Cross-platform Development for Sailfish OS and Android: Architectural Patterns and “Dictionary Trainer” Application Case Study

Denis Laure, Andrey Vasilyev, Ilya Paramonov, Natalia Kasatkina

P.G. Demidov Yaroslavl State University
Yaroslavl, Russia

November 11, 2016
Introduction

- There are a lot of mobile platforms on the market
- End-user applications should support multiple platforms
- Business applications should provide native interface

Mobile applications development for multiple platforms

- Every popular mobile platform has unique look and feel
- Every platform provides native incompatible tooling
- You might need to use separate teams to develop such solution
Cross-platform development presumes the use of a single technology to develop application for several platforms:

- Decreases the time to deliver application to all platforms
- Decreases the number of technologies used
- Does not guarantee the bug-free environment
- Might not provide native look and feel of each platform

### Sailfish OS

- Qt framework is the way to develop native application
- Alien Dalvik project allows to run Android applications
Cross-platform Development using Qt

- Qt Quick is a modern way to develop user interface
- Qt provides native look for Android, iOS
- Qt allows rapid development with JavaScript and allows to fine-tune bottle necks with C++
- Native look-and-feel is not only in styling

Choosing an Architecture

- Qt does not force any architectural pattern, but provides tools
- The choice of the architecture affects how much code can be reused across different platforms
- We have considered following candidates: MVC, MVVM, Flux
Brief overview of MVC and MVVM

MVC pattern

Controller → View → Model

MVVM pattern

View → Data binding → ViewModel → Model

MVVM/MVC Issues

The growth of the application may lead to a growth of connections between views and models that are difficult to track and maintain.
Flux architecture was proposed by Facebook to support development of large web applications using React.

Flux is based on the unidirectional data flow.
Implementation of Flux

We have used QuickFlux library to implement Flux architecture.

- **AppDispatcher** implements **Dispatcher** component and delivers messages.
- **AppListener** listens for action notifications.
- **ActionCreator** eases creation of actions.

In our application:

- **View** is responsible for displaying data from stores and invoking actions.
- **Store** waits for actions, modifies itself and notifies / manages the **View**.
**Approach. Handling C++ Code**

- C++ handles heavy-duty, non-visual parts
- Qt framework API is already cross-platform

**Initialization issue**

- Sailfish OS uses `libsailfishapp`
- Android uses standard Qt Framework approach

You should include code for both platforms into the application

```cpp
#if defined(Q_OS_ANDROID)
   // Android-specific code
#else
   // Sailfish OS-specific code
#endif
```
Approach. Handling QML Code

QML code handles both the View and Store

**Store responsibilities**

- Manages application data storage
- Manages the navigation

Both platforms provide page stack, but in a different way

```javascript
if (Qt.platform.os === "linux") {
    // Sailfish OS-specific code
} else if (Qt.platform.os === "android") {
    // Android-specific code
}
```

Only *push*, *pop*, *replace* actions handlers are separated
Android and Sailfish OS user interface is quite different
- Android phones provide physical buttons
- Sailfish OS actively uses gestures

It is not possible to provide look and feel only by changing style

Technical differences
- Qt provides Android style for Qt Quick Controls
- Sailfish OS provides Silica to build
- Sailfish OS includes the Cover - extra view
Approach. Other files

The project consists not only from application code
- Icons and translations are shared between versions
- Platform-specific files take place in a common project – they are ignored by non-target platform

**Android-specific resources**
- Application manifest
- Gradle build scripts
- Keystore

**Sailfish OS-specific resources**
- Application launch files
- RPM build files (.yaml, and .spec)
Screen examples. Start screen

Dictionary trainer

- TEST ALL WORDS
- TEST LESS STUDIED WORDS
- WORDLIST

0%
Words learned 0 out of 53

Dictionary Trainer

- Test all words
- Test less studied words
- Wordlist

0%
Words learned 0 out of 53
Screen examples. Question screen

Puzzle

Misuse of a word due to the false folk etymology or wrong application of a term in a sense that does not belong to the word. This sort of misuse is mostly based on similarity in sound.

CHOOSE

All sorts of omission in a sentence.

CHOOSE

A statement that requires thinking over a confusing or difficult problem that needs to be solved.

CHOOSE

A reversal in the order of words in one of two parallel phrases

CHOOSE

SKIP QUESTION
Screen examples. Question screen

**Puzzle**

A statement that requires thinking over a confusing or difficult problem that needs to be solved. "How many blocks of stone did it take to complete one of the pyramids?"

---

**Enjambment ['ɪn 'dʒæməmənt]**

- Running on of one thought into the next line, couplet or stanza without breaking the syntactical pattern.

“A thing of beauty is a joy forever: Its loveliness increases; it will never Pass into nothingness but still will keep…”

---

Andrey Vasilyev

Cross-platform development for Sailfish OS…

FRUCT 19
The approach allows to quickly and easily develop applications for Android and Sailfish OS that look and behave natively.

Mostly all platform-specific code went to the View layer.

Even though Sailfish OS allows to launch Android applications without making any changes to them, such applications will not look and behave native on Sailfish OS and therefore will lead to a poor user experience.

Moreover, usage of Qt framework allows to compile created applications for other platforms (such as iOS). Therefore, the approach may be extended to support other platforms.