QtCon BR

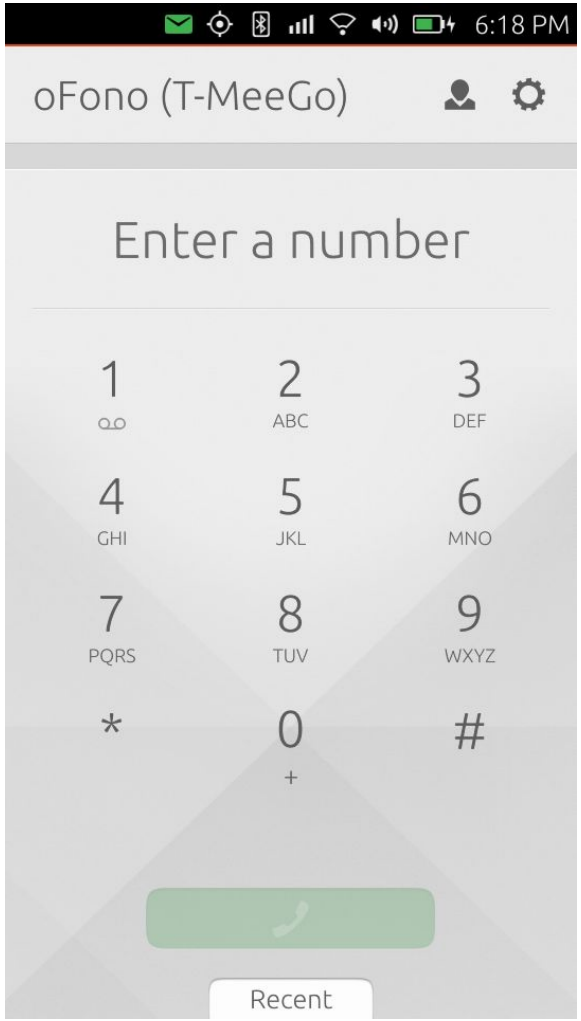
Ubuntu Phone: 10 Lições Sobre Desenvolvimento de Apps de Telefonia e Mensagens com Qt/QML

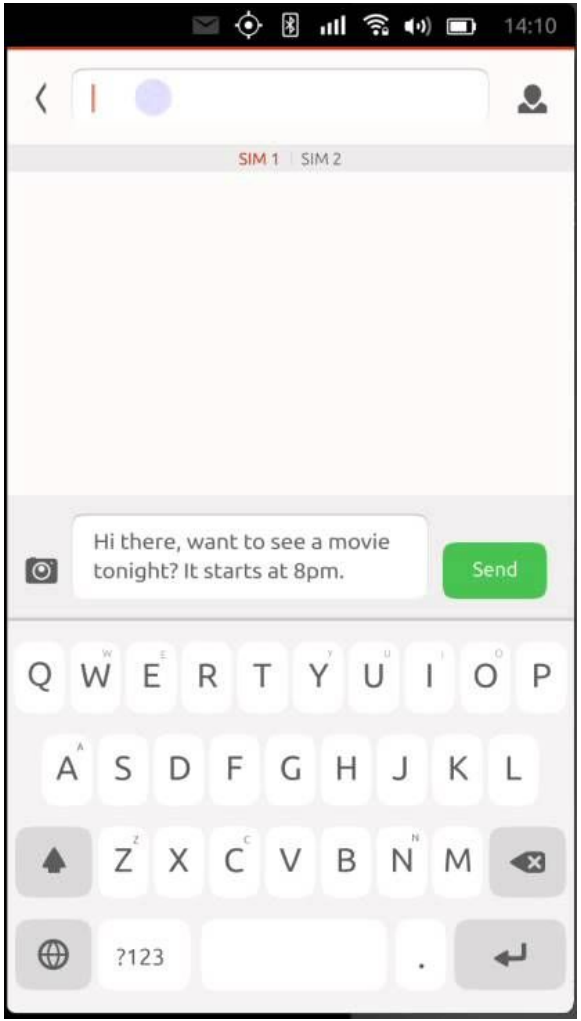
Tiago Salem – SUSE
contato@tiagosalem.com.br

Olá!

Tiago Salem

Ex-Mandriva, ex-Canonical, atualmente SUSE.
@tiagosh / tiagosalem.com.br


6:18 PM
oFono (T-MeeGo)
Enter a number
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#
Recent


14:10
SIM 1
SIM 2
Hi there, want to see a movie tonight? It starts at 8pm.
Send
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
?123

Qt

QtCon BR

Ubuntu Phone - Desafios

- Um único sistema rodar em diferentes form factors
- SDK para apps em desenvolvimento
- Ciclo de vida dos aplicativos é diferente em cada dispositivo.
- Novo formato de pacotes (click)
- Confinamento de apps exigia comunicação via dbus

Ubuntu Phone - Desafios

- Deep sleep dos telefones
- Interação com um dispositivo touch é diferente de um dispositivo com mouse e teclado.

1.

Life Cycle e Event Loop

Qt

SIGSTOP -> SIGCONT

Unity 8 envia sinais Unix para "congelar" e "descongelar" um processo.

Comportamentos:

Telefone

Programas quando estão em segundo plano recebem SIGSTOP

Computador

Programas em segundo plano não recebem SIGSTOP

DBus

Processo que faz chamada síncrona para um app em SIGSTOP (pausado) também fica travado.

Lição sobre Life Cycle:

- Evitar pausar o event loop do Qt;
- Efeitos colaterais dão trabalho para corrigir.
- Chamadas dbus síncronas para um processo pausado ficam travadas até o timeout.

2.

QTimer

“

Evite programação “orientada a QTimer”, especialmente se não há garantia que seu app estará rodando.

Qt

```
QTimer::singleShot(5000, this,
SLOT(fazerAlgo()));
```

Processo recebe SIGSTOP

fazerAlgo() ???

QTimer e Life Cycle

Postergar eventos?

```
QTimer::singleShot(0, this,
SLOT(fazerAlgo()));
```

Event loop ...

fazerAlgo()

```
QMetaObject::invokeMethod(this,
"fazerAlgo", Qt::QueuedConnection,
Q_ARG(QString, umaString));
```

Event loop ...

fazerAlgo(umaString)

QTimer e Event Loop

Lição sobre QTimer:

- Evitar pausar o event loop do Qt (de novo);
- Quando for usar QTimer, avaliar se é realmente necessário;
- QTimer não foi feito para postergar eventos no event loop. Use QMetaObject::invokeMethod()

Qt

3.

QtDBus

Use as Abstrações do Qt

Não reinvente a roda

Exemplo:

Variant

Use QVariant, mas cuidado para não exagerar. Serialização e desserialização dos dados pode custar caro em termos de processamento.

QDBusContext

Facilita muito para expor classes inteiras no dbus.

Evita precisar mexer em muito código para expor métodos novos no dbus.

Lição sobre QtDBus:

- Sempre que possível: definir interfaces em XML e use qdbusxml2cpp pra gerar o código.
  - Evita mexer no código toda vez que mudar a interface.

4.

API Síncrona
x
API Assíncrona

Cenário ideal:

Api toda assíncrona

Retorno das funções **não** acontecem de forma imediata.

Cenário Real:

Api Mista

Algumas funções podem exigir retorno imediato.

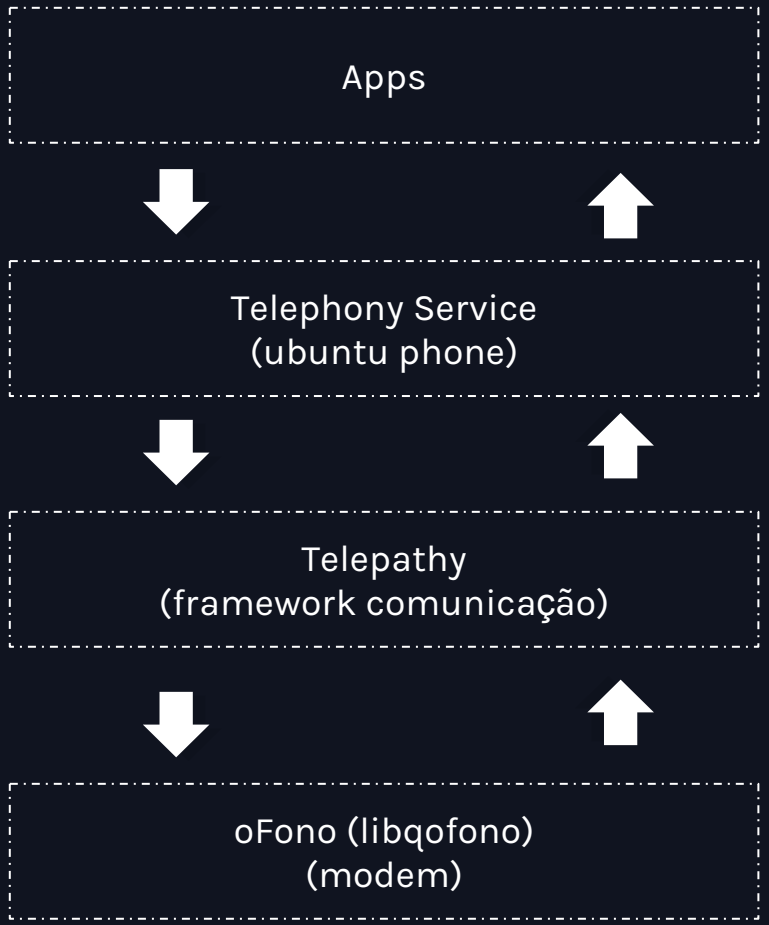
Telepathy

Exemplo:

Apps de telefonia utilizam Telepathy como backend.

Apesar do esforço para manter tudo assíncrono, foi preciso respeitar a API do telepathy e oFono (modem).

Fluxo de chamadas



Como interligar API síncrona e assíncrona:

```
QDBusPendingCall async = iface->asyncCall("RemoteMethod", value1, value2);
QDBusPendingCallWatcher *watcher = new QDBusPendingCallWatcher(async, this);

QObject::connect(watcher, SIGNAL(finished(QDBusPendingCallWatcher*)),
                 this, SLOT(callFinishedSlot(QDBusPendingCallWatcher*)));
```

http://doc.qt.io/qt-5/qdbuspendingcallwatcher.html

Lição sobre API:

- Quando possível evitar chamadas síncronas;
- Lembrar que quando há chamadas de API's diversas em cascata, nem sempre é possível chamadas assíncronas;

5.

QSqlDatabase

Comportamentos:

Backends

Nem todos se comportam de forma igual.

Ex: count() e size()

SQLite

Banco de dados é um arquivo.

Compartilhar entre vários processos pode ser problemático.

Lição sobre QSqlDatabase:

- Quando possível ter um processo isolado para acessar o banco de dados. (history-service)
- Backends tem implementações diferentes, portanto podem ter comportamentos diferentes.
- Implemente um mecanismo para facilitar atualização do schema do banco de dados.

Qt

6.

QThreads

“

Em muitos casos uma thread pode ser evitada agendando eventos no event loop.

Problema real:

Backend de áudio travando indicadores ao tocar som de nova mensagem.

"Solução":

Todos os eventos de som executados em uma thread separada.

Nota:

Threads em Qt são fáceis de criar...

Mas sincronizar muitas threads pode ainda ser complicado dependendo da complexidade do seu programa.

Prefira usar QMutexLocker em vez de QMutex.lock() e unlock()...

Evita que você esqueça de dar um "unlock" antes de sair dos métodos.

Lição sobre QThread's:

- Quando possível use o event loop da thread principal;
- Use QMutexLocker;
- Evite muitas threads. Complexidade exponencial.

Pois é! Threads...

Qt

7.

QtTest / QmlTest

“

Se seu código está com 100% de cobertura (coverage), algo está errado.

SOFTWARE EM DESENVOLVIMENTO

Código com muitos testes:

Chances de quebrar testes aumentam.

Nem sempre indica falha.

Tempo de desenvolvimento:

Se existe uma janela de tempo para se entregar um projeto, avaliar bem o tempo “gasto” escrevendo testes.

Escrever o teste/ arrumar o teste:

Dependendo do caso, escrever o teste é rápido, mas descobrir o que causa uma falha e consertá-la demora muito mais tempo.

SOFTWARE JÁ EM MANUTENÇÃO

Código com poucos testes:

Probabilidade de regressões não detectadas no desenvolvimento são altas.

Solução?

Equilíbrio

OUTROS DESAFIOS RELACIONADOS AOS TESTES:

1) Autopilot

Projeto interno e **novo** de automação de testes baseado no driver testability do Qt. Testes escritos em python.

2) Múltiplas plataformas

Mesmos testes rodando em tela de telefone, tablet e computador.

3) Múltiplas arquiteturas

Ubuntu era compilado em pelo menos 6 arquiteturas diferentes.

4) Múltiplos releases

Mesmo projeto precisava ser compilado em 3 releases diferentes do Ubuntu.

Se seu aplicativo é escrito em QML, escreva QmlTests.

- Fácil de manter
- Escrito na mesma linguagem do software principal.
- É possível reaproveitar pedaços/idéias do código original no teste.
- Consegue simular, de certa forma, um humano interagindo com o app.

Lição sobre Testes:

- Use um framework de testes que realmente tenha efetividade comprovada;
- Evite misturar tecnologias quando possível;
- Foque na arquitetura em que seu produto realmente irá rodar.
- Lembre-se que em alguns casos o tempo de descobrir e resolver uma regressão pode ser menor do que o esforço de manter muitos testes.
- Se seu código é Qt: Use **QtTest** e **QmlTest**.
- Evite testes baseados em timing:
    - Ex: Clica em um item .. espera 10 segundos.. faz o teste.

8.

Qml e Property Bindings

Bindings são excelentes! 

Qt

Bindings requerem cuidado! 

BINDINGS

Condicionar o valor de uma propriedade com o de outra.

```
Item {
    property var valor: 10
    property var metade: valor / 2
}
```

BINDINGS

Condicionar o valor de uma propriedade com o de outra.

```
Item {
    property var valor: 10
    property var metade: valor / 2

    height: {
        if (metade <= 5) {
            return 300;
        }
        return 500;
    }
}
```

BINDINGS

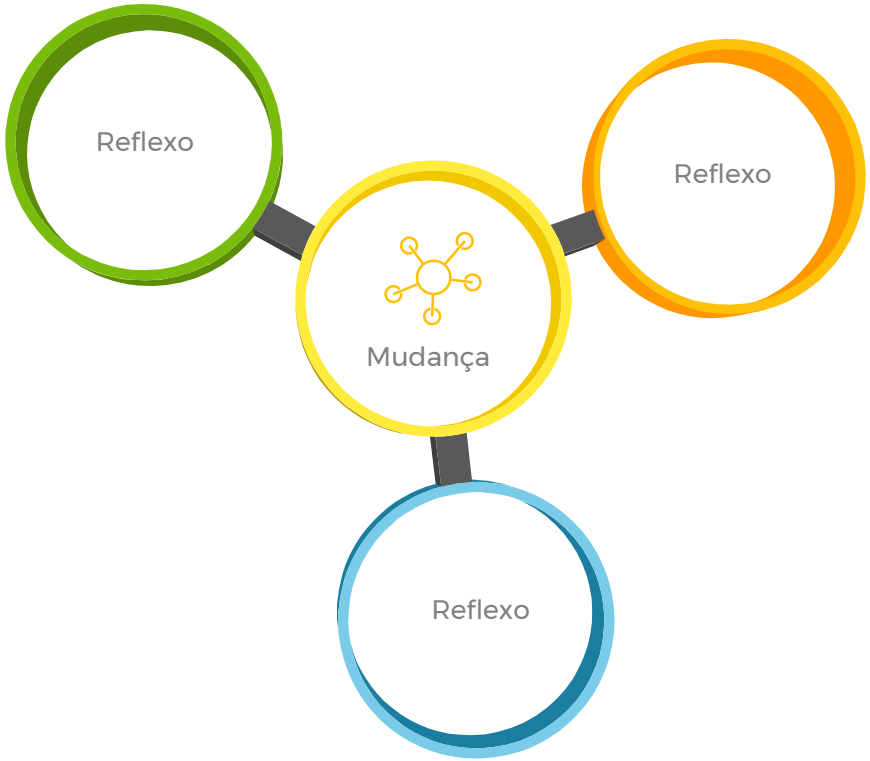
Condicionar o valor de uma propriedade com o de outra.

```
Item {
    property var valor: 10
    property var metade: valor / 2

    height: {
        if (metade <= 5) {
            return 300;
        }
        return 500;
    }
    width: height
}
```

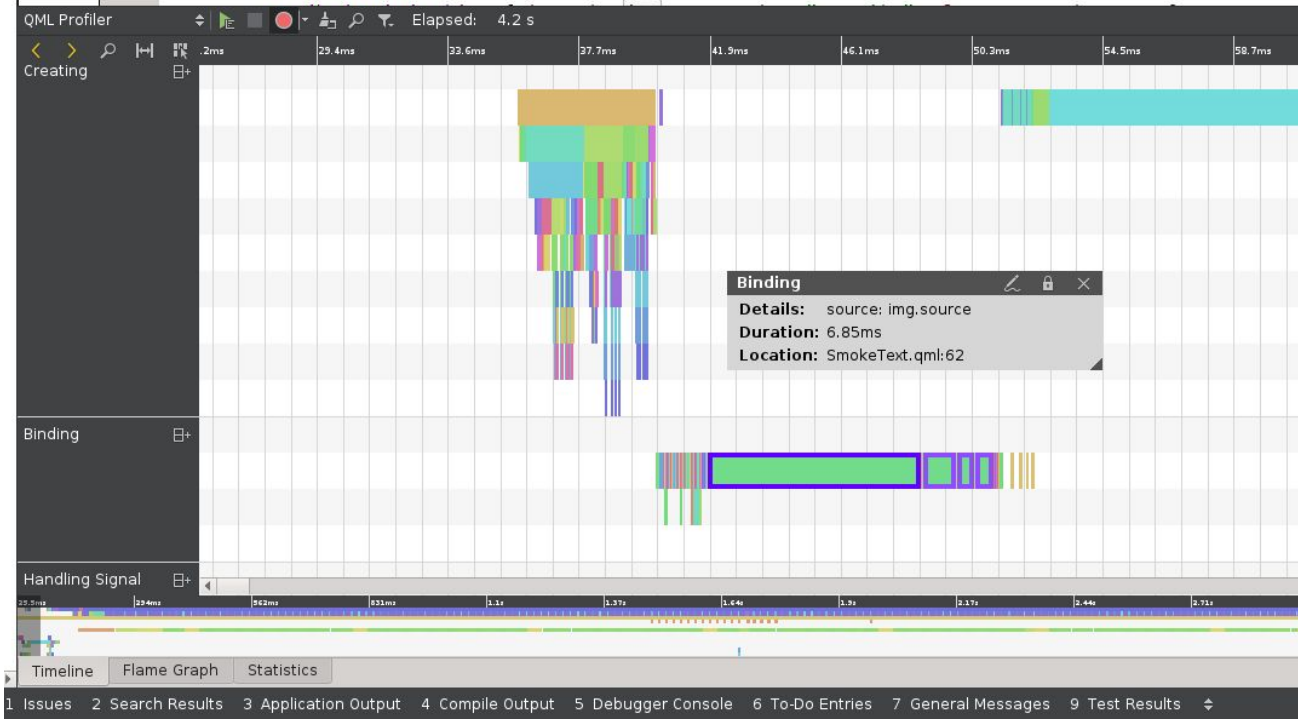
“Chuva” de reavaliações

EM OUTRAS PALAVRAS:



QML Profiler

```
69          }
70      }
~0.2%   71  SequentialAnimation {
72          id: anim
73          running: false
<0.1%   74  PauseAnimation { duration: 500}
<0.1%   75  ParallelAnimation {
```



QML Profiler

QML Profiler | Elapsed: 5.5 s

Location	Type	Time in Percent	Total Time	Calls	Time per Call	Longest Time	Shortest T
samegame.qml:74	Signal	0.79 %	14.639 ms	1	14.639 ms	14.639 ms	14.639 ms
samegame.qml:142	Signal	12.22 %	227.645 ms	1	227.645 ms	227.645 ms	227.645 ...
Button.qml:70	Signal	12.30 %	229.128 ms	1	229.128 ms	229.128 ms	229.128 ...
BoomBlock.qml:61	Binding	1.40 %	26.127 ms	308	84.827 µs	2.921 ms	48.028 µs
BoomBlock.qml:85	Binding	1.59 %	29.563 ms	308	95.985 µs	4.997 ms	57.560 µs
samegame.qml:155	Binding	0.01 %	209.708 µs	1	209.708 µs	209.708 µs	209.708 µs
BoomBlock.qml:103	Binding	0.06 %	1.075 ms	200	5.377 µs	31.529 µs	3.666 µs
BoomBlock.qml:98	Binding	0.13 %	2.336 ms	392	5.959 µs	49.861 µs	2.933 µs
Button.qml:60	Binding	0.03 %	537.835 µs	2	268.918 µs	338.759 µs	199.076 µs
BoomBlock.qml:71	Create	0.35 %	6.529 ms	192	34.006 µs	115.487 µs	29.330 µs
BoomBlock.qml:51	Create	0.43 %	7.967 ms	192	41.495 µs	108.520 µs	35.929 µs
BoomBlock.qml:55	Create	0.42 %	7.792 ms	192	40.584 µs	104.855 µs	34.829 µs
BoomBlock.qml:102	Create	0.02 %	376.888 µs	8	47.111 µs	84.323 µs	40.695 µs
BoomBlock.qml:97	Create	0.35 %	6.467 ms	192	33.683 µs	196.510 µs	28.230 µs

Events | Timeline | Callees | Callers

Lição sobre Bindings:

- É uma das melhores ferramentas do QML;
- Cuidado com os binding loops;
- Cuidado com a “chuva de reavaliações”;
- Em caso de problemas de performance: Qml Profiler;

Qt

Qt

9.

Ubuntu UI toolkit

OBJETIVOS:

- Manter consistência visual entre apps.
- Facilitar o desenvolvimento de apps.
- Permitir desenvolver apps convergentes.

TOOLKIT QML EM DESENVOLVIMENTO

Constante mudança de API:

Aplicativos que paravam de funcionar de um dia para o outro.

Constante mudança de design/foco:

Toolkit precisava se adaptar com as mudanças de design.

Impedir compartilhar código com outros sistemas:

Jolla acabou desenvolvendo seu próprio toolkit, também em QML.

Lição sobre Toolkits:

- Quando puder, utilize um UI Toolkit já estabilizado e consolidado;
- Desenvolver apps com Toolkits em constante mudança é como construir uma casa com o chão tremendo;

10.

Outras lições

Outras Lições:

- Evitar mexer em diversas áreas ao mesmo tempo:
  - Novo UI toolkit: constante mudança;
  - Novo modelo de Pacotes (deb -> click -> snap);
  - Novo conceito: Convergência;
  - Novo framework de testes (autopilot);
  - Novo servidor gráfico: MIR.

Qt

QtCon BR

OBRIGADO!

Perguntas?

tiagosalem.com.br ou @tiagosh

Qt