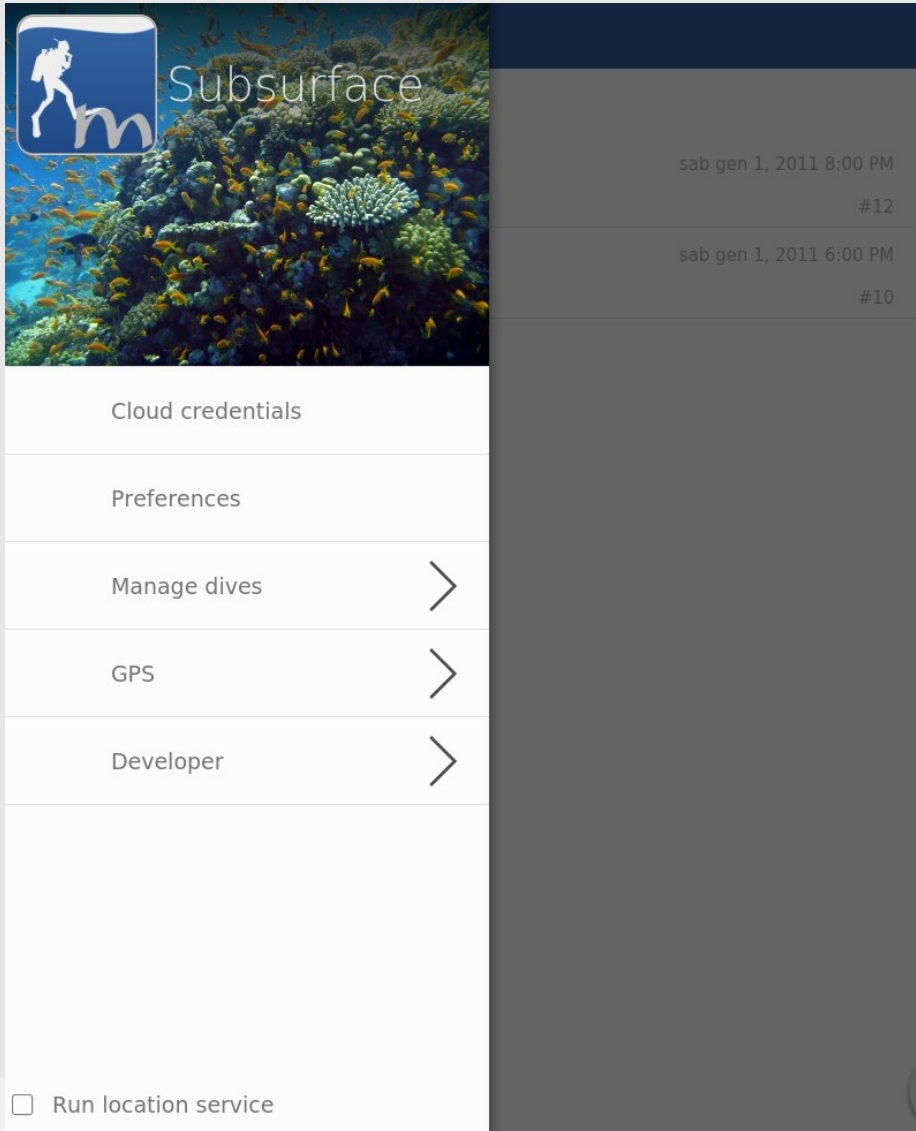


Success Story

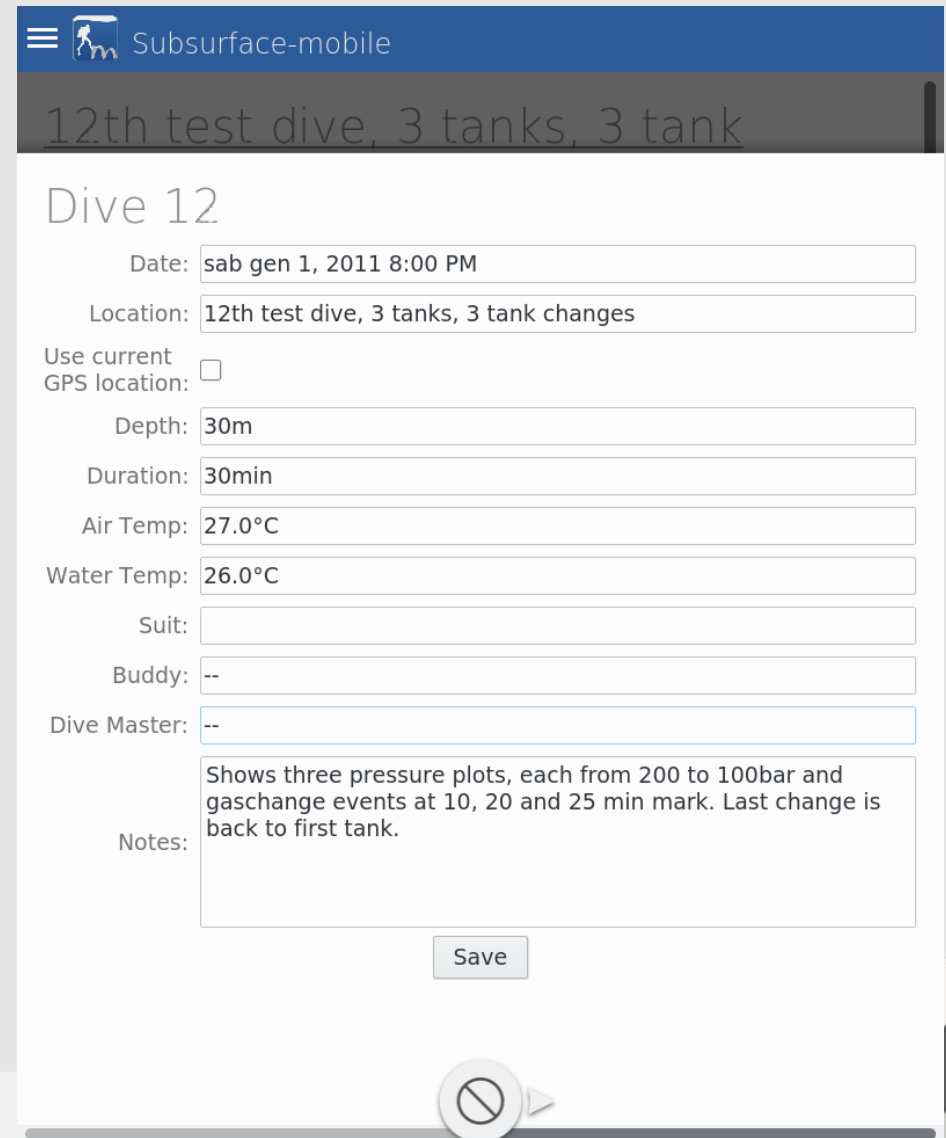
- SubSurface
- Dive log app written by Linus Torvalds, Dirk Hohndel and others
- Desktop app
- Used to be GTK+, migrated to Qt
- Started an Android version in Qt sharing most of code
- Early adopter for Kirigami, good feedback so far



SubSurface

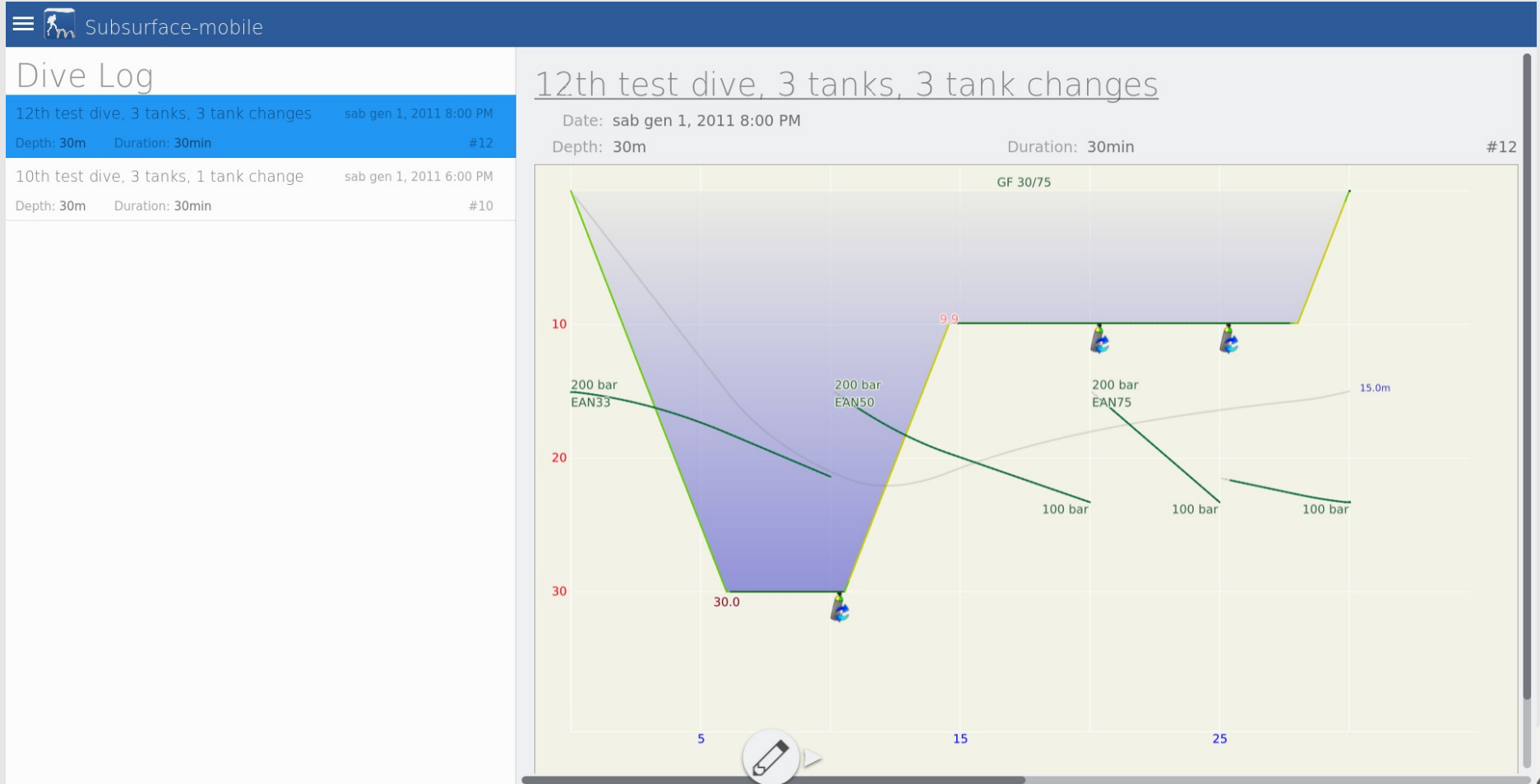


The home screen of the SubSurface app features a large background image of a vibrant coral reef with many small yellow fish. In the top left corner, there is a logo of a diver in a blue square with a white 'm' below it, and the word 'Subsurface' in white text. Below the image, there are two entries for dives: 'sab gen 1, 2011 8:00 PM #12' and 'sab gen 1, 2011 6:00 PM #10'. A white sidebar menu is on the left with the following items: 'Cloud credentials', 'Preferences', 'Manage dives', 'GPS', and 'Developer', each with a right-pointing chevron. At the bottom left, there is a checkbox labeled 'Run location service'.



The dive entry form in the SubSurface app has a dark blue header with a hamburger menu icon, a diver icon, and the text 'Subsurface-mobile'. Below the header, the title '12th test dive, 3 tanks, 3 tank' is displayed in a dark grey bar. The main form area is white and contains the following fields: 'Date: sab gen 1, 2011 8:00 PM', 'Location: 12th test dive, 3 tanks, 3 tank changes', 'Use current GPS location: ', 'Depth: 30m', 'Duration: 30min', 'Air Temp: 27.0°C', 'Water Temp: 26.0°C', 'Suit: ', 'Buddy: --', and 'Dive Master: --'. A 'Notes' section contains the text: 'Shows three pressure plots, each from 200 to 100bar and gaschange events at 10, 20 and 25 min mark. Last change is back to first tank.' A 'Save' button is located at the bottom right of the form. At the very bottom of the screen, there is a circular navigation button with a play icon and a slash through it.

SubSurface



Key takeaways

- Publicity (ehi, Linus is using Kirigami :p)
- It **must** be tier 1 (no dependencies besides Qt) to spark interest (makes some things more difficult for us)
- Many are going to be used *primarily* for Android applications, both by outside contributors *and* KDE people
- **But** encourages ports to Plasma Mobile



Future

- Smooth transition to QQC2:
 - pay attention to not have things that openly collide with them
 - Very useful stuff only in QQC1 (Action, ScrollView, StackView...) will still be used: probably depending from both for the time being
- Tier1 means not being able to use a lot of interesting kf5 stuff that have to be potentially reimplemented
- Packaging on mobile platforms can be tricky, Android is being fleshed out, iOS still at very early stages



Resources

- **HIG:**
https://community.kde.org/KDE_Visual_Design_Group/KirigamiHIG
- **API:**
<https://api.kde.org/playground-api/libs-apidocs/kirigami/html/index.html>
- **Gallery on Android:**
<https://play.google.com/store/apps/details?id=org.kde.kirigamigallery>
- **IRC: #plasma on freenode**
- **Mailing list: plasma-devel@kde.org**

